NORTHEASTERN UNIVERSITY
GRADUATE SCHOOL OF COMPUTER SCIENCE
Ph.D. THESIS APPROVAL FORM

THESIS TITLE: Infinite Words as Determined by Languages of their Prefixes

AUTHOR: Tim Smith

Ph.D. Thesis Approved to complete all degree requirements for the Ph.D. Degree in Computer Science.

Rajiv (RAJKOHAN RAJARAHAN) 4/13/15
Thesis Advisor/Date

Jel Alac 4/16/15
Date

Amal K. Reaser (Arnold L. Rosenberg) 4/16/15
Thesis Reader/Date

J. I. Kangal 4/15/15
Date

Ravi Sooderam (Ravi Sundaram) 4/13/15
Thesis Reader/Date

GRADUATE SCHOOL APPROVAL:

R. V. Churi 4/21/15
Director, Graduate School

COPY RECEIVED IN GRADUATE SCHOOL OFFICE:

Recipient’s Signature 4/21/2015
Date

Distribution: Once completed, this form should be scanned and attached to the front of the electronic dissertation document (page 1). An electronic version of the document can then be uploaded to the Northeastern University-UMI website.
INFINITE WORDS AS DETERMINED BY LANGUAGES OF THEIR PREFIXES

TIM SMITH

A dissertation submitted in partial fulfillment of the requirements for the degree of Ph.D. in computer science

College of Computer and Information Science
Northeastern University
Boston, MA, USA

April 2015
ABSTRACT

We explore a notion of complexity for infinite words relating them to languages of their prefixes. An infinite language $L$ determines an infinite word $\alpha$ if every string in $L$ is a prefix of $\alpha$. If $L$ is regular, it is known that $\alpha$ must be ultimately periodic; conversely, every ultimately periodic word is determined by some regular language. In this dissertation, we investigate other classes of languages and infinite words to see what connections can be made among them within this framework. We make particular use of pumping lemmas as a tool for studying these classes and their relationship to infinite words.

- First, we investigate infinite words determined by various types of automata. We consider finite automata, pushdown automata, a generalization of pushdown automata called stack automata, and multihead finite automata, and relate these classes to ultimately periodic words and to a class of infinite words which we call multilinear.

- Second, we investigate infinite words determined by the parallel rewriting systems known as L systems. We show that certain infinite L systems necessarily have infinite subsets in other L systems, and use these relationships to categorize the infinite words determined by a hierarchy of these systems, showing that it collapses to just three distinct classes of infinite words.

- Third, we investigate infinite words determined by indexed grammars, a generalization of context-free grammars in which nonterminals are augmented with stacks which can be pushed, popped, and copied to other nonterminals at the derivation proceeds. We prove a new pumping lemma for the languages generated by these grammars and use it to show that they determine exactly the class of morphic words.

- Fourth, we extend our investigation of infinite words from determination to prediction. We formulate the notion of a prediction language which masters, or learns in the limit, some set of infinite words. Using this notion we explore the predictive capabilities of several types of automata, relating them to periodic and multilinear words. We also consider the number of guesses required to master infinite words of particular description sizes.
## CONTENTS

1 **INTRODUCTION** 1  
  1.1 Background and related work 2  
  1.2 Our contributions 3  
  1.3 Proof techniques 4  
  1.4 Organization of the dissertation 4  

2 **PRELIMINARIES** 6  
  2.1 Words and morphisms 6  
  2.2 Prefix languages 7  

3 **INFINITE WORDS DETERMINED BY AUTOMATA** 9  
  3.1 Introduction 9  
    3.1.1 Related work 10  
    3.1.2 Outline of chapter 12  
  3.2 Preliminaries 12  
  3.3 Ultimately periodic words 13  
  3.4 Multilinear words 15  
  3.5 Multihead finite automata 17  
  3.6 Conclusion 22  

4 **INFINITE WORDS DETERMINED BY L SYSTEMS** 23  
  4.1 Introduction 23  
    4.1.1 Proof techniques 24  
    4.1.2 Related work 24  
    4.1.3 Outline of chapter 26  
  4.2 Preliminaries 26  
  4.3 Pumping lemma for T0L systems 28  
  4.4 Infinite subsets of L systems 31  
  4.5 Categorizations 32  
    4.5.1 PD0L classes 32  
    4.5.2 D0L classes 33  
    4.5.3 CD0L classes 33  
  4.6 $\omega$(PD0L), $\omega$(D0L), and $\omega$(CD0L) 34  
    4.6.1 Separating the classes 35  
    4.6.2 Characterizing the words in each class 36  
  4.7 Conclusion 37  

5 **INFINITE WORDS DETERMINED BY Indexed LANGUAGES** 38  
  5.1 Introduction 38  
    5.1.1 Proof techniques 39  
    5.1.2 Related work 40  
    5.1.3 Outline of chapter 40  
  5.2 Preliminaries 40  

---

iii
CONTENTS

5.2.1 Morphic words and L systems 40
5.2.2 Indexed languages 41
5.3 Linear indexed languages 42
5.4 Pumping lemma for indexed languages 43
  5.4.1 Derivation trees 43
  5.4.2 Lemma about derivation trees 44
  5.4.3 Pumping lemma for indexed languages 47
5.5 Applications 53
  5.5.1 Frequent and rare symbols 53
  5.5.2 CD0L and morphic words 54
5.6 Conclusion 55

6 PREDICTION OF INFINITE WORDS 56
6.1 Introduction 56
  6.1.1 Framework 57
  6.1.2 Related work 59
  6.1.3 Outline of chapter 60
6.2 Prediction languages for infinite words 61
  6.2.1 Prediction of purely periodic words 61
  6.2.2 Prediction of ultimately periodic words 62
  6.2.3 Prediction of multilinear words 63
6.3 Number of guesses 71
  6.3.1 Maximum agreement 73
  6.3.2 Agreement of ultimately periodic words 73
  6.3.3 Agreement of multilinear words 74
  6.3.4 Agreement of D0L words 79
  6.3.5 Agreement of morphic words 80
6.4 Conclusion 80

7 CONCLUSION 81
  7.1 Summary of results 81
  7.2 Further work 82

ACKNOWLEDGMENTS 84

BIBLIOGRAPHY 85
INTRODUCTION

An infinite word is an infinite concatenation of symbols drawn from a finite alphabet. Infinite words can be used to represent a variety of objects and processes in mathematics, computer science, and beyond. Real numbers, symbolic time series, and other data streams can be viewed as infinite words, and their descriptional, computational, and combinatorial properties can be investigated in this setting.

In this dissertation, we explore a notion of complexity for infinite words relating them to languages of their prefixes. Consider two infinite words $\alpha$ and $\beta$ over the alphabet \{a, b\}:

$$\alpha = \text{ababab}\ldots$$
$$\beta = \text{abaabaaab}\ldots$$

The infinite word $\alpha$ simply repeats the string ab, while in $\beta$ the number of as between the bs is increasing. In some sense, $\beta$ seems to be “more complex” than $\alpha$. How can we formalize this notion? There are various ways, but the one which we will explore in this dissertation we call “prefix language complexity”. A prefix language is a language $L$ for which there is an infinite word $\gamma$ such that every string in $L$ is a prefix of $\gamma$. If $L$ is infinite, we say that $L$ determines $\gamma$. For example, take the language

$$L = \{ab, \text{abab, ababab,}\ldots\}$$

Every string in $L$ is a prefix of the infinite word $\alpha$, so $L$ is a prefix language which determines $\alpha$. Where $C$ is a class of languages, we denote by $\omega(C)$ the class of infinite words determined by the prefix languages in $C$. Much as the computational complexity of a decision problem is characterized by the complexity classes to which it belongs, the prefix language complexity of an infinite word is characterized by the language classes in which it is determined. In our example, since the language $L$ can be denoted by the regular expression $(ab)^+$, $\alpha$ is in $\omega(\text{REG})$, the class of infinite words determined by regular languages. We will see later that no regular language can determine $\beta$.

The seed of this approach appears in a paper by Ronald Book [Boo77], who formulated a “prefix property” intended to allow languages to “approximate” infinite words, and showed that for certain classes of languages, if a language in the class has the prefix property, then it is regular. We take this idea beyond the regular prefix languages considered by Book, investigating a variety of more powerful language classes to see what infinite words they determine.
1.1 BACKGROUND AND RELATED WORK

The model used in this dissertation, in which infinite words are determined by languages of their prefixes, builds on Book’s 1977 paper [Boo77]. A follow-up by Latteux [Lat78] 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 [Ber89] for a survey of results on infinite words whose coprefix language is context-free.

There are several other notions of complexity for infinite words. One is their computational complexity in terms of time and space classes. Research in this area has typically focused on sequence generators, devices which run indefinitely and output an infinite word piece by piece. This was the model of the seminal 1965 paper of Hartmanis and Stearns [HS65], as well as a 1970 follow-up by Fischer, Meyer, and Rosenberg [FMR70], and several later papers beginning with Hromkovič, Karhumäki, and Lepistö [HKL94], who investigated the computational complexity of infinite words generated by different kinds of iterative devices, followed by [HK97] and [DM03].

The question of what mechanisms or devices suffice to generate particular infinite words (sometimes called the “descriptive” or “descriptive” complexity of infinite words [KLO]) has also been studied, particularly in connection with the iterative processes underlying L systems. Pansiot [Pan85] considered various classes of infinite words obtained by iterated mappings. Culik & Karhumäki [CK94] considered iterative devices generating infinite words. Culik & Salomaa [CS82] investigated infinite words associated with D0L and DT0L systems; their notion of “strong uniform convergence” is equivalent to our notion of a language “determining” an infinite word.

In another model, an automaton is associated with an infinite word $\alpha$ if when given a number $n$ as input in some base $k$, it outputs the $n$th symbol of $\alpha$. In 1972 Alan Cobham used this approach to associate finite automata with uniform tag sequences [Cob72], leading to a literature on these “automatic sequences” [AS03].

Other types of complexity applied to infinite words in the literature include subword complexity and Kolmogorov complexity. The “subword complexity” (or “factor complexity”) [CN10] of an infinite word $\alpha$ is characterized by a function $f(n)$ counting the number of distinct length-$n$ contiguous subwords (“factors”) of $\alpha$. The “Kolmogorov complexity” [LV08] of an infinite word $\alpha$ is the length of the shortest description of $\alpha$ in some fixed universal description language $U$. In “prefix-free Kolmogorov complexity” (sometimes just “prefix complexity”), no description in $U$ is a proper prefix of another description. In “resource-bounded Kolmogorov complexity”, the power of $U$ is restricted with respect to some resource, e.g. time or space. Whereas Kolmogorov complexity measures the length of the shortest descrip-
tion of \( \alpha \) (perhaps within some resource bounds) in a universal language, our notion of complexity is concerned with prefix languages which can individually determine \( \alpha \).

In this dissertation, we consider only languages containing finite strings. There are also \( \omega \)-languages [Sta97], languages containing infinite words. Associated with these are \( \omega \)-automata or stream automata (e.g. Büchi automata). Each \( \omega \)-automaton takes infinite words as input and recognizes an \( \omega \)-language, or set of infinite words. By contrast, in our framework, an automaton takes finite strings as input and recognizes a language of finite strings, which then determines at most one infinite word.

1.2 OUR CONTRIBUTIONS

We give a number of results aimed at building up a classification of infinite words with respect to which classes of languages can determine them.

- **Automata.** We first consider language classes defined by various kinds of automata [HU79, WW86]. Finite and pushdown automata, which recognize respectively the regular and context-free languages, determine only ultimately periodic words. We show that the same is true for 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 further show that the class of multilinear words is properly contained in the class of infinite words determined by one-way multihead deterministic finite automata (1:multi-DFA).

- **L systems.** Next, we consider language classes defined by L systems, parallel rewriting systems which were originally introduced to model growth in simple multicellular organisms [RS80, KRS97]. L systems can be restricted and generalized in various ways, yielding a hierarchy of language classes. We give a pumping lemma for \( \omega \)L systems, a particular type of L systems. With the help of this lemma we categorize the infinite words determined by a variety of L systems, showing that the whole hierarchy collapses to just three distinct classes of infinite words: \( \omega(\text{PD}0\text{L}) \), \( \omega(\text{D}0\text{L}) \), and \( \omega(\text{CD}0\text{L}) \).

- **Indexed grammars.** We then consider the class of languages defined by the indexed grammars of Alfred Aho [Aho68]. This is a broad language class, containing all of the L systems and automata classes previously mentioned. We show that the infinite words determined by indexed languages are exactly the morphic words, infinite words generated by iterating a morphism
under a coding. To obtain this result, we prove a new pumping lemma for the indexed languages, which may be of independent interest.

- **Prediction.** Finally, we extend our investigation of infinite words from determination to prediction. To do so, we formulate the notion of a prediction language. Whereas a prefix language determines a single infinite word, a prediction language masters, or learns in the limit, a set of infinite words. We study the predictive capabilities of finite automata, deterministic nonerasing stack automata, and multihead finite automata, relating them to periodic and multilinear words. We then consider the number of guesses required to master infinite words of particular description sizes, and in the case of multilinear words we relate this notion to open problems in the study of word equations. We hope that this framework may have eventual applications in prediction of symbolic time series or other data streams.

### 1.3 Proof Techniques

We will make particular use of pumping lemmas as a tool for studying language classes and their relationship to infinite words. The pumping lemmas for regular and context-free languages are well-known. Pumping lemmas also exist for broader classes of languages [Ogd69, Hay73, Gil96, Rab12]; these pumping lemmas are more complex and more difficult to prove than those for the regular and context-free languages. In this work, in addition to using existing pumping lemmas, we prove new pumping lemmas to characterize the infinite words determined by certain language classes.

The connection of pumping lemmas to infinite words is through prefix languages. In some cases, a pumping lemma implies that every infinite prefix language in the pumping language class contains a succession of strings with some common structure; this structure can then be used to characterize the infinite word determined by the prefix language. In other cases, a pumping lemma can be used to show that every infinite language in the pumping language class has an infinite subset in some smaller language class. Applied to prefix languages, this limits the infinite words of the larger class to the infinite words of the smaller class.

### 1.4 Organization of the Dissertation

The dissertation is organized as follows. Chapter 2 gives preliminary definitions and propositions. Chapter 3 investigates infinite words determined by various types of automata. Chapter 4 investigates infinite words determined by L systems. Chapter 5 investigates infinite words
determined by indexed grammars. Chapter 6 deals with prediction of infinite words. Chapter 7 gives our conclusions.

Parts of this dissertation are based on the author’s conference papers: Chapter 3 is based on [Smi13c], Chapter 4 is based on [Smi13b] and also includes material from [Smi13a], and Chapter 5 is based on [Smi14a].
In this chapter, we give preliminary definitions and propositions.

Where \( X \) and \( Y \) are sets, \( X \subseteq Y \) means that \( X \) is a subset of \( Y \) and \( X \subset Y \) means that \( X \) is a proper subset of \( Y \). We denote the cardinality of \( X \) by \( |X| \). The power set (set of all subsets) of \( X \) is denoted by \( \mathcal{P}(X) \).

For a list or tuple \( v \), \( v[i] \) denotes the \( i \)th element of \( v \); indexing starts at 1.

## 2.1 Words and Morphisms

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 \( A^\omega \). We call finite words **strings** and infinite words **streams** or **\( \omega \)-words**. The length of \( x \) is denoted by \( |x| \). We denote the empty string by \( \lambda \). 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 \( S' \). The \( i \)th symbol of \( S \) is denoted by \( S[i] \); indexing starts at 1. A **power** of a string \( x \) is a string of the form \( x^i \) for some \( i \geq 1 \). For a non-empty string \( x \), \( x^\omega \) denotes the infinite word \( xxx \cdots \). Such a word is called **purely periodic**. An infinite word of the form \( xy^\omega \), where \( x \) and \( y \) are strings and \( y \neq \lambda \), is called **ultimately periodic**.

A **morphism** on an alphabet \( A \) is a map \( h \) from \( A^* \) to \( A^* \) such that for all \( x,y \in A^* \), \( h(xy) = h(x)h(y) \). Notice that \( h(\lambda) = \lambda \). The morphism \( h \) is **nongerasing** if for all \( a \in A \), \( h(a) \neq \lambda \). The morphism \( h \) is a **coding** if for all \( a \in A \), \( |h(a)| = 1 \). The morphism \( h \) is a **weak coding** if for all \( a \in A \), \( |h(a)| \leq 1 \). The morphism \( h \) is an **identity** if for all \( a \in A \), \( h(a) = a \). For a language \( L \), we define \( h(L) = \{ h(x) \mid x \in L \} \).

A string \( x \in A^* \) is **mortal** (for \( h \)) if there is an \( m \geq 0 \) such that \( h^m(x) = \lambda \). The morphism \( h \) is **prolongable** on a symbol \( a \) if \( h(a) = ax \) for some \( x \in A^* \), and \( x \) is not mortal. In this case \( h^\omega(a) \) denotes the infinite word \( a \cdot h(x) \cdot h^2(x) \cdots \). An infinite word \( \alpha \) is **pure morphic** if there is a morphism \( h \) and symbol \( a \) such that \( h \) is prolongable on \( a \) and \( \alpha = h^\omega(a) \). An infinite word \( \alpha \) is **morphic** if there is a morphism \( h \), coding \( e \), and symbol \( a \) such that \( h \) is prolongable on \( a \) and \( \alpha = e(h^\omega(a)) \). For example, let:

\[
\begin{align*}
h(s) &= \text{sbba} & e(s) &= a \\
h(a) &= aa & e(a) &= a \\
h(b) &= b & e(b) &= b
\end{align*}
\]
Then $e(h^ω(s)) = a^1ba^2ba^4ba^8ba^{16}b\cdots$ is a morphic word. Every purely periodic word is pure morphic, and every ultimately periodic word is morphic. For results on morphic words, see [AS03].

A finite substitution on $A$ is a map $σ$ from $A^*$ to $P(A^*)$ such that (1) for all $x ∈ A^*$, $σ(x)$ is finite and non-empty, and (2) for all $x, y ∈ A^*$, $σ(xy) = \{x'y' | x' is in σ(x) and y' is in σ(y)\}$. Notice that $σ(λ) = \{λ\}$. $σ$ is nonerasing if for all $α ∈ A$, $σ(α) ∉ λ$. For a language $L$, we define $σ(L) = \{x' | x' is in σ(x) for some x ∈ L\}$.

2.2 Prefix Languages

A prefix language is a language $L$ such that for all $x, y ∈ L$, $x$ 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 $x ∈ L$ is a prefix of $α$. For example, the infinite prefix language $\{λ, ab, abab, ababab, . . .\}$ determines the infinite word $(ab)ω$. The following propositions are basic consequences of the definitions.

**Proposition 2.1.** A language determines at most one infinite word.

**Proof.** Suppose a language $L$ determines distinct infinite words $α$ and $β$. Then $α$ and $β$ differ at some position $i$. Take any $x ∈ L$ such that $|x| ≥ i$. Then since $L$ determines $α$ and $β$, $x$ is a prefix of $α$ and $β$. But then $x[i] = α[i] = β[i]$, a contradiction. So a language determines at most one infinite word. □

**Proposition 2.2.** A language $L$ determines an infinite word iff $L$ is an infinite prefix language.

**Proof.** $⇒$ Suppose $L$ determines an infinite word $α$. Then $L$ is infinite and every $x ∈ L$ is a prefix of $α$. So for all $x, y ∈ L$, $x$ and $y$ are prefixes of $α$; hence $x$ is a prefix of $y$ or $y$ is a prefix of $x$. Hence $L$ is an infinite prefix language.

$⇐$ Suppose $L$ is an infinite prefix language. For each $i ≥ 1$, let $x_i$ be any string in $L$ such that $|x_i| ≥ i$. Let $α$ be the infinite word $\prod_{i≥1} x_i[i]$. Now suppose some $x ∈ L$ is not a prefix of $α$. Then $x$ differs from $α$ at some position $i$. Then $x$ differs from $x_i$ at $i$. But then neither $x$ nor $x_i$ is a prefix of the other, a contradiction. So $L$ determines $α$. □

Notice that while a language determines at most one infinite word, an infinite word may be determined by more than one language. In particular, we will make use of the following fact.

**Proposition 2.3.** If a language $L$ determines an infinite word $α$ and $L'$ is an infinite subset of $L$, then $L'$ determines $α$.

**Proof.** If $L$ determines $α$, then every $x ∈ L$ is a prefix of $α$, so if $L'$ is an infinite subset of $L$, then every $x ∈ L'$ is a prefix of $α$, hence $L'$ determines $α$. □
For a language class $C$, let

$$\omega(C) = \{ \alpha \mid \alpha \text{ is an infinite word and some } L \in C \text{ determines } \alpha \}.$$ 

For example, $\omega(\text{REG})$ is the set of infinite words determined by regular languages. On several occasions in this dissertation, we will encounter language classes $C_1, C_2$ such that $C_2 \subset C_1$ and $\omega(C_1) = \omega(C_2)$. The class $C_2$ is a proper subset of the class $C_1$, but they determine the same set of infinite words. Notice that it does not follow that every $C_1$ prefix language is a $C_2$ prefix language, but only that every infinite word determined by a $C_1$ prefix language can be determined by a $C_2$ prefix language.
In this chapter, we characterize the infinite words determined by one-way stack automata. Recall that an infinite language \( L \) determines an infinite word \( \alpha \) if every string in \( L \) is a prefix of \( \alpha \). If \( L \) is regular or context-free, it is known that \( \alpha \) 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.

3.1 Introduction

In this chapter, 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 [GGH67b, GGH67a]; see [HU79] and [WW86] 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 [Boo77] in using the concept of prefix languages. Recall that 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 languages and automata can determine them.

We begin with the ultimately periodic words, those of the form \( x_1y_1y_2\cdots \), where \( x \) and \( y \) are strings and \( y \) is not empty. As observed in [Boo77], every infinite regular prefix language determines an ulti-
mately periodic word. Since the converse is also true, an infinite word is in \( \omega(\text{REG}) \) iff it is ultimately periodic. It is further known that \( \omega(\text{CFL}) \), the class of infinite words determined by context-free languages, equals \( \omega(\text{REG}) \). This follows from a result of Book [Boo77], 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 checking stack automata (1-DCSA). We extend this result to the nondeterministic case (1-NCSA) using a pumping lemma. 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 string \( q \), followed by strings \( r_1, \ldots, r_m \) 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-DNEA (deterministic nonerasing stack automata). We find with the help of a pumping lemma due to Ogden [Ogd69] that each of these classes determines exactly the multilinear infinite words. That is, \( \omega(1-\text{NSA}) = \omega(1-\text{DSA}) = \omega(1-\text{NNESA}) = \omega(1-\text{DNEA}) \).

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 \( \omega(1:\text{multi-DFA}) \) which is not multilinear. The problem of further characterizing the class of infinite words determined by 1:multi-DFA remains open.

Our results appear in Figures 3.1 and 3.2.

3.1.1 Related work

The present chapter is drawn from the author’s conference paper [Smi13c].

The class of infinite words which we call multilinear appears in [EHK11], where it is not given a name. There, infinite words are compared in terms of reducibility by finite-state transducers (FSTs). This comparison gives rise to a hierarchy of “degrees”, or equivalence classes under this reduction. The class of proper multilinear words (multilinear infinite words which are not ultimately periodic) is shown to constitute a degree, and furthermore is shown to be “prime”, in the sense that every FST reduction of an infinite word in that degree yields either an infinite word in the same degree, or else an ultimately periodic or finite word. Whether there exist any
other prime degrees than the proper multilinear words is left as an open question.

In obtaining their results, \cite{EHK11} establish a proposition about “breaking points of periodicity” in multilinear words (their Proposition 32), of which we were unaware at the time of publication of \cite{Smi13c}. Their proposition resembles our Theorem 3.15, but with a weaker condition: for adjacent terms $r_i$ and $r_{i+1}$ where $r_i$ is a growth term (i.e., $a_i > 0$), their proposition guarantees that $r_{i+1}$ is not a prefix of $r_i^\omega$, whereas our theorem guarantees that $r_i$ and $r_{i+1}$ do not even start with the same symbol. Their proposition leaves a need for bounded “look-ahead” to detect the transition from $r_i$ to $r_{i+1}$ (see their Remark 33), which is not required with our theorem.

Certain integer sequences which arise in the study of quasiperiodic continued fractions \cite{Per13} have a structure resembling that of multilinear infinite words. In \cite{Davo7} these sequences are expressed in the framework of formal languages and words.
We assume a familiarity with finite automata and pushdown 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 [HU79] and [WW86] for formal definitions and results. We use the abbreviations shown in Table 3.1. From the definitions, we have

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

A multihead finite automaton is a finite automaton with one or more input heads. Here we are concerned only with $1:\text{multi-DFA},$ the class of one-way multihead deterministic finite automata. This

<table>
<thead>
<tr>
<th>Language Class</th>
<th>Stack Automata</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-NSA</td>
<td>one-way nondeterministic stack automata</td>
</tr>
<tr>
<td>1-DSA</td>
<td>one-way deterministic stack automata</td>
</tr>
<tr>
<td>1-NNESA</td>
<td>one-way nondeterministic nonerasing stack automata</td>
</tr>
<tr>
<td>1-DNESA</td>
<td>one-way deterministic nonerasing stack automata</td>
</tr>
<tr>
<td>1-NCSA</td>
<td>one-way nondeterministic checking stack automata</td>
</tr>
<tr>
<td>1-DCSA</td>
<td>one-way deterministic checking stack automata</td>
</tr>
</tbody>
</table>

Table 3.1: Types of stack automata considered in this dissertation.
3.3 Ultimately periodic words

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

**Theorem 3.1.** Every ultimately periodic word $\alpha$ is in $\omega(\text{REG})$.

**Proof.** The infinite word $\alpha$ has the form $xy^\omega$ for some strings $x$ and $y$ where $y \neq \lambda$. Then the regular language $xy^*$ determines $\alpha$. So $\alpha$ is in $\omega(\text{REG})$. \qed

As observed by Book [Boo77], 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 3.2** (Book). Suppose $\alpha$ is in $\omega(\text{CFL})$. Then $\alpha$ is ultimately periodic.

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

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 [Boo77] 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 $\omega(1-NCSA) \neq \omega(\text{REG})$.) We employ a weak pumping lemma for 1-NCSA which we obtain using results from Greibach [Gre78]. For a stronger pumping lemma (i.e., one which
is universal rather than existential) recently proved for this class of languages, see [Smi14b].

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

\[ s = x_1 y_1 x_2 y_2 \cdots x_k y_k x_{k+1}, \]

\[ |y_1 \cdots y_k| \geq 1, \text{ and} \]

\[ \text{for all } i \geq 0, x_1 y_1^i x_2 y_2^i \cdots x_k y_k^i x_{k+1} \text{ is in } L. \]

\( L \) is \textbf{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 employ results from Greibach [Gre78] 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 3.3.** Suppose \( L \) is in 1-NCSA. Then \( L \) is weakly \( k \)-iterative for some \( k \geq 1 \).

**Proof.** A checking stack automaton \( M \) is \textbf{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 \( k\text{-VISIT(REGL)} \) of [Gre78]. Then by Lemma 4.22 of [Gre78], \( L \) is weakly \( k \)-iterative. So say there is no such \( M \). Then \( L \) is not in the class \( \text{FINITEVISIT(REGL)} \) of [Gre78]. Then by Lemma 4.25 of [Gre78], \( L \) is weakly 1-iterative. \( \square \)

**Theorem 3.4.** Suppose \( \alpha \) is in \( \omega(1\text{-NCSA}) \). Then \( \alpha \) is \textbf{ultimately periodic}.

**Proof.** Since \( \alpha \) is in \( \omega(1\text{-NCSA}) \), some \( L \in 1\text{-NCSA} \) determines \( \alpha \). Take any such \( L \). Then \( L \) is infinite and every \( s \in L \) is a prefix of \( \alpha \). By Lemma 3.3, \( L \) 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 \( \alpha \), 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 \( \alpha \) is ultimately periodic. \( \square \)

Summarizing, we have the following.
**Theorem 3.5.** \( \omega(\text{REG}) = \omega(\text{CFL}) = \omega(1-\text{DCSA}) = \omega(1-\text{NCSA}) \), and \( \alpha \) is in this class of infinite words iff \( \alpha \) is ultimately periodic.

**Proof.** Immediate from Theorems 3.1, 3.2, and 3.4 and the inclusions \( \text{REG} \subseteq \text{CFL} \) and \( \text{REG} \subseteq 1-\text{DCSA} \subseteq 1-\text{NCSA} \). \( \square \)

### 3.4 Multilinear Words

We now turn to a class of infinite words which we call multilinear. The class of multilinear infinite words properly includes the ultimately periodic words. An infinite word is **multilinear** if it has the form

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

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:

- \( ab \prod_{n \geq 0} cd = abcdcdcd \ldots \)
- \( \prod_{n \geq 0} a^{n+1} b = abaabaab \ldots \)
- \( \prod_{n \geq 0} 10^{2n} = 1100100010000001 \ldots \) (characteristic sequence of the perfect squares)

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

**Theorem 3.6.** Suppose \( \alpha \) is in \( \omega(1-\text{NSA}) \). Then \( \alpha \) is multilinear.

**Proof.** Since \( \alpha \) is in \( \omega(1-\text{NSA}) \), some \( L \in 1-\text{NSA} \) determines \( \alpha \). Take any such \( L \). Then \( L \) is infinite and every \( s \in L \) is a prefix of \( \alpha \). By Ogden’s pumping lemma for one-way stack automata [Ogd69], there are

- strings \( \mu \) and \( \nu \),
- strings \( \rho_i, \sigma_i, \) and \( \tau_i \) for each \( i \geq 0 \),
- strings \( \alpha_j, \beta_j, \phi_j, \chi_j \) and \( \psi_j \) 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 \( i \geq 1 \), \( \rho_i = \alpha_0 \beta_i \phi_1 \beta_i^{-1} \alpha_1 \beta_i^{-1} \phi_2 \beta_i^{-1} \alpha_2 \phi_{m-1} \beta_{2m-2}^{-1} \alpha_{m-1} \).
which is ultimately periodic and hence multilinear.

Theorem 3.7. Suppose \( \alpha \) is multilinear. Then \( \alpha \) is in \( \omega(1\text{-DNESA}) \).

Proof. The infinite word \( \alpha \) has the form:

\[
q \prod_{n \geq 0} r_1^{a_1n+b_1} r_2^{a_2n+b_2} \ldots r_m^{a_mn+b_m}
\]

Let \( \mathcal{A} \) be a one-way deterministic nonerasing stack automaton, operating as follows. First, \( \mathcal{A} \) checks that the input begins with \( q \). In what follows, \( \mathcal{A} \) will push counter symbols onto its stack; so far, the stack is empty. Next, for each \( i \) between 1 and \( m \), \( \mathcal{A} \) 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 \( r_m \), \( \mathcal{A} \) pushes a counter symbol onto its stack and proceeds as before. If any input symbol causes a check to fail, \( \mathcal{A} \) rejects; otherwise, when \( \mathcal{A} \) reaches end of input, it accepts. Now \( \mathcal{A} \) recognizes \( \text{Prefix}(\alpha) \), the full prefix language of \( \alpha \). Since \( \text{Prefix}(\alpha) \) determines \( \alpha \), \( \alpha \) is in \( \omega(1\text{-DNESA}) \).
Theorem 3.8. \(\omega(1-\text{NSA}) = \omega(1-\text{DSA}) = \omega(1-\text{NNESA}) = \omega(1-\text{DNESA})\), and \(\alpha\) is in this class of infinite words iff \(\alpha\) is multilinear.

Proof. Immediate from Theorems 3.6 and 3.7 and the inclusions \(1\)-\text{NSA} \(\supseteq\) \(1\)-\text{DSA} \(\supseteq\) \(1\)-\text{DNESA} and \(1\)-\text{NSA} \(\supseteq\) \(1\)-\text{NNESA} \(\supseteq\) \(1\)-\text{DNESA}. \(\square\)

3.5 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 \textit{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 \textit{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 \textit{growth term}. Any pair \([q, t]\) can be \textit{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

\[\alpha = \prod_{n \geq 0} (abc)^n aba^n = ababcaababcaab\cdots\]

The multilinear pair \([\lambda, [[abc, 1, 0], [ab, 0, 1], [a, 1, 0]]]\) expresses \(\alpha\) 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

\[\alpha = ab \prod_{n \geq 0} a^{n+1} b(cab)^{n+1}\]
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 3.12), and next to show that every multilinear pair satisfying condition 2a is equivalent to a pair satisfying conditions 1, 2a, and 2b (Theorem 3.15). The notion of strong equivalence is used in the proof of Theorem 3.15, 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 3.9.** 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 \(s[i] \neq r_2[i]\). 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 \([s[1] \cdots s[i-1], 0, 1], [s[i] \cdots s[s[i]], 0, 1], [r_2, a_2, b_2]\). 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 \([s, 0, 1], [r_2, a_2, b_2 - 1]\) is equivalent to the original term list, we can return \([s, 0, 1], [r_2[i] \cdots r_2[i-1], 0, 1], [r_2[i] \cdots r_2[2]], 0, 1], [r_2, a_2, b_2 - 1]\). So say there is no such \(i\). Then every symbol in \(s\) and \(r_2\) equals \(r_2[1]\). So return \([r_2[1], r_2 \cdot a_2, s[1] + |r_2[2| \cdot b_2]\). \(\square\)

**Lemma 3.10.** 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 \(r_1^{|r_2|} = r_2^{r_1 |r_1|}\), so by Theorem 1.5.3 of [AS03], there are \(k, l > 0\) such that \(r_1 = z^k\) and \(r_2 = z^l\) for some string \(z\). Then \([z, \frac{|r_1|}{|r_2|} \cdot a_1 + \frac{|r_2|}{|r_1|} \cdot r_2 = \frac{|r_2|}{|r_1|} \cdot b_1 + \frac{|r_2|}{|r_1|} \cdot b_2]\) meets the condition. So say \(r_1^\omega \neq r_2^\omega\). Let \(p\) be the longest common prefix of \(r_1^\omega\) and \(r_2^\omega\). If \(p = \lambda\), then \([r_1, a_1, b_1], [r_2, a_2, b_2]\) already meets the condition. Otherwise, let \(c = (|p| \mod |r_1|) + 1\), let \(d = \frac{|p|}{|r_1|}\) (rounded down), let \(e = (|p| \mod |r_2|) + 1\), 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 \([r_1, a_1, b_1 + d], [r_2[e] \cdots r_2], 0, 1], [r_2, a_2, b_2 - f - 1]\). 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 3.9, 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]\,
- \([r_1[c] \cdots r_1[r_1[1] \cdots r_1[c - 1]], a_1, b_1 + d]\),
- \([r_2[e] \cdots r_2[r_2[1]]], 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 3.9, its first two terms and last two terms can be replaced with equivalent term lists meeting condition 2a.

**Lemma 3.11.** 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, \( r_1 = p y \) and \( x = p z \) for some strings \( y, z \). Let \( u \) be the term list \([[p, 0, 1], [yp, a_1, b_1]]\). By Lemma 3.9, 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 3.12.** 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 3.9, 3.10, and 3.11, 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 3.13.** 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 3.12, 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 3.9 and 3.10, 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 3.14. 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 [AS03], 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 \(r_1^\omega\) and \(s r_1^\omega\). Then \(p\) is a prefix of \(r_1\). 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 \(p x\) for some \(x \neq \lambda\) and \(p\) has the form \(r_1^i y\) for some \(i > 0\) 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 3.13, 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. \(\square\)

Theorem 3.15. Let \(\alpha\) be a multilinear infinite word. Then \(\alpha\) has the form

\[
q \prod_{n > 0} r_1^{a_1 n + b_1} r_2^{a_2 n + b_2} \cdots r_m^{a_m n + b_m}
\]

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. \(\alpha\) 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 3.12, 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 3.14, 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 3.13, 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 3.16.** Every multilinear infinite word is in $\omega(1:multi\text{-}DFA)$.

**Proof.** Take any multilinear infinite word $\alpha$. By Theorem 3.15, $\alpha$ 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 $\alpha$ is ultimately periodic, so $\alpha$ is in $\omega(\text{REG}) \subseteq \omega(1:multi\text{-}DFA)$. 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 3.15, this symbol is different from the first symbol of $r_1$, $A$ 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 $\text{Prefix}(\alpha)$, the full prefix language of $\alpha$. Since $\text{Prefix}(\alpha)$ determines $\alpha$, $\alpha$ is in $\omega(1:multi\text{-}DFA)$.

Finally, we give a simple example of an infinite word in $\omega(1:multi\text{-}DFA)$ which is not multilinear.

**Theorem 3.17.** Not every infinite word in $\omega(1:multi\text{-}DFA)$ is multilinear.

**Proof.** Let $\alpha$ be the infinite word $\prod_{n \geq 0} a^{2^n} b = a^1 b a^2 b a^4 b a^8 b \cdots$. Clearly $\alpha$ 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 $\text{Prefix}(\alpha)$, the full prefix language of $\alpha$. Since $\text{Prefix}(\alpha)$ determines $\alpha$, $\alpha$ is in $\omega(1:\text{multi-DFA})$.

3.6 CONCLUSION

In this chapter 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. One opportunity for future work would be to further characterize the infinite words determined by one-way multihead deterministic finite automata ($1:\text{multi-DFA}$), beyond the result established in this chapter, that the multilinear infinite words are properly contained by this class.
A deterministic L system generates an infinite word $\alpha$ if each word in its derivation sequence is a prefix of the next, yielding $\alpha$ as a limit. We generalize this notion to arbitrary L systems via the concept of prefix languages. As we have seen, 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 a single infinite word. Where $C$ is a class of L systems (e.g. $0L$, $DT0L$), we denote by $\omega(C)$ the class of infinite words determined by the prefix languages in $C$. This allows us to speak of infinite $0L$ words, infinite $DT0L$ words, etc. We categorize the infinite words determined by a variety of L systems, showing that the whole hierarchy collapses to just three distinct classes of infinite words: $\omega(PD0L)$, $\omega(D0L)$, and $\omega(CD0L)$. Our results are obtained with the help of a pumping lemma which we prove for $T0L$ systems.

4.1 Introduction

L systems are parallel rewriting systems which were originally introduced to model growth in simple multicellular organisms. With applications in biological modelling, fractal generation, and artificial life, L systems have given rise to a rich body of research [RS80, KRS97]. L systems can be restricted and generalized in various ways, yielding a hierarchy of language classes.

The simplest L systems are $D0L$ systems (deterministic Lindenmayer systems with 0 symbols of context), in which a morphism is successively applied to a start string or “axiom”. The resulting sequence of words comprises the language of the system. If the morphism is prolongable on the start string, then each word in the derivation sequence will be a prefix of the next, yielding an infinite word as a limit. An infinite word obtained in this way is called an infinite $D0L$ word.

Two well-studied generalizations of $D0L$ systems are $0L$ systems, which introduce nondeterminism by changing the morphism to a finite substitution, and $DT0L$ systems, in which the morphism is replaced by a set of morphisms or “tables”. In each case, there is no longer just one possible derivation sequence; rather, there are many possible derivations, depending on which letter substitutions or tables are chosen at each step. This raises the question of under what conditions such a system can be said to determine an infinite word.

We answer this question with the concept of a prefix language. Recall that 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 a single infinite word. Where C is a class of L systems (e.g. 0L, DT0L), we denote by ω(C) the class of infinite words determined by the prefix languages in C. This allows us to speak of infinite 0L words, infinite DT0L words, etc.

With this notion in place, we categorize the infinite words determined by a variety of L systems. We consider four production features (D,P,F,T) and five filtering features (E,C,N,W,H). Each production feature may be present or absent, and at most one filtering feature may be present, giving a total of $2^4 \cdot 6 = 96$ classes of L systems. We show that this whole hierarchy collapses to just three classes of infinite words: ω(PD0L), ω(D0L), and ω(CD0L). Our results appear in Figure 4.1.

The inclusions among these three classes are proper, giving ω(PD0L) ⊂ ω(D0L) ⊂ ω(CD0L). The class ω(CD0L) contains exactly the morphic words, while ω(D0L) properly contains the pure morphic words.

4.1.1 Proof techniques

We obtain our categorization results by showing that all infinite languages in certain classes of L systems have infinite subsets in certain smaller classes of L systems. This limits the infinite words of the larger class to the infinite words of the smaller class. First, we provide a necessary and sufficient condition under which a T0L system is infinite, in the form of a pumping lemma. In getting this result, we adapt a proof technique used in [Rab12] to obtain a pumping lemma for ET0L systems. It follows from our pumping lemma that every infinite T0L language has an infinite D0L subset. With this result, we show that every infinite ET0L language has an infinite CD0L subset, and we make further use of the pumping lemma to show that every infinite PT0L language has an infinite PD0L subset. A separate argument shows that every infinite ED0L (EPD0L) language has an infinite D0L (PD0L) subset.

4.1.2 Related work

The present chapter is drawn from the author’s conference paper [Smi13b] and also includes material from the author’s conference paper [Smi13a], in particular the pumping lemma for T0L systems.

Our results on infinite subsets can be restated in the framework of set immunity [YS05]. For a language class C, a language L is C-immune if L is infinite and no infinite subset of L is in C. For example, our result that every infinite ET0L language has an infinite CD0L subset could be stated: no ET0L language is CD0L-immune. In addition to categorizing the infinite words determined by L systems, our results characterize the immunity relationships among these systems.
Figure 4.1: Inclusion diagram showing classes of L systems colored by the infinite words they determine. Green classes (diamonds) determine exactly $\omega(PD_0L)$, blue classes (rectangles) determine exactly $\omega(D_0L)$, and yellow classes (ellipses) determine exactly $\omega(CD_0L)$. Inclusions and equalities are from [KRS97].
Corollary 4.6, which states our pumping lemma for DT0L systems, can also be proved via a connection with non-negative integer matrices. Each table in a DT0L system can be associated with a “growth matrix” indicating for each production, how many times each symbol appears on the righthand side of that production. Jungers et al. [JPB08] consider the “joint spectral radius” $\rho$ of a finite set of such matrices, distinguishing four cases. In cases (1) and (2) ($\rho = 0$ or $\rho = 1$ with bounded products), the associated DT0L system is finite, whereas in cases (3) and (4) ($\rho > 1$ or $\rho = 1$ with unbounded products), by their Corollary 1 and Proposition 2, assuming every symbol is reachable, the system is pumpable.

### 4.1.3 Outline of chapter

The chapter is organized as follows. Section 4.2 gives preliminary definitions and propositions. Section 4.3 presents our pumping lemma for T0L systems. Section 4.4 gives results on infinite subsets of certain classes of L systems. Section 4.5 categorizes the infinite words determined by the hierarchy of L systems. Section 4.6 separates and characterizes the classes $\omega$(PD0L), $\omega$(D0L), and $\omega$(CD0L). Section 4.7 gives our conclusions.

### 4.2 Preliminaries

Many classes of L systems appear in the literature. Following [KRS97], we consider four production features (D,P,E,T) and five filtering features (E,C,N,W,H). Each production feature may be present or absent, and at most one filtering feature may be present, for a total of $2^4 \cdot 6 = 96$ classes of L systems. The meanings of the features are given below.

<table>
<thead>
<tr>
<th>Feature</th>
<th>Meaning</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>none</td>
<td>A 0L system is a tuple $G = (A, \sigma, w)$ where $A$ is an alphabet, $\sigma$ is a finite substitution on $A$, and $w$ is in $A^*$. The language of $G$ is $L(G) = {s \in \sigma^i(w) \mid i \geq 0}$.</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>Deterministic</td>
<td>A D0L system is a tuple $G = (A, h, w)$ where $A$ is an alphabet, $h$ is a morphism on $A$, and $w$ is in $A^*$. The language of $G$ is $L(G) = {h^i(w) \mid i \geq 0}$.</td>
</tr>
<tr>
<td>P</td>
<td>Propagating</td>
<td>A PD0L system is a D0L system $(A, h, w)$ such that $h$ is nonerasing.</td>
</tr>
</tbody>
</table>

Continued on next page
### Feature | Meaning | Example
--- | --- | ---
F | Finite axiom set | A **DF₀L system** is a tuple $G = (A, h, F)$ where $A$ is an alphabet, $h$ is a morphism on $A$, and $F$ is a finite set of strings in $A^*$. The language of $G$ is $L(G) = \{h^i(f) \mid f \in F$ and $i \geq 0\}$.
T | Tables | A **DT₀L system** is a tuple $G = (A, H, w)$ where $A$ is an alphabet, $H$ is a finite non-empty set of morphisms on $A$ (called “tables”), and $w$ is in $A^*$. The language of $G$ is $L(G) = \{s \mid h_1 \cdots h_i(w) = s$ for some $h_1, \ldots, h_i \in H\}$.
E | Extended | An **ED₀L system** is a tuple $G = (A, h, w, B)$ where $A$ and $B$ are alphabets and $B \subseteq A$, $h$ is a morphism on $A$, and $w$ is in $A^*$. The language of $G$ is $L(G) = \{s \in B^* \mid h^i(w) = s$ for some $i \geq 0\}$.
H | Homomorphism | An **HD₀L system** is a tuple $G = (A, h, w, g)$ such that $G' = (A, h, w)$ is a D₀L system and $g$ is a morphism on $A$. The language of $G$ is $L(G) = \{g(s) \mid s$ is in $L(G')\}$.
C | Coding | A **CD₀L system** is an HD₀L system $(A, h, w, g)$ such that $g$ is a coding.
N | Nonerasing | An **ND₀L system** is an HD₀L system $(A, h, w, g)$ such that $g$ is nonerasing.
W | Weak coding | A **WD₀L system** is an HD₀L system $(A, h, w, g)$ such that $g$ is a weak coding.

These features combine to form complex L systems. For example, an **EPD₀L system** is an ED₀L system $(A, h, w, B)$ such that $h$ is nonerasing. A **T₀L system** is a tuple $G = (A, T, w)$ where $A$ is an alphabet, $T$ is a finite non-empty set of finite substitutions on $A$ (called “tables”), and $w$ is in $A^*$. The language of $G$ is $L(G) = \{s \mid \sigma_1 \cdots \sigma_i(w) \ni s$ for some $\sigma_1, \ldots, \sigma_i \in T\}$. If for all $\sigma \in T$, $\sigma$ is nonerasing, then $G$ is a **PT₀L system**. An **ET₀L system** is a tuple $G = (A, T, w, B)$ where $(A, T, w)$ is a T₀L system $G'$ and $B \subseteq A$. The language of $G$ is $L(G) = L(G') \cap B^*$. See [RS80] and [KRS97] for more definitions.

We call an L system $G$ infinite iff $L(G)$ is infinite. Finiteness of all the L systems considered in this chapter is decidable from Theorem 4.1 of [KRS97]. When speaking of language classes, we denote the class of
D0L languages simply by D0L, and similarly with other classes. An L system feature set is a subset of \( \{D,P,F,T\} \cup \{E,C,N,W,H\} \) containing at most one of \( \{E,C,N,W,H\} \). Let \( \mathcal{L}(S) \) be the language class of L systems with feature set \( S \). For example, \( \mathcal{L}([C,D,T]) = CDT0L \). From the definitions of the features, we have the following inclusions.

**Proposition 4.1** (Structural inclusions). Let \( S \) be an L system feature set. Then:

- \( \mathcal{L}(S \cup \{D\}) \subseteq \mathcal{L}(S) \),
- \( \mathcal{L}(S \cup \{P\}) \subseteq \mathcal{L}(S) \),
- \( \mathcal{L}(S) \subseteq \mathcal{L}(S \cup \{F\}) \),
- \( \mathcal{L}(S) \subseteq \mathcal{L}(S \cup \{T\}) \).

Let \( S \) be an L system feature set containing none of \( \{E,C,N,W,H\} \). Then:

- \( \mathcal{L}(S) \subseteq \mathcal{L}(S \cup \{E\}) \),
- \( \mathcal{L}(S) \subseteq \mathcal{L}(S \cup \{C\}) \),
- \( \mathcal{L}(S \cup \{C\}) \subseteq \mathcal{L}(S \cup \{N\}) \subseteq \mathcal{L}(S \cup \{H\}) \),
- \( \mathcal{L}(S \cup \{C\}) \subseteq \mathcal{L}(S \cup \{W\}) \subseteq \mathcal{L}(S \cup \{H\}) \).

Beyond these structural inclusions, many relationships are known among the language classes; see [KRS97] for a survey. In comparing L system classes, [KRS97] considers two languages to be equal if they differ by the empty word only; otherwise, propagating classes would be automatically different from nonpropagating ones. See Figure 4.1 for a depiction of the known inclusions and equalities.

### 4.3 Pumping Lemma for T0L Systems

In this section we prove a pumping lemma for T0L systems, which we will apply in Section 4.4 to obtain results about infinite subsets of certain classes of L systems. The notion of pumping used in our lemma is as follows. We say that a T0L system \( G = (A,T,w) \) is **pumpable** iff there are \( a, b \in A \) such that (1) some \( s_0 \in L(G) \) contains \( a \), and (2) for some composition \( t \) of tables from \( T \), \( t(a) \) includes a string \( s_1 \) containing distinct occurrences of \( a \) and \( b \) and \( t(b) \) includes a string \( s_2 \) containing \( b \). See Figure 4.2 for a depiction of the pumping process.

**Lemma 4.2.** Suppose the T0L system \( G = (A,T,w) \) is pumpable. Then \( L(G) \) is infinite.

**Proof.** Since \( s_0 \) is in \( L(G) \) and \( t \) is a composition of tables from \( T \), \( t^i(s_0) \subseteq L(G) \) for every \( i \geq 0 \). A simple induction shows that for all \( i \geq 0 \), \( t^i(s_0) \) includes a string containing \( a \) and at least \( i \) copies of \( b \). Hence \( L(G) \) is infinite. \( \square \)
To prove that every infinite T0L system is pumpable, we will need some definitions regarding derivations of T0L systems. A T0L derivation $D$ is a forest (disjoint union of trees) such that for some $n \geq 0$, every path in $D$ from a root to a leaf has length $n$. The roots correspond to symbols of the start string and $n$ is the number of steps in the derivation. For $0 \leq i \leq n$, let level($i$) be the set of nodes at distance $i$ from a root. The nodes in level(0) are the roots of $D$ and those in level($n$) are the leaves. Each node is labeled with a string of length at most 1. For a node $v$, let label($v$) be the label of $v$ and let children($v$) be the set of children of $v$. For a set of nodes $S$, let join($S$) be the string formed by concatenating the labels of the nodes in $S$ in the order in which they would be encountered in a depth-first traversal of $D$. (An ordering is assumed on the roots and on the children of each node.)

Let $G = (A, T, w)$ be a T0L system. For $x \in A^*$, we say that $D$ is a derivation of $x$ in $G$ if

- for each node $v$ of $D$, label($v$) is in $A \cup \{\lambda\}$,
- join(level(0)) = $w$,
- for each $0 \leq i < n$, there is a $t \in T$ such that for each node $v \in$ level($i$), there is an $s \in t$(label($v$)) such that join(children($v$)) = $s$, and
- join(level($n$)) = $x$.

**Lemma 4.3.** Suppose the T0L system $G = (A, T, w)$ is infinite. Then $G$ is pumpable.
Proof. Our proof adapts the technique of “marked sets” (sets of symbols appearing at different levels of the derivation and making a certain contribution to the derived string) from [Rab12]. Let \( p \) be the highest \( |s| \) such that \( s \) is in \( t(c) \) for some \( t \in T \) and \( c \in A \). Let \( m = |A| \cdot 2^{|A|} \). Since \( G \) is infinite, there is an \( x \in A^* \) and derivation \( D \) of \( x \) in \( G \) such that \( |x| > |w| \cdot p^m \). Without loss of generality, choose \( D \) such that for every node \( v \) in \( D \) with label \( \lambda \), \( v \) is not a root and \( v \) is the only child of its parent. Then \( D \) consists of \( |w| \) trees each of maximum outdegree at most \( p \). We call a node marked if (1) it is a leaf and its label is not \( \lambda \), or (2) it is the ancestor of a marked leaf. Call any node with more than one marked child a branch node. Since \( |x| > |w| \cdot p^m \), some tree in \( D \) has more than \( p^m \) marked leaves. Then by Lemma 14 of [Rab12], \( D \) has a path from a root to a leaf with more than \( m \) branch nodes. For \( 0 \leq i \leq n \), let \( \text{marked}(i) \) be the set \( \{ c \in A \mid \text{for some marked } v \in \text{level}(i), \text{label}(v) = c \} \). Then there are \( 0 \leq l_1 < l_2 < n \), branch nodes \( v_1 \in \text{level}(l_1) \) and \( v_2 \in \text{level}(l_2) \), and \( c \in \text{marked}(l_1) \) such that \( v_1 \) is an ancestor of \( v_2 \), label\( (v_1) = \text{label}(v_2) = a \), and \( \text{marked}(l_1) = \text{marked}(l_2) \). For each \( l_1 < i < l_2 \), let \( t_i \) be any \( t \in T \) such that for each node \( v \in \text{level}(i) \), there is an \( s \in t(\text{label}(v)) \) such that \( \text{join} (\text{children}(v)) = s \). Let \( t \) be the composition \( t_{l_2-1} \circ t_{l_2-2} \circ \cdots \circ t_{l_1} \). We have \( \text{join} (\text{level}(l_2)) \in \text{t} (\text{join} (\text{level}(l_1))) \).

Now, since \( v_1 \) is a branch node, its set of descendants in \( \text{level}(l_2) \) contains, in addition to \( v_2 \), a marked node labelled by some \( d \in A \). Since \( \text{marked}(l_1) = \text{marked}(l_2) \), every \( c \in A \) which labels a marked node in \( \text{level}(l_2) \) also labels a marked node in \( \text{level}(l_1) \). By definition, every marked node in \( \text{level}(l_1) \) has a marked descendant in \( \text{level}(l_2) \). A simple induction then shows that for every \( i \geq 0 \), there is a \( c \in A \) such that \( \text{level}(l_2) \) contains a marked node labelled by \( c \), and some \( s \in t^i(d) \) contains \( c \). Hence for every \( i \geq 0 \), \( t^i(d) \) contains a non-empty string. So there are \( j \geq 0 \), \( k \geq 1 \) and \( b \in A \) such that \( t^j(d) \) includes a string containing \( b \) and \( t^k(b) \) includes a string containing \( b \). Then since \( t(a) \) includes a string containing distinct occurrences of \( a \) and \( d \), \( t^{i+1}(a) \) includes a string containing distinct occurrences of \( a \) and \( b \). Then \( t^{k(j+1)}(a) \) includes a string containing distinct occurrences of \( a \) and \( b \) and \( t^{k(j+1)}(b) \) includes a string containing \( b \). So \( G \) is pumpable.

\[\square\]

**Theorem 4.4.** A TOL system is infinite iff it is pumpable.

**Proof.** Immediate from Lemmas 4.2 and 4.3. \[\square\]

**Corollary 4.5.** A 0L system \( G = (A, \sigma, w) \) is infinite iff there are \( a, b \in A \) such that (1) some \( s \in L(G) \) contains \( a \), and (2) for some \( i \geq 0 \), \( \sigma^i(a) \) includes a string containing distinct occurrences of \( a \) and \( b \) and \( \sigma^i(b) \) includes a string containing \( b \).
Corollary 4.6. A DT0L system $G = (A, H, w)$ is infinite iff there are $a, b \in A$ such that (1) some $s \in L(G)$ contains $a$, and (2) for some composition $h$ of morphisms from $H$, $h(a)$ contains distinct occurrences of $a$ and $b$ and $h(b)$ contains $b$.

4.4 INFINITE SUBSETS OF L SYSTEMS

In this section we show that all infinite languages in certain classes of L systems have infinite subsets in certain smaller classes of L systems. This limits the infinite words of the larger class to the infinite words of the smaller class. We make use of the pumping lemma for T0L systems proved in Section 4.3.

Theorem 4.7. Every infinite T0L language has an infinite D0L subset.

Proof. Take any infinite T0L language $L$ with T0L system $G = (A, T, w)$. By Theorem 4.4, $G$ is pumpable for some $a, b \in A$, $s_0, s_1, s_2 \in A^*$, and composition $t$ of tables from $T$. Let $h$ be a morphism on $A$ such that $h(a) = s_1$, $h(b) = s_2$ unless $a = b$, and for every other $c \in A$, $h(c) = s$ for some $s \in t(c)$. Consider the language $L'$ of the D0L system $(A, h, s_0)$. For all $c \in A$, $h(c)$ is in $t(c)$, so since $s_0$ is in $L$, we have $L' \subseteq L$. A simple induction shows that for all $i \geq 0$, $h^i(s_0)$ contains $a$ and at least $i$ copies of $b$. Then $L'$ is an infinite D0L subset of $L$. □

Theorem 4.8. Every infinite PT0L language has an infinite PD0L subset.

Proof. Take any infinite PT0L language $L$ with PT0L system $G = (A, T, w)$. Follow the proof of Theorem 4.7, and note that since $t$ is a composition of tables from $T$ and every table in $T$ is now nonerasing, $t$ is nonerasing, hence $h$ is nonerasing. Then $L'$ is an infinite PD0L subset of $L$. □

Theorem 4.9. Let $G = (A, h, w, B)$ be an infinite ED0L system. Then there are $k \geq 0$, $p \geq 1$ such that the language of the D0L system $(A, h^p, h^k(w))$ is an infinite subset of $L(G)$.

Proof. Let $\text{alph}(s)$ be the set of symbols which appear in the string $s$. Since $L(G)$ is infinite, there is an $m \geq 0$ such that the sequence $w, h(w), h^2(w), \ldots, h^m(w)$ contains more than $2^{|B|}$ strings in $L(G)$. For every $s \in L(G)$, $\text{alph}(s) \subseteq B$. Hence there is a $C \subseteq B$ and $i, j$ such that $0 \leq i < j \leq m$ and $\text{alph}(h^i(w)) = \text{alph}(h^j(w)) = C$. Then for every string $s$ such that $\text{alph}(s) = C$, $\text{alph}(h^{j-i}(s)) = C$. Let $k = i$ and $p = j - i$. Then for every $n \geq 0$, $\text{alph}(h^{k+p}(w)) = C$. Hence for every $n \geq 0$, $h^{k+p}(w)$ is in $L(G)$. So take the D0L system $G' = (A, h^p, h^k(w))$. We have $L(G') \subseteq L(G)$. Suppose some string $s$ occurs twice in the derivation sequence of $G'$. Then $s$ occurs twice in the derivation sequence of $G$, making $L(G)$ finite, a contradiction. So $L(G')$ is infinite. Therefore $L(G')$ is an infinite subset of $L(G)$. □
Corollary 4.10. Every infinite ED0L language has an infinite D0L subset.

Corollary 4.11. Every infinite EPD0L language has an infinite PD0L subset.

Proof. Take any infinite EPD0L system \( G = (A, h, w, B) \). By Theorem 4.9, there are \( k \geq 0, p \geq 1 \) such that the language of the D0L system \( G' = (A, h^p, h^k(w)) \) is an infinite subset of \( L(G) \). Since \( h \) is nonerasing, \( h^p \) is nonerasing. Hence \( L(G') \) is an infinite PD0L subset of \( L(G) \).

Theorem 4.12. Every infinite ET0L language has an infinite CD0L subset.

Proof. Take any infinite ET0L language \( L \). By Theorem 2.7 of [KRS97], ET0L = CT0L. Hence there is a coding \( e \) and T0L language \( L' \) such that \( L = e(L') \). Since \( L \) is infinite, \( L' \) is infinite. Then by Theorem 4.7, \( L' \) has an infinite D0L subset \( L'' \). Since \( L'' \) is infinite and \( e \) is a coding, \( e(L'') \) is infinite. Since \( L'' \subseteq L' \), \( e(L'') \subseteq e(L') \). Therefore \( e(L'') \) is an infinite CD0L subset of \( L \).

Theorem 4.13. Let \( S \) be an L system feature set not containing \( F \). Then every infinite \( \mathcal{L}(S \cup \{ F \}) \) language has an infinite \( \mathcal{L}(S) \) subset.

Proof. Take any infinite L system \( G \) with feature set \( S \cup \{ F \} \). Since \( G \) has a finite axiom set, \( L(G) \) is a finite union of \( \mathcal{L}(S) \) languages. Then since \( L(G) \) is infinite, at least one of these \( \mathcal{L}(S) \) languages is infinite. Therefore \( L(G) \) has an infinite \( \mathcal{L}(S) \) subset.

Theorem 4.14. Let \( C \) and \( D \) be language classes such that every infinite language in \( C \) has an infinite subset in \( D \). Then \( \omega(C) \subseteq \omega(D) \).

Proof. Take any \( \alpha \in \omega(C) \). Some \( L \in C \) determines \( \alpha \). Then \( L \) is infinite, so \( L \) has an infinite subset \( L' \) in \( D \). Then \( L' \) determines \( \alpha \). So \( \alpha \) is in \( \omega(D) \). Hence \( \omega(C) \subseteq \omega(D) \).

4.5 CATEGORIZATIONS

In this section we categorize the infinite words determined by each class of L systems, making use of the results about infinite subsets from the previous section. We partition the 96 classes into three sets, called Set1, Set2, and Set3, and show that for every \( C_1 \in \text{Set1}, \ C_2 \in \text{Set2}, \) and \( C_3 \in \text{Set3}, \) \( \omega(C_1) = \omega(\text{PD0L}), \omega(C_2) = \omega(\text{D0L}), \) and \( \omega(C_3) = \omega(\text{CD0L}). \)

4.5.1 PD0L classes

Let \( \text{Set1} = \{\text{PD0L}, \text{PDF0L}, \text{P0L}, \text{PF0L}, \text{PDT0L}, \text{PDTF0L}, \text{PT0L}, \text{PTF0L}, \text{EPD0L}, \text{EPDF0L}\} \).
**Theorem 4.15.** For every \( C \in \text{Set}_1 \), every infinite \( C \) language has an infinite PD0L subset.

**Proof.** Take any \( C \in \text{Set}_1 \). By structural inclusion, \( C \subseteq \text{PTF0L} \) or \( C \subseteq \text{EPDF0L} \). By Theorem 4.13, every infinite PTF0L language has an infinite PT0L subset. By Theorem 4.8, every infinite PT0L language has an infinite PD0L subset. Hence every infinite PTF0L language has an infinite PD0L subset. By Theorem 4.13, every infinite EPDF0L language has an infinite EPD0L subset. By Corollary 4.11, every infinite EPD0L language has an infinite PD0L subset. Hence every infinite \( C \) language has an infinite PD0L subset. \( \square \)

**Theorem 4.16.** For every \( C \in \text{Set}_1 \), \( \omega(C) = \omega(\text{PD0L}) \).

**Proof.** Take any \( C \in \text{Set}_1 \). By structural inclusion, PD0L \( \subseteq C \). Hence \( \omega(\text{PD0L}) \subseteq \omega(C) \). By Theorem 4.15, every infinite \( C \) language has an infinite PD0L subset. Then by Theorem 4.14, \( \omega(C) \subseteq \omega(\text{PD0L}) \). Therefore \( \omega(C) = \omega(\text{PD0L}) \). \( \square \)

### 4.5.2 D0L classes

Let \( \text{Set}_2 = \{ \text{D0L}, \text{DF0L}, \text{0L}, \text{F0L}, \text{DT0L}, \text{DTF0L}, \text{T0L}, \text{TF0L}, \text{ED0L}, \text{EDF0L} \} \).

**Theorem 4.17.** For every \( C \in \text{Set}_2 \), every infinite \( C \) language has an infinite D0L subset.

**Proof.** Take any \( C \in \text{Set}_2 \). By structural inclusion, \( C \subseteq \text{TF0L} \) or \( C \subseteq \text{EDF0L} \). By Theorem 4.13, every infinite TF0L language has an infinite T0L subset. By Theorem 4.7, every infinite T0L language has an infinite D0L subset. Hence every infinite TF0L language has an infinite D0L subset. By Theorem 4.13, every infinite EDF0L language has an infinite ED0L subset. By Corollary 4.10, every infinite ED0L language has an infinite D0L subset. Hence every infinite EDF0L language has an infinite D0L subset. Hence every infinite \( C \) language has an infinite D0L subset. \( \square \)

**Theorem 4.18.** For every \( C \in \text{Set}_2 \), \( \omega(C) = \omega(\text{D0L}) \).

**Proof.** Take any \( C \in \text{Set}_2 \). By structural inclusion, D0L \( \subseteq C \). Hence \( \omega(\text{D0L}) \subseteq \omega(C) \). By Theorem 4.17, every infinite \( C \) language has an infinite D0L subset. Then by Theorem 4.14, \( \omega(C) \subseteq \omega(\text{D0L}) \). Therefore \( \omega(C) = \omega(\text{D0L}) \). \( \square \)

### 4.5.3 CD0L classes

Let \( \text{Set}_3 = \{ \text{CD0L}, \text{ND0L}, \text{WD0L}, \text{HD0L}, \text{CPD0L}, \text{NPD0L}, \text{WPD0L}, \text{HPD0L}, \text{CDF0L}, \text{NDF0L}, \text{WDF0L}, \text{HDF0L}, \text{CPDF0L}, \text{NPDF0L}, \text{WPDF0L} \} \).
4.5 $\omega(PDOL), \omega(DOL), \text{and} \ \omega(CDOL)$

In this section, we separate the three classes of infinite words obtained in the previous section, giving $\omega(PDOL) \subset \omega(DOL) \subset \omega(CDOL)$. We observe that $\omega(DOL)$ properly contains the pure morphic words and we show that $\omega(CDOL)$ contains exactly the morphic words.
4.6.1 Separating the classes

From Theorem 2.3 of [Pan85], the infinite words generated by iterating nonerasing morphisms are a proper subset of the pure morphic words, which in turn are a proper subset of the morphic words. Our classes \( \omega(\text{PDOL}), \omega(\text{DOL}), \) and \( \omega(\text{CDOL}) \) are defined more generally using prefix languages, but similar arguments serve to separate them.

**Theorem 4.21.** \( \omega(\text{PDOL}) \subset \omega(\text{DOL}). \)

**Proof.** By structural inclusion, \( \omega(\text{PDOL}) \subset \omega(\text{DOL}). \) To separate the two classes, we use an infinite word from [CN03]. Let \( A = \{0, 1, 2\}. \) Let \( f \) be a morphism on \( A \) such that \( f(0) = 01222, f(1) = 10222, \) and \( f(2) = \lambda. \) Let \( \alpha = f^\omega(0) = 01222102221022201222 \cdots. \) Then \( \alpha \) is a pure morphic word, hence \( \alpha \) is in \( \omega(\text{DOL}). \) In [CN03] it is shown that there is no nonerasing morphism \( g \) on \( A \) such that \( g^\omega(0) = \alpha. \) We generalize this result to show that \( \alpha \) is not in \( \omega(\text{PDOL}). \) First, we show that if \( g \) is a nonerasing morphism on \( A \) and \( g(\alpha) = \alpha, \) then \( g \) is an identity morphism. We adapt the proof of Example 3 in [CN03].

Let \( \tau \) be the Thue-Morse word \( \tau = 01101001 \cdots = u^\omega(0), \) where \( u \) is a morphism on \( \{0, 1\} \) such that \( u(0) = 01 \) and \( u(1) = 10. \) Let \( d \) be a morphism on \( A \) such that \( d(0) = 0, d(1) = 1, \) and \( d(2) = \lambda. \) As observed by [CN03], \( d(\alpha) = \tau. \) Notice that the only subwords of \( \alpha \) in \( \{0, 1\}^\ast \) are in \( \{\lambda, 0, 1, 01, 10\} \) and the only subwords of \( \alpha \) in \( (2)^\ast \) are in \( \{\lambda, 2, 22, 222\} \). Notice also that \( \alpha \) does not contain the subword 212.

Suppose \( g \) is a nonerasing morphism on \( A \) and \( g(\alpha) = \alpha. \) Suppose \( g(2) \) is not in \( 2^\ast. \) Let \( s = d(g(2)). \) Then \( s \) is not empty. Since 222 is a subword of \( \alpha \) and \( g(\alpha) = \alpha, \) \( g(222) \) is a subword of \( \alpha. \) Then since \( d(\alpha) = \tau, \tau \) contains \( d(g(222)) = ss, \) a contradiction, since \( \tau \) is known to be cubefree. So \( g(2) \) is in \( 2^\ast. \) Then since \( \alpha \) contains \( g(222), \) and 2222 is not a subword of \( \alpha, \) and \( g \) is nonerasing, \( g(2) = 2. \)

Suppose \( g(0) \neq 0. \) Then since \( \alpha \) starts with 0, \( g(0) = 01x \) for some \( x \in A^\ast. \) Since 1222 is a subword of \( \alpha, g(1222) = g(1) \) 222 is a subword of \( \alpha. \) Then since 2222 is not a subword of \( \alpha, g(1) \) cannot end with 2. So \( g(1) = y a \) for some \( y \in A^\ast \) and \( a \in \{0, 1\}. \) Now since 10 is a subword of \( \alpha, \) so is \( g(10) = ya01. \) But \( \alpha \) contains no subword of the form \( a01, \) a contradiction. So \( g(0) = 0. \)

Suppose \( g(1) \neq 1. \) Then since \( \alpha \) begins with 012, \( g(1) = 12z \) for some \( z \in A^\ast. \) Since 2221 is a subword of \( \alpha, g(2221) = 22212z \) is a subword of \( \alpha, \) a contradiction, since \( \alpha \) does not contain the subword 212. So \( g(1) = 1. \) Then \( g \) is an identity morphism.

So suppose \( \alpha \) is in \( \omega(\text{PDOL}). \) Then there is a PDOL system \( G = (A, h, w) \) such that \( L(G) \) determines \( \alpha. \) Since \( h \) is nonerasing, \( h(\alpha) \) is an infinite word. Suppose \( h(\alpha) \neq \alpha. \) Then there is a prefix \( p \) of \( \alpha \) such that \( h(p) \) is not a prefix of \( \alpha. \) Since \( L(G) \) determines \( \alpha, p \) is a prefix of some \( s \) in \( L(G). \) Then \( h(p) \) is a prefix of \( h(s). \) But then since \( h(s) \) is in \( L(G), h(p) \) is a prefix of \( \alpha, \) a contradiction. So \( h(\alpha) = \alpha. \) Then from above, \( h \) is an identity morphism. But then \( h(w) = w, \) so \( L(G) \) is
finite, a contradiction. Therefore \( \alpha \) is not in \( \omega(\text{PDOL}) \). Hence \( \omega(\text{PDOL}) \subset \omega(\text{DOL}) \).

**Theorem 4.22.** \( \omega(\text{DOL}) \subset \omega(\text{CDOL}) \).

**Proof.** By structural inclusion, \( \omega(\text{DOL}) \subset \omega(\text{CDOL}) \). Let \( \alpha = abba^{\omega} \). Since \( \alpha \) is ultimately periodic, \( \alpha \) is morphic, hence \( \alpha \) is in \( \omega(\text{CDOL}) \). Suppose \( \alpha \) is in \( \omega(\text{DOL}) \). Then there is a DOL system \( G = (A, h, w) \) such that \( L(G) \) determines \( \alpha \). Clearly \( h(a) \) cannot include \( b \), and if \( h(a) = \lambda \), \( L(G) \) is finite, a contradiction. So since \( h(a) \) must be a prefix of \( \alpha \), \( h(a) = a \). Then \( a \ h(b) \) is a prefix of \( \alpha \), hence \( h(b) = \lambda \) or \( h(b) = b \). But then \( L(G) \) is finite, a contradiction. So \( \alpha \) is not in \( \omega(\text{DOL}) \). Hence \( \omega(\text{DOL}) \subset \omega(\text{CDOL}) \).

4.6.2 **Characterizing the words in each class**

That \( \omega(\text{DOL}) \) includes every pure morphic word is immediate from the definitions. In [Hon10], the infinite word \( ab^a \omega \) is given as an example of an infinite DOL word which is not pure morphic. Hence \( \omega(\text{DOL}) \) properly contains the pure morphic words. Next, we show that \( \omega(\text{CDOL}) \) contains exactly the morphic words. The adherence of a language \( L \), denoted \( \text{Adherence}(L) \), is the set \( \{ \alpha | \alpha \text{ is an infinite word and for every prefix } p \text{ of } \alpha, \text{there is an } s \in L \text{ such that } p \text{ is a prefix of } s \} \).

**Lemma 4.23.** Suppose \( L \) is in DOL and \( \alpha \) is in \( \text{Adherence}(L) \). Then \( \alpha \) is morphic.

**Proof.** From [Hea84], either (1) \( \alpha \) is ultimately periodic, or (2) \( \alpha = w \times h(x) \cdot h^2(x) \cdot \cdots \) for some morphism \( h \) and strings \( w, x \) such that \( h(w) = wx \) and \( x \) is not mortal. If (1), \( \alpha \) is morphic. If (2), \( \alpha \) is an infinite DOL word, so by Proposition 10.2.2 of [Hon10], \( \alpha \) is morphic.

**Theorem 4.24.** \( \alpha \) is in \( \omega(\text{CDOL}) \) iff \( \alpha \) is morphic.

**Proof.** That \( \omega(\text{CDOL}) \) includes every morphic word is immediate from the definitions. So take any \( \alpha \in \omega(\text{CDOL}) \). Then there is a CDOL system \( G = (A, h, w, e) \) such that \( L(G) \) determines \( \alpha \). Then \( L(G) \) is infinite. Hence the language \( L \) of the DOL system \( (A, h, w) \) is infinite. As noted in [Hea84], a language has empty adherence iff the language is finite. Therefore there is an \( \alpha' \in \text{Adherence}(L) \). By Lemma 4.23, \( \alpha' \) is morphic. Now for any prefix \( p \) of \( \alpha' \), there is a string \( s \) in \( L \) with \( p \) as a prefix. Then \( e(p) \) is a prefix of \( e(s) \). Then since \( e(s) \) is in \( L(G) \), \( e(p) \) is a prefix of \( \alpha \). So for every prefix \( p \) of \( \alpha' \), \( e(p) \) is a prefix of \( \alpha \). Since \( e \) is a coding, \( e(\alpha') \) is infinite. So \( e(\alpha') = \alpha \). Then because a coding of a morphic word is still a morphic word, \( \alpha \) is morphic. Hence \( \alpha \) is in \( \omega(\text{CDOL}) \) iff \( \alpha \) is morphic.
4.7 CONCLUSION

In this chapter we have categorized the infinite words determined by L systems, showing that a variety of classes of L systems collapse to just three classes of infinite words. To associate L systems with infinite words, we used the concept of prefix languages. The broadest class of prefix languages we considered was that of ET0L; we found that this class (along with several smaller classes of L systems) determines exactly the morphic words. In Chapter 5 we will show that even the class of indexed languages, a superset of ET0L, can determine only morphic words.
In this chapter, we characterize the infinite words determined by indexed languages. We show that if an indexed language $L$ determines an infinite word $\alpha$, then $\alpha$ is a morphic word, i.e., $\alpha$ can be generated by iterating a morphism under a coding. Since the other direction, that every morphic word is determined by some indexed language, also holds, this implies that the infinite words determined by indexed languages are exactly the morphic words. To obtain this result, we prove a new pumping lemma for the indexed languages, which may be of independent interest.

5.1 Introduction

The indexed languages, introduced in 1968 by Alfred Aho [Aho68], fall between the context-free and context-sensitive languages in the Chomsky hierarchy. More powerful than the former class and more tractable than the latter, the indexed languages have been applied to the study of natural languages [Gaz88] in computational linguistics. Indexed languages are generated by indexed grammars, in which nonterminals are augmented with stacks which can be pushed, popped, and copied to other nonterminals as the derivation proceeds. Two automaton characterizations are the nested stack automata of [Aho69], and the order-2 pushdown automata within the Maslov pushdown hierarchy [Mas76].

The class of indexed languages $IL$ includes all of the stack automata classes whose infinite words are characterized in [Smi13c], as well as all of the rewriting system classes whose infinite words are characterized in [Smi13b]. In particular, $IL$ properly includes $ET\Omega$ [ERS76], a broad class within the hierarchy of parallel rewriting systems known as $L$ systems. $L$ systems have close connections with a class of infinite words called morphic words, which are generated by repeated application of a morphism to an initial symbol, under a coding [AS03]. In [Smi13b] it is shown that every infinite word determined by an $ET\Omega$ language is morphic. This raises the question of whether the indexed languages too determine only morphic words, or whether indexed languages can determine infinite words which are not morphic.

To answer this question, we employ a new pumping lemma for $IL$. In Book’s paper, as well as in [Smi13b] and [Smi13c], pumping lemmas played a prominent role in characterizing the infinite words determined by various language classes. A pumping lemma for a lan-
guage class $C$ is a powerful tool for proving that certain languages do not belong to $C$, and thereby for proving that certain infinite words cannot be determined by any language in $C$. For the indexed languages, a pumping lemma exists due to Hayashi [Hay73], as well as a “shrinking lemma” due to Gilman [Gil96]. We were not successful in using these lemmas to characterize the infinite words determined by IL, so instead have proved a new pumping lemma for this class (Theorem 5.7), which may be of independent interest.

Our lemma generalizes a pumping lemma recently proved for T0L languages [Rab12]. Roughly, it states that for any indexed language $L$, any sufficiently long word $w \in L$ may be written as $u_1 \cdots u_n$, each $u_i$ may be written as $v_{i,1} \cdots v_{i,n_i}$, and the $v_{i,j}$s may be replaced with $u_i$s to obtain new words in $L$. Using this lemma, we extend to IL a theorem about frequent and rare symbols proved in [Rab12] for ET0L, which can be used to prove that certain languages are not indexed. We also use the lemma to obtain the new result that every infinite indexed language has an infinite subset in a smaller class of $L$ systems called CD0L. This implies that every infinite word determined in IL can also be determined in CD0L, and thus that every such word is morphic. Since every morphic word can be determined by some CD0L language [Smi13b], we therefore obtain a complete characterization of the infinite words determined by indexed languages: they are exactly the morphic words.

5.1.1 Proof techniques

Our pumping lemma for IL generalizes the one proved in [Rab12] for ET0L. Derivations in an ET0L system, like those in an indexed grammar, can be viewed as having a tree structure, but with certain differences. In ET0L, symbols are rewritten in parallel, and the tree is organized into levels corresponding to the steps of the derivation. Further, each node in the tree has one of a finite set of possible labels, corresponding to the symbols in the ET0L system. The proof in [Rab12] classifies each level of the tree according to the set of symbols which appear at that level, and then finds two levels with the same symbol set, which are used to construct the pumping operation. By contrast, the derivation tree of an indexed grammar is not organized into levels in this way, and there is no bound on the number of possible labels for the nodes, since each nonterminal can have an arbitrarily large stack. We deal with these differences by assigning each node a “type” based on the set of nonterminals which appear among its descendants immediately before its stack is popped. These types then play a role analogous to the symbol sets of [Rab12] in our construction of the pumping operation.
5.1.2 Related work

The present chapter is drawn from the author’s conference paper [Smi14a].

Hayashi’s 1973 pumping lemma for indexed languages is proved in a dense thirty-page paper [Hay73]. The main theorem states that if a given terminal derivation tree is big enough, new terminal derivation trees can be generated by the insertion of other trees into the given one. Hayashi applies his theorem to give a new proof that the finiteness problem for indexed languages is solvable and to show that certain languages are not indexed. Gilman’s 1996 “shrinking lemma” for indexed languages [Gil96] is intended to be easier to employ, and operates directly on terminal strings rather than on derivation trees. Our lemma generalizes the recent ET0L pumping lemma of Rabkin [Rab12]. Like Gilman’s lemma, it is stated in terms of strings rather than derivation trees, making it easier to employ, while like Hayashi’s lemma and unlike Gilman’s, it provides a pumping operation which yields an infinity of new strings in the language.

Another connection between indexed languages and morphic words comes from Braud and Carayol [BC10], in which morphic words are related to a class of graphs at level 2 of the pushdown hierarchy. The string languages at this level of the hierarchy are the indexed languages.

5.1.3 Outline of chapter

The chapter is organized as follows. Section 5.2 gives preliminary definitions and propositions. Section 5.3 characterizes the infinite words determined by the class of linear indexed languages, a subset of IL. Section 5.4 proves our pumping lemma for indexed languages. Section 5.5 gives applications for the lemma, in particular characterizing the infinite words determined by indexed languages. Section 5.6 gives our conclusions.

5.2 Preliminaries

5.2.1 Morphic words and L systems

Morphic words, which were defined in Chapter 2, have close connections with the parallel rewriting systems known as L systems, as we saw in Chapter 4. Many classes of L systems appear in the literature; here we review the definitions of HD0L and CD0L. For more on L systems, including the class ET0L, see Chapter 4. An HD0L system is a tuple $G = (A, h, w, g)$ where $A$ is an alphabet, $h$ and $g$ are morphisms on $A$, and $w$ is in $A^*$. The language of $G$ is $L(G) = \{g(h^i(w)) \mid i \geq 0\}$. If $g$ is a coding, $G$ is a CD0L system. HD0L and CD0L are the sets of
HD0L and CD0L languages, respectively. From [KRS97] and [ERS76] we have CD0L ⊆ HD0L ⊆ ET0L ⊆ IL. In Chapter 4 it is shown that \( \omega(\text{CD0L}) = \omega(\text{HD0L}) = \omega(\text{ET0L}) \), and \( \alpha \) is in this class of infinite words iff \( \alpha \) is morphic.

### 5.2.2 Indexed languages

The class of indexed languages IL consists of the languages generated by indexed grammars. These grammars extend context-free grammars by giving each nonterminal its own stack of symbols, which can be pushed, popped, and copied to other nonterminals as the derivation proceeds. Indexed grammars come in several forms [Aho68, Gaz88, HU79], all generating the same class of languages, but varying with respect to notation and which productions are allowed. The following definition follows the form of [HU79].

An **indexed grammar** is a tuple \( G = (N, T, F, P, S) \) in which \( N \) is the nonterminal alphabet, \( T \) is the terminal alphabet, \( F \) is the stack alphabet, \( S \in N \) is the start symbol, and \( P \) is the set of productions of the forms

\[
A \rightarrow r \\
A \rightarrow Bf \\
Af \rightarrow r
\]

with \( A, B \in N, f \in F, \) and \( r \in (N \cup T)^* \). In an expression of the form \( Af_1 \cdots f_n \) with \( A \in N \) and \( f_1, \ldots, f_n \in F \), the string \( f_1 \cdots f_n \) can be viewed as a stack joined to the nonterminal \( A \), with \( f_1 \) denoting the top of the stack and \( f_n \) the bottom. For \( r \in (N \cup T)^* \) and \( x \in F^* \), we write \( r(x) \) to denote \( r \) with every \( A \in N \) replaced by \( Ax \). For example, with \( A, B \in N \) and \( c, d \in T \), \( cdAB(f) = cdAfBf \). For \( q, r \in (NF^* \cup T)^* \), we write \( q \rightarrow r \) if there are \( q_1, q_2 \in (NF^* \cup T)^* \), \( A \in N \), \( p \in (N \cup T)^* \), and \( x, y \in F^* \) such that \( q = q_1 Ax q_2, r = q_1 p(y) q_2, \) and one of the following is true: (1) \( A \rightarrow p \) is in \( P \) and \( y = x \), (2) \( A \rightarrow pf \) is in \( P \) and \( y = fx \), or (3) \( Af \rightarrow p \) is in \( P \) and \( x = fy \). Let \( \rightarrow^* \) be the reflexive, transitive closure of \( \rightarrow \). For \( A \in N \) and \( x \in F^* \), let \( L(Ax) = \{s \in T^* \mid Ax \rightarrow^* s\} \). The language of \( G \), denoted \( L(G) \), is \( L(S) \). The class IL of indexed languages is \( \{L(G) \mid G \text{ is an indexed grammar}\} \).

See Example 5.8 and Figure 5.1 for a sample indexed grammar and derivation tree. For convenience, we will work with a form of indexed grammar which we call “grounded”, in which terminal strings are produced only at the bottom of the stack. \( G \) is **grounded** if there is a symbol \( S \in F \) (called the bottom-of-stack symbol) such that every production has one of the forms

\[
S \rightarrow A$ \\
A \rightarrow r \\
A \rightarrow Bf \\
Af \rightarrow r \\
A$ \rightarrow s
\]

with \( A, B \in N \setminus S, f \in F \setminus $, \( r \in (N \setminus S)^+, \) and \( s \in T^* \).

**Proposition 5.1.** For every indexed grammar \( G \), there is a grounded indexed grammar \( G' \) such that \( L(G') = L(G) \).
Proof. Let \( G = (N, T, F, P, S) \). Add a nonterminal \( S' \) to \( N \) and replace every occurrence of \( S \) in \( P \) with \( S' \). Then add a symbol \( $ \) to \( F \) and add to \( P \) the production \( S \rightarrow S'$. Next, for every \( t \in T \), add a nonterminal \( X_t \) to \( N \) and replace every occurrence of \( t \) in \( P \) with \( X_t \). Then add a nonterminal \( X_{\lambda} \) to \( N \) and replace every production of the form \( A \rightarrow \lambda \) with \( A \rightarrow X_{\lambda} \). Finally, for every \( s \in T \cup \{\lambda\} \), add to \( P \) the production \( X_s \rightarrow X_s \). The resulting grammar \( G' \) is grounded and \( L(G') = L(G) \).

5.3 LINEAR INDEXED LANGUAGES

Before tackling the class of indexed languages IL, we consider the smaller class of linear indexed languages LIL. Linear indexed languages are generated by linear indexed grammars (LIGs), indexed grammars in which only designated nonterminals receive a copy of the stack. Specifically, in each production of an LIG, at most one nonterminal on the righthand side is designated as the stack recipient. When a production is applied during a derivation, only its designated nonterminal receives a copy of the stack from the nonterminal on the lefthand side of the production; any other nonterminals on the righthand side receive empty stacks. Linear indexed grammars were introduced in [Gaz88] and named in [VS87].

The class LIL of linear indexed languages properly contains the context-free languages and is properly contained by the indexed languages. Vijay-Shanker and Weir [VSW94] showed that four extensions of context-free grammars (combinatory categorial grammars, head grammars, linear indexed grammars, and tree-adjoining grammars) all generate exactly the language class LIL. To characterize the infinite words determined by LILs, we use an existing pumping lemma for this class, due to [VS87].

Theorem 5.2. \( \omega(LIL) = \omega(REG) \).

Proof. Take any infinite word \( \alpha \) in \( \omega(LIL) \). Some \( L \in LIL \) determines \( \alpha \). Take any such \( L \). Then \( L \) is infinite and every \( s \in L \) is a prefix of \( \alpha \). By the pumping lemma for linear indexed languages of [VS87], there is a string \( xw_1v_1w_2yw_3v_2w_4z \) such that \( |w_1w_2w_3w_4| \geq 1 \) and for all \( n \geq 0 \), \( xw_1^n v_1w_2^n yw_3^n v_2^n w_4^n z \) is in \( L \). Suppose \( |w_1| \geq 1 \). Then because every string in \( L \) is a prefix of \( \alpha \), and every prefix of such a string

1 In [DP84], the term “linear indexed grammar” refers to an even smaller subset of indexed grammars, those in which the righthand side of each production contains at most one nonterminal. These are equivalent to the RIIL grammars of [Aho68]. Following a suggestion of Eli Shamir, we propose calling these grammars “indexed linear grammars” or ILGs, and the corresponding class of languages ILL. ILL properly contains CFL but is properly contained by LIL. For example, the language \( \{a^m b^n c^m d^n b^n c^n \mid m, n \geq 0\} \) can be generated by an ILG, but not by an ILG (due to Theorem 2.8 of [DP84]).
is also a prefix of $\alpha$, $xw_1$ is a prefix of $\alpha$, as are $xw_1w_1$, $xw_1w_1w_1$, and so on. Consequently $\alpha = xw_1^\omega$, so $\alpha$ is ultimately periodic. So say $|w_1| = 0$. Then at least one of $|w_2|$, $|w_3|$, and $|w_4|$ must be $\geq 1$, so by the same line of reasoning, $\alpha$ must be $xv_1w_2^\omega$, $xv_1yw_3^\omega$, or $xv_1yw_2w_4^\omega$. In each case, $\alpha$ is ultimately periodic. Therefore every infinite word in $\omega(LIL)$ is ultimately periodic, so by Theorem 5.1, $\omega(LIL) \subseteq \omega(REG)$. Since REG $\subseteq$ LIL, we get $\omega(LIL) = \omega(REG)$. \hfill $\Box$

5.4 PUMPING LEMMA FOR Indexed LANGUAGES

In this section we present our pumping lemma for indexed languages (Theorem 5.7) and give an example of its use. Our pumping lemma generalizes the ET0L pumping lemma of [Rab12]. Like that lemma, it allows positions in a word to be designated as “marked”, and then provides guarantees about the marked positions during the pumping operation. We begin with definitions regarding derivation trees of indexed grammars, followed by a lemma about paths in those trees, and then the main theorem.

5.4.1 Derivation trees

Let $G = (N, T, F, P, S)$ be a grounded indexed grammar. A derivation tree $D$ of a string $s$ has the following structure. Each internal node of $D$ has a label in $NF^*$ (a nonterminal with a stack), and each leaf has a label in $T^*$ (a terminal string). Each internal node has either a single leaf node as a child, or one or more internal children. The root of $D$ is labelled by the start symbol $S$, and the terminal yield of $D$ is the string $s$. See Figure 5.1 for an example of a derivation tree.

Any positions in the string $s$ may be chosen to be “marked”. If $s$ contains marked positions, then we will take $D$ to be marked in the following way. Mark every leaf whose label contains a marked position of $s$, and then mark every internal node which has a marked descendant. Call any node with more than one marked child a branch node.

A path $H$ in $D$ is a list of nodes $(v_0, \ldots, v_m)$ with $m \geq 0$ such that for each $1 \leq i \leq m$, $v_i$ is a child of $v_{i-1}$. For convenience, we will sometimes refer to nodes in $H$ by their indices; e.g. node $i$ in the context of $H$ means $v_i$. When we say that there is a branch node between $i$ and $j$ we mean that the branch node is between $v_i$ (inclusive) and $v_j$ (exclusive).

For a node $v$, $D(v)$ means the subtree of $D$ whose root is $v$. We denote the terminal yield of $D(v)$ by $\text{yield}(v)$. For nodes $v_1, v_2$ in $D$, we say that $v_1$ is to the left of $v_2$, and $v_2$ is to the right of $v_1$, if neither node is descended from the other, and if $v_1$ would be encountered before $v_2$ in a depth-first traversal of $D$ in which edges are chosen
from left to right. A node $v_1$ is **reachable** from a node $v_2$ if $v_1$ is identical to $v_2$ or descended from $v_2$.

We define several operations on internal nodes of $D$. Each such node $v$ has the label $Ax$ for some $A \in N$ and $x \in F^*$. Let $\sigma(v) = A$ and $\eta(v) = |x|$. $\sigma(v)$ gives the nonterminal symbol of $v$ and $\eta(v)$ gives the height of $v$’s stack. We say that a node $v'$ is in the scope of $v$ iff $v'$ is an internal node and there is a path in $D$ from $v$ to $v'$ such that for every node $v''$ on the path (including $v'$), $\eta(v'') \geq \eta(v)$. Let $\beta(v)$ be the set of nodes $v'$ such that $v'$ is in the scope of $v$ but no child of $v'$ is in the scope of $v$. The set $\beta(v)$ can be viewed as the “last” nodes in the scope of $v$. Notice that for all $v' \in \beta(v)$, $\eta(v') = \eta(v)$. Finally, we give $v$ a “type” $\tau(v)$ based on which nonterminal symbols appear in $\beta(v)$. Let $\tau(v)$ be a 3-tuple such that:

- $\tau(v)[1] = \{ A \in N \mid \text{for all } v' \in \beta(v), \sigma(v') \neq A\}$
- $\tau(v)[2] = \{ A \in N \mid \text{for some } v' \in \beta(v), \sigma(v') = A, \text{ and for all marked } v' \in \beta(v), \sigma(v') \neq A\}$
- $\tau(v)[3] = \{ A \in N \mid \text{for some marked } v' \in \beta(v), \sigma(v') = A\}$

Notice that for each $v$, $\tau(v)$ partitions $N$: every $A \in N$ occurs in exactly one of $\tau(v)[1]$, $\tau(v)[2]$, and $\tau(v)[3]$. So there are $3^{|N|}$ possible values for $\tau(v)$.

### 5.4.2 Lemma about derivation trees

In this subsection we will prove a lemma about paths in derivation trees, which we will use in the next subsection to prove our pumping lemma for indexed languages. Again let $G = (N, T, F, P, S)$ be a grounded indexed grammar.

**Lemma 5.3.** Let $H = (v_0, \ldots, v_m)$ be a path in a derivation tree $D$ from the root to a leaf (excluding the leaf) with more than $(|N| \cdot 3^{|N|})^{|N|} \cdot 3^{|N|} + 1$ branch nodes. Then there are $0 \leq b_1 < t_1 < t_2 < b_2 \leq m$ such that

- $\sigma(b_1) = \sigma(t_1)$ and $\sigma(t_2) = \sigma(b_2)$,
- $b_2$ is in $\beta(b_1)$ and $t_2$ is in $\beta(t_1)$,
- $\tau(b_1) = \tau(t_1)$, and
- there is a branch node between $b_1$ and $t_1$ or between $t_2$ and $b_2$.

**Idea.** To aid the reader’s understanding, we first give a sketch, followed below by the full proof. If $H$ is flat, i.e. if all of the nodes in $H$ have the same stack, then after $|N| \cdot 3^{|N|}$ branch nodes, there will have been two nodes with the same $\sigma$ and $\tau$, with a branch node between them. Then we can set $b_1$ and $t_1$ to these two nodes and set $t_2 = b_2 = m$, since $m$ will be in $\beta(v)$ for every node $v$ on the
path, because \( H \) is flat. If \( H \) is not flat, then consider just the “base” of \( H \), i.e. the nodes in \( H \) with the smallest stack. These nodes are separated by “hills” in which the stack is bigger. The base of \( H \) can be viewed as a flat path with gaps corresponding to the hills. Then at most \(|N| \cdot 3^{|N|}\) of the hills can contain branch nodes. We can then use an inductive argument to bound the number of branch nodes in each hill. In this argument, each hill is itself treated as a path, which is shorter than the original path \( H \) and so subject to the induction. Since node \( 0 \) and node \( m \) in \( H \) can serve as a potential \( b_1 \) and \( b_2 \) for any of the hills, each hill has fewer configurations of \( \sigma \) and \( \tau \) to “choose from” if it is to avoid containing nodes which could serve as \( t_1 \) and \( t_2 \). Working out the details of the induction gives the bound stated in the lemma.

We now give a full proof of Lemma 5.3. We will prove two supporting lemmas (Lemmas 5.4 and 5.5). Let \( D \) be a derivation tree of \( G \). We will be working with a variation of a path which we call a descent. A descent \( H \) in \( D \) is a list of internal nodes \( \langle v_0, \ldots, v_m \rangle \) with \( m \geq 0 \) such that \( v_m \) is in \( \beta(v_0) \) and for each \( i \geq 1 \), \( v_i \) is a descendant (not necessarily a child) of \( v_{i-1} \). As with paths, we will sometimes refer to nodes in \( H \) by their indices; e.g. \( \sigma(i) \) in the context of \( H \) means \( \sigma(v_i) \). For \( 0 \leq i < m \), we say there is a split in \( H \) between \( i \) and \( i+1 \) iff any of the nodes on the path in \( D \) from \( v_i \) (inclusive) to \( v_{i+1} \) (exclusive) is a branch node. If this path in \( D \) has more than one branch node, we still say that there is just one split between \( i \) and \( i+1 \) in \( H \). Thus \( H \) has at most \( m \) splits.

We call \( H \) controlled if for all \( 0 < i \leq m \), \( i \) is a child (not merely a descendant) of \( i−1 \). Notice that any path in \( D \) from the root to a leaf (excluding the leaf) is a controlled descent. We call \( H \) flat if for all \( 0 \leq i \leq m \), \( \eta(i) = \eta(0) \). We call \( H \) limited iff for all \( 0 \leq b_1 < t_1 < t_2 \leq b_2 \leq m \) such that

\[
\begin{align*}
&\sigma(b_1) = \sigma(t_1) \text{ and } \sigma(t_2) = \sigma(b_2), \\
&b_2 \text{ is in } \beta(b_1) \text{ and } t_2 \text{ is in } \beta(t_1), \text{ and} \\
&\tau(b_1) = \tau(t_1),
\end{align*}
\]

there are no splits between \( b_1 \) and \( t_1 \) or between \( t_2 \) and \( b_2 \).

**Lemma 5.4.** Let \( H = (v_0, \ldots, v_m) \) be a limited flat descent. Then \( H \) has at most \(|N| \cdot 3^{|N|}\) splits.

**Proof.** For any \( i \) in \( H \), there are \(|N|\) possible values for \( \sigma(i) \) and \( 3^{|N|} \) possible values for \( \tau(i) \). Suppose there are more than \(|N| \cdot 3^{|N|}\) splits between \( 0 \) and \( m \). Then there are \( b_1, t_1 \) such that \( 0 \leq b_1 < t_1 < m \), \( \sigma(b_1) = \sigma(t_1) \), \( \tau(b_1) = \tau(t_1) \), and there is a split between \( b_1 \) and \( t_1 \). Let \( t_2 = b_2 = m \). Obviously \( \sigma(t_2) = \sigma(b_2) \). By the definition of a descent, \( m \) is in \( \beta(0) \). Then since \( H \) is flat, \( m \) is in \( \beta(i) \) for all \( i \) in \( H \).
Hence \( b_2 \) is in \( \beta(b_1) \) and \( t_2 \) is in \( \beta(t_1) \). But then \( H \) is not limited, a contradiction. So \( H \) has at most \(|N| \cdot 3^{|N|}\) splits.

Let \( R \) be the set of possible values of \( \tau(v) \); i.e. the set of 3-tuples each of which partitions \( N \). We have \(|R| = 3^{|N|}\). For any descent \( H = (v_0, \ldots, v_m) \) and \( W \subseteq N \times N \times R \), \( H \ respects \ W \) iff for every \( 0 \leq i < j \leq m \) such that \( j \) is in \( \beta(i) \), the triple \((\sigma(i), \sigma(j), \tau(i))\) is in \( W \).

**Lemma 5.5.** Take any \( W \subseteq N \times N \times R \). Any limited controlled descent which respects \( W \) has at most \(|N| \cdot 3^{|N|}|W|^{k+1}\) splits.

**Proof.** Let \( k = |N| \cdot 3^{|N|} \), the bound from Lemma 5.4. For \( x \in N \), let \( f(x) = k^{|x|} - 2k \). We will show by induction on \(|W|\) that any limited controlled descent which respects \( W \) has at most \( f(|W|) \) splits.

If \(|W| = 0\), then no limited controlled descent respects \( W \), so the statement holds trivially. So say \(|W| \geq 1\). Suppose for induction that for every \( W' \) such that \(|W'| < |W|\), any limited controlled descent which respects \( W' \) has at most \( f(|W'|) \) splits.

Take any \( n \) such that there is a limited controlled descent which respects \( W \) and has exactly \( n \) splits. We will show that \( n \leq f(|W|) \).

Take the lowest \( m \) such that there is a limited controlled descent \( H = (v_0, \ldots, v_m) \) which respects \( W \) and has exactly \( n \) splits.

Suppose \( m = 0 \). Then \( n = 0 \). Since \(|N| \geq 1\), \( k \geq 3 \). Then since \(|W| \geq 1\) and \( f \) is increasing, \( f(|W|) \geq f(1) \geq 3^{|W|+1} - 2 \cdot 3 \geq 3 \). Then \( n \leq f(|W|) \) as desired. So say \( m \geq 1 \).

Let \( E \) be the list consisting of every node \( v_i \) in \( H \) for which \( \eta(i) = \eta(0) \). Call \( i, j \) base-adjacent iff \( i < j \), \( i \) and \( j \) are in \( E \), and no node between \( i \) and \( j \) is in \( E \). Call the interval from \( i \) to \( j \) a hill iff \( i, j \) are base-adjacent. Call a hill a split hill iff it contains at least one split.

Take any base-adjacent \( i, j \) such that the hill between \( i \) and \( j \) has at least as many splits as any hill between \( 0 \) and \( m \). Let \( x \) be the number of splits between \( i \) and \( j \). We will show \( x \leq 2 + f(|W| - 1) \). Suppose \( j = i + 1 \). Then \( x \leq 1 \). Suppose \( j = i + 2 \). Then \( x \leq 2 \). So say \( j > i + 2 \). Let \( i' = i + 1 \) and \( j' = j - 1 \). Then \( 0 < i' < j' < m \), and since \( H \) is controlled, \( \eta(i') = \eta(j') = 1 + \eta(0) \) and \( j' \) is in \( \beta(i') \).

Suppose there are no splits between \( 0 \) and \( i' \) or between \( j' \) and \( m \). Then there are \( n \) splits between \( i' \) and \( j' \). Let \( m' = j' - i' \). Then the limited controlled descent \((v_{i'}, \ldots, v_{j'})\) respects \( W \) and has \( n \) splits. But \( m' < m \), a contradiction, since by the construction of \( m \), there is no such \( m' \).

So there is such a split. Now, since \( m \) is in \( \beta(0) \), \((\sigma(0), \sigma(m), \tau(0))\) is in \( W \). Let \( W' = W - (\sigma(0), \sigma(m), \tau(0)) \). Suppose there are \( i'', j'' \) such that \( i' \leq i'' < j'' \leq j' \), \( j'' \) is in \( \beta(i'') \), \( \sigma(i'') = \sigma(0), \sigma(j'') = \sigma(m) \), and \( \tau(i'') = \tau(0) \). Let \( b_1 = 0, t_1 = i'', t_2 = j'', \) and \( b_2 = m \). Since there is a split between \( 0 \) and \( i' \) or between \( j' \) and \( m \), there is a split between \( 0 \) and \( i'' \) or between \( j'' \) and \( m \), hence between \( b_1 \) and \( t_1 \) or between \( t_2 \) and \( b_2 \). But then \( H \) is not limited, a contradiction. So there are no such \( i'', j'' \). Hence the limited controlled descent \((v_{i'}, \ldots, v_{j'})\) respects
Since \(|W'| < |W|\), by the induction hypothesis there are at most 
f(|W'|) \text{ splits between } i' \text{ and } j'. \text{ Then allowing a split between } i \text{ and } i' \text{ and a split between } j' \text{ and } j, x \leq 2 + f(|W'|).

Now, the list \(E\) is a flat descent. If \(E\) was not limited, then \(H\) would not be limited. So \(E\) is limited. Then by Lemma 5.4, it has at most \(k\) splits. Then there are at most \(k\) split hills in \(H\). Recall that \(k \geq 3\). Each split hill has at most \(x\) splits. Therefore

\[
n \leq kx
\]

\[
n \leq k(2 + f(|W| - 1))
\]

\[
n \leq 2k + k(|W|^{-1} + |W| - 2k)
\]

\[
n \leq 2k + k(|W| + 1 - 2k)
\]

\[
n \leq f(|W|),
\]

completing the induction. Therefore any limited controlled descent which respects \(W\) has at most \(|N| \cdot 3^{|N|} |W| + 1\) splits. \(\square\)

Now we can complete the proof of Lemma 5.3.

**Lemma 5.6.** Let \(H = (v_0, \ldots, v_m)\) be a path in a derivation tree \(D\) from the root to a leaf (excluding the leaf) with more than \((|N| \cdot 3^{|N|} |N|^{2\cdot 3^{|N|} + 1}\) branch nodes. \(\text{Then there are } 0 \leq b_1 < t_1 < t_2 \leq b_2 \leq m \text{ such that}\)

- \(\sigma(b_1) = \sigma(t_1) \text{ and } \sigma(t_2) = \sigma(b_2),\)
- \(b_2 \text{ is in } \beta(b_1) \text{ and } t_2 \text{ is in } \beta(t_1),\)
- \(\tau(b_1) = \tau(t_1), \text{ and}\)
- \(\text{there is a branch node between } b_1 \text{ and } t_1 \text{ or between } t_2 \text{ and } b_2.\)

**Proof.** Since \(H\) is a path from the root to a leaf, \(H\) is a controlled descent. Suppose \(H\) is limited. Let \(W = N \times N \times R\). Clearly \(H\) respects \(W\). Then by Lemma 5.5, \(H\) has at most \((|N| \cdot 3^{|N|} |N|^{2\cdot 3^{|N|} + 1}\) splits. Then since by definition, every branch node is immediately followed by a split, \(H\) has at most \((|N| \cdot 3^{|N|} |N|^{2\cdot 3^{|N|} + 1}\) branch nodes, a contradiction. So \(H\) is not limited and the lemma holds. \(\square\)

### 5.4.3 Pumping lemma for indexed languages

We are now ready to prove our pumping lemma for indexed languages (Theorem 5.7), which generalizes the pumping lemma for ETOL languages of [Rab12]. As noted, this lemma allows arbitrary positions in a word to be designated as “marked”, and then provides guarantees about the marked positions during the pumping operation. The only difference between our pumping operation and that of Theorem 15 of [Rab12] is that in the latter, there are guaranteed to
be at least two marked positions in the $v_{ij}$ of part 4, whereas in our lemma, this $v_{ij}$ might not contain any marked positions and could even be an empty string (which nonetheless maps under $\phi$ to $u_i$, which does contain a marked position).

**Theorem 5.7.** Let $L$ be an indexed language. Then there is an $l \geq 0$ (which we will call a threshold for $L$) such that for any $w \in L$ with at least $l$ marked positions,

1. $w$ can be written as $w = u_1 u_2 \cdots u_n$ and each $u_i$ can be written $u_i = v_{i,1} v_{i,2} \cdots v_{i,n_i}$ (we will denote the set of subscripts of $v$, i.e. $\{(i,j) \mid 1 \leq i \leq n \text{ and } 1 \leq j \leq n_i\}$, by $I$);

2. there is a map $\phi : I \to \{1, \ldots, n\}$ such that if each $v_{ij}$ is replaced with $u_{\phi(i,j)}$, then the resulting word is still in $L$, and this process can be applied iteratively to always yield a word in $L$;

3. if $v_{ij}$ contains a marked position then so does $u_{\phi(i,j)}$;

4. there is an $(i, j) \in I$ such that $\phi(i, j) = i$, and there is at least one marked position in $u_i$ but outside of $v_{ij}$.

**Idea.** To aid the reader’s understanding, we first give a sketch, followed by the full proof. We take a grounded indexed grammar $G = (N, T, F, P, S)$ with language $L$ and set the threshold $l$ using the bound from Lemma 5.3 together with some properties of the productions of $G$. Then we take any $w \in L$ with at least $l$ marked positions and take a derivation tree $D$ for $w$. Some path in $D$ from the root to a leaf then has enough branch nodes to give us the $b_1$, $t_1$, $t_2$, and $b_2$ from Lemma 5.3. We then need to construct the map $\phi$ and the factors $u_i$ and $v_{ij}$. To do this, we use the nodes in $\beta(b_1)$ and $\beta(t_1)$. The nodes in $\beta(b_1)$ will correspond to $v_{ij}$s and those in $\beta(t_1)$ will correspond to $u_i$s. The operation $\phi$ will then map each node in $\beta(b_1)$ to a node in $\beta(t_1)$ with the same $\sigma$, and which is marked if the node being mapped is marked. This is possible because $\tau(b_1) = \tau(t_1)$. The justification for this construction is that in $D$, between $b_1$ and $t_1$ the stack grows from $x$ to $yx$ for some $x, y \in F^*$, and then shrinks back to $x$ between $\beta(t_1)$ and $\beta(b_1)$. Since $\sigma(b_1) = \sigma(t_1)$, the steps between $b_1$ and $t_1$ can be repeated, growing the stack from $x$ to $yx$ to $yyx$, and so on. Then the $ys$ can be popped back off by repeating the steps between the nodes in $\beta(t_1)$ and $\beta(b_1)$. This construction gives us parts 1, 2, and 3 of the theorem. Part 4 follows from the fact that there is a branch node between $b_1$ and $t_1$ or between $t_2$ and $b_2$. In the former case, the $u_i$ in part 4 corresponds to the yield produced between $b_1$ and $t_1$ involving the branch node, and the $v_{ij}$ is specially constructed as an empty factor which maps to $u_i$. In the latter case, since $t_2$ is in $\beta(t_1)$, $b_2$ is in $\beta(b_1)$, and $\sigma(t_2) = \sigma(b_2)$, the $u_i$ in part 4 corresponds to $t_2$ and the $v_{ij}$ corresponds to $b_2$. $\square$
Proof. We now give a full proof of the theorem. Let \( G = (N,T,F,P,S) \) be a grounded indexed grammar such that \( L(G) = L \). Let \( d \) be the highest \( i \) such that there is a production in \( P \) with \( i \) nonterminals on the righthand side. Let \( e \) be the highest \( i \) such that there is a production in \( P \) with \( i \) terminals on the righthand side. Let \( z = (|N| \cdot 3^{|N|} |N|^2 \cdot 3^{|N|} + 1 \). Let \( l = ed^2 + 1 \).

If \( L \) is finite, then trivially the theorem holds. So say \( L \) is infinite. Let \( w \) be a word in \( L \) with at least \( l \) marked positions. Take any derivation tree \( D \) of \( w \). Suppose no path in \( D \) from the root to a leaf has more than \( z \) branch nodes. The maximum outdegree of \( D \) is at most \( d \). Then by Lemma 14 of [Rab12], \( D \) has at most \( d^2 \) marked leaves. Recall that each leaf has a label in \( T^* \) (a terminal string) and marked leaves are those whose label contains a marked position of \( w \). Each marked leaf has a label of length at most \( e \). But then \( w \) has at most \( ed^2 \) marked positions, a contradiction.

So some path \( H = (v_0, \ldots, v_m) \) in \( D \) from the root to a leaf (excluding the leaf) has more than \( z \) branch nodes. Then by Lemma 5.3, there are \( 0 \leq b_1 < t_1 < t_2 \leq b_2 \leq m \) such that \( \sigma(b_1) = \sigma(t_1) \), \( \sigma(t_2) = \sigma(b_2) \), \( b_2 \) is in \( \beta(b_1) \), \( t_2 \) is in \( \beta(t_1) \), \( \tau(b_1) = \tau(t_1) \), and there is a branch node between \( b_1 \) and \( t_1 \) or between \( t_2 \) and \( b_2 \).

We specify the subscripts of \( I \) by defining \( n \) and each \( n_i \). Let \( n = 4 + |\beta(t_1)| \). Let \( n_1 = n_n = 1 \). We will give names to the nodes in \( \beta(b_1) \) and \( \beta(t_1) \), as follows. Let \( n_2 \) be the number of nodes in \( \beta(b_1) \) to the left of \( t_1 \), plus one. From left to right, call these nodes \( N_{2,2}, N_{2,3}, \ldots, N_{2,n_2} \). Call the nodes in \( \beta(t_1) \) from left to right \( N_3, N_4, \ldots, N_{n-2} \). For each \( i \) from 3 to \( n-2 \), let \( n_i \) be the number of nodes in \( \beta(b_1) \) which are reachable from \( N_i \). From left to right, call these nodes \( N_{i,1}, N_{i,2}, \ldots, N_{i,n_i} \). Let \( n_{n-1} \) be the number of nodes in \( \beta(b_1) \) to the right of \( t_1 \), plus one. From left to right, call these nodes \( N_{n-1,1}, N_{n-1,2}, \ldots, N_{n-1,n_{n-1}-1} \).

We now define each \( u_i \) and each \( v_{ij} \), as well as the map \( \phi \) which will be used in the replacement operation. Let \( \phi(b_1) \) be the yield of \( D \) to the left of \( b_1 \), and let \( \phi(1,1) = 1 \). Let \( v_{2,1} = \lambda \) and let \( \phi(2,1) = 2 \). Let \( v_{n-1,n_{n-1}} = \lambda \) and let \( \phi(n-1,n_{n-1}) = n-1 \). Let \( v_{n,1} \) be the yield of \( D \) to the right of \( b_1 \) and let \( \phi(n,1) = n \). For all other \( i,j \) for which there is a node \( N_{ij} \), proceed as follows. Set \( v_{ij} = \text{yield}(N_{ij}) \). If \( N_{ij} \) is \( b_2 \), then set \( \phi(i,j) = i \). (Notice that \( N_i = t_2 \), so \( \sigma(N_i) = \sigma(N_{ij}) \)). Otherwise, if \( \text{yield}(N_{ij}) \) contains a marked position, then \( \sigma(N_{ij}) \) is in \( \tau(b_1)[3] \). Since \( \tau(b_1) = \tau(t_1) \), there is an \( N_k \) such that \( \sigma(N_k) = \sigma(N_{ij}) \) and \( \text{yield}(N_k) \) contains a marked position. Set \( \phi(i,j) = k \). Otherwise, \( \text{yield}(N_{ij}) \) does not contain a marked position, so \( \sigma(N_{ij}) \) is in \( \tau(b_1)[2] \). Then there is an \( N_k \) such that \( \sigma(N_k) = \sigma(N_{ij}) \). Set \( \phi(i,j) = k \). Finally, for \( i \) from 1 to \( n \), let \( u_i = v_{i,1} \cdots v_{i,n_i} \).

We now have that each \( u_i \) is the yield of the nodes in some part of \( D \). In particular, \( u_1 \) is the yield to the left of \( b_1 \), \( u_2 \) is the yield under \( b_1 \) to the left of \( t_1 \), \( u_3 \cdots u_{n-2} \) is the yield under \( t_1 \), \( u_{n-1} \) is
the yield under \( b_1 \) to the right of \( t_1 \), and \( u_n \) is the yield to the right of \( b_1 \). Thus \( u_1 \cdots u_n \) is the yield of \( D \), namely \( w \). This gives us part 1 of the theorem.

To establish part 2, we will argue that the derivation tree \( D \) of \( s \) can be “pumped” to produce new derivation trees which yield strings in accordance with the replacement operation. Let \( x \in F^* \) be the stack at node \( b_1 \). Since \( b_2 \) is in \( \beta(b_1) \), and \( b_2 \) is a descendant of \( t_1 \), the stack at node \( t_1 \) has the form \( yx \) for some \( y \in F^* \). Thus in \( D \), the stack grows from \( x \) to \( yx \) between \( b_1 \) and \( t_1 \), remains at or above \( yx \) until \( t_2 \), and then shrinks back to \( x \) at \( b_2 \). We will construct a new derivation tree \( D' \) in which the stack grows from \( x \) to \( yx \) to \( yyx \), then shrinks from \( yyx \) to \( yx \) to \( x \). The construction is as follows. Initialize \( D' \) to a copy of \( D \). Next, make a copy \( C \) of the subtree \( D(b_1) \). In \( C \), for every ancestor of any node in \( \beta(b_1) \), put \( y \) on top of its stack. For each node \( N_{i,j} \) in \( \beta(b_1) \), proceed as follows. Notice that \( N_{i,j} \) has stack \( x \) in \( D \) and hence stack \( yx \) in \( C \), while \( N_{\phi(i,j)} \) has stack \( yx \) in \( D \). Further, \( \sigma(N_{i,j}) = \sigma(N_{\phi(i,j)}) \). So in \( C \), replace the subtree \( C(N_{i,j}) \) with the subtree \( D(N_{\phi(i,j)}) \). Finally, in \( D' \), replace the subtree \( D'(t_1) \) with \( C \). The resulting derivation tree \( D' \) now obeys the rules of \( G \) and has a yield equal to the result of performing the replacement operation of part 2 on \( w \). This procedure can be repeated to produce a new tree \( D'' \) in which the stack grows to \( yyx \), with a yield corresponding to two iterations of the replacement operation of part 2, and so on. This gives us part 2 of the theorem. Part 3 follows from the construction of the \( \phi \) operation.

We now establish part 4 of the theorem. Recall that there is a branch node in \( H \) between \( b_1 \) and \( t_1 \) or between \( t_2 \) and \( b_2 \). If there is a branch node between \( b_1 \) and \( t_1 \), then by definition, this branch node has at least two marked children. Take any one of these marked children which is not on the path from node \( b_1 \) to node \( t_1 \). This child is either to the left of \( t_1 \) or to the right of \( t_1 \). If it is to the left of \( t_1 \), then some marked \( N_{2,i} \) is reachable from it for some \( 2 \leq i \leq n_2 \). Since \( N_{2,i} \) is marked, \( \text{yield}(N_{2,i}) \) contains a marked position. So \( v_{2,i} \) contains a marked position. Then since \( \phi(2, 1) = 2 \) and \( u_2 \) contains \( v_{2,1} \) and \( v_{2,i} \), the element \( (2, 1) \in I \) satisfies part 4. Similarly, if the marked child is to the right of \( t_1 \), then some marked \( N_{n-1,i} \) is reachable from it for some \( 1 \leq i \leq n_{n-1} - 1 \). Since \( N_{n-1,i} \) is marked, \( \text{yield}(N_{n-1,i}) \) contains a marked position. So \( v_{n-1,i} \) contains a marked position. Then since \( \phi(n - 1, n_{n-1}) = n - 1 \) and \( u_{n-1} \) contains \( v_{n-1,i} \) and \( v_{n-1,n-1} \), the element \( (n - 1, n - 1) \in I \) satisfies part 4. Otherwise, if there is no branch node in \( H \) between \( b_1 \) and \( t_1 \), then there is a branch node between \( t_2 \) and \( b_2 \). We have \( t_2 = N_i \) and \( b_2 = N_{i,j} \) for some \( 3 \leq i \leq n - 2 \) and \( 1 \leq j \leq n_i \). By definition, the branch node between \( t_2 \) and \( b_2 \) has at least two marked children. Take any one of these marked children which is not on the path from node \( t_2 \) to node \( b_2 \). Some marked \( N_{i,k} \) is reachable from this child where \( k \neq j \). So
yield($N_{i,k}$) contains a marked position, hence $v_{i,k}$ contains a marked position. Since $N_{i,j}$ is $b_2$, $\phi(i,j) = i$ by construction. Then since $u_i$ contains both $v_{i,j}$ and $v_{i,k}$, the element $(i,j) \in I$ satisfies part 4. This establishes part 4 of the theorem, which completes the proof.

Example 5.8. We now give an example of how our pumping operation works on a derivation tree of an indexed grammar. Let the nonterminal alphabet $N$ be \{S, X, Y, A, B\}, the terminal alphabet $T$ be \{a, b\}, the stack alphabet $F$ be \{$,$, f\}, and the set of productions $P$ be \{S $\rightarrow$ X$, X \rightarrow Xf$, X $\rightarrow$ YA, Yf $\rightarrow$ YA, Y$\rightarrow$ ab, Af $\rightarrow$ AB, A$\rightarrow$ abb, Bf $\rightarrow$ B, B$\rightarrow$ b\}. Let G be the indexed grammar ($N, T, F, P, S$). Notice that G is grounded and that $L(G)$ determines the infinite word $ab^1ab^2ab^3\cdots$. Figure 5.1 depicts a derivation tree of G with terminal yield $ab^1ab^2ab^3ab^4$. Notice how stacks are copied as the derivation proceeds; for example, the production $X \rightarrow YA$ applied to $Xff$ copies X’s stack $ff$ to both Y and A, yielding $Yff$ Aff$.

Figure 5.1: A derivation tree for the string $ab^1ab^2ab^3ab^4$. 

Take every position in the terminal yield of the tree to be marked. Let $b_1$ and $t_1$ be the nodes labelled X$, and Xf$, respectively, and let $t_2$ and $b_2$ be the nodes labelled Af$, and A$, respectively, in the Yff$ subtree. We have $\sigma(b_1) = \sigma(t_1) = X$, $\sigma(t_2) = \sigma(b_2) = A$, and $\tau(b_1) = \tau(t_1) = [(S, X), [], (Y, A, B)]$. Additionally, $b_2$ is in $\beta(b_1)$, which consists of all the nodes labelled A$, B$, or Y$, and $t_2$ is in $\beta(t_1)$, which consists of all the nodes labelled Af$, Bf$, or Yf$. Also, there
is a branch node between \( t_2 \) and \( b_2 \), namely \( t_2 \) itself. This satisfies the conditions of Lemma 5.3. We break up the terminal yield as

\[
\overline{ab\, abb} \overline{abb\, b\, abbb\, b},
\]

where the outer brackets delimit the \( u_i \)'s and the inner brackets delimit the \( v_{i,j} \)'s. Set \( \phi(1,1) = 1, \phi(1,2) = 2, \phi(2,1) = 2, \phi(2,2) = 4, \phi(3,1) = 2, \phi(3,2) = 4, \) and \( \phi(4,1) = 4 \). Applying the pumping operation yields

\[
\overline{ab\, abb} \overline{abb\, b\, abbb\, b\, b\, b} = ab^1 ab^2 ab^3 ab^4 ab^5,
\]

which is indeed in \( L(G) \). Applying it again yields

\[
ab^1 ab^2 ab^3 ab^4 ab^5 ab^6,
\]

and so on.

We now follow [Rab12] in giving a more formal description of the replacement operation in part 2 of Theorem 5.7. This operation produces the words \( w^{(t)} \) for all \( t \geq 0 \), where

\[
\begin{align*}
\nu_{i,j}^{(0)} &= v_{i,j} \\
u_{i,j}^{(t)} &= \nu_{i,j}^{(t-1)} v_{i,j}^{(t)} \\
u_{i,j}^{(t+1)} &= \nu_{i,j}^{(t)} \\
u_{i,j}^{(t+1)} &= \nu_{i,j}^{(t)} \\
u_{i,j}^{(t+1)} &= \nu_{i,j}^{(t)} \\
u_{i,j}^{(t+1)} &= \nu_{i,j}^{(t)} \\
u_{i,j}^{(t+1)} &= \nu_{i,j}^{(t)} \\
u_{i,j}^{(t+1)} &= \nu_{i,j}^{(t)} \\
u_{i,j}^{(t+1)} &= \nu_{i,j}^{(t)} \\
u_{i,j}^{(t+1)} &= \nu_{i,j}^{(t)} \\
u_{i,j}^{(t+1)} &= \nu_{i,j}^{(t)} \\
u_{i,j}^{(t+1)} &= \nu_{i,j}^{(t)} \\
u_{i,j}^{(t+1)} &= \nu_{i,j}^{(t)} \\
u_{i,j}^{(t+1)} &= \nu_{i,j}^{(t)} \\
u_{i,j}^{(t+1)} &= \nu_{i,j}^{(t)} \\
u_{i,j}^{(t+1)} &= \nu_{i,j}^{(t)} \end{align*}
\]

Notice that \( w^{(0)} = w \). The following lemma states that the number of marked symbols tends to infinity as the replacement operation is repeatedly applied.

**Lemma 5.9.** If \( L \) is an indexed language with threshold 1, and \( w \in L \) has at least \( l \) marked symbols, then for all \( t \geq 0 \), \( w^{(t)} \) has at least \( t \) marked symbols.

**Proof.** Call each \( v_{i,j} \) a \( v \)-word and call a \( v \)-word marked if it contains a marked position. We will show by induction on \( t \) that for all \( t \geq 0 \), \( w^{(t)} \) contains at least \( t \) occurrences of marked \( v \)-words. Obviously the statement holds for \( t = 0 \). So say \( t \geq 1 \) and suppose for induction that \( w^{(t-1)} \) contains at least \( t - 1 \) occurrences of marked \( v \)-words. By part 3 of Theorem 5.7, for every marked \( v_{i,j} \) in \( w^{(t-1)} \), \( v_{i,j}^{(t)} \) contains a marked position. Then \( w^{(t)} \) contains at least \( t - 1 \) occurrences of marked \( v \)-words. Now by part 4 of the theorem, there is an \( (i,j) \in I \) such that \( \phi(i,j) = i \) and there is at least one marked position in \( u_i \) but outside of \( v_{i,j} \). Then \( v_{i,j}^{(t)} = u_i \) contains at least one more occurrence of a marked \( v \)-word than \( v_{i,j} \). Now since \( w \) contains \( v_{i,j} \), \( w^{(t-1)} \) contains \( v_{i,j} \). Then \( w^{(t)} \) contains at least one more occurrence of a marked \( v \)-word than \( w^{(t-1)} \). So \( w^{(t)} \) contains at least \( t \) occurrences of marked \( v \)-words, completing the induction. Thus for all \( t \geq 0 \), \( w^{(t)} \) contains at least \( t \) occurrences of marked \( v \)-words, hence \( w^{(t)} \) contains at least \( t \) marked positions. \( \square \)
5.5 APPLICATIONS

In this section we give some applications of our pumping lemma for indexed languages (Theorem 5.7). We prove for IL a theorem about frequent and rare symbols which is proved in [Rab12] for ET0L languages. Then we characterize the infinite words determined by indexed languages.

5.5.1 Frequent and rare symbols

Let $L$ be a language over an alphabet $A$, and $B \subseteq A$. $B$ is nonfrequent if there is a constant $c_B$ such that $\#_B(w) \leq c_B$ for all $w \in L$. Otherwise it is called frequent. $B$ is called rare if for every $k \geq 1$, there is an $n_k \geq 1$ such that for all $w \in L$, if $\#_B(w) \geq n_k$ then the distance between any two appearances in $w$ of symbols from $B$ is at least $k$.

**Theorem 5.10.** Let $L$ be an indexed language over an alphabet $A$, and $B \subseteq A$. If $B$ is rare in $L$, then $B$ is nonfrequent in $L$.

**Proof.** Suppose $B$ is rare and frequent in $L$. By Theorem 5.7, $L$ has a threshold $l \geq 0$. Since $B$ is frequent in $L$, there is a $w \in L$ with more than $l$ symbols from $B$. If we mark all of the symbols from $B$ in $w$, parts 1 to 4 of the theorem apply. By part 3 of the theorem, if $v_{t,j}$ contains a marked position then so does $u_{\phi(i,j)}$, and by part 4 of the theorem, there is an $(i,j) \in I$ such that $\phi(i,j) = i$ and there is at least one marked position in $u_i$ but outside of $v_{t,j}$. Then $u_i^{(1)}$ contains at least two marked positions. So take any two marked positions in $u_i^{(1)}$ and let $d$ be the distance between them. Let $k = d + 1$. Since $B$ is rare in $L$, there is an $n_k \geq 1$ such that for all $w' \in L$, if $\#_B(w') \geq n_k$ then the distance between any two appearances in $w'$ of symbols from $B$ is at least $k$. By Lemma 5.9, $w^{(n_k)}$ contains at least $n_k$ marked symbols, so $\#_B(w^{(n_k)}) \geq n_k$. Then the distance between any two appearances in $w^{(n_k)}$ of symbols from $B$ is at least $k$. Since $u_i$ appears in $w$ and $u_i^{(1)}$ contains $u_i$, $u_i^{(1)}$ appears in $w^{(t)}$ for all $t \geq 1$. Then $u_i^{(1)}$ appears in $w^{(n_k)}$ and contains two symbols from $B$ separated by $d < k$, a contradiction. So if $B$ is rare in $L$, then $B$ is nonfrequent in $L$. □

Theorem 5.10 gives us an alternative proof of the result of Hayashi [Hay73] and Gilman [Gil96] that the language $I$ below is not indexed.

**Corollary 5.11.** ([Hay73] Theorem 5.3; [Gil96] Corollary 4). The language $L = \{(ab^n)^n \mid n \geq 1\}$ is not indexed.

**Proof.** The subset $(a)$ of $(a,b)$ is rare and frequent in $L$. So by Theorem 5.10, $L$ is not indexed. □
5.5.2 CD0L and morphic words

Next, we turn to characterizing the infinite words determined by indexed languages. We show that every infinite indexed language has an infinite CD0L subset, which then implies that \(\omega(\text{IL})\) contains exactly the morphic words. This is a new result which we were not able to obtain using the pumping lemmas of [Hay73] or [Gil96].

**Theorem 5.12.** Let \(L\) be an infinite indexed language. Then \(L\) has an infinite CD0L subset.

**Proof.** By Theorem 5.7, \(L\) has a threshold \(l \geq 0\). Take any \(w \in L\) such that \(|w| \geq l\), and mark every position in \(w\). Then parts 1 to 4 of the theorem apply. Now for each \((i, j) \in I\), create a new symbol \(x_{i,j}\). Let \(X\) be the set of these symbols. For \(i\) from 1 to \(n\), let \(x_i = x_{i,1}x_{i,2} \cdots x_{i,n_i}\). Let \(x = x_1x_2 \cdots x_n\). Let \(h\) be a morphism such that \(h(x_{i,j}) = x_{\phi(i,j)}\) for all \((i, j) \in I\). Let \(g\) be a morphism such that \(g(x_{i,j}) = v_{i,j}\) for all \((i, j) \in I\). Let \(A\) be the alphabet of \(L\). Let \(G\) be the HD0L system \((X \cup A, h, x, g)\). Then for all \(t \geq 0\), \(g(h^t(x)) = w^{(t)}\). By Lemma 5.9, for all \(t \geq 0, |w^{(t)}| \geq t\). Then \(L(G)\) is an infinite HD0L subset of \(L\). By Theorem 18 of [Smi13b], every infinite HD0L language has an infinite CD0L subset. Therefore \(L\) has an infinite CD0L subset.

**Theorem 5.13.** \(\omega(\text{IL})\) contains exactly the morphic words.

**Proof.** For any infinite word \(\alpha \in \omega(\text{IL})\), some \(L \in \text{IL}\) determines \(\alpha\). Then \(L\) is an infinite indexed language, so by Theorem 5.12, \(L\) has an infinite CD0L subset \(L'\). Then \(L'\) determines \(\alpha\), so \(\alpha\) is in \(\omega\)(CD0L). Then by Theorem 23 of [Smi13b], \(\alpha\) is morphic. For the other direction, by Theorem 23 of [Smi13b], every morphic word is in \(\omega\)(CD0L), so since CD0L \(\subset\) IL, every morphic word is in \(\omega\)(IL).

Theorem 5.13 lets us use existing results about morphic words to show that certain languages are not indexed, as the following example shows.

**Corollary 5.14.** Let \(L\) be the language

\[
\{0, 0:1, 0:1:01, 0:1:01:11, 0:1:01:11:001, \ldots\}
\]

containing for each \(n \geq 0\) a word with the natural numbers up to \(n\) written in backwards binary and colon-separated. Then \(L\) is not indexed.

**Proof.** The language \(L\) determines the infinite word

\[
\alpha = 0:1:01:11:001:101:011:111:\ldots
\]

By Theorem 3 of [CK94], \(\alpha\) is not morphic. Then by our Theorem 5.13, \(\alpha\) is not in \(\omega\)(IL), so no language in IL determines \(\alpha\), hence \(L\) is not indexed.
5.6 Conclusion

In this chapter we have characterized the infinite words determined by indexed languages, showing that they are exactly the morphic words. In doing so, we proved a new pumping lemma for the indexed languages, which may be of independent interest and which we hope will have further applications. One direction for future work is to look for more connections between formal languages and morphic words via the notion of prefix languages. It would be interesting to see what other language classes determine the morphic words, and what language classes are required to determine infinite words that are not morphic.
In this chapter, we extend our investigation of infinite words from determination to prediction. To do so, we formulate the notion of a prediction language. Whereas a prefix language determines a single infinite word, a prediction language masters, or learns in the limit, a set of infinite words. Within this framework we study the predictive capabilities of different types of automata, relating their prediction languages to periodic and multilinear words. We then discuss the number of guesses needed to master infinite words with descriptions of certain sizes, and study the related question of how many symbols two distinct infinite words of a certain description size can agree on before differing.

6.1 Introduction

One motivation for studying prediction of infinite words comes from its position as a kind of underlying “simplest case” of other prediction tasks. For example, take the problem of designing an intelligent agent, a purposeful autonomous entity able to explore and interact with its environment. At each moment, it receives data from its sensors, which it stores in its memory. We would like the agent to analyze the data it is receiving, so that it can make predictions about future data and carry out actions in the world on the basis of those predictions. That is, we would like the agent to discover the laws of nature governing its environment.

Without any constraints on the problem, this is a formidable task. The data being received by the agent might be present in multiple channels, corresponding to sight, hearing, touch, and other senses, and in each channel the data given at each instant could have a complex structure, e.g. a visual field or tactile array. The data source could be nondeterministic or probabilistic, and furthermore could be sensitive to actions taken by the agent, leading to a feedback loop between the agent and its environment. The laws governing the environment could be mathematical in nature or arise from intensive computational processing.

A natural approach to tackling such a complex problem is to start with the easiest case. How, then, can we simplify the above scenario? First, let’s say that instead of receiving data through multiple channels, the agent has only a single channel of data. And let’s say that instead of the data having a complex structure like a visual field, it simply consists of a succession of symbols, and that the set of pos-
sible symbols is finite. Let’s say that the data source is completely
deterministic, and moreover that the data is not sensitive to the ac-
tions or predictions of the agent, but is simply output one symbol at
time without depending on any input.

Under these simplifying assumptions, the problem we are left with
is that of predicting an infinite word. That is, the agent’s environment
now consists of some infinite word, which it is the agent’s task to pre-
dict on the basis of the symbols it has seen so far. Even this is a diffi-
cult problem. But we hope that by exploring and making progress in
this simple setting, we can develop techniques which may help with
the more general prediction problems encountered in the original sce-
nario.

The prediction scenario which we would like to model is there-
fore as follows. We have an “emitter” (generator) and a “predictor”
guesser). The emitter takes no input, but just outputs symbols one at
time. The predictor receives each symbol output by the emitter, and
tries to guess the next symbol before it appears. We say that the pre-
dictor “masters” the emitter if there is a point after which all of the
predictor’s guesses are correct. We would like to design a predictor
which masters as broad a class of emitters as possible, as efficiently
as possible.

6.1.1 Framework

To proceed, we formulate the setup in terms of language theory. The
sequence of symbols output by the emitter constitutes an infinite
word, and the predictor can be modeled as a language which speci-
fies which predictions to make in response to the emitter. Let A be an
alphabet. A prediction language is a language L such that for every
s ∈ A*, there is exactly one c ∈ A such that L contains sc. The symbol
c can be viewed as the predictor’s guess in response to an emitter’s
output s. L therefore specifies the predictor’s behavior against every
possible emitter.

Let α be an infinite word. A prediction language L masters α if
there is an n ≥ 0 such that for every prefix p of α such that |p| > n,
L contains p. The number n can be viewed as the point after which
all of the predictor’s guesses against an emitter of α are correct. Let
M(L) (called the “mastery” of L) be the set of infinite words mastered
by L.

With this framework in place, we can ask, for two language classes
C₁ and C₂, whether there is any prediction language L ∈ C₁ such
that ω(C₂) ⊆ M(L). Such a language can be viewed as a predictor
from the class C₁ capable of mastering any emitter from the class C₂.
We will seek answers to this question for various language classes,
aiming to find efficient predictors for broad classes of infinite words.
Below and for the rest of the chapter, all strings, languages, and infi-
finite words are considered to share a common background alphabet $A$. Two basic negative results limit what we can achieve.

**Theorem 6.1.** For $|A| \geq 2$, no prediction language masters every infinite word.

**Proof.** Take any prediction language $L$. Let $\alpha$ be an infinite word such that for $i \geq 1$, the $i$th symbol of $\alpha$ is defined recursively as follows. Let $s$ be the prefix of $\alpha$ of length $i-1$. Since $L$ is a prediction language, there is exactly one $c \in A$ such that $L$ contains $sc$. Then since $|A| \geq 2$, there is a $d \in A$ such that $L$ does not contain $sd$. Set the $i$th symbol of $\alpha$ to $d$. Then $L$ does not contain any non-empty prefix of $\alpha$. So $L$ does not master $\alpha$. \hfill $\square$

**Theorem 6.2.** For $|A| \geq 2$, for every prediction language $L$, there is a prediction language which masters every infinite word which $L$ masters, and also masters an infinite word which $L$ does not master.

**Proof.** By Theorem 6.1, there is an infinite word $\alpha$ which $L$ does not master. Let $S$ be the set of prefixes of $\alpha$. Let $T = \{sc \in L \mid s \text{ is a prefix of } \alpha \text{ and } c \text{ is in } A\}$. Let $L' = (L \setminus T) \cup S$.

To see that $L'$ is a prediction language, take any $s \in A^*$. Call a string of the form $sc$ with $c \in A$ an increment of $s$. The prediction language $L$ contains exactly one increment of $s$. If $s$ is a prefix of $\alpha$, then $(L \setminus T)$ contains no increment of $s$, while $S$ contains exactly one increment of $s$, and therefore $L'$ contains exactly one increment of $s$. If $s$ is not a prefix of $\alpha$, then $(L \setminus T)$ contains exactly one increment of $s$, while $S$ contains no increment of $s$, and therefore $L'$ again contains exactly one increment of $s$. So $L'$ is a prediction language.

Clearly $L'$ masters $\alpha$. To see that $L'$ masters every infinite word which $L$ masters, take any such word $\beta$. There is an $n \geq 0$ such that for every prefix $p$ of $\beta$ such that $|p| > n$, $L$ contains $p$. Since $\alpha \neq \beta$, there is an $m \geq 1$ such that $\alpha[m] \neq \beta[m]$. Take any prefix $p$ of $\beta$ such that $|p| > \max(m, n)$. Write $p$ as $sc$ where $c$ is in $A$. Then $p$ is in $L$, but since $s$ is not a prefix of $\alpha$, $p$ is not in $T$. Hence $p$ is in $L'$, and so $L'$ masters $\beta$.

Hence $L'$ is a prediction language which masters every infinite word which $L$ masters, and also masters an infinite word which $L$ does not master. \hfill $\square$

Theorem 6.1 can be understood as saying that there is no “perfect” predictor, while Theorem 6.2 can be understood as saying that there is no “best” predictor. With these negative results in mind, the most we can hope to achieve is to design a “good” predictor, i.e., one which efficiently masters a broad class of emitters. In this direction, we will begin with simple classes of infinite words, design prediction languages to master those, and move on to broader classes.
We hope that the results of this investigation may have applications in prediction of symbolic times series or other data streams. For example, if it is known that a particular real-world emitter has a certain structure, e.g. that of some type of automaton, then our framework could indicate what type of device (e.g. some more powerful automaton) is needed to predict the emitter’s output, and provide an algorithm for doing so based on the corresponding prediction language. Similarly, if it is known that a real-world predictor has a structure related to the language classes we study, then our framework could characterize the set of infinite words capable of being mastered by that predictor.

6.1.2 Related work

Our concept of “mastering” an infinite word is a form of “learning in the limit”, a concept which originates with the seminal paper of Gold [Gol67], where it is applied to language learnability. An early work on prediction of periodic sequences is [Shu71], where these sequences appear in the setting of two-player emission-prediction games. Inference of ultimately periodic sequences is treated in [Joh88], where an algorithm is presented which, in our terminology, masters every ultimately periodic word using a number of guesses linear in the preperiod and period lengths, and additionally can be implemented in time and space linear in the size of the input string, using techniques from string matching. The algorithm works by finding the LRS (longest repeated suffix) of the input and predicting the symbol which followed that suffix on its previous occurrence.

In [O’C88], finite-state automata are considered as predicting machines and the question of which sequences appear “random” to these machines is answered. A binary sequence is said to appear random to a predicting machine if no more than half of the predictions made of the sequence’s terms by that machine are correct. Further work on this concept appears in [BL92]. In [FMG92] the finite-state predictability of an infinite sequence is defined as the minimum fraction of prediction errors that can be made by a finite-state predictor, and it is proved that finite-state predictability can be obtained by an efficient prediction procedure using techniques from data compression. In [Bla95] a random prediction method for binary sequences is given which ensures that the proportion of correct predictions approaches the frequency of the more common symbol (0 or 1) in the sequence. In [LLA98], “inverse problems” for D0L systems are discussed (in the title and throughout the paper, the term “finite automata” refers to morphisms). These problems ask, given a word, to find a morphism and initial string which generate that word (bounds are assumed on the size of the morphism and initial string). An approach is given for solving this problem by trying different string
lengths for the righthand side of the morphism until a combination is found which is compatible with the input. A genetic algorithm is described to search the space of word lengths. In [CGL02], an evolutionary algorithm is used to search for the finite-state machine with the highest prediction ratio for a given purely periodic word, in the space of all automata with a fixed number of states. In [Dru13], the problem of successfully predicting a single 0 in an infinite binary word being revealed sequentially to the predictor is considered; only one prediction may be made, but at a time of the predictor’s choosing.

An early and influential approach to predicting infinite sequences is that of program-size complexity [Sol64]. Unfortunately this model is incomputable, and in [Leg06] it is shown furthermore that some sequences can only be predicted by very complex predictors which cannot be discovered mathematically due to problems of Gödel incompleteness. [Leg06] concludes that “perhaps the only reasonable solution would be to add additional restrictions to both the algorithms which generate the sequences to be predicted, and to the predictors.” This suggestion is akin to the approach followed in the present chapter, where the prediction languages and infinite words considered are of various restricted classes. Following on from [Leg06], in [Hib08] the formalism of sequence prediction is extended to a competition between two agents, which is shown to be a computational resources arms race.

In [CS82] two ways are given of associating a language $L$ with a set of infinite words: the “adherence” of $L$ and the “limit” of $L$. The adherence of $L$ (see also Section 4.6.2) is the set $\{\alpha \mid \alpha \text{ is an infinite word and for every prefix } p \text{ of } \alpha, \text{ there is an } s \in L \text{ such that } p \text{ is a prefix of } s\}$. The limit of $L$ is the set $\{\alpha \mid \alpha \text{ is an infinite word and for every } k \geq 0, \text{ there is an } s \in L \text{ such that } s \text{ is a prefix of } \alpha \text{ and } |s| > k\}$. Where $L$ is a prediction language, a third way is our notion of the “mastery” of $L$, or set of infinite words mastered by $L$. The mastery of $L$ is a subset of the limit of $L$ (which is in turn a subset of the adherence of $L$). For an example in which the containment is proper, take the prediction language $L = (a|b)^*a$, which masters every infinite word ending in $a^\omega$. The infinite word $(ab)^\omega$ belongs to the limit and adherence of $L$, but not the mastery.

6.1.3 Outline of chapter

The rest of the chapter is organized as follows. Section 6.2 considers several types of infinite words with respect to which classes of prediction languages can master them, focusing on prediction languages recognized by different types of automata. Section 6.3 considers the number of “guesses” required to reach mastery and the related topic of maximum agreement of distinct infinite words. Section 6.4 gives our conclusions.
In this section we consider several types of infinite words with respect to which classes of prediction languages can master them, focusing on prediction languages recognized by different types of automata.

6.2.1 Prediction of purely periodic words

We begin with the purely periodic words, those of the form \( x^\omega \) for some string \( x \neq \lambda \).

**Theorem 6.3.** For \( |\Lambda| \geq 2 \), no regular prediction language masters every purely periodic word.

**Proof.** Suppose some regular prediction language \( L \) masters every purely periodic word. Let equivalence relation \( R_L \) be defined by: \( x R_L y \) iff for every string \( z \), \( xz \) is in \( L \) iff \( yz \) is in \( L \). By the Myhill-Nerode theorem, \( R_L \) is of finite index. Let \( p \) be the number of equivalence classes of \( R_L \). Take the purely periodic word \( \alpha = (a^p b)^\omega \). Since \( L \) masters \( \alpha \), there is an \( m \geq 0 \) such that for every prefix \( s \) of \( \alpha \) such that \( |s| > m \), \( L \) contains \( s \). Let \( u = (a^p b)^m \). Then for some \( 0 \leq i < j \leq p \), \( ua^i R_L ua^j \). Since \( |u| > m \), \( ua^p b \) is in \( L \), hence \( ua^i a^{p-j} b \) is in \( L \), hence \( ua^i a^{p-j} a \) is not in \( L \), a contradiction, since it is a prefix of \( \alpha \) of length greater than \( m \). Therefore \( L \) does not master \( \alpha \). \( \square \)

**Theorem 6.4.** Some 1-DNESA prediction language masters every purely periodic word.

**Proof.** Let \( M \) be a one-way deterministic nonerasing stack automaton (1-DNESA), operating as follows. \( M \) pushes the first symbol of the input onto its stack. Then \( M \) reads its stack from the bottom up and moves along the input, checking that each stack symbol matches each input symbol. Eventually it reaches the top of the stack; call this one “pass” over the input. If there were no mismatches during the pass, \( M \) returns to the bottom of the stack (without advancing the input) and does another pass, and so on. If there is a mismatch during the pass, \( M \) completes the pass, then continues doing passes until one succeeds with no mismatches. Then \( M \) pushes the next input symbol onto the stack, and proceeds as before. When \( M \) reaches the end of the input, it accepts if the last input symbol was a match, and rejects otherwise. (If the last input symbol was pushed onto the stack, \( M \) accepts only if it is, say, the first symbol in the alphabet.)

To see that \( L(M) \) is a prediction language, write \( M \)’s input in the form \( sc \) with \( c \in A \). Observe that \( M \) accepts \( sc \) only if \( c \) matches a particular symbol \( d \in A \) determined as a function of \( s \). Since \( s \) determines exactly one such \( d \), \( L(M) \) is a prediction language.
Now, take any purely periodic word $\alpha = x^\omega$. Initially, $M$ pushes the first symbol of $x$ onto its stack. Subsequently, $M$ only pushes a symbol onto its stack after a successful pass. With just one symbol on the stack, there will eventually be a successful pass, if the input is long enough to contain more than one copy of $x$. Suppose some successful pass is followed by an unsuccessful pass. Let $i$ be the position of the input head in $x$ (counting from zero, so $0 \leq i < |x|$) at the beginning of the successful pass and let $h$ be the height of the stack. Then the position of the input head in $x$ after the successful pass is $(i + h) \mod |x|$. Then after $|x| - 1$ unsuccessful passes, the position of the input head in $x$ will be $(i + h|x|) \mod |x| = i$. So the next pass after that will be successful. Hence every unsuccessful pass will eventually be followed by a successful pass which pushes a symbol onto the stack. Suppose the height of the stack never reaches $|x|$. Then after some point, every pass is successful, so $L(M)$ masters $\alpha$. So say the height of the stack eventually reaches $|x|$. Then since the last pass before the stack reached that height was successful, and the input symbol following that pass is now at the top of the stack, the previous $|x|$ symbols of the input match the stack. Then every subsequent pass will be successful, and $L(M)$ masters $\alpha$. \hfill $\square$

6.2.2 Prediction of ultimately periodic words

We now consider prediction of ultimately periodic words, those of the form $xy^\omega$, where $x$ and $y$ are strings and $y \neq \lambda$.

**Theorem 6.5.** Some 1:multi-DFA prediction language masters every ultimately periodic word.

**Proof.** We employ a variation of the “tortoise and hare” cycle detection algorithm [Knu81], adapted to our setting. Let $M$ be a one-way 2-head deterministic finite automaton (1:2-DFA), operating as follows. $M$ starts with both heads on the first input symbol. To begin, $M$ moves one head one square to the right; call this the right head and the other the left head. Subsequently, $M$ compares the input symbols under its two heads. If they match, $M$ moves each head one square to the right. Otherwise, $M$ moves the left head one square to the right and the right head two squares to the right, and proceeds as before. If the last input symbol is a match, $M$ accepts. If the last input symbol is a mismatch, $M$ rejects. If the last input symbol was passed over by the right head following a mismatch, $M$ accepts only if this symbol is, say, the first symbol in the alphabet.

To see that $L(M)$ is a prediction language, write $M$’s input in the form $sc$ with $c \in A$. Observe that $M$ accepts $sc$ only if $c$ matches a particular symbol $d \in A$ determined as a function of $s$. Since $s$ determines exactly one such $d$, $L(M)$ is a prediction language.

Now, take any ultimately periodic word $\alpha = xy^\omega$. Since both heads move to the right, both heads will eventually reach the periodic part
y\(^\omega\) of \(\alpha\), if the input is long enough. Now after each mismatch, the distance between the two heads increases by 1. If this distance stops growing, then there are no more mismatches, so \(L(M)\) masters \(\alpha\). Otherwise, the distance between the heads in the periodic part will reach a multiple of \(|y|\). But then each head will point to the same position in \(y\) as the other, so they will match from that point on. So again \(L(M)\) masters \(\alpha\).

6.2.3 Prediction of multilinear words

We turn now to prediction of multilinear infinite words. Recall that these words properly include the ultimately periodic words and have 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 \(a_i + b_i > 0\).

We are interested in the question of whether or not some 1:multi-DFA prediction language masters every multilinear word. We do not know the answer to this question, but we present two partial results. First, we define a subset of the multilinear words which we call unitary multilinear words, and show that some 1:multi-DFA prediction language masters every unitary multilinear word. Then we define another subset of the multilinear words which we call c-delimited multilinear words, and show that for each \(c \in \mathcal{A}\), some 1:multi-DFA prediction language masters every c-delimited multilinear word.

We say that a multilinear word is **unitary** if it can be written such that for all \(1 \leq i \leq m\), \(|r_i| = 1\). The unitary multilinear words include the ultimately periodic words (just break the period into individual symbols), as well as non-periodic words like

\[
\prod_{n \geq 0} a^{n+1} b = abaabaab\cdots
\]

Clearly the unitary multilinear words are a proper subset of the multilinear words.

**Theorem 6.6.** Some 1:multi-DFA prediction language masters every unitary multilinear infinite word.

**Proof.** Let \(M\) be a one-way 3-head deterministic finite automaton (1:3-DFA), operating as follows. \(M\) starts with all three heads on the first input symbol. Call the three heads \(h_1\), \(h_2\), and \(h_3\); we will use these labels also to refer to the positions of the heads in the input string. We say that two positions agree if they contain the same symbol and disagree otherwise. Whenever \(M\) moves \(h_3\), provided end of input...
is not reached, $M$ compares the symbol at the new position of $h_3$ to another symbol (specified below); we call this a guess. If the two symbols are equal, we say that the guess is correct; otherwise we say it is incorrect. When $h_3$ reaches end of input, $M$ accepts iff its last guess was correct.

To begin, $M$ moves $h_2$ and $h_3$ together until $h_3$ disagrees with $h_3 - 1$. Then $M$ leaves $h_2$ in place and moves $h_3$ until $h_3$ disagrees with $h_3 - 1$. During this process, whenever $h_3$ moves to a new input position, $M$'s guess is the symbol at the previous position. The input can be broken into segments, each of which consists of a repeated string of the same symbol, such that this symbol differs between adjacent segments. Now $h_1$ is at the start of segment 1, $h_2$ is at the start of segment 2, and $h_3$ is at the start of segment 3. Throughout the computation, the number of segments between $h_1$ and $h_2$ (that is, the number of symbol switches on the interval from $h_1$ to $h_2$) will equal the number of segments between $h_2$ and $h_3$.

From now on, $M$ will alternate between two procedures, a matching procedure and a correction procedure. The matching procedure is a loop in which $M$ moves the three heads together, with occasional adjustments to handle increasing segment length. If $M$ detects a problem due to unexpected disagreement among the heads, then $M$ exits the matching procedure. In the correction procedure, $M$ aligns the heads at the start of segments, increases the number of segments between the heads by one, and then re-enters the matching procedure.

**Matching Procedure** The details of the matching procedure are as follows. Whenever the matching procedure is entered, each of the three heads is at the start of a segment. If the three heads do not all agree with one another when the matching procedure is entered, then $M$ moves each of them forward one step and exits the matching procedure. (Moving the heads forward one step ensures that they will be advanced to the next segment by the correction procedure.) Otherwise, they do agree, so each of the three heads is reading a segment consisting of repetitions of some symbol $c$. $M$ now moves all three heads together for as long as they all agree with one another. When this is no longer the case, $M$ exits the matching procedure unless $h_2$ and $h_3$ are reading $c$. If $h_2$ and $h_3$ are reading $c$, then $h_1$ is reading a symbol other than $c$ and has therefore reached the start of the next segment. Continuing the matching procedure, $M$ now moves $h_2$ one step at a time and $h_3$ two steps at a time, until they reach a symbol other than $c$. If $h_2$ reaches such a symbol before $h_3$ or vice versa, $M$ exits the matching procedure. Otherwise, $h_2$ and $h_3$ reach a symbol other than $c$ together, and now all three heads are at the start of segments. $M$ then returns to the start of the matching procedure. During the matching procedure, whenever $h_3$ moves to a new input position, $M$'s guess is the symbol at $h_2$. 
correction procedure The details of the correction procedure are as follows. First, M moves $h_1$, $h_2$, and $h_3$ to the start of segments (if they are not already at the start of segments) by advancing $h_1$ until it disagrees with $h_1 - 1$, advancing $h_2$ until it disagrees with $h_2 - 1$, and advancing $h_3$ until it disagrees with $h_3 - 1$. (M can use its state to record the symbol immediately preceding each head.) Then M moves $h_2$ forward one segment by first advancing it one step, and then advancing it until it disagrees with $h_2 - 1$. Finally M moves $h_3$ forward two segments by first advancing it one step, then advancing it until it disagrees with $h_3 - 1$, and then doing those two things again. Now the number of segments between $h_1$ and $h_2$ has increased by one, as has the number of segments between $h_2$ and $h_3$. M then exits the correction procedure and restarts the matching procedure. During the correction procedure, whenever $h_3$ moves to a new input position, M’s guess is the symbol at the previous position.

analysis We first show that whenever the matching procedure is entered, the number of segments between $h_1$ and $h_2$ equals the number of segments between $h_2$ and $h_3$. Clearly this is the case the first time the matching procedure is entered. Now suppose that the matching procedure is exited after having been entered with the number of segments between $h_1$ and $h_2$ equal to the number of segments between $h_2$ and $h_3$. If the heads did not all agree with one another when the matching procedure was entered, then the matching procedure advanced each of them one step, and now the correction procedure will move them to the start of the next segment beyond the one they were at when they entered the matching procedure. At that point the number of segments between $h_1$ and $h_2$ again equals the number of segments between $h_2$ and $h_3$. Otherwise, if the heads all agreed with one another when the matching procedure was entered, then the heads must have ceased to all agree with one another during the matching procedure. When they last all agreed with one another, $h_1$ was at some segment $i$, $h_2$ was at some segment $j$, and $h_3$ was at some segment $k$, with $j - i = k - j$. Since then, no head has passed the start of the next segment. Therefore when the correction procedure is entered, it will move $h_1$ to the start of segment $i + 1$, $h_2$ to the start of segment $j + 1$, and $h_3$ to the start of segment $k + 1$. At that point the number of segments between $h_1$ and $h_2$ again equals the number of segments between $h_2$ and $h_3$. Since the remainder of the correction procedure preserves this equality, this will be the case each time the matching procedure is entered.

To see that $L(M)$ is a prediction language, write M’s input in the form $sc$ with $c \in A$. When $h_3$ reaches $c$, M will guess some symbol $d \in A$ determined as a function of $s$. Then when $h_3$ reaches end of input, M will accept iff this guess was correct; i.e., if $d = c$. Since $s$ determines exactly one such $d$, $L(M)$ is a prediction language.
Now, to see that $L(M)$ masters every unitary multilinear infinite word, take any such word $\alpha$. Write $\alpha$ as above in terms of $q, m, r_i, a_i$, and $b_i$, such that $|r_i| = 1$ and without loss of generality, $r_i \neq r_{i+1}$ for all $1 \leq i < m$ and $r_m \neq r_1$ unless $m = 1$. (There is no loss of generality because if $r_i \neq r_{i+1}$, we can merge them by adding $a_i$ to $a_{i+1}$ and $b_i$ to $b_{i+1}$.) Consider $M$’s behavior on prefixes of $\alpha$. Notice first that the matching procedure always moves the heads forward at least one step, and the correction procedure increases the number of segments between the heads by one each time it is run. So eventually the matching procedure will be entered with all of the heads past the initial string $q$, with each of the three heads at the start of a segment, and with the number of segments between the heads being a multiple of $m$. At this point the segments under the three heads have a common $r_i, a_i$, and $b_i$. Now, in the matching procedure, $M$ first moves all three heads together for as long as they all agree with one another. If $a_i = 0$, then the three heads will agree and reach the next segment. If $a_i > 0$, then since $|r_i| = 1$, the three segments have the forms $c^j, c^{j+k}$, and $c^{j+2k}$, respectively, for some $j \geq 1, k \geq 1$, and symbol $c = r_i$. Now after $j$ steps, $h_1$ will disagree with $h_2$ and $h_3$. At this point the matching procedure moves $h_2$ one step at a time and $h_3$ two steps at a time, until they reach a symbol other than $c$. This will happen after $h_2$ has moved $k$ steps and $h_3$ has moved $2k$ steps. Now all three heads are at the start of the segments of the $r_{i+1}$ term, and we return to the start of the matching procedure. If at any point we reach end of input, then since the number of segments between $h_2$ and $h_3$ is a multiple of $m$, the last symbol of the input will match $h_2$, and $M$ will accept. Therefore $L(M)$ masters $\alpha$.

Having dealt with the subset of multilinear words which are unitary, we now define another subset of the multilinear words. Let $c$ be a symbol. A multilinear infinite word $\alpha$ is $c$-delimited if $\alpha$ can be written such that $r_m = c, a_m = 0, b_m = 1$, and $c$ does not appear in any $r_i$ for $1 \leq i < m$. ($c$ may appear in $q$.) For example, the multilinear word

$$\prod_{n \geq 0} a^{n+1}b = abaabaab\cdots$$

is b-delimited, and the multilinear word

$$\prod_{n \geq 0} (ab)^{n+1}(bc)^{n+1}d = abbcdbabbbcbcdababbbcbcbb\cdots$$

is d-delimited.

**Theorem 6.7.** Let $c$ be a symbol. Then some 1:multi-DFA prediction language masters every $c$-delimited multilinear word.
Proof. Let $M$ be a one-way 6-head deterministic finite automaton (1:6-DFA), operating as follows. $M$ starts with all six heads on the first input symbol. Call the six heads $h_0$, $h_1$, $h_2$, $h_3$, $h_4$, and $h_5$; we will use these labels also to refer to the positions of the heads in the input string. We say that two positions agree if they contain the same symbol and disagree otherwise. After initial positioning, the rightmost of the six heads will always be $h_5$. Whenever $M$ moves $h_5$, provided end of input is not reached, $M$ compares the symbol at the new position of $h_5$ to another symbol (specified below); we call this a guess. If the two symbols are equal, we say that the guess is correct; otherwise we say it is incorrect. When $h_5$ reaches end of input, $M$ accepts iff its last guess was correct.

To begin, for each $0 \leq i \leq 5$, $M$ moves head $h_i$ to the $i + 1$th occurrence of the symbol $c$ in the input string. From now on, $M$ will alternate between two procedures, a matching procedure and a correction procedure. Whenever the matching procedure is entered, every head is at an occurrence of $c$ and there are no occurrences of $c$ between the heads. For each head, call the next occurrence of $c$ past the one it started at the “target occurrence” for that head. During the matching procedure, for each head, $M$ records in its state whether or not that head has reached its target occurrence. No head is permitted to move past its target occurrence until every head has reached its target occurrence. Once every head has reached its target occurrence, the target for each head is set to the next occurrence of $c$ past the previous target occurrence, and so on.

**Matching Procedure** The matching procedure consists of a loop in which $M$ repeats the following six operations.

1. $M$ moves $h_1$, $h_2$, and $h_5$ together for as long as they all agree with one another. Whenever $h_5$ moves to a new input position, $M$’s guess is the symbol at $h_2$.

2. $M$ moves $h_0$, $h_3$, and $h_4$ together for as long as they all agree with one another.

3. $M$ moves $h_2$, $h_4$, and $h_5$ together for as long as they all agree with one another. Whenever $h_5$ moves to a new input position, $M$’s guess is the symbol at $h_4$.

4. $M$ moves $h_3$, $h_4$, and $h_5$ together for as long as they all agree with one another. Whenever $h_5$ moves to a new input position, $M$’s guess is the symbol at $h_4$.

5. $M$ moves $h_0$, $h_1$, and $h_2$ together for as long as they all agree with one another.

6. $M$ moves $h_2$, $h_4$, and $h_5$ together for as long as they all agree with one another. Whenever $h_5$ moves to a new input position, $M$’s guess is the symbol at $h_2$. 
If any guess is incorrect or if some operation would require a head to be moved past its target occurrence before the other heads have reached their target occurrences, \( M \) exits the matching procedure and enters the correction procedure.

**Correction Procedure** In the correction procedure, for each of the six heads, \( M \) advances that head until it reaches its target occurrence of \( c \), if it has not already done so. Once all of the heads are at their target occurrences, \( M \) exits the correction procedure and restarts the matching procedure.

**Analysis** To see that \( L(M) \) is a prediction language, write \( M \)’s input in the form \( sa \) with \( a \in \mathbb{A} \). The rightmost of the six heads is always \( h_5 \), because for any other head to pass it, that head would have to pass its target occurrence of \( c \) before \( h_5 \) had reached its own target occurrence, which is not permitted. When \( h_5 \) reaches \( a \), \( M \) will guess some symbol \( b \in \mathbb{A} \) determined as a function of \( s \), as specified in the matching procedure. (During initialization and in the correction procedure, take \( b \) to be an arbitrary fixed symbol in \( \mathbb{A} \).) Then when \( h_5 \) reaches end of input, \( M \) will accept iff this guess was correct; i.e., if \( b = a \). Since there is exactly one such \( b \) for \( s \), \( L(M) \) is a prediction language.

To see that \( L(M) \) masters every \( c \)-delimited multilinear word, take any such word \( \alpha \). Suppose \( \alpha \) is ultimately periodic. Then \( \alpha \) has the form \( x(y)c^\omega \) for some \( x, y \in \mathbb{A}^* \) such that \( c \) is not in \( y \). Suppose \( L(M) \) does not master \( \alpha \). Then on a sufficiently long prefix of \( \alpha \), \( M \) will repeatedly enter and exit the matching procedure. So \( M \) will eventually enter the matching procedure with every head past \( x \) and the heads at consecutive occurrences of \( c \). At this point \( M \) will move \( h_1, h_2, \) and \( h_5 \) together, guessing \( h_2 \). Since the heads were at occurrences of \( c \), they are separated by multiples of the period of \( \alpha \). Therefore all of \( M \)’s guesses will be correct. When \( h_1, h_2, \) and \( h_5 \) reach their target occurrences of \( c \), the correction procedure will move the other heads forward. Then the matching procedure will be re-entered and \( h_1, h_2, \) and \( h_5 \) will resume their advance. Therefore \( L(M) \) masters \( \alpha \).

So say \( \alpha \) is not ultimately periodic. Then by Theorem 3.15, \( \alpha \) can be written as

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

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 \( m \geq 2 \), \( r_1[1] \neq r_m[1] \).
Since $\alpha$ is not ultimately periodic, the above form contains one or more growth terms, those for which $a_i > 0$. Suppose $L(M)$ does not master $\alpha$. Then on a sufficiently long prefix of $\alpha$, $M$ will repeatedly enter and exit the matching procedure. So $M$ will eventually enter the matching procedure with every head past $q$ and the heads at consecutive occurrences of $c$. At this point there are strings $u, v, w, x, y$ such that for each $0 \leq i \leq 5$, $h_i$ is at the start of an input segment of the form $uv^i wx^i y$, where $v$ and $x$ correspond to growth terms, $v[1] \neq w[1]$, and $x[1] \neq y[1]$. We depict the positions of the heads below. By $h_s$ we mean that head $h$ is at the first symbol of string $s$.

\[\cdots h_0 \ u \ w \ y \ \cdots\]  
\[\cdots h_1 \ uv \ wx \ y \ \cdots\]  
\[\cdots h_2 \ uvv \ wxx \ y \ \cdots\]  
\[\cdots h_3 \ uvvv \ wxxx \ y \ \cdots\]  
\[\cdots h_4 \ uvvvv \ wxxxx \ y \ \cdots\]  
\[\cdots h_5 \ uvvvvv \ wxxxxx \ y \ \cdots\]

Now, in the first operation, $M$ moves $h_1, h_2, \text{ and } h_5$ until they disagree, which will happen after $|uv|$ symbols, when $h_1$ reaches $w$. If end of input is reached, $M$ guesses $h_2$, which agrees with $h_5$, and so $M$ will accept.

\[\cdots h_0 \ u \ w \ y \ \cdots\]  
\[\cdots uv \ h_1 \ wx \ y \ \cdots\]  
\[\cdots uv \ h_2 \ v \ wxx \ y \ \cdots\]  
\[\cdots h_3 \ uvv \ wxxx \ y \ \cdots\]  
\[\cdots h_4 \ uvvv \ wxxxx \ y \ \cdots\]  
\[\cdots uv \ h_5 \ uvv \ wxxxxx \ y \ \cdots\]

In the second operation, $M$ moves $h_0, h_3, \text{ and } h_4$ until they disagree, which will happen after $|u|$ symbols, when $h_0$ reaches $w$. End of input cannot be reached in this operation, since $h_4$ will not pass $h_5$.

\[\cdots u \ h_0 \ w \ y \ \cdots\]  
\[\cdots uv \ h_1 \ wx \ y \ \cdots\]  
\[\cdots uv \ h_2 \ v \ wxx \ y \ \cdots\]  
\[\cdots u \ h_3 \ vvv \ wxxx \ y \ \cdots\]  
\[\cdots u \ h_4 \ vvvv \ wxxxx \ y \ \cdots\]  
\[\cdots uv \ h_5 \ vvvv \ wxxxxx \ y \ \cdots\]

In the third operation, $M$ moves $h_2, h_4, \text{ and } h_5$ until they disagree, which will happen after $|v|$ symbols, when $h_2$ reaches $w$. If end of
input is reached, $M$ guesses $h_4$, which agrees with $h_5$, and so $M$ will accept.

\[
\begin{array}{c}
\cdots u h_o w y \cdots \\
\cdots u v h_r w x y \cdots \\
\cdots u v v h_2 w x x y \cdots \\
\cdots u h_3 v v v w x x x y \cdots \\
\cdots u v h_4 v v v w x x x x y \cdots \\
\cdots u v v v h_5 v v v w x x x x x y \cdots \\
\end{array}
\]

In the fourth operation, $M$ moves $h_3$, $h_4$, and $h_5$ until they disagree, which will happen after $|v v v w x x x|$ symbols, when $h_3$ reaches $y$. If end of input is reached, $M$ guesses $h_4$, which agrees with $h_5$, and so $M$ will accept.

\[
\begin{array}{c}
\cdots u h_o w y \cdots \\
\cdots u v h_r w x y \cdots \\
\cdots u v v h_2 w x x y \cdots \\
\cdots u v v v w x x x h_3 y \cdots \\
\cdots u v v v v w x x x h_4 x y \cdots \\
\cdots u v v v v v w x x x x h_5 x x y \cdots \\
\end{array}
\]

In the fifth operation, $M$ moves $h_0$, $h_1$, and $h_2$ until they disagree, which will happen after $|w|$ symbols, when $h_0$ reaches $y$. End of input cannot be reached in this operation, since $h_2$ will not pass $h_5$.

\[
\begin{array}{c}
\cdots u w h_o y \cdots \\
\cdots u v w h_1 x y \cdots \\
\cdots u v v w h_2 x x y \cdots \\
\cdots u v v v w x x x h_3 y \cdots \\
\cdots u v v v v w x x x h_4 x y \cdots \\
\cdots u v v v v v w x x x x h_5 x x y \cdots \\
\end{array}
\]

In the sixth operation, $M$ moves $h_2$, $h_4$, and $h_5$ until they disagree, which will happen after $|x|$ symbols, when $h_4$ reaches $y$. If end of input is reached, $M$ guesses $h_2$, which agrees with $h_5$, and so $M$ will accept.

\[
\begin{array}{c}
\cdots u w h_o y \cdots \\
\cdots u v w h_1 x y \cdots \\
\cdots u v v w x h_2 x y \cdots \\
\cdots u v v v w x x x h_3 y \cdots \\
\cdots u v v v v w x x x x h_4 y \cdots \\
\cdots u v v v v v w x x x x x h_5 x y \cdots \\
\end{array}
\]
Now M returns to the top of the loop and performs the first operation again, moving \( h_1, h_2, \) and \( h_5 \) until they disagree. After \(|xy|\) symbols, \( h_1, h_2, \) and \( h_5 \) will reach \( y \) without having disagreed. If end of input is reached, M guesses \( h_2 \), which agrees with \( h_5 \), and so M will accept.

Now the six heads are all back at the beginning of corresponding segments, and the entire procedure will repeat. If this continues until end of input, M will accept. If it does not continue until end of input, then since M does not miss a guess during this procedure, it must be that some operation requires a head to be moved past its target occurrence of \( c \) before the other heads have reached their target occurrences. This can happen only during an application of the above procedure in which \( c \) occurs in \( u \) or \( w \) (since \( c \) cannot occur in a growth term). If \( c \) occurs in \( u \) (other than as the first symbol, in which case there is no problem) then in the first operation, \( h_1, h_2, \) and \( h_5 \) will try to pass their target occurrences, causing the correction procedure to advance the other heads. Then the matching procedure will be re-entered with all of the heads again at consecutive occurrences of \( c \). Since M does not miss a guess during this process, it will repeat and M will accept. If \( c \) occurs in \( w \) (other than as the first symbol, in which case there is no problem), then in the fourth operation, \( h_3, h_4, \) and \( h_5 \) will try to pass their target occurrences, causing the correction procedure to advance the other heads. Again the matching procedure will be re-entered with all of the heads at consecutive occurrences of \( c \), and M will not miss a guess. Consequently, M accepts every sufficiently long prefix of \( \alpha \), and so \( L(M) \) masters \( \alpha \). 

### 6.3 Number of Guesses

Besides the question of whether a given prediction language can master a given infinite word, it is also interesting to consider the number of “guesses” required to reach mastery. Let \( L \) be a prediction language and \( \alpha \) be an infinite word. If \( L \) masters \( \alpha \), define

\[
g(L, \alpha) = \min \{ i | \text{ every prefix of } \alpha \text{ of length at least } i \text{ is in } L \}.\]

We say that \( L \) masters \( \alpha \) in \( g(L, \alpha) \) guesses. We are particularly interested to know, with respect to some notion of the “size” of a description of \( \alpha \), a bound on \( g(L, \alpha) \) as a function of that size. For example,
let us say that \( \alpha \) has an **ultimately periodic description of size** \( n \) if \( \alpha = x y^{\omega} \) for some strings \( x, y \) such that \( |x y| = n \). For each \( n \geq 1 \), let \( U_n = \{ \alpha \mid \alpha \) has an ultimately periodic description of size \( n \} \). Notice that each \( U_i \) is finite. Now for any finite set \( S \) of infinite words, if \( L \) masters every infinite word in \( S \), define

\[
g(L, S) = \max \{ g(L, \alpha) \mid \alpha \in S \}.
\]

For a prediction language \( L \) which masters every ultimately periodic word, the function \( g(L, U_n) \) gives the maximum number of guesses needed by \( L \) to master an infinite word with an ultimately periodic description of size \( n \). To illustrate these concepts, we apply them to the 1-multi-DFA prediction language from the proof of Theorem 6.5.

**Theorem 6.8.** There is a 1-multi-DFA prediction language \( L \) such that \( L \) masters every ultimately periodic word and \( g(L, U_n) \in \Theta(n^2) \).

**Proof.** Let \( M \) be the one-way 2-head deterministic finite automaton (1:2-DFA) described in the proof of Theorem 6.5, and let \( L = L(M) \). First, we establish \( g(L, U_n) \in \Omega(n^2) \). Take any \( n \geq 2 \). Let \( m = n - 1 \) and let \( \alpha = (a^m b)^{\omega} \). Then \( \alpha \) is in \( U_n \). Now because the distance between the two heads of \( M \) increases by two when a \( b \) is passed (by one when the right head passes it and by one when the left head passes it), \( \frac{m}{2} \) \( b \)'s must be passed for the distance between the two heads to reach a multiple of \( m + 1 \), the period length, at which point mastery is achieved. Once \( \frac{m}{2} \) \( b \)’s have been passed, the right head is at position \( m \left( \frac{m}{2} \right) \), and all prefixes of \( \alpha \) extending past this point will be accepted. So \( g(L, \alpha) \geq m \left( \frac{m}{2} \right) \geq \frac{(n-1)^2}{2} \). Hence \( g(L, U_n) \in \Omega(n^2) \).

Now we establish \( g(L, U_n) \in O(n^2) \). Take any \( n \geq 1 \) and \( \alpha \in U_n \). Write \( \alpha \) as \( x y^{\omega} \) for strings \( x, y \) such that \( |x y| = n \). When the left head of \( M \) reaches the start of \( y \), the two heads will be separated by some distance \( d \leq |x| \), so the right head will be at position at most \( 2|x| \). To \( d \) at most \( |y| - 1 \) must be added to make a multiple of \( |y| \). Therefore the two heads can mismatch at most \( |y| - 1 \) more times. As long as mastery has not been achieved, \( M \) can go at most \( |y| - 1 \) steps without a mismatch, since if it went \( y \) steps, then the same \( y \) symbols it passed would repeat and all of its subsequent guesses would be correct. Thus when the right head reaches position \( 2|x| + (|y| - 1)^2 \), \( M \) will have achieved mastery. Therefore \( g(L, \alpha) \leq 2|x| + (|y| - 1)^2 \). Hence \( g(L, U_n) \in O(n^2) \).

Since \( g(L, U_n) \in \Omega(n^2) \) and \( g(L, U_n) \in O(n^2) \), we have \( g(L, U_n) \in \Theta(n^2) \), which was to be shown.

The above theorem gives a 1-multi-DFA prediction language \( L \) which masters every ultimately periodic word and has quadratic \( g(L, U_n) \). We do not know whether there is a 1-multi-DFA prediction language \( L \) which masters every ultimately periodic word and has subquadratic \( g(L, U_n) \). If the restriction to 1-multi-DFA prediction languages is
lifted, then a linear $g(L, U_n)$ is achievable, and in fact [Job88] gives an algorithm which, viewed as a prediction language, masters every ultimately periodic word, has a linear $g(L, U_n)$ (see the proof of Theorem 5 in that paper), and additionally can be implemented in time and space linear in the size of the input string, using techniques from string matching.

Stepping back from particular prediction languages, we can ask, for a class of infinite words, what bounds exist on the number of guesses required for mastery due solely to the nature of the descriptions of the infinite words in that class. We turn now to the study of such bounds, which arise from the notion of agreement of infinite words.

### 6.3.1 Maximum agreement

Closely related to the number of guesses required to achieve mastery of an infinite word is the topic of agreement of infinite words. This topic is concerned with the question, for a finite set of infinite words $S$ (of cardinality at least 2), of what is the largest $n$ such that two distinct infinite words in $S$ share a prefix of length $n$. We call $n$ the maximum agreement of $S$. The answer to this question constitutes a lower bound on $g(L, S)$ for any prediction language $L$ which masters every infinite word in $S$. This bound follows from the fact that if two distinct infinite words $\alpha, \beta \in S$ share a prefix of length $n$, then any $L$ which masters $\alpha$ and $\beta$ requires more than $n$ guesses to master at least one of them, because $\alpha$ and $\beta$ first differ at some position $n' > n$, and $L$’s guess for this position must be wrong for $\alpha$ or for $\beta$.

Below, we study the question of agreement for various classes of infinite words. For each class, we define a notion of “size” for the infinite words in that class, and then we study the maximum agreement of the infinite words in that class of a given size. Our results are summarized in Table 6.1. There and below, $A$ is the alphabet of the infinite words under consideration.

### 6.3.2 Agreement of ultimately periodic words

Recall that an infinite word $\alpha$ has an ultimately periodic description of size $n$ if $\alpha = xy^\omega$ for some strings $x, y$ such that $|xy| = n$. The following two theorems give an exact characterization of the maximum agreement of distinct infinite words with ultimately periodic descriptions of a given size. The lower bound is established by an explicit construction and the upper bound is obtained using an analog of the Fine–Wilf theorem for ultimately periodic words due to [CdLV99, CdL00].

**Theorem 6.9.** For every $n \geq 1$, there are two distinct infinite words with ultimately periodic descriptions each of size at most $n$ which agree on the first $2n - 2$ symbols.
<table>
<thead>
<tr>
<th>Class of Words</th>
<th>Max. Agreement</th>
<th>n</th>
</tr>
</thead>
<tbody>
<tr>
<td>ultimately periodic</td>
<td>$2n - 2$</td>
<td>sum of preperiod and period lengths</td>
</tr>
<tr>
<td>multilinear</td>
<td>$\geq 6n - 9$</td>
<td>sum of string lengths</td>
</tr>
<tr>
<td>D0L</td>
<td>$\Omega(n^{</td>
<td>A</td>
</tr>
<tr>
<td>morphic</td>
<td>$\geq (</td>
<td>\frac{n}{2}</td>
</tr>
</tbody>
</table>

Table 6.1: Maximum agreement of distinct infinite words of description size $n$ in various classes (for sufficiently large $n$). For multilinear words we use the notion of “expanded size” defined below.

Proof. For $n = 1$, take $a^\omega$ and $b^\omega$. For $n \geq 2$, take $\alpha = a(a^{n-2}b)^\omega$ and $\beta = (a^{n-1}b)^\omega$. Then $\alpha$ and $\beta$ share the prefix $a^{n-1}ba^{n-2}$ but differ on the next symbol after that. Therefore $\alpha$ and $\beta$ are distinct infinite words with ultimately periodic descriptions each of size $n$ which agree on the first $2n - 2$ symbols.

**Theorem 6.10.** Let $\alpha$ and $\beta$ be distinct infinite words with ultimately periodic descriptions each of size at most $n$. Then $\alpha$ and $\beta$ agree on at most the first $2n - 2$ symbols.

Proof. Write $\alpha$ as $ux^\omega$ and $\beta$ as $vy^\omega$, where $|ux| \leq n$ and $|vy| \leq n$. Suppose $\alpha$ and $\beta$ agree on the first $2n - 1$ symbols. Then $\alpha$ and $\beta$ have a common prefix of length $|ux| + |vy| - 1$. Then by Theorem 6 of [CdLV99] (see the sentence immediately following the theorem, as well as Lemma 2 and Proposition 5 of [CdLo01]), $\alpha = \beta$, a contradiction. Therefore $\alpha$ and $\beta$ agree on at most the first $2n - 2$ symbols.

**6.3.3 Agreement of multilinear words**

For descriptions of multilinear words we define three notions of size.

Recall that a multilinear word $\alpha$ has the form

$$q \prod_{n \geq 0} r_1^{a_1n + b_1} r_2^{a_2n + b_2} \ldots r_m^{a_mn + b_m}$$

We call the above form a **multilinear description** $D$ of $\alpha$. For $i \geq 0$, we denote by $D[i]$ the subword of $\alpha$ evaluated at $n = i$, i.e.

$$r_1^{a_1i + b_1} r_2^{a_2i + b_2} \ldots r_m^{a_mi + b_m}.$$ 

Below, for any integer $i \geq 0$, $(i)_2$ denotes the binary representation of $i$, without leading zeros. For example, $(19)_2$ is the string 10011 and $(0)_2$ is the empty string $\lambda$.

We say that $\alpha$ has a **multilinear description of**
• **succinct size** \( |q| + \sum_{i=1}^{m} |r_i| + |(a_i)_2| + |(b_i)_2| \)

• **verbose size** \( |q| + \sum_{i=1}^{m} |r_i| + a_i + b_i \)

• **expanded size** \( |q| + \sum_{i=1}^{m} |r_i| \cdot (a_i + b_i) \)

We give lower bounds on the maximum agreement of distinct infinite words with multilinear descriptions of a given size for each of these three notions. We then show that obtaining an upper bound on maximum agreement for the notion of expanded size is related to certain open problems in the study of word equations.

**Theorem 6.11.** For every \( p \geq 4 \), there are two distinct infinite words with multilinear descriptions each of succinct size at most \( p \) which agree on the first \( 2^{p-4} \) symbols.

*Proof.* Take \( \alpha = \prod_{n \geq 0} a \) and \( \beta = \prod_{n \geq 0} (a)^{2^{p-4}} b \). Then \( \alpha \) and \( \beta \) are distinct infinite words with multilinear descriptions each of succinct size at most \( p \), and agree on the first \( 2^{p-4} \) symbols. \( \square \)

**Theorem 6.12.** For every \( p \geq 2 \), there are two distinct infinite words with multilinear descriptions each of verbose size at most \( p \) which agree on the first \( (\lfloor \frac{p}{2} \rfloor - 1)^2 \) symbols.

*Proof.* Let \( k = \lfloor \frac{p}{2} \rfloor \) and take \( \alpha = \prod_{n \geq 0} a \) and \( \beta = \prod_{n \geq 0} (a^k)^k b \). Then \( \alpha \) and \( \beta \) are distinct infinite words with multilinear descriptions each of verbose size at most \( p \), and agree on the first \( k^2 = (\lfloor \frac{p}{2} \rfloor - 1)^2 \) symbols. \( \square \)

**Theorem 6.13.** For every \( p \geq 3 \), there are two distinct infinite words with multilinear descriptions each of expanded size at most \( p \) which agree on the first \( 6p - 9 \) symbols.

*Proof.* Let \( k = p - 3 \). Take \( \alpha = \prod_{n \geq 0} a^k (ab)^n a^n \) and \( \beta = \prod_{n \geq 0} a^k (ba)^n a^n \).

Then \( \alpha \) and \( \beta \) have multilinear descriptions each of expanded size at most \( p \). We have

\[
\alpha = a^1 a^2 a^2 a^3 a^3 a^3 \ldots
\]

\[
\beta = a^1 a^2 a^2 a^3 a^3 \ldots
\]

The first point of disagreement is marked in bold. As can be seen, \( \alpha \) and \( \beta \) are distinct infinite words which agree on the first \( 6k + 9 = 6(p - 3) + 9 = 6p - 9 \) symbols, which was to be shown. \( \square \)
We now show that obtaining a linear upper bound on maximum agreement for the notion of expanded size is related to certain open problems in the study of word equations. The following two open problems are from [HK07].

- **Open Problem 1.** Let $S$ be the infinite system of word equations
  \[
  \{x_0u_1^ix_1u_2^ix_2\cdots u_j^ix_j = y_0v_1^iy_1v_2^iy_2\cdots v_k^iy_k \mid i \geq 0\}
  \]
  Does there exist $q \geq 0$ such that (for any $j, k \geq 0$) the system $S$ is equivalent to the subsystem induced by $i = 0, 1, 2, \ldots, q$?

- **Open Problem 2.** Is the system $\{u_1^i = v_1^iy_1^i \cdots v_k^iy_k^i \mid i \geq 0\}$ equivalent to the subsystem induced by $i = 1, 2, 3$?

We relate these two open problems to maximum agreement of multilinear words with the following theorems.

**Theorem 6.14.** Suppose that for some $b, c \geq 0$, for every $p \geq 0$, no two distinct infinite words with multilinear descriptions each of expanded size at most $p$ agree on the first $cp + b$ symbols. Then Open Problem 1 of [HK07] has a positive answer.

**Proof.** Suppose Open Problem 1 has a negative answer. Let $q = c + b$. Then for some $j, k \geq 0$, the system $S$ is not equivalent to the subsystem induced by $i = 0, 1, 2, \ldots, q$. So $S$ has a solution for $0 \leq i \leq q$ which is not a solution for $i \geq 0$. Hence for some $r > q$, there are strings $x_0, u_1, x_1, u_2, x_2, \ldots, u_j, x_j$ and $y_0, v_1, y_1, v_2, y_2, \ldots, v_k, y_k$ which are a solution for all $0 \leq i \leq q$ but not for $i = r$. Let $l = |x_0x_1x_2\cdots x_j|$, $m = |u_1u_2\cdots u_j|$, and $p = m + l$. Let

\[
\alpha = \prod_{n \geq 0} x_0u_1^n x_1u_2^n x_2\cdots u_j^n x_j
\]

\[
\beta = \prod_{n \geq 0} y_0v_1^n y_1v_2^n y_2\cdots v_j^n y_j
\]

Let $D_\alpha$ and $D_\beta$ be the above multilinear descriptions of $\alpha$ and $\beta$, respectively. For $0 \leq i \leq q$, we have $D_\alpha(i) = D_\beta(i)$. Clearly $q \geq 1$, so we have $D_\alpha(0) = D_\beta(0)$ and $D_\alpha(1) = D_\beta(1)$. Then the expanded size of $D_\alpha$ equals the expanded size of $D_\beta$ equals $p$, and for all $i \geq 0$, $|D_\alpha(i)| = |D_\beta(i)| = mi + l$. Then since $D_\alpha(r) \neq D_\beta(r)$, $\alpha$ and $\beta$ are
distinct multilinear words. Now \( \alpha \) and \( \beta \) agree on the first \( \sum_{i=0}^{q} (m_i + 1) \) symbols. We have

\[
\sum_{i=0}^{q} (m_i + 1) = l(q + 1) + \sum_{i=0}^{q} m_i \\
= l(q + 1) + m \sum_{i=0}^{q} i \\
\geq lq + mq \\
\geq q(m + 1) \\
\geq qp \\
\geq p(c + b) \\
\geq cp + b
\]

So \( \alpha \) and \( \beta \) agree on the first \( cp + b \) symbols, a contradiction. Therefore Open Problem 1 has a positive answer.

**Theorem 6.15.** Suppose that for every \( p \geq 0 \), no two distinct infinite words with multilinear descriptions each of expanded size at most \( p \) agree on the first \( 9p \) symbols. Then Open Problem 2 of [HK07] has a positive answer.

**Proof.** Suppose Open Problem 2 has a negative answer. Then the system \( \{u^i_1 = v^i_1 v^i_2 \cdots v^i_k \mid i \geq 0\} \) is not equivalent to the subsystem induced by \( i = 1, 2, 3 \). This means that the system has a different set of solutions for \( i \geq 0 \) than it does for \( 1 \leq i \leq 3 \). So the system has a solution for \( 1 \leq i \leq 3 \) which is not a solution for \( i \geq 1 \) (for \( i = 0 \), everything is a solution.) Hence for some \( j > 3 \) and \( k \geq 0 \), there are strings \( u_1, v_1, v_2, \ldots, v_k \) which are a solution for all \( 1 \leq i \leq 3 \) but not for \( i = j \). Let \( m = |u_1| \) and \( p = 4m \). Let

\[
\alpha = \prod_{n \geq 0} (a^{p-m})^n u_1^n \\
\beta = \prod_{n \geq 0} (a^{p-m})^n v_1^n v_2^n \cdots v_k^n
\]

Let \( D_\alpha \) and \( D_\beta \) be the above multilinear descriptions of \( \alpha \) and \( \beta \), respectively. Since \( u_1 = v_1 v_2 \cdots v_k \), the expanded size of \( D_\alpha \) equals the expanded size of \( D_\beta \) equals \( p \). Now, for all \( i \geq 1 \), we have \(|D_\alpha(i)| = |D_\beta(i)| = p_i \). Then since \( D_\alpha(i) \neq D_\beta(i) \), \( \alpha \) and \( \beta \) are distinct multilinear words. For \( 1 \leq i \leq 3 \), we have \( D_\alpha(i) = D_\beta(i) \). Further, \( D_\alpha(4) \) and \( D_\beta(4) \) agree on the first \( 4(p - m) \) symbols. Then \( \alpha \) and \( \beta \) agree on the first \( p + 2p + 3p + 4(p - m) = 10p - 4m = 9p \) symbols, a contradiction. So Open Problem 2 has a positive answer.

We now mention another formulation of Open Problem 2. One of the authors of [HK07], Štěpán Holub, on his website\(^1\) offers to pay 200 € for the answer to the following “prize problem”:

-------------------
• **Prize Problem.** Is there a positive integer \( n \geq 2 \) and words \( u_1, u_2, \ldots, u_n \) such that both equalities

\[
\begin{align*}
(u_1u_2 \cdots u_n)^2 &= u_1^2u_2^2 \cdots u_n^2, \\
(u_1u_2 \cdots u_n)^3 &= u_1^3u_2^3 \cdots u_n^3,
\end{align*}
\]

hold and the words \( u_i, \ i = 1, \ldots, n \), do not pairwise commute (that is, \( u_iu_j \neq u_ju_i \) for at least one pair of indices \( i, j \in \{1, 2, \ldots, n\} \))?

That the Prize Problem is equivalent to Open Problem 2 follows from known results; we give a proof for completeness.

**Proposition 6.16.** Open Problem 2 of \([HK07]\) has a negative answer iff Holub’s Prize Problem has a positive answer.

**Proof.** \( \implies \) Suppose Open Problem 2 has a negative answer. Then for some \( j > 3 \) and \( n \geq 0 \), there are strings \( u_1, v_1, v_2, \ldots, v_n \) which are a solution to the system of Open Problem 2 for all \( 1 \leq i \leq 3 \) but not for \( i = j \). Clearly \( n \geq 2 \). This solution cannot be cyclic, since if it was then it would be a solution for \( i = j \). (A solution is cyclic if for some string, the answers are all powers of that string.) Then by Proposition 1.3.2 of \([Lot97]\), the words \( v_i, \ i = 1, \ldots, n \), do not pairwise commute. Therefore the Prize Problem has a positive answer.

\( \iff \) Suppose the Prize Problem has a positive answer. Then there is an \( n \geq 2 \) and strings \( u_1, u_2, \ldots, u_n \) such that both equalities

\[
\begin{align*}
(u_1u_2 \cdots u_n)^2 &= u_1^2u_2^2 \cdots u_n^2, \\
(u_1u_2 \cdots u_n)^3 &= u_1^3u_2^3 \cdots u_n^3,
\end{align*}
\]

hold and the strings \( u_i, \ i = 1, \ldots, n \), do not pairwise commute. Then there are \( k, l \) such that \( u_ku_1 \neq u_lu_k \). Then neither \( u_k \) nor \( u_1 \) is empty. Then by Theorem 1.5.3 of \([AS03]\), there is no string \( z \) such that \( u_k \) and \( u_l \) are both powers of \( z \). Now by Theorem 1 of \([AD68]\), the equation \( x_1^m x_2^m \cdots x_m^m = y^m \) has only cyclic solutions. Therefore \( n \geq 4 \) and \( (u_1u_2 \cdots u_n)^n \neq u_1^n u_2^n \cdots u_n^n \). Then the system in Open Problem 2 has a solution for \( i = 1, 2, 3 \) which is not a solution for \( i \geq 0 \), giving that problem a negative answer.

Therefore Open Problem 2 has a negative answer iff the Prize Problem has a positive answer. \( \square \)

Putting the above results together, we connect maximum agreement of multilinear words with the 200 \( \in \) Prize Problem.

**Theorem 6.17.** Suppose that for every \( p \geq 0 \), no two distinct infinite words with multilinear descriptions each of expanded size at most \( p \) agree on the first \( 9p \) symbols. Then Holub’s Prize Problem has a negative answer.

**Proof.** From Theorem 6.15 and Proposition 6.16. \( \square \)
6.3.4 Agreement of D0L words

We say that an infinite word \( \alpha \) has a D0L description of size \( n \) if for some morphism \( h \) and string \( w \), the language \( \{h^i(w) \mid i \geq 0\} \) determines \( \alpha \) and \( |w| + \sum_{c \in \Lambda} |h(c)| = n \). If additionally \( h(c) \neq \lambda \) for all \( c \in \Lambda \), we say that \( \alpha \) has a PD0L description of size \( n \). If \( |w| = 1 \), we say that \( \alpha \) has a pure morphic description of size \( n \). We give a lower bound on the maximum agreement of distinct infinite words with descriptions of these types. A computable upper bound on D0L words is decidable was a hard problem solved in [CH84]. In [Hon03] it is observed that no such bound has been explicitly given in the general case. For progress on special cases, see [Hon03, Hon09].

**Theorem 6.18.** Suppose \( |\Lambda| \geq 2 \). For every \( n \geq |\Lambda| + 2 \), there are two distinct infinite words, each with a description which is at once D0L, PD0L, pure morphic, and of size at most \( n \), which agree on the first \( (n - |\Lambda| - 1)^{|\Lambda|} \) symbols.

**Proof.** Let \( j = |\Lambda| \), write \( A \) as \( \{a_1, \ldots, a_j\} \), and let \( k = n - j - 1 \). Let \( w = a_1 \). Let \( h_1 \) and \( h_2 \) be morphisms over \( A \). Let \( h_1(a_1) = h_2(a_1) = a_1^0a_2 \). For \( 2 \leq i < j \), let \( h_1(a_i) = h_2(a_i) = a_{i+1} \). Let \( h_1(a_1) = a_1 \) and \( h_2(a_j) = a_j \). Let \( \alpha \) be the D0L word determined by \( \{h_1^i(w) \mid i \geq 0\} \) and let \( \beta \) be the D0L word determined by \( \{h_2^i(w) \mid i \geq 0\} \). Notice that the above descriptions of \( \alpha \) and \( \beta \) are each of size \( j + k + 1 = n \) and are at once D0L, PD0L, and pure morphic. Now for all \( i \geq 0 \), we have \( |h_1^i(w)| = |h_2^i(w)| > k^i \). Further, from the definitions of \( h_1 \) and \( h_2 \), we have \( h_1^i(w) = h_2^i(w) \) for \( 0 \leq i < j \), and the first appearance of \( a_j \) is as the last symbol of \( h_1^{j-1}(w) = h_2^{j-1}(w) \). Then \( h_1^j(w) \) and \( h_2^j(w) \) agree on all but the last symbol, which is \( a_1 \) for the former and \( a_j \) for the latter. Hence \( \alpha \) and \( \beta \) are distinct words which agree on the first \( k^j = (n - |\Lambda| - 1)^{|\Lambda|} \) symbols, which was to be shown. \( \square \)

The above theorem shows that with no restriction on the alphabet, maximum agreement of D0L words ceases to be polynomial in the description size, leading to dramatic effects. For example, following the construction in the proof, take the morphisms \( h_1 \) and \( h_2 \) defined below.

\[
\Lambda = \{a_1, \ldots, a_{80}\}
\]

\[
\begin{align*}
  h_1 &: a_1 \to a_1^0a_2 \\
  h_2 &: a_1 \to a_1^0a_2 \\
  a_1 &\to a_{i+1} \text{ for } 2 \leq i \leq 79 \\
  a_1 &\to a_{i+1} \text{ for } 2 \leq i \leq 79 \\
  a_{80} &\to a_1 \\
  a_{80} &\to a_{80}
\end{align*}
\]

The pure morphic words \( h_1^\omega(a_1) \) and \( h_2^\omega(a_1) \) are distinct but do not differ until after a prefix of length \( 10^{80} \), roughly estimated to be the number of atoms in the observable universe!
6.3.5 Agreement of morphic words

We say that an infinite word $\alpha$ has a **morphic description of size** $n$ if for some morphism $h$, coding $e$, and symbol $c$ such that $\{h^i(w) \mid i \geq 0\}$ is an infinite prefix language, $\alpha = e(h^\omega(c))$ and $1 + \sum_{c \in A} |h(c)| = n$.

We give a lower bound on the maximum agreement of distinct infinite words with morphic descriptions of a given size. A computable upper bound on agreement of morphic words would imply decidability of equivalence of morphic words, which is Open Problem 7.11.1 of [AS03].

**Theorem 6.19.** For every $n \geq 6$, there are two distinct binary infinite words with morphic descriptions each of size at most $n$ which agree on the first $(\lfloor \frac{n}{2} \rfloor - 1) \lceil \frac{n}{2} \rceil - 1$ symbols.

**Proof.** Let $j = \lfloor \frac{n}{2} \rfloor - 1$ and write $A$ as $\{a_1, \ldots, a_i\}$. Let $h_1$, $h_2$, and $e$ be morphisms over $A$. Let $h_1(a_i) = h_2(a_i) = a_i^j a_2$. For $2 \leq i < j$, let $h_1(a_i) = h_2(a_i) = a_{i+1}$. Let $h_1(a_{j}) = a_1$ and $h_2(a_{j}) = a_2$. Let $e(a_i) = a$ for $1 \leq i < j$ and let $e(a_{j}) = b$. Let $\alpha$ be the binary morphic word $e(h_1^\omega(a_1))$ and let $\beta$ be the binary morphic word $e(h_2^\omega(a_1))$. Then $\alpha$ and $\beta$ have morphic descriptions each of size $2j + 1 \leq n$. Now for all $i \geq 0$, we have $|h_1^i(a_1)| = |h_2^i(a_1)| > j^i$. Further, from the definitions of $h_1$ and $h_2$, we have $h_1^i(a_1) = h_2^i(a_1)$ for $0 \leq i < j$, and the first appearance of $a_j$ is as the last symbol of $h_1^{j-1}(a_1) = h_2^{j-1}(a_1)$. Then $h_1^i(a_1)$ and $h_2^i(a_1)$ agree on all but the last symbol, which is $a_1$ for the former and $a_j$ for the latter. Since $e(a_1) \neq e(a_{j})$, $\alpha$ and $\beta$ are distinct words which agree on the first $j^j = (\lfloor \frac{n}{2} \rfloor - 1) \lceil \frac{n}{2} \rceil - 1$ symbols, which was to be shown. \qed

6.4 conclusion

In this chapter, we extended our investigation of infinite words from determination to prediction, via the notion of a prediction language. We examined several types of infinite words with respect to which classes of prediction languages can master them, focusing on prediction languages recognized by different types of automata. We also considered the number of guesses needed to master infinite words with descriptions of certain sizes, as well as the related question of how many symbols two distinct infinite words of a certain description size can agree on before differing. Open problems in our investigation include determining whether or not some 1:multi-DFA prediction language masters every multilinear word, and establishing a tighter bound on agreement of multilinear words, which as we noted is connected to a 200 € prize problem. It would also be interesting to generalize the notion of mastery to a setting in which the emitter of the symbols to be predicted is nondeterministic or probabilistic.
In this dissertation we have explored a notion of complexity for infinite words relating them to languages of their prefixes. Much as the computational complexity of a decision problem is characterized by the complexity classes to which it belongs, the prefix language complexity of an infinite word is characterized by the language classes in which it is determined. We investigated a variety of classes of languages and infinite words to see what connections could be made among them within this framework, and we gave a number of results aimed at building up a classification of infinite words with respect to which classes of languages can determine them. We made particular use of pumping lemmas as a tool for studying these classes and their relationship to infinite words.

7.1 Summary of Results

Figure 7.1 overviews our results. First, we investigated infinite words determined by various types of automata. We showed that languages recognized by finite automata (REG), pushdown automata (CFL), and checking stack automata (1-NCSA and 1-DCSA) can determine only ultimately periodic words, whereas those of deterministic nonerasing stack automata (1-DNES), nondeterministic nonerasing stack automata (1-NNES), deterministic stack automata (1-DSA), and nondeterministic stack automata (1-NSA) determine a class of infinite words which we call multilinear. We found that multihead finite automata (1-multi-DFA) can determine every multilinear infinite word, and can also determine infinite words which are not multilinear.

Second, we investigated infinite words determined by the parallel rewriting systems known as L systems. With the help of a pumping lemma which we proved for T0L systems, we showed that certain infinite L systems necessarily have infinite subsets in other L systems. We used these relationships to categorize the infinite words determined by a hierarchy of 96 L systems, showing that it collapses to just three distinct classes of infinite words: ω(PD0L), ω(D0L), and ω(CD0L).

Third, we investigated infinite words determined by indexed grammars. We observed that the class of linear indexed languages LIL determines only ultimately periodic words, making use of an existing pumping lemma for that class. We then extended to the class of indexed languages IL a pumping lemma recently proved for the class of L systems ETOL, and used it to show that ω(IL) consists of exactly the morphic words.
Fourth, we extended our investigation of infinite words from determination to prediction. We formulated the notion of a prediction language which masters, or learns in the limit, some set of infinite words. Using this notion we explored the predictive capabilities of several types of automata, relating them to periodic and multilinear words. We found that no regular prediction language masters every purely periodic word, and we showed that there is a 1-DNESA prediction language which does so. We showed that there are 1:multi-DFA prediction languages which master every ultimately periodic word and which master certain subsets of the multilinear words. We then considered the number of guesses required to master infinite words of particular description sizes, and in the case of multilinear words we related this notion to open problems in the study of word equations.

7.2 Further Work

Together, these results form a foundation for a classification of infinite words with respect to which classes of languages can determine and predict them. Because the notion of a prefix language determining an infinite word can be applied to arbitrary language classes, it offers
many opportunities for further research. For any language class, we can ask what class of infinite words it determines. From the other direction, for any infinite word, we can ask in what language classes it can be determined. Specific tasks of interest include:

- Characterize the infinite words determined by 1:multi-DFA, beyond the result we established, that the multilinear infinite words are properly contained by this class.

- Situate the automatic sequences [AS03] within the framework of prefix languages. Is there an existing language class which determines exactly these sequences?

Similarly, in the area of predicting infinite words, many avenues can be explored. For any class of languages, we can ask what sets of infinite words can be mastered by those languages, and for any set of infinite words we can ask in what language classes it can be mastered. Specific tasks of interest include:

- Prove or disprove: For some alphabet $\Lambda$ with $|\Lambda| \geq 2$, some 1:multi-DFA prediction language masters every multilinear infinite word over $\Lambda$.

- Show that for every $p \geq 0$, no two distinct infinite words with multilinear descriptions each of expanded size at most $p$ agree on the first $9p$ symbols. (From Theorem 6.17, this would solve Holub’s 200 € prize problem.)
ACKNOWLEDGMENTS

I want to thank my advisor, Rajmohan Rajaraman, for supporting this work, for encouraging and helping me, for all of his comments and suggestions and advice, and for achieving a perfect balance of hands-off and hands-on advising. Raj is an academic role model and all-around great guy.

I would also like to thank the other members of my thesis committee: Prof. Javed Aslam, Prof. Arnold Rosenberg, Prof. Ravi Sundaram, and my external examiner, Prof. Juhani Karhumäki of the University of Turku.

I would like to thank my family for their love and support, and especially I would like to thank the many friends I have made at Northeastern during the undertaking of this work.

COLOPHON

This document was typeset using the typographical look-and-feel classicthesis developed by André Miede. The style was inspired by Robert Bringhurst’s seminal book on typography “The Elements of Typographic Style”. classicthesis is available for both \LaTeX{} and \LaX: 

http://code.google.com/p/classicthesis/

Final Version as of April 22, 2015 (classicthesis version 4.1).


[CdLV99] Arturo Carpi, Aldo de Luca, and Stefano Varricchio, *Special factors in some combinational structures*, Developments


bridge, 1997, corrected reprint of the 1983 original, with a new preface. (Cited on page 78.)


