

CONSTANTS AND SPLICING SYSTEMS

BY

T. ELIZABETH GOODE LAUN

BS, Regents College at Albany, 1989

MAT, State University College at Cortland, 1990

MA, State University of New York at Binghamton, 1994

DISSERTATION

Submitted in partial fulfillment of the requirements for

the degree of Doctor of Philosophy in Mathematics

in the Graduate School

Binghamton University

State University of New York

1999

©Copyright by T. Elizabeth Goode Laun 1999  
All Rights Reserved

## ACKNOWLEDGEMENTS

I thank my co-advisors Tom Head and Dennis Pixton for taking me as their student, and guiding my progress throughout this journey. I appreciate all the time they invested in me, and the support they have given me during our years together. I especially appreciate the fascinating and novel ideas in mathematics and in other disciplines which Tom Head has shared with me during this time. And I am grateful to Dennis Pixton for his extraordinary patience and his willingness to help me approach problems which I found difficult.

I thank Fernando Guzmán for serving on all of my committees while I was a graduate student, and for attending all the talks which prepared me to do the work in this thesis. I also thank K.J. Reddy for having me in his laboratory, and Susannah Gal for teaching me how to play with DNA.

I thank Arthur Weinberger and Betsy Gümüştöp for cheering me on, and being there when I needed them the most.

Most of all I thank my mother and step-father for the thousand things they have done which made it possible for me to complete this work. They have supported me at every turn during the last eleven years, and anyone who doesn't like all the letters that now come after my name can gripe to them.

Last but not least, I thank my sons Duncan and Ian for all the love and understanding they have given me as they watched me struggle to do my best. They learned along the way to sometimes give up their own desires, so that I might fulfill mine.

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1	Biological Origins . . . . .	1
2	A Brief Outline . . . . .	3
<b>2</b>	<b>Beginnings</b>	<b>5</b>
1	Definitions . . . . .	5
2	The Syntactic Monoid . . . . .	6
3	Splicing Definitions . . . . .	6
4	History and Motivation . . . . .	8
4.1	$H$ systems . . . . .	8
4.2	Extended $H$ systems . . . . .	9
5	More Definitions . . . . .	10
5.1	Respectful and Useful Rules . . . . .	10
5.2	Reflexivity and Constants . . . . .	10
6	Previous Results . . . . .	13
6.1	A New Look at $EH(FIN, FIN) = REG$ . . . . .	15
7	Examples and Conjectures . . . . .	15
<b>3</b>	<b>Rules</b>	<b>17</b>
1	Respecting Rule Sets . . . . .	17
2	A Counterexample . . . . .	19
3	Constants and Reflexive Respecting Rules . . . . .	20

<b>4</b>	<b>Reflexive Splicing Languages</b>	<b>22</b>
1	Preliminaries . . . . .	22
2	Head's Theorem - A Proof . . . . .	24
3	A Characterization . . . . .	26
4	A Non-Reflexive Splicing Language . . . . .	28
<b>5</b>	<b>Main Results</b>	<b>33</b>
1	Simultaneous Pumping Theorem . . . . .	33
2	Main Result and Algorithm . . . . .	37
<b>6</b>	<b>Simple and Null Context Splicing</b>	<b>43</b>
1	A characterization . . . . .	44
<b>7</b>	<b>Semi-Simple Languages</b>	<b>48</b>
1	Semi-simple $\not\subseteq$ Simple . . . . .	48
2	Null Context $\not\subseteq$ Semi-Simple . . . . .	50
3	Arrow Graphs . . . . .	52
4	Arrow Graph Results . . . . .	53
<b>8</b>	<b>Semi-Null Splicing Systems</b>	<b>59</b>
1	Semi-Null $\not\subseteq$ NCH . . . . .	59
2	A Lemma . . . . .	60
3	One Rule Semi-Null Languages . . . . .	61
4	Constants in One Rule Semi-Null Languages . . . . .	63
<b>9</b>	<b>Wet Splicing</b>	<b>67</b>
1	Wet Definitions . . . . .	67
2	Motivation . . . . .	69
3	The Wet Example . . . . .	69
3.1	String Specifications . . . . .	70
4	The Experimental Results . . . . .	70
5	Conclusion . . . . .	73

# Chapter 1

## Introduction

### 1 Biological Origins

In 1987 T. Head presented the concept of *splicing systems* as a mathematical model of biological systems containing double-stranded DNA and enzymes which cut and paste DNA [7]. Head recognized that the action of such enzymes was generative in nature, and he laid the groundwork for a new field of study in formal language theory.

The correspondence between the biological model of splicing and the theoretical model of splicing is in terms of strings and rules. In biological terms, splicing starts with double-stranded DNA(dsDNA), together with a set of cutting and pasting enzymes, all assumed to be in solution with appropriate reagents needed for cutting and pasting to occur. An initial set of dsDNA is transformed by the action of cutting and pasting enzymes, called restriction enzymes and ligases, respectively. Each restriction enzyme has a specific DNA sequence which it *recognizes*, and it cuts dsDNA in a specific pattern if the DNA contains its *recognition site*. Pasting, or ligation, occurs if the pieces generated by cutting have complementary single-stranded *overhangs*. Over time, the system evolves as the iterated cutting and pasting action of the enzymes produces new dsDNA strands.

In its original formulation by Head, a *splicing system* consists of a finite initial set of sequences, with infinitely many copies of each sequence available to the system. This initial set is encoded as a finite language written over a finite alphabet. The

cutting action of each restriction enzyme in the system is encoded as a *splicing rule*. The rules recombine the strings in a way which emulates the biological system. The pasting, or ligation step is not encoded - it is simply assumed to occur whenever it is biologically possible. The set of initial strings together with all the new strings produced by iterated splicing in the system is called the *splicing language* generated by the system.

The original splicing model was concerned with finite initial sets of strings and finite rule sets. It is well known now that any such system generates a splicing language which is regular in terms of the Chomsky hierarchy of classifying languages, [3], [4], [22], [21]. Later generalizations include the notion of systems having initial sets and rule sets residing in any of the families of languages familiar to language theorists, not just the finite ones. We, however, are concerned with languages which arise as the result of splicing finite initial sets with finite rule sets. This follows Head's 1987 paper [7], in which he presented the problem of characterizing such languages.

I became interested in splicing languages as both a tool for better understanding biological phenomena, and as an exercise in formal language theory. I was fascinated by this bridge between the biological sciences and mathematical sciences. I wished to answer the following questions: Is it decideable whether a given regular language is a splicing language? Which regular languages are splicing languages? What properties are common to all splicing languages? Does splicing really work *in vitro*? This thesis holds the results which I have so far obtained in my quest to understand splicing languages.

One of the main problems treated in this thesis was proposed by Head. He challenged his audience in [8] to find an algorithm which takes a regular language  $L \subseteq A^*$  as input, and gives as output the answer to the following question: Is there a finite set of constant factors  $F$  of  $L$  such that  $L \setminus A^*FA^*$  is finite? If so, then Head's work reveals that  $L$  is a reflexive splicing language of a very specific type. Such an algorithm is presented in Chapter 5, using a result developed by D. Pixton.

Constants seem to play an important role in splicing. In particular, reflexive splicing languages have only rules which have constant sites. Also, Head's theorem

characterizes certain splicing languages based on finite constant factor sets. We conjecture that every splicing language contains a constant word. All results in this thesis support the stronger conjecture that every infinite splicing language contains infinitely many constant words.

## 2 A Brief Outline

Chapter 2 contains definitions basic to any study in formal language theory, including the definition of the syntactic monoid. The splicing definition used throughout this thesis is given, as well as other splicing definitions found in the literature. Included are the definitions of  $H$  systems and extended  $H$  systems, as introduced by Paun [16]. The notions of respectful rules, reflexivity, and constants are introduced. This chapter contains a new proof that every regular language is the homomorphic image of a splicing language. This chapter also contains a new proof that regular languages are exactly the set of extended splicing languages generated with finite initial sets and finite rule sets.

Chapter 3 contains results about the types of rule sets which respect regular languages. In particular, this chapter contains a proof that regular languages have regular respecting sets of rules. Also given is a proof that regular languages have regular reflexive sets of respecting rules. In the last section, there is an example demonstrating that there are non-regular languages which have regular respecting rule sets.

Chapter 4 contains an alternate proof of Head's theorem characterizing a language  $L \subseteq A^*$  which has a finite subset  $F$  of constants such that  $L \setminus A^*FA^*$  is finite. The third section of this chapter contains a characterization of the languages residing in the family of reflexive splicing languages. Also in this chapter is the answer to the following question: Is every splicing language which is generated by a finite splicing system a reflexive splicing language? The answer is no, and is shown by counterexample.

Chapter 5 contains the main results of this thesis – namely an algorithm which

determines if a given regular language  $L \subseteq A^*$  has a finite subset  $F$  of constants such that  $L \setminus A^*FA^*$  is finite. Also presented is Pixton's Simultaneous Pumping Theorem, which is central to the main proof of this chapter.

Later chapters focus on splicing systems having rules of particular types. Chapter 6 is a review of simple splicing languages and null context languages. This chapter contains a characterization of simple splicing languages in terms their factors and constant symbols. Strictly locally testable languages are defined. Also in this chapter is a statement of Head's theorem which says that the set of null context languages is precisely the set of strictly locally testable languages.

Semi-simple splicing languages are introduced in Chapter 7. The first section of this chapter contains an example demonstrating that there are semi-simple languages which are not simple. Also given is an example which shows that there exist null context languages which are not semi-simple. In the third section of this chapter, the concept of an arrow graph is developed. The arrow graph is used to prove another main result – namely that every semi-simple language is strictly locally testable.

Semi-null languages are defined in Chapter 8. An example is given to demonstrate that there are semi-null languages which are not null-context, and therefore not strictly locally testable. Conditions under which one rule semi-simple languages are infinite are given. A proof that any one rule semi-null language which is infinite contains infinitely many constant words is presented.

In Chapter 9 holds laboratory results. In 1997 K.J. Reddy and I set out to verify that splicing systems behave *in vitro* as predicted by Head's model. Our results indeed support the claim that *in vitro* splicing systems display the dynamical behavior which Head's model predicts.

# Chapter 2

## Beginnings

We begin with definitions and a review of some of the history of the splicing model. We shall present some of the results which are relevant to the questions in which we are interested. We present the model of splicing that we will use throughout this work. We will discuss this model as it relates to generalized notions of splicing, namely  $H$  systems and extended  $H$  systems, as introduced by Paun. We shall review some results by Head, Paun and Pixton which motivate the focus of this work. We will present a different proof of a result of Paun, namely that every regular language can be generated by an extended  $H$  system having finite initial set and finite rule set. We will also present some examples which support our conjecture that every splicing language contains a constant word. We first need some preliminary definitions.

### 1 Definitions

Let  $A$  denote a finite alphabet, and  $A^*$  the free monoid generated by  $A$ , where the identity element, or empty word, is indicated by the symbol  $1$ . Also, let  $A^+ = A^* \setminus \{1\}$ . Given  $L \subseteq A^*$ , we define  $L^*$  to be the monoid in  $A^*$  generated by the words in  $L$ . If  $L = \{w\}$ , then we will confuse the notation by omitting the set brackets, writing  $L$  simply as  $w$ , and  $L^*$  as  $w^*$ . As expected,  $L^+$  denotes  $L^* \setminus 1$ . Given  $w \in A^*$ , we denote the length of  $w$  by  $|w|$ . Notice that  $|1| = 0$ .

## 2 The Syntactic Monoid

Let  $L \subseteq A^*$ . We define the congruence relation  $\equiv_L$  on  $A^*$  as follows:

$$u \equiv_L v \text{ if and only if } xuy \in L \text{ iff } xvy \in L, \text{ for all } x, y \in A^*.$$

The quotient  $A^*/\equiv_L$  is called the syntactic monoid of  $L$ , and denoted by  $\text{Syn } L$ . The following are well known facts about syntactic monoids of regular languages. The reader is referred to [10], [19] for proofs, if needed.

**Theorem 2.1.** *A language  $L$  is regular if and only if  $\text{Syn } L$  is finite.*

**Theorem 2.2.** *Let  $L \subset A^*$ . Let  $\eta : A^* \rightarrow A^*/\equiv_L$  be the natural mapping from  $A^*$  into  $\text{Syn } L$ . If  $L$  is regular then  $\eta^{-1}(x)$  is regular for all  $x \in \text{Syn } L$ .*

## 3 Splicing Definitions

Let  $A$  denote a finite alphabet. A *splicing rule*, also called a *rule for simplicity*, is a quadruple  $(u, v; u', v')$ , where  $u, v, u'$  and  $v'$  come from  $A^*$ . Given a rule  $r = (u, v; u', v')$  and two strings  $w = xuvy$  and  $w' = x'u'v'y'$  in  $A^*$ , we may *splice*  $w$  and  $w'$  via  $r$ , generating the splicing product  $xuv'y'$ . We will denote this by the following shorthand:

$$(xuvy, x'u'v'y') \vdash_{(u, v; u', v')} xuv'y'$$

If  $z$  is generated by splicing words  $w$  and  $w'$  via  $r$ , then we may write  $(w, w') \vdash_r z$ . If  $r$  is understood, then we may simply write  $(w, w') \vdash z$ . A rule  $r = (u, v; u', v')$  determines the *sites*  $uv$  and  $u'v'$ . Evidently, words  $w$  and  $w'$  can be spliced via  $r$  whenever they contain  $uv$  and  $u'v'$ , respectively. Given  $w, w'$  and  $r$ , notice that there may be more than one possible splicing product.

Given  $L \subseteq A^*$  and a rule  $r$ , we define  $r(L)$  as follows:

$$r(L) = \{z \in A^* \mid \exists w, w' \in L \text{ such that } (w, w') \vdash_r z\}$$

Given an alphabet  $A$  and a set of rules  $S \subseteq (A^*)^4$ , the pair  $\sigma = (A, S)$  is called a *splicing scheme*. Given a splicing scheme  $\sigma = (A, S)$  and a language  $L$ , we define one

iteration of splicing  $L$  via  $\sigma$  as follows:

$$\sigma(L) = \{z \in A^* \mid \exists w, w' \in L \text{ and } r \in S \text{ such that } (w, w') \vdash_r z\}$$

Notice that an equivalent definition of  $\sigma(L)$  may be given:

$$\sigma(L) = \bigcup_{r \in S} r(L)$$

Given a splicing scheme  $\sigma = (A, S)$  and  $I \subseteq A^*$ , we define iterated splicing as follows:

$$\begin{aligned} \sigma^0(I) &= I \\ \sigma^{i+1}(I) &= \sigma^i(I) \cup \sigma(\sigma^i(I)), \quad i \geq 0 \\ \sigma^*(I) &= \bigcup_{i \geq 0} \sigma^i(I). \end{aligned}$$

In other words,  $\sigma^*(I)$  is the smallest language containing  $I$  and closed under splicing with respect to  $\sigma$ . A splicing scheme  $\sigma$  together with its initial set  $I$  constitutes a *splicing system*  $(A, I, S)$ , or just  $(I, \sigma)$ , where  $\sigma = (A, S)$ . The language  $\sigma^*(I)$  is called the *splicing language* generated by  $(I, \sigma)$ . We will also use the notation  $L(A, I, S)$ , or simply  $L(I, \sigma)$  interchangeably with  $\sigma^*(I)$ .

We are expressly interested in splicing systems in which the initial set  $I$  and the rule set  $S$  of  $\sigma$  are finite. It is well known that, for such  $\sigma$  and  $I$ ,  $\sigma^*(I)$  is regular. This result was first attributed to Culik and Harju [4], and reproven by Pixton in [22]. A generalization of this result was then given by Pixton in [21]. We shall discuss this more in Section 4. We shall take the following liberty and use the term *splicing language* to refer to languages of this restricted type, unless we specify otherwise.

**Definition 2.3.** *Given  $L \subseteq A^*$ , we say  $L$  is a **splicing language** if  $L = \sigma^*(I)$  for  $I \subseteq A^*$  and  $\sigma = (A, S)$ , where  $S$  and  $I$  are finite.*

In order to clarify our motivation for taking this definition of splicing, rather than a more general definition, we must review some of the history of the development of the splicing model.

## 4 History and Motivation

As stated in the previous section, splicing was first defined by Head as an operation on strings written over an alphabet  $A$ . The initial set of strings, and the set of splicing rules are both finite in Head's original model. One generalization of splicing, introduced by Paun [16], involves initial sets and rule sets which are not finite. Paun uses a slightly different notation for rule sets, introduced below.

### 4.1 $H$ systems

According to Paun, an  $H$  scheme is a pair  $(A, S)$ , where  $A$  is an alphabet and  $S \subseteq A^* \# A^* \$ A^* \# A^*$  is a set of splicing rules, where the symbols  $\$, \#$  are not in  $A$  [15]. Two words written over  $A$  may be spliced via a rule  $r$  in  $S$  exactly as defined in Section 3, namely,

$$(xuvy, x'u'v'y') \vdash_{(u\#v\$u'\#v')} xuv'y'.$$

Note that the rule set  $S \subseteq A^* \# A^* \$ A^* \# A^*$  can be infinite, and that we may classify  $S$  according to the Chomsky hierarchy or any other classification of languages. We shall use Paun's notation for splicing rules when it is important to treat rule sets as languages. Let us denote by  $FIN$ ,  $REG$ ,  $CS$ , and  $RE$  the families of finite, regular, context-sensitive and recursively enumerable languages, respectively.

We have the following definition due to Paun [16]:

**Definition 2.4.** *Let  $S \in F$ , where  $F$  is a given family of languages. Then the  $H$ -scheme  $\sigma = (A, S)$  is said to be of  $F$  type.*

For two families of languages,  $F_1$  and  $F_2$ , we define  $H(F_1, F_2)$  as follows:

$$H(F_1, F_2) = \{\sigma^*(L) \mid L \in F_1, \sigma = (A, S), \text{ and } S \in F_2\}.$$

It is worthwhile to mention the following facts about  $H$  systems in general, which can be found in [17]:

- (i)  $FIN \subseteq H(FIN, FIN) \subset REG$

(ii)  $REG = H(REG, FIN)$

(iii)  $H(REG, REG)$  contains  $RE$  languages which are not  $CS$ .

The statement that  $H(FIN, FIN) \subset REG$  was first treated by Culik and Harju [3], [4], and reproven by Pixton [21] in the following generalized form:

**Theorem 2.5.** *If  $F$  is a full AFL (a nontrivial family of languages closed under union, concatenation, Kleene +, arbitrary morphisms, inverse morphisms and intersection with regular sets), then  $H(F, FIN) \subseteq F$ .*

The reader who is familiar with the work of Paun and others may be acquainted with results pertaining to a further generalization of splicing systems, called extended  $H$  systems.

## 4.2 Extended $H$ systems

An *extended  $H$  system* is a quadruple

$$\gamma = (A, T, I, S),$$

where  $A$  is an alphabet,  $T \subseteq A$ ,  $I \subseteq A^*$ , and  $S \subseteq A^* \# A^* \$ A^* \# A^*$ , where  $\$, \#$  are symbols not in  $A$ . In  $(A, T, I, S)$ ,  $T$  is the terminal alphabet,  $I$  is the set of initial strings, and  $S$  is the rule set. The *underlying  $H$  scheme* is  $\sigma = (A, S)$ , *augmented* with  $T$  and  $I$ . The language generated by  $\gamma$  is

$$L(\gamma) = \sigma^*(I) \cap T^*,$$

where  $\sigma$  is the underlying  $H$  scheme of  $\gamma$ .

For two families of languages  $F_1$  and  $F_2$ , we denote by  $EH(F_1, F_2)$  the family of languages generated by  $EH$  systems of the form  $\gamma = (A, T, I, S)$  where  $I \in F_1$  and  $S \in F_2$ . One result of interest is this:

$$EH(FIN, FIN) = REG.$$

One direction of the containment ( $\subseteq$ ) is a direct result of the work of Culik and Harju, and Pixton. The other direction was presented by Paun, Rozenberg and Salomaa in [18]. We shall present a proof of the second containment in Section 6.1 of this chapter, using a result due to Head. Before doing this, we shall need a few more definitions.

## 5 More Definitions

The following definitions can be applied to splicing systems which have arbitrary rule sets and initial sets.

### 5.1 Respectful and Useful Rules

The following is a notion concerning rule sets and languages:

**Definition 2.6.** *Given a language  $L \subseteq A^*$  and a splicing rule  $r$ , we say  $r$  **respects**  $L$  if  $r(L) \subseteq L$ .*

In other words, we say a rule  $r$  respects  $L$  if  $L$  is closed under splicing via  $r$ . For  $L \subseteq A^*$ , we define  $R(L)$  as follows:

$$R(L) = \{r \in (A^*)^4 \mid r \text{ respects } L\}.$$

In particular, if  $\sigma = (A, S)$  and  $L = \sigma^*(I)$  for some  $I$ , then every rule in  $S$  respects  $L$ , or simply said,  $S$  respects  $L$ . Put another way, we see that  $S \subseteq R(L)$ . There may, of course, be rules which respect  $L$  which are not included in  $S$ . Notice also that, given a rule  $r$  and a language  $L$ , if  $r$  contains a site which appears in no word in  $L$ , then  $r$  respects  $L$  in a vacuous sense. Let us define the *factor set* of  $L$  as follows:

$$\text{Fac } L = \{w \in A^* \mid \exists x, y \in A^* \text{ such that } xwy \in L\}.$$

Notice that  $1 \in \text{Fac } L$  unless  $L$  is empty.

We shall now define the notion of usefulness.

**Definition 2.7.** *Given  $L \subseteq A^*$ , we define the set of **useful rules** for  $L$  as the set of all rules of the form  $r = (u, v; u', v')$  where  $uv$  and  $u'v'$  are in  $\text{Fac } L$ .*

Notice that useful rules for  $L$  may or may not respect  $L$ .

### 5.2 Reflexivity and Constants

The following definition is due to Head [8].

**Definition 2.8.** Given a rule set  $S$ , we say  $S$  is **reflexive** if  $(u, v; u', v') \in S$  implies that  $(u, v; u, v)$  and  $(u', v'; u', v')$  are also in  $S$ . If  $\sigma = (A, S)$ , and  $S$  is reflexive, we say  $\sigma$  is a **reflexive splicing scheme**.

**Definition 2.9.**  $L$  is a **reflexive splicing language** if there exists a reflexive splicing scheme  $\sigma = (A, S)$  and a set  $I$  such that  $L = \sigma^*(I)$ . We say  $(I, \sigma)$  is a **reflexive splicing system**.

**Definition 2.10.** Given a finite set  $S$  of rules, the **reflexive closure** of  $S$ , denoted  $\bar{S}$ , is defined as follows:

$$\bar{S} = S \cup \{(u, v; u, v) \mid \exists u', v' \text{ such that } (u, v; u', v') \in \bar{S} \text{ or } (u', v'; u, v) \in \bar{S}\}.$$

Note that  $\bar{S}$  is both finite and reflexive. Further, if  $X$  is a reflexive rule set containing  $S$ , then  $X$  must contain  $\bar{S}$ . So  $\bar{S}$  is the smallest reflexive set of rules containing  $S$ .

**Definition 2.11.** If  $\sigma = (A, I, S)$  and  $\bar{S}$  is the reflexive closure of  $S$ , then we denote the system  $(A, I, \bar{S})$  by  $\bar{\sigma}$ .

The notion of a *constant* word was introduced by Schützenberger in [23].

**Definition 2.12.** Given  $L \subseteq A^*$  and  $c \in A^*$ , we say  $c$  is a **constant** of  $L$  if for all  $x, y, x'$  and  $y'$  in  $A^*$ ,  $xcy, x'cy' \in L$  implies  $xcy' \in L$ .

Given  $L \subseteq A^*$ , we denote by  $C(L)$  the set of all constants of  $L$ :

$$C(L) = \{c \in A^* \mid c \text{ is a constant of } L\}$$

**Lemma 2.13.** Let  $L \subseteq A^*$ , and let  $c \in C(L)$ . Then  $A^*cA^* \subseteq C(L)$ .

*Proof.* The proof is left to the reader. □

Notice that if a string  $c \in A^+$  does not appear as a factor of any word in  $L$ , then  $c$  vacuously satisfies the definition of a constant word of  $L$ . We therefore define the set of constants appearing as factors of  $L$  as follows:

$$C_{fac}(L) = \text{Fac } L \cap C(L)$$

**Lemma 2.14.** *Let  $\sigma = (A, S)$  be a reflexive splicing scheme. Then for any  $I$ , every site in  $S$  is a constant of  $L(I, \sigma)$ .*

*Proof.* Let  $I \subset A^*$  be given. Let  $r = (u, v; u', v')$  be a rule in  $S$ . Consider words  $xuvy$  and  $x'uvy'$  in  $L(I, \sigma)$ . Since  $S$  is reflexive, the rules  $r_1 = (u, v; u, v)$  and  $r_2 = (u', v'; u', v')$  are also in  $S$ . We may splice as follows via  $r_1$ :  $(xuvy, x'uvy') \vdash_{r_1} xuvy' \in L$ . We may also splice using  $x'uvy'$  first, thus obtaining  $x'uvy \in L(I, \sigma)$ . So site  $uv$  is a constant of  $L(I, \sigma)$ . In a similar fashion, we may splice using  $r_2$ , and show  $u'v'$  is a constant of  $L(I, \sigma)$ .  $\square$

**Definition 2.15.** *Let  $L \subseteq A^*$ . The rule  $r = (u, v; u', v')$  is called a **reflexive rule with respect to  $L$**  if site  $uv$  and site  $u'v'$  are both constants of  $L$ .*

**Definition 2.16.** *Let  $S$  be a set of rules, and  $L \subseteq A^*$ . Then we say  $S$  is **reflexive with respect to  $L$**  if all rules in  $S$  are reflexive with respect to  $L$ .*

**Note 2.17.** *Let  $\sigma = (A, S)$  be a reflexive splicing scheme. Let  $I \subseteq A^*$ . Then  $S$  is reflexive with respect to  $\sigma^*(I)$ .*

*Proof.* Let  $r = (u, v; u', v') \in S$ . Consider site  $uv$ . Let  $xuvy$  and  $x'uvy'$  be words in  $\sigma^*(I)$ . Since  $S$  is reflexive, the rule  $(u, v; u, v)$  is in  $S$ . Splicing, we see  $xuvy'$  is in  $\sigma^*(I)$ . So  $uv$  is a constant of  $\sigma^*(I)$ . In a similar fashion, we see  $u'v'$  is also a constant of  $\sigma^*(I)$ . Thus  $S$  is reflexive with respect to  $\sigma^*(I)$ .  $\square$

**Proposition 2.18.** *Let  $\sigma = (A, S)$  be a splicing scheme, and  $I \subset A^*$ . Let  $\bar{\sigma}$  be the reflexive closure of  $\sigma$ . If each rule in  $S$  is reflexive with respect to  $\sigma^*(I)$ , then  $\sigma^*(I) = \bar{\sigma}^*(I)$ .*

*Proof.* Since  $S \subseteq \bar{S}$ , it is clear that  $\sigma^*(I) \subseteq \bar{\sigma}^*(I)$ . For the other inclusion, we shall induct on  $k$ , where  $w \in \bar{\sigma}^k(I)$ . For the basis, note  $\bar{\sigma}^0(I) = I \subseteq \sigma^*(I)$ . Assume the hypothesis is true for all  $i < k$ , where  $k \geq 1$ . If  $w \in \bar{\sigma}^k(I) \setminus I$ , then there are words  $z = xuvy$  and  $z' = x'u'v'y'$  in  $\bar{\sigma}^{k-1}(I)$ , and a rule  $r = (u, v; u', v') \in \bar{S}$  such that

$$(z = xuvy, z' = x'u'v'y') \vdash_r xuv'y' = w.$$

If  $r \in S$ , then we are done, since  $z$  and  $z'$  are in  $\sigma^*(I)$  by the induction hypothesis. Otherwise,  $r \in \bar{S} \setminus S$ , and we note that  $r = (u, v; u, v)$  where  $uv$  is a site of a rule in  $S$ . Since  $S$  is a reflexive set of rules with respect to  $\sigma^*(I)$ ,  $uv$  is a constant of  $\sigma^*(I)$ . Since  $z = xuvy$  and  $z' = x'uvy'$  are in  $\sigma^*(I)$  by induction hypothesis, the word  $xuvy' \in \sigma^*(I)$  by the definition of constant. Since  $v = v'$ ,  $w = xuv'y' = xuvy' \in \sigma^*(I)$ .  $\square$

The following simple lemma about constants will often be used as the definition of constants of splicing languages.

**Lemma 2.19.** *The string  $u \in A^*$  is a constant of  $L$  if and only if the rule  $(u, 1; u, 1)$  respects  $L$ .*

The proof of the lemma an obvious application of the definitions. A more general version of Lemma 2.19, also easily verified from the definitions, has this formulation:

**Lemma 2.20.** *Let  $u, v \in A^*$ . The string  $uv$  is a constant of  $L$  if and only if the rule  $(u, v; u, v)$  respects  $L$ .*

Notice that Lemma 2.19 can be modified to the following statement about constant factors:

**Lemma 2.21.** *The rule  $(u, 1; u, 1)$  is useful in  $L$  and respects  $L$  if and only if  $u$  is a constant factor in  $L$ .*

## 6 Previous Results

The following result due to Head appeared in *Splicing languages generated with one sided context* [8], and is central to this thesis:

**Theorem 2.22.** *Let  $L \subseteq A^*$  be a regular language.  $L$  is a splicing language generated by a reflexive splicing system in  $H(FIN, FIN)$  in which each rule has either the form  $(u, 1; v, 1)$  or the form  $(1, u; 1, v)$  if and only if there exists a finite set  $F$  of constant factors of  $L$  such that  $L \setminus A^*FA^*$  is finite.*

We shall give a proof of this theorem in Chapter 4 which is slightly different from the one given by Head. In [8], Head presented the problem of finding an algorithm to determine the existence of such a finite set  $F$ , given an arbitrary regular language  $L$ . We shall present such an algorithm in Chapter 5.

Another theorem due to Head is the following:

**Theorem 2.23.** *Let  $L \subseteq A^*$  be a regular language. Let  $c$  be a symbol not in  $A$ . Then the language  $cL$  is a splicing language in  $H(FIN, FIN)$ .*

While we omit the proof, we will point out that it is constructive. In particular, all rules may be chosen to be of the form  $(cuv, 1; cu, 1)$ , where  $u$  and  $v$  are in  $A^*$ . In fact, the language  $cL$  has the form of a reflexive splicing language of the sort described in Theorem 2.22. The finite set of constants  $F$  such that  $cL \setminus A^*FA^*$  is finite is simply the set  $\{c\}$ .

We have the following corollary:

**Corollary 2.24.** *Let  $L \subseteq A^*$  be a regular language, and let  $c \notin A$ . Then  $L \cup cL$  is in  $H(FIN, FIN)$ .*

*Proof.* If one takes  $(A \cup \{c\}, I, S)$  to be a splicing system which generates  $cL$ , where the rules in  $S$  are constructed according to the algorithm in the proof of Theorem 2.23, one can append one more rule to  $S$ , namely  $(1, c; c, 1)$ . This new rule acts as only a cutting rule, and generates exactly the words in  $L$  from the words in  $cL$ . Taking  $S' = S \cup \{(1, c; c, 1)\}$ , and  $A' = A \cup \{c\}$ , we have the new splicing system  $(A', I, S')$ . It is clear that  $L(A', I, S') = cL \cup L$ . □

Another corollary of Theorem 2.23 is worth mention, originally due to Gatterdam [6]:

**Corollary 2.25.** *Every regular language is the homomorphic image of a splicing language in  $H(FIN, FIN)$ .*

*Proof.* Given regular  $L$ , one has the splicing language  $cL$ .  $L$  is the image of  $cL$  by mapping every letter in  $A$  to itself, and the symbol  $c$  to the empty word.  $\square$

## 6.1 A New Look at $EH(FIN, FIN) = REG$

We now have the tools with which to reprove the following, found in [16]:

**Theorem 2.26.**  $EH(FIN, FIN) = REG$

*Proof.* Let  $L \in EH(FIN, FIN)$ . Then  $L = L(\gamma)$  for  $\gamma = (A, T, I, S)$ . From the definition of  $EH$  systems, it is clear that  $L = \sigma^*(I) \cap T^*$  for  $\sigma = (A, S)$ , and that  $\sigma^*(I)$  is an  $H$  language in  $H(FIN, FIN)$ . Theorem 2.5 says that  $\sigma^*(I)$  is regular. Hence  $L$  is regular. Now let  $L \subseteq A^*$  be a regular language. By Theorem 2.23, we have  $cL \in H(FIN, FIN)$ . Let  $(I, S)$  be the  $H$  system which generates  $L' = cL \cup L$ , as seen in Corollary 2.24. Then the  $EH$  system  $\gamma = (A \cup \{c\}, A, I, S)$  generates  $L' \cap A^* = L$ .  $\square$

## 7 Examples and Conjectures

Again, we emphasize that throughout this paper, unless otherwise specified, our use of the terms splicing scheme and splicing system should be understood to denote exactly those schemes and systems which have finite rule sets and finite initial sets.

The simplest first example of a splicing language is  $L = a^+$ , generated by the system  $(A, I, S) = (\{a\}, \{aa\}, \{(a, 1; a, 1)\})$ . It is easy to see that any free monoid over a finite set of generators is a splicing language. It is well known that not every regular language is a splicing language. The most commonly cited example is  $L = (aa)^*$ , first mentioned by Gatterdam in [6]. The proof follows from the fact that the only rules which are useful in  $L$  must have sites in  $a^*$ . However, none of these rules respect  $L$ , so  $L$  is not a splicing language.

Our first examples bring to light a question about constants and splicing languages. Notice that every word in the splicing language  $L = a^*$  is constant in  $L$ . In  $L = (aa)^*$ , no words are constant. It is natural to ask if splicing languages contain only

constant words. We know that there are splicing languages containing words which are not constant, and there are non-splicing languages in which every word is constant. Consider the following:

$L = b(aa)^* \cup (aa)^*$  is a splicing language, by Corollary 2.24, but no word in  $(aa)^*$  is constant in  $L$ .  $L = a^*ba^*ba^*$  is not a splicing language, as we shall see in Chapter 4, and every word in  $L$  is constant in  $L$ .

So far, all of our examples of splicing languages have constant words. Hence we conjecture the following:

**Conjecture 2.27.** *If  $L$  is a non-empty splicing language, then  $L$  contains a constant word.*

If  $L$  is finite, then  $L$  has a constant word – take for instance a longest word in  $L$ . In fact, for an infinite splicing language  $L$ , we conjecture:

**Conjecture 2.28.** *If  $L$  is an infinite splicing language, then  $L$  has an infinite set of constant factors.*

# Chapter 3

## Rules

In this chapter, we shall present some interesting results about regular languages, and their respecting rule sets. Given a language  $L \subseteq A^*$ , we can consider the set  $R(L)$  of all splicing rules which respect  $L$  as a subset of  $A^* \# A^* \$ A^* \# A^*$ , using the notation of Paun. It turns out that the syntactic monoid of a language is a useful tool when one wishes to investigate  $R(L)$ . In this first part of this chapter we will show that rules which respect a given language also respect, in a sense, the syntactic monoid of that language. We will then show that  $R(L)$  for a regular language  $L$  is regular. We shall also show by example that there are non-regular languages which have regular respecting rule sets.

### 1 Respecting Rule Sets

Recall the definition of respecting rule: A rule  $r$  respects  $L$  if  $L$  is closed under splicing via  $r$ . Let  $R(L)$  denote the set of all rules which respect  $L$ ,  $\eta$  denote the natural map from  $A^*$  into  $\text{Syn } L$ , and  $\equiv_L$  denote the syntactic congruence on  $A^*$ , as usual.

We have first a lemma about rules and syntactic monoids:

**Lemma 3.1.** *Let  $L \subseteq A^*$ . Let  $r = u_1 \# u_2 \$ u_3 \# u_4 \in R(L)$ . Let  $\bar{u}_1, \bar{u}_2, \bar{u}_3, \bar{u}_4 \in A^*$  be such that  $\bar{u}_i \equiv_L u_i$ ,  $1 \leq i \leq 4$ . Then  $\bar{r} = \bar{u}_1 \# \bar{u}_2 \$ \bar{u}_3 \# \bar{u}_4 \in R(L)$ .*

*Proof.* Let  $z = x\bar{u}_1\bar{u}_2y$  and  $z' = x'\bar{u}_3\bar{u}_4y'$  be words in  $L$ . We wish to show that the splicing product  $x\bar{u}_1\bar{u}_4y'$  is in  $L$ . Since  $\bar{u}_1\bar{u}_2 \equiv_L u_1u_2$ ,  $x\bar{u}_1\bar{u}_2y \in L$  implies  $xu_1u_2y \in L$ .

Similarly,  $x'\bar{u}_3\bar{u}_4y' \in L$  implies  $x'u_3u_4y' \in L$ . Since  $r = u_1\#u_2\$u_3\#u_4 \in R(L)$ , the splicing product  $xu_1u_4y' \in L$ . Now  $u_1u_4 \equiv_L \bar{u}_1\bar{u}_4$  implies  $x\bar{u}_1\bar{u}_4y' \in L$ , as desired.  $\square$

Now we can prove the main theorem of this chapter for regular languages:

**Theorem 3.2.** *Let  $L$  be a regular language. Then  $R(L)$  is regular.*

*Proof.* Let  $L \subseteq A^*$  be a regular language. Let  $m = \text{Card}(\text{Syn } L)$ . Let us denote the syntactic classes in  $A^*$  of  $\text{Syn } L$  as  $U_i, i \in \{1, \dots, m\}$ . Notice that, for each  $U_i$ ,  $1 \leq i \leq m$ , we may choose a representative  $u_i$  in  $A^*$  such that  $U_i = \eta^{-1}(\eta(u_i))$ . By Lemma 2.2, the  $U_i$  are regular. For each quadruple  $(u_i, u_j, u_k, u_l)$ ,  $1 \leq i, j, k, l \leq m$ , if  $u_i\#u_j\$u_k\#u_l \in R(L)$ , then by Lemma 3.1 the entire set  $\{U_i\#U_j\$U_k\#U_l\} \subseteq R(L)$ . So  $R(L) = \bigcup \{U_i\#U_j\$U_k\#U_l\}$ , where the union runs over the finite set of all quadruples  $(U_i, U_j, U_k, U_l)$  for which the rule  $r = u_i\#u_j\$u_k\#u_l \in R(L)$ . Thus  $R(L)$  is regular.  $\square$

Because some of the rules in  $R(L)$  may not be useful, we define the set of all *useful and respectful* rules for  $L$  as follows:

$$R_{fac}(L) = \{u\#v\$u'\#v' \in R(L) \mid uv, u'v' \in \text{Fac } L\}.$$

Now we have the natural question: If  $L$  is regular, is  $R_{fac}(L)$  regular? The answer is found in a simple corollary to Theorem 3.2.

**Corollary 3.3.** *If  $L$  is regular, then  $R_{fac}(L)$  is regular.*

*Proof.* Again, let  $\eta$  be the map from  $A^*$  into  $\text{Syn } L$ ,  $m = \text{Card}(\text{Syn } L)$ , and let  $u_i$  be a representative of the class  $U_i \subseteq A^*$  for  $1 \leq i \leq m$ . Consider every quadruple  $(U_i, U_j, U_k, U_l)$  for which the rule  $r = u_i\#u_j\$u_k\#u_l \in R(L)$ ,  $1 \leq i, j, k, l \leq m$ . We wish to select only those quadruples for which  $u_iu_j$  and  $u_ku_l$  are in  $\text{Fac } L$ . By the syntactic congruence, if  $u_i \in \text{Fac } L$ , then  $U_i \subseteq \text{Fac } L$ . So in fact, we have:

$$R_{fac}(L) = \bigcup \{U_i\#U_j\$U_k\#U_l \mid u_iu_j, u_ku_l \in \text{Fac } L \text{ and } u_i\#u_j\$u_k\#u_l \in R(L)\}, \quad 1 \leq i, j, k, l \leq m.$$

$\square$

We naturally ask: If  $R(L)$  is regular, is  $L$  regular? The answer is no, which we will demonstrate by a counterexample.

## 2 A Counterexample

We denote the set  $U_{fac}(L)$  of all useful rules in  $L$  as follows:

$$U_{fac}(L) = \{u\#v\$u'\#v' \mid uv, u'v' \in \text{Fac } L\}.$$

Notice that useful rules in  $L$  are not necessarily respecting rules in  $L$ . We define the set  $N(L)$  of *useless rules* for  $L$  also:

$$N(L) = A^* \# A^* \$ A^* \# A^* \setminus U_{fac}(L)$$

Now we have the following note:

**Note 3.4.**  $R(L) = N(L) \cup R_{fac}(L)$ .

*Proof.* The note is clear when one notices that useless rules respect  $L$  in a vacuous sense. □

Let  $a \in A$  and  $x \in A^*$ . We shall use the following notation:

$$|x|_a = \text{the number of occurrences of the symbol } a \text{ in } x.$$

We now have the following proposition:

**Proposition 3.5.** *There exists a non-regular language  $L$  for which  $R(L)$  is regular.*

*Proof.* Consider the non-regular language  $L = \{a^n b^n \mid n \geq 0\}$ . We have first a claim:

**Claim 3.6.** *The set  $R_{fac}(L)$  of useful rules which respect  $L$  is empty.*

*Proof.* Consider any rule  $r = u\#v\$u'\#v'$  which might be in  $R_{fac}(L)$ . The sites  $uv$  and  $u'v'$  must appear as factors of words in  $L$ , since  $r$  is useful. So the sites in  $r$  have the form  $a^i b^j$  where  $0 \leq i, j$ . Since  $r$  respects  $L$ ,  $uv'$  is also a factor appearing in  $L$ , and must also have the form  $a^m b^n$  for some  $m, n \geq 0$ .

**case i :** If  $u = a^i$  and  $v' = a^k b^j$ , then  $v = a^s b^t$  and  $u' = a^l$  for some integers  $s, t$  and  $l$ . There are words  $z = xuvy = x a^i a^s b^t y$  and  $z' = x' u' v' y' = x' a^l a^k b^j y'$  in  $L$  such that  $|x|_a > j + |y'|_b$ . Thus the splicing product  $x a^{i+k} b^j y'$  of  $z$  and  $z'$  via  $r$  is not in  $L$ .

**case ii:** If  $u = a^i b^j$  for  $j > 0$ , then  $v' = b^k$ , since  $r$  is useful and respects  $L$ . So  $v = b^l$  and  $u' = a^s b^t$  for some integers  $l, s$  and  $t$ . Here we can find  $z = x a^i b^j b^l y$  and  $z' = x' a^s b^t b^k y'$  in  $L$  such that  $|x|_a + i > j + k + |y'|_b$ . Thus the splicing product  $x a^i b^j b^k y'$  of  $z$  and  $z'$  via  $r$  is not in  $L$ .

□

Let us define the set  $\mathcal{S}_{fac}(L) \subseteq A^* \# A^*$  of *useful rule sites in  $L$*  as follows:

$$\mathcal{S}_{fac}(L) = \{u \# v \mid uv \in \text{Fac } L\}.$$

Now let  $h$  be the erasing homomorphism defined from  $(A \cup \{\#\})^*$  into  $A^*$  by the rule  $h(a) = a$  for  $a \in A$  and  $h(\#) = 1$ . Notice then the set  $\mathcal{S}_{fac}(L)$  of useful rule sites of  $L$  can be described as:

$$\mathcal{S}_{fac}(L) = A^* \# A^* \cap h^{-1}(\text{Fac } L).$$

Since  $\text{Fac } L = a^* b^*$  is regular, we see that  $\mathcal{S}_{fac}(L)$  is regular. We have also that the set  $U_{fac}(L)$  of useful rules for  $L$  can be written in the form:

$$U_{fac}(L) = \mathcal{S}_{fac}(L) \$ \mathcal{S}_{fac}(L)$$

Since  $\mathcal{S}_{fac}(L)$  is regular, we see that  $U_{fac}(L)$  is regular. Now we recall that  $N(L) = A^* \# A^* \$ A^* \# A^* \setminus U_{fac}(L)$ , and we have immediately that  $N(L)$  is regular. Further, by Claim 3.6,  $R_{fac}(L)$  is empty, hence regular. So  $R(L) = N(L) \cup R_{fac}(L)$  is regular. □

### 3 Constants and Reflexive Respecting Rules

Let  $L \subseteq A^*$ . Recall  $C(L)$  denotes the set of constants of  $L$ , and  $C_{fac}(L)$  denotes the set of constant factors of  $L$ . In the following note, we shall demonstrate that the property of being a constant of  $L$  in some sense respects  $\text{Syn } L$ .

**Note 3.7.** *Let  $L \subseteq A^*$ , and let  $c \in C(L)$ . If  $c' \equiv_L c$ , then  $c' \in C(L)$  as well.*

*Proof.* Let  $c \in C(L)$ , and  $c' \in A^*$  be such that  $c' \equiv_L c$ . Consider  $x c' y$  and  $x' c' y'$  in  $L$ , for some  $x, y \in A^*$ . Since  $c' \equiv_L c$ , we see that  $x c y$  and  $x' c y'$  are also in  $L$ . Now  $c \in C(L)$  implies  $x c y' \in L$ . Again, since  $c \equiv_L c'$ , we see  $x c' y' \in L$ . So we see  $c' \in C(L)$ . □

**Lemma 3.8.** *Let  $L \subseteq A^*$  be a regular language. Then  $C(L)$  is regular.*

*Proof.* By Note 3.7,  $C(L)$  can be written as the union of classes in  $\text{Syn } L$ . So  $C(L)$  is regular by Theorem 2.2.  $\square$

Recall that a reflexive rule  $r$  for  $L$  is a rule having sites which are constants of  $L$ . We define the set  $R^{refl}(L)$  of *reflexive respecting rules* for  $L$  as follows:

$$R^{refl}(L) = \{u\#v\$u'\#v' \in R(L) \mid uv, u'v' \in C(L)\}.$$

In other words,  $R^{refl}(L)$  is the set of rules in  $R(L)$  which are reflexive rules for  $L$ . We have the following proposition, which is basically a corollary of Theorem 3.2:

**Proposition 3.9.** *Let  $L \subseteq A^*$  be a regular language. Then  $R^{refl}(L)$  is regular.*

*Proof.* Let us define the set  $\mathcal{S}^{refl}(L) \subseteq A^*\#A^*$  of *reflexive rule sites* of  $L$  as follows:

$$\mathcal{S}^{refl}(L) = \{u\#v \in A^* \mid uv \in C(L)\}.$$

Let  $h$  be the erasing homomorphism from  $(A \cup \{\#\})^*$  into  $A^*$  defined as before:  $h(a) = a$  for  $a \in A$  and  $h(\#) = 1$ . We may rewrite  $\mathcal{S}^{refl}(L)$  as follows:

$$(\star) \quad \mathcal{S}^{refl}(L) = A^*\#A^* \cap h^{-1}(C(L))$$

We may now write  $R^{refl}(L)$  as follows:

$$R^{refl}(L) = \{\mathcal{S}^{refl}(L) \$ \mathcal{S}^{refl}(L)\} \cap R(L)$$

By Lemma 3.8, we know that  $C(L)$  is regular. So we have that  $\mathcal{S}^{refl}(L)$  is regular, by Equation  $\star$ . We know by Theorem 3.2 that  $R(L)$  is regular. Hence  $R^{refl}(L)$  is regular.  $\square$

# Chapter 4

## Reflexive Splicing Languages

In this chapter, we shall focus on the reflexive splicing languages. In the first section of this chapter we present a new constructive proof of Theorem 4.6 which uses the concept of the syntactic monoid. In Section 3, we give a characterization of reflexive splicing languages in terms of one iteration of splicing in a reflexive system. This characterization uses the construction given in the new proof of Theorem 4.6. In the last section of this chapter, we shall show by example that not every splicing language in  $H(FIN, FIN)$  is necessarily a reflexive splicing language.

First, some definitions and other preliminaries:

### 1 Preliminaries

Recall that, given a language  $L \subseteq A^*$ , and a rule  $r$ , we define  $r(L)$  to be the set of all words which can be obtained by exactly one iteration of splicing in  $L$  via  $r$ :

$$r(L) = \{w \in A^* \mid \exists x, y \in L \text{ such that } (x, y) \vdash_r w\}.$$

Also, if  $\sigma = (A, S)$ , we define  $\sigma(L)$ :

$$\sigma(L) = \bigcup_{r \in S} r(L)$$

Recall also the following definitions:

**Definition 4.1.** Given a rule set  $S$ , we say  $S$  is reflexive if  $(u, v; u', v') \in S$  implies that  $(u, v; u, v)$  and  $(u', v'; u', v')$  are also in  $S$ . If  $\sigma = (A, S)$ , and  $S$  is reflexive, we say  $\sigma$  is a reflexive splicing scheme.

Given  $I \subseteq A^*$  and a reflexive splicing scheme  $\sigma = (A, S)$ , we say that  $(A, I, S)$  is a reflexive splicing system. A language is a reflexive splicing language if there is a reflexive splicing system  $(A, I, S)$  such that  $L = \sigma^*(I)$  for  $\sigma = (A, S)$ .

**Definition 4.2.** Given a language  $L \subseteq A^*$  and a splicing rule  $r$ , we say  $r$  respects  $L$  if  $r(L) \subseteq L$ .

Given  $L \subseteq A^*$ , recall that  $R(L)$  denotes the set of all rules which respect  $L$ . Recall also that the set  $C(L)$  denotes the set of constants of  $L$ .

**Lemma 4.3.** Let  $L \subseteq A^*$ , and let  $\equiv_L$  denote the syntactic congruence of  $L$  on  $A^*$ . Let  $x, y, c \in A^*$ . If  $x \equiv_L y$ , and  $c \in C(L)$ , then the rules  $r = (cx, 1; cy, 1)$  and  $r' = (1, xc; 1, yc)$  are in  $R(L)$ .

*Proof.* Let  $\alpha x \beta$  and  $\bar{\alpha} c y \bar{\beta}$  be words in  $L$ . A splice via  $r$  produces the following:

$$(\alpha x \beta, \bar{\alpha} c y \bar{\beta}) \vdash_r \alpha c x \bar{\beta}.$$

We wish to verify that this splicing product is in  $L$ . Since  $\alpha x \beta$  and  $\bar{\alpha} c y \bar{\beta}$  are in  $L$ , and  $c$  is constant,  $\alpha c y \bar{\beta}$  is in  $L$ . Since  $x \equiv_L y$ ,  $\alpha c x \bar{\beta}$  is in  $L$ . A symmetric argument shows that a word resulting from a splice of words in  $L$  via  $r'$  is also in  $L$ .  $\square$

It is convenient for proofs to note the following [Pixton]:

**Note 4.4.** Let  $I, L \subseteq A^*$  and let  $\sigma = (A, S)$  be a splicing scheme. The statement  $\sigma^*(I) \subseteq L$  is true if the following conditions hold:

(i)  $I \subseteq L$

(ii)  $\sigma(L) \subseteq L$

Note that condition (ii) is equivalent to  $\sigma^*(L) \subseteq L$ , or simply the condition that  $L$  is closed under splicing via rules in  $S$ .

Finally, we include a statement of the famous Pigeon Hole Principle:

**Lemma 4.5.** (*Pigeon Hole Principle*)

Let  $S$  be a finite collection of  $m$  pairwise disjoint sets, where  $m \geq 0$ . Let  $Y$  be a set such that  $Y \subset \bigcup_{X \in S} X$ . Let  $q \geq 0$ . If  $\text{Card}(Y) \geq m(q - 1) + 1$ , then there exists a set  $X \in S$  which contains at least  $q$  elements from  $Y$ .

*Proof.* The proof is left to the reader. □

## 2 Head's Theorem - A Proof

We offer a different proof of Theorem 4.6, originally due to Head [8].

**Theorem 4.6.** *Let  $L \subseteq A^*$  be a regular language.  $L$  is a splicing language generated by a reflexive splicing system in  $H(\text{FIN}, \text{FIN})$  in which each rule is either of the form  $(u, 1; v, 1)$  or of the form  $(1, u; 1, v)$  if and only if there exists a finite set  $F$  of constant factors in  $L$  such that  $L \setminus A^*FA^*$  is finite.*

*Proof.* Let  $L = L(A, I, S)$ , where  $I$  and  $S$  are finite,  $S$  is reflexive, and each rule in  $S$  is either of the form  $(u, 1; v, 1)$  or of the form  $(1, u; 1, v)$ . Let

$$U = \{u \in A^* \mid u \text{ is a site in } S\}.$$

Note  $U$  is finite since  $S$  is finite, and  $U \subseteq C(L)$  since  $S$  is reflexive. Any word in  $L \setminus I$  is obtained by splicing in  $L$ , and must contain a site  $u \in U$  as a factor. Since  $I$  is finite,  $L \setminus A^*UA^*$  is finite.

Let  $L \subseteq A^*$  be a regular language for which there exists a finite subset  $F \subseteq C(L)$  such that  $L \setminus A^*FA^*$  is finite. Let  $\text{Syn } L$  denote the syntactic monoid of  $L$ , and  $\equiv_L$  the syntactic congruence, as usual. Since  $L$  is regular,  $\text{Card}(\text{Syn } L)$  is finite (Theorem 2.1). Let

$$M = \text{Card}(\text{Syn } L),$$

$$N = 2M + 1,$$

$$m_1 = \max\{|x| \mid x \in F\},$$

$$m_2 = \max\{|w| \mid w \in L \setminus A^*FA^*\},$$

$$K = \max\{m_1, m_2\}.$$

We have the following lemma, using the integers just defined:

**Lemma 4.7.** *Let  $x = a_1 a_2 \dots a_N \in A^*$ . Then there exist integers  $i, j, k$ , such that  $1 \leq i < j < k \leq N$  and for which the following holds:*

$$a_i \dots a_N \equiv_L a_j \dots a_N \equiv_L a_k \dots a_N$$

*Proof.* Since  $\text{Card}(\text{Syn } L) = M$ , and  $N = 2M + 1$ , the Pigeon Hole Principle dictates that such  $i, j$  and  $k$  exist.  $\square$

We construct the splicing scheme  $\sigma = (A, S)$  as follows:

For each  $c \in F$ , define the following sets of rules:

$$\begin{aligned} S_1^c &= \{(1, xc; 1, yc) \mid x, y \in A^*, |x|, |y| \leq N, \text{ and } x \equiv_L y\} \\ S_2^c &= \{(cx, 1; cy, 1) \mid x, y \in A^*, |x|, |y| \leq N, \text{ and } x \equiv_L y\} \end{aligned}$$

Let

$$\begin{aligned} I &= \{w \in L \mid |w| < K + 2N\}, \\ S &= \bigcup_{c \in F} S_1^c \cup \bigcup_{c \in F} S_2^c, \quad \text{and} \\ \sigma &= (A, I, S). \end{aligned}$$

First, notice that every site in  $S$  is constant in  $L$  (Lemma 2.13). Since  $x \equiv_L x$  for all  $x \in A^*$ , the set  $S$  is a reflexive set of rules. Since  $|c|$  is bounded above by  $K$ , and  $|x|, |y|$  are bounded above by  $N$ , we see that  $S$  is finite as well. Clearly,  $I$  is also finite. We need only prove the claim below:

**Claim 4.8.**  $L = \sigma^*(I)$ .

*Proof.* ( $\sigma^*(I) \subseteq L$ ): We use the logic of Note 4.4. If  $w \in I$ , then  $w \in L$  by the definition of  $I$ . By Lemma 4.3,  $S \subseteq R(L)$ . Thus  $S(L) \subseteq L$ , and  $\sigma^*(I) \subseteq L$ .

( $L \subseteq \sigma^*(I)$ ): Assume otherwise. Let  $w$  be a shortest word in  $L \setminus \sigma^*(I)$ . Since  $w \notin I$ ,  $|w| \geq K + 2N$ . Since  $|w| \geq K$ ,  $w \notin L \setminus A^*FA^*$ . So  $w = w_1cw_2$  for some  $c \in F$  and  $w_1, w_2 \in A^*$ . Further, since  $|w| \geq K + 2N$ , and  $|c| \leq K$ , we see that at least one of  $w_1$  or  $w_2$  has length greater than  $N$ . Let us say that  $|w_1| = \bar{N} > N$ . So

$w_1 = \alpha a_1 \dots a_N$ , where  $|\alpha| = \bar{N} - N$ . By Lemma 4.7, there are integers  $i, j, k$  where  $1 \leq i < j < k \leq N$  and such that the following holds:

$$a_i \dots a_N \equiv_L a_j \dots a_N \equiv_L a_k \dots a_N$$

We define rule  $r$ :

$$r = (1, a_k \dots a_N c; 1, a_j \dots a_k \dots a_N c)$$

By the construction of  $S$ , we see that the rule  $r$  is in  $S$ .

Since  $w = \alpha a_1 \dots a_i \dots a_j \dots a_k \dots a_N c w_2 \in L$ , we have

$$(a_j \dots a_N) \equiv_L (a_k \dots a_N) \implies (\alpha a_1 \dots a_i \dots a_{j-1})(a_k \dots a_N) c w_2 \in L \text{ and}$$

$$(a_i \dots a_N) \equiv_L (a_j \dots a_N) \implies (\alpha a_1 \dots a_{i-1})(a_j \dots a_k \dots a_N) c w_2 \in L.$$

Further, both of these words are shorter than  $w = w_1 c w_2$ , since  $i < j < k$ . So these words are in  $\sigma^*(I)$ , and we may splice them via  $r$  as follows:

$$\begin{aligned} & ((\alpha a_1 \dots a_i \dots a_{j-1}) a_k \dots a_N c w_2, a_1 \dots a_{i-1} (a_j \dots a_k \dots a_N c w_2)) \vdash_r \\ & \alpha a_1 \dots a_i \dots a_{j-1} a_j \dots a_k \dots a_N c w_2 = w_1 c w_2 \end{aligned}$$

Thus,  $w = w_1 c w_2$  is in  $\sigma^*(I)$ , contradicting our assumption that it is not. A symmetric proof works for the case in which  $|w_2| > N$ . We therefore conclude that  $L \subseteq \sigma^*(I)$ .  $\square$

$\square$

### 3 A Characterization

We have the following characterization of reflexive splicing languages:

**Theorem 4.9.** *Let  $L$  be a regular language. Then  $L$  is a reflexive splicing language if and only if there exists a finite, reflexive splicing scheme  $\sigma$  such that  $\sigma(L) \subseteq L$  and  $L \setminus \sigma(L)$  is finite.*

*Proof.* Let  $L$  be a reflexive splicing language. Then there exists a reflexive splicing scheme  $\sigma = (A, S)$  and a finite subset  $I$  of  $A^*$  such that  $L = \sigma^*(I)$ . Since  $L$  is closed

under splicing via the rules of  $\sigma$ ,  $\sigma(L) \subseteq L$ . Any word in  $L$  which is not obtained by splicing two words in  $L$  must be in the initial set  $I$ , which is finite. In other words,  $(L \setminus \sigma(L)) \subseteq I$  is finite.

Now let us assume that  $L$  has the property that there exists a finite, reflexive splicing scheme  $\sigma = (A, S)$  such that  $\sigma(L) \subseteq L$  and  $L \setminus \sigma(L)$  is finite. Consider the rule set  $S$  of  $\sigma$ . Let  $F$  be the set of sites appearing in  $S$ , *i.e.*,

$$F = \{uv \in A^* \mid \exists(u, v; u', v') \text{ or } (u', v'; u, v) \text{ in } S\}.$$

Let  $L' = L \cap A^*FA^*$ , *i.e.*,

$$L' = \{w \in L \mid w = xuvy \text{ for some } uv \in F\}.$$

**Claim 4.10.** *Every site  $uv$  in  $F$  is constant in  $L'$ .*

*Proof.* Let  $xuvy$  and  $x'uvy'$  be words in  $L' \subseteq L$ . Since  $S$  is reflexive, the rule  $r = (u, v; u, v) \in S$ . So  $(xuvy, x'uvy') \vdash_r xuvy'$ , and  $xuvy' \in \sigma(L)$ . Since we have assumed  $\sigma(L) \subseteq L$ , we see that  $xuvy' \in (L \cap A^*uvA^*) \subseteq (L \cap A^*FA^*) = L'$ . A similar argument shows that  $x'uvy \in L'$  also.  $\square$

We may apply Theorem 4.6 to  $L'$ , since  $F$  is a finite set of constants in  $L'$  such that  $L' \setminus A^*FA^*$  is empty, thus finite. The theorem says there exists a finite reflexive splicing system  $(A, I', S')$ , such that  $L' = L(A, I', S')$ . From our proof of Theorem 4.6 we may assume that each site in  $S'$  contains a factor from  $F$ . We construct the splicing system  $\bar{\sigma}$  as follows: Let

$$\begin{aligned} \bar{I} &= I' \cup (L \setminus \sigma(L)), \\ \bar{S} &= S \cup S', \\ \bar{\sigma} &= (A, \bar{S}). \end{aligned}$$

Note that  $\bar{I}$  is finite. Also note  $\bar{S}$  is both finite and reflexive, since  $S'$  and  $S$  are both finite and reflexive.

**Claim 4.11.**  $L = \bar{\sigma}^*(\bar{I})$ .

*Proof.* ( $L \subseteq \bar{\sigma}^*(\bar{I})$ ): Say  $w \in L$ . Either  $w$  is in  $\sigma(L)$ , or not. If  $w \notin \sigma(L)$ , then  $w \in L \setminus \sigma(L) \subseteq \bar{I} \subseteq \bar{\sigma}^*(\bar{I})$ . If  $w \in \sigma(L)$  there exist  $x, y \in L$  and  $r \in S$  such that  $(x, y) \vdash_r w$ . Since  $x$  and  $y$  both have sites from  $S$ , they are both in  $L'$ . Now  $L' = L(A, I', S') \subseteq L(A, \bar{I}, \bar{S})$ , since  $I' \subseteq \bar{I}$  and  $S' \subseteq \bar{S}$ . So  $x, y \in \bar{\sigma}^*(\bar{I})$ , and  $\bar{\sigma}^*(\bar{I})$  is closed under splicing via the rules in  $S$ . In particular,  $(x, y) \vdash_r w \in \bar{\sigma}^*(\bar{I})$ .

$(\bar{\sigma}^*(\bar{I}) \subseteq L)$ : We use the logic of Note 4.4. Let  $w \in \bar{I}$ . Either  $w \in I'$  or  $w \in L \setminus \sigma(L)$ . Since  $I' \subseteq L' \subseteq L$  and  $L \setminus \sigma(L) \subseteq L$ ,  $w \in L$  in either case. Now say  $w \in \bar{\sigma}(L)$ . There exist  $x, y \in L$  and  $r \in \bar{S}$  such that  $(x, y) \vdash_r w$ . If  $r \in S$ , then  $w \in \sigma(L) \subseteq L$ . If  $r \in S'$ , then both  $x$  and  $y$  must contain sites from  $S'$ . So  $x$  and  $y$  must both lie in  $L \cap A^*FA^* = L'$ . Since  $L'$  is closed under splicing via the rules in  $S'$ , we have  $w \in L' \subseteq L$ . □

□

## 4 A Non-Reflexive Splicing Language

Recall  $L$  is in  $H(FIN, FIN)$  if and only if  $L$  can be generated by a splicing system  $(A, I, S)$  in which  $I$  and  $S$  are finite. We shall use the following notation for the set of reflexive splicing languages in  $H(FIN, FIN)$ :

$$H_{ref}(FIN, FIN) = \{L(A, I, S) \mid I, S \text{ are finite, and } S \text{ is reflexive}\}.$$

It is clear from the definitions that  $H_{ref}(FIN, FIN) \subseteq H(FIN, FIN)$ . In fact, the next theorem shows that this containment is proper.

**Theorem 4.12.**  $H(FIN, FIN) \neq H_{ref}(FIN, FIN)$ .

*Proof.* We shall show that  $L = a^*ba^*ba^* \cup a^*ba^* \cup a^*$  is a splicing language, and that it cannot be generated by a reflexive splicing system.

**Claim 4.13.** *The language  $L = a^*ba^*ba^* \cup a^*ba^* \cup a^*$  is a splicing language.*

*Proof.* It is helpful to note that  $L$  is the set of all words over  $A = \{a, b\}$  which contain at most two occurrences of the symbol  $b$ . We construct the splicing system  $\sigma$

as follows: Let

$$I = (1 + a + aa)b(1 + a)b(1 + a + aa),$$

$$S = \{(ba, b; b, a), (a, bb; 1, a), (a, 1; bb, a), (1, bb; b, 1)\}, \text{ and}$$

$$\sigma = (A, I, S).$$

We shall show that  $L = \sigma^*(I)$ .

( $\sigma^*(I) \subseteq L$ ): Clearly  $I \subseteq L$ . To demonstrate that  $\sigma(L) \subseteq L$ , we must show that each of the four rules in  $S$  respects  $L$ .

( $ba, b; b, a$ ): Take  $xbaby$  and  $x'bay'$  in  $L$ , which can be spliced via  $(ba, b; b, a)$ . Note

$|x|_b = 0$  and  $|y'|_b \leq 1$ . The result of splicing is the word  $xbaay'$ , which is in  $L$  because it contains at most two  $b$ 's.

( $a, bb; 1, a$ ): In this case we take  $xabby$  and  $x'ay'$  in  $L$ . Note that  $|x|_b = 0$  and

$|y'|_b \leq 2$ . Splicing via  $(a, bb; 1, a)$  yields  $xaay'$ , which contains at most two  $b$ 's, and is therefore in  $L$ .

( $a, 1; bb, a$ ): Here take  $xay$  and  $x'bbay'$  in  $L$ , noting that  $|x|_b \leq 2$  and  $|y'|_b = 0$ . The

splicing result  $xaay'$  contains at most two  $b$ 's, and is in  $L$ .

( $1, bb; b, 1$ ): Take  $xbbby$  and  $x'by'$  in  $L$ , noting that  $|x|_b = 0$  and  $|y'|_b \leq 1$ . The splicing

product is  $xy'$ , which contains at most one  $b$ , and is therefore in  $L$ .

( $L \subseteq \sigma^*(I)$ ): First we shall show that  $a^*ba^*ba^* \subset \sigma^*(I)$ . This we shall do in two steps. In Step 1 we show that  $L_1 = (1 + a + aa)ba^*b(1 + a + aa)$  is contained in  $\sigma^*(I)$ . In Step 2 we show that arbitrarily long sequences of  $a$ 's can be spliced onto the ends of the words in  $L_1$ , producing  $a^*ba^*ba^* \subset \sigma^*(I)$ . Then we show that  $a^*ba^* \cup a^* \subseteq L$ . This third and last step involves using a rule to cut off either one or two  $b$ 's from words in  $a^*ba^*ba^*$ , generating all of  $L$  in the process.

**Step 1** : Note that for  $x, y \in \{1, a, aa\}$ , the words  $xbbby$  and  $xbaby$  are in  $I \subset \sigma^*(I)$ .

For the induction step, we take  $xba^i by \in \sigma^*(I)$ ,  $i > 0$ , and splice via  $r_1 = (ba, b; b, a)$  as follows:

$$(xbaby, xba^i by) \vdash_{r_1} xbaa^i by = xba^{i+1} by.$$

Thus we have  $L_1 = (1 + a + aa)ba^*b(1 + a + aa) \subset \sigma^*(I)$ .

**Step 2 :** We shall show that  $a^*ba^*ba^* \subset \sigma^*(I)$ . Let  $m \geq 0$ . We already have  $xba^mby$  in  $\sigma^*(I)$  for  $x, y \in \{1, a, aa\}$ . Take  $abb$  and  $a^iba^mby$ ,  $i \geq 1$  and splice via  $r_2 = (a, bb; 1, a)$ , yielding

$$(abb, a^iba^mby) \vdash_{r_2} a^{i+1}ba^mby.$$

This induction shows that  $a^*ba^mb(1 + a + aa) \subset \sigma^*(I)$ . Since  $m$  is arbitrary, we have  $a^*ba^*b(1 + a + aa) \subset \sigma^*(I)$ . Now take  $bba$ , and  $a^nb^mba^i$ , where  $n, m \geq 0$  and  $i \geq 1$ , and splice with the rule  $r_3 = (a, 1; bb, a)$ :

$$(a^nb^mba^i, bba) \vdash_{r_3} a^nb^mba^{i+1}$$

Since  $n$  and  $m$  were arbitrary, we see  $a^*ba^*ba^* \subset \sigma^*(I)$ .

**Step 3 :** Let  $n, m \geq 0$ . Splice  $bb$  and  $ba^nb^mba^m$  using  $r_4 = (1, bb; b, 1)$ :

$$(bb, ba^nb^mba^m) \vdash_{r_4} a^nb^mba^m.$$

Using the same words and the same rule, we can also cut off both  $b$ 's in  $ba^nb^mba^m$ , obtaining  $a^m \in \sigma^*(I)$ . Thus we see  $a^*ba^* \cup a^* \subset \sigma^*(I)$ .

□

**Claim 4.14.**  $L = a^*ba^*ba^* \cup a^*ba^* \cup a^*$  cannot be generated by a reflexive splicing system.

*Proof.* Assume otherwise. Let  $(A, I, S)$  be a reflexive splicing system which generates  $L$ . Recall that the sites in  $S$  are constants of  $L$  (Lemma 2.14). We may discard any rules in  $S$  which are not useful, i.e. which have sites which are not factors in  $L$ . One can easily verify that a factor of  $L$  is a constant of  $L$  if and only if it contains two  $b$ 's, i.e.,  $C_{fac}(L) = a^*ba^*ba^*$  is the set of all constants of  $L$  which are factors in  $L$ . Thus, a rule is in  $S$  only if both of its sites are in  $C_{fac}(L)$ .

Let  $T$  be the set of sites in  $S$ . Since  $T \subseteq a^*ba^*ba^*$ , every word in  $T$  can be written as  $a^i ba^j ba^k$  for some  $i, j, k \geq 0$ . Let

$$K = \max\{j \mid \exists a^i ba^j ba^k \in T\} \text{ and}$$

$$M = 2K + 1.$$

Define

$$\mathcal{Q}_S = \{a^*ba^tba^* \mid t \geq M\}.$$

We shall show that there is no rule in  $S$  which can be used to generate any word in the set  $\mathcal{Q}_S$  from words in  $L$ . Since  $\mathcal{Q}_S$  is an infinite subset of  $L$ , we will thus prove that, regardless of choice of finite  $I$ , the reflexive system  $(A, I, S)$  can not possibly generate  $L$ . We exhaustively examine the types of rule which have sites in  $C(L)$ :

- (i) Rules of type  $(a^i ba^j ba^k, a^l; a^m ba^n ba^p, a^q)$ : This rule type generates words of the form  $xa^i ba^j ba^{k+q} y'$  from words  $xa^i ba^j ba^{k+l} y$  and  $x'a^m ba^n ba^{p+q} y'$  in  $L$ . Notice that in any such product string, there are exactly  $j$   $a$ 's between the two occurrences of the symbol  $b$ . Since such  $j$  is bounded above by  $K$ ,  $w \notin \mathcal{Q}_S$ .
- (ii) Rules of type  $(a^i ba^j, a^k ba^l; a^m ba^n ba^p, a^q)$ : This rule type generates words of the form  $xa^i ba^{j+q} y'$  from words  $xa^i ba^{j+k} ba^l y$  and  $x'a^m ba^n ba^{p+q} y'$  in  $L$ . Since  $|x|_b = 0$  and  $|y'|_b = 0$ , such splicing products are not in  $a^*ba^*ba^*$ .
- (iii) Rules of type  $(a^i, a^j ba^k ba^l; a^m ba^n ba^p, a^q)$ : This rule type generates words of the form  $xa^{i+q} y'$ , where again,  $|x|_b = 0$  and  $|y'|_b = 0$ . Such products are not in  $a^*ba^*ba^*$ .
- (iv) Rules of type  $(a^i ba^j ba^k, a^l; a^m ba^n, a^p ba^q)$ : This type of rule generates splicing products with three  $b$ 's. Such products are not in  $a^*ba^*ba^*$ .
- (v) Rules of type  $(a^i ba^j ba^k, a^l; a^m, a^n ba^p ba^q)$ : This rule type generates words with four  $b$ 's. As in the previous case, such products are not in  $a^*ba^*ba^*$ .
- (vi) Rules of type  $(a^i ba^j, a^k ba^l; a^m ba^n, a^p ba^q)$ : Given  $xa^i ba^{j+k} ba^l y$  and  $x'a^m ba^{n+p} ba^q y'$  from  $L$ , a splice with this type of rule yields a product of the form  $w = xa^i ba^{j+p} ba^q y'$  which is in  $a^*ba^*ba^*$ . Since  $j + p \leq 2K < M$ , we see  $w \notin \mathcal{Q}_S$ .

The remaining rule forms are symmetric to those presented here, and their discussion yields similar results through symmetric arguments. From the analysis above, the only rules which can be in  $S$  are of Type  $(i)$  or Type  $(vi)$ , or types symmetric to these. Such rules do not produce any of the words in  $\mathcal{Q}_S$ . Thus  $L$  cannot be generated by a reflexive splicing system.  $\square$

By Claim 4.13,  $L$  is a splicing language in  $H(FIN, FIN)$ , and by Claim 4.14,  $L$  is not a reflexive splicing language. Hence we have  $H(FIN, FIN) \neq H_{ref}(FIN, FIN)$ .  $\square$

# Chapter 5

## Main Results

In this chapter, we shall present the Simultaneous Pumping Theorem [20], which is a generalization of the standard pumping lemma for regular languages [10]. This will be the tool with which we craft the main theorem of this chapter. In Theorem 5.15, we answer the question posed by Head in [8]. As we shall see, there does exist an algorithm which takes a regular language  $L$  as input, and returns as output the answer to the question: Is there a finite set  $F$  of constant factors of  $L$  such that all but finitely many words in  $L$  have a factor from  $F$ ?

### 1 Simultaneous Pumping Theorem

**Lemma 5.1.** *[Pixton] Let  $M$  be a finite monoid. There exists a positive integer  $N$  with the following property: If  $n \geq N$  and  $u_1, \dots, u_n \in M$ , then there exist  $q$  and  $q'$ ,  $1 \leq q < q' \leq n$ , so that if  $x = u_1 \dots u_{q-1}$ ,  $y = u_q \dots u_{q'-1}$  and  $z = u_{q'} \dots u_n$ , then  $xy = x$  and  $yz = z$ .*

*Proof.* Let  $M$  be a finite monoid. Let  $K = \text{Card}(M)$ . Let  $N = K^2 + 1$ . Let  $n \geq N$ , and let  $w$  be a product of  $n$  elements in  $M$ , i.e.,  $w = u_1 u_2 \dots u_n$ . We define the following prefix factors of  $w$  which are products in  $M$ :

$$p_j = u_1 \dots u_j, \quad 1 \leq j \leq n.$$

**Claim 5.2.** *There exist  $K + 1$  different subscripts  $j(1) < \dots < j(K + 1)$  so that*

$$p_{j(1)} = p_{j(2)} = \dots = p_{j(K+1)}.$$

*Proof.* Note  $n \geq N = K^2 + 1 = K((K + 1) - 1) + 1$ , so we may apply the Pigeon Hole Principal to the various  $p_j$  in  $M$ ,  $1 \leq j \leq n$ . Thus, for some  $x \in M$ ,  $x = p_{j(1)} = p_{j(2)} \dots = p_{j(K+1)}$ .  $\square$

Let us define the integer  $j(0) = 0$ , and define the empty prefix  $p_{j(0)} = 1$ . For each  $p_{j(i)}$ ,  $1 \leq i \leq K + 1$  we define the suffix sequence  $\xi_i$  of  $p_{j(i)}$  as follows:

$$\xi_i = u_{j(i-1)+1} \dots u_{j(i)}.$$

Hence

$$p_{j(i)} = p_{j(i-1)}\xi_i.$$

Notice that, for  $i = 1$ , we have  $u_{j(i-1)+1} = u_{j(0)+1} = u_{0+1} = u_1$ , so the definition of  $\xi_1 = u_1 \dots u_{j(1)}$  makes sense. We can now parse  $w = u_1 u_2 \dots u_n$  as follows:

$$w = (u_1 \dots u_{j(1)})(u_{j(1)+1} \dots u_{j(2)}) \dots (u_{j(K)+1} \dots u_{j(K+1)})\zeta,$$

where  $\zeta$  is the suffix  $u_{j(K+1)+1} \dots u_n$  of  $w$ . From the definition of the  $\xi_i$ , we see that  $w$  can also be parsed as:

$$w = \xi_1 \xi_2 \dots \xi_{K+1} \zeta.$$

Now we define the products  $s_i$  for  $1 \leq i \leq K + 2$ :

$$s_i = \begin{cases} \xi_i \dots \xi_{K+1} \zeta & \text{for } 1 \leq i \leq K + 1 \\ \zeta & \text{for } i = K + 2. \end{cases}$$

We have another claim:

**Claim 5.3.** *There exist integers  $l$  and  $k$  such that  $2 \leq l < k \leq K + 2$  and such that  $s_l = s_k$ .*

*Proof.* There are  $K + 1$  of the  $s_i$ , since  $i$  ranges from 2 to  $K + 2$ . The Pigeon Hole Principle dictates that at least two of the  $s_i$  are actually the same element in  $M$ .  $\square$

Now we are prepared to finish the proof of our original lemma. Let  $q = j(l - 1) + 1$  and  $q' = j(k - 1) + 1$ . Then, if

$$\begin{aligned}x &= u_1 \dots u_{q-1} \\y &= u_q \dots u_{q'-1} \quad \text{and} \\z &= u_{q'} \dots u_n\end{aligned}$$

we may write

$$\begin{aligned}x &= u_1 u_2 \dots u_{j(l-1)} = \xi_1 \dots \xi_{l-1} \\y &= u_{j(l-1)+1} \dots u_{j(k-1)} = \xi_l \dots \xi_{k-1} \quad \text{and} \\z &= u_{j(k-1)+1} \dots u_n = \xi_k \dots \xi_{K+1} \zeta.\end{aligned}$$

We intend by this notation to indicate that if  $k = K + 2$ , then  $z = \zeta$ . By Claim 5.2 we see

$$\begin{aligned}xy &= \xi_1 \dots \xi_{k-1} \\&= p_{j(k-1)} \\&= p_{j(l-1)} \\&= \xi_1 \dots \xi_{l-1} \\&= x\end{aligned}$$

By Claim 5.3 we have

$$\begin{aligned}yz &= \xi_l \dots \xi_{K+1} \zeta \\&= s_l \\&= s_k \\&= \xi_k \dots \xi_{K+1} \zeta \\&= z\end{aligned}$$

□

**Lemma 5.4.** (*Two-Sided Pumping Lemma for Regular Languages*)

[Pirton] *Let  $L \subseteq A^*$  be a regular language. Then there exists  $N$  such that if  $w \in A^*$*

and  $|w| \geq N$  then  $w = xyz$ ,  $x, y, z \in A^*$ ,  $y \neq 1$ , and  $\forall k \geq 0$  and  $\alpha, \beta \in A^*$ , the following hold:

$$\begin{aligned} \alpha x \beta \in L &\iff \alpha x y^k \beta \in L \text{ and} \\ \alpha z \beta \in L &\iff \alpha y^k z \beta \in L. \end{aligned}$$

*Proof.* First note that  $\text{Syn } L$  is finite, since  $L$  is regular (2.1). Let  $N$  be the integer described by Lemma 5.1 for  $\text{Syn } L$ . Let  $\eta$  be the natural map from  $A^*$  into  $\text{Syn } L$ . Let  $n \geq N$ , and let  $w \in A^*$  be such that  $|w| = n$ . We may write  $w$  as  $w = a_1 \dots a_n$ , where  $a_i \in A$ ,  $1 \leq i \leq n$ . Note  $\eta(w) = \eta(a_1) \dots \eta(a_n)$  in  $\text{Syn } L$ . By Lemma 5.1, there exist integers  $j$  and  $k$  such that  $1 \leq j < k \leq n$  and such that the product  $\eta(a_1) \dots \eta(a_n)$  can be factored as

$$\begin{aligned} \bar{x} &= \eta(a_1) \dots \eta(a_{j-1}) \\ \bar{y} &= \eta(a_j) \dots \eta(a_{k-1}) \text{ and} \\ \bar{z} &= \eta(a_k) \dots \eta(a_n) \end{aligned}$$

where  $\bar{x}\bar{y} = \bar{x}$  and  $\bar{y}\bar{z} = \bar{z}$ . So, letting  $x = a_1 \dots a_{j-1}$ ,  $y = a_j \dots a_{k-1}$ , and  $z = a_k \dots a_n$ , we see  $w$  can be factored as  $w = xyz$  where  $y \neq 1$ . Further,  $xy^k \equiv_L x$  and  $y^k z \equiv_L z$  for all  $k \geq 0$ . The conclusion now follows immediately from the definition of the syntactic congruence  $\equiv_L$ .  $\square$

**Theorem 5.5.** (*Simultaneous Pumping Theorem*)

[Pirton] *Let  $\mathcal{F}$  be a finite set of regular languages. There exists a positive integer  $N$  such that if  $w \in A^*$  and  $|w| \geq N$ , then  $w$  can be factored as  $w = xyz$  where  $x, y, z \in A^*$ ,  $y \neq 1$ , such that the following holds: For all  $L \in \mathcal{F}$  and for all  $k \geq 0$*

$$\begin{aligned} (1) \quad \alpha x \beta \in L &\iff \alpha x y^k \beta \in L \text{ and} \\ (2) \quad \alpha z \beta \in L &\iff \alpha y^k z \beta \in L. \end{aligned}$$

*Proof.* Let  $\mathcal{F} = \{L_1, \dots, L_p\}$  be a finite set of regular languages. For each  $L_i \in \mathcal{F}$ ,  $1 \leq i \leq p$ , let  $M_i = \text{Syn } L_i$  and let  $\eta_i : A^* \rightarrow A^* / \equiv_{L_i}$  be the natural map from  $A^*$  into  $M_i$ . Let  $M = M_1 \times \dots \times M_p$ . Let  $\Pi = (\eta_1, \dots, \eta_p)$  be the product map from  $(A^*)^p$  into  $M$ . Let  $N$  be the integer defined for  $M$  by Lemma 5.1. Let  $w \in A^*$  be

such that  $|w| \geq N$ . So  $w = a_1 \dots a_n$  for some  $n \geq N$ . Let  $[a_j]_i$  denote the image of  $a_j$  under  $\eta_i$  for  $1 \leq j \leq n$  and  $1 \leq i \leq p$ . Consider the element  $\bar{w} = \Pi(w)$  in  $M$ :

$$\begin{aligned}\bar{w} = \Pi(w) &= ([w]_1, \dots, [w]_p) \\ &= ([a_1 \dots a_n]_1, \dots, [a_1 \dots a_n]_p) \\ &= ([a_1]_1, \dots, [a_1]_p) \dots ([a_n]_1, \dots, [a_n]_p).\end{aligned}$$

Since  $\bar{w}$  can be seen as the product of  $n$  elements in  $M$ , we may apply Lemma 5.1 to  $\bar{w}$  as follows: There exist  $j, k$  with  $1 \leq j < k \leq n$  such that the products

$$\begin{aligned}\bar{x} &= ([a_1]_1, \dots, [a_1]_p) \dots ([a_{j-1}]_1, \dots, [a_{j-1}]_p), \\ \bar{y} &= ([a_j]_1, \dots, [a_j]_p) \dots ([a_{k-1}]_1, \dots, [a_{k-1}]_p), \text{ and} \\ \bar{z} &= ([a_k]_1, \dots, [a_k]_p) \dots ([a_n]_1, \dots, [a_n]_p)\end{aligned}$$

have the property that

$$\bar{x}\bar{y} = \bar{x} \text{ and } \bar{y}\bar{z} = \bar{z}.$$

Thus, we factor  $w \in A^*$  as  $w = xyz$ , where  $x = a_1 \dots a_{j-1}$ ,  $y = a_j \dots a_{k-1}$ , and  $z = a_k \dots a_n$ . The following equations hold:

$$\begin{aligned}\Pi(x)\Pi(y) &= \bar{x}\bar{y} = \bar{x} = \Pi(x) \quad \text{and} \\ \Pi(y)\Pi(z) &= \bar{y}\bar{z} = \bar{z} = \Pi(z).\end{aligned}$$

Since  $\Pi(x) = \Pi(xy)$ ,  $\Pi(x)$  must agree in its every coordinate with  $\Pi(xy)$ . Thus, for  $1 \leq i \leq p$ ,  $\eta_i(x) = \eta_i(xy)$  and we have  $x \equiv_{L_i} xy$ . Similarly, we have  $yz \equiv_{L_i} z$ . By the definition of the syntactic congruences  $\equiv_{L_i}$ , we have the desired conclusion for each  $L_i \in \mathcal{F}$ .  $\square$

## 2 Main Result and Algorithm

In this section we shall show that, given a regular language  $L \subseteq A^*$ , there is a computable bound on the length of the constant factors in  $L$  which must be considered in order to determine if there is a finite set  $F$  of constant factors in  $L$  such that

$L \setminus A^*FA^*$  is finite. Head asked in [8] if such an algorithm exists. Our result answers in the affirmative. We begin with several definitions:

Let  $L \subseteq A^*$ . For  $k \geq 0$ , we define the following set:

$$L_k = \{w \in L \mid |w| \leq k\}.$$

**Definition 5.6.** Let  $L \subseteq A^*$  and  $w \in A^*$ . A string  $x \in A^*$  is called an ***L-factor*** of  $w$  if  $x$  is a factor of  $w$  and  $x$  is also a word in  $L$ . The set of all *L-factors* of  $w$  is simply  $\text{Fac } w \cap L$ .

We define a *prime* language as follows:

**Definition 5.7.** Let  $L \subseteq A^*$ . We say  $L$  is ***prime*** if  $L \cap (A^+LA^* \cup A^*LA^+)$  is empty.

We need three lemmas:

**Lemma 5.8.** Let  $L$  and  $P$  be regular languages written over alphabet  $A$ . Say  $P$  is prime. Let  $k$  be the integer defined by the Simultaneous Pumping Theorem for the set  $\mathcal{F} = \{L, P\}$ . Let  $w \in L \cap (A^*PA^* \setminus A^*P_kA^*)$ , and let  $xcy$  be any factorization of  $w$  such that  $c \in P$ . Then there exists a factorization of  $w$  as  $w = xuzvy$ ,  $z \neq 1$  such that for every  $j > k$ , the word  $w(j) = xuz^{j+1}vy \in L \cap A^*PA^*$ . Further, any *P-factor* of  $w(j)$  which does not lie in  $xu$  or  $vy$  must be longer than  $j$ .

*Proof.* Let  $w \in L \cap (A^*PA^* \setminus A^*P_kA^*)$ . Then  $w = xcy$  for some  $x, y \in A^*$  and  $c \in P$ . Since  $|c| > k$ , there is a factorization  $c = uzv$ ,  $z \neq 1$ , given by the Simultaneous Pumping Theorem (Theorem 5.5), such that  $uz^i v \in P$  and  $xuz^i vy \in L \cap A^*PA^*$  for all  $i \geq 0$ . Let us define  $w(j) = xuz^{j+1}vy$ .

Let  $c'$  be a *P-factor* of  $w(j)$  which is not a factor of  $xu$  or  $vy$ . We shall show that  $|c'| > j$ . We may factor  $w(j)$  as  $w(j) = xuz^{j+1}vy = x'c'y'$ , for some  $x', y' \in A^*$ . We examine the cases:

- (i) First,  $c'$  cannot be a proper factor of  $uz^{j+1}v$ , because  $P$  is prime. For the same reason,  $uz^{j+1}v$  cannot be a proper factor of  $c'$ .
- (ii) Say  $c' = x_2uz^n z_1$ , where  $x = x_1x_2$ ,  $z = z_1z_2$ , and  $n < j + 1$ . Note that  $c' \in P$ ,  $|c'| > k$ , and  $c'$  contains  $u$ . Thus, Theorem 5.5 says  $x_2uz^i z^n z_1 \in P$  for all  $i \geq 0$ .

But  $c'$  is a proper prefix factor of any such  $x_2uz^iz^n z_1$  for  $i > 0$ . This contradicts  $P$  prime, and therefore cannot occur. The symmetric case ( $c' = z'_2 z'^n v y'_1$  for  $z = z'_1 z'_2$  and  $y = y'_1 y'_2$ ) cannot occur either, and the proof follows along the same lines.

(iii) If  $c'$  contains  $z^{j+1}$ , then  $|c'| > j$ , as desired.

The above analysis treats all cases in which  $c'$  is a  $P$ -factor of  $w(j)$ , but not a factor of  $xu$  or  $vy$ . Thus, we have that every  $P$ -factor of  $w(j)$  which is not a factor of  $xu$  or  $vy$  is longer than  $j$ .  $\square$

For the next lemma, we need a definition:

**Definition 5.9.** Let  $L \subseteq A^*$  and let  $w = a_1 a_2 \dots a_m \in A^*$ . We define the set of **indexed  $L$ -factors of  $w$** , denoted by  $\| w \|_L$ , as follows:

$$\| w \|_L = \{(i, x) \in \mathbb{N} \times L \mid w = uxv \text{ for } u, v \in A^* \text{ such that } |u| = i - 1\}.$$

In other words,  $(i, x)$  is an indexed  $L$ -factor of  $w$  if  $x \in L$  and if  $w = a_1 \dots a_{i-1} x v$  for some  $v \in A^*$  and  $a_1, \dots, a_{i-1} \in A$ . If  $(i, x)$  is an indexed  $L$ -factor of  $w$ , then clearly  $x$  is an  $L$ -factor of  $w$ .

**Note 5.10.** Let  $P \subset A^*$  be a prime set. Let  $w \in A^+$  and say  $(i, x)$  and  $(i', x')$  are elements in  $\| w \|_P$ . If  $i = i'$ , then  $x = x'$ .

*Proof.* If  $i = i'$  and  $x \neq x'$ , then either  $x = x'y$  or  $x' = xy'$  for some  $y, y' \neq 1$ . Either case contradicts the fact that  $P$  is prime.  $\square$

**Lemma 5.11.** Let  $L$  and  $P$  be regular languages written over alphabet  $A$ . Say  $P$  is prime. Let  $k$  be the integer defined by the Simultaneous Pumping Theorem for the set  $\mathcal{F} = \{L, P\}$ . Let  $j > k$ . If  $L \cap (A^* P_j A^* \setminus A^* P_k A^*)$  is not empty, then  $L \cap (A^* P A^* \setminus A^* P_j A^*)$  is not empty.

*Proof.* First we make the following claim:

**Claim 5.12.** Given  $w \in L \cap (A^* P_j A^* \setminus A^* P_k A^*)$ , we can construct a word  $\bar{w} \in L \cap (A^* P A^* \setminus A^* P_j A^*)$  such that

$$\text{Card } \| \bar{w} \|_{P_j} < \text{Card } \| w \|_{P_j}.$$

*Proof.* Let  $w \in L \cap (A^*P_jA^* \setminus A^*P_kA^*)$ . Let us order the set  $\| w \|_{P_j}$  according to the natural ordering on  $\mathbb{N}$ . Let  $T(w) = \{(n_1, c_1), \dots, (n_t, c_t)\}$  be that ordered set of pairs. By Note 5.10, this ordering is well defined, and  $t$  is the cardinality of  $T(w)$ .

It is apparent from  $T(w)$  that  $c_t$  is the last  $P_j$ -factor appearing in  $w$ . So let us parse  $w$  as  $w = xc_t y$ . By Lemma 5.8, there exists a word  $\bar{w} = xuz^{j+1}vy \in L \cap A^*PA^*$  having the property that its every  $P$ -factor not lying in  $xu$  or  $vy$  is in  $P \setminus P_j$ . We shall show that  $\bar{w}$  satisfies  $\text{Card } \|\bar{w}\|_{P_j} < \text{Card } \|w\|_{P_j}$ .

Notice every  $P_j$ -factor of  $\bar{w}$  lies in  $xu$ , for if  $c'$  were a  $P_j$ -factor of  $\bar{w}$  lying in  $vy$ , then the pair  $(n', c')$  would appear in  $T(w)$ . However,  $n'$  would be larger than  $n_t$ , since  $c'$  would appear in  $w$  after  $c_t = uzv$ . This contradicts  $n_t$  being the largest index of any pair in  $T(w)$ .

Since every  $P_j$ -factor of  $\bar{w}$  lies in  $xu$ , and  $xu$  is also a factor of  $w$ , we have that every  $P_j$ -factor of  $\bar{w}$  is also a factor of  $w$ . Further, the  $P_j$ -factor  $c_t$  from the pair  $(n_t, c_t)$  in  $T(w)$  does not appear in  $\|\bar{w}\|_{P_j}$ , because  $c_t \in P_j$  and  $c_t$  is not contained in  $xu$  (Lemma 5.8). Thus  $\text{Card } \|\bar{w}\|_{P_j} < \text{Card } \|w\|_{P_j}$ .  $\square$

Now we can prove our lemma. Let  $w \in A^*P_jA^* \setminus A^*P_kA^*$ . We apply the claim above iteratively, generating after finitely many applications a word  $w_0 \in L \cap A^*PA^*$  having no  $P_j$ -factors at all. It is thus apparent that  $L \cap (A^*PA^* \setminus A^*P_jA^*) \neq \emptyset$ .  $\square$

**Lemma 5.13.** *Let  $L$  and  $P$  be regular languages written over alphabet  $A$ . Say  $P$  is prime. Let  $k$  be the integer defined by the Simultaneous Pumping Theorem for the set  $\mathcal{F} = \{L, P\}$ . Let  $j > k$ . If  $L \cap (A^*PA^* \setminus A^*P_jA^*)$  is not empty, then it is infinite.*

*Proof.* Let  $w \in L \cap (A^*PA^* \setminus A^*P_jA^*)$ . Then  $w = xcy$  for some  $c \in P \setminus P_j$ . Let  $c = uzv$  be a factorization of  $c$  guaranteed by the Simultaneous Pumping Theorem. Consider the infinite set below:

$$S(xuzv y) = \{xuz^i v y \mid i \geq j + 1\}$$

By the Simultaneous Pumping Theorem, every word in  $S(xuzv y)$  is a word in  $L \cap (A^*PA^*)$ . By Lemma 5.8, every word in  $S(xuzv y)$  has only  $P$ -factors which appear in  $w$ , or  $P$ -factors which are longer than  $j$ . Hence every word in  $S(xuzv y)$  is in  $L \cap (A^*PA^* \setminus A^*P_jA^*)$ .  $\square$

**Theorem 5.14.** *Let  $L, P \subseteq A^*$  be regular languages, with  $P$  being prime. Let  $k$  be the integer defined by the Simultaneous Pumping Theorem for the set  $\mathcal{F} = \{L, P\}$ . Then there exists a finite set  $F \subseteq P$  such that  $L \setminus A^*FA^*$  is finite if and only if  $L \setminus A^*P_kA^*$  is finite.*

*Proof.* If  $L \setminus A^*P_kA^*$  is finite, then  $F = P_k$ , and we are done. So let us assume  $L \setminus A^*P_kA^*$  is infinite. We wish to show that for any finite  $F \subseteq P$ , the set  $L \setminus A^*FA^*$  is infinite.

Let  $F \subseteq P$  be a finite set. Then for some  $j \in \mathbb{N}$ ,  $F \subseteq P_j$ , and  $L \setminus A^*P_jA^* \subseteq L \setminus A^*FA^*$ . It is therefore sufficient to show that  $L \setminus A^*P_jA^*$  is infinite.

If  $j \leq k$  then  $L \setminus A^*P_kA^* \subseteq L \setminus A^*P_jA^*$ , and we are done. Suppose  $j > k$ . If  $L \cap (A^*P_jA^* \setminus A^*P_kA^*)$  is empty, then  $L \setminus A^*P_jA^* = L \setminus A^*P_kA^*$ , and we are done. So assume  $L \cap (A^*P_jA^* \setminus A^*P_kA^*)$  is not empty. Then by Lemma 5.11,  $L \cap (A^*PA^* \setminus A^*P_jA^*)$  is not empty. By Lemma 5.13,  $L \cap (A^*PA^* \setminus A^*P_jA^*)$  is infinite. Thus  $L \setminus A^*P_jA^*$  is infinite, and we are done.  $\square$

Before stating the theorem which is basically a corollary to Theorem 5.14, we shall mention the following definitions for a language  $L \subseteq A^*$ : Recall the set  $C(L)$  of *constants* of  $L$  is denoted by

$$C(L) = \{w \in A^* \mid w \text{ is constant of } L\}.$$

Let us denote the set of *prime constants* of  $L$  by

$$P(L) = C(L) \setminus (A^*C(L)A^+ \cup A^+C(L)A^*).$$

Notice that  $P(L)$  is a prime language, and it is regular since  $C(L)$  is regular (Lemma 3.8).

Finally, for  $k \geq 0$ , and  $L \subseteq A^*$ , we denote the constants and prime constants of  $L$  which have length bounded by  $k$  as follows:

$$C_k(L) = \{w \in C(L) \mid |w| \leq k\} \quad \text{and}$$

$$P_k(L) = C_k(L) \cap P(L).$$

**Theorem 5.15.** *Let  $L \subseteq A^*$  be a regular language. Let  $k$  be the integer defined by the Simultaneous Pumping Theorem for  $\mathcal{F} = \{L, P(L)\}$ . Then the following are equivalent:*

(i) *There exists a reflexive splicing system  $(A, I, S)$  such that  $L = L(A, I, S)$  where  $I$  and  $S$  are finite, and each rule in  $S$  has either the form  $(uv, 1; u, 1)$  or the form  $(1, u; 1, vu)$ .*

(ii) *There exists a finite set  $F \subset C(L)$  such that  $L \setminus A^*FA^*$  is finite.*

(iii) *The set  $L \setminus A^*P_k(L)A^*$  is finite.*

*Proof.* By Head's Theorem 4.6, we have (i)  $\Leftrightarrow$  (ii). The implication (iii)  $\Rightarrow$  (ii) is immediate, since  $P_k(L) \subset C(L)$ .

(ii)  $\Rightarrow$  (iii): Say there exists a finite subset  $F \subset C(L)$  such that  $L \setminus A^*FA^*$  is finite. We claim there exists a finite subset  $F' \subset P(L)$  such that  $L \setminus A^*F'A^*$  is finite. Notice that every constant  $c$  in  $C(L)$  contains a prime constant. For each  $c \in F$ , write  $c = up_cv$ , where  $p_c \in P(L)$  and  $u, v \in A^*$ . Let  $F' = \{p_c \mid c \in F\}$ . Clearly  $F'$  is a finite subset of  $P(L)$ . Since  $F \subset A^*F'A^*$ , it follows that  $A^*FA^* \subset A^*F'A^*$ . Thus  $L \setminus A^*F'A^* \subset L \setminus A^*FA^*$ , and  $L \setminus A^*F'A^*$  must be finite. That (iii) follows is immediate from Theorem 5.14.  $\square$

In fact, what Theorem 5.15 says is that, given  $L \subseteq A^*$ , if one wishes to determine whether  $L$  is a reflexive splicing language of the type specified in Theorem 4.6, there is an upper bound on the  $j$  for which one must check whether  $L \setminus A^*P_jA^*$  is finite. Since  $j$  and  $P_j$  can be algorithmically determined based on  $L$ , this result yields an algorithm which answers Head's challenge.

# Chapter 6

## Simple and Null Context Splicing

In this chapter we will review some results about splicing languages generated by rules of special types. Specifically, we will be looking at systems having rule sets which contain only rules whose sites are constant factors in the splicing language.

In [13], Mateescu, Paun, Rozenberg and Salomaa introduced a class of splicing languages whose rules have constant symbols for sites. This class of languages, called simple splicing languages, was one of the first subsets of  $H(FIN, FIN)$  to be extensively treated in the literature.

Head treated a generalization of simple splicing systems in [7]. His generalization situates the simple splicing languages as a subclass of a class of splicing languages generated by rules having sites which are constants. Head proved that this class of languages, known as the null context languages, are exactly the class of strictly locally testable languages defined by McNaughton and Papert in [14]. He reformulated and expanded upon his 1987 results in [9].

We begin with a definition of simple splicing systems due to Mateescu *et.al.*[13], and a characterization of simple splicing languages.

**Definition 6.1.** *Let  $\tau = (A, I, S)$  be a splicing system in which all rules in  $S$  have the form  $(a, 1; a, 1)$ , where  $a \in A$ . Then  $\tau$  is called a **simple splicing system**. A splicing language  $L$  is said to be a **simple splicing language** if  $L$  can be generated by a simple splicing system.*

# 1 A characterization

We have the following characterization of simple splicing languages:

**Proposition 6.2.** *A language  $L \subseteq A^*$  is a simple splicing language if and only if all but finitely many factors in  $L$  contain a symbol which is a constant of  $L$ .*

*Proof.* Let  $L$  be a simple splicing language generated by  $(I, \sigma)$ , where  $\sigma = (A, S)$  and the rules in  $S$  have the form prescribed by the definition of Mateescu *et.al.* Let  $\mathcal{A} = \{a \in A \mid (a, 1; a, 1) \in S\}$ . We make the following claim:

**Claim 6.3.** *For all  $k \geq 0$ ,  $\text{Fac } \sigma^k(I) \cap (A \setminus \mathcal{A})^* = \text{Fac } \sigma^0(I) \cap (A \setminus \mathcal{A})^*$ .*

*Proof.* (By induction) The claim is clearly true for  $k = 0$ . Assume true for all  $i \leq k$ , where  $k > 0$ .

Let  $w \in \sigma^{k+1}(I) \setminus \sigma^k(I)$ . Then  $w$  is the product of splicing two words in  $\sigma^k(I)$ . Let  $z = xay$  and  $z' = x'ay'$  be words in  $\sigma^k(I)$ , and  $r = (a, 1; a, 1) \in S$  such that  $(z, z') \vdash_r w = xay'$ . We consider the factors of  $w$ . Since  $xa \in \text{Fac } \sigma^k(I)$ , all factors of  $w$  contained in  $xa$  satisfy the induction hypothesis. Likewise, the factors of  $ay'$  satisfy the hypothesis. Any other factor contained in  $w$  must be of the form  $x_1ax_2$ , where  $x_1$  is a suffix of  $x$ , and  $x_2$  is a prefix of  $y'$ . Obviously, such a factor contains  $a$ , and is not in  $\text{Fac } L \cap (A \setminus \mathcal{A})^*$ .  $\square$

Since  $\sigma^0(I)$  is finite,  $\text{Fac } \sigma^0(I)$  is finite. Thus  $\text{Fac } I \cap (A \setminus \mathcal{A})^*$  is finite. By Claim 6.3, it is clear that all but finitely many factors in  $L = \sigma^*(I)$  contain a constant symbol from  $\mathcal{A}$ .

Now say  $L \subseteq A^*$  is a language having the property that all but finitely many factors in  $L$  contain a constant symbol of  $L$ . Let  $F$  be that set of factors of  $L$  which

contain no constant symbols. We define the following:

$$\begin{aligned}
m &= 1 + \max\{|x| \mid x \in F\}, \\
C &= \{c \in A \mid c \text{ is constant in } L\}, \\
n &= \text{Card}(C), \\
K &= (2n + 1)m, \\
S &= \{(c, 1; c, 1) \mid c \in C\} \\
I &= \{w \in L \mid |w| < K\}, \\
\sigma &= (A, I, S).
\end{aligned}$$

**Claim 6.4.**  $L = \sigma^*(I)$ .

*Proof.* First we show by contradiction that  $L \subseteq \sigma^*(I)$ . Assume there is a word in  $L$  which is not in  $\sigma^*(I)$ . Let  $w$  be a shortest such word. Clearly,  $|w| \geq K$ . So  $w$  contains  $2n + 1$  consecutive factors, each at least  $m$  symbols long and each thus containing a constant symbol. Since there are only  $n$  constant symbols, there must be some constant symbol, say  $c$ , which appears at least three times in  $w$  (Pigeon Hole Principle). Take then a factorization of  $w$  as  $w = xcucvcy$ , where  $x, u, v, y \in A^*$ . Notice that the words  $z = xcucy$  and  $z' = xcvcy$  are in  $L$  since  $c$  is a constant symbol of  $L$ . Since  $z$  and  $z'$  are strictly shorter than  $w$ , they are in  $S^*(I)$ . Splicing  $z$  and  $z'$  using  $(c, 1; c, 1)$  in  $S$ , we obtain  $w \in \sigma^*(I)$ , a contradiction.

Now we shall show  $\sigma^*(I) \subseteq L$ . It is clear that  $I \subseteq L$ . Take strings  $z = xcy$  and  $z' = x'cy'$  in  $L$ , and a rule  $(c, 1; c, 1)$  in  $S$ , and consider the splicing product  $w = xcy'$ . Since  $z$  and  $z'$  are in  $L$  and  $c$  is a constant of  $L$ , we have  $w$  in  $L$ . So  $\sigma(L) \subseteq L$ . □

□

This proof follows very much along the lines of proofs given by Head in [7] and [9]. In fact, Head defines the class of simple splicing languages as a bottom level class of an hierarchy whose union is the class of all null context splicing languages. We have included the above proof because it is straightforward, and it identifies very

specifically the characteristic common to all simple splicing languages in terms of factor sets. This should be of interest to those readers interested in local testability. For the sake of completeness, we include now the definitions of null context splicing systems and  $S_k$  splicing systems. For completeness, we also include Theorem 6.8, a result due to Head [9].

**Definition 6.5.** A **null context splicing system** (*NCH system*) is a system  $G = (A, I, S)$ , in which all rules have the form  $(u, 1; u, 1)$ , where  $u \in A^*$ . A language  $L$  is called a **null context splicing language** if there exists a null context splicing system  $G$  for which  $L = L(G)$ .

**Definition 6.6.** An  $S_{-1}H$  **splicing system** is an NCH system  $(A, I, S)$  in which  $S$  is the empty set. For  $k \geq 0$ , an  $S_k$  **splicing system** ( *$S_kH$  system*) is an NCH system  $G = (A, I, S)$  for which each rule in  $S$  has sites no longer than  $k$ . A language  $L$  is called an  $S_k$  **splicing language** ( *$S_kH$  language*) if there exists an  $S_kH$  system  $G$  for which  $L = L(G)$ . The family of  $S_k$  splicing languages is denoted by  $S_kH$ .

Notice that the union of the families  $S_kH$  for  $k \geq -1$  is the class of NCH languages.

**Definition 6.7.** ([14] modified using [5]) A language  $L$  is **strictly locally testable** (*SLT*) if there is a positive integer  $k$  for which every factor of  $L$  of length  $k$  is constant.

**Theorem 6.8.** The class of strictly locally testable languages is equal to the class of null context languages.

Theorem 6.8 implies that a language  $L$  is a null context splicing language if and only if all but finitely many factors in  $L$  are constants of  $L$ . Put differently,  $L$  is an NCH language if and only if all but finitely many factors in  $L$  contain a constant word. This result comes from the work of Head [9] and DeLuca and Restivo [5]. It subsumes, of course, Proposition 6.2. We have included the proof Proposition 6.2 above in order to emphasize the connection between simple splicing languages and constant factors.

Another sort of generalization of simple splicing systems can be considered. Rather than demanding that all sites be constant words, we can consider splicing systems in

which every rule has the form  $(a, 1; b, 1)$ , where  $a$  and  $b$  are alphabet symbols. We shall consider systems containing such rules in Chapter 7. The next level generalization along this line is systems having rules of the form  $(u, 1; v, 1)$ , where  $u, v \in A^+$ . We shall treat systems of this type which have exactly one rule in Chapter 8.

# Chapter 7

## Semi-Simple Languages

In this chapter we shall introduce a generalization of simple splicing systems, called semi-simple splicing systems. In semi-simple systems, all sites are alphabet symbols. We do not, however, require that each site be constant, as was the case in simple splicing systems.

**Definition 7.1.** *Let  $(A, I, S)$  be a splicing system in which  $I$  and  $S$  are finite and every rule in  $S$  has the form  $(a, 1; b, 1)$ , where  $a, b$  are in  $A$ . We say  $\sigma = (A, S)$  is a **semi-simple splicing scheme** and  $(A, I, S)$  is a **semi-simple system**. Any language which can be generated by such a system is called a **semi-simple language**.*

We shall show the following for semi-simple languages:

- (i) There are semi-simple languages which are not simple.
- (ii) There are null context languages which are not semi-simple.
- (iii) Every semi-simple language is strictly locally testable.

### 1 Semi-simple $\not\subseteq$ Simple

**Proposition 7.2.** *There exist semi-simple splicing languages which are not simple splicing languages.*

*Proof.* Consider the semi-simple splicing system  $(I, \sigma)$  in which  $\sigma = (\{a, b, c\}, S)$ ,  $I = \{abccab\}$ , and  $S = \{(a, 1; b, 1)\}$ . Let  $L = L(A, I, S)$ . We shall show that  $L$  has no constant symbols, and thus cannot be a simple splicing language.

**Claim 7.3.** *There are no words of the form  $xbby$  for  $x, y \in A^*$  in  $L$ , and the word  $ab$  is not in  $L$ .*

*Proof.* (By induction on  $k$  in  $\sigma^k(I)$ )

For  $k = 0$ , clearly  $ab \notin \sigma^0(I) = I$ . Also, there is no word of the form  $xbby$  in  $I$ . Let  $k > 0$  and  $i \leq k$ . Assume for all  $x, y \in A^*$  there is no word of the form  $xbby$  in  $\sigma^i(I)$ . Assume also that  $ab$  is not in  $\sigma^i(I)$ . Take words  $z = xay$  and  $z' = x'by'$  in  $\sigma^k(I)$ , and splice via  $(a, 1; b, 1)$ , obtaining  $w = xay' \in \sigma^{k+1}(I)$ . Note  $y' \neq by''$ , because if it were,  $z'$  would be of the form  $x'bby''$ , contradicting our induction hypothesis. So  $w \neq ab$ . Further,  $w$  cannot contain two consecutive  $b$ 's, because if it does, then either  $x$  or  $y'$  does, and that would contradict our induction hypothesis.  $\square$

**Claim 7.4.** *No word in  $L$  contains more than 2  $b$ 's, and any word in  $L$  which contains 2  $b$ 's must end in  $b$ . Further, if  $w \in L$  has the form  $w = uav$ , then  $|u|_b \leq 1$ .*

*Proof.* (By induction) No word in  $\sigma^0(I) = I$  contains three  $b$ 's. Also,  $abccab$  contains 2  $b$ 's, and ends in  $b$ . Further,  $abccab = uav$  where either  $u = 1$  or  $u = abcc$ . In either case,  $|u|_b \leq 1$ .

Let  $k > 0$  and  $i \leq k$ . Assume that no word in  $\sigma^i(I)$  contains three  $b$ 's, and that any word in  $\sigma^i(I)$  which contains two  $b$ 's ends in  $b$ . Further, assume that if  $w \in \sigma^i(I)$  and  $w = uav$ , then  $|u|_b \leq 1$ .

Let  $z = xay$  and  $z' = x'by'$  be words in  $\sigma^k(I)$ , and  $w = xay'$  the product of splicing. Since  $z, z'$  are in  $\sigma^k(I)$ ,  $|x|_b \leq 1$  and  $|y'|_b \leq 1$ . So  $w$  contains at most 2  $b$ 's.

Let us parse  $w = xay'$  as  $w = uav$ . Again, note  $|x|_b \leq 1$  by the induction hypothesis. If  $|x| \geq |u|$ , then  $x = u\beta$  for some  $\beta \in A^*$ , and  $|u|_b \leq 1$ , as desired. If  $|x| < |u|$ , then there exists  $\delta \in A^*$  such that  $u = xa\delta$  and  $y' = \delta av$ . If  $|\delta|_b = 0$ , then  $|u|_b \leq 1$ , and we are done. If  $|\delta|_b > 0$ , then  $\delta = u'bv'$  for some  $u', v' \in A^*$ . Further, since  $y'$  contains  $\delta$ , that means  $z' = x'by'$  contains 2  $b$ 's. So  $y'$  must end in  $b$ , by

the induction hypothesis. So in fact,  $y' = \delta = u'b$ . But  $y' = \delta av \neq \delta$ . This is a contradiction. Thus, if  $w = uav$ , then  $|u|_b \leq 1$ .

Finally, let us say  $w = xay'$  contains 2  $b$ 's. We have that  $xay'$  is such that  $|x|_b \leq 1$  by the argument just given. Thus, either  $|x|_b = |y'|_b = 1$ , or  $|y'|_b = 2$ . But  $|y'|_b = 2$  cannot occur, since  $z' = x'by'$  satisfies the induction hypothesis. So it must be that  $|y'|_b = 1$ , in which case  $z' = x'by'$  contains 2  $b$ 's. By the induction hypothesis, then,  $z'$  ends in  $b$ . Thus  $y'$  ends in  $b$ , and  $w = xay'$  ends in  $b$ .  $\square$

**Claim 7.5.** *No word in  $L$  contains three consecutive  $c$ 's.*

*Proof.* (Induction) It is true for  $\sigma^0(I)$ . Assume true for  $\sigma^i(I)$ ,  $i \leq k$  and  $k > 0$ . Let  $z = xay$  and  $z' = z'by'$  be words in  $\sigma^k(I)$ , as usual. The splicing product  $w = xay'$  cannot contain three consecutive  $c$ 's, because neither  $x$  or  $y'$  does, by induction hypothesis.  $\square$

Now let us note the following facts about  $L = \sigma^*(I)$ . The word  $abccab \in L$  can be parsed as  $(1)a(bccab)$  and as  $(abcc)a(b)$ . Using these parsings, if  $a$  were a constant of  $L$ , the word  $(1)a(b) = ab$  would be in  $L$ . But  $ab \notin L$ , by Claim 7.3. So  $a$  is not a constant of  $L$ .

The word  $abccab$  can be parsed as  $(abc)c(ab)$  and as  $(ab)c(cab)$ . If  $c$  were constant in  $L$ , then these parsings would yield  $(abc)c(cab) \in L$ . Claim 7.5, however, shows that no such word can be in  $L$ . So  $c$  cannot be a constant of  $L$ .

Finally, the word  $abccab \in L$  can be parsed as  $(abcca)b(1)$  and as  $(a)b(ccab)$ . If  $b$  were constant in  $L$ , these parsings would yield  $(abcca)b(ccab) \in L$ . By Claim 7.4, however, no word with 3  $b$ 's is in  $L$ .

Thus, none of the symbols in  $A$  are constants of  $L$ , and  $L$  cannot be a simple splicing language.  $\square$

## 2 Null Context $\not\subseteq$ Semi-Simple

**Proposition 7.6.** *There exist null context splicing languages which are not semi-simple splicing languages.*

*Proof.* Consider the null context splicing language  $L$  generated by the system  $(A, I, S) = (\{a, b\}, \{abababb\}, \{(ba, 1; ba, 1)\})$ .

**Claim 7.7.**  $L = a(ba)^+bb$ .

*Proof.* Consider the following splice of words in  $L$  for  $i \geq 2$ :

$$(a(ba)^{i-1}(\underline{ba})bb, abab\underline{abb}) \vdash_{(ba, 1; ba, 1)} a(ba)^{i+1}bb$$

So we see  $a(ba)^i bb \subseteq L$  for  $i \geq 2$ . We may also obtain  $ababb$  by splicing as follows:

$$(abababb, abab\underline{abb}) \vdash_{(ba, 1; ba, 1)} ababb$$

Thus  $a(ba)^+bb \subseteq L$ .

For the other inclusion, note that  $I = \{abababb\} \subset a(ba)^+bb$ . Further, it is clear that given two words  $z = a(ba)^i bb$  and  $z' = a(ba)^j bb$  in  $a(ba)^+bb$ , any splice via  $r$  yields another word in  $a(ba)^+bb$ .  $\square$

Now we claim the following:

**Claim 7.8.**  $L = a(ba)^+bb$  is not a semi-simple splicing language.

*Proof.* We shall show that any semi-simple rule which is useful in  $L$  does not respect  $L$ . Since  $L$  is infinite, we shall thus show that  $L$  cannot be a semi-simple splicing language. We look at the four cases:

(i)  $(a, 1; a, 1)$ : Consider the following splice using words in  $L$  and  $(a, 1; a, 1)$ :

$$(\underline{a}babb, ab\underline{abb}) \vdash_{(a, 1; a, 1)} abb. \text{ Since } abb \notin L, (a, 1; a, 1) \notin R(L).$$

(ii)  $(a, 1; b, 1)$ : Consider the splice of words in  $L$ :  $(\underline{a}babb, abab\underline{b}) \vdash_{(a, 1; b, 1)} a$ . Since  $a \notin L$ ,  $(a, 1; b, 1) \notin R(L)$ .

(iii)  $(b, 1; a, 1)$ : Consider the splice of words in  $L$ :  $(ab\underline{abb}, ab\underline{abb}) \vdash_{(b, 1; a, 1)} abbb \notin L$ . So  $(b, 1; a, 1) \notin R(L)$ .

(iv)  $(b, 1; b, 1)$ : Consider finally the splice of words in  $L$ :  $(ab\underline{abb}, abab\underline{b}) \vdash_{(b, 1; b, 1)} abb \notin L$ . So  $(b, 1; b, 1) \notin R(L)$ .

□

By the above claims, we see that  $L = a(ba)^+bb$  is a null context splicing language which is not a semi-simple splicing language. □

### 3 Arrow Graphs

We begin with a semi-simple splicing scheme  $\sigma = (A, S)$ , and a finite initial set  $I$ . Let  $L = \sigma^*(I)$ . We augment  $(I, \sigma)$  with new symbols  $X$  and  $Y$  not in  $A$  as follows:

$$\begin{aligned}\bar{A} &= A \cup \{X, Y\} \\ \bar{S} &= S \cup \{(X, 1; X, 1), (Y, 1; Y, 1)\} \\ \bar{\sigma} &= (\bar{A}, \bar{S}) \\ \bar{I} &= XIY \\ \bar{L} &= \bar{\sigma}^*(\bar{I})\end{aligned}$$

**Definition 7.9.** An **arrow** is a triple  $e = (a, w, a')$  in  $\bar{A} \times A^* \times \bar{A}$ .

We normally write an arrow as  $e = a \xrightarrow{w} a'$ . Given such an arrow, we say  $a$  is the *initial vertex* of  $e$  and  $a'$  is the *terminal vertex* of  $e$ .

**Definition 7.10.** Let  $e$  and  $e'$  be arrows. We say the pair  $e, e'$  is **adjacent** if the terminal vertex of  $e$  is the initial vertex of  $e'$ .

Let  $\bar{L} = L(\bar{A}, \bar{I}, \bar{S})$ . We define the arrow graph of  $(\bar{A}, \bar{I}, \bar{S})$  as follows:

**Definition 7.11.** The **arrow graph**  $G$  of  $(\bar{A}, \bar{I}, \bar{S})$  is a directed graph having vertex set  $\bar{A}$  and edge set  $E \subseteq \bar{A} \times A^* \times \bar{A}$ . An arrow  $e = a \xrightarrow{w} a'$  is an **edge** in  $E$  if there exists  $b \in \bar{A}$  for which  $(a, 1; b, 1)$  in  $\bar{S}$ , and  $bwa'$  in  $\text{Fac } \bar{L}$ .

**Definition 7.12.** A **path** from  $a_0$  to  $a_n$  is a finite sequence  $\pi = \langle e_1, e_2, \dots, e_n \rangle$  of edges where the initial vertex of  $e_1$  is  $a_0$ , the terminal vertex of  $e_n$  is  $a_n$ , and the pairs  $e_k, e_{k+1}$  are adjacent for  $1 \leq k < n$ .

We have a non-standard definition for the label of a path:

**Definition 7.13.** If  $\pi = \langle e_1, e_2, \dots, e_n \rangle$  and  $e_k = a_{k-1} \xrightarrow{w_k} a_k$ , the **label** of  $\pi$  is defined as follows:  $\lambda(\pi) = a_0 w_1 a_1 w_2 \dots a_{n-1} w_n a_n$ .

We define the language accepted by  $G$ , written as  $L(G)$  as follows:

$$L(G) = \{\lambda(\pi) \mid \pi \text{ is a path from } X \text{ to } Y\}.$$

The product of two arrows  $e_1$  and  $e_2$  is defined if and only if the pair  $e_1, e_2$  is adjacent.

**Definition 7.14.** Let  $e_1 = a_1 \xrightarrow{w_1} a_2$  and  $e_2 = a_2 \xrightarrow{w_2} a_3$  for  $a_2 \in A$ . The **product** of  $e_1$  and  $e_2$ , written  $e_1 \cdot e_2$ , is the arrow  $a_1 \xrightarrow{w_1 a_2 w_2} a_3$ .

**Note 7.15.** The product  $e_1 \cdot e_2 \cdots e_n$  is well-defined if the  $e_j \cdot e_{j+1}$  are defined for  $1 \leq j < n$ , i.e. the product operation is associative.

**Definition 7.16.** An edge  $e$  is **prime** if it is not the product of two edges.

**Definition 7.17.** Given  $G = (\bar{A}, E)$ , we define the **prime subgraph** of  $G$ , denoted  $G_0$ , as follows:  $G_0 = (\bar{A}, E_0)$  where  $E_0 = \{e \in E \mid e \text{ is prime}\}$ .

## 4 Arrow Graph Results

Let  $\sigma = (A, S)$  be a semi-simple splicing scheme, and  $I$  a finite initial set. Let  $L = \sigma^*(I)$ , and let  $\bar{L}$  be the semi-simple splicing language generated by the augmented splicing system  $(\bar{A}, \bar{I}, \bar{S})$ , as defined in the previous section. Let  $G$  be the arrow graph associated with  $(\bar{A}, \bar{I}, \bar{S})$ , and  $G_0$  the prime subgraph of  $G$ .

**Lemma 7.18.**  $\bar{L} = XLY$ .

*Proof.* ( $\bar{L} \subseteq XLY$ ): Let  $w \in \bar{I}$ . Then  $w \in XIY \subseteq XLY$ . Take two words  $z = XwY$  and  $z' = Xw'Y$  in  $XLY$ , and splice via a rule  $r = (a, 1; b, 1) \in \bar{S}$ . If  $r = (X, 1; X, 1)$ , the result is  $Xw'Y$  is in  $XLY$  since  $w' \in L$ . If  $r = (Y, 1; Y, 1)$ , the result  $XwY$  again is in  $XLY$ . If  $r = (a, 1; b, 1)$  for  $a, b \in A$ , then  $w = xay$  and  $w' = x'by'$ . Since  $r \in S$  in this case, the product  $xay'$  of splicing  $w$  and  $w'$  is in  $L = \sigma^*(I)$ , so  $Xxay'Y \in XLY$ .

$(XLY \subseteq \bar{L})$ : (Induction on  $k$  in  $\sigma^k(I)$ .) Let  $w \in \sigma^0(I)$ . Then  $XwY \in XIY = \bar{I} \subseteq \bar{L}$ . Let  $k > 0$ , and assume for all  $i \leq k$  that  $w \in \sigma^i(I)$  implies  $XwY \in \bar{L}$ . Take  $z, z'$  in  $\sigma^k(I)$ , and consider  $w$ , the product of splicing  $z$  and  $z'$  via a rule  $r = (a, 1; b, 1)$  in  $S$ . So  $z = xay$ ,  $z' = x'by'$  and  $w = xay'$ . By induction hypothesis,  $XzY$  and  $Xz'Y$  are in  $\bar{L}$ . Since  $S \subset \bar{S}$ , the splicing product  $Xxay'Y = XwY$  is in  $\bar{L}$ , as desired.  $\square$

**Lemma 7.19.** *The set  $E$  of edges in  $G = (\bar{A}, E)$  is closed under the product operation.*

*Proof.* Let  $e_1 = a_1 \xrightarrow{w_1} a_2$  and  $e_2 = a_2 \xrightarrow{w_2} a_3$  be adjacent edges in  $G$ . Since  $e_1$  is an edge in  $G$ , there is a rule  $r_1 = (a_1, 1; b_1, 1) \in \bar{S}$  for some  $b_1 \in \bar{A}$ , and a factor  $b_1w_1a_2$  in  $\bar{L}$ . Since  $e_2$  is an edge, there is a rule  $r_2 = (a_2, 1; b_2, 1) \in \bar{S}$  and a factor  $b_2w_2a_3$  in  $\bar{L}$ . Choose words  $w_1 = x_1b_1w_1a_2y_1$  and  $w_2 = x_2b_2w_2a_3y_2$  in  $\bar{L}$ , and splice via  $r_2$ , obtaining  $x_1b_1w_1a_2w_2a_3y_2 \in \bar{L}$ . So  $b_1w_1a_2w_2a_3y_2$  is a factor in  $\bar{L}$ . This and  $r_1 \in \bar{S}$  determine that the edge  $e = a_1 \xrightarrow{w_1a_2w_2} a_3$  is in  $E$ . Since  $e = e_1 \cdot e_2$ , we have the desired conclusion.  $\square$

Given a single edge  $e$ , we denote the label of  $e$  by  $\lambda(e)$ , rather than  $\lambda(\langle e \rangle)$ .

**Lemma 7.20.** *If  $\pi = \langle e_1, e_2, \dots, e_n \rangle$  is a path in  $G$  from  $a_0$  to  $a_n$ , then  $e = e_1 \cdot e_2 \cdots e_n$  is an edge in  $G$  from  $a_0$  to  $a_n$  and  $\lambda(\pi) = \lambda(e)$ .*

*Proof.* Since  $\pi$  is a path, consecutive edges are adjacent. Let  $e_k = a_{k-1} \xrightarrow{w_k} a_k$  for  $1 \leq k < n$ . The product  $e = e_1 \cdot e_2 \cdots e_n$  is well-defined, by Note 7.15. Further,  $e = a_0 \xrightarrow{w_1a_1w_2 \cdots w_n} a_n$ . By Lemma 7.19,  $e$  is an edge from  $a_0$  to  $a_n$  in  $G$ . By definition,  $\lambda(\pi) = a_0w_1a_1 \dots w_na_n = (a_0)(w_1a_1 \dots w_n)(a_n) = \lambda(e)$ .  $\square$

**Lemma 7.21.**  *$X \xrightarrow{w} Y$  is an edge in  $G$  if and only if  $XwY \in \bar{L}$ .*

*Proof.* Let  $e = X \xrightarrow{w} Y$  be an edge in  $G$ . We know that  $(X, 1; X, 1) \in \bar{S}$  is the only rule containing site  $X$ . Since  $e$  is an edge, there must be a factor  $XwY$  in  $\bar{L}$ . By Lemma 7.18, it must be that  $XwY \in \bar{L}$ .

Now say  $XwY \in \bar{L}$ . Then  $XwY \in \text{Fac } \bar{L}$ . Since  $(X, 1; X, 1) \in \bar{S}$ ,  $X \xrightarrow{w} Y$  is an edge.  $\square$

**Lemma 7.22.**  $L(G) = \bar{L}$ .

*Proof.* Let  $XwY \in L(G)$ . Then there is a path  $\pi = \langle e_1, e_2, \dots, e_n \rangle$  from  $X$  to  $Y$  such that  $\lambda(\pi) = XwY$ . Let  $e = e_1 \cdot e_2 \cdots e_n$ . By Lemma 7.20  $e$  is an edge in  $G$  from  $X$  to  $Y$ , and in fact  $e = X \xrightarrow{w} Y$ . By definition of edge, there is  $b \in \bar{A}$  such that  $(X, 1; b, 1)$  and  $bwY \in \text{Fac } \bar{L}$ . Since  $(X, 1; X, 1)$  can be the only such rule, we have  $XwY \in \text{Fac } \bar{L}$ . By Lemma 7.18,  $XwY \in \bar{L}$ .

Now say  $XwY$  is in  $\bar{L}$ . Then  $XwY$  is a factor in  $\bar{L}$ , and since  $(X, 1; X, 1) \in \bar{S}$ , we have edge  $e = X \xrightarrow{w} Y$  in  $G$ . Since  $e$  begins at  $X$  and ends at  $Y$ ,  $XwY = \lambda(e) \in L(G)$ .  $\square$

**Lemma 7.23.** *Every edge is the product of a sequence of prime edges.*

*Proof.* (Induction on  $|\lambda(e)|$ .) Let  $e = a_0 \xrightarrow{w} a_1$  be an edge in  $G$ . If  $|\lambda(e)| = 2$ , then  $w = 1$  and cannot be factored as  $uav$ , and  $e$  is prime. Let  $k > 2$ , and assume that any edge having length less than or equal to  $k$  is the product of a sequence of prime edges. Let  $e = a_0 \xrightarrow{w} a_1$  be an edge in  $G$  whose label has length  $k + 1$ . If  $e$  is prime, we are done. If not, then  $e = e_1 \cdot e_2$  where  $|\lambda(e_1)|, |\lambda(e_2)|$  are less than  $k$ . By the induction hypothesis, both  $e_1$  and  $e_2$  are products of prime edges. So  $e_1 = e_{1(1)} \cdots e_{1(k)}$ ,  $e_2 = e_{2(1)} \cdots e_{2(l)}$ , and  $e = e_1 \cdot e_2 = e_{1(1)} \cdots e_{1(k)} \cdot e_{2(1)} \cdots e_{2(l)}$  is also the product of prime edges.  $\square$

**Lemma 7.24.** (*Prefix Edge*) Let  $a_1 \xrightarrow{w_1 a w_2} a_2$  be an edge in  $G$ , where  $a \in A$  and  $w_1, w_2 \in A^*$ . Then  $a_1 \xrightarrow{w_1} a$  is an edge in  $G$ .

*Proof.* Since  $a_1 \xrightarrow{w_1 a w_2} a_2$  is an edge, there is  $b_1 \in \bar{A}$  such that  $(a_1, 1; b_1, 1) \in \bar{S}$  and  $b_1 w_1 a w_2 a_2 \in \text{Fac } \bar{L}$ . So  $b_1 w_1 a \in \text{Fac } \bar{L}$ , and  $a_1 \xrightarrow{w_1} a$  is an edge in  $G$ .  $\square$

**Lemma 7.25.** *If  $a_1 \xrightarrow{w} a_2$  is a prime edge in  $G$ , then  $w$  is a factor in  $I$ .*

*Proof.* (Induction on smallest  $k$  for which there exists  $b_1 \in \bar{A}$ ,  $(a_1, 1; b_1, 1) \in \bar{S}$  and a factor  $b_1 w a_2$  in  $\bar{\sigma}^k(\bar{I})$ .) Let  $e = a_1 \xrightarrow{w} a_2$  be a prime edge in  $G$  for which there is  $b_1$  such that  $(a_1, 1; b_1, 1) \in \bar{S}$  and  $b_1 w a_2 \in \bar{\sigma}^0(\bar{I}) = \bar{I}$ . Since  $\bar{I} = XIY$ ,  $w$  must be in  $\text{Fac } I$ . Let  $k > 0$ , and  $i \leq k$ . Assume that if  $e = a \xrightarrow{w} a'$  is a prime edge for which there exists  $b \in A$ ,  $(a, 1; b, 1) \in \bar{S}$  and  $b w a \in \text{Fac } \bar{\sigma}^i(\bar{I})$ , then  $w \in \text{Fac } I$ .

Now let  $e = a_1 \xrightarrow{w} a_2$  be a prime edge for which there exists  $b_1 \in A$ ,  $(a_1, 1; b_1, 1) \in \bar{S}$  and  $b_1 w a_2 \in \text{Fac}(\bar{\sigma}^{k+1}(\bar{I}) \setminus \bar{\sigma}^k(\bar{I}))$ . So there is a word  $\bar{x} b_1 w a_2 \bar{y} \in \bar{\sigma}^{k+1}(\bar{I})$ . Since  $\bar{x} b_1 w a_2 \bar{y} \notin \bar{\sigma}^k(\bar{I})$ , there must be words  $z = x a y$  and  $z' = x' b y'$  in  $\bar{\sigma}^k(\bar{I})$  and a rule  $(a, 1; b, 1)$  in  $\bar{S}$  such that

$$(x a y, x' b y') \vdash_{(a, 1; b, 1)} x a y' = \bar{x} b_1 w a_2 \bar{y}.$$

Note that, since  $b_1 w a_2 \notin \text{Fac} \bar{\sigma}^k(\bar{I})$ , there must be  $\gamma \in \bar{A}^*$  and a prefix  $\delta \neq 1$  of  $y'$  such that  $\gamma a \delta = b_1 w a_2$ . (Otherwise  $b_1 w a_2$  is either entirely contained in  $x a$  or  $y'$ , a contradiction.) We examine the cases:

- (i) Case  $\gamma \neq 1$ . Then  $b_1 w a_2 = b_1 w_1 a w_2 a_2$  for  $w_1, w_2 \in A^*$  such that  $w_2 a_2 = \delta$ . By Lemma 7.24,  $e_1 = a_1 \xrightarrow{w_1} a$  is an edge. Since  $w_2 a_2 = \delta$  is a prefix of  $y'$ , we see that  $b w_2 a_2$  is a factor of  $z'$  in  $\bar{L}$ . Since  $(a, 1; b, 1) \in \bar{S}$ ,  $e_2 = a \xrightarrow{w_2} a_2$  is an edge in  $G$ . Finally, we see  $e_1 \cdot e_2 = e$ , contradicting the fact that  $e$  is prime.
- (ii) Case  $\gamma = 1$  and  $a \delta = b_1 w a_2$ . Then  $\delta = w a_2$ , and  $z' = x' b y' = x' b w a_2 \eta$  for some  $\eta \in A^*$ . Note  $b w a_2$  is a factor in  $\bar{\sigma}^k(\bar{I})$ , and  $(a, 1; b, 1)$  in  $\bar{S}$  imply there is an edge  $e' = a \xrightarrow{w} a_2$ . If  $e'$  is not prime, then there exist edges  $\tilde{e}$  and  $e_2$ , and  $a' \in A$  such that  $\tilde{e} = a \xrightarrow{u} a'$ ,  $e_2 = a' \xrightarrow{v} a_2$ , and  $u a' v = w$ . So  $e = a_1 \xrightarrow{w} a_2 = a_1 \xrightarrow{u a' v} a_2$ , and the prefix edge  $e_1 = a_1 \xrightarrow{u} a'$  exists by Lemma 7.24. The product  $e_1 \cdot e_2 = e$ , contradicting the fact that  $e$  is prime. Thus  $e'$  must be prime. Since  $b w a_2$  is a factor in  $\bar{\sigma}^k(\bar{I})$ , and  $(a, 1; b, 1)$  is in  $\bar{S}$ ,  $w$  must be in  $\text{Fac} I$  by the induction hypothesis.

□

**Lemma 7.26.** *The prime subgraph  $G_0$  of  $G$  is finite.*

*Proof.* This is an immediate consequence of the definition of  $G_0$ , Lemma 7.25, and the fact that  $I$  is finite. □

**Lemma 7.27.**  $L(G_0) = \bar{L}$ .

*Proof.* Since  $G_0$  is a subgraph of  $G$ , and  $L(G) = \bar{L}$  by Lemma 7.22,  $L(G_0) \subseteq \bar{L}$ . Now let  $X w Y \in \bar{L}$ . Then  $e = X \xrightarrow{w} Y$  is an edge by Lemma 7.21. By Lemma 7.23,

$e = e_1 \cdots e_n$ , where each  $e_i$  is a prime edge. Since  $\lambda(e) = \lambda(e_1 \cdots e_n) = XwY$  by Lemma 7.20, we see  $XwY \in L(G_0)$ .  $\square$

**Lemma 7.28.** (*Approximate Factorization*) Let  $L = L(A, I, S)$  where  $\sigma = (A, S)$  is semi-simple. Let the arrow graph  $G = (\bar{A}, E)$  be defined as usual for the augmented language  $\bar{L} = L(\bar{A}, \bar{I}, \bar{S})$ . Let  $N = \max\{|\lambda(e)| \mid e \text{ is a prime edge in } G\}$ . If  $e = a_1 \xrightarrow{xy} a_2$  is an edge in  $G$ , and  $|w| > N$ , then there are edges  $e_1 = a_1 \xrightarrow{xu} a$  and  $e_2 = a \xrightarrow{vy} a_2$  in  $G$  such that  $e = e_1 \cdot e_2$  and  $w = uav$ .

*Proof.* Let  $e = a_1 \xrightarrow{xy} a_2$  be an edge in  $G$  such that  $|w| > N$ . By Lemma 7.23, there exists a sequence  $e_1, e_2, \dots, e_n$  of prime edges such that  $e = e_1 \cdot e_2 \cdots e_n$ . Take the smallest  $j$  such that  $|\lambda(e_1 \cdots e_j)| > |a_1x|$ . Note  $|\lambda(e \cdot e')| = |\lambda(e)| + |\lambda(e')| - 1$ , by definition of the product and of  $\lambda$ . Since  $|\lambda(e_j)| < N$ , we have

$$\begin{aligned} |\lambda(e_1 \cdots e_j)| &= |\lambda(e_1 \cdots e_{j-1})| + |\lambda(e_j)| - 1 \\ &< |a_1x| + N - 1 \\ &\leq |a_1xw|. \end{aligned}$$

Thus  $\lambda(e_1 \cdots e_j) = a_1xu$  where  $ua$  is a prefix of  $w$ , and the factorization of  $e$  as  $(e_1 \cdots e_j) \cdot (e_{j+1} \cdots e_n)$  has the desired property.  $\square$

**Lemma 7.29.** Let  $(A, I, S)$  be a semi-simple splicing system, and say  $L = L(A, I, S)$ . Let arrow graph  $G = (\bar{A}, E)$  be defined as usual for the augmented system  $(\bar{A}, \bar{I}, \bar{S})$ , and let  $\bar{L} = L(\bar{A}, \bar{I}, \bar{S})$ . Let  $N = \max\{|\lambda(e)| \mid e \text{ is a prime edge in } G\}$ . If  $w \in A^*$  and  $|w| > N$ , then  $w$  is a constant of  $L$ .

*Proof.* Let  $xwy$  and  $x'wy'$  be words in  $L$ . So  $XxwyY$  and  $Xx'wy'Y$  are in  $\bar{L} = XLY$ , and there are edges  $e = X \xrightarrow{xy} Y$  and  $e' = X \xrightarrow{x'wy'} Y$  in  $G$ . By Lemma 7.28, there is an approximate factorization  $\tilde{e} \cdot e_2$  of  $e'$ , where  $\tilde{e} = X \xrightarrow{x'u} a$ ,  $e_2 = a \xrightarrow{vy'} Y$  and  $w = uav$ . Now  $e = X \xrightarrow{xuavy} Y$ , and by Lemma 7.24 there is a prefix edge  $e_1 = X \xrightarrow{xu} a$  of  $e$ . Consider  $e_1 \cdot e_2$ . First, note that this product is defined since the  $e_1, e_2$  are an adjacent pair. Further,  $e_1 \cdot e_2 = X \xrightarrow{xuavy'} Y$  is an edge from  $X$  to  $Y$  in  $G$ . Thus  $\lambda(e_1 \cdot e_2) = Xuavy'Y \in L(G) = XLY$ , and we see  $xuavy' = xwy' \in L$ . A

similar argument using an approximate factorization of  $e$  and prefix edge of  $e'$  reveals that  $x'wy$  is in  $L$  as well.  $\square$

**Theorem 7.30.** *If  $L$  is a semi-simple language, then  $L$  is strictly locally testable.*

*Proof.* Let  $L = L(I, \sigma)$  where  $\sigma$  is a semi-simple splicing scheme. If  $L$  is finite, then  $L$  is strictly locally testable. Construct the arrow graph  $G$  of the augmented semi-simple language  $\bar{L} = XLY$ , as usual. Let  $N = \max\{|\lambda(\pi)| \mid \pi \text{ is a prime edge in } G\}$ . By Lemma 7.29, every word in  $L$  having length greater than  $N$  is a constant word in  $L$ , and  $L$  is strictly locally testable by Definition 6.7 given in Chapter 6.  $\square$

# Chapter 8

## Semi-Null Splicing Systems

In Chapter 7, we discussed semi-simple splicing systems in which all rules have sites which are single alphabet symbols. The rule sites in such a system are not necessarily constant factors of the splicing language. We saw, however, that splicing languages generated by semi-simple systems contain only finitely many words which are not constant. We now shall generalize further to a class of splicing languages which have sites in  $A^+$ .

**Definition 8.1.** *Let  $(A, I, S)$  be a splicing system in which  $I$  and  $S$  are finite and every rule in  $S$  has the form  $(u, 1; v, 1)$ , where  $u, v \in A^+$ . We say  $\sigma = (A, S)$  is a **semi-null splicing scheme**,  $(A, I, S)$  is a **semi-null splicing system**, and  $\sigma^*(I)$  is a **semi-null splicing language**.*

If  $L$  is a semi-null splicing language which can be generated by a semi-null system having exactly one rule, we say  $L$  is a *one rule* semi-null language.

### 1 Semi-Null $\not\subseteq$ NCH

The next proposition shows that the class of semi-null languages lies outside the class of null-context splicing languages.

**Proposition 8.2.** *There exist semi-null languages which are not null context splicing languages.*

*Proof.* Consider the language  $L = b(aa)^+$ . It is easy to verify that  $L$  is generated by the semi-null system  $(A, I, S)$  where  $S = \{(baa, 1; b, 1)\}$  and  $I = \{baa\}$ . Assume  $L$  is a null-context splicing language. Recall, all rules in a null context splicing system  $(A, I, S)$  generating  $L$  have either the form  $(u, 1; u, 1)$  or  $(1, u; 1, u)$ , and each site is a constant of  $L$ . So the rules must be of the form  $(ba^i, 1; ba^i, 1)$  or  $(1, ba^i; 1, ba^i)$ , where  $i > 0$ . Notice that, given two words  $ba^{2k}$  and  $ba^{2l}$  in  $L$ , and  $r \in S$ , the splicing product cannot have length greater than  $\max\{|ba^{2k}|, |ba^{2l}|\}$ . Thus, any finite initial set and finite null context rule set can only generate a finite subset of  $L$ . This contradicts  $L = L(A, I, S)$ .  $\square$

We shall give a set of conditions under which one rule semi-null languages are infinite. We shall then show that infinite semi-null splicing languages have infinitely many constant factors, and thus contain infinitely many constant words. First, a lemma.

## 2 A Lemma

**Lemma 8.3.** *Let  $L \subseteq A^*$  be a regular language. Let  $u \in A^*$  and  $v \in A^+$ . If the rule  $r = (uv, 1; u, 1) \in R_{fac}(L)$ , then  $facC(L)$  is infinite.*

*Proof.* Let  $\text{Syn } L$  denote the syntactic monoid of  $L$ . Let  $K = \text{Card}(\text{Syn } L)$ . We have the following claims:

**Claim 8.4.** *If  $xv^i y \in L$  for some  $x, y \in A^*$ , then  $\{xv^i y \mid i \geq 1\} \subseteq L$ .*

*Proof.* Say  $xv^i y \in L$  for some  $x, y \in A^*$ . Since  $r = (uv, 1; u, 1)$  respects  $L$ , the following splice generates a product in  $L$  for each  $i \geq 1$ :

$$(xv^i y, xv^i y) \vdash_r xv^{i+1} y \in L.$$

$\square$

**Claim 8.5.** *There exist  $j, k, 1 \leq j < k \leq K + 1$  such that  $uv^{j+i} \equiv_L uv^{k+i}$  for all  $i \geq 0$ .*

*Proof.* By the Pigeon Hole Principle, there exist integers  $j, k$  where  $1 \leq j < k \leq K+1$ , such that:

$$uv^j \equiv_L uv^k$$

Consequently, for this  $j$  and  $k$ , and for all  $i \geq 0$ ,  $uv^{j+i} \equiv_L uv^{k+i}$  holds as well.  $\square$

Let  $k$  be as defined in Claim 8.5. We have yet another claim.

**Claim 8.6.** *The string  $uv^{k+1}$  is a constant factor in  $L$ .*

*Proof.* Because  $r$  is useful, there exist  $x, y \in A^*$  such that  $xuvy \in L$ . By Claim 8.4, the word  $xuv^{k+1}y$  is in  $L$ . So, let  $xuv^{k+1}y$  and  $x'uv^{k+1}y'$  be words in  $L$ . We may splice via  $r$  as follows:

$$(xuvv^ky, x'uv^{k+1}y') \vdash_r xuvv^{k+1}y' = xuv^{k+2}y' \in L.$$

By Claim 8.5, we see  $xuv^{j+2}y' \in L$ . Notice that  $(k - j - 1) \geq 0$  since  $j < k$ . So by Claim 8.4, the word  $xuv^{(j+2)+(k-j-1)}y' = xuv^{k+1}y' \in L$ , as desired.  $\square$

We know by Claim 8.6 that the word  $uv^{k+1}$  is a constant factor of  $L$ . So there exist  $x, y \in A^*$  such that  $xuv^{k+1}y \in L$ . By Lemma 2.13, every word in the set  $\mathcal{S} = \{uv^{k+1}v^i \mid i \geq 0\}$  is a constant of  $L$ . By Claim 8.4, the set  $xuv^{k+1}v^*y \subseteq L$ . So every word in  $\mathcal{S}$  is a factor in  $L$ . Thus  $\text{fac}C(L)$  is infinite.  $\square$

### 3 One Rule Semi-Null Languages

Let  $S = \{(u, 1; v, 1)\}$ , where  $u$  and  $v$  are non-empty words over alphabet  $A$ . Let  $I$  be an initial set such that  $u, v \in \text{Fac } I$ . Let  $\sigma = (A, S)$ . Let  $L = \sigma^*(I)$ .

Let **A** and **B** represent the following statements:

**A:**  $\exists \alpha, \beta, \gamma \in A^*$ ,  $\beta \neq 1$  such that  $\alpha u = \gamma v \beta \in \text{Fac } L$

**B:**  $\exists \alpha, \beta, \gamma \in A^*$ ,  $\beta \neq 1$  such that  $\alpha u = u \beta \in \text{Fac } L$  and  $v \beta \in \text{Fac } L$

**Note 8.7.** *If **B** holds and  $|\alpha| \geq |u|$ , then **A** holds.*

*Proof.* This is clear when one considers that  $\mathbf{B}$  and  $|\alpha| \geq |u|$  means that  $\beta$  contains  $u$  as a factor. Since  $v\beta$  is a factor in  $L$ , there is a word  $xv\beta y \in L$ , where  $\beta = \beta'u\beta''$ . So  $(v\beta')u = v(\beta'u)$ ,  $v\beta' \neq 1$  since  $v \neq 1$ , and  $v(\beta'u)$  is a factor in  $L$ .  $\square$

We define the following integers for  $k \geq 0$ :

$$P^k = \max\{|\xi| \mid \exists \eta \text{ such that } \xi u \eta \in \sigma^k(I)\}$$

$$S^k = \max\{|\eta| \mid \text{exists } \xi \text{ such that } \xi v \eta \in \sigma^k(I)\}$$

We shall also define statements  $\mathbf{A}^k$  and  $\mathbf{B}^k$  as follows:

$$\mathbf{A}^k : \exists \alpha, \beta, \gamma \in A^*, \beta \neq 1 \text{ such that } \alpha u = \gamma v \beta \text{ and } v \beta \in \text{Fac } \sigma^k(I)$$

$$\mathbf{B}^k : \exists \alpha, \beta, \gamma \in A^*, \beta \neq 1 \text{ such that } \alpha u = u \beta \text{ and } v \beta \in \text{Fac } \sigma^k(I)$$

Now, a lemma:

**Lemma 8.8.** *The following implications hold for  $L = \sigma^*(I)$ :*

$$(i) P^{k+1} > P^k \implies \mathbf{B}^k$$

$$(ii) S^{k+1} > S^k \implies \mathbf{A}^k.$$

*Proof.* (i) Suppose  $\xi u \eta \in \sigma^{k+1}(I)$  and  $|\xi| > P^k$ . Since  $\xi u \eta \notin \sigma^k(I)$ , there must exist words  $xu\eta$  and  $\bar{x}v\bar{\eta}$  in  $\sigma^k(I)$  such that  $\xi u \eta = xu\bar{\eta}$ . Since  $|x| \leq P^k$ , it follows that  $|x| < |\xi|$ , so there is an  $\alpha \neq 1$  such that  $\xi = x\alpha$ . Notice that  $x\alpha u \eta = xu\bar{\eta}$  implies  $\alpha u \eta = u\bar{\eta}$ . So we may parse  $\bar{\eta}$  as  $\beta\bar{\eta}'$  in such a way that  $\alpha u = u\beta$ , and  $\beta \neq 1$  since  $\alpha \neq 1$ . Finally,  $\bar{x}v\bar{\eta} = \bar{x}v\beta\bar{\eta}'$ , so  $v\beta \in \text{Fac } \sigma^k(I)$ . So  $\mathbf{B}^k$  holds for  $L$ .

(ii) Suppose  $\xi v \eta \in \sigma^{k+1}(I)$  and  $|\eta| > S^k$ . Then there exist  $xu\eta$  and  $\bar{x}v\bar{\eta} \in \sigma^k(I)$  such that  $\xi v \eta = xu\bar{\eta}$ . Since  $|\eta| > |\bar{\eta}|$ , we may parse  $\eta$  as  $\beta\bar{\eta}$  where  $\beta \neq 1$ . Since  $\xi v \beta \bar{\eta} = xu\bar{\eta}$ , we see  $\xi v \beta = xu$ . So there exist  $\alpha$  and  $\gamma$  in  $A^*$  such that  $x = x'\alpha$ ,  $\xi = x'\gamma$  and  $\alpha u = \gamma v \beta$ , where  $x'$  denotes the common prefix of  $\xi$  and  $x$ . Notice  $\alpha = 1$  if  $|x| \leq |\xi|$ , and  $\gamma = 1$  if  $|x| \geq |\xi|$ . In any case, we have

$xu = x'\alpha u = x'\gamma v\beta = \xi v\beta$ . So  $\alpha u = \gamma v\beta$ . Since  $\alpha u$  is a factor of  $xu \in \text{Fac } \sigma^k(I)$ ,  $v\beta \in \text{Fac } \sigma^k(I)$ . Thus **A**<sup>k</sup> holds for  $L$ .

□

**Theorem 8.9.** *Let  $L$  be a one rule semi-null splicing language. Then  $L$  is infinite if and only if at least one of the statements **A** or **B** holds for  $L$ .*

*Proof.* Let  $L = \sigma^*(I)$  where  $\sigma = (A, S)$  and  $S = \{(u, 1; v, 1)\}$ ,  $u, v \in A^+$ . First we shall show that  $L$  is infinite if statement **A** or statement **B** holds for  $L$ .

Say **A** holds for  $L$ . Then there exists a word  $x\alpha u y = x\gamma v\beta y \in L$ . We splice as follows for all  $i > 0$ :

$$(x\alpha u y, x\gamma v\beta^i y) \vdash x\alpha u\beta^i y = x\gamma v\beta^{i+1} y \in L.$$

Since  $\beta \neq 1$ , we see  $L$  is infinite.

Say **B** holds for  $L$ . There exists a word  $xu y'$  in  $L$ , since  $(u, 1; v, 1)$  is useful. There also exists a word  $x'v\beta y$  in  $L$ ,  $\beta \neq 1$  by assumption. Splicing these words, we have  $(xuy', x'v\beta y) \vdash xu\beta y = x\alpha u y \in L$ . Splicing now for each  $i > 0$ , we obtain

$$(x\alpha^i u y, x'v\beta y) \vdash x\alpha^i u\beta y = x\alpha^{i+1} u y \in L.$$

Since  $\beta \neq 1$  implies  $\alpha \neq 1$ , we see  $L$  is infinite.

Now let us assume  $L$  is infinite. We wish to show that one or both of the statements **A** and **B** hold.

Recall that the only rule which is used to generate  $L$  is  $(u, 1; v, 1)$ . So every word  $z$  in  $L$  which is obtained by splicing must be of the form  $xu\bar{y}$ , where  $xu y$  and  $\bar{x}v\bar{y}$  are words in  $L$  are such that  $(xuy, \bar{x}v\bar{y}) \vdash z$ . If, for some  $k$ , both  $P^{k+1} = P^k$  and  $S^{k+1} = S^k$ , there would be a bound on  $|x|$  and  $|\bar{y}|$ , and thus on  $|z|$ . Since  $L$  is infinite, this cannot happen, and at least one of the antecedents of Lemma 8.8 must hold. Either consequence suffices for our purposes. □

## 4 Constants in One Rule Semi-Null Languages

We begin with a lemma.

**Lemma 8.10.** *If  $(u, v; u', v')$  respects  $L$ , then for all  $\alpha, \beta, \alpha'$  and  $\beta'$  in  $A^*$ , the rule  $(\alpha u, v\beta; \alpha' u', v'\beta')$  respects  $L$  as well.*

*Proof.* Obvious application of the splicing rule  $(u, v; u'v')$  to  $x\alpha uv\beta y$  and  $x'\alpha' u'v'\beta' y'$  in  $L$  yields the desired result.  $\square$

Before stating the next theorem, it will be helpful to notice that a special case of Lemma 8.3 occurs if **A** holds for a one rule semi-null language  $L$ . The lemma applies to any language for which statement **A** holds, not just semi-null languages. We state the special case lemma in its most general form:

**Lemma 8.11.** *Let  $L \subseteq A^*$  be such that  $(u, 1; v, 1)$  respects  $L$  and such that **A** holds for  $L$ . Then  $C_{fac}(L)$  is infinite.*

*Proof.* Since  $r = (u, 1; v, 1)$  respects  $L$ , the rule  $(\alpha u, 1; v, 1)$  also respects  $L$  by Lemma 8.10. Since **A** holds, we have  $\alpha u = \gamma v\beta$ , so  $\bar{r} = (\gamma v\beta, 1; v, 1) \in R(L)$ . Since **A** also asserts that  $\gamma v\beta \in \text{Fac } L$ , we have that  $\bar{r}$  is useful in  $L$ . By another application of Lemma 8.10, we have that the rule  $(\gamma v\beta, 1; \gamma v, 1) \in R(L)$ . Since it is clearly also useful, we have  $(\gamma v\beta, 1; \gamma v, 1) \in R_{fac}(L)$ . Finally, Lemma 8.3 implies that there exists an integer  $J$  such that  $\gamma v\beta^j \in C(L)$  for all  $j \geq J$ . Thus  $C_{fac}(L)$  is infinite.  $\square$

Now we shall show that, if **B** holds for  $L$ , and **A** does not, then  $C(L)$  is infinite.

**Proposition 8.12.** *Let  $\sigma = (A, S)$  be a splicing scheme in which  $S = \{(u, 1; v, 1)\}$ . Let  $I$  be finite, and say **B** holds for  $L = \sigma^*(I)$ , but **A** does not hold for  $L$ . If  $L$  is infinite, then  $C_{fac}(L)$  is infinite.*

*Proof.* We shall first prove that  $C(L)$  is infinite by showing that any sufficiently long string in  $A^*$  which ends with  $u$  is a constant of  $L$ . We shall then show that arbitrarily long strings ending in  $u$  appear as factors in  $L$ . We make the claim:

**Claim 8.13.** *Let  $w \in A^+$  have length exceeding the length of any word in  $I$ . Then  $wu \in C(L)$ .*

*Proof.* We make the following induction claim:

**Claim 8.14.** For  $k \geq 0$  and  $w \in A^+$  such that  $|w|$  exceeds the length of any word in  $I$ ,  $pwuq \in \sigma^k(I)$  and  $\tilde{p}wu\tilde{q} \in L$  together imply  $\tilde{p}wuq \in L$ .

*Proof.* Let  $k = 0$ . Note that because  $w$  is longer than any word in  $I$ ,  $pwuq$  is not in  $\sigma^0(I) = I$ . Thus  $pwuq \notin \sigma^0(I)$ , and the statement  $pwuq \in \sigma^0(I)$  and  $\tilde{p}wu\tilde{q} \in L$  together imply  $\tilde{p}wuq \in L$  is vacuously true. We assume the statement is true for all  $i < k$  where  $k > 0$ . Let  $pwuq \in \sigma^k(I) \setminus \sigma^{k-1}(I)$ . Since  $k > 0$ , there exist  $z$  and  $\bar{z}$  such that  $(z, \bar{z}) \vdash pwuq$ . We know  $z, \bar{z} \in \sigma^{k-1}(I)$ , and we can factor these strings as  $z = xuy$ ,  $\bar{z} = \bar{x}v\bar{y}$ , and further,  $xu\bar{y} = pwuq$ .

**case 1:** If  $|xu| \leq |pw|$ , then  $\bar{y} = \gamma uq$  for  $\gamma \in A^*$ . So  $\bar{z} = \bar{x}v\bar{y} = \bar{x}v\gamma uq$ , and  $v(\gamma u) = (v\gamma)u \in \text{Fac}(L)$ , and  $\gamma u \neq 1$ . This violates our assumption that statement **A** does not hold for  $L$ .

**case 2:** If  $|pw| < |xu| \leq |pwu|$ , then  $u = u_1u_2 = u_2u_3$ , where  $w = w_1u_1$  for some  $w_1 \in A^*$ , and  $\bar{y} = u_3q$ . Note  $u_2 = 1$  implies we are in case 1. So  $\bar{x}v\bar{y} = \bar{x}vu_3q$ , and  $\tilde{p}wu\tilde{q} = \tilde{p}w_1uu_3\tilde{q}$ . Splicing, we obtain

$$(\tilde{p}w_1uu_3\tilde{q}, \bar{x}vu_3q) \vdash \tilde{p}w_1uu_3q = \tilde{x}w_1u_1uq = \tilde{p}wuq \in L.$$

**case3:** If  $|xu| > |pwu|$ , then there are  $u_1$  and  $u_3$  such that  $u_1u = uu_3$ ,  $xu = pwuu_3$ , and  $q = u_3\bar{y}$ . Since  $xuy = pwuu_3y \in \sigma^{k-1}(I)$ , the induction assumption says  $\tilde{p}wuu_3y = \tilde{p}wu_1uy \in L$ . Splicing,

$$(\tilde{p}wu_1uy, \bar{x}v\bar{y}) \vdash \tilde{p}wu_1u\bar{y} = \tilde{p}wuu_3\bar{y} = \tilde{p}wuq \in L.$$

Hence, in every case which can occur, the condition  $pwuq \in \sigma^k(I)$  and  $\tilde{p}wu\tilde{q} \in L$  implies that  $\tilde{p}wuq \in L$ . □

Since  $pwuq \in L$  naturally implies that there is an  $n$  such that  $pwuq \in \sigma^n(I)$ , half of our proof that  $wu \in C(L)$  is done. The above argument also proves that  $pwu\tilde{q} \in L$ , so we have in fact that  $wu \in C(L)$ , as desired. □

**Claim 8.15.** There are arbitrarily long factors in  $L$  of the form  $xu$ ,  $x \in A^*$ .

*Proof.* Let  $N = \max\{|y'| \mid xuy' \in L \text{ and } y \text{ contains no factor } u\}$ . Notice then that, given words  $z = xuy$  and  $z' = x'vy'$  in  $L$ , the splicing product  $xuy'$  is such that  $|y'| \leq N$  if  $y'$  contains no factor  $u$ . Since  $L$  is infinite, and all words in  $L \setminus I$  are obtained by splicing via  $(u, 1; v, 1)$ , and must contain  $u$ , we see there must be arbitrarily long factors of the form  $xu$  in  $L$ .  $\square$

Claim 8.13 and Claim 8.15 together imply that  $C_{fac}(L)$  is infinite.  $\square$

**Theorem 8.16.** *Let  $L$  be a one rule semi-null splicing language.  $L$  is infinite if and only if  $C_{fac}(L)$  is infinite.*

*Proof.* Let  $\sigma = (A, S)$  where  $S = \{(u, 1; v, 1)\}$  for  $u, v \in A^+$ . Let  $L = \sigma^*(I)$  be infinite. Theorem 8.9 says either statement **A** or **B** holds for  $L$ .

Say **A** holds for  $L$ . The rule  $(u, 1; v, 1)$  respects  $L$ . Since this is the only rule in  $S$ , and  $L$  is infinite, the rule is clearly useful. Lemma 8.11 says that  $C_{fac}(L)$  is therefore infinite.

Now say **B** holds for  $L$ , but **A** does not. By Proposition 8.12, we have that  $C_{fac}(L)$  is infinite.

In the other direction, if  $C_{fac}(L)$  is infinite, there must be words of arbitrary length in  $L$ .  $\square$

Notice that, while we have infinitely many constant factors of  $L$ , we have not shown that the site  $u$  is constant.

# Chapter 9

## Wet Splicing

The scientific community has recently taken great interest in biomolecular models of computation. In particular, Leonard Adleman's seminal 1994 work [11] inspired a surge of research focused on exploring the possibility of using DNA or other biomolecules to solve mathematical problems which are computationally hard [2], [12]. In light of such developments, it seemed worthwhile to demonstrate the viability of Head's model using experimental methods in the laboratory. It was with this aim that an investigation of the feasibility of performing the splicing operation as originally defined was undertaken.

Results of our study indicate that the splicing operation can be easily performed in a single-step laboratory procedure. In this procedure, restriction digestion and religation of linear double-stranded DNA (dsDNA) take place simultaneously in a single buffer, producing the new strings which are predicted by the model. While the significance of this finding is yet to be realized completely, the importance of experimentally verifying the predictive capability of the  $H$ -system model of splicing is clear.

### 1 Wet Definitions

We use the notation of Head, introduced in [7]. Let  $A$  be a finite set of alphabet symbols.

A splicing system  $\sigma = (A, I, B, C)$  consists of the finite alphabet  $A$ , a finite set  $I$  of initial strings in  $A^*$ , and finite sets  $B$  and  $C$  of triples of the form  $(c, x, d)$  with  $c, x$  and  $d$  in  $A^*$ . Each such triple is called a *pattern*. For the triple  $(c, x, d)$ , the string  $cx d$  is called a *site*, and the string  $x$  is called a *crossing*. For the purposes of this discussion it will be helpful to consider the initial set to be linear strings of dsDNA over an alphabet consisting of the nucleotide pairs formed by hydrogen bonding between the bases adenine, guanine, cytosine, and thymine. These nucleotide pairs can be represented by the set  $\{[A/T], [T/A], [C/G], [G/C]\}$ , which we shorten to  $\{a, t, c, g\}$ . Single stranded DNA are denoted by the capitalized alphabet  $\{A, T, C, G\}$ . It is also helpful to think of a site as the recognition site of a restriction enzyme, and the crossing as the single-stranded overhang which is formed when the enzyme cleaves dsDNA containing the site. Patterns in  $B$  are called *left-handed*, and correspond to recognition sites of restriction enzymes which produce either 5' or blunt end overhangs. Patterns in  $C$  are called *right-handed*, and correspond to the sites of enzymes which produce 3' overhangs.

Strings are spliced according to the following algorithm: If there are strings  $ucx d v$  and  $p e x f q$ , and patterns  $(c, x, d)$  and  $(e, x, f)$  of the same hand, then the strings  $u c x f q$  and  $p e x d v$  are formed by splicing  $ucx d v$  and  $p e x f q$  together. Again, it is helpful to consider patterns of the same handedness which contain the same crossing as corresponding to restriction enzyme sites which produce the same overhangs, *i.e.* overhangs which allow subsequent religation to occur.

The language  $L = L(I, \sigma)$  consists of the strings in  $I$  and all strings in the closure of  $I$  under the operation of splicing, *i.e.* strings which can be generated by splicing in  $L$  are adjoined to  $L$ . Strings in  $L(I, \sigma)$  which cannot be used for splicing are called the *adult* strings in  $L(I, \sigma)$ .

Strings of DNA are measured in length by the number of base pairs which they contain. For example, the dsDNA strand represented by *agatacc* has length 6. If a strand has length exceeding 1000 base pairs, its length is sometimes expressed in kilobase pairs, denoted kbp.

## 2 Motivation

The motivation for this experiment was to produce laboratory verification of Head's theoretical model of splicing systems. The experiment was designed to test the hypothesis that a particular splicing system will converge to a fixed set of strings. The initial set was taken to be two distinct sequences of linear dsDNA, each with dephosphorylated 5' ends, one having a *Bgl*I restriction site and the other having a *Dra*III restriction site. The action of iterated cleavage and religation was predicted to result in a dynamical splicing system which would converge to a particular set of adult strings. The experiment was designed so that this progression would be clearly apparent if the *wet* splicing system behaved as predicted by the theoretical, or *dry* model.

## 3 The Wet Example

The wet splicing system  $\sigma = (A, I, B, C)$  for our experiment was constructed as follows:

$A$  is the set  $\{a, g, c, t\}$  introduced above, and  $I = \{\alpha gccgcaccggc\beta, \gamma caccacgtg\delta\}$ , with  $\alpha, \beta, \gamma$ , and  $\delta$  in  $\{a, c, t, g\}^* = A^*$ . These initial strings are sequences which appear in the genome of the bacteriophage lambda. The strings are assumed to be dephosphorylated on their 5' ends in order to prevent blunt end ligation.

The substrings *gccgcaccggc* and *caccacgtg* appearing in the initial strings are recognition sites for the restriction enzymes *Bgl*I and *Dra*III. Note (*gccg, cac, cggc*) and (*cac, cac, gtg*) are patterns which encode specific actions of these enzymes in the presence of T4 DNA ligase, and together they are the elements in set  $C$ . Set  $B$  is empty in this example. Both restriction sites, when cut, leave 3' *CAC* single-stranded overhangs which allow religation. There are no other recognition sites for these enzymes in either initial string, so the splicing language  $L(I, \sigma)$  for this example is as follows:

$$L(I, \sigma) = I \cup \{\alpha gccgcaccggc\beta, \gamma caccacgtg\delta\}.$$

We were motivated to verify experimentally that the system described above could generate *in vitro* the splicing language predicted by the dry model.

### 3.1 String Specifications

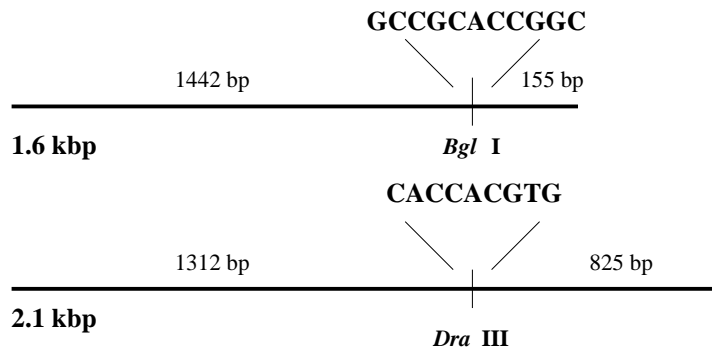
The two substrings of phage lambda DNA which constitute the initial set in the primary example described above were chosen for the following properties: String 1, approximately 1.6 kbp long, was chosen because it contains exactly one *Bgl*I site, about 1.45 kbp from one end. String 2, which we shall refer to as the 2.1 kbp string, contains exactly one *Dra*III site, approximately 1.3 kbp from one end. The 2.1 kbp string contains no *Bgl*I restriction site and the 1.6 kbp string contains no *Dra*III site.

Both sites, when cut by the appropriate enzyme, leave 3' overhangs of the sequence CAC which allow subsequent ligation of fragments to occur. Further, if two fragments originating from the two different strings are ligated together, neither restriction site is present in the product string, and no subsequent cleavage of these molecules is possible, *i.e.* these are the adult strings in the splicing language. Two such adult strings can be formed in this system. No blunt end ligation could occur at all, since the initial molecules were dephosphorylated on their 5' ends. In theory therefore, repeated cleavage and ligation steps should reduce the quantity of the original 2.1 and 1.6 kbp molecules, and increase the quantity of the adult molecules, which are approximately 0.98 and 2.7 kbp in length. Such strands would be readily detectable and easily differentiated by agarose gel electrophoresis and subsequent UV analysis [1]. The initial molecules and the adult product molecules expected after cleavage and religation are shown in Figure 1.

## 4 The Experimental Results

A preliminary run of the simultaneous restriction digestion and ligation experiment was performed, and the results of this reaction are shown in Figure 2. A time sequence

**INITIAL MOLECULES**



**ADULT MOLECULES**

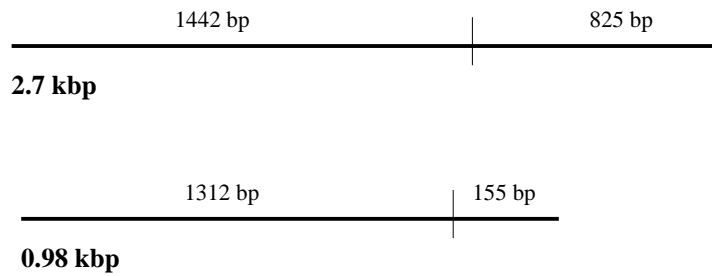


Figure 9.1: The initial molecules and the adult molecules which are the expected products in the primary example.

study of the primary simultaneous restriction digestion and ligation reaction, called Reaction T, was then performed. The results of the experiment appear in Figure 3.

We predicted that the dynamical behavior of this wet splicing system would be a time-related decrease in the initial strings, and a corresponding increase in the adult strings. Intermediate fragments having 3' CAC overhangs were expected as well.

The restriction enzymes chosen apparently work with good efficiency at the concentrations used and in the T4 DNA ligase buffer. The *BglI* and *DraIII* began to cleave the initial strands of DNA in T4 DNA ligase buffer within 5 minutes after Reaction T was begun, as seen in Lane 3 of the gel in Figure 3. Digestion of the 2.1 and 1.6 kbp fragments apparently continued throughout the reaction, although the majority of both initial fragments were cleaved within an hour at room temperature, as apparent from Lane 5. The resulting DNA pieces with overhangs appeared within 5 minutes, as seen in Lane 3. The ligase worked well at 22°C, as expected, and the fragments with overhangs began to religate immediately. The 5 minute aliquot in Lane 3 of the gel appearing in Figure 3 showed some apparent 2.7 and 0.98 kbp adult products of religation. Within 60 minutes the two adult DNA strands of lengths 2.7 and 0.98 kbp were clearly visible (Lane 5). The complete time sequence indicates a progressive decrease in the quantities of the initial 2.1 and 1.6 kbp strings, and a progressive increase in the quantities of the adult strings, as expected.

The quantities of the initial molecules decreased over time, until virtually all initial molecules disappeared. The intermediate fragments with overhangs first increased in apparent quantity as the initial molecules are cleaved by the restriction enzymes during the first minutes of the reaction, and then the quantities of these intermediate fragments decreased as they are used up during the religation process, forming adult molecules which cannot be recleaved. The adult molecules showed a steady increase in quantity throughout the reaction, and were apparently not involved in further interactions with other molecules or enzymes.

## 5 Conclusion

This splicing system behaved as predicted by the dry mathematical model proposed by Head. The initial set consisting of the two dsDNA strings was dynamically transformed by the process of cleavage and religation into the predicted set of adult strings of dsDNA. The adult strands showed no evidence of further cleavage, and the initial strings gradually disappeared from the reaction mixture over time. The intermediate fragments were visible as quickly as five minutes after the reaction was initiated. Their concentrations initially appeared to steadily increase until they reached a maximum concentration in the reaction, after which time the quantities of the intermediate fragments in the reaction mixture appeared to monotonically diminish. The apparent quantities of the adult strings steadily increased throughout the duration of the reaction.

In conclusion, we have initial results which seem promising. Further research must be done in the areas suggested above, as well as in areas which will broaden the scope of the investigation of the splicing system model.

Figure 9.2: Preliminary Reaction: Simultaneous restriction digestion and ligation of 1.6 kbp and 2.1 kbp dsDNA fragments. Lanes 1 and 4 are the standard molecular weight markers (phage lambda DNA cut with *HindIII* and *BstEII*, respectively). Lane 2 contains the initial dsDNA strings of lengths 2.1 and 1.6 kbp. Lane 3 contains the products after overnight incubation at room temperature. Note that the expected adult strings of lengths 2.7 and .98 kbp appear. The faint bands appearing in Lane 3 are unreligated fragments from restriction digestion of the initial strings.

Figure 9.3: Reaction T: Time sequence study of simultaneous restriction digestion and ligation of 1.6 kbp and 2.1 kbp dsDNA fragments. Lanes 1 and 7 are standard molecular weight markers (phage lambda DNA cut with *Hind*III and *Bst*EII, respectively). Lane 2 contains the initial dsDNA strings of lengths 2.1 and 1.6 kbp. Lanes 3, 4 and 5 contain the products after incubation at room temperature for 5, 15 and 60 minutes, respectively. Lane 6 contains the product strings after overnight incubation at 16°C. Note that the bands containing the expected adult strings of lengths 2.7 and .98 kbp appear increasingly bright as time passes, while the bands of both the original dsDNA and the unreligated fragments grow fainter with time.

# Bibliography

- [1] A.Ausubel, R.Brent, R.E. Kingston, D.D. Moore, J.G. Seidman and J.A. Smith, and K.Struhl, *Current protocols in molecular biology*, Greene Publishing Associates and Wiley-Interscience, 1994.
- [2] Dan Boneh, Christopher Dunworth, and Richard J. Lipton, *Breaking DES using a molecular computer*, Tech. Report CS-TR-489-95, Princeton University, May 1995.
- [3] K. Culik II and T. Harju, *The regularity of splicing systems and dna*, LNCS, vol. 372, ICALP '89, 1989, pp. 222–233.
- [4] ———, *Splicing semigroups of dominoes and DNA*, Discrete Appl. Math. **31** (1991), 261–277.
- [5] A. Deluca and A. Restivo, *A characterization of strictly locally testible languages and its application to subsemigroups of a free semigroup*, Inform. and Control **44** (1980), 300–319.
- [6] R. W. Gatterdam, *Splicing systems and regularity*, Internat. J. Comput. Math. **31** (1989), 63–67.
- [7] T. Head, *Formal language theory and DNA: an analysis of the generative capacity of specific recombinant behaviors.*, Bull. Math. Biol. **49** (1987), no. 6, 737–759.
- [8] ———, *Splicing languages generated with one sided context*, Computing With Bio-molecules—Theory and Experiments (G. Paun, ed.), Springer-Verlag, Singapore, 1998, pp. 269–282.

- [9] ———, *Splicing representations of strictly locally testible languages*, Discrete Applied Mathematics **87** (1998), 139–147.
- [10] J. Hopcroft and J. Ullman, *Introduction to automata theory, languages, and computation*, Addison-Wesley, 1979.
- [11] L. Adleman, *Molecular computation of solutions to combinatorial problems*, Science **266** (1994), 1021–1024.
- [12] R. J. Lipton, *DNA solution of hard computational problems*, Science **268** (1995), 542–545.
- [13] A. Mateescu, G. Păun, G. Rozenberg, and A. Salomaa, *Simple splicing systems*, Discrete Applied Mathematics (1996).
- [14] R. McNaughton and S. Papert, *Counter-free automata*, MIT Press, 1971.
- [15] G. Păun, *On the power of the splicing operation*, International Journal of Computer Mathematics **59** (1995), 27–35.
- [16] ———, *On the splicing operation*, Discrete Appl. Math. **70** (1996), no. 1, 57–79.
- [17] G. Păun (ed.), *Computing with biomolecules*, Springer Verlag, Berlin, Heidelberg, New York, 1998.
- [18] G. Păun, G. Rozenberg, and A. Salomaa, *Computing by splicing*, Theoretical Computer Science **168** (1996), no. 2, 321–336.
- [19] J.E. Pin, *Varieties of formal languages*, Plenum Publishing Co., 1986.
- [20] D. Pixton, *Simultaneous pumping lemma*, From conversations in 1999.
- [21] ———, *Splicing in abstract families of languages*, to appear.
- [22] ———, *Regularity of splicing languages*, Discrete Appl. Math. **69** (1996), no. 1–2, 99–122.
- [23] M.P. Schützenberger, *Sur certaines opérations de fermeture dans les langages rationnels*, Symposia Math. **15** (1975), 245–253.