Prof. Dr. Jürgen Dassow
Otto-von-Guericke-Universität Magdeburg
Fakultät für Informatik

F O R M A L L A N G U A G E S

A N D

B I O L O G I C A L P R O C E S S E S

Vorlesungsmanuskript

Magdeburg, April - July 2008

Introduction

In the end of the fifties as N. Chomsky has introduced the well-known classes of regular, context-free and context-sensitive languages the aim was to model the syntax of natural languages. Based on the Backus-Naur form for the description of the syntax of programming languages, in the beginning of the sixties S. Ginsburg and H.G. Rice noticed that the grammars introduced by Chomsky can be used for programming languages, too. Since that time until at least the middle of the seventies most investigations to formal languages followed this approach. The central feature of such grammars is a sequential process of rewriting of subwords.

On the other hand one has to mention that already since the fifties there exist some devices nearly related to formal languages which were motivated and/or applied to biological phenomena. The well-known Kleene Theorem on the description of regular languages by means of algebraic operations was discovered by S.C. Kleene as he represented the events in nerve nets. Furthermore, it was known that cellular automata are able to a self-replicating behaviour known from biological organisms or colonies of organisms. But in both cases, in order to model the biological processes finite automata or collections of finite automata have been used.

Since the seventies the situation changed completely. Motivated by biological processes new types of grammars have been introduced and their investigation dominated in a certain sense the development of the theory of formal languages.

In 1968 the first approach was initiated by A. Lindenmayer (see [16]. Cell divisions, changes of states of the cells, death of cells etc. were modelled by production as one uses in Chomsky grammars. However, the rewriting process by application of rules is a parallel one because cell divisions, changes of cell states etc. proceed in parallel. The large interest in these Lindenmayer systems originated from the biological motivation as well as by the interest in a comparison between sequential and parallel processes in computer science. The monograph [13] presents a summary of the state of the theory of developmental systems and languages in 1975 and considers intensively motivation from and application to bilogy, whereas the monograph [27] emphasizes the mathematical theory of such systems. Further summaries and material can be found in [26], [17], [28], [29], [15]. In [25] the authors use Lindenmayer systems to generate graphical representations of plants.

Although DNA sequences are twisted strands (in a 3-dimensional space) it is very natural to model them by (linear) strings/words. Mutations of DNA sequences, genes, chromosomes etc. caused by deletions, insertions, splicings, inversions etc. can be described by operations on words. Iterated applications of these operations model the evolution of molecules. Thus we have sequential process, again, however, the basic step is not a rewriting. After the first investigations in this direction by T. Head (see [11]) in the last

decade a lot of papers appeared studying the behaviour of formal languages under these operations. Moreover, one has to mention that these considerations are nearly related to some aspects of molecular computing (see [1], [18]). The book [23] is the first monograph on this topic, summaries are contained in [2], [12], [24], [7].

An approach – called membrane systems – to describe the behaviour of a single cell was startet by Gh. Păun in the paper [21]. A cell is considered as an object with membranes which define substructures of the cell, e.g. the kernel of the cell. Changes of the objects in the different regions of the cell are described by rules associated with the regions. However, the rules are not applied to words as in the two types of grammars mentioned above, the rules are applied to multisets since the objects in a region form a multiset. The books [22] and [2] summarize parts of the theory developed for these grammatical systems.

We mention that these three new types of grammars/languages are natural by their motivation from biology as well as by the fact that they allow nice characterizations of well-known classes of formal languages.

In this lecture we shall emphasize Lindenmayer systems, languages and systems using operations as splicing and membrane systems. We shall omit grammars with valuations (see [5]), eco-grammar systems (see [4]) and other language generating devices modelling aspects of biology.

Throughout this lecture we assume that the students/reader is familiar with the basic concepts of the theory of formal languages as usually presented in basic courses on Theoretical Computer Science and with some facts of mathematics (especially linear algebra, theory of difference equations, combinatorial formulae, etc). The notation, some definitions and results are summarized in the first chapter.

Jürgen Dassow April - July 2008

Contents

3.3 Splicing as an operation

3.3.1 Non-iterated splicing

In this section we formalize the operation splicing presented in Figure 3.9 such that it is an operation applicable to words and languages.

Definition 3.1 *A splicing scheme is a pair (V, R), where*
– *V is an alphabet and*
– *R is a subset of $V^*\#V^*\$V^*\#V^*$.*

The elements of R are called *splicing rules*. Any splicing rule $r_1\#r_2\$r_3\#r_4$ identifies four words r_1, r_2, r_3 and r_4. As one can see from Figure 3.9 taking into consideration the upper strand only, the essential part to get a splicing is the existence of two subwords r_1r_2 and r_3r_4 modelling the recognition sites such that a splitting can be done between r_1 and r_2 as well as between r_3 and r_4.

Obviously, this can be obtained by an quadruple (r_1, r_2, r_3, r_4), too. However, in the sequel we shall consider the sets of splicing rules as languages, and thus we prefer to present them as words over $V \cup \{\#, \$\}$.

Definition 3.2 *i) We say that $w \in V^*$ and $z \in V^*$ are obtained from $u \in V^*$ and $v \in V^*$ by the splicing rule $r = r_1\#r_2\$r_3\#r_4$, written as $(u, v) \models_r (w, z)$, if the following conditions hold:*
– *$u = u_1r_1r_2u_2$ and $v = v_1r_3r_4v_2$,*
– *$w = u_1r_1r_4v_2$ and $z = v_1r_3r_2u_2$.*

This definition describes the situation given in Figure 3.9, where we only consider the upper strand, again.

We now give a slight modification of this formalization by emphasizing the getting the new word w and omitting the word z which is obtained, too. As we shall see below, this can be done because z will have some features, we are not interested in, such that we do not take it into consideration.

Definition 3.3 *i) For two words $u \in V^*$ and $v \in V^*$ and a splicing rule $r = r_1\#r_2\$r_3\#r_4$, we define the word w obtained from u, $v$and r by a simple splicing, written as $(u, v) \vdash_r w$, by the following conditions:*
– *$u = u_1r_1r_2u_2$ and $v = v_1r_3r_4v_2$,*
– *$w = u_1r_1r_4v_2$*
ii) For a language L over V and a splicing scheme (V, R), we set

$$spl(L, R) = \{w \mid (u, v) \vdash_r w,\ u \in L,\ v \in L,\ r \in R\}.$$

For two language families $\mathcal{L}_1$ and $\mathcal{L}_2$, we set,

$$\begin{aligned} spl(\mathcal{L}_1, \mathcal{L}_2) \quad = \quad & \{L' \mid L' = spl(L, R) \text{ for some } L \in \mathcal{L}_1 \\ & \text{and some splicing scheme } (V, R) \text{ with } R \in \mathcal{L}_2\}. \end{aligned}$$

Example 3.4 We consider the language $L = \{a^n b^n \mid n \geq 0\}$ and the splicing scheme (V, R) with $V = \{a, b\}$ and $R = \{a\#b\$a\#b\}$. First we note that the only rule r of R is only applicable to words $a^n b^n$ with $n \geq 1$. Let $u = a^n b^n$ and $v = a^m b^m$ be two arbitrary words from L with $m, n \geq 1$. Then we obtain

$$\begin{aligned}(a^n b^n, a^m b^m) &= (a^{n-1}abb^{n-1}, a^{m-1}abb^{m-1}) \\ &\vdash_r a^n b^m \,.\end{aligned}$$

Since n and m are arbitrary positive integers, we get

$$spl(L, R) = \{a^n b^m \mid n, m \geq 1\} \,.$$

Example 3.5 For

$$L = \{c\}\{a, b\}^+\{c'\} \text{ and } R = \{ca^n b^n \#c'\$c'\# \mid n \geq 1\}$$

we obtain

$$spl(L, R) = \{c\}\{a^n b^n \mid n \geq 1\}$$

since the only simple splicing is $(ca^n b^n c', cvc') \vdash_r ca^n b^n$ applying the rule $ca^n b^n \#c'\$c'\#$.

(We note that the other word z which is obtained by this splicing is $z = cvc'c'$. It contains two times the letter c' such that it is not of interest.)

Example 3.6 Let L and L' be two arbitrary languages over V. Further, let $(V \cup \{c\}), R)$ be a splicing scheme with

$$R = \{\#xc\$c\# \mid x \in L'\} \,.$$

Then we get

$$spl(L\{c\}, R) = \{w \mid wx \in L \text{ for some } x \in L'\}$$

because splicing is only possible if $u = wxc$ and $v = w'c$ for some words $wx, w' \in L$ and $x \in L'$.

(We note that the other word z obtained by splicing is $z = w'cxc$ which we are not interested in since it contains two times the letter c.)

Example 3.7 We want to show that

$$\{a^n b^n \mid n \geq 1\} \notin spl(\mathcal{L}(REG), \mathcal{L}(RE)) \,,$$

or more precisely, that $L = \{a^n b^n \mid n \geq 1\}$ cannot be obtained from a regular set by (arbitrary) splicings. Note that, by Example 3.5, we can get $\{c\}L$ from a regular set by splicing with a context-free set.

Assume that there are a regular language K and a splicing scheme (V, R) such that $spl(K, R) = L$. Let $\mathcal{A} = (x, Z, z_0, F, \delta)$ be a finite deterministic automaton with $T(\mathcal{A}) = K$. Let m be the cardinality of Z.

By definition, there are words $u = u_1 r_1 r_2 u_2$ and $v = v_1 r_3 r_4 v_2$ and a splicing rule $r = r_1 \# r_2 \$ r_3 \# r_4 \in R$ such that

$$(u, v) \vdash_r = u_1 r_1 r_4 v_2 = a^{m+1} b^{m+1} \,.$$

Obviously, $u_1r_1 = a^{m+1}z$ or $r_4v_2 = z'b^{m+1}$ for certain z and z'. We now discuss the former case; the latter one can be handled analogously. By the pumping lemma for regular languages (see Theorem 1.12),

$$u' = a^{m+1+t}zr_2u_2 = a^tu_1r_1r_2u_2 \in K\,.$$

Thus

$$(u', v) = (a^tu_1r_1r_2u_2, v_1r_3r_4v_2) \vdash a^tu_1r_1r_4v_2 = a^{t+m+1}b^{m+1}\,.$$

Therefore $a^{t+m+1}b^{m+1} \in spl(K, R)$ in contrast to $a^{t+m+1}b^{m+1} \notin L$.

In the following theorem we determine the language families $spl(\mathcal{L}_1, \mathcal{L}_2)$ or upper and lower bounds for these families where $\mathcal{L}_1$ and $\mathcal{L}_2$ vary over all language families from the Chomsky hierarchy.

Theorem 3.8 *The table of Figure 3.11 holds, where at the intersection of the row marked by X and the column marked by Y we give Z if $\mathcal{L}(Z) = spl(\mathcal{L}(X), \mathcal{L}(Y))$ and Z_1/Z_2 if $\mathcal{L}(Z_1) \subset spl(\mathcal{L}(X), \mathcal{L}(Y)) \subset \mathcal{L}(Z_2)$.*

	FIN	REG	CF	CS	RE
FIN	FIN	FIN	FIN	FIN	FIN
REG	REG	REG	REG/CF	REG/RE	REG/RE
CF	CF	CF	RE	RE	RE
CS	RE	RE	RE	RE	RE
RE	RE	RE	RE	RE	RE

Figure 3.11: Relations for the families $spl(\mathcal{L}_1, \mathcal{L}_2)$

Theorem 3.8 can be considered as a result on the power of the splicing operation. We see an indifferent picture. On one hand side its power is large since context-free splicing rules applied to context-free languages give already all recursively enumerable languages. On the other side, if we start with regular languages, then we cannot obtain such easy languages as $\{a^nb^n \mid n \geq 1\}$ (see Example 3.7) and by regular splicing rules we have almost no change of the family.

Before we give the proof of Theorem 3.8 we present some lemmas which will be used in the proof and are of own interest since they can be applied to other language families, too. The first lemma follows directly from the definitions.

Lemma 3.9 *For any language families $\mathcal{L}_1, \mathcal{L}_2, \mathcal{L}_1', \mathcal{L}_2'$ with $\mathcal{L}_1 \subseteq \mathcal{L}_1'$ and $\mathcal{L}_2 \subseteq \mathcal{L}_2'$, we have $spl(\mathcal{L}_1, \mathcal{L}_2) \subseteq spl(\mathcal{L}_1', \mathcal{L}_2')$.* □

Lemma 3.10 *If $\mathcal{L}_1$ is closed under concatenation with symbols, then $\mathcal{L}_1 \subseteq spl(\mathcal{L}_1, \mathcal{L}_2)$ for all language families $\mathcal{L}_2$.*

Proof. Let $L \subseteq V^*$ be an arbitrary language in $\mathcal{L}_1$ and c a symbol not in V. We set $L' = L\{c\}$ and consider the splicing system $(V \cup \{c\}, R)$ with the single element set $R = \{\#c\$c\#\}$. Then we obtain $spl(L', R) = L$ because the only possible simple splicings are given by $(uc, vc) \vdash u$ where u and v are arbitrary elements of L. □

Lemma 3.11 *If $\mathcal{L}$ is closed under concatenation, homomorphism, inverse homomorphisms and intersections with regular sets, then $spl(\mathcal{L}, \mathcal{L}(REG)) \subseteq \mathcal{L}$.*

Proof. Let L be an arbitrary language of $\mathcal{L}$. Then we set $L_1 = L\{\$\}L$. Let

$$h_1 : (V \cup \{\$, \#\})^* \to V \cup \{\$\}$$

be the homomorphism defined by

$$h_1(a) = a \text{ for } a \in V,\ h_1(\$) = \$,\ h_1(\#) = \lambda\,.$$

Then $h_1^{-1}(L_1)$ consists of all words which can be obtained from words of L_1 by putting some occurrences of # between some letters of $V \cup \{\$\}$. Thus

$$L_2 = h_1^{-1}(L_1) \cap V^*\{\#\}V^*\{\$\}V^*\{\#\}V^* = \{w_1\#w_2\$w_3\#w_4 \mid w_1w_2, w_3w_4 \in L\}\,.$$

Let

$$V' = \{a' \mid a \in V\},\ V'' = \{a'' \mid a \in V\},\ V''' = \{a''' \mid a \in V\}\,.$$

Moreover, for a word $w = a_1a_2 \ldots a_n$ with $a_i \in V$ for $1 \leq i \leq n$, we set $w' = a_1'a_2' \ldots a_n'$. Furthermore, we consider the homomorphism

$$h_2 : (V \cup V' \cup \{\#, \$\})^* \to (V \cup \{\#, \$\})^*$$

defined by

$$h_2(a) = a \text{ for } a \in V,\ h_2(\$) = \$,\ h_2(\#) = \#,\ h_2(a') = a \text{ for } a' \in V'$$

and the regular set

$$K = V^*\{\#\}(V')^*\{\$\}(V')^*\{\#\}V^*\,.$$

Then

$$L_3 = h_2^{-1}(L_2) \cap K = \{w_1\#w_2'\$w_3'\#w_4 \mid w_1w_2 \in L,\ w_3w_4 \in L\}$$

is a language in $\mathcal{L}$ by the closure properties of $\mathcal{L}$.

Now let (V, R) be a splicing scheme with a regular set of splicing rules. Using the homomorphisms

$$\begin{aligned} h_3 &: (V \cup V' \cup V'' \cup V''' \cup \{\#, \$\})^* \to (V \cup \{\#, \$\})^* \\ h_4 &: (V \cup V' \cup V'' \cup V''' \cup \{\#, \$\})^* \to (V \cup V' \cup \{\#, \$\})^* \end{aligned}$$

defined by

$$\begin{aligned} &h_3(a) = a \text{ for } a \in V,\ h_3(\$) = \$,\ h_3(\#) = \#,\ h_3(a') = \lambda \text{ for } a \in V, \\ &\quad h_3(a'') = a \text{ for } a \in V, h_3(a''') = \lambda \text{ for } a \in V, \\ &h_4(a) = a \text{ for } a \in V,\ h_4(\$) = \$,\ h_4(\#) = \#,\ h_4(a') = a \text{ for } a \in V, \\ &h_4(a'') = a' \text{ for } a \in V, h_4(a''') = a' \text{ for } a \in V \end{aligned}$$

and the regular set

$$K' = (V')^*V^*\{\#\}(V'')^*(V''')^*\{\$\}(V''')^*(V'')^*\{\#\}V^*(V')^*\,.$$

we get

$$L_4 = h_4(h_3^{-1}(R) \cap K') = \{u_1 r_1 \# r_2' u_2' \$ v_1' r_3' \# r_4 v_2 \mid u_1, u_2, v_1, v_2 \in V^*, r_1 \# r_2 \$ r_3 \# r_4 \in R\}\,.$$

L_3 is regular by the closure properties of $\mathcal{L}(REG)$.

Now we define the homomorphism

$$h_5 : (V \cup V' \cup \{\#, \$\})^* \to (V \cup \{\#, \$\})^*$$

defined by

$$h_5(a) = a \text{ for } a \in V,\ h_5(\$) = \lambda,\ h_5(\#) = \lambda,\ h_5(a') = \lambda \text{ for } a \in V\,.$$

Then $h_5(L_3 \cap L_4) \in \mathcal{L}$ consists of all words of the form $u_1 r_1 r_4 v_2$ and thus $h_5(L_3 \cap L_4) = spl(L, R) \in \mathcal{L}$. Thus $spl(\mathcal{L}, \mathcal{L}(REG)) \subseteq \mathcal{L}$. □

Lemma 3.12 *If $\mathcal{L}$ is closed under homomorphism, inverse homomorphisms and intersections with regular sets, then $spl(\mathcal{L}(REG), \mathcal{L}) \subseteq \mathcal{L}$.*

Proof. From a regular set L we construct as above the language

$$L' = \{w_1 \# w_2' \$ w_3' \# w_4 \mid w_1 w_2 \in L, w_3 w_4 \in L\}$$

and from a set $R \in \mathcal{L}$ of splicing rules we construct the set

$$R' = \{u_1 r_1 \# r_2' u_2' \$ v_1' r_3' \# r_4 v_2 \mid u_1, u_2, v_1, v_2 \in V^*, r_1 \# r_2 \$ r_3 \# r_4 \in R\}$$

as in the proof of Lemma 3.11 and from these two sets $spl(L, R)$ which then belongs to $\mathcal{L}$. □

Proof of Theorem 3.8 We prove the statements row by row from left to right.

If L is a finite language, then we can only apply to words of L such rules $r_1 \# r_2 \$ r_3 \# r_4$ of R where $r_1 r_2$ and $r_3 r_4$ are subwords of words in L. Hence we have only to consider a finite set of splicing rules. By application of a finite set of splicing rules to a finite set of words we only obtain a finite set. Thus $spl(\mathcal{L}(FIN), \mathcal{L}(RE)) \subseteq \mathcal{L}(FIN)$.

If we combine this result with that of Lemmas 3.10 and 3.9, for all families $X \in \{FIN, REG, CF, CS, RE\}$, we get

$$\begin{aligned} \mathcal{L}(FIN) &\subseteq spl(\mathcal{L}(FIN), \mathcal{L}(FIN)) \subseteq spl(\mathcal{L}(FIN), \mathcal{L}(X)) \\ &\subseteq spl(\mathcal{L}(FIN), \mathcal{L}(RE)) \subseteq \mathcal{L}(FIN) \end{aligned}$$

and thus

$$spl(\mathcal{L}(FIN), \mathcal{L}(X)) = \mathcal{L}(FIN)\,.$$

By Lemmas 3.10, 3.9 and 3.12, we get

$$\mathcal{L}(REG) \subseteq spl(\mathcal{L}(REG), \mathcal{L}(FIN)) \subseteq spl(\mathcal{L}(REG), \mathcal{L}(REG)) \subseteq \mathcal{L}(REG)$$

which proves the first two statements of the row belonging to REG.

By Lemma 3.9, we have $\mathcal{L}(REG) \subseteq spl(\mathcal{L}(REG), \mathcal{L}(X))$ for $X \in \{CF, CS, RE\}$. Moreover, this inclusion is strict by Example 3.5 because $\{c\}\{a^n b^n \mid n \geq 1\}$ is not a regular language.

By the closure properties of $\mathcal{L}(CF)$ and $\mathcal{L}(RE)$ (see Chapter 1) and Lemma 3.12,

$$spl(\mathcal{L}(REG), \mathcal{L}(CF)) \subseteq \mathcal{L}(CF) \text{ and } spl(\mathcal{L}(REG), \mathcal{L}(RE)) \subseteq \mathcal{L}(RE)\,.$$

Moreover,

$$spl(\mathcal{L}(REG), \mathcal{L}(CS)) \subseteq spl(\mathcal{L}(REG), \mathcal{L}(RE)) \subseteq \mathcal{L}(RE)$$

by Lemma 3.9. These inclusions are strict by Example 3.7.

$\mathcal{L}(CF) = spl(\mathcal{L}(FIN), \mathcal{L}(CF)) = spl(\mathcal{L}(REG), \mathcal{L}(CF))$ can be shown as above for regular languages.

By Lemma 1.6, for any recursively enumerable language L, there are context-free languages L_1 and L_2 such that

$$L = \{x \mid xy \in L_1,\ y \in L_2 \text{ for some } x, y\}\,.$$

As in Example 3.6 we can prove that $L \in spl(\mathcal{L}(CF), \mathcal{L}(CF))$. Therefore we obtain $\mathcal{L}(RE) \subseteq spl(\mathcal{L}(CF), \mathcal{L}(CF))$.

$spl(\mathcal{L}(RE), \mathcal{L}(RE)) \subseteq \mathcal{L}(RE)$ can be proved by constructing a Turing machine which accepts $spl(L, R)$ for given (recursively enumerable) languages L and R. (We omit a detailed construction. Informally, the machine works as follows: The given word w is nondeterministically divided into four subwords $w = u_1 r_1 r_4 v_2$; then we choose nondeterministically words r_2, u_2, v_1, r_3 and check whether $r_1 \# r_2 \$ r_3 \# r_4 \in R$, $u_1 r_1 r_2 u_2 \in L$ and $v_1 r_3 r_4 v_2 \in L$.)

For $X \in \{CF, CS, RE\}$, combining these two inclusions with Lemma 3.9 gives

$$\begin{aligned} \mathcal{L}(RE) &\subseteq spl(\mathcal{L}(CF), \mathcal{L}(CF)) \subseteq spl(\mathcal{L}(CF), \mathcal{L}(X)) \\ &\subseteq spl(\mathcal{L}(CF), \mathcal{L}(RE)) \subseteq spl(\mathcal{L}(RE), \mathcal{L}(RE)) \\ &\subseteq \mathcal{L}(RE) \end{aligned}$$

which implies

$$spl(\mathcal{L}(CF), \mathcal{L}(X)) = \mathcal{L}(RE)\,.$$

By Lemma 1.7, for any recursively enumerable language L, there is a context-sensitive language L' such that $L' \subseteq L\{c_1 c_2^n c_3 \mid n \geq 0\}$ and for any $w \in L$ there is an n such that $w c_1 c_2^n c_3 \in L'$. It is easy to see that $spl(L', \{\# c_1 \$ c_3 \#\}) = L$. Thus $\mathcal{L}(RE) \subseteq spl(\mathcal{L}(CS), \mathcal{L}(FIN))$. As in the case of context-free languages we can now prove that

$$\mathcal{L}(RE) = spl(\mathcal{L}(CS), \mathcal{L}(X)) = spl(\mathcal{L}(RE), \mathcal{L}(X))$$

for $X \in \{FIN, REG, CF, CS, RE\}$. □

3.3.2 Iterated splicing

Simple splicing is an operation which generates one word from two words. This situation is similar to a derivation step in a grammar or L system where we generate one word from one word. However, in the theory of languages we consider the reflexive and transitive closure of the derivation relation. This corresponds to an iterated performing of derivation steps. We now present the analogous concept for the splicing operation.

Definition 3.13 *A splicing system is a triple $G = (V, R, A)$ where*
– V is an alphabet,
– R is a subset of $V^*\#V^*\$V^*\#V^*$ and
– A is a subset of V^*.

Definition 3.14 *The language $L(G)$ generated by a splicing system G is defined by the following settings:*

$$\begin{aligned} spl^0(G) &= A, \\ spl^{i+1}(G) &= spl(spl^i(G), R)) \cup spl^i(G) \text{ for } i \geq 0, \\ L(G) &= \bigcup_{i \geq 0} spl^i(G). \end{aligned}$$

The essential difference to language generation by grammars and L systems is that we start with a set of words instead of a single word. Moreover, this start language can be infinite.

Furthermore, we mention that splicing systems have a biological meaning. Evolution is based on changes in the DNA strands. Such changes can be originated by splicings. Thus the application of a splicing rule can be considered as a step in the evolution. Therefore the elements generated by a splicing system can be considered as those DNAs which can be obtained during an evolution from elements of a given set A by evolution steps modelled by the splicing rules in R.

Example 3.15 We consider the splicing system

$$G = (\{a, b\}, \{a\#b\$a\#b\}, \{a^n b^n \mid n \geq 1\}).$$

By Example 3.4 we have

$$\begin{aligned} spl^0(G) &= \{a^n b^n \mid n \geq 1\}, \\ spl^1(G) &= spl(\{a^n b^n \mid n \geq 1\}, \{a\#b\$a\#b\}) \cup \{a^n b^n \mid n \geq 1\} \\ &= \{a^r b^s \mid r, s \geq 1\} \cup \{a^n b^n \mid n \geq 1\} \\ &= \{a^r b^s \mid r, s \geq 1\}, \\ spl^2(G) &= spl(\{a^r b^s \mid r, s \geq 1\}, \{a\#b\$a\#b\}) \cup \{a^r b^s \mid r, s \geq 1\} \\ &= \{a^r b^s \mid r, s \geq 1\} \cup \{a^r b^s \mid r, s \geq 1\} \\ &= \{a^r b^s \mid r, s \geq 1\}. \end{aligned}$$

Thus we get $spl^2(G) = spl^1(G)$. This implies by induction

$$\begin{aligned} spl^m(G) &= spl(spl^{m-1}(G), \{a\#b\$a\#b\}) \cup spl^{m-1}(G) \\ &= spl(spl^1(G), \{a\#b\$a\#b\}) \cup spl^1(G) \\ &= spl^2(G) \\ &= spl^1(G). \end{aligned}$$

Therefore

$$L(G) = \bigcup_{i \geq 0} spl^i(G) = \{a^r b^s \mid r, s \geq 1\},$$

i.e., that the iteration does not increase the power (see Example 3.4).

The situation completely changes if we consider the splicing system

$$G' = (\{a, b\}, \{a\#b\$a\#b\}, \{(a^n b^n)^2 \mid n \geq 1\}).$$

We obtain

$$\begin{aligned} spl^1(G') &= \{a^n b^m \mid n, m \geq 1\} \cup \{a^n b^n a^n b^m \mid n, m \geq 1\} \\ &\quad \cup \{a^n b^m a^m b^m \mid n, m \geq 1\} \cup \{a^n b^n a^n b^m a^m b^m \mid n, m \geq 1\}. \end{aligned}$$

By

$$(a^n b^m a^m b^m, a^r b^r a^r b^r) \vdash a^n b^m a^m b^r$$

we have $a^n b^m a^m b^r \in spl^2(G)$, but $a^n b^m a^m b^r \notin spl^1(G)$.

We shall show that

$$L(G') = \{\{a\}^+ \{b^n a^n \mid n \geq 1\}^* \{b\}^+.$$

.

We shall prove by induction that $spl^m(G')$ contains only words of this form. Above we have seen that this statement holds for $spl^1(G')$. The splicing of two such words

$$a^r b^{n_1} a^{n_1} b^{n_2} a^{n_2} \ldots b^{n_s} a^{n_s} b^t \text{ and } a^p b^{m_1} a^{m_1} b^{m_2} a^{m_2} \ldots b^{m_k} a^{m_k} b^q$$

results in

$$a^r b^{n_1} a^{n_1} b^{n_2} a^{n_2} \ldots b^{n_f} a^{n_f} b^{m_g} a^{m_g} b^{m_{g+1}} a^{m_{g+1}} \ldots b^{m_k} a^{m_k} b^q,$$

which of the same form, again. Thus, if $spl^m(G')$ only contains such words, then this also holds for $spl^{m+1}(G')$.

It remains to prove that all such words can be obtained. We prove this by induction on the number of changes from a to b. If we only have one change, then we are interested in the words $a^r b^t$ with $r, t \geq 1$. All these words are already in $spl^1(G')$.

From the words $a^r b^{n_1} a^{n_1} b^{n_2} a^{n_2} \ldots b^{n_s} a^{n_s} b^t$ with $s+1$ changes and $a^p b^m a^m b^q$ we get $a^r b^{n_1} a^{n_1} b^{n_2} a^{n_2} \ldots b^{n_s} a^{n_s} b^m a^m b^q$ with $s+2$ changes.

Example 3.16 Let

$$G = (\{a, b, c\}, \{\#c\$c\#a\}, \{c^m a^n b^n \mid n \geq 1\}$$

where $m \geq 1$ is a fixed number. Then we get

$$spl^r(G) = \{c^t a^n b^n \mid 0 \leq t \leq m, n \geq 1\} \text{ for } r \geq 1,$$

which implies

$$L(G) = \{c^t a^n b^n \mid 0 \leq t \leq m, n \geq 1\}.$$

We slightly extend the definition of splicing systems by allowing an intersection with T^* where T is a subset of the underlying alphabet. This is analogous to the situation in grammars where we take in the language only words over the terminal alphabet. The following definition formalizes this idea.

Definition 3.17 *i) An extended splicing system is a quadruple $G = (V, T, R, A)$ where $H = (V, R, A)$ is a splicing system and T is a subset of V.*

ii) The language generated by an extended splicing system G is defined as $L(G) = L(H) \cap T^*$.

Example 3.18 Let

$$G = (\{a, b, c\}, \{a, b\}, \{\#c\$c\#a\}, \{c^m a^n b^n \mid n \geq 1\}$$

where $m \geq 1$ is a fixed number. From Example 3.16 we obtain

$$\begin{aligned} L(G) &= \{c^t a^n b^n \mid 0 \leq t \leq m, n \geq 1\} \cap \{a, b\}^* \\ &= \{a^n b^n \mid n \geq 1\}. \end{aligned}$$

We now extend Definitions 3.14 and 3.17 to language families.

Definition 3.19 *For two language families $\mathcal{L}_1$ and $\mathcal{L}_2$, we define $Spl(\mathcal{L}_1, \mathcal{L}_2)$ ($ESpl(\mathcal{L}_1, \mathcal{L}_2)$) as the set of all languages $L(G)$ generated by some splicing system $G = (V, R, A)$ (extended splicing system $G = (V, T, R, A)$) with $A \in \mathcal{L}_1$ and $R \in \mathcal{L}_2$.*

We now give the position of the sets $Spl(\mathcal{L}_1, \mathcal{L}_2)$ where $\mathcal{L}_1$ and $\mathcal{L}_2$ are families of the Chomsky hierarchy within the Chomsky hierarchy.

Theorem 3.20 *The table of Figure 3.12 holds, where at the intersection of the row marked by X and the column marked by Y we give Z if $\mathcal{L}(Z) = Spl(\mathcal{L}(X), \mathcal{L}(Y))$ and Z_1/Z_2 if $\mathcal{L}(Z_1) \subset Spl(\mathcal{L}(X), \mathcal{L}(Y)) \subset \mathcal{L}(Z_2)$.*

We omit the proof of Theorem 3.20. Most of the results can easily be obtained from the proof of the following theorem which is the statement for the families $ESpl(\mathcal{L}_1, \mathcal{L}_2)$.

Theorem 3.21 *The table of Figure 3.13 holds, where at the intersection of the row marked by X and the column marked by Y we give Z if $\mathcal{L}(Z) = ESpl(\mathcal{L}(X), \mathcal{L}(Y))$.*

Before giving the proof of Theorem 3.21 we present some lemmas which will be used in the proof.

The first lemma is the counterpart of Lemma 3.9 which follows from the definitions, again.

	FIN	REG	CF	CS	RE
FIN	FIN/REG	FIN/RE	FIN/RE	FIN/RE	FIN/RE
REG	REG	REG/RE	REG/RE	REG/RE	REG/RE
CF	CF	CF/RE	CF/RE	CF/RE	CF/RE
CS	CS/RE	CS/RE	CS/RE	CS/RE	CS/RE
RE	RE	RE	RE	RE	RE

Figure 3.12: Relations for the families $Spl(\mathcal{L}_1, \mathcal{L}_2)$

	FIN	REG	CF	CS	RE
FIN	REG	RE	RE	RE	RE
REG	REG	RE	RE	RE	RE
CF	CF	RE	RE	RE	RE
CS	RE	RE	RE	RE	RE
RE	RE	RE	RE	RE	RE

Figure 3.13: Relations for the families $ESpl(\mathcal{L}_1, \mathcal{L}_2)$

Lemma 3.22 *For any language families $\mathcal{L}_1, \mathcal{L}_2, \mathcal{L}'_1, \mathcal{L}'_2$ with $\mathcal{L}_1 \subseteq \mathcal{L}'_1$ and $\mathcal{L}_2 \subseteq \mathcal{L}'_2$, we have $ESpl(\mathcal{L}_1, \mathcal{L}_2) \subseteq ESpl(\mathcal{L}'_1, \mathcal{L}'_2)$.* □

Lemma 3.23 *If a language family $\mathcal{L}$ is closed under concatenation with symbols, then $\mathcal{L} \subseteq ESpl(\mathcal{L}, \mathcal{L}(FIN))$.*

Proof. Let L be an arbitrary language of $\mathcal{L}$ over the alphabet V, and let c be a letter not contained in c. Then we consider the splicing system

$$G = (V \cup \{c\}, V, \{\#c\$c\#\}, L\{c\}) .$$

it is easy to see that

$$\begin{aligned} spl^0(G) &= L\{c\}, \\ spl^n(G) &= L \cup L\{c\} \text{ for } n \geq 1, \\ L(G) &= L. \end{aligned}$$

Thus $L \in ESpl(\mathcal{L}, \mathcal{L}(FIN)$ which proves the statement. □

Lemma 3.24 $\mathcal{L}(REG) \subseteq Espl(\mathcal{L}(FIN), \mathcal{L}(FIN))$

Proof. Let L be an arbitrary regular language over T^*. Then there exists a regular grammar $G = (N, T, P, S)$ such that $L = L(G)$ and all rules of P have the form $X \to aY$ or $X \to a$ where X and Y are nonterminals and a is a terminal (see Chapter 1).

We construct the extended splicing system

$$H = (N \cup T \cup \{Z\}, T, R_1 \cup R_2, \{S\} \cup A_1 \cup A_2)$$

with

$$\begin{aligned} R_1 &= \{\#X\$Z\#aY \mid X \to aY \in P,\ X,Y \in N,\ a \in T\}, \\ R_2 &= \{\#X\$ZZ\#a \mid X \to a \in P,\ X \in N,\ a \in T\}, \\ A_1 &= \{ZaY \mid X \to aY \in P,\ X,Y \in N,\ a \in T\}, \\ A_2 &= \{ZZa \mid X \to a \in P,\ X \in N,\ a \in T\}. \end{aligned}$$

Note that the set of splicing rules and the set of start words are finite.

Now we apply the splicing rules in the following order:

$$\begin{aligned} (S, Za_1A_1) &\vdash_{R_1} a_1A_1 && \text{where } S \to a_1A_1 \in P \\ (a_1A_1, Za_2A_2) &\vdash_{R_1} a_1a_2A_2 && \text{where } A_1 \to a_2A_2 \in P, \\ (a_1a_2A_2, Za_3A_3) &\vdash_{R_1} a_1a_2a_3A_3 && \text{where } A_2 \to a_3A_3 \in P, \\ &\dots && \\ (a_1a_2\dots a_{n-2}A_{n-2}, Za_{n-1}A_{n-1}) &\vdash_{R_1} a_1a_2\dots a_{n-1}A_{n-1} && \text{where } A_{n-2} \to a_{n-1}A_{n-1} \in P, \\ (a_1a_2\dots a_{n-1}A_{n-1}, ZZa_n) &\vdash_{R_1} a_1a_2\dots a_n && \text{where } A_{n-1} \to a_n \in P. \end{aligned}$$

This can be considered as a simulation of the derivation

$$\begin{aligned} S &\Longrightarrow a_1A_1 \Longrightarrow a_1a_2A_2 \Longrightarrow \dots \\ &\Longrightarrow a_1a_2\dots a_{n-2}A_{n-2} \\ &\Longrightarrow a_1a_2\dots a_{n-2}a_{n-1}A_{n-1} \\ &\Longrightarrow a_1a_2\dots a_{n-2}a_{n-1}a_n. \end{aligned}$$

This proves $L = L(G) \subseteq L(H)$.

It is easy to see that there are no other possibilities to obtain a word of T^* by iterated splicing. Therefore $L(H) \subseteq L$, too.

Hence any regular language L is in $ESpl(\mathcal{L}(FIN), \mathcal{L}(FIN))$. □

Lemma 3.25 *For any family $\mathcal{L}$ which is closed under union, concatenation, Kleene-closure, homomorphisms, inverse homomorphisms and intersections with regular sets, $ESpl(\mathcal{L}, \mathcal{L}(FIN)) \subseteq \mathcal{L}$.*

Proof. We omit the long and technically hard proof. A complete proof can be found in [12]. □

Lemma 3.26 *For any recursively enumerable language $L \subseteq T^*$, there is an extended splicing system $G = (V, T, R, A)$ with a finite set A and a regular set R of splicing rules such that $L(G) = L$.*

Proof. Let L be an arbitrary recursively enumerable language, and let $G = (N, T, P, S)$ be the phrase structure grammar such that $L(G) = L$. Then we construct the extended splicing system $H = (V, T, R, A)$ with

$$\begin{aligned} U &= N \cup T \cup \{B\}, \\ V &= U \cup \{X, X', Y, Z\} \cup \{Y_a \mid a \in U\} \\ A &= \{XBSY, ZY, XZ\} \cup \{ZvY \mid u \to v \in P\} \\ &\quad \{ZY_a \mid a \in U\} \cup \{X'aZ \mid a \in U\} \end{aligned}$$

and R consists of all rules of the following forms:

1) $Xw\#aY\$Z\#Y_a$ for $a \in U, w \in U^*$,
2) $X'a\#Z\$X\#wY_a$ for $a \in U, w \in U^*$,
3) $X'w\#Y_a\$Z\#Y$ for $a \in U, w \in U^*$,
4) $X\#Z\$X'\#wY$ for $w \in U^*$,
5) $Xw\#uY\$Z\#vY$ for $u \to v \in P, w \in U^*$,
6) $\#ZY\$XB\#wY$ for $w \in T^*$,
7) $\#Y\$XZ\#$.

The letters X, X', Y, Z and Y_a for $a \in U$ are used as endmarkers (more precisely, as the first or last letter of the word. This leads to the situation that the rules 1) – 5) involve the complete words.

In the first step we have to apply a splicing rule to two words of A. If we do not take $XBSY$ as one of these words, the only possible simple splicing are

$$(ZY, XZ) \vdash_7 Z \text{ and } (ZvY, XZ) \vdash_7 Zv$$

(where the index of $\vdash$ refers to the type of the rule which is used), and in both cases there is no splicing rule which can be applied to the resulting word. Thus we have to start with $XBSY$.

Assume that we have obtained $XBwY$. Then we get the following sequence of splicings using the word obtained in the last step together with a word of A:

$$\begin{aligned}(XBw'aY, ZY_a) &\vdash_1 XBw'Y_a,\\ (X'aZ, XBw'Y_a) &\vdash_2 X'aBw'Y_a,\\ (X'aBwY_a, ZY) &\vdash_3 X'aBw',\\ (XZ, X'aBw'Y) &\vdash_4 XaBw'Y.\end{aligned}$$

Therefore we have performed a shift of the last letter a to the beginning of the word. This process can be iterated such that we can get any word Xw_2Bw_1 where $w = w_1w_2$. Further we see that B is used to mark the beginning of the original word w.

Without blocking the splicing the above sequence is the only possible one besides the special situation $Xw_2Bw_1'uY$ where u is a left hand side of a production $u \to v \in P$. Then we also can apply one rule of type 5 and get

$$(Xw_2Bw_1'uY, ZvY) \vdash_5 Xw_2Bw_1'vY\,.$$

Thus we can get the following sequence of results of splicings

$$XBw_1'uw_2Y, \ldots,\ Xw_2Bw_1'uY,\ Xw_2Bw_1'vY, \ldots,\ XBw_1'vw_2Y\,.$$

Therefore we have simulated a derivation step of G (besides the endmarkers).

Note that during one complete shift we can apply some rules to non-overlapping words. This is can be done in G by some derivation steps, too.

If we finish the simulation of a terminating derivation in G, then we get a word $XBwY$ with $w \in T^*$ and $w \in L$. We apply a splicing rule of type 6) and 7) and yield

$$\begin{aligned}(ZY, XBwY) &\vdash_6 wY,\\ (wY, XZ) &\vdash_7 w.\end{aligned}$$

Thus we have shown that $L = L(G) \subseteq L(H)$.

Furthermore, it can be seen that other sequences of splicing rules lead to a blocking situation and the obtained word is not a word of T^*. Therefore $L(H) \subseteq L$, too. □

Lemma 3.27 *For any extended splicing system $G = (V, T, R, A)$, $L(G)$ is a recursively enumerable set.*

Proof. The proof can be given by constructing a corresponding phrase structure grammar. We omit the detailed construction. □

Proof of Theorem 3.21 By Lemmas 3.22, 3.24 and 3.25, we obtain

$$\mathcal{L}(REG) \subseteq ESpl(\mathcal{L}(FIN), \mathcal{L}(FIN)) \subseteq ESpl(\mathcal{L}(REG), \mathcal{L}(REG)) \subseteq \mathcal{L}(REG)\,.$$

These relations imply

$$\mathcal{L}(REG) = ESpl(\mathcal{L}(FIN), \mathcal{L}(FIN)) = ESpl(\mathcal{L}(REG), \mathcal{L}(FIN))\,.$$

By Lemmas 3.23 and 3.25, we get

$$\mathcal{L}(CF) \subseteq ESpl(\mathcal{L}(CF), \mathcal{L}(FIN)) \subseteq \mathcal{L}(CF)$$

which yields $\mathcal{L}(CF) = ESpl(\mathcal{L}(CF), \mathcal{L}(FIN))$.

Analogously, we obtain $\mathcal{L}(RE) = ESpl(\mathcal{L}(RE), \mathcal{L}(FIN))$.

In the proof of Theorem 3.8 we have shown that, for any recursively enumerable language L, there is a context-sensitive language L' and a regular set R of splicing rules such that $L = spl(L', R)$. It is easy to see (or to prove analogously to Lemma 3.23) that $L = L(G)$ for the extended splicing system $G = (T \cup \{c_1, c_2, c_3\}, T, R, L')$.

Therefore we have $\mathcal{L}(RE) \subseteq ESpl(\mathcal{L}(CS), \mathcal{L}(FIN))$. Together with Lemma 3.22 and $\mathcal{L}(RE) = ESpl(\mathcal{L}(RE), \mathcal{L}(FIN))$ we get $\mathcal{L}(RE) = ESpl(\mathcal{L}(CS), \mathcal{L}(FIN))$.

Lemma 3.26 and 3.27 can be formulated as $\mathcal{L}(RE) \subseteq ESpl(\mathcal{L}(FIN), \mathcal{L}(REG))$ and $Espl(\mathcal{L}(RE), \mathcal{L}(RE) \subseteq \mathcal{L}(RE)$. By combination with Lemma 3.22, we obtain $\mathcal{L}(RE) = ESpl(\mathcal{L}(X), \mathcal{L}(Y))$ for $X \in \{FIN, REG, CF, CS, RE\}$ and $Y \in \{REG, CF, CS, RE\}$.

□

3.3.3 Remarks on descriptional complexity

In the theory of descriptional complexity one studies hierarchies which can be obtained by restricting some parameters which can be seen immediately from the (extended) splicing system.

First we define the parameters or measures which we shall consider and the corresponding language families.

Definition 3.28 *i) For a splicing system $G = (V, R, A)$ or an extended splicing system $G = (V, T, R, A)$ we define the complexity measures $r(G)$, $a(G)$ and $l(G)$ by*

$$\begin{aligned} r(G) &= \max\{|u| \mid u = u_i \text{ for some } u_1\#u_2\$u_3\#u_4 \in R,\ 1 \leq i \leq 4\}, \\ a(G) &= \#(A), \\ l(G) &= \max\{|z| \mid z \in A\}. \end{aligned}$$

ii) For a language family $\mathcal{L}$ and $n \geq 1$ and $m \in \{a, l\}$, we define the families $\mathcal{L}_n(r, \mathcal{L})$ and $\mathcal{L}_n(m, \mathcal{L})$ as the set of languages $L(G)$ where $G = (V, R, A)$ is a splicing system with $r(G) \leq n$ and $A \in \mathcal{L}$ and with $m(G) \leq n$ and $R \in \mathcal{L}$, respectively.

iii) Analogously, for $m \in \{r, a, l\}$, we define the sets $\mathcal{L}_n(em, \mathcal{L})$ taking extended splicing systems (instead of splicing systems).

$r(G)$ is called the radius of G since it gives the maximal neighbourhood of the place of splitting which is involved in the splicing. The other two measures concern the size of the (finite) set of start words where the size is measured by the cardinality of the set or the maximal length of words in it.

As a first result on the descriptional complexity of splicing systems we show that we obtain an infinite hierarchy between the classes $\mathcal{L}(FIN)$ and $Spl(\mathcal{L}(FIN), \mathcal{L}(FIN))$ with respect to the radius.

Theorem 3.29 *For any $n \geq 1$,*

$$\mathcal{L}(FIN) \subset \mathcal{L}_n(r, \mathcal{L}(FIN)) \subset Spl(\mathcal{L}(FIN), \mathcal{L}(FIN))$$

and

$$\mathcal{L}_n(r, \mathcal{L}(FIN)) \subset \mathcal{L}_{n+1}(r, \mathcal{L}(FIN))$$

Proof. All inclusions follow by definition and the construction in the proof of Lemma 3.23.

In order to prove that the inclusion $\mathcal{L}(FIN) \subset \mathcal{L}_1(r, \mathcal{L}(FIN))$ is proper, we consider the splicing system

$$G = (\{a\}, \{a\#\$\#a\}, \{a\})$$

for which

$$\begin{aligned} spl^i(G) &= \{a, a^2, \ldots, a^{2^i}\}, \\ L(G) &= \{a\}^+ \end{aligned}$$

holds (the statement on $spl^i(G)$ can easily be proved by induction on i; the only new words in $spl^{i+1}(G)$ are obtained by $(a^{2^i}, a^k) \vdash a^{2^i+k}$ where $1 \leq k \leq 2^i$) which generates an infinite language and satisfies $r(G) \leq 1$.

We now prove that $\mathcal{L}_n(r, \mathcal{L}(FIN)) \subset \mathcal{L}_{n+1}(r, \mathcal{L}(FIN))$ for $n \geq 1$, which implies the strictness of $\mathcal{L}_n(r, \mathcal{L}(FIN)) \subset Spl(\mathcal{L}(FIN), \mathcal{L}(FIN))$, too.

For $n \geq 1$, let

$$L_n = \{a^{2n}b^{2n}a^m b^{2n}a^{2n} \mid m \geq 2n+1\}.$$

The splicing system

$$G_n = (\{a, b\}, \{a^{n+1}\#a^n\$a^{n+1}\#a^n\}, \{a^{2n}b^{2n}a^{2n+2}b^{2n}a^{2n}\})$$

satisfies $r(G_n) = n + 1$. Let

$$(u_1r_1r_2u_2, v_1r_3r_4v_2) \vdash w,\ u_1r_1r_2u_2 = a^{2n}b^{2n}a^s b^{2n}a^{2n},\ v_1r_3r_4v_2 = a^{2n}b^{2n}a^t b^{2n}a^{2n}$$

for some integers $s, t \geq 2n + 1$. Since $r_1r_2 = r_3r_4 = a^{2n+1}$, in both word we have to perform the split in the inner part a^m with $m \geq 2n+1$ which leads to $w = a^{2n}b^{2n}a^r b^{2n}a^{2n}$

with $2n+1 \leq r \leq s+t-2n-1$. Because we start with a word where the inner part has the length $2n+2$ we can produce by iterated applications any length in the inner part. Therefore $L(G_n) = L_n$. Thus $L_n \in \mathcal{L}_{n+1}(r, \mathcal{L}(FIN))$.

Now let us assume that $L_n = L(G)$ for some splicing system $G = (\{a,b\}, R, A)$ with $A \in \mathcal{L}(FIN)$ and $r(G) \leq n$. Let $p = r_1\#r_2\$r_3\#r_4$ be a splicing rule of R. Then $|r_1r_2| \leq 2n$. We apply p to the words $u = u_1r_1r_2u_2 = a^{2n}b^{2n}a^rb^{2n}a^{2n}$ and $v = v_1r_3r_4v_2$

Let $r_1r_2 \in \{a\}^+$. Then we can apply p by splitting the prefix a^{2n} of u. We get the word $w_1 = u_1r_1r_4v_2$. Since w_1 has to be an element of $L(G)$ and therefore of L_n and u_1r_1 contains only a's, r_4v_2 contains the subword $z_2b^{2n}a^kb^{2n}a^{2n}$ for some $k \geq 2n+1$. If we now apply p by splitting the suffix a^{2n} of u, then we get

$$w_2 = a^{2n}b^{2n}a^{k'}b^{2n}z_1z_2b^{2n}a^kb^{2n}a^{2n} \in L(G)$$

which does not belong to L_n in contrast to $L(G) = L_n$.

In the other cases, i.e., r_1r_2 is contained in $\{a\}^+\{b\}^+$ or $\{b\}^+$ or $\{b\}^+\{a\}^+$, we also find two different places where r can be used in u and at least one of these words does not belong to L_n.

Hence we have shown that L_n cannot be generated by a splicing system G with $r(G) \leq n$.

This yields $\mathcal{L}_n(r, \mathcal{L}(FIN)) \subset \mathcal{L}_{n+1}(r, \mathcal{L}(FIN))$. □

The situation changes completely if we switch to another family $\mathcal{L}$ as can be seen from the following theorem where the hierarchies collapse at the first level.

Theorem 3.30 *For $\mathcal{L} \in \{\mathcal{L}(REG), \mathcal{L}(CF), \mathcal{L}(RE)\}$ and $n \geq 1$, $\mathcal{L}_n(r, \mathcal{L}) = \mathcal{L}$.*

Proof. Let $\mathcal{L} \in \{\mathcal{L}(REG), \mathcal{L}(CF), \mathcal{L}(RE)\}$.

For a language $L \in \mathcal{L}$ and $L \subseteq V^*$, we consider the splicing system

$$G = (V \cup \{c\}, \{\#c\$c\#\}, L)\,.$$

Then the splicing rule cannot be applied which yields $spl^i(G) = L$ and therefore $L(G) = L$. Therefore

$$\mathcal{L} \subseteq \mathcal{L}_1(r, \mathcal{L}). \tag{3.1}$$

Furthermore, by definition and Theorem 3.20 we have

$$\mathcal{L}_n(r, \mathcal{L}) \subseteq \mathcal{L}_m(r, \mathcal{L}) \subseteq \mathcal{L}(\mathcal{L}, \mathcal{L}(FIN)) = \mathcal{L} \tag{3.2}$$

for $m \geq n$. From (3.1) and (3.2) we get the statement of the lemma. □

We now investigate the hierarchies obtained for the measures related to the size of the set of start words in the case of extended systems.

Theorem 3.31 *For any $n \geq 1$,*

$$\mathcal{L}_n(ea, \mathcal{L}(REG)) = \mathcal{L}(RE)\,.$$

Proof. By definition and Theorem 3.21, for any $n \geq 1$,

$$\mathcal{L}_1(ea, \mathcal{L}(REG)) \subseteq \mathcal{L}_n(ea, \mathcal{L}(REG)) \subseteq \mathcal{L}(\mathcal{L}(FIN), \mathcal{L}(REG) = \mathcal{L}(RE)\,.$$

Therefore it is sufficient to prove that any recursively enumerable language L is contained in $\mathcal{L}_1(ea, \mathcal{L}(REG))$.

Let $L \in \mathcal{L}(RE)$. Then $L = L(G)$ for some extended splicing system $G = (V, T, R, A)$ with a finite set $A = \{w_1, w_2, \ldots, w_n\}$. If $n = 1$, then $L \in \mathcal{L}_1(ea, \mathcal{L}(REG))$. If $n \geq 2$ we construct the extended splicing system

$$G' = (V \cup \{c, c'\}, T, R', \{w\})$$

with two additional letters c and c',

$$R' = R \cup \{\#c'c\$c\#c', \#c\$c'\#, \#c'\$c\#\}$$

and

$$u = c'cw_1cw_2cw_3c \ldots cw_ncc'\,.$$

Let i be an integer with $1 \leq i \leq n$. We have the following sequence of splicings

$(u, u) \vdash c'$	using $\#c'c\$c\#c'$
$(u, c') \vdash c'cw_1cw_2c \ldots cw_{i-1}cw_i = u_i$	using $\#c\$c'\#$
$(c', u_i) \vdash w_i$	using $\#c'\$c\#$

Thus we have $w_i \in spl^3(G')$ for $1 \leq i \leq n$. Taking these words and the rules of $R \subset R'$ we can generate any word of $L(G)$ and therefore $L(G) \subseteq L(G')$.

If we apply a rule $r_1\#r_2\$r_3\#r_4$ to a word w, then $w = u_1r_1r_2u_2$ or $w = u_1r_1r_2u_2cx_2$ or $w = x_1cu_1r_1r_2u_2cx_2$ or $w = x_1cu_1r_1r_2u_2$ for some words $u_1, u_2 \in V^*$ and $x_1, x_2 \in (V \cup \{c, c'\})^*$. The same situation holds with respect to the second word w' containing r_3r_4. We discuss now the case that w is of the third type and w' is of the second type, i.e., $w = x_1cu_1r_1r_2u_2cx_2$ and $w' = v_1r_3r_4v_2cy_2$. Then we get $(w, w') \vdash x_1cu_1r_1r_4v_2cy_2$ which corresponds to a splicing of two words over V neighboured by c. Hence any generation of a word over V can be obtained by a first phase using only rules from $R' \setminus R$ and yielding words from A and a second phase using only rules of R and yielding words of $L(G)$. Hence $L(G') \subseteq L(G)$. □

Theorem 3.32 *For any $n \geq 2$,*

$$\mathcal{L}_1(el, \mathcal{L}(REG)) \subset \mathcal{L}_n(el, \mathcal{L}(REG)) = \mathcal{L}(RE)\,.$$

Proof. By definition and Theorem 3.21, for any $n \geq 2$,

$$\mathcal{L}_2(ea, \mathcal{L}(REG)) \subseteq \mathcal{L}_n(ea, \mathcal{L}(REG)) \subseteq \mathcal{L}(\mathcal{L}(FIN), \mathcal{L}(REG) = \mathcal{L}(RE)\,.$$

Therefore it is sufficient to prove that any recursively enumerable language L is contained in $\mathcal{L}_2(ea, \mathcal{L}(REG))$.

Let $L \in \mathcal{L}(RE)$. Then $L = L(G)$ for some extended splicing system $G = (V, T, R, A)$ with a finite set $A = \{w_1, w_2, \ldots, w_n\}$. For any i, $1 \leq i \leq n$, let c_i and c_i' be two new symbols. We set

$$G' = (V \cup \bigcup_{i=1}^{n} \{c_i, c_i'\}, T, R', A')$$

with

$$\begin{aligned} A' &= \{c_i a \mid 1 \leq i \leq n, a \in V\} \cup \bigcup_{i=1}^{n} \{c_i, c_i'\}, \\ R_i &= \{c_i x \# \$ c_i \# a \mid x, xa \text{ are prefixes of } w_i,\ a \in V\} \\ &\quad \cup \bigcup_{i=1}^{n} \{c_i w_i \# \$ \# c_i', \# c_i \$ c_i \# w_i c_i', w_i \# c_i' \$ c_i' \#\}, \\ R' &= R \cup \bigcup_{i=1}^{n} R_i . \end{aligned}$$

Let $w_i = a_{i,1}a_{i,2}\ldots a_{i,r_i}$. We have the following splicings

$$\begin{array}{ll} (c_i a_{i,1}, c_i a_{i,2}) \vdash c_i a_{i,1} a_{i,2} & \text{using } c_i a_{i,1} \# \$ c_i \# a_{i,2}, \\ (c_i a_{i,1} a_{i,2}, c_i a_{i,3}) \vdash c_i a_{i,1} a_{i,2} a_{i,3} & \text{using } c_i a_{i,1} a_{i,2} \# \$ c_i \# a_{i,3}, \\ \ldots \quad \ldots & \ldots \quad \ldots \\ (c_i a_{i,1} a_{i,2} \ldots a_{i,r_i-1}, c_i a_{i,r_i}) \vdash c_i a_{i,1} a_{i,2} \ldots a_{i,r_i} & \text{using } c_i a_{i,1} a_{i,2} \ldots a_{i,r_i-1} \# \$ c_i \# a_{i,r_i}. \end{array}$$

Therefore $c_i w_i$ is obtained. We continue by

$$\begin{array}{ll} (c_i w_i, c_i') \vdash c_i w_i c_i' & \text{using } c_i w_i \# \$ \# c_i', \\ (c_i, c_i w_i c_i') \vdash w_i c_i' & \text{using } \# c_i \$ c_i \# w_i c_i', \\ (w_i c_i', c_i') \vdash w_i & \text{using } w_i \# c_i' \$ c_i' \#. \end{array}$$

Thus we get w_i for $1 \leq i \leq n$. Using the splicing rules from R we can now generate all words of $L(G) = L$. Thus $L \subseteq L(G')$.

Since any start word contains at least one symbol c_i or c_i', we have to cancel these symbols at a certain step. These cancellations are only possible if – besides the endmarkers c_i and c_i' – a word $w_i \in A$ is produced. If we apply rules from R before c_i and c_i' have been cancelled, then the word – besides the endmarkers – is a prefix of w_i and we can generate from it w_i or it is not a prefix of w_i and there is no continuation which cancels the endmarkers. Thus the above presented steps by splicing are the only possible ones, Hence $L(G') \subseteq L$, too.

Obviously, if we generate a language $L \subset T^*$ by a system G, where the maximal length of the axioms is 1, then the set of axioms has to contain at least one letter a of T. Then $a \in L(G)$. However, there are (finite) recursively enumerable sets which contain only words of length greater than 2. Thus $\mathcal{L}_1(el, \mathcal{L}(REG)) \subset \mathcal{L}(RE)$ which proves the statement. □

3.3.4 Splicing on multisets

The theory developed up to now has some bad features. First, we have considered only the operation $\vdash_R$ which regards only one of the two words produced by the splicing. Second, the derivation process by splicing has been studied as a sequential process, i.e., in any step we applied *one* splicing rule to two words. In nature, splicings occur as a parallel process, i.e., some rules are applied in parallel. Third, by definition the words taken for some splicing did not vanish, they can be used later again. Thus one supposes that, for

any of the start words and the generated words, an (at least potentially) infinite number of copies is present. This does not reflect reality. Therefore we now modify the concept in order to cover all these aspects.

For two words w and v and a splicing rule $p = u_1\#u_2\$u_3\#u_4$ such that $w = w_1u_1u_2w_2$ and $v = v_1u_3u_4v_2$, we write

$$[w, v] \underset{p}{\Longrightarrow} [w', v']$$

where $w' = w_1u_1u_4v_2$ and $v' = v_1u_3u_2w_2$.

Let $[w, v] \underset{p}{\Longrightarrow} [w', v']$ and $a \in V$. From the definitions, we obtain immediately

$$l([w, v]) = l([w', v']) \text{ and } \#_a([w, v]) = \#_a([w', v']). \tag{3.3}$$

Definition 3.33 *A multiset splicing system (MSS for short) is a triple $G = (V, P, M)$ where*
– *V is an alphabet,*
– *P is a finite set of splicing rules over V such that, for any rule $u_1\#u_2\$u_3\#u_4 \in R$, $u_i \neq \lambda$ holds for $1 \leq i \leq 4$, and*
– *M is a finite multiset over V.*

Definition 3.34 *For two multisets $M = [w_1, w_2, \ldots, w_n]$ and $M' = [v_1, v_2, \ldots, v_n]$ of words over V and a set P of splicing rules over V, we define*

- *a sequential derivation step $M \Longrightarrow_s M'$ by $[w_1, w_2] \underset{p}{\Longrightarrow} [v_1, v_2]$ and $w_i = v_i$ for $3 \leq j \leq n$ for some $p \in P$ and some appropriate order of the elements in M and M',*

- *a maximally parallel derivation step $M \Longrightarrow_{mp} M'$ by $[w_{2i-1}, w_{2i}] \underset{p_i}{\Longrightarrow} [v_{2i-1}, v_{2i}]$ for $1 \leq i \leq k \leq \frac{n}{2}$ and $w_i = v_i$ for $2k+1 \leq j \leq n$ for some $p_i \in P$ and some appropriate order of the elements in M and M', and by the requirement that there is no multiset $[w, w'] \subseteq [w_{2k+1}, w_{2k+2}, \ldots, w_n]$ to which a splicing rule $p \in P$ can be (successfully) applied,*

- *a strongly maximally parallel derivation step $M \Longrightarrow_{smp} M'$ by $M \underset{mp}{\Longrightarrow} M'$ for some k (as in the preceding item) and there is no M'' with $M \Longrightarrow_{mp} M''$ for some $k' > k$.*

As usually, by $\Longrightarrow_s^*$, $\Longrightarrow_{mp}^*$, and $\Longrightarrow_{smp}^*$, we denote the reflexive and transitive closures of $\Longrightarrow_s$, $\Longrightarrow_{mp}$, and $\Longrightarrow_{smp}$, respectively.

Definition 3.35 *We define the sequential, maximally parallel and strongly maximally parallel multiset language $mL(G, s)$, $mL(G, mp)$ and $mL(G, smp)$ generated by G as*

$$\begin{aligned} mL(G, s) &= \{K \mid M \Longrightarrow_s^* K\}, \\ mL(G, mp) &= \{K \mid M \Longrightarrow_{mp}^* K\}, \\ mL(G, smp) &= \{K \mid M \Longrightarrow_{smp}^* K\}. \end{aligned}$$

Example 3.36 We consider the multiset splicing system

$$G = (\{a, b, c, d\}, \{r_1, r_2, r_3\}, M)$$

with

$$\begin{aligned} r_1 &= a\#b\$d\#d\,, \\ r_2 &= b\#b\$d\#d\,, \\ r_3 &= b\#b\$c\#d\,, \\ M &= [ab, abb, cd, cdd]\,. \end{aligned}$$

By a sequential application of r_1 we obtain from M the multisets

$$M_1 = [ad, cdb, abb, cd] \text{ and } M_2 = [ad, cdbb, ab, cd]\,,$$

and by applications of r_2 and r_3 we get

$$M_3 = [abd, cdb, ab, cd],\ M_4 = [abd, cb, ab, cdd] \text{ and } M_5 = [abdd, cb, ab, cd],$$

respectively. We can apply only r_3 to M_1, and this is possible for two different pairs of M_1. These applications yield

$$M_6 = [abd, cb, ad, cdb] \text{ and } M_7 = [abdb, bc, ad, cd].$$

We can only apply r_3 to M_2 and obtain

$$M_8 = [cdbd, cb, ad, ab].$$

No rule can be applied to M_3. Further, r_1 can be applied to two different pairs of M_4, which gives M_6 and M_8,again, and no other rule can be applied to M_4. No rule can be used for M_5. Since M_6, M_7 and M_8 do not allow the application of some rule, we have

$$mL(G, s) = \{M, M_1, M_2, M_3, M_4, M_5, M_6, M_7, M_8\}\,.$$

We consider now the maximally parallel mode of derivation. If we apply r_1 to abb and cdd, then there is no rule which can be applied to ab and cd. Thus this derivation is maximally parallel and gives M_2. M_2 only allows the application of r_3; and since its application is a maximally parallel derivation step, we get M_8.

If we apply r_1 to ab and cdd, then we can apply r_3 to abb and cd, too, and obtain M_6, to which no rule can be applied.

If we apply r_2 to M, then abb and cdd are involved and there is no rule which can be applied to ab and cd. Thus this derivation is maximally parallel and gives M_3, and no further derivation is possible. If we apply r_3 to M, this yields M_5 and M_6 in a maximally parallel way.

All together, this implies

$$mL(G, mp) = \{M, M_2, M_3, M_5, M_6, M_8\}\,.$$

Since there is a parallel derivation step which involves all four words of M, this is the only strongly maximally parallel derivation from M. Therefore

$$mL(G, smp) = \{M, M_6\}.$$

For $Y \in \{s, mp, smp\}$, we denote by $m\mathcal{L}(Y)$ the family of all languages $mL(G, Y)$ which can be generated by a multiset splicing system G in the derivation mode Y. If we restrict to multiset splicing systems $G = (V, P, M)$ with a multiset M of cardinality n, we use the notation $m\mathcal{L}_n(Y)$.

Lemma 3.37 *For a multiset splicing system $G = (V, P, M)$, $a \in V$, $Y \in \{s, mp, smp\}$, and any $K \in mL(G, Y)$,*

$$\#(K) = \#(M),\ l(K) = l(M),\ \text{and } \#_a(K) = \#_a(M).$$

Proof. The assertions immediately follow from (3.3) and from the fact that any splicing rule p has to be applied to two words and yields two words. □

In the following, we compare the language classes generated in the different derivation modes, $m\mathcal{L}_n(Y)$, $Y \in \{s, mp, smp\}$.

Lemma 3.38 *For two integers n and m, $m \neq n$, and two derivation modes $Y \in \{s, mp, smp\}$ and $Y' \in \{s, mp, smp\}$, $m\mathcal{L}_n(Y)$ and $m\mathcal{L}_m(Y')$ are incomparable.*

Proof. Let L be a set of multisets in $m\mathcal{L}_n(Y)$. Then $\#(K) = n$ for any multiset K of L. Analogously, $\#(K') = m$ for any multiset K' of some $L' \in m\mathcal{L}_m(Y')$. Thus $L \notin m\mathcal{L}_m(Y')$ and $L' \notin m\mathcal{L}_n(Y)$. □

Lemma 3.39 *i) For $n \in \{1, 2, 3\}$, $m\mathcal{L}_n(s) = m\mathcal{L}_n(mp) = m\mathcal{L}_n(smp)$.*
ii) For $n \geq 4$, $m\mathcal{L}_n(mp)$ and $m\mathcal{L}_n(smp)$ are both incomparable to $m\mathcal{L}_n(s)$.
iii) For $n \geq 5$, $m\mathcal{L}_n(mp)$ is not contained in $m\mathcal{L}_n(smp)$.
iv) For $n \geq 6$, the classes $m\mathcal{L}_n(s)$, $m\mathcal{L}_n(mp)$, and $m\mathcal{L}_n(smp)$ are pairwise incomparable.

Proof. i) If $n = 1$ then no application of splicing rules is possible, and therefore the statement is true. If $n = 2$ or $n = 3$, then exactly two elements are taken to apply a splicing rule in all modes of derivation. This implies the statement.

ii) We give the proof for $n = 4$. In order to obtain a proof for $n \geq 5$ we can add some words to the initial multiset to which the splicing rules cannot be applied.

a) First we prove that $m\mathcal{L}_n(mp)$ and $m\mathcal{L}_n(smp)$ is not contained in $m\mathcal{L}_n(s)$. Let

$$G = (\{a, b\}, \{a\#b\$b\#a\}, [ab, ab, ba, ba]).$$

Then

$$[ab, ab, ba, ba] \Longrightarrow_{mp} [aa, aa, bb, bb] \text{ and } [ab, ab, ba, ba] \Longrightarrow_{smp} [aa, aa, bb, bb]\,.$$

Since there is no rule applicable to elements of $[aa, aa, bb, bb]$, we obtain

$$mL(G, mp) = mL(G, smp) = \{[ab, ab, ba, ba], [aa, aa, bb, bb]\}\,.$$

On the other hand, since a sequential application of a splicing rule changes at most two elements, $[ab, ab, ba, ba] \Longrightarrow_s [aa, aa, bb, bb]$ is impossible as well as $[aa, aa, bb, bb] \Longrightarrow_s [ab, ab, ba, ba]$. Therefore $mL(H, s) \neq mL(G, mp)$ for any MSS H.

b) Now we show that neither $m\mathcal{L}_n(mp)$, nor $m\mathcal{L}_n(smp)$ contains $m\mathcal{L}_n(s)$. Let

$$G = (\{a, b, c, d, e, f, g, h\}, \{a\#b\$c\#d, e\#f\$g\#h\}, [ab, cd, ef, gh]).$$

Then

$$mL(G, s) = \{M_0, M_1, M_2, M_3\},$$

where

$$M_0 = [ab, cd, ef, gh],\ M_1 = [ad, cb, ef, gh],\ M_2 = [ab, cd, eh, gf],$$
$$M_3 = [ad, cb, eh, gf].$$

Let us assume that $mL(G, s) = mL(H, Y)$ for some $H = (\{a, b, c, d, e, f, g, h\}, P, M), Y \in \{mp, smp\}$.

Without the loss of generality, we may assume that $M = M_0$. To obtain the word ad without also obtaining ah or af, P needs to contain the rule $a\#b\$c\#d$. Similarly, to obtain eh, P must contain $e\#f\$g\#h$. This means that

$$M_0 \Longrightarrow_Y M_3$$

is the only possible derivation step from M_0. To obtain M_1 and M_2 from M_3 in any way, P needs to contain the $a\#d\$c\#b$ and $e\#h\$g\#f$, but then the only possibility is

$$M_3 \Longrightarrow_Y M_0$$

which means that $mL(G, s)$ can not be generated by any MSS H in the maximally parallel, or in the strongly maximally parallel modes.

iii) We only give the proof for $n = 5$, the modifications for $n \geq 6$ are similar to that of point ii).

Consider the MSS

$$G = (\{a, b, c, d, e, f, g, h, i, j\}, P, [ab, cd, ef, gh, ij])$$

with

$$P = \{a\#b\$c\#d, c\#d\$e\#f, a\#b\$e\#f, c\#d\$g\#h, c\#d\$i\#j\}.$$

This system generates the following five multisets in the maximally parallel mode:

$$M_0 = [ab, cd, ef, gh, ij],\ M_1 = [ad, cb, ef, gh, ij],\ M_2 = [ab, cf, ed, gh, ij],$$

$$M_3 = [af, ch, eb, gd, ij],\ M_4 = [af, cj, eb, gh, id].$$

Let us assume that $H = (\{a, b, c, d, e, f, g, h, i, j\}, R, M)$ with some set R of splicing rules and some multiset M satisfies

$$mL(H, smp) = mL(G, mp) = \{M_0, M_1, M_2, M_3, M_4\}.$$

Let us observe an important property of the language $mL(G, mp)$. It is impossible to choose a pair M_i, M_j of multisets from $M_1,\ M_2,\ M_3$, and M_4 in such a way that $M_i \Longrightarrow_{smp} M_j$ or $M_j \Longrightarrow_{smp} M_i$ holds (in fact, this is true also for the relation $\Longrightarrow_{mp}$). To see this, consider for example M_1 and M_2. To obtain the word $ab \in M_2$ from M_1, the system

would need the rule $a\#d\$c\#b$, but this rule would also produce $cd \notin M_2$. Or to obtain $ad \in M_1$ from M_2, the rule $a\#b\$e\#d$ is needed, but this would also produce $eb \notin M_1$.

Now let us try to construct the MSS H. We distinguish three cases for the axiom M of H.

Case 1. $M = M_0$. To derive M_1 or M_2, the system needs only one rule, $a\#b\$c\#d$ or $c\#d\$e\#f$, respectively, while to derive M_3 or M_4, it needs two rules, $a\#b\$e\#f,\ c\#d\$g\#h$ or $a\#b\$e\#f,\ c\#d\$i\#j$, respectively. This means that in the strongly maximally parallel derivation mode, the multisets of the language $mL(G, mp)$ can not be derived from M alone because in the strongly maximal parallel mode we can not apply only one rule when it would be possible to apply two. As a consequence of this statement and the fact that no two multisets of M_1, M_2, M_3, and M_4 can be used to derive one from the other, we can conclude that M_0 can not be the axiom of H.

Case 2. $M = M_i,\ i \in \{1, 2\}$. If $M = M_1$, then the rule $a\#d\$c\#b$ can be used to derive M_0, and no other multisets can be derived from M_1 in one step. The same holds for M_2, but we need to use the rule $c\#f\$e\#d$ to reach M_0. From M_0 it is impossible to derive all three remaining multisets in one strongly maximally parallel derivation step each, because, as we already have seen in Case 1 above, to reach M_3 and M_4 we need steps which use two rules, while to reach M_1 or M_2 we need only one rule. Other combinations are also impossible because of the property of $mL(G, mp)$ mentioned above; that is, the impossibility of deriving any one of M_1, M_2, M_3, or M_4 from any other. This means that $M_i,\ i \in \{1, 2\}$ can not be the axiom of H.

Case 3. $M = M_i,\ i \in \{3, 4\}$. The reasoning is similar to the reasoning of Case 2 above. Starting from M_3 (or M_4), only M_0 can be derived using $a\#f\$e\#b,\ c\#h\$g\#d$ (or $a\#f\$e\#b,\ c\#j\$i\#d$). But it is impossible to reach all remaining multisets from M_0 in one step each as we have already seen above, and no two multisets of M_1, M_2, M_3, and M_4 can be used to derive one from the other, so $M = M_i,\ i \in \{3, 4\}$ can not be the axiom of H.

Considering these three cases, we can conclude that it is not possible to get $mL(G, mp)$ by any MSS H in the strongly maximally parallel mode of derivation.

iv) Again, we only give the proof for $n = 6$, the modifications for $n \geq 7$ are left to the reader. By points ii) and iii), it is sufficient to show that $m\mathcal{L}_n(smp)$ is not contained in $m\mathcal{L}_n(mp)$.

Consider the MSS

$$G = (\{e, f, g, h, i, j, k, l\}, \{e\#f\$g\#h, i\#j\$k\#l, e\#f\$i\#j, e\#h\$i\#l\}, [ef, gh, ij, kl, ef, ij])$$

which in the strongly maximal parallel mode generates the following three multisets:

$$M_0 = [ef, gh, ij, kl, ef, ij],\ M_1 = [eh, gf, il, kj, ej, if],\ M_2 = [el, gf, ih, kj, ej, if].$$

(In the first derivation step we apply in parallel the first three rules which is the only possible derivation step in the *smp* mode yielding M_1 from the axiom M_0, now the fourth rule is the only rule applicable to M_1, and this gives M_2 to which no rule can be applied.)

Let us assume that $H = (\{e, f, g, h, i, j, k, l\}, P, M)$ with some set P of splicing rules and some initial multiset M satisfies

$$mL(H, mp) = mL(G, smp) = \{M_0, M_1, M_2\}.$$

Let us try to construct H. We distinguish three cases for the axiom M of H.

Case 1. $M = M_0$. $M_0 \Longrightarrow_{mp} M_2$ is impossible since the generation of el requires the combination of ef and kl which generates kf, too, and $kf \notin M_2$. Therefore we have the derivation $M_0 \Longrightarrow_{mp} M_1$ and it is easy to see that this requires the rules $e\#f\$g\#h$, $i\#j\$k\#l$ and $e\#f\$i\#j$. Then we also have $M_0 \Longrightarrow_{mp} [ej, gh, if, kl, ej, if]$, and thus we can generate a multiset not in $mL(G, smp)$ which gives a contradiction.

Case 2. $M = M_1$. As above, we can show that $M_2 \Longrightarrow_{mp} M_0$ is impossible, because obtaining ef would also yield gl, therefore $M_1 \Longrightarrow_{mp} M_0$ holds which means that P needs to contain the rules $e\#h\$g\#f, i\#l\$k\#j, e\#j\$i\#f$. However, then $M_1 \Longrightarrow_{mp} M_2$ is impossible since th rules above would also change the words ej, if. Since $M_0 \Longrightarrow_{mp} M_2$ is also impossible (see Case 1), we can conclude that $mL(G, smp)$ can not be generated from M_1.

Case 3. $M = M_2$. Since $M_2 \Longrightarrow_{mp} M_0$ is impossible, again, we need the derivation $M_1 \Longrightarrow_{mp} M_0$. As in Case 2 we can show that this implies that $M_2 \Longrightarrow_{mp} M_1$ cannot hold. Therefore $mL(G, smp)$ can neither be generated from M_2, and this concludes our proof. □

Besides comparisons of the generative power, one can also discuss decision problems. However, there are trivial answers with respect to the emptiness problem and finiteness problem, since all the considered types of languages are non-empty and finite. Therefore one can list all elements of the generated language and can answer the membership problem and the universality problem (given a multiset splicing system $G = (V, P, M)$, decide whether or not all multisets M' with words of $\bigcup_{i=0}^{l}(M)V^i$ and $\#(M') = \#(M)$ can be generated). It remains an open problem to discuss the complexity of the membership and universality problem.

3.4 Sticker Systems

In this section the basic operations are bondings and ligases, i.e., we glue together some parts of double strands according to the Watson-Crick complementarity; e.g. from the pieces

```
AACGTAGCGATTT          and       GGCCAATAGGGAAACC
   CATCGCTAAACCGG                      TTATCCCT
```

we obtain the double strand

```
AACGTAGCGATTTGGCCAATAGGGAAACC
   CATCGCTAAACCGGTTATCCCT
```

In order to describe the double strands with overhangs we introduce the following notations. Let V be an alphabet, and let $\varrho \subset V \times V$ be a symmetric relation (i.e., $(a, b) \in \varrho$ implies $(b, a) \in \varrho$[2]. We say that a and b are complementary if $(a, b) \in \varrho$.

We set

$$\begin{bmatrix} V \\ V \end{bmatrix}_\varrho = \{ \begin{bmatrix} a \\ b \end{bmatrix} \mid a, b \in V,\ (a, b) \in \varrho \}$$

[2]If one considers the biologically interesting case of $V = \{A, C, G, T\}$, then ϱ is the relation given by the Watson-Crick-complementarity.

and consider this set as an alphabet. In the sequel the word $\begin{bmatrix} a_1 \\ b_1 \end{bmatrix} \begin{bmatrix} a_2 \\ b_2 \end{bmatrix} \ldots \begin{bmatrix} a_n \\ b_n \end{bmatrix}$ over $\begin{bmatrix} V \\ V \end{bmatrix}$ will often be written as $\begin{bmatrix} a_1 a_2 \ldots a_n \\ b_1 b_2 \ldots b_n \end{bmatrix}$.

The elements of $\begin{bmatrix} V \\ V \end{bmatrix}^*_\varrho$ describe the double strands where the upper and lower part are letter by letter in the relation ϱ. In order to describe the overhangs we set

$$\binom{V^*}{\lambda} = \{\binom{w}{\lambda} \mid w \in V^*\}$$

and

$$\binom{\lambda}{V^*} = \{\binom{\lambda}{w} \mid w \in V^*\}.$$

The elements of these sets describe single upper strands and single lower strands, respectively. Now we define

$$\begin{aligned}
L_\varrho(V) &= (\binom{V^*}{\lambda} \cup \binom{\lambda}{V^*}) \begin{bmatrix} V \\ V \end{bmatrix}^*_\varrho, \\
R_\varrho(V) &= \begin{bmatrix} V \\ V \end{bmatrix}^*_\varrho \binom{V^*}{\lambda} \cup \binom{\lambda}{V^*}), \\
LR_\varrho(V) &= (\binom{V^*}{\lambda} \cup \binom{\lambda}{V^*}) \begin{bmatrix} V \\ V \end{bmatrix}^+_\varrho (\binom{V^*}{\lambda} \cup \binom{\lambda}{V^*}), \\
W_\varrho(V) &= L_\varrho(V) \cup R_\varrho(V) \cup LR_\varrho(V).
\end{aligned}$$

Obviously, $L_\varrho(V)$, $R_\varrho(V)$, and $LR_\varrho(V)$ are constructs which describe double strands with overhangs to the left side, to the right side, and to both sides. The three strands given in the beginning of this section can be presented as

$$\begin{aligned}
&\binom{\mathtt{AAC}}{\lambda} \begin{bmatrix} \mathtt{GTAGCGATTT} \\ \mathtt{CATCGCTAAA} \end{bmatrix} \binom{\lambda}{\mathtt{CCGG}}, \\
&\binom{\mathtt{GGCC}}{\lambda} \begin{bmatrix} \mathtt{AATAGGA} \\ \mathtt{TTATCCCT} \end{bmatrix} \binom{\mathtt{AACC}}{\lambda} \\
&\binom{\mathtt{AAC}}{\lambda} \begin{bmatrix} \mathtt{GTAGCGATTTGGCCAATAGGA} \\ \mathtt{CATCGCTAAACCGGTTATCCCT} \end{bmatrix} \binom{\mathtt{AACC}}{\lambda}
\end{aligned}$$

We note that $\binom{w}{\lambda} \begin{bmatrix} w_1 \\ w_2 \end{bmatrix}$ cannot be written as (a pair) $\binom{ww_1}{w_2}$ since we loose the information which letters are in the relation ϱ.

Let $x \in LR_\varrho(V)$. Then x can be decomposed as $x = x_1 x_2 x_3$ with

$$x_2 \in \begin{bmatrix} V \\ V \end{bmatrix}^+_\varrho \quad \text{and} \quad x_1, x_3 \in (\binom{V^*}{\lambda} \cup \binom{\lambda}{V^*}). \tag{3.4}$$

Thus

$$x_1 = \binom{w_1}{\lambda} \text{ or } x_1 = \binom{\lambda}{w_1} \quad \text{and} \quad x_3 = \binom{w_3}{\lambda} \text{ or } x_3 = \binom{\lambda}{w_3}$$

for some $w_1 \in V^*$ and $w_3 \in V^*$. We define the delay of x by

$$d(x) = |w_1| + |w_3|.$$

The delay of a word is the sum of its overhangs to the right and to the left. Obviously, a delay can be defined for the elements of $L_\varrho(V)$ and $R_\varrho(V)$, too.

We define now the sticking operation $\mu_r : LR_\varrho(V) \times W_\varrho(V) \to LR_\varrho(V)$ which allows the prolongation of an element of $LR_\varrho(V)$ to the right by an element of $W_\varrho(V)$. Let $x \in W_\varrho(V)$ be decomposed as $x = x_1x_2x_3$ as in (**??**). Let $y \in W_\varrho(V)$. Then we define $\mu_r(x, y)$ as

1. $x_1x_2 \begin{bmatrix} u \\ v \end{bmatrix} y'$ if $x_3 = \binom{u}{\lambda}$ and $y = \binom{\lambda}{v} y'$ for some $u, v \in V^*$ and $y' \in R_\varrho(V)$,
2. $x_1x_2 \begin{bmatrix} u \\ v \end{bmatrix} y'$ if $x_3 = \binom{\lambda}{v}$ and $y = \binom{u}{\lambda} y'$ for some $u, v \in V^*$ and $y' \in R_\varrho(V)$,
3. $x_1x_2 \begin{bmatrix} u \\ v \end{bmatrix} \binom{u'}{\lambda}$ if $x_3 = \binom{uu'}{\lambda}$ and $y = \binom{\lambda}{v}$ for some $u, v, u' \in V^*$ and $y' \in R_\varrho(V)$,
4. $x_1x_2 \begin{bmatrix} u \\ v \end{bmatrix} \binom{\lambda}{v'}$ if $x_3 = \binom{u}{\lambda}$ and $y = \binom{\lambda}{vv'}$ for some $u, v, v' \in V^*$ and $y' \in R_\varrho(V)$,
5. $x_1x_2 \binom{uv}{\lambda}$ if $x_3 = \binom{u}{\lambda}$ and $y = \binom{v}{\lambda}$ for some $u, v \in V^*$,
6. $x_1x_2 \begin{bmatrix} v \\ u \end{bmatrix} \binom{\lambda}{u'}$ if $x_3 = \binom{\lambda}{uu'}$ and $y = \binom{v}{\lambda}$ for some $u, v, u' \in V^*$,
7. $x_1x_2 \begin{bmatrix} v \\ u \end{bmatrix} \binom{v'}{\lambda}$ if $x_3 = \binom{\lambda}{u}$ and $y = \binom{vv'}{\lambda}$ for some $u, v, v' \in V^*$,
8. $x_1x_2 \binom{uv}{\lambda}$ if $x_3 = \binom{\lambda}{u}$ and $y = \binom{\lambda}{v}$ for some $u, v \in V^*$.

The pictures in Figure 3.14 illustrate the Cases 1, 3 and 4. The reader may verify that the given cases record all possible cases of a continuation to the right (note that $x_3 = \binom{\lambda}{\lambda}$ is allowed).

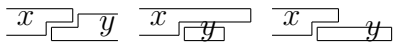

Figure 3.14: Pictorial representation of the operation μ_r in the Cases 1, 3 and 4.

Obviously, in an analogous way we can define the prolongation to the left by an operation μ_l. We omit the details.

Definition 3.40 *i) A* sticker system *is a quadruple $G = (V, \varrho, A, D)$ where*
– V is an alphabet,
– $\varrho \subset V \times V$ is a symmetric relation on V,
– A is a finite subset of $LR_\varrho(V)$, and
– D is a finite subset of $W_\varrho(V) \times W_\varrho(V)$.

ii) We say that $y \in LR_\varrho(V)$ is derived by $x \in LR_\varrho(V)$ in one step (written as $x \Longrightarrow y$) iff

$$y = \mu_l(\mu_r(x, y_2), y_1) \text{ for some } (y_1, y_2) \in D\,.$$

(Note that $\mu_l(\mu_r(x, y_2), y_1) = \mu_r(\mu_l(x, y_1), y_2)$ since the prolongation to the right and to the left are independent from each other.) By $\Longrightarrow^*$ we denote the reflexive and transitive closure of $\Longrightarrow$.

iii) The molecule language $ML(G)$ and the word language $wL(G)$ generated by G are defined by

$$ML(G) = \{z \mid x \Longrightarrow^* z,\ x \in A,\ z \in \begin{bmatrix} V \\ V \end{bmatrix}_{\varrho}^{+}\}$$

and

$$wL(G) = \{w \mid \begin{bmatrix} w \\ v \end{bmatrix} \in ML(G) \text{ for some } v \in V^+\}\,.$$

By definition the molecule language of G consists of all double strands without overhangs which can be obtained from the elements of A by simultaneous prolongations to the left and to the right by elements of D. If we restrict to the upper strand of the molecules of the molecule language, then we obtain the word language of G. Obviously, the upper strands can be obtained from the molecules by the homomorphism which maps $\begin{bmatrix} a \\ b \end{bmatrix}$ to a. Thus the word language of a sticker systems is a homomorphic images of its molecule language.

Example 3.41 We consider the sticker system

$$G = (\{a,b,c\}, \{(a,a),(b,b),(c,c)\}, \{\begin{bmatrix} a \\ a \end{bmatrix}\}, D)$$

where

$$D = \{(\begin{pmatrix} b \\ \lambda \end{pmatrix}, \begin{pmatrix} b \\ \lambda \end{pmatrix}), (\begin{pmatrix} c \\ \lambda \end{pmatrix}, \begin{pmatrix} \lambda \\ \lambda \end{pmatrix}), (\begin{pmatrix} \lambda \\ b \end{pmatrix}, \begin{pmatrix} \lambda \\ \lambda \end{pmatrix}), (\begin{pmatrix} \lambda \\ c \end{pmatrix}, \begin{pmatrix} \lambda \\ b \end{pmatrix})\,.$$

We are only interested in molecules in

$$M = ML(G) \cap \begin{bmatrix} c \\ c \end{bmatrix}^* \begin{bmatrix} b \\ b \end{bmatrix}^* \begin{bmatrix} a \\ a \end{bmatrix} \begin{bmatrix} b \\ b \end{bmatrix}^*\,.$$

Any word in M has a derivation of the following form

$$\begin{aligned}
\begin{bmatrix} a \\ a \end{bmatrix} &\Longrightarrow \begin{pmatrix} b \\ \lambda \end{pmatrix} \begin{bmatrix} a \\ a \end{bmatrix} \begin{pmatrix} b \\ \lambda \end{pmatrix} \Longrightarrow \begin{pmatrix} b^2 \\ \lambda \end{pmatrix} \begin{bmatrix} a \\ a \end{bmatrix} \begin{pmatrix} b^2 \\ \lambda \end{pmatrix} \Longrightarrow \ldots \Longrightarrow \begin{pmatrix} b^n \\ \lambda \end{pmatrix} \begin{bmatrix} a \\ a \end{bmatrix} \begin{pmatrix} b^n \\ \lambda \end{pmatrix} \\
&\Longrightarrow \begin{pmatrix} b^{n-1} \\ \lambda \end{pmatrix} \begin{bmatrix} ba \\ ba \end{bmatrix} \begin{pmatrix} b^n \\ \lambda \end{pmatrix} \Longrightarrow \begin{pmatrix} b^{n-2} \\ \lambda \end{pmatrix} \begin{bmatrix} b^2a \\ b^2a \end{bmatrix} \begin{pmatrix} b^n \\ \lambda \end{pmatrix} \Longrightarrow \ldots \Longrightarrow \begin{bmatrix} b^na \\ b^na \end{bmatrix} \begin{pmatrix} b^n \\ \lambda \end{pmatrix} \\
&\Longrightarrow \begin{pmatrix} c \\ \lambda \end{pmatrix} \begin{bmatrix} b^na \\ b^na \end{bmatrix} \begin{pmatrix} b^n \\ \lambda \end{pmatrix} \Longrightarrow \begin{pmatrix} c^2 \\ \lambda \end{pmatrix} \begin{bmatrix} b^na \\ b^na \end{bmatrix} \begin{pmatrix} b^n \\ \lambda \end{pmatrix} \Longrightarrow \ldots \Longrightarrow \begin{pmatrix} c^n \\ \lambda \end{pmatrix} \begin{bmatrix} b^na \\ b^na \end{bmatrix} \begin{pmatrix} b^n \\ \lambda \end{pmatrix} \\
&\Longrightarrow \begin{pmatrix} c^{n-1} \\ \lambda \end{pmatrix} \begin{bmatrix} cb^nab \\ cb^nab \end{bmatrix} \begin{pmatrix} b^{n-1} \\ \lambda \end{pmatrix} \Longrightarrow \begin{pmatrix} c^{n-2} \\ \lambda \end{pmatrix} \begin{bmatrix} c^2b^nab^2 \\ c^2b^nab^2 \end{bmatrix} \begin{pmatrix} b^{n-2} \\ \lambda \end{pmatrix} \Longrightarrow \ldots \\
&\Longrightarrow \begin{bmatrix} c^nb^nab^n \\ c^nb^nab^n \end{bmatrix}.
\end{aligned}$$

(first we add n times to the left as well as to the right a b in the upper strand, then we add n times to the left a b in the lower strand, then we add m times to the left a c in the upper strand, then we add m times simultaneously c to the left and b to the right in the lower strand; obviously, since we want to generate a double strand without overhangs $n = m$ has to hold). In a certain sense this derivation is the unique one for $\begin{bmatrix} c^nb^nab^n \\ c^nb^nab^n \end{bmatrix}$;

the only change which is allowed concerns the order of the generation of the letter c in the upper strand and of the letter b in the lower part, which have not to be generated in the sequence given above, it can also happen in a mixed form, but we have to generate n times c and n times b; also the application of $(\binom{\lambda}{c}, \binom{\lambda}{b})$ can be done earlier, if c is already present in the upper overhang to the left and all bs added to the left have already their counterpart in the lower strand.

Therefore we get for the word language

$$wL(G) = \{c^n b^n a b^n \mid n \geq 1\}.$$

It is easy to show (e.g. by a pumping lemma) that $wL(G)$ is not a context-free language.

We now present four special types of sticker systems or requirements to the derivations in the systems.

Definition 3.42 *i) A sticker system $G = (V, \varrho, A, D)$ is called*

- one-sided *if, for each pair $(u, v) \in D$, $u = \binom{\lambda}{\lambda}$ or $v = \binom{\lambda}{\lambda}$ hold,*
- regular*, if, for each pair $(u, v) \in D$, $u = \binom{\lambda}{\lambda}$ holds,*
- simple*, if, for each pair $(u, v) \in D$, $uv \in \binom{V^*}{\lambda}$ or $uv \in \binom{\lambda}{V^*}$ hold.*

Obviously, the sticker system given in Example 3.41 is not one-sided and not regular since it contains the element $(\binom{\lambda}{c}, \binom{\lambda}{b}$ in its set D. On the other hand, G of Example 3.41 is simple since we use for the prolongations only pairs which prolong to both sides only the upper strand or only the lower strand of the molecule.

From the definition it can be seen that regular sticker systems have an analogy to regular grammars since the molecule and the string can only be prolonged to the right, respectively.

Definition 3.43 *i) For a a sticker system $G = (V, \varrho, A, D)$ and a natural number $d \geq 1$, we define the language $ML_d(G)$ as the set of all molecules which have a derivation*

$$x = x_0 \Longrightarrow x_1 \Longrightarrow x_2 \Longrightarrow \ldots \Longrightarrow x_k \text{ with } x_k \in \begin{bmatrix} V \\ V \end{bmatrix}_{\varrho}^{*} \text{ with } d(x_i) \leq d \text{ for } 0 \leq i \leq k.$$

ii) We say that a molecule language $L \subset \begin{bmatrix} V \\ V \end{bmatrix}_{\varrho}^{*}$ or a word language $L' \subset V^*$ can be generated with bounded delay*, if there are a sticker system $G = (V, \varrho, A, D)$ and a natural number $d \geq 1$ such that $L = ML(G) = ML_d(G)$ and $L' = wL(G)$, respectively, are valid.*

The words of the language $ML_d(G)$ can be generated by a derivation where the length of the overhangs is bounded by d. If all words of $ML(G)$ can be generated by a derivation where the length of the overhangs is bounded, then $ML(G)$ is said to be a language with bounded delay.

We mention that the generation of $\begin{bmatrix} c \\ c \end{bmatrix}^n \begin{bmatrix} b \\ b \end{bmatrix}^n \begin{bmatrix} a \\ a \end{bmatrix} \begin{bmatrix} b \\ b \end{bmatrix}^n$ and $c^n b^n a b^n$ by the sticker system given in of Example 3.41 requires an overhang of length n to the right. This follows from the fact that the shortening of the right overhang is only possible if the sub-molecule $\begin{bmatrix} b \\ b \end{bmatrix}^n$

between the c-part and the a-part has already been produced. We shall see below that the languages generated in Example 3.41 cannot be derived with bounded delay since the word language is not context-free (see Theorem 3.52).

We denote the families of word languages generated by arbitrary sticker systems, one-sided sticker systems and regular sticker systems by ASL, OSL and RSL, respectively. If we allow only simple systems, we add the letter S before X with $X \in A, O, R\}$. Moreover, if we restrict to languages which can be generated by bounded delay, we add (b) after SL. Furthermore, we combine these restriction. Thus $ASL(b)$ and $SRSL$ are the families of languages which can be generated by arbitrary sticker systems with bounded delay and by regular simple sticker systems, respectively.

We now investigate the generative power of sticker systems. The first two statements follow directly from the definitions.

Lemma 3.44 *For $X \in \{A, O, R, SA, SO, SR\}$, $XSL(b) \subseteq XSL$.* □

Lemma 3.45 *For $y \in \{(b), \lambda\}$, the diagram given in Figure 3.15 is valid (if X and Y are connected by a line and Y has an upper position than X, then $X \subseteq Y$).* □

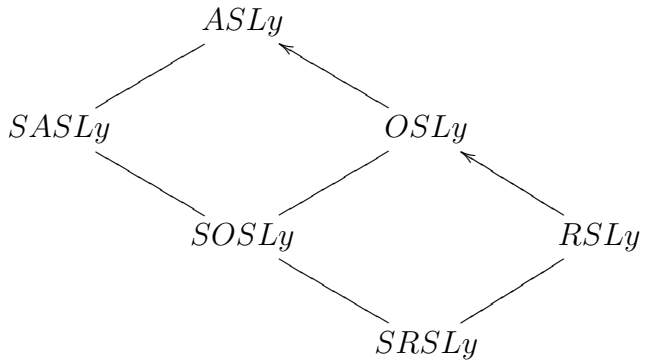

Figure 3.15: Hierarchy of (bounded) language families generated by sticker systems

Lemma 3.46 $ASL \subseteq \mathcal{L}(CS)$

Proof. Let $G = (V, \varrho, A, D)$ be a sticker system. We consider the alphabet consisting of all pairs (a, c) with $a, c \in V \cup \{\lambda\}$ and $(a, c) \in \varrho$ if a and c are both non-empty words. It is easy to construct a phrase structure grammar which simulates the sticking of x to the left and the sticking of y to the right where $(x, y) \in D$. Since no erasing is performed during the simulations, the grammar is a context-sensitive one. □

Lemma 3.47 $OSL \subseteq \mathcal{L}(REG)$

Proof. Let $G = (V, \varrho, A, D)$ be a sticker system. Let

$$d = \max\{d(x) \mid x \in A \text{ or } (x, u) \in D \text{ or } (u, x) \in D \text{ for some } u\}.$$

Now assume that there is a derivation of some molecule z with an upper overhang at the right end which is longer than d. This situation can only occur if in the last step some upper single strand has been added. Then the only elements of D which result in a prolongation to the right have to simple. If the are upper strands the delay is increased; if it is a lower strand, then the delay is decreased. Obviously, the order in which we apply the single strands can be arbitrarily chosen. Finally we have to reach a molecule without overhangs. Therefore we can choose the order of the simple adding in such a way that the overhang is always smaller than d. Thus any molecule can be generated by a derivation where all intermediate steps have a delay $\leq d$.

We now construct the context-free grammar $G' = (N, T, P, S)$ with

$$\begin{aligned} N &= \left\{ \left\langle \begin{matrix} u \\ \lambda \end{matrix} \right\rangle_l, \left\langle \begin{matrix} u \\ \lambda \end{matrix} \right\rangle_r, \left\langle \begin{matrix} \lambda \\ u \end{matrix} \right\rangle_l, \left\langle \begin{matrix} \lambda \\ u \end{matrix} \right\rangle_r \mid u \in V^*, 0 \leq |u| \leq d \right\} \cup \{S\}, \\ T &= \begin{bmatrix} V \\ V \end{bmatrix}_\varrho \end{aligned}$$

(the nonterminals store the existing overhang to the left or to the right),and P consisting of all rules of the form

$$S \to \left\langle \begin{matrix} u_1 \\ u_2 \end{matrix} \right\rangle_l \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \left\langle \begin{matrix} v_1 \\ v_2 \end{matrix} \right\rangle_r \text{ with } \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \begin{pmatrix} v_1 \\ v_2 \end{pmatrix} \in A$$

(by these rules we generate all elements of A),

$$\begin{aligned} \left\langle \begin{matrix} u_1 \\ u_2 \end{matrix} \right\rangle_l \to \left\langle \begin{matrix} u_1' \\ u_2' \end{matrix} \right\rangle_l \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} &\text{ such that } \begin{bmatrix} x_1 y_1 u_1 \\ x_2 y_2 u_2 \end{bmatrix} = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} \\ &\text{for some } \left(\begin{pmatrix} u_1' \\ u_2' \end{pmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \begin{pmatrix} y_1 \\ y_2 \end{pmatrix}, \begin{pmatrix} \lambda \\ \lambda \end{pmatrix} \right) \in D \end{aligned}$$

(if the left end is $\binom{u_1}{u_2}$ we add to the left the sticker $\binom{u_1'}{u_2'} \left[\begin{smallmatrix} y_1 \\ y_2 \end{smallmatrix}\right] \binom{y_1}{y_2}$ according to an element of D to the left end and get $\binom{u_1'}{u_2'} \left[\begin{smallmatrix} x_1 y_1 u_1 \\ x_2 y_2 u_2 \end{smallmatrix}\right]$),

$$\begin{aligned} \left\langle \begin{matrix} u_1 \\ u_2 \end{matrix} \right\rangle_r \to \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} \left\langle \begin{matrix} u_1' \\ u_2' \end{matrix} \right\rangle_r &\text{ such that } \begin{bmatrix} u_1 y_1 x_1 \\ u_2 y_2 x_2 \end{bmatrix} = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} \\ &\text{for some } \left(\begin{pmatrix} \lambda \\ \lambda \end{pmatrix}, \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} \begin{pmatrix} u_1' \\ u_2' \end{pmatrix} \right) \in D \end{aligned}$$

(we extend to the right),

$$\left\langle \begin{matrix} \lambda \\ \lambda \end{matrix} \right\rangle_l \to \lambda \left\langle \text{ and } \begin{matrix} \lambda \\ \lambda \end{matrix} \right\rangle_r \to \lambda$$

(if there is no overhang, then we finish the derivation).

It is easy to see that $L(G') = ML(G) = ML_d(G)$. In order to get the word language, we only consider the upper strands, which can be obtained by a homomorphism from $L(G')$. Hence $wL(G')$

Moreover, any derivation starts with a rule $S \to \left\langle \begin{smallmatrix} u_1 \\ u_2 \end{smallmatrix} \right\rangle_l \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \left\langle \begin{smallmatrix} v_1 \\ v_2 \end{smallmatrix} \right\rangle_r$ with $\begin{pmatrix} u_1 \\ u_2 \end{pmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \begin{pmatrix} v_1 \\ v_2 \end{pmatrix} \in A$. Then we extend from $\left\langle \begin{smallmatrix} u_1 \\ u_2 \end{smallmatrix} \right\rangle_l$ to the left by rules of the form $A \to Bz$ and from $\left\langle \begin{smallmatrix} v_1 \\ v_2 \end{smallmatrix} \right\rangle_r$ to the right by rules of the form $A \to zB$ where $w \in T^*$ and $A, B \in N$. Therefore $L(G')$ is a finite union of languages of the form $X\{w\}Y$ where X and Y are generated by rules of the form $A \to zB$ or $A \to Bz$ and $w \in T^+$. Hence X and Y are regular, which implies that all $X\{w\}Y$ and thus $L(G')$ are regular, too.

In order to get the word language, we only consider the upper strands, which can be obtained by a homomorphism from $L(G')$. By the closure properties of $\mathcal{L}(REG)$, $wL(G')$ is regular, too. □

Lemma 3.48 *$SOSL(b) = SOSL$ and $SRSL(b) = SRSL$.*

Proof. The statements follow immediately from the remarks in the beginning of the proof of Lemma 3.47. □

Lemma 3.49 *$\mathcal{L}(REG) \subseteq RSL(b)$.*

Proof. Let L be a regular language. Then $L = L(\mathcal{A})$ for some deterministic finite automaton $\mathcal{A} = (X, Z, z_1, F, \delta)$. Let $Z = \{z_1, z_2, \ldots, z_k\}$.

We construct a sticker system $G = (X, \varrho, A, D)$ with $\varrho = \{(a, a) \mid a \in X\}$ (because we are only interested in the word language it is sufficient to consider only molecules of the form $\begin{bmatrix} w \\ w \end{bmatrix}$ With any state z_j we associate the words $\begin{bmatrix} w \\ w \end{bmatrix} \begin{bmatrix} u \\ \lambda \end{bmatrix}$ with $|wu| = k+1$ and $|u| = j$. If w is a word of length $k+1$ and we want to remember a state z_j, then we choose x and u as the prefix and suffix of w of lengths $k+1-j$ and j, respectively. A word $z \in L$ can be written as $w = w_1 w_2 \ldots w_r$ where $|w_i| = k+1$ for $1 \leq i \leq r-1$ and $1 \leq |w_r| \leq k+1$. We consider the states $s_i = \delta(z_0, w_1 w_2 \ldots w_i) = \delta(s_{i-1}, z_i)$. By a partition of w_i by the above method into x_i and u_i to remember the state s_i.

We now define A and D by

$$\begin{aligned} A_1 &= \left\{ \begin{bmatrix} x \\ x \end{bmatrix} \mid x \in L,\ 0 \leq |x| \leq k+1 \right\}, \\ A_2 &= \left\{ \begin{bmatrix} y \\ y \end{bmatrix} \begin{pmatrix} u \\ \lambda \end{pmatrix} \mid |xu| = k+1, |u| = j, \delta(z_0, xu) = z_j \right\}, \\ A &= A_1 \cup A_2 \end{aligned}$$

(any word $x \in L$ of length at most $k+1$ is in $L(G)$ by $A_1 \subseteq L(G)$; otherwise we consider the prefix of the word, i.e., w_1 in the above notation and remember $z_j = s_1$),

$$\begin{aligned} D_1 &= \left\{ \begin{pmatrix} \lambda \\ \lambda \end{pmatrix}, \begin{pmatrix} \lambda \\ v \end{pmatrix} \begin{bmatrix} x \\ x \end{bmatrix} \begin{pmatrix} u \\ \lambda \end{pmatrix} \right) \mid |v| = j, |xu| = k+1, |u| = i, \delta(z_j, xu) = z_i) \}, \\ D_2 &= \left\{ \begin{pmatrix} \lambda \\ \lambda \end{pmatrix}, \begin{pmatrix} \lambda \\ v \end{pmatrix} \begin{bmatrix} x \\ x \end{bmatrix} \right) \mid |v| = j, 1 \leq |x| \leq k+1, \delta(z_j, x) \in F, \\ D &= D_1 \cup D_2 \end{aligned}$$

(by the rules of D_1, we extend the word by xu, which leads from the remembered state z_j to the state z_i which is stored by u of length i; by the rules of D_2, we read the last

subword of the partition we add the word without an overhang and stop the generation if an accepting state is reached; otherwise we have no applicable rule).

It is easy to see by theses explanations that $L = wL(G)$.

Obviously, G is a regular sticker system. Moreover, by our construction, all overhangs are bounded by k. Therefore, $ML(G) = ML_k(G)$ which shows that $wL(G)$ is of bounded delay. □

Lemma 3.50 *$ASL(b) = \mathcal{L}(LIN)$.*

Proof. We prove only $ASL(b) \subseteq \mathcal{L}(LIN)$ and refer to [23] for a proof of the converse inclusion.

In a sticker system, a word is generated by adding to the left and to the right elements of $W_\varrho(V)$, i.e., looking only on the upper strand a derivation has the form

$$z \Longrightarrow p_1 z q_1 \Longrightarrow p_2 p_1 z q_1 q_2 \Longrightarrow \ldots \Longrightarrow p_n p_{n-1} \ldots p_2 p_1 z q_1 q_2 \ldots q_{n-1} q_n. \tag{3.5}$$

In a linear grammar, the situation is opposite since a derivation has the form

$$\begin{aligned} S &\Longrightarrow p'_1 A_1 q'_1 \Longrightarrow p'_1 p'_2 A_2 q'_2 q'_1 \Longrightarrow \ldots \Longrightarrow p'_1 p'_2 \ldots p'_n A_n q'_n q'_{n-1} \ldots q'_1 \\ &\Longrightarrow p'_1 p'_2 \ldots p'_n A z' q'_n q'_{n-1} \ldots q'_1. \end{aligned}$$

Thus the idea of the linear is to start with the elements added in the last step of the generation in the sticker, to move "backwards" in the generation and to stop with a generation of an element of the start set of the sticker system.

We now give the formalization of this idea. Let $G = (V, \varrho, A, D)$ be a sticker system. Let $L = ML_d(G) = ML(G)$ for some constant d. Then the delays are bounded by d. We construct the linear grammar $G' = (N, T, P, S)$ with

$$\begin{aligned} N &= \{\langle \begin{pmatrix} u_1 \\ u_2 \end{pmatrix}, \begin{pmatrix} v_1 \\ v_2 \end{pmatrix} \rangle \mid \begin{pmatrix} u_1 \\ u_2 \end{pmatrix}, \begin{pmatrix} v_1 \\ v_2 \end{pmatrix} \in \begin{pmatrix} \lambda \\ V \end{pmatrix} \cup \begin{pmatrix} V \\ \lambda \end{pmatrix}, \ |u_1|, |u_2|, |v_1|, |v_2| \leq d\} \cup \{S\}, \\ T &= \begin{bmatrix} V \\ V \end{bmatrix}_\varrho, \end{aligned}$$

(by the nonterminals we store the overhangs by going from the outer part to the inner part and P consisting of all rules of the following forms

$$S \to \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} \langle \begin{pmatrix} u_1 \\ u_2 \end{pmatrix}, \begin{pmatrix} v_1 \\ v_2 \end{pmatrix} \rangle \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} \text{ where } (\begin{bmatrix} w_1 \\ w_2 \end{bmatrix} \begin{pmatrix} u_1 \\ u_2 \end{pmatrix}, \begin{pmatrix} v_1 \\ v_2 \end{pmatrix} \begin{bmatrix} z_1 \\ z_2 \end{bmatrix}) \in D$$

(we generate the outer elements p_n and q_n of the derivation (3.5),

$$\begin{aligned} \langle \begin{pmatrix} u_1 \\ u_2 \end{pmatrix}, \begin{pmatrix} v_1 \\ v_2 \end{pmatrix} \rangle \to \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} \langle \begin{pmatrix} u'_1 \\ u'_2 \end{pmatrix}, \begin{pmatrix} v'_1 \\ v'_2 \end{pmatrix} \rangle \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} \\ \text{where } \left(\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \begin{bmatrix} x'_1 \\ x'_2 \end{bmatrix} \begin{pmatrix} u'_1 \\ u'_2 \end{pmatrix}, \begin{pmatrix} v'_1 \\ v'_2 \end{pmatrix} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} \begin{pmatrix} y'_1 \\ y'_2 \end{pmatrix} \right) \in D, \\ \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} = \begin{bmatrix} u_1 x_1 x'_1 \\ u_2 x_2 x'_2 \end{bmatrix}, \ \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} = \begin{bmatrix} y_1 y'_1 v_1 \\ y_2 y'_2 v_2 \end{bmatrix}, \end{aligned}$$

(we proceed to the "middle"),

$$\langle \begin{pmatrix} u_1 \\ u_2 \end{pmatrix}, \begin{pmatrix} v_1 \\ v_2 \end{pmatrix} \rangle \to \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \end{bmatrix}$$
$$\text{where } \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \neq \begin{bmatrix} \lambda \\ \lambda \end{bmatrix}, \ \begin{pmatrix} w_1' \\ w_2' \end{pmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \begin{pmatrix} z_1' \\ z_2' \end{pmatrix} \in A, \ \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} = \begin{bmatrix} u_1 w_1' \\ u_2 w_2' \end{bmatrix}, \ \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} = \begin{bmatrix} z_1' v_1 \\ z_2' v_2 \end{bmatrix},$$

(if the overhangs of the nonterminal fit to some element of A, we finish the derivation),

$$S \to \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} \text{ where } \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} \in A$$

(we generate directly the elements from A which belong to the language $ML(G)$). By these explanations it is easy to see that $L(G') = wL(G)$. □

Lemma 3.51 *There is a regular language which is not in SOSL.*

Proof. We consider the language $L = \{b\}\{a\}^+\{b\}$, which is regular since it is given as a regular expression. Let us assume that $L = wL(G)$ for some simple one-sided sticker system $G = (V, \varrho, A, D)$. Because A is finite, and L is infinite, one needs upper and lower strands which can generate in the upper part an arbitrary number of as and the corresponding letter in the lower part, i.e., D contains at least one pair of one of the forms

$$\left(\begin{pmatrix} \lambda \\ \lambda \end{pmatrix}, \begin{pmatrix} y_1 \\ \lambda \end{pmatrix} \right) \text{ and } \left(\begin{pmatrix} \lambda \\ \lambda \end{pmatrix}, \begin{pmatrix} \lambda \\ y_2 \end{pmatrix} \right) \quad \text{or} \quad \left(\begin{pmatrix} y_1 \\ \lambda \end{pmatrix}, \begin{pmatrix} \lambda \\ \lambda \end{pmatrix} \right) \text{ and } \left(\begin{pmatrix} \lambda \\ y_2 \end{pmatrix}, \begin{pmatrix} \lambda \\ \lambda \end{pmatrix} \right)$$

with $y_1 \in \{a\}^+$ and $y_2 \in (V')^+$, where V' consists of all letters c with $(a, c) \in \varrho$.

We only discuss the first case; the other one can be handled analogously.

Let $|y_1| = k_1$ and $|y_2| = k_2$. Then, for ba^2b, we have a derivation

$$\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} \begin{pmatrix} z_1 \\ z_2 \end{pmatrix} \Longrightarrow^* \begin{bmatrix} ba^2b \\ w_1 \end{bmatrix}$$

for some $\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} \begin{pmatrix} z_1 \\ z_2 \end{pmatrix} \in A$ and some $w_2 \in (V')^+$. However, this molecule can be extended to the right by adding k_2 times $\begin{pmatrix} y_1 \\ \lambda \end{pmatrix}$ and k_1 times $\begin{pmatrix} \lambda \\ y_2 \end{pmatrix}$. Because we have $k_2|y_1| = k_1|y_2| = k_1k_2$, we get the result

$$\begin{bmatrix} ba^2b \\ w_1 \end{bmatrix} \begin{pmatrix} y_1 \\ \lambda \end{pmatrix}^{k-2} \begin{pmatrix} \lambda \\ y_2 \end{pmatrix}^{k_1} = \begin{bmatrix} ba^2ba^{k_1k_2} \\ w_2 \end{bmatrix}$$

for some $w_2 \in (V')^+$ of length $k_1k_2 + 4$. Hence $ba^2ba^{k_1k_2} \in wL(G)$, but $ba^2ba^{k_1k_2} \notin L$ in contrast to $L = wL(G)$. □

If we combine the Lemmas 3.44 – 3.51 and the fact that $SASL$ contains a non-context-free language by Example 3.41, we obtain the following hierarchy.

Theorem 3.52 *The diagram of Figure 3.16 holds (where an arrow $X \to Y$ is used for the proper inclusion $X \subset Y$; if X and Y are connected by a line and Y has an upper position than X, then $X \subseteq Y$).* □

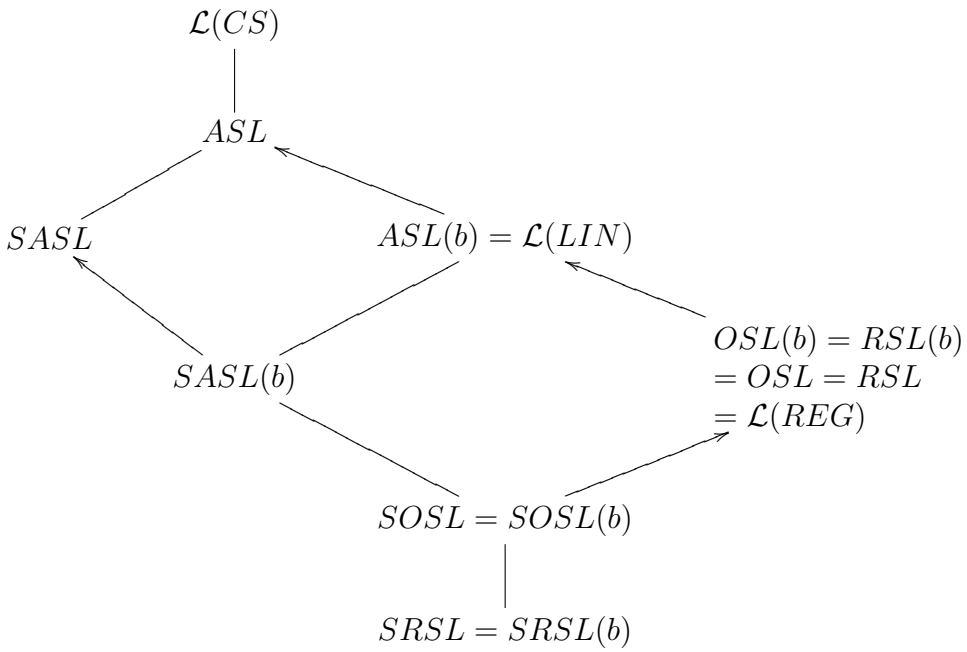

Figure 3.16: Hierarchy of language families generated by sticker systems