Contents

1 Fundamentals **9**
1.1 Sets and Multisets of Words . . . 9
1.2 Polynomials and Linear Algebra . . . 15
1.3 Graph Theory . . . 16
1.4 Intuitive Algorithms . . . 18

A SEQUENTIAL GRAMMARS **21**

2 Basic Families of Grammars and Languages **23**
2.1 Definitions and Examples . . . 23
2.2 Normal forms . . . 34
2.3 Iteration Theorems . . . 50

3 Languages as Accepted Sets of Words **57**
3.1 Turing Machines versus Phrase Structure Grammars . . . 57
3.1.1 Turing Machines and Their Accepted Languages . . . 57
3.1.2 Nondeterministic Turing Machines and Their Accepted Languages . 66
3.1.3 A Short Introduction to Computability and Complexity . . . 73
3.2 Finite Automata versus Regular Grammars . . . 80
3.3 Push-Down Automata versus Context-Free Languages . . . 87

4 Algebraic Properties of Language Families **95**
4.1 Closure Properties of Language Families . . . 95
4.2 Algebraic Characterizations of Language Families . . . 106
4.2.1 Characterizations of Language Families by Operations . . . 106
4.2.2 Characterizations of Regular Language Families by Congruence Relations . . . 115

5 Decision Problems for Formal Languages **119**

6 Descriptional Complexity of Grammars and Languages **129**
6.1 Measures of Descriptional Complexity . . . 129
6.2 Completeness of the Measures . . . 129
6.3 Comparison of Language Families with Respect to Descriptional Complexity 129
6.4 Decision Problems Related to Descriptional Complexity . . . 129

7 Some Special Properties of Context-Free Languages 131
7.1 Ambiguity of Context-Free Languages . . . 131
7.2 Semilinearity of Context-Free Languages . . . 131

B Formal Languages and Linguistics 133

8 Some Extensions of Context-Free Grammars 135
8.1 Families of Weakly Context-Sensitive Grammars . . . 135
8.2 Index Grammars . . . 135
8.3 Tree-Adjoining Grammars . . . 135
8.4 Head Grammars . . . 135
8.5 Comparison of Generative Power . . . 135

9 Contextual Grammars and Languages 137
9.1 Basic Families of Contextual Languages . . . 137
9.2 Maximally Locally Contextual Grammars . . . 137

10 Restart Automata 139

C Formal Languages and Biology 141

11 Lindenmayer Systems 143
11.1 The Basic Model – 0L Systems . . . 143
11.1.1 Two Biological Examples . . . 143
11.1.2 Definitions and Examples . . . 146
11.1.3 The Basic Hierarchy . . . 152
11.1.4 Adult Languages . . . 156
11.1.5 Decision Problems . . . 161
11.1.6 Growth Functions . . . 165
11.2 Lindenmayer Systems with Interaction . . . 171
11.2.1 Definitions and Examples . . . 171
11.2.2 Some Results on Lindenmayer Systems with Interaction . . . 176

12 Formal Languages and DNA Molecules 187
12.1 Basics from biology . . . 187
12.2 Adleman's experiment . . . 193
12.3 Splicing as an operation . . . 195
12.3.1 Non-iterated splicing . . . 195
12.3.2 Iterated splicing . . . 201
12.3.3 Remarks on descriptional complexity . . . 208
12.3.4 Splicing on Multisets . . . 212
12.4 Sticker Systeme . . . 218

13 Membrane Systems 231
13.1 Basic Membrane Systems and Their Power . . . 231
13.2 Membrane Systems with Symport/Antiport Rules . . . 231

D Formal Languages and Pictures **233**

14 Chain Code Picture Languages **235**
14.1 Chain Code Pictures 235
14.2 Hierarchy of Chain Code Picture Languages 243
14.3 Decision Problem for Chain Code Picture Languages 247
14.3.1 Classical Decision Problems 247
14.3.2 Decidability of Properties Related to Subpictures 257
14.3.3 Decidability of "Geometric" Properties 260
14.3.4 Stripe Languages 263
14.4 Some Generalizations 269
14.5 Lindenmayer Chain Code Picture Languages and Turtle Grammars 271
14.5.1 Definitions and some Theoretical Considerations 271
14.5.2 Applications for Simulations of Plant Developments 275
14.5.3 Space-Filling Curves 277
14.5.4 Kolam Pictures 280

15 Siromoney Matrix Grammars and Languages **283**
15.1 Definitions and Examples 285
15.2 Hierarchies of Siromoney Matrix Languages 290
15.3 Hierarchies of Siromoney Matrix Languages 290
15.4 Decision Problems for Siromoney Matrix Languages 293
15.4.1 Classical Problems 293
15.4.2 Decision Problems related to Submatrices and Subpictures 298
15.4.3 Decidability of geometric properties 302

16 Collage Grammars **309**
16.1 Collage Grammars 311
16.2 Collage Grammars with Chain Code Pictures as Parts 320

Bibliography **325**

Chapter 12

Formal Languages and DNA Molecules

12.1 Basics from biology

We do not want to give a precise introduction to DNA molecules from the biological and chemical point of view. We here only mention some facts which are important for the mutations and changes of DNA molecules and are the fundamentals for the operations with DNA strands to perform computations or to describe evolution.

The nucleotides which form the DNA strands are molecules that consist of a base - which is adenine, cytosine, guanine, or thymine - a sugar group and a phosphate group. Figure ?? gives the nucleotide with the thymine base. The left part is the thymine base and the right part gives two phosphate groups. In the sequel we shall denote the nucleotides by `A`, `C`, `G` and `T`, depending on its base adenine, cytosine, guanine, and thymine, respectively. The five carbon groups CH within the sugar group in the middle part are denoted by 1', 2', 3', 4' and 5'. One can see that groups 3' and 5' are connected to phosphate groups.

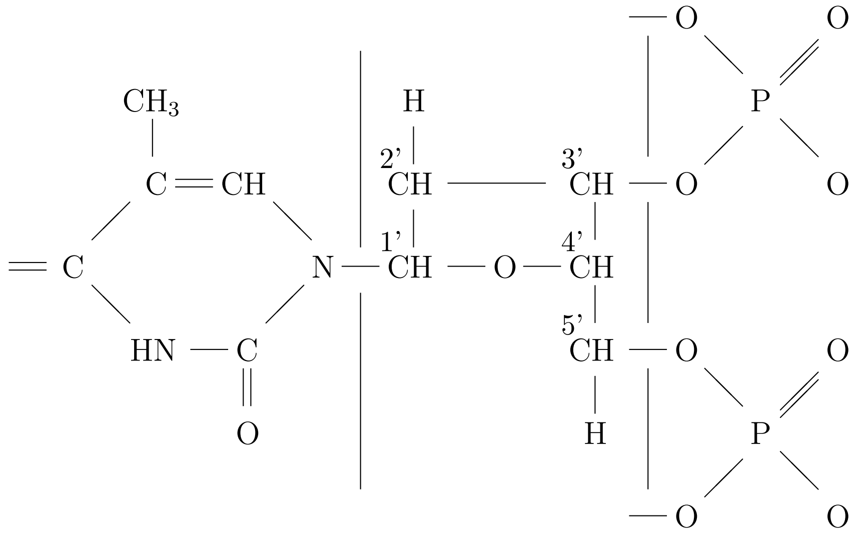


Figure 12.1: Diagram of a molecule with thymine base

Thus, the phosphate groups are able to link two bases. We note that one assumes that the connection is directed from the 5' part to the 3' part. Using some such links we get a sequence of connected bases which is called a single stranded DNA molecule. An example consisting of a thymine, a guanine and a cytosine group is shown in the upper part of Figure 12.2; the lower part shows the single strand formed by a guanine, a cytosine and an adenine group (note that we go from left to right in the upper part and from right to left in the lower part to ensure the direction from 5' to 3').

Moreover, the leftmost C in the thymine group in Figure 12.1 has two free bonds. The same holds for the adenine group. Therefore, the thymine group and the adenine group can be connected via hydrogen bonds (this is an attractive force between the hydrogen attached to an electronegative atom of a molecule and an electronegative atom of another molecule). Furthermore, the guanine group and the cytosine group have three free bonds each, and hence they can be connected, too. This possibility of pairing adenine with thymine (or thymine with adenine) and guanine with cytosine (or cytosine with guanine) is called the Watson-Crick complementarity.[1] Thus we get the molecule of the form shown in Figure 12.2. Such a molecule is a double stranded DNA molecule. However, we mention that Figure 12.2 only gives schematic presentation of a double stranded DNA molecule; in reality, the molecule is twisted in the three-dimensional space, i. e., it is far from the linear structure as given in Figure **??**.

We mention that the connection of the thymine and adenine group and guanine and cytosine group are very weak. They can already be destroyed by heating to approx. 90^0C. The link of the bases via the phosphate group is much stronger.

From the point of formal languages or words over an alphabet, a DNA molecule can be described as a word of pairs

$$\begin{matrix}\mathtt{A}\\\mathtt{T}\end{matrix} \text{ or } \begin{matrix}\mathtt{T}\\\mathtt{A}\end{matrix} \text{ or } \begin{matrix}\mathtt{C}\\\mathtt{G}\end{matrix} \text{ or } \begin{matrix}\mathtt{G}\\\mathtt{C}\end{matrix}$$

where we have written the components of the pair above each other. Obviously, the double stranded DNA is already completely determined if we only know one of its single stranded parts. By the Watson-Crick complementarity, the other single stranded molecule as well as the connections are uniquely determined. Thus, in many cases, it is sufficient to consider a single stranded DNA molecule which can be represented by a word over the alphabet $\{\mathtt{A},\mathtt{C},\mathtt{G},\mathtt{T}\}$.

First we give a method to extract DNA strands of a certain length from a set of DNA strands. We first produce a gel which is put into a rectangular container. Then along one side of the container we form some wells, e.g., by means of a comb. Then we fill a small amount of DNA strands into the wells and add a charge at the ends of the container. Since DNA strands are negatively charged they move through the gel from left to right. Obviously, the speed depends on the length of the strands. Therefore taking into account the duration and the place we can select strands of a certain length (see Figure 12.3).

We now come to some operations which change the DNA under consideration.

Figure 12.4 shows the polymerase, where in the direction from 5' to 3' we complete a partial double strand to a complete double strand. The transferase is an operation where we add in one strand in the direction from 5' to 3' further nucleotides.

[1]Other possible pairings are so weak that they have not be considered.

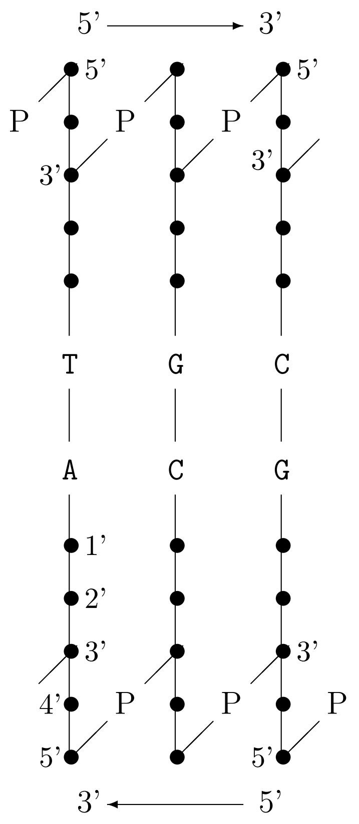


Figure 12.2: Diagram of a double stranded DNA molecule

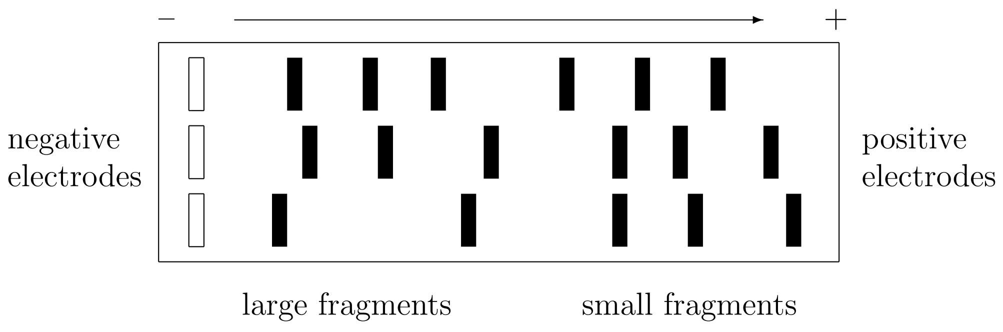


Figure 12.3: Measuring the length of DNA molecules by gel electrophoresis

An important operation is the polymerase chain reaction. One cycle consists of three steps. First we separate the bonds between the two strands by a heating to a temperature near to the boiling temperature (see upper part of Figure 12.5). Then we assume that in the solution are so-called primers which connect at appropriate positions by the Watson-Crick complementarity. For simplicity, in Figure 12.5, we use primers for the right end of the upper strand and the left end of the lower strand; in reality they can be somewhere in the strand. If we cool the solution, then the primers are connected with the corresponding ends (see the middle part of Figure 12.5). Finally, by a polymerase we can fill the missing parts and obtain two copies of the original DNA strand (see lower part of Figure 12.5).

This cycle can be iterated. After some cycles we have drastically increased the number of the strand we are interested in. Now there is a chance by some filtering to check whether this strand is contained in a solution or in a tube.

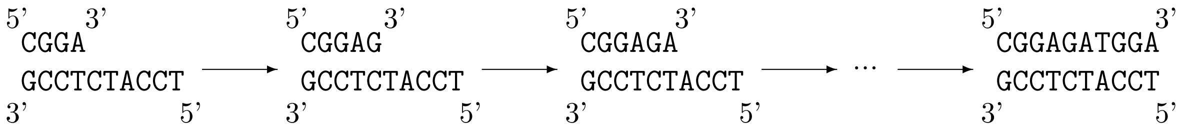


Figure 12.4: Polymerase

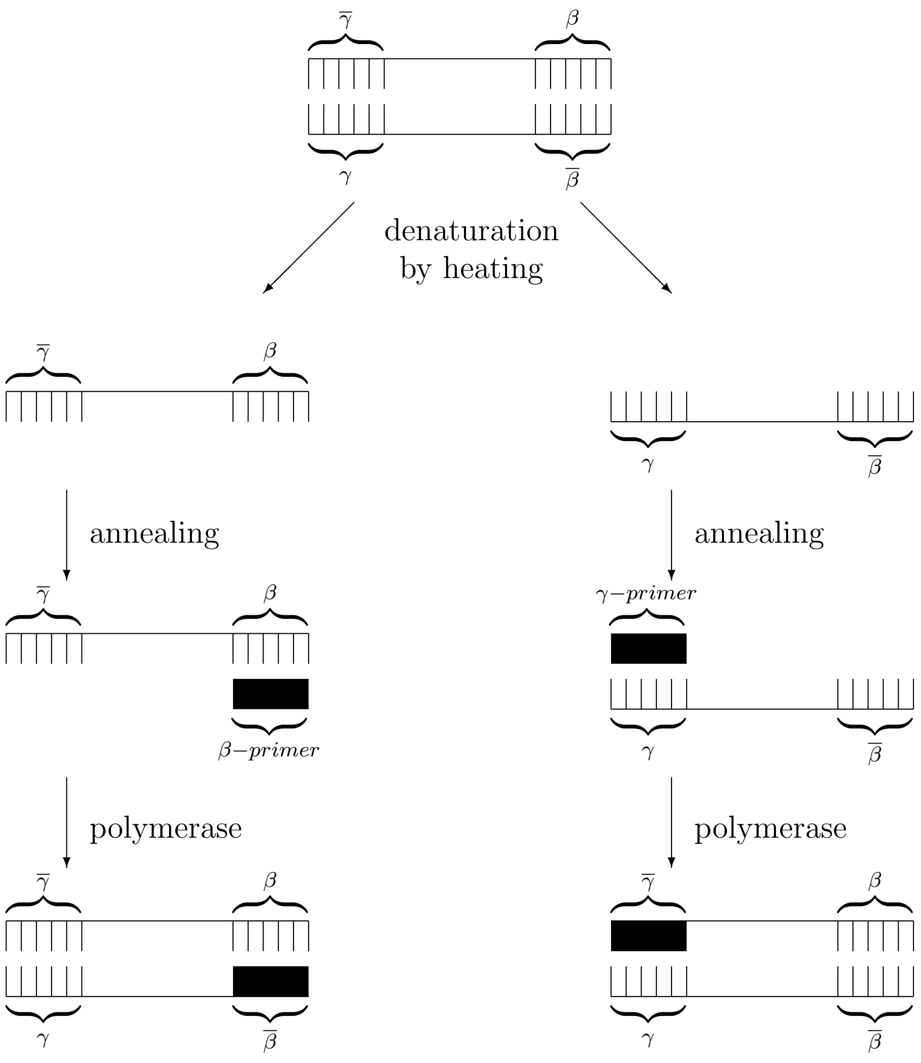


Figure 12.5: Polymerase chain reaction

We now consider the endonuclease which is an operation where the strand is cut at certain places. There are some enzymes which recognize a part of the strand and its direction and are able to cut the phosphodiester bond between some nucleotides.

In the left part of Figure 12.6 this procedure is shown for the restriction enzyme *NdeI* which is produced by the bacteria *Neisseria denitrificans.* It has the recognition site `CATATG` in the upper strand. If we take into consideration the direction, then the recognition site in the lower part is the same. The cut is performed after the first `A` in both strands (taking into consideration the direction). The bonds between both strands of the molecule are separated between the cuts. We obtain two new strands with some overhangs. In this case, we speak of so-called sticky ends.

The right part of Figure 12.6 shows the same procedure for the restriction enzyme *HaeIII* (isolated from the bacteria *Heamophilus aegyptius*) with the recognition site *GGCC*. The cut is performed after the second `G`. In this case we obtain so-called blunt ends.

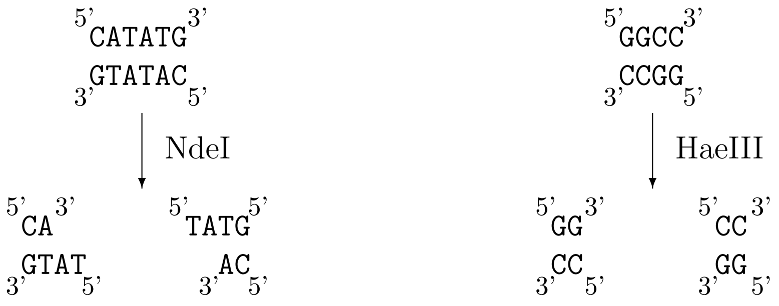


Figure 12.6: Endonuclease

The endonuclease can be reversed, i. e., intuitively the two double strands obtained by the endonuclease are again glued together which results in the original doubled stranded molecule. More formally, two steps are performed. First, a hydrogen bond connects the overhangs of two double strands according to the Watson-Crick complementarity. Then a ligase is done which connects the phosphate groups. For an illustration, see Figure 12.7.

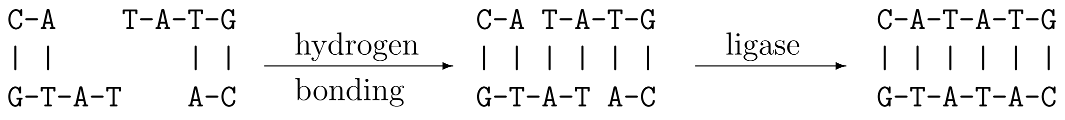


Figure 12.7: Hydrogen bonding and DNA ligase

Finally, we introduce the splicing operation. It consists of a endonuclease, which cuts two double strands according to two enzymes in such a way that the obtained overhangs are identical in both strands. Therefore we can glue them together by a hydrogen bonds and ligase after an exchange of the ends. Thus starting from two DNA strands we obtain two new DNA strands. Illustrations of the splicing operation with sticky and blunt ends are given in Figures 12.8 and 12.9, respectively.

In order to formalize the splicing operation we consider it in a more formal way. We set

$$\overline{\mathtt{A}} = \mathtt{T},\ \overline{\mathtt{C}} = \mathtt{G},\ \overline{\mathtt{G}} = \mathtt{C},\ \overline{\mathtt{T}} = \mathtt{A},$$

i. e., the overlined version of a is the letter which corresponds to a by the Watson-Crick complementarity. If $p = a_1 a_2 \ldots a_n$ is a word over $\{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{T}\}$ which represents the upper strand of a word, then we denote by $\overline{p} = \overline{a_1}\,\overline{a_2} \ldots \overline{a_n}$ the corresponding lower strand, where both strands are read from left to right. Let the two double strands

$$\frac{\alpha_1\, x_1\, y\, z_1\, \beta_1}{\alpha_1\, x_1\, y\, z_1\, \beta_1} \qquad \text{and} \qquad \frac{\alpha_2\, x_2\, y\, z_2\, \beta_2}{\alpha_2\, x_2\, y\, z_2\, \beta_2}$$

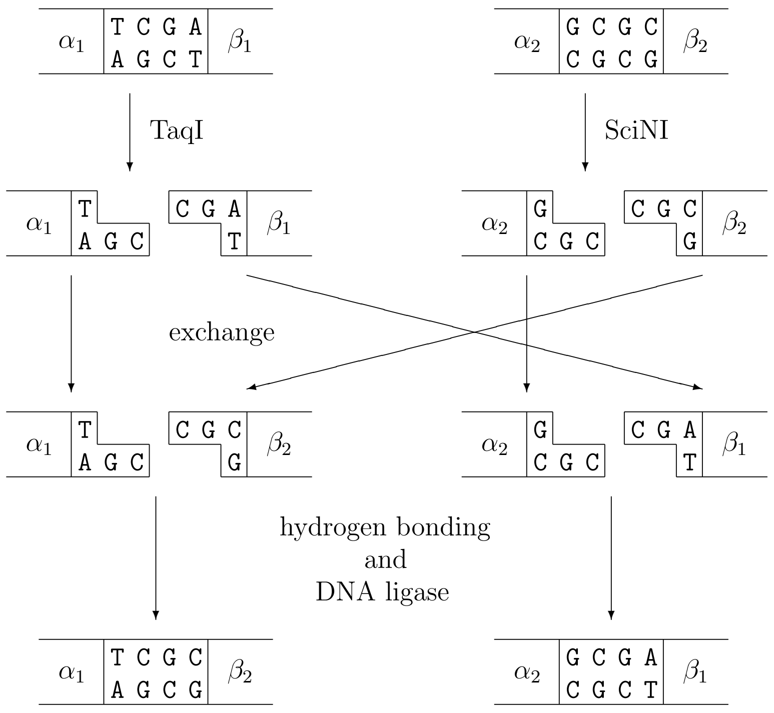


Figure 12.8: Splicing with sticky ends

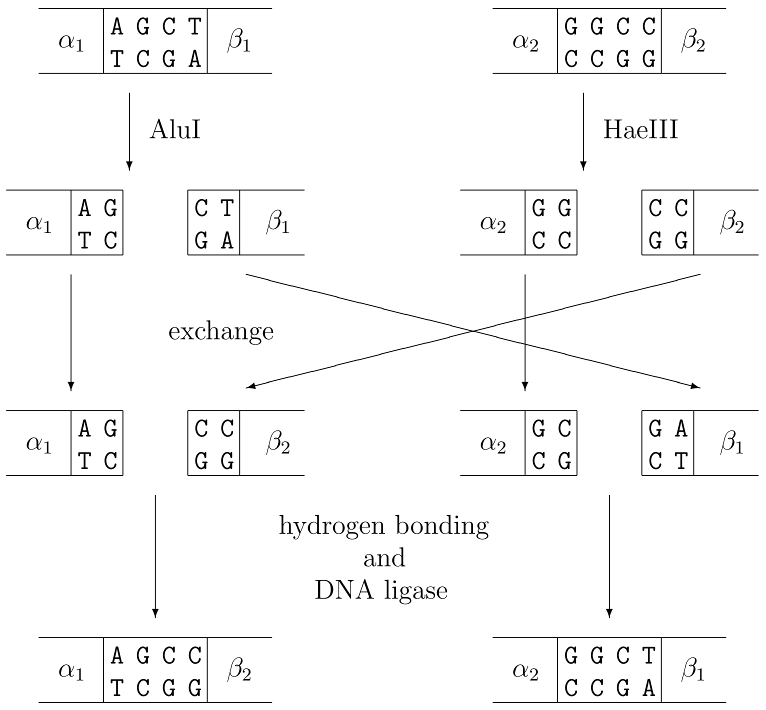


Figure 12.9: Splicing with blunt ends

with the recognition sites x_1yz_1 and x_2yz_2 in the upper strands and the common overhang y be given. If we have blunt ends, then $y = \lambda$ holds. Then the cutting of the two strands leads to

$$\frac{\alpha_1\, x_1}{\alpha_1\, x_1\, y} \qquad \frac{y\, z_1\, \beta_1}{z_1\, \beta_1} \qquad \text{and} \qquad \frac{\alpha_2\, x_2}{\alpha_2\, x_2\, y} \qquad \frac{y\, z_2\, \beta_2}{z_2\, \beta_2}$$

and the hydrogen bonds and ligases give

$$\frac{\alpha_1\, x_1\, y\, z_2\, \beta_2}{\alpha_1\, x_1\, y\, z_2\, \beta_2} \qquad \text{and} \qquad \frac{\alpha_2\, x_2\, y\, z_1\, \beta_1}{\alpha_2\, x_2\, y\, z_1\, \beta_1}.$$

Using the notation

$$u_1 = \frac{\alpha_1}{\alpha_1},\ r_1 = \frac{x_1}{x_1y},\ r_2 = \frac{yz_1}{z_1},\ u_2 = \frac{\beta_1}{\beta_1}, v_1 = \frac{\alpha_2}{\alpha_2},\ r_3 = \frac{x_2}{x_2y},\ r_4 = \frac{yz_2}{z_2},\ \text{and}\ v_2 = \frac{\beta_2}{\beta_2}$$

we get that the words

$$u_1r_1r_2u_2 \text{ and } v_1r_3r_4v_2 \text{ are transformed into } u_1r_1r_4v_2 \text{ and } v_1r_3r_2u_2. \qquad (12.1)$$

In the sequel, we shall use the latter variant to describe a splicing.

12.2 Adleman's experiment

In this section we shall demonstrate how one can solve non-biological problems by applying the operations considered in the preceding section. We partly follow the ideas by L. M. Adleman who was one of the first scientists solving a hard problem by easy calculations with DNA molecules.

We regard the Hamilton path problem. It requires to find a path in a graph which starts and ends in two given nodes and contains each node of the graph exactly once.

Let us consider the graph H shown in Figure 12.10. Obviously, H has a Hamiltonian path which starts in the node labelled by 0 and follows the labels of the nodes in their natural order (thus ending in the node labelled by 6).

By Theorem 3.41, we know that the Hamilton path problem is $\mathbf{N}P$-complete. Hence we cannot expect that there is an algorithm solving the Hamilton path problem in polynomial time by Turing machines (or by register machines or by a programming languages). Therefore the Hamilton path problem can be considered as a hard problem.

A very simple algorithm to find a Hamiltonian path in a graph G with n nodes or to find that there exists no Hamiltonian path in G consists of the following steps.

1. Construct all paths in G.
2. Take only paths of length n.
3. Take only paths starting in v_0 and ending in v_1.
4. Take only paths containing all nodes.

We now show how we can perform the steps 1. - 3. by means of DNA molecules.

For this purpose we model the nodes by single upper DNA strands of length 20 given in their 5'-3' orientation. For instance we choose

node labelled by 2 corresponds to `TATCGGATCGGTATATCCGA`,
node labelled by 3 corresponds to `GCTATTCGAGCTTAAAGCTA`,
node labelled by 4 corresponds to `GGCTAGGTACGAGCATGCTT`.

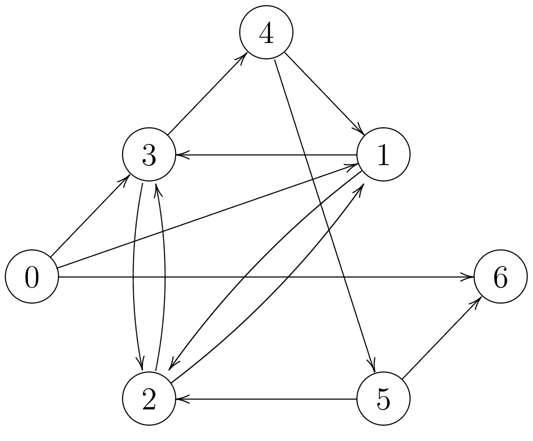


Figure 12.10: Graph whose Hamiltonian path problem is solved by DNA operations by Adleman

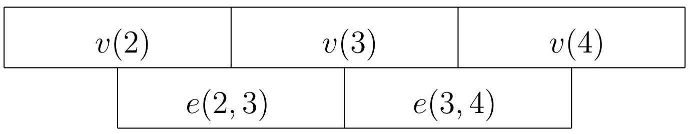
To model the edges we use single lower strands of length 20, too, in their 3'-5' orientation. Because we want to model edges we have to take into them information from the two nodes which are connected. One simple possibility is to take the Watson-Crick complementary of the second half of the strand modelling the start node of the edge and the first half of the end node of the edge. Thus we obtain that the

edge from 2 to 3 is modelled by `CATATAGGCTCGATAAGCTC`,
edge from 3 to 4 is modelled by `GAATTTCGATCCGATCCATG`.

Then by hydrogen bonding and ligase the following double stranded DNA molecule

```
TATCGGATCGGTATATCCGAGCTATTCGAGCTTAAAGCTAGGCTAGGTACGAGCATGCTT
                    CATATAGGCTCGATAAGCTCGAATTTCGATCCGATCCATG
```

can be build. Its structure is of the form



where $v(i)$ represent the node labelled by i and $e(i,j)$ represents the edge going from the node labelled by i to that labelled by j. This structure can be considered as a model of the path from 2 to 4 via 3.

Therefore we can build all paths if we put the models of nodes and edges in a tube. Thus we have performed Step 1 of the above algorithm.

The second step requires the filtering of strands with a certain length. This can be done by the method presented in the preceding section (see Figure 12.3).

In order to perform step 3 we can take the polymerase chain reaction by which we can produce a lot of molecules which start and stop with a certain sequence of DNA molecules. Then we can filter out those with this start and end sequence.

We do not discuss the methods to do the fourth step.

All together we can produce a tube which contains with high probability a molecule which represents a hamiltonian path, i. e., we can solve the Hamilton path problem by means of DNA molecules and operations on it.

However, two critical remarks are necessary. First, in order to get a probability which is very near to one, we need a very large number of molecules, at least much more molecules as we can put in a tube. Second, the execution of the steps by the methods given above takes some time; L. M. Adleman needs hours to solve the Hamilton path problem for the graph H of Figure 12.10, i. e., its solving by DNA structures takes more time than the solving by electronic computers.

On the other side, Adleman implemented its solving process by methods which only need a number of steps which is linear in the number of nodes. This contrasts the well-known fact that the Hamilton path problem is NP-complete (which means that we cannot expect an polynomial algorithm for this problem if we restrict to classical deterministic and sequential algorithms). Moreover, R. J. Lipton (see [21]) has presented a general method which allows a polynomial DNA computation for a lot of NP-complete problems. Therefore DNA computing can be considered as a method to solve hard problems in polynomial time (if we have fast implementations of the DNA operations).

Note that the existence of polynomial DNA algorithms for NP-complete problems is not surprising, since it is based on a parallelism since many molecules act in each step (for instance, in Step 1 of our algorithm we have determined all paths in parallel). We know that NP-complete problems can be solved in polynomial time by nondeterministic algorithms, where the all nondeterministic chosen paths are also handled in parallel.

12.3 Splicing as an operation

In Section 12.1 we have mentioned splicing as an operation which occurs in the development/evolution of DNA molecules. In this section, we formalize this operation and obtain an operation on words and languages. We study the power of the splicing operation on words, languages and language families.

12.3.1 Non-iterated splicing

We start with a formalization of the splicing such that it is an operation applicable to words and languages and allows a definition of a derivation and a device similar to grammars.

Definition 12.1 *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. Obviously, this can be done 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 12.2 *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 (12.1). The words r_1r_2 and r_3r_4 describe the recognition sites of the enzymes and the splitting can be done between r_1 and r_2 as well as between r_3 and r_4 (if we only consider the upper strand). Note that, in the case of sticky ends, r_2 and r_4 have to have a common non-empty prefix. This will not be required in the sequel, but one has to have it in mind, if one is interested in modelling splicing which occurs in biology.

We now give a slight modification of this formalization by emphasizing the getting of 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 12.3 *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,

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

Example 12.4 We consider the language $L = \{a^nb^n \mid n \geq 0\}$ 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^nb^n with $n \geq 1$. Let $u = a^nb^n$ and $v = a^mb^m$ be two arbitrary words from L with $m,n \geq 1$. Then we obtain

$$(a^nb^n, a^mb^m) = (a^{n-1}abb^{n-1}, a^{m-1}abb^{m-1}) \vdash_r a^nb^m.$$

Since n and m are arbitrary positive integers, we get

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

Example 12.5 For the splicing system $(\{a,b,c,c'\},R)$ with

$$R = \{ca^nb^n\#c'\$c'\# \mid n \geq 1\}$$

and the language

$$L = \{c\}\{a,b\}^+\{c'\},$$

we obtain

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

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

(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 if we restrict ourselves to words over $\{a,b,c\}$ or in $\{a,b,c\}^*\{c'\}$.)

Example 12.6 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 simple splicing is only possible if $u = wxc$ and $v = w'c$ for some words wx, $w' \in L$, and $x \in L'$. Finally, by the definition of the right quotient D_r,

$$spl(L\{c\},R) = D_r(L,L').$$

(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 12.7 We want to show that

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

or more precisely, that $L = \{a^nb^n \mid n \geq 1\}$ cannot be obtained from a regular set by (arbitrary) splicings. Note that, by Example 12.5, 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 $spl(K,R) = L$. By the pumping lemma for regular languages (see Theorem 2.31), there is a constant m such that any word $z \in K$ with $|z| \geq m$ has a decomposition $z = z_1z_2z_3$ with $|z_1z_2| \leq m$, $|z_2| > 0$, and $z_1z_2^iz_3 \in K$ for all $i \geq 0$.

By definition, there are words $u = u_1r_1r_2u_2$ and $v = v_1r_3r_4v_2$ and a splicing rule $r = r_1\#r_2\$r_3\#r_4 \in R$ such that

$$(u,v) \vdash_r = u_1r_1r_4v_2 = a^{m+1}b^{m+1}.$$

Obviously, $u_1r_1 = a^{m+1}z$ or $r_4v_2 = z'b^{m+1}$ for certain words z and z', respectively. We only discuss the former case; the latter one can be handled analogously. If we decompose u according to the pumping lemma, we get $u = z_1z_2z_3$ with $z_2 = a^t$ for some $t \geq 1$. Consequently,

$$u' = z_1z_2^2z_3 = a^{m+1+t}zr_2u_2 = a^tu_1r_1r_2u_2 \in K.$$

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 some language families from the Chomsky hierarchy and the family of finite languages.

Theorem 12.8 *The table of Figure 12.11 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 12.11: Relations for the families $spl(\mathcal{L}_1, \mathcal{L}_2)$

Theorem 12.8 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 $\{a^n b^n \mid n \geq 1\}$ (see Example 12.7) and by regular splicing rules we have almost no change of the family.

Before we give the proof of Theorem 12.8 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 12.9 *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 12.10 *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 12.11 *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

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

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

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

and the regular set

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

We get

$$L_4 = h_4(h_3^{-1}(R) \cap K') = \{u_1r_1\#r_2'u_2'\$v_1'r_3'\#r_4v_2 \mid u_1, u_2, v_1, v_2 \in V^*, r_1\#r_2\$r_3\#r_4 \in R\}.$$

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

Now we define the homomorphism

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

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 $u_1r_1r_4v_2$ and thus $h_5(L_3 \cap L_4) = spl(L, R) \in \mathcal{L}$. Therefore $spl(\mathcal{L}, \mathcal{L}(REG)) \subseteq \mathcal{L}$. □

Lemma 12.12 *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 a set $R \in \mathcal{L}$ of splicing rules, we construct the languages

$$L' = \{w_1\#w_2'\$w_3'\#w_4 \mid w_1w_2 \in L, w_3w_4 \in L\}$$

and

$$R' = \{u_1r_1\#r_2'u_2'\$v_1'r_3'\#r_4v_2 \mid u_1, u_2, v_1, v_2 \in V^*, r_1\#r_2\$r_3\#r_4 \in R\}$$

as in the proof of Lemma 12.11 and from these two sets $spl(L, R)$ which then belongs to $\mathcal{L}$. □

Proof of Theorem 12.8 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_1r_2 and r_3r_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 12.10 and 12.9, 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 12.10, 12.9, and 12.12, 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 12.9, 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 12.5 because $\{c\}\{a^nb^n \mid n \geq 1\}$ is not a regular language.

By the closure properties of $\mathcal{L}(CF)$ and $\mathcal{L}(RE)$ (see Section 4.1) and Lemma 12.12,

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

Moreover,

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

by Lemma 12.9. These inclusions are strict by Example 12.7.

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

By Lemma 4.26, for any recursively enumerable language L, there are context-free languages L_1 and L_2 such that $L = D_r(L_1, L_2)$. As in Example 12.6 we can prove that $L \in spl(\mathcal{L}(CF), \mathcal{L}(CF))$. Therefore we obtain

$$\mathcal{L}(RE) \subseteq spl(\mathcal{L}(CF), \mathcal{L}(CF)). \tag{12.2}$$

Furthermore,

$$spl(\mathcal{L}(RE), \mathcal{L}(RE)) \subseteq \mathcal{L}(RE) \tag{12.3}$$

can be proved by constructing a grammar which generates $spl(L, R)$ for given (recursively enumerable) languages L and R. (We omit a detailed construction. Informally, we first construct a grammar which generates $L\S L\S R$, where § is a new symbol which separates the words. If a word $w_1\S w_2\S r_1\#r_2\$r_3\#r_4$ is generated, we look for subwords r_1r_2 in w_1 and $r_3r_4$in w_2. In the affirmative case, the word is $u_1r_1r_2u_2\S v_1r_3r_4v_2\S r_1\#r_2\$r_3\#r_4$. By some cancellations we obtain the word $u_1r_1r_4v_2$. It is easy to see that the tasks can be solved by nonterminals moving in the word.)

For $X \in \{CF, CS, RE\}$, combining (12.2), (12.3), and Lemma 12.9 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 4.27, for any recursively enumerable language L, there is a context-sensitive language L' such that $L' \subseteq L\{c_1c_2^nc_3 \mid n \geq 0\}$, and for any $w \in L$, there is an n such that $wc_1c_2^nc_3 \in L'$. 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 $X \in \{FIN, REG, CF, CS, RE\}$. □

12.3.2 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 12.13 *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 12.14 *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 12.15 We consider the splicing system

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

By Example 12.4, we have

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

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

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

i. e., that the iteration does not increase the power (see Example 12.4).

The situation completely changes if we consider the splicing system

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

We obtain

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

By

$$(a^nb^ma^mb^m, a^rb^ra^rb^r) \vdash a^nb^ma^mb^r$$

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 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

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

results in

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

which is 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 $a^r b^{n_1} a^{n_1} b^{n_2} a^{n_2} \dots b^{n_s} a^{n_s} b^t$ with $s+1$ changes and $a^p b^m a^m b^q$ we get $a^r b^{n_1} a^{n_1} b^{n_2} a^{n_2} \dots b^{n_s} a^{n_s} b^m a^m b^q$ with $s+2$ changes.

Example 12.16 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 12.17 *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 12.18 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 12.16, 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 12.14 and 12.17 to language families.

Definition 12.19 *For two language families $\mathcal{L}_1$ and $\mathcal{L}_2$, we define the sets $Spl(\mathcal{L}_1, \mathcal{L}_2)$ and $ESpl(\mathcal{L}_1, \mathcal{L}_2))$ as the sets of all languages $L(G)$ generated by some splicing system $G = (V, R, A)$ and by some extended splicing system $G = (V, T, R, A))$ with $A \in \mathcal{L}_1$ and $R \in \mathcal{L}_2$, respectively.*

We now give the position of the sets $Spl(\mathcal{L}_1, \mathcal{L}_2)$ in the Chomsky hierarchy where $\mathcal{L}_1$ and $\mathcal{L}_2$ are some families of the Chomsky hierarchy or the family of finite languages.

Theorem 12.20 *The table of Figure 12.12 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 $\mathcal{L}(Z_1) \subset Spl(\mathcal{L}(X), \mathcal{L}(Y)) \subset \mathcal{L}(Z_2)$.*

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

We omit the proof of Theorem 12.20. 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 12.21 *The table of Figure 12.13 holds, where at the intersection of the row marked by X and the column marked by Y we give Z if $\mathcal{L}(Z) = ESpl(\mathcal{L}(X), \mathcal{L}(Y))$.*

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

Before giving the proof of Theorem 12.21 we present some lemmas which will be used in the proof.

The first lemma is the counterpart of Lemma 12.9 which follows from the definitions, again.

Lemma 12.22 *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 $ESpl(\mathcal{L}_1, \mathcal{L}_2) \subseteq ESpl(\mathcal{L}'_1, \mathcal{L}'_2)$.* $\square$

Lemma 12.23 *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 V. 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 12.24 $\mathcal{L}(REG) \subseteq Espl(\mathcal{L}(FIN), \mathcal{L}(FIN))$.

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 Theorem 2.28).

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:

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

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 12.25 *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 [10]. □

Lemma 12.26 *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

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

and R consists of all rules of the following forms:

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

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 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 12.27 *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 12.21 By Lemmas 12.22, 12.24 and 12.25, 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 12.23 and 12.25, 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 12.8 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 12.23) 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 12.22 and $\mathcal{L}(RE) = ESpl(\mathcal{L}(RE), \mathcal{L}(FIN))$ we get $\mathcal{L}(RE) = ESpl(\mathcal{L}(CS), \mathcal{L}(FIN))$.

Lemma 12.26 and 12.27 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 12.22, we obtain $\mathcal{L}(RE) = ESpl(\mathcal{L}(X), \mathcal{L}(Y))$ for $X \in \{FIN, REG, CF, CS, RE\}$ and $Y \in \{REG, CF, CS, RE\}$.

□

12.3.3 Remarks on descriptional complexity

In this section we study 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 12.28 *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*

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

ii) For a language family $\mathcal{L}$ and $n \ge 1$ 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 $r(G) \le n$ and $A \in \mathcal{L}$ and with $m(G) \le n$ 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 12.29 *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 12.23.

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

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

for which

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

hold (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 $u = u_1r_1r_2u_2 = a^{2n}b^{2n}a^r b^{2n}a^{2n}$ 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 and $r_4v_2 = z_2b^{2n}a^k b^{2n}a^{2n}$ for some $z_2 \in \{a\}^*$ and some $k \geq 2n + 1$. If we now apply p to u and v by splitting the suffix a^{2n} of u, we get

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

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 12.30 *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

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

with $c \notin V$. Then the splicing rule cannot be applied which yields $spl^i(G) = L$ and therefore $L(G) = L$. Hence

$$\mathcal{L} \subseteq \mathcal{L}_1(r, \mathcal{L}). \tag{12.4}$$

Furthermore, by definition and Theorem 12.20 we have

$$\mathcal{L}_n(r, \mathcal{L}) \subseteq \mathcal{L}_m(r, \mathcal{L}) \subseteq \mathcal{L}(\mathcal{L}, \mathcal{L}(FIN)) = \mathcal{L} \tag{12.5}$$

for $m \geq n$. From (12.4) and (12.5) 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 12.31 *For any $n \geq 1$, $\mathcal{L}_n(ea, \mathcal{L}(REG)) = \mathcal{L}(RE)$.*

Proof. By definition and Theorem 12.21, 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

$$\begin{array}{ll}
(u,u) \vdash c' & \text{using } \#c'c\$c\#c', \\
(u,c') \vdash c'cw_1cw_2c\dots cw_{i-1}cw_i = u_i & \text{using } \#c\$c'\#, \\
(c',u_i) \vdash w_i & \text{using } \#c'\$c\#.
\end{array}$$

Thus we have $w_i \in spl^3(G')$ for $1 \le i \le n$. 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 $r_1\#r_2\$r_3\#r_4 \in R$ 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 12.32 *For any $n \ge 2$, $\mathcal{L}_1(el, \mathcal{L}(REG)) \subset \mathcal{L}_n(el, \mathcal{L}(REG)) = \mathcal{L}(RE)$.*

Proof. By definition and Theorem 12.21, for any $n \ge 2$,

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

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 $A = \{w_1, w_2, \dots, w_n\}$. For any i, $1 \le i \le n$, 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

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

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

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

Therefore c_iw_i is obtained. We continue by

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

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. □

12.3.4 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

$$l([w, v]) = l([w', v']) \text{ and } \#_a([w, v]) = \#_a([w', v']). \tag{12.6}$$

Definition 12.33 *A* <u>*multiset splicing system*</u> *(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 $r_1\#r_2\$r_3\#r_4 \in P$, $r_i \neq \lambda$ for $1 \leq i \leq 4$, and
– M is a finite multiset over V.

Note that, in this section, we require that all words r_1, r_2, r_3, and r_4 occurring in splicing rules have to be non-empty. Especially, this implies that single letters cannot be spliced.

Definition 12.34 *For two multisets $M = [w_1, w_2, \dots, w_n]$ and $M' = [v_1, v_2, \dots, v_n]$ of words over V and a set P of splicing rules over V, we define*

- *a <u>sequential derivation step</u> $M \underset{s}{\Longrightarrow} M'$ 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 <u>maximally parallel derivation step</u> $M \underset{mp}{\Longrightarrow} M'$ 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 $[w, w'] \subseteq [w_{2k+1}, w_{2k+2}, \dots, w_n]$ to which a splicing rule $p \in P$ can be (successfully) applied,*
- *a <u>strongly maximally parallel derivation step</u> $M \underset{smp}{\Longrightarrow} M'$ by $M \underset{mp}{\Longrightarrow} M'$ for some k (as in the preceding item) and there is no M'' with $M \underset{mp}{\Longrightarrow} M''$ for some $k' > k$.*

As usually, by $\overset{*}{\underset{s}{\Longrightarrow}}$, $\overset{*}{\underset{mp}{\Longrightarrow}}$, and $\overset{*}{\underset{smp}{\Longrightarrow}}$, we denote the reflexive and transitive closures of $\underset{s}{\Longrightarrow}$, $\underset{mp}{\Longrightarrow}$, and $\underset{smp}{\Longrightarrow}$, respectively.

Definition 12.35 *We define the <u>sequential, maximally parallel and strongly maximally parallel multiset languages</u> $mL(G,s)$, $mL(G,mp)$ and $mL(G,smp)$ generated by G as*

$$\begin{aligned} mL(G,s) &= \{K \mid M \overset{*}{\underset{s}{\Longrightarrow}} K\}, \\ mL(G,mp) &= \{K \mid M \overset{*}{\underset{mp}{\Longrightarrow}} K\}, \\ mL(G,smp) &= \{K \mid M \overset{*}{\underset{smp}{\Longrightarrow}} K\}. \end{aligned}$$

Example 12.36 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

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

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. The only rule which can be applied to M_5 is r_1, and its application to M_5 yields M_7. 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. Since M_5 allows only the applicability of r_1 resulting in M_7, and no rule can be applied to M_6, M_7, and M_8, we get

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

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

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

Proof. The assertions immediately follow from (12.6) 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 12.38 *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 $L' \notin m\mathcal{L}_n(Y)$. $\square$

Lemma 12.39 *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 involved in a splicing rule in all the derivation modes which implies the statement.

ii) We give the proof for $n = 4$. In order to obtain a proof for $n \geq 5$ we proceed as follows. If L is a language with the desired properties and four words in the initial set and G is a multiset splicing system generating L, then we extend the alphabet by $n - 4$ letters $a_1, a_2, \ldots, a_{n-4}$ and add these letters to the initial multiset; since we cannot apply splicing rules to these new words in the initial set,

$$M' \in mL(G, Y) \text{ if and only if } M' \cup \{a_1, a_2, \ldots, a_{n-4}\} \in mL(G', Y)$$

for any $Y \in \{s, mp, smp\}$. Thus the non-inclusions for $n = 4$ also hold for $n \geq 4$.

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

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

Then

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

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, the derivation $[ab, ab, ba, ba] \underset{s}{\Longrightarrow} [aa, aa, bb, bb]$ is impossible as well as $[aa, aa, bb, bb] \underset{s}{\Longrightarrow} [ab, ab, ba, ba]$. Therefore $mL(H, s) \neq mL(G, mp)$ for any MSS H.

b) Now we show that neither $m\mathcal{L}_n(mp)$ nor $m\mathcal{L}_n(smp)$ contain $m\mathcal{L}_n(s)$. 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 $H = (\{a, b, c, d, e, f, g, h\}, P, M)$ and $Y \in \{mp, smp\}$.

We only discuss the case that $M = M_0$, we other cases lead analogously to contradictions. 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 $mL(G, s)$ cannot 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 two multisets M_i and M_j of multisets from M_1, M_2, M_3, and M_4 in such a way that $M_i \underset{smp}{\Longrightarrow} M_j$ or $M_j \underset{smp}{\Longrightarrow} M_i$ holds (in fact, this is true also for the relation $\underset{mp}{\Longrightarrow}$). 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. $M = M_0$. $M_0 \underset{mp}{\Longrightarrow} M_2$ 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 $M_0 \underset{mp}{\Longrightarrow} M_1$ 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 $M_0 \underset{mp}{\Longrightarrow} [ej, gh, if, kl, ej, if]$, 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 $M_2 \underset{mp}{\Longrightarrow} M_0$ is impossible, because obtaining ef would also yield gl, therefore $M_1 \underset{mp}{\Longrightarrow} M_0$ holds which means that P needs to contain the rules $e\#h\$g\#f, i\#l\$k\#j, e\#j\$i\#f$. However, then $M_1 \underset{mp}{\Longrightarrow} M_2$ is impossible since th rules above would also change the words ej, if. Since $M_0 \underset{mp}{\Longrightarrow} M_2$ 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 $M_2 \underset{mp}{\Longrightarrow} M_0$ is impossible, again, we need the derivation $M_1 \underset{mp}{\Longrightarrow} M_0$. As in Case 2 we can show that this implies that $M_2 \underset{mp}{\Longrightarrow} M_1$ 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.

12.4 Sticker Systeme

In this section the basic operation is polymerase, i.e., we glue together some parts of double strands according to the Watson-Crick complementarity; e. g. from the pieces

```
AACGTAGCGATTT                 GGCCAATAGGGAAACC
    CATCGCTAAACCGG      and        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

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

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

[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.

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

$$\begin{pmatrix} V^* \\ \lambda \end{pmatrix} = \{\begin{pmatrix} w \\ \lambda \end{pmatrix} \mid w \in V^*\}$$

and

$$\begin{pmatrix} \lambda \\ V^* \end{pmatrix} = \{\begin{pmatrix} \lambda \\ w \end{pmatrix} \mid w \in V^*\}\,.$$

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

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

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

$$\begin{array}{l} \begin{pmatrix} \mathtt{AAC} \\ \lambda \end{pmatrix} \begin{bmatrix} \mathtt{GTAGCGATTT} \\ \mathtt{CATCGCTAAA} \end{bmatrix} \begin{pmatrix} \lambda \\ \mathtt{CCGG} \end{pmatrix}, \\ \begin{pmatrix} \mathtt{GGCC} \\ \lambda \end{pmatrix} \begin{bmatrix} \mathtt{AATAGGGA} \\ \mathtt{TTATCCCT} \end{bmatrix} \begin{pmatrix} \mathtt{AACC} \\ \lambda \end{pmatrix} \\ \begin{pmatrix} \mathtt{AAC} \\ \lambda \end{pmatrix} \begin{bmatrix} \mathtt{GTAGCGATTTGGCCAATAGGGA} \\ \mathtt{CATCGCTAAACCGGTTATCCCT} \end{bmatrix} \begin{pmatrix} \mathtt{AACC} \\ \lambda \end{pmatrix} \end{array}$$

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

Let $x \in LR_\varrho(V)$. Then x can be decomposed as

$$x = x_1x_2x_3 \ \text{ with } \ x_2 \in \begin{bmatrix} V \\ V \end{bmatrix}_\varrho^+ \ \text{and } x_1, x_3 \in (\begin{pmatrix} V^* \\ \lambda \end{pmatrix} \cup \begin{pmatrix} \lambda \\ V^* \end{pmatrix})\,. \tag{12.7}$$

Thus

$$x_1 = \begin{pmatrix} w_1 \\ \lambda \end{pmatrix} \text{ or } x_1 = \begin{pmatrix} \lambda \\ w_1 \end{pmatrix} \quad \text{and} \quad x_3 = \begin{pmatrix} w_3 \\ \lambda \end{pmatrix} \text{ or } x_3 = \begin{pmatrix} \lambda \\ w_3 \end{pmatrix}$$

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 now define 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 $x \in LR_\varrho(V)$ be decomposed as $x = x_1x_2x_3$ as in (12.7). Let $y \in W_\varrho(V)$. Then we define $\mu_r(x, y)$ as

1. $x_1x_2 \left[\begin{smallmatrix} u \\ v \end{smallmatrix}\right] y'$ 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. $x_1x_2 \left[\begin{smallmatrix} u \\ v \end{smallmatrix}\right] y'$ 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. $x_1x_2 \left[\begin{smallmatrix} u \\ v \end{smallmatrix}\right] \binom{u'}{\lambda}$ 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. $x_1x_2 \left[\begin{smallmatrix} u \\ v \end{smallmatrix}\right] \binom{\lambda}{v'}$ 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. $x_1x_2 \left[\begin{smallmatrix} v \\ u \end{smallmatrix}\right] \binom{\lambda}{u'}$ if $x_3 = \binom{\lambda}{uu'}$ and $y = \binom{v}{\lambda}$ for some $u, v, u' \in V^*$,
7. $x_1x_2 \left[\begin{smallmatrix} v \\ u \end{smallmatrix}\right] \binom{v'}{\lambda}$ 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 12.14 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).

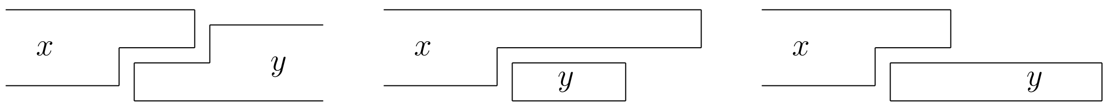


Figure 12.14: Pictorial representation of the operation μ_r in the Cases 1, 3 and 4.

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

Definition 12.40 *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 $\overset{*}{\Longrightarrow}$ 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

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

and

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

respectively.

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 12.41 We consider the sticker system

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

where

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

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 $\left[\begin{smallmatrix} c^n b^n a b^n \\ c^n b^n a b^n \end{smallmatrix}\right]$; 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

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

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 12.42 *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 12.41 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 12.41 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 12.43 *i) For 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 $L \subset \left[\begin{smallmatrix} V \\ V \end{smallmatrix}\right]_\varrho^*$ 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 $ML(G) = ML_d(G)$ and $L = ML(G)$ 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. Thus it is obvious that $ML_d(G) \subseteq ML(G)$. If all words of $ML(G)$ can be generated by a derivation where the length of the overhangs is bounded, then $ML(G) = ML_d(G)$.

We mention that the generation of $\left[\begin{smallmatrix} c \\ c \end{smallmatrix}\right]^n \left[\begin{smallmatrix} b \\ b \end{smallmatrix}\right]^n \left[\begin{smallmatrix} a \\ a \end{smallmatrix}\right] \left[\begin{smallmatrix} b \\ b \end{smallmatrix}\right]^n$ and $c^n b^n a b^n$ by the sticker system given in of Example 12.41 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 $\left[\begin{smallmatrix} b \\ b \end{smallmatrix}\right]^n$

between the c-part and the a-part has already been produced. We shall see below that the languages generated in Example 12.41 cannot be derived with bounded delay since the word language is not context-free (see Theorem 12.51).

We denote the families of word languages generated by arbitrary sticker systems, one-sided sticker systems and regular sticker systems by $\mathcal{L}(ASL)$, $\mathcal{L}(OSL)$ and $\mathcal{L}(RSL)$, respectively. If we allow only simple systems, we add the letter S before A, O, R, respectively. Moreover, if we restrict to languages which can be generated by bounded delay, we add (b) after SL. Furthermore, we combine these restriction. Thus $\mathcal{L}(ASL(b))$ and $\mathcal{L}(SRSL)$ 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 12.44 *For $y \in \{(b), \lambda\}$, the diagram given in Figure 12.15 is valid (if X and Y are connected by a line and Y has a position above X, then $X \subseteq Y$).* □

$$\begin{tikzcd}
 & \mathcal{L}(ASLy) \arrow[dl, dash] \arrow[dr, dash] & & \\
\mathcal{L}(SASLy) \arrow[dr, dash] & & \mathcal{L}(OSLy) \arrow[dl, dash] \arrow[dr, dash] & \\
 & \mathcal{L}(SOSLy \arrow[dr, dash] & & \mathcal{L}(RSLy) \arrow[dl, dash] \\
 & & \mathcal{L}(SRSLy) &
\end{tikzcd}$$

Figure 12.15: Hierarchy of (bounded) language families generated by sticker systems

Lemma 12.45 *For $X \in \{O, R, SO, SR\}$, $\mathcal{L}(XSL(b)) = \mathcal{L}(XSL)$.*

Proof. Let $G = (V, \varrho, A, D)$ be a (simple) one-sided or (simple) regular 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 elements of D which result in a prolongation to the right have to be simple. If there 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 addings 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 $\leq d$. Hence $ML(G) = ML_d(G)$.

Therefore any language generated by a (simple) one-sided or (simple) regular sticker system, is a language with bounded delay. This implies the statements. □

Lemma 12.46 *$\mathcal{L}(ASL) \subseteq \mathcal{L}(CS)$.*

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 12.47 *$\mathcal{L}(OSL) \subseteq \mathcal{L}(REG)$.*

Proof. Let $G = (V, \varrho, A, D)$ be a one-sided sticker system. By the proof of Lemma 12.45, $ML(G) = ML_d(G)$ for some 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 overhangs 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),

$$\begin{aligned} \left\langle \begin{matrix} u_1 \\ u_2 \end{matrix} \right\rangle_l &\to \left\langle \begin{matrix} u_1' \\ u_2' \end{matrix} \right\rangle_l \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} \text{ such that } \begin{bmatrix} x_1y_1u_1 \\ x_2y_2u_2 \end{bmatrix} = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} \\ &\qquad \text{for some } \left(\begin{pmatrix} u_1' \\ u_2' \end{pmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \begin{pmatrix} y_1 \\ y_2 \end{pmatrix}, \begin{pmatrix} \lambda \\ \lambda \end{pmatrix} \right) \in D \end{aligned}$$

(if the left end is $\binom{u_1}{u_2}$ we add to the left the sticker $\binom{u_1'}{u_2'} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} \binom{y_1}{y_2}$ according to an element of D to the left end and get $\binom{u_1'}{u_2'} \begin{bmatrix} x_1y_1u_1 \\ x_2y_2u_2 \end{bmatrix}$),

$$\begin{aligned} \left\langle \begin{matrix} u_1 \\ u_2 \end{matrix} \right\rangle_r &\to \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} \left\langle \begin{matrix} u_1' \\ u_2' \end{matrix} \right\rangle_r \text{ such that } \begin{bmatrix} u_1y_1x_1 \\ u_2y_2x_2 \end{bmatrix} = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} \\ &\qquad \text{for some } \left(\begin{pmatrix} \lambda \\ \lambda \end{pmatrix}, \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} \begin{pmatrix} u_1' \\ u_2' \end{pmatrix} \right) \in D \end{aligned}$$

(we extend to the right),

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

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

It is easy to see that $L(G') = ML(G) = ML_d(G)$.

Moreover, any derivation starts with a rule $S \to \left\langle {u_1 \atop u_2} \right\rangle_l \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \left\langle {v_1 \atop v_2} \right\rangle_r$ where $\binom{u_1}{u_2} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \binom{v_1}{v_2}$ is in A. Then we extend from $\left\langle {u_1 \atop u_2} \right\rangle_l$ to the left by rules of the form $A \to Bz$ and from $\left\langle {v_1 \atop v_2} \right\rangle_r$ 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 12.48 $\mathcal{L}(REG) \subseteq \mathcal{L}(RSL(b))$.

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 $Z = \{z_1, z_2, \dots, z_k\}$.

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 $\begin{bmatrix} w \\ w \end{bmatrix} \binom{u}{\lambda}$ with $|wu| = k + 1$ and $|u| = j$. Or conversely, 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 $w = w_1 w_2 \dots w_r$ where $|w_i| = k + 1$ for $1 \le i \le r - 1$ and $1 \le |w_r| \le k + 1$. We consider the states $s_i = \delta(z_0, w_1 w_2 \dots w_i) = \delta(s_{i-1}, w_i)$. By a partition of w_i into x_i and u_i as mentioned above we can remember the state s_i.

We now define A and D by

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

(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 $z_j = s_1$),

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

(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 12.49 $\mathcal{L}(ASL(b)) = \mathcal{L}(LIN)$.

Proof. We first prove that $\mathcal{L}(ASL(b)) \subseteq \mathcal{L}(LIN)$ holds.

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

$$z \Longrightarrow p_1 z q_1 \Longrightarrow p_2 p_1 z q_1 q_2 \Longrightarrow \ldots \Longrightarrow p_n p_{n-1} \ldots p_2 p_1 z q_1 q_2 \ldots q_{n-1} q_n. \tag{12.8}$$

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 grammar is to start with the elements added in the last step of the generation in the sticker system, 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 $L = ML_d(G)$ for some constant d. Then the delays are bounded by d. We construct the linear grammar $G' = (N, T, P, S)$ with

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

(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

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

(we generate the outer elements p_n and q_n of the derivation (12.8),

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

(we proceed to the "middle"),

$$\begin{aligned} &\left\langle \begin{pmatrix} u_1 \\ u_2 \end{pmatrix}, \begin{pmatrix} v_1 \\ v_2 \end{pmatrix} \right\rangle \to \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} \\ &\text{where } \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \neq \begin{bmatrix} \lambda \\ \lambda \end{bmatrix},\ \begin{pmatrix} w'_1 \\ w'_2 \end{pmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \begin{pmatrix} z'_1 \\ z'_2 \end{pmatrix} \in A,\ \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} = \begin{bmatrix} u_1 w'_1 \\ u_2 w'_2 \end{bmatrix},\ \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} = \begin{bmatrix} z'_1 v_1 \\ z'_2 v_2 \end{bmatrix}, \end{aligned}$$

(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 $L(G') = wL_d(G)$. □

Lemma 12.50 *There is a regular language which is not in $\mathcal{L}(SOSL)$.*

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

$$\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \begin{pmatrix} z_1 \\ z_2 \end{pmatrix} \Longrightarrow^* \begin{bmatrix} ba^2b \\ w_1 \end{bmatrix}$$

for some $\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \begin{pmatrix} z_1 \\ z_2 \end{pmatrix} \in A$ and some $w_1 \in (V'')^+$ where V'' consists of all letters c with $(a, c) \in varrho$ or $(b, c) \in varrho$. 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 12.44 – 12.50 and the fact that $\mathcal{L}(SASL)$ contains a non-context-free language by Example 12.41, we obtain the following hierarchy.

Theorem 12.51 *The diagram of Figure 12.16 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 $X \subseteq Y$).* □

By Theorem 12.51 simple regular (and simple one-sided) sticker systems are not able to generate all regular languages. However, this is possible, if we add to the generation process the application of a homomorphisms. More precisely, all regular languages can be obtain as the image of simple regular sticker languages under special homomorphisms, so-called codings.

A homomorphism $h : X^* \to Y^*$ is called a *coding*, if $h(x) \in Y$ holds for any $x \in X$. Thus a coding maps letters to letters.

$$\begin{tikzcd}
 & & \mathcal{L}(CS) & & \\
\mathcal{L}(ASL) \arrow[urr, dash] \arrow[d, dash] & & & & \mathcal{L}(ASL(b)) = \mathcal{L}(LIN) \arrow[ull] \arrow[d, dash] \\
\mathcal{L}(SASL) & & \begin{array}{c} \mathcal{L}(OSL) = \mathcal{L}(OSL(b)) \\ = \mathcal{L}(RSL) = \mathcal{L}(RSL(b)) \\ = \mathcal{L}(REG) \end{array} \arrow[ull] \arrow[urr] & & \mathcal{L}(SASL(b)) \\
 & & \mathcal{L}(SOSL) = \mathcal{L}(SOSL(b)) \arrow[ull] \arrow[u] \arrow[urr, dash] & & \\
 & & \mathcal{L}(SRSL) = \mathcal{L}(SRSL(b)) \arrow[u, dash] & &
\end{tikzcd}$$

Figure 12.16: Hierarchy of language families generated by sticker systems

Theorem 12.52 *For any regular language L, there are a simple regular sticker system G with bounded delay and a coding h such that $h(wL(G)) = L$.*

Proof. Let $G' = (N, T, P, S)$ be a regular grammar in normal form (i. e., any non-erasing rule has the form $A \to aB$ or $A \to a$ with $A, B \in N$ and $a \in T$) such that $L(G') = L$. We construct the sticker system $G = (V, \varrho, A, D)$ with

$$\begin{aligned}
V &= \{[X,a]_i \mid X \in N, a \in T, i \in \{1,2\}\}, \\
\varrho &= \{([X,a]_i, [X,a]_i) \mid X \in N, a \in T, i \in \{1,2\}\}, \\
A &= \left\{ \begin{bmatrix} [S,a]_1 \\ [S,a]_1 \end{bmatrix} \begin{pmatrix} \lambda \\ [X,a]_2 \end{pmatrix} \mid S \to aX \in P, X \to bY \in P \text{ for some } Y \text{ or } X \to b \in P \right\} \\
&\qquad \cup [S,a]_1 \left\{ \begin{bmatrix} [S,a]_1 \\ [S,a]_1 \end{bmatrix} \mid S \to a \in P \right\}, \\
D &= \left\{ \left(\begin{pmatrix} \lambda \\ \lambda \end{pmatrix}, \begin{pmatrix} \lambda \\ [X,a]_1[Y,b]_2 \end{pmatrix} \right) \mid X \to aY \in P, Y \to bZ \in P \text{ for some } Z \text{ or } Y \to b \in P \right\} \\
&\quad \cup \left\{ \left(\begin{pmatrix} \lambda \\ \lambda \end{pmatrix}, \begin{pmatrix} [X,a]_2[Y,b]_1 \\ \lambda \end{pmatrix} \right) \mid X \to aY \in P, Y \to bZ \in P \text{ for some } Z \text{ or } Y \to b \in P \right\} \\
&\quad \cup \left\{ \left(\begin{pmatrix} \lambda \\ \lambda \end{pmatrix}, \begin{pmatrix} \lambda \\ [X,a]_1 \end{pmatrix} \right) \mid X \to a \in P \right\} \cup \left\{ \left(\begin{pmatrix} \lambda \\ \lambda \end{pmatrix}, \begin{pmatrix} [X,a]_2 \\ \lambda \end{pmatrix} \right) \mid X \to a \in P \right\}.
\end{aligned}$$

Let $n \geq 2$ be an even natural number. It is easy to see that there is a derivation

$$S \Longrightarrow a_1A_1 \Longrightarrow a_1a_2A_2 \Longrightarrow \ldots \Longrightarrow a_1a_2\ldots a_{n-1}A_{n-1} \Longrightarrow a_1a_2\ldots a_{n-1}a_n$$

in G' if and only if there is a derivation

$$\begin{aligned}
\begin{bmatrix}[S,a_1]_1\\ [S,a_1]_1\end{bmatrix}\begin{pmatrix}\lambda\\ [A_1,a_2]_2\end{pmatrix} &\Longrightarrow \begin{bmatrix}[S,a_1]_1[A_1,a_2]_2\\ [S,a_1]_1[A_1,a_2]_2\end{bmatrix}\begin{pmatrix}[A_2,a_3]_1\\ \lambda\end{pmatrix}\\
&\Longrightarrow \begin{bmatrix}[S,a_1]_1[A_1,a_2]_2[A_2,a_3]_1\\ [S,a_1]_1[A_1,a_2]_2[A_2,a_3]_1\end{bmatrix}\begin{pmatrix}\lambda\\ [A_3,a_4]_2\end{pmatrix}\\
&\dots\\
&\Longrightarrow \begin{bmatrix}[S,a_1]_1[A_1,a_2]_2[A_2,a_3]_1\dots[A_{n-2},a_{n-1}]_1\\ [S,a_1]_1[A_1,a_2]_2[A_2,a_3]_1\dots[A_{n-2},a_{n-1}]_1\end{bmatrix}\begin{pmatrix}\lambda\\ [A_{n-1},a_n]_2\end{pmatrix}\\
&\Longrightarrow \begin{bmatrix}[S,a_1]_1[A_1,a_2]_2[A_2,a_3]_1\dots[A_{n-2},a_{n-1}]_1[A_{n-1},a_n]_2\\ [S,a_1]_1[A_1,a_2]_2[A_2,a_3]_1\dots[A_{n-2},a_{n-1}]_1[A_{n-1},a_n]_2\end{bmatrix}
\end{aligned}$$

in G. Moreover, the analogous situation holds for odd n (the difference is that the last overhang is then in the lower strand).

Moreover, we define the homomorphism h by $h([X,a]_i) = a$ for $X \in N$, $a \in T$, and $i \in \{1,2\}$. Obviously, we get $h(wL(G)) = L(G') = L$ which proves the statement. □

We note that we used only overhangs of length 1 in the proof of Theorem 12.52. However, the reason for this restriction comes from the homomorphism. If we do not apply homomorphisms, then we cannot restrict the delay.

For $X \in \{A,O,R\}$, we denote the family of languages which can be generated by an X sticker system with a delay $d' \leq d$ by $\mathcal{L}(XSL(d))$.

It is obvious that

$$\mathcal{L}(XSL(1)) \subseteq \mathcal{L}(XSL(2)) \subseteq \mathcal{L}(XSL(3)) \subseteq \dots \subseteq \mathcal{L}(XSL(d)) \subseteq \dots \qquad (12.9)$$

We now prove that the hierarchy (12.9) is infinite, i. e., there are infinitely many proper inclusions in (12.9).

Theorem 12.53 *For any natural number $n \geq 1$ and any $X \in \{A,O,R\}$, there is a number d such that $n \leq d$ and $\mathcal{L}(XSL(d)) \subset \mathcal{L}(XSL(d+1))$.*

Proof. We give the proof only for the case of regular sticker system. The proof for one-sided and arbitrary sticker systems can be presented analogously; we have to use Lemma 12.49 in the case of arbitrary systems.

Assume that the statement is not true. Then there exists a number t such that $\mathcal{L}(RSL(t)) = \mathcal{L}(RSL(b))$.

Let $L \subseteq X^*$ be a regular language where X has $n \geq 2$ letters. Then $L \in \mathcal{L}(RSL(b))$ by Theorem 12.51. By our assumption there is a regular sticker system G with a delay $d \leq t$. Now we construct the context-free regular grammar $G' = (N,X,P,S)$ from G as in the proof of Lemma 12.47. We note that N consists of all possible overhangs. Since there are only $s = \frac{n^{d+1}-1}{n-1}$ words of length $\leq d$ over X, G' has at most $4s$ nonterminals.

This implies that any regular language over X can be generated by a context-free grammar with less than $\frac{n^{t+1}-1}{n-1}$ nonterminals. This contradicts Theorem **??** of Section 6.2. □

We mention that we have only shown the infiniteness of the hierarchy (12.9) but we do not know where the proper inclusions are. Especially, we do not know whether all inclusions from (12.9) are proper.