Chapter 11

Formal Languages and DNA Molecules

11.1 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.

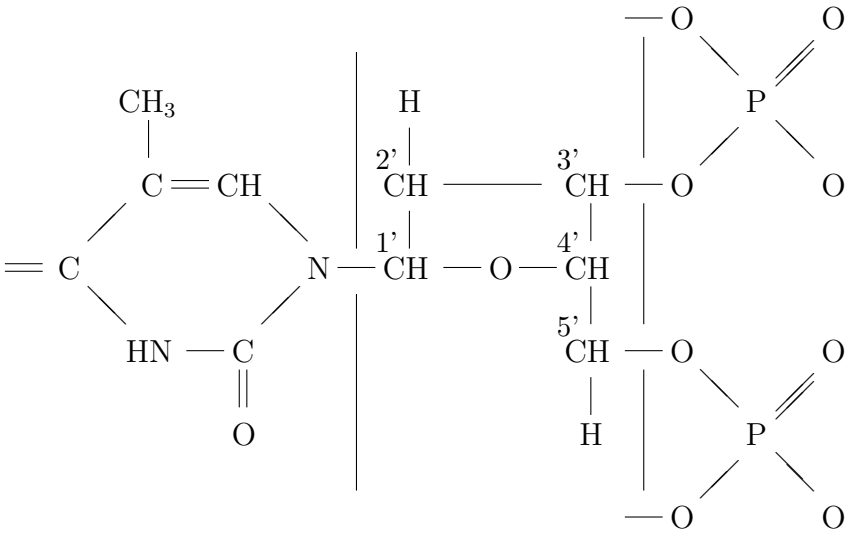


Figure 11.1: Diagram of a molecule with thymine base

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 11.2; 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 11.1 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 11.2. Such a molecule is a double stranded DNA molecule. However, we mention that Figure 11.2 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 appr. 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

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

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 $\{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{G}\}$.

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 charges 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 11.3).

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

Figure 11.4 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.

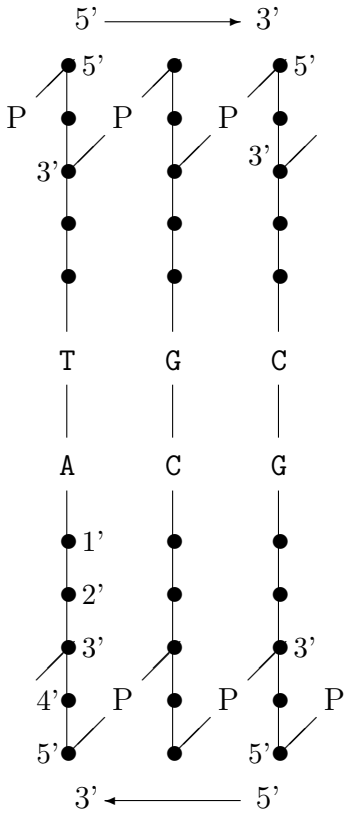


Figure 11.2: Diagram of a double stranded DNA molecule

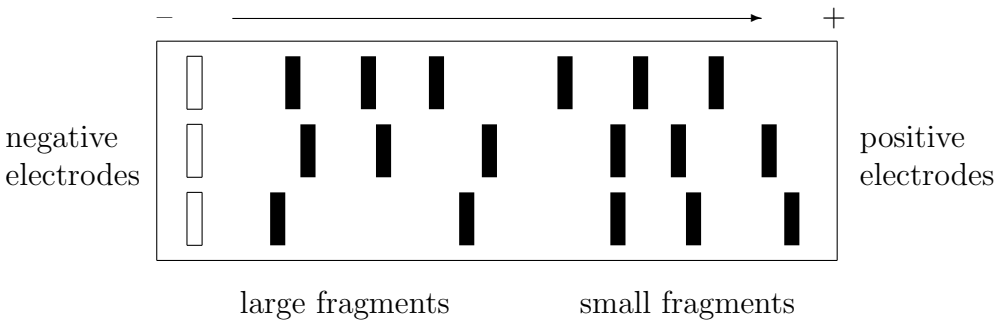


Figure 11.3: Measuring the length of DNA molecules by gel electrophoresis

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 11.5). 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 11.5, 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 11.5). Finally, by a polymerase we can fill the missing parts and obtain two copies of the original DNA strand (see lower part of Figure 11.5).

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.

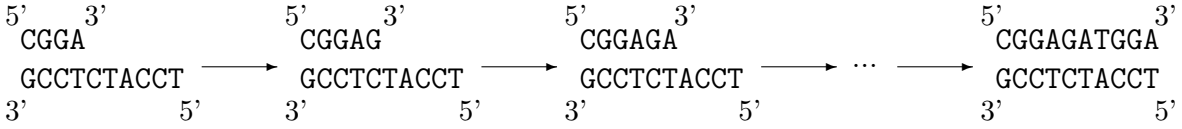


Figure 11.4: Polymerase

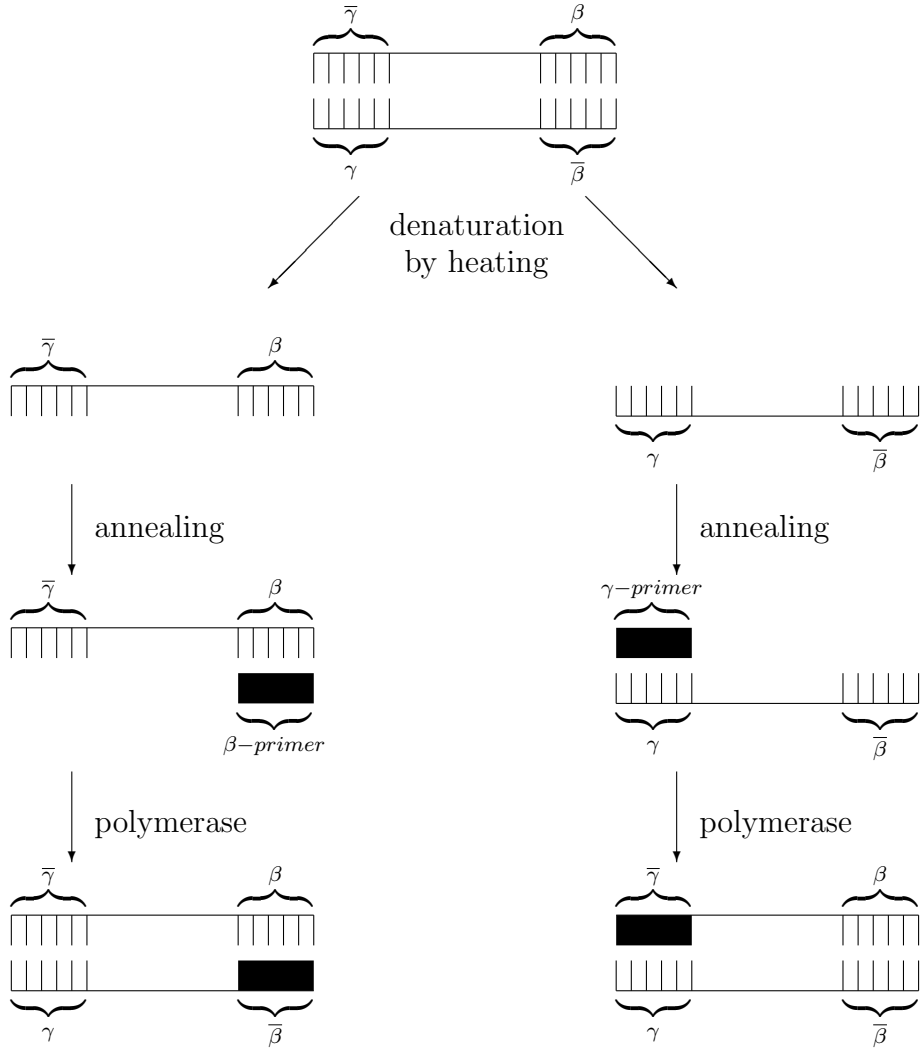


Figure 11.5: Polymerase chain reaction

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 11.6 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 11.6 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.

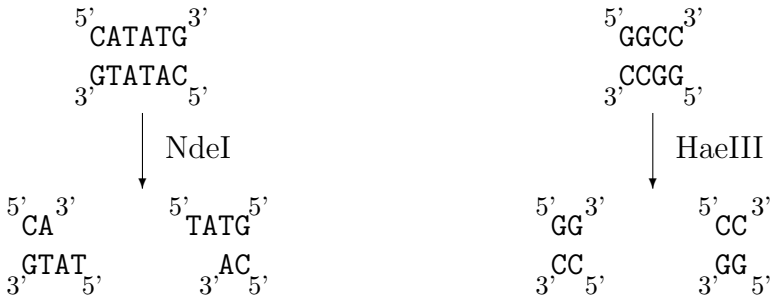


Figure 11.6: Endonuclease

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 11.7.

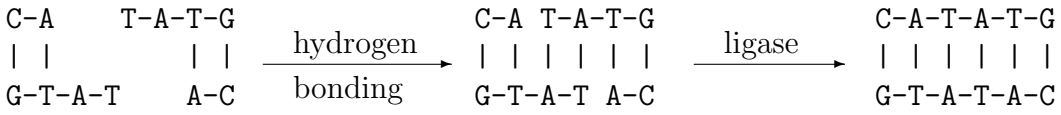


Figure 11.7: Hydrogen bonding and DNA ligase

Finally, we introduce the splicing operation. It consists of a endonuclease, which cuts two double strand 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 11.8 and 11.9, respectively.

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

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

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

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

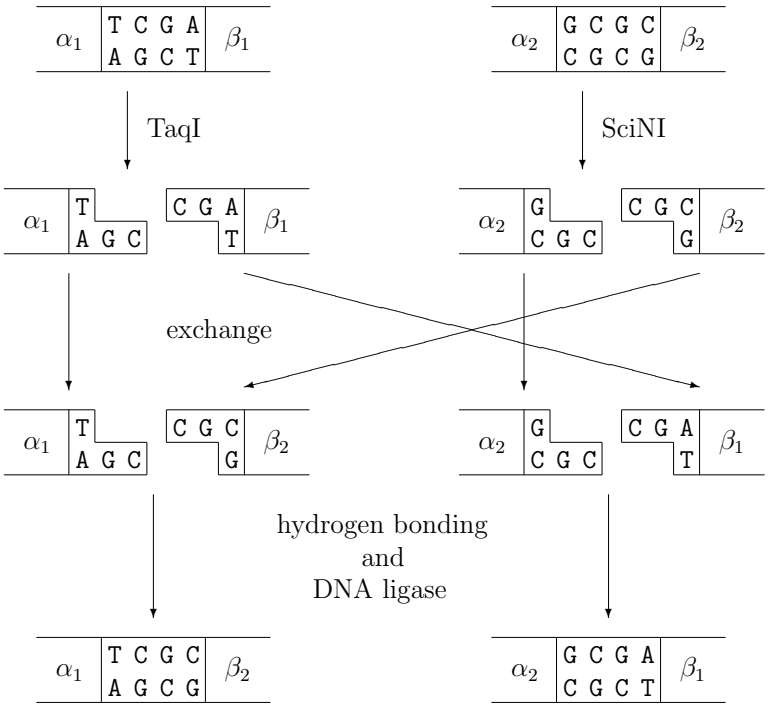


Figure 11.8: Splicing with sticky ends

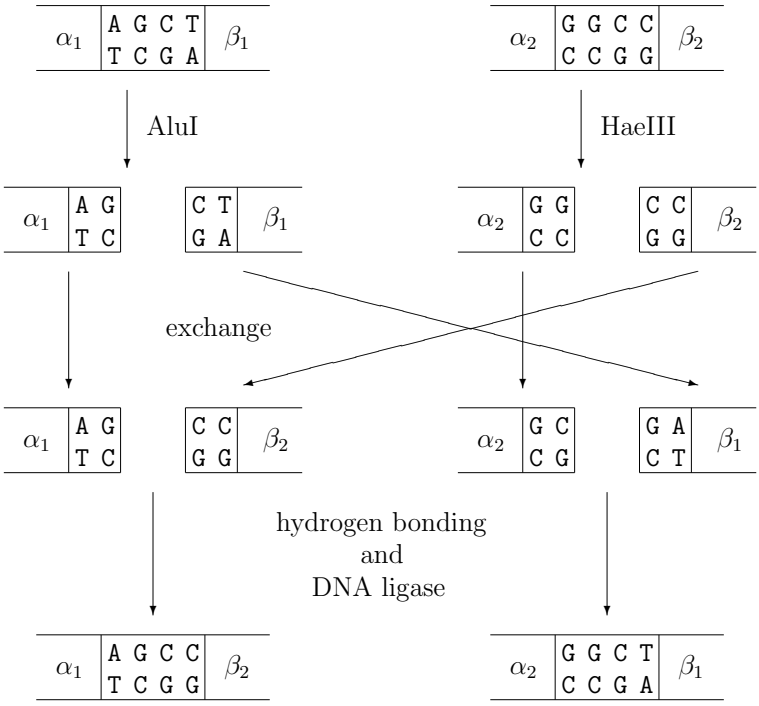


Figure 11.9: Splicing with blunt ends

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

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

we get that the words

$$u_1 r_1 r_2 u_2 \text{ and } v_1 r_3 r_4 v_2 \text{ are transformed into } u_1 r_1 r_4 v_2 \text{ and } v_1 r_3 r_2 u_2. \tag{11.1}$$

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

11.2 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 11.10. 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 $GGCTAGGTACGAGCATGCTT$.

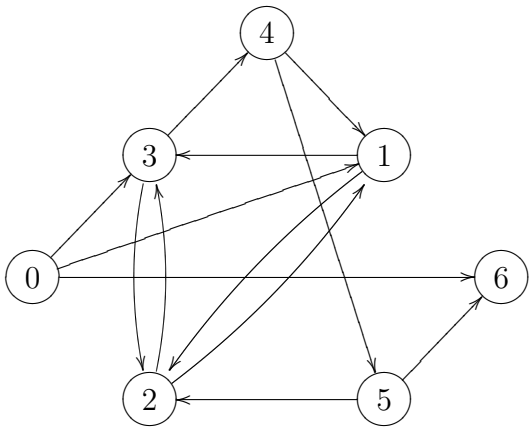


Figure 11.10: Graph whose Hamiltonian path problem is solved by DNA operations by Adleman

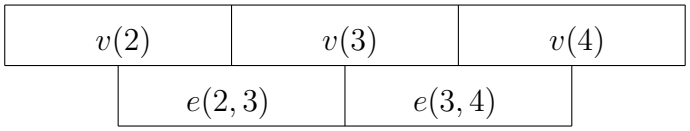
To model the edges we use single lower strands of length 20, too, in their 3'-5' orientation. Because we want to model edges we have to take into them information from the two nodes which are connected. One simple possibility is to take the Watson-Crick complementary of the second half of the strand 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 **??**).

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 11.10, 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 contrast 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. We know that NP-complete problems can be solved in polynomial time by nondeterministic algorithms.

11.3 Splicing as an operation

In Section **??** 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.

11.3.1 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 11.1 *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 11.2 *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 (11.1). The words r_1r_2 and r_3r_3 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 11.3 *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 11.4 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 11.5 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

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

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 $\{a, b, c\}$.)

Example 11.6 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 11.7 We want to show that

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

or more precisely, that $L = \{a^n b^n \mid n \geq 1\}$ cannot be obtained from a regular set by (arbitrary) splicings. Note that, by Example 11.5, 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 $z = z_1 z_2 z_3$ with $|z_1 z_2| \leq m$, $|z_2| > 0$, and $z_1 z_2^i z_3 \in K$ for all $i \geq 0$.

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

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

Obviously, $u_1 r_1 = a^{m+1} z$ or $r_4 v_2 = z' b^{m+1}$ 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 $u = z_1 z_2 z_3$ with $z_2 = a^t$ for some $t \geq 1$. Consequently,

$$u' = z_1 z_2^2 z_3 = a^{m+1+t} z r_2 u_2 = a^t u_1 r_1 r_2 u_2 \in K.$$

Thus

$$(u', v) = (a^t u_1 r_1 r_2 u_2, v_1 r_3 r_4 v_2) \vdash a^t u_1 r_1 r_4 v_2 = a^{t+m+1} b^{m+1}.$$

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 11.8 *The table of Figure 11.11 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 11.11: Relations for the families $spl(\mathcal{L}_1, \mathcal{L}_2)$

Theorem 11.8 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 11.7) and by regular splicing rules we have almost no change of the family.

Before we give the proof of Theorem 11.8 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 11.9 *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 11.10 *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 11.11 *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

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

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

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

and the regular set

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

We get

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

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 11.12 *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 11.11 and from these two sets $spl(L, R)$ which then belongs to $\mathcal{L}$. □

Proof of Theorem 11.8 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 11.10 and 11.9, 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 11.10, 11.9, and 11.12, 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 11.9, 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 11.5 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 **??**) and Lemma 11.12,

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

Moreover,

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

by Lemma 11.9. These inclusions are strict by Example 11.7.

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 **??**, 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 11.6 we can prove that $L \in spl(\mathcal{L}(CF), \mathcal{L}(CF))$. Therefore we obtain

$$\mathcal{L}(RE) \subseteq spl(\mathcal{L}(CF), \mathcal{L}(CF)). \tag{11.2}$$

Furthermore,

$$spl(\mathcal{L}(RE), \mathcal{L}(RE)) \subseteq \mathcal{L}(RE) \tag{11.3}$$

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 (11.2), (11.3), and Lemma 11.9 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 4.26, 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 $X \in \{FIN, REG, CF, CS, RE\}$. □

11.3.2 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 11.13 *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 11.14 *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 11.15 We consider the splicing system

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

By Example 11.4, we have

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

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

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

Therefore

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

i. e., that the iteration does not increase the power (see Example 11.4).

The situation completely changes if we consider the splicing system

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

We obtain

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

By

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

we have $a^nb^ma^mb^r \in spl^2(G)$, but $a^nb^ma^mb^r \notin spl^1(G)$.

We shall show that

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

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

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

results in

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

which 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^rb^t with $r, t \geq 1$. All these words are already in $spl^1(G')$.

From the words $a^rb^{n_1}a^{n_1}b^{n_2}a^{n_2}\ldots b^{n_s}a^{n_s}b^t$ with $s+1$ changes and $a^pb^ma^mb^q$ we get $a^rb^{n_1}a^{n_1}b^{n_2}a^{n_2}\ldots b^{n_s}a^{n_s}b^ma^mb^q$ with $s+2$ changes.

Example 11.16 Let

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

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

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

which implies

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

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 11.17 *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 11.18 Let

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

where $m \geq 1$ is a fixed number. From Example 11.16, we obtain

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

We now extend Definitions 11.14 and 11.17 to language families.

Definition 11.19 *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 11.20 *The table of Figure 11.12 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 11.12: Relations for the families $Spl(\mathcal{L}_1, \mathcal{L}_2)$

We omit the proof of Theorem 11.20. 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 11.21 *The table of Figure 11.13 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 11.13: Relations for the families $ESpl(\mathcal{L}_1, \mathcal{L}_2)$

Before giving the proof of Theorem 11.21 we present some lemmas which will be used in the proof.

The first lemma is the counterpart of Lemma 11.9 which follows from the definitions, again.

Lemma 11.22 *For any language families $\mathcal{L}_1, \mathcal{L}_2, \mathcal{L}'_1, \mathcal{L}'_2$ with $\mathcal{L}_1 \subseteq \mathcal{L}'_1$ and $\mathcal{L}_2 \subseteq \mathcal{L}'_2$, we have $ESpl(\mathcal{L}_1, \mathcal{L}_2) \subseteq ESpl(\mathcal{L}'_1, \mathcal{L}'_2)$.* □

Lemma 11.23 *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 11.24 $\mathcal{L}(REG) \subseteq Espl(\mathcal{L}(FIN), \mathcal{L}(FIN))$.

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:

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

This can be considered as a simulation of the derivation

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

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

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

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

Lemma 11.25 *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 11.26 *For any recursively enumerable language $L \subseteq T^*$, there is an extended splicing system $G = (V, T, R, A)$ with a finite set A and a regular set R of splicing rules such that $L(G) = L$.*

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

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

and R consists of all rules of the following forms:

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

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 11.27 *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 11.21 By Lemmas 11.22, 11.24 and 11.25, 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 11.23 and 11.25, 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 11.8 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 11.23) 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 11.22 and $\mathcal{L}(RE) = ESpl(\mathcal{L}(RE), \mathcal{L}(FIN))$ we get $\mathcal{L}(RE) = ESpl(\mathcal{L}(CS), \mathcal{L}(FIN))$.

Lemma 11.26 and 11.27 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 11.22, we obtain $\mathcal{L}(RE) = ESpl(\mathcal{L}(X), \mathcal{L}(Y))$ for $X \in \{FIN, REG, CF, CS, RE\}$ and $Y \in \{REG, CF, CS, RE\}$.

□

11.3.3 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 11.28 *i) For a splicing system $G = (V, R, A)$ or an extended splicing system $G = (V, T, R, A)$ we define the complexity measures $r(G)$, $a(G)$ and $l(G)$ by*

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

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

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

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

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

Theorem 11.29 *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 11.23.

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

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

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

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

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

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

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

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

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

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

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

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

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

Theorem 11.30 *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

$$\mathcal{L} \subseteq \mathcal{L}_1(r, \mathcal{L}). \tag{11.4}$$

Furthermore, by definition and Theorem 11.20 we have

$$\mathcal{L}_n(r, \mathcal{L}) \subseteq \mathcal{L}_m(r, \mathcal{L}) \subseteq \mathcal{L}(\mathcal{L}, \mathcal{L}(FIN)) = \mathcal{L} \tag{11.5}$$

for $m \geq n$. From (11.4) and (11.5) 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 11.31 *For any $n \geq 1$, $\mathcal{L}_n(ea, \mathcal{L}(REG)) = \mathcal{L}(RE)$.*

Proof. By definition and Theorem 11.21, 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

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

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

If we apply a rule $r_1\#r_2\$r_3\#r_4 \in R$ 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 11.32 *For any $n \geq 2$, $\mathcal{L}_1(el, \mathcal{L}(REG)) \subset \mathcal{L}_n(el, \mathcal{L}(REG)) = \mathcal{L}(RE)$.*

Proof. By definition and Theorem 11.21, for any $n \geq 2$,

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

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

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

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

with

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

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

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

Therefore $c_i w_i$ is obtained. We continue by

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

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

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

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