Contents

neighbours and its l right neighbours. Obviously, the first and last letters do not have k and l neighbours, respectively. Therefore we add a new letter \$ and prolongate the word by powers of \$ to the right and to the left such that any letter has k left and l right neighbours. Furthermore, we require a completeness condition to ensure that we have a rule for any situation which can occur. Then, for any letter in a word, we have k left and l right neighbours and a rule with respect to these neighbours. Again, the application of rules is a purely parallel process of rewritings.

Formally we get the following concepts.

Definition 2.32 *Let k and l be two non-negative integers. A $<k,l>$ Lindenmayer system ($<k,l>$L system for short) is a quadruple $G = (V, \$, P, \omega)$ where*

1. *V is an alphabet, and \$ is a symbol not occurring in V (used as an endmarker),*
2. *P is a finite set of quadruples (u, a, v, w) where*
   - *(a) $u = \$^r u'$ for some $r \in \mathbf{N}_0$ and some $u' \in V^*$ with $|u| = k - r$,*
   - *(b) $a \in V$,*
   - *(c) $v = v'\$^s$ for some $s \in \mathbf{N}_0$ and some $v' \in V^*$ with $|v| = l - s$,*
   - *(d) $w \in V^*$*

   and, for any triple (u, a, v) with the properties a), b) and c), there is a $w \in V^*$ such that $(u, a, v, w) \in P$.
3. *ω is a non-empty word over V.*

As usual we write $(u, a, v) \to w$ instead of (u, a, v, w). Moreover, if we consider a $<k, 0>$L or $<0, l>$L system, then we omit the non-existing context to the right or to the left, and write only $(u, a) \to w$ or $(a, v) \to w$, respectively.

Definition 2.33 *Let G be a $<k,l>$L system as in Definition 2.32.*

i) Let x be a non-empty word over V and $y \in V^*$. We say that x directly derives y (written as $x \Longrightarrow_G y$ or $x \Longrightarrow y$ if G is understood) if the following conditions are satisfied:

- *$x = a_1 a_2 \dots a_n$ with $a_i \in V$ for $1 \le i \le n$,*
- *$y = y_1 y_2 \dots y_n$,*
- *$(u_i, a_i, v_i) \to y_i \in P$ where*

$$u_i = \begin{cases} \$^{k-i+1} a_1 a_2 \dots a_{i-1} & \text{for } 1 \le i \le k \\ a_{i-k} a_{i-k+1} \dots a_{i-1} & \text{for } k < i \end{cases}$$

and

$$v_i = \begin{cases} a_{i+1} a_{i+2} \dots a_{i+l} & \text{for } i + l \le n \\ a_{i+1} a_{i+2} \dots a_n \$^{l+i-n} & \text{for } n < i + l \end{cases}$$

ii) The language $L(G)$ generated by G is defined as

$$L(G) = \{z \mid \omega \Longrightarrow_G^* z\}$$

where $\Longrightarrow_G^*$ denotes the reflexive and transitive closure of $\Longrightarrow_G$.

Example 2.34 We consider the $< 1, 0 >$L system $G_7 = (\{a, b, c\}, \$, P_7, c)$ with

$$\begin{aligned} P_7 &= \{(\$, a) \to a^2, (\$, b) \to b, (\$, c) \to a, (\$, c) \to ba^2, (a, a) \to a^2\} \\ &\quad \cup\{(p, q) \to q \mid (p, q) \in \{a, b, c\} \times \{a, b, c\} \setminus \{(a, a)\}\}\,. \end{aligned}$$

First we have the derivations $c \Longrightarrow a$ and $c \Longrightarrow ba^2$. If we have a word a^n, then any letters is doubled according to the rules, which leads to a^{2n}. Starting from a we get all words a^{2^n} for $n \geq 0$. If we have a word ba^m with $m \geq 1$, then we replace b by b, the first a by a, and all remaining a's by a^2. Thus we get

$$ba^{2^n+1} = baa^{2^n} \Longrightarrow baa^{2^{n+1}} = ba^{2^{n+1}+1}\,.$$

Therefore we obtain

$$L(G_7) = \{c\} \cup \{a^{2^n} \mid n \geq 0\} \cup \{ba^{2^n+1} \mid n \geq 0\}\,.$$

Example 2.35 We consider the $< 1, 1 >$L system $G_8 = (\{a, b\}, \$, P_8, ab^2)$ with P consisting of the following rules:

$$\begin{aligned} &(u, a, b) \to a^2 \quad \text{for} \quad u \in \{a, b, \$\}, \\ &(a, b, v) \to b^3 \quad \text{for} \quad v \in \{a, b, \$\}, \\ &(u, z, v) \to z \quad \text{in all other cases} \end{aligned}$$

Assume that we have a word $a^n b^{2n}$. Then we have to replace the last letter a by a^2, the first letter b by b^3 and the remaining letters x by x. Therefore

$$a^n b^{2n} = a^{n-1}abb^{2n-1} \Longrightarrow a^{n-1}a^2b^3b^{2n-1} = a^{n+1}b^{2(n+1)}$$

for $n \geq 1$, and hence

$$L(G_8) = \{a^n b^{2n} \mid n \geq 1\}\,.$$

Example 2.36 We consider the $< 1, 0 >$L system $G_9 = (\{a, b, o, r\}, \$, P_9, ar)$ with P_9 consisting of the following rules:

$$\begin{aligned} &(\$, a) \to o,\ (o, a) \to b,\ (o, b) \to o,\ (o, r) \to ar, \\ &(u, o) \to a \quad \text{for} \quad u \in \{a, b, o, r, \$\}, \\ &(u, z) \to z \quad \text{in all other cases} \end{aligned}$$

We note that the system is deterministic because, for any pair (u, a), there is exactly one rule $(u, a) \to w$. Then we get the only derivation

$$\begin{aligned} ar &\Longrightarrow or \Longrightarrow aar \Longrightarrow oar \Longrightarrow abr \Longrightarrow obr \Longrightarrow aor \\ &\Longrightarrow oaar \Longrightarrow abar \Longrightarrow obar \Longrightarrow aoar \Longrightarrow oabr \Longrightarrow abbr \\ &\Longrightarrow obbr \Longrightarrow aobr \Longrightarrow oaor \Longrightarrow abaar \Longrightarrow obaar \Longrightarrow \ldots \end{aligned}$$

We shall not determine the language in detail, but we note some properties of the sequence generated.

Fact 1: Each word of $L(G_9)$ starts with o or a.

The statement holds for the start word, and in the sequel o and a alternate as the first letter by the rules $(\$, a) \to o$ and $(\$, o) \to a$.

Fact 2: No word of $L(G_9)$ has the subword oo.

If we want to produce an o which is not in the beginning of the word, then we have to apply the rule $(o, b) \to o$. This requires that the word to which we apply the rule is of the form x_1obx_2 for some words x_1 and x_2. If x_1 ends on a letter different from o, then we get $x_1'aox_2'$. That means, in order to produce oo as a subword in v' with $v \Longrightarrow v'$ the word v has already to contain the subword oo. Because the start word does not contain oo as a subword, no word of $L(G_9)$ contains oo.

Fact 3: For any words $u, v \in \{a, b, o\}^+$ and $z \in \{a, b\}^*$ we have derivations $ubzr \Longrightarrow^* u'ozr$ and $vazr \Longrightarrow^* v'abzr$ for some u' and v' with $|u'| = |u|$ and $|v'| = |v| - 1$.

We prove the statement by simultaneous induction on the length of u and v.

Let $|u| = 1$. By Fact 1, $u = o$ or $u = a$, we have the derivations $obzr \Longrightarrow aozr$ and $abzr \Longrightarrow obzr \Longrightarrow aozr$, respectively. Thus the induction basis holds for $ubzr$. Analogously we prove it for $vazr$.

Let $|v| \geq 2$. We distinguish three cases.

Case 1: $v = v_1o$. Then $vazr = v_1oazr \Longrightarrow v_1'abzr$, and $|v_1'| = |v_1| = |v| - 1$. Therefore the induction step is done.

Case 2: $v = v_1b$. Then $vazr = v_1bazr$. Now we apply the induction assumption for $v_1bz'r$ with $z' = az$ (this can be done since $|v_1| = |v| - 1 < |v|$)and get $v_1bz'r \Longrightarrow^* v_1'oz'r = v_1'oazr \Longrightarrow v_1''abzr$ and $|v_1''| = |v_1'| = |v_1| = |v| - 1$.

Case 3: $v = v_1a$. Then $vazr = v_1aazr \Longrightarrow^* v_1'abazr$ by induction hypothesis. Moreover, $|v_1'ab| = |v_1'| + 2 = |v_1| + 1 = |v|$. By Case 2 (with $v' = v_1'ab$ and $|v'| = |v|$), we know that $v_1'abazr = v'azr \Longrightarrow^* v''abzr$ with $|v''| = |v'| - 1 = |v_1'ab| - 1 = |v| - 1$.

Analogously we prove the statement for $|u| \geq 2$.

Now assume that in some step of the derivation we have an extension of the word with respect to the length. By the rules, the only possibility is $xor \Longrightarrow yar$. Let $|yar| = s + 2$. Now, by Fact 3, we have the following derivation

$$yar \Longrightarrow^* y_1abr \Longrightarrow^* y_2abbr \Longrightarrow^* y_3abbbr \Longrightarrow^* \ldots \Longrightarrow^* y_{s-1}ab^{s-1}r \Longrightarrow^* ab^sr \qquad (2.11)$$

where $s = |y| + 1 = |y_i| + i$ for $1 \leq i \leq$, followed by the derivation

$$ab^sr \Longrightarrow ob^sr \Longrightarrow p_1ob^{s-1}r \Longrightarrow p_2ob^{s-2}r \Longrightarrow \ldots \Longrightarrow p_sor \Longrightarrow p_{s+1}aar \qquad (2.12)$$

where $|p_i| = i$ for $1 \leq i \leq s$ and $|p_{s+1}| = s$. Therefore we get a word of length $s + 3$.

This proves that $L(G_9)$ is infinite.

Example 2.37 For any $k \in \mathbf{N}_0$ and any $l \in \mathbf{N}_0$, the context-free language

$$L = \{a^nb^{2n} \mid n \geq 1\} \cup \{a^{2n}b^n \mid n \geq 1\}$$

cannot be generated by a $< k, l >$L systems. This can be seen as follows.

Assume the contrary, i.e., there is a $< k, l >$L system $G = (\{a, b\}, \$, P, \omega)$ for some non-negative integers k and l such that $L(G) = L$.

To words $a^n b^{2n}$ and $a^{2n} b^n$ with sufficiently large n, we can only apply rules with left hand sides $(\$^r a^{k-r}, a, a^l)$, $(a^k, a, a^s b^{l-s})$, $(a^r b^{k-r}, b, b^l)$ and $(b^l, b, b^s \$^{l-s})$ with $0 \leq r \leq k$ and $0 \leq s \leq l$. We prove some facts on the rules with these left hand sides.

Fact 1: If $(\$^r a^{k-r}, a, a^l) \to w \in P$, then $w \in a^*$, and if $(b^k, b, b^s \$^{l-s}) \to v \in P$, then $v \in b^*$.

We prove the statement only for (a^k, a, a^l); the proof for the other cases can be given analogously.

Let $n > k$. Let $a^n b^{2n} = a^k a a^{n-k+1} b^{2n}$. If $(a^k, a, a^l) \to w \in P$ and w contains a b, then all a's in a word which is generated from $a^n b^{2n}$ in one derivation step have there origin in the the first $k + 1$ letters of $a^n b^{2n}$. The same holds for words generated from $a^{2n} b^n$. However, then we cannot produce words which start with an arbitrarily large numbers of a's, since a finite number of letters can produce only words of limited length.

Fact 2: If $(\$^r a^{k-r}, a, a^l) \to w_1 \in P$ and $(\$^r a^{k-r}, a, a^l) \to w_2 \in P$, then $w_1 = w_2$, and if $(b^k, b, b^s \$^{l-s}) =\!\!\to v_1 \in P$ and $(b^k, b, b^s \$^{l-s}) =\!\!\to v_2 \in P$, then $v_1 = v_2$.

Again, we prove the statement only for (a^k, a, a^l). Let $(a^k, a, a^l) \to w_1$ and $(a^k, a, a^l) \to w_2$ be two rules in P. Let $n \geq k + l + 2$, then we have the derivations $a^n b^{2n} = a^k a a^{n-k-1} b^{2n} = u_1 w_1 u_2$ and $a^n b^{2n} \Longrightarrow u_1 w_2 u_2$ where u_1 and u_2 are obtained from a^k and a^{n-k-1}, respectively. By Fact 1, u_1 contains only the letter a and u_2 starts with a. If $u_1 w_1 u_2 = a^r b^s \in L(G) = L$ then $u_1 w_2 u_2 = a^{r'} b^s$. If $r = r'$, then $w_1 = w_2$, and we are done. If $r \neq r'$, then we can assume without loss of generality, that $r = 2s$ and $r' = s/2$. Thus $r - r' = 3s/2$ which is arbitrarily large for arbitrarily large n. However, $r - r' = |w_1| - |w_2|$ is bounded.

Fact 3: For sufficiently large n, the subword $a^k b^l$ of $a^n b^{2n}$ or $a^{2n} b^n$ generates a uniquely determined word $a^{p'} b^{q'}$ for some $p' \geq 0$ and $q' \geq 0$.

The proof is analogous to that of Fact 2.

Let $(a^k, a, a^l) \to a^s$ and $(b^k, b, b^l) \to b^t$ be the only rules for (a^k, a, a^l) and (b^k, b, b^l), respectively. We note that $s \geq 1$ and $t \geq 1$, since we cannot generate an infinite number of occurrences of a and/or b, otherwise.

Moreover, by Facts 1 and 2, the the first k letters and the last l letters of $a^n b^{2n}$ and $a^{2n} b^n$, where n is sufficiently large, generate uniquely determined words a^p and b^q, respectively. Thus, for sufficiently large n, we have the unique derivations

$$a^n b^{2n} \Longrightarrow a^p a^{(n-k-l)s} a^{p'} b^{q'} b^{2n-k-l)t} b^q = a^{p+p'+(n-k-l)s} b^{q+q'+(2n-k-l)t} \tag{2.13}$$

and

$$a^{2n} b^n \Longrightarrow a^p a^{(2n-k-l)s} a^{p'} b^{q'} b^{n-k-l)t} b^q = a^{p+p'+(2n-k-l)s} b^{q+q'+(n-k-l)t} \,. \tag{2.14}$$

Fact 4: There is a number n_0 such that, for all $n \geq n_0$, any word $a^n b^{2n}$ generates only words $a^m b^{2m}$ for some m and any word $a^{2n} b^n$ generates only words $a^{2m'} b^{m'}$ for some m'.

Assume the contrary, i.e., $a^n b^{2n} \Longrightarrow a^{2m} b^m$ for some m or $a^{2n} b^n \Longrightarrow a^{m'} b^{2m'}$ for some m'. We only discuss the former case; the latter one can be handle analogously. By (2.13), we get

$$2m = p + p' + (n - k - l)s \text{ and } m = q + q' + (2n - k - l)t \,.$$

By an easy calculation we get

$$n = \frac{p + p' - 2q - 2q' + k(2t - s) + l(2t - s)}{4t - s}.$$

This is a contradiction, since the left side is unbounded, but the right side is a constant.

Fact 5: $s = t = 1$.

Let n be sufficiently large. By (2.13), from $a^n b^{2n}$ for sufficiently large n we derive $a^m b^{2m}$ with $m = p + p' + (n - k - l)s$. Let $w = a^{n'} b^{2n'}$ be the word which generates $a^{m+1} b^{2(m+1)}$ (w has to have this form by Fact 4). By (2.13) we have

$$p + p' + (n' - k - l)s = m + 1 = p + p' + (n - k - l)s + 1\,.$$

Thus $(n' - n)s = 1$. This can only hold iff $s = 1$ and $n' = n + 1$.

Analogously, we show $t = 1$.

Let $a^n b^{2n} \Longrightarrow a^m b^{2m}$ and $a^{2n} b^n \Longrightarrow a^{2m'} b^{m'}$. Then we have

$$\begin{array}{ll} m = p + p' + (n - k - l), & 2m = q + q' + (2n - k - l), \\ 2m' = p + p' + (2n - k - l), & m' = q + q' + (n - k - l). \end{array}$$

by (2.13), (2.14) and Fact 5. By an easy calculation one gets $p + p' = q + q'$ and then

$$m = 2m - m = q + q' + (2n - k - l) - (p + p' + (n - k - l) = n\,.$$

Therefore we only generate a finite language in contrast to the infinity of $L = L(G)$.

2.2.2 Some results on Lindenmayer systems with interaction

For $k \in \mathbf{N}_0$ and $l \in \mathbf{N}_0$, by $\mathcal{L}(< k, l > L)$ we denote the family of all languages generated by $< k, l >$L systems. Further we set

$$\mathcal{L}(IL) = \bigcup_{k \geq 0, l \geq 0} \mathcal{L}(< k, l > L)\,.$$

From the definitions we get directly the following statement.

Corollary 2.38 *i)* $\mathcal{L}(< 0, 0 > L) = \mathcal{L}(0L)$.
ii) $\mathcal{L}(< k, l > L) \subseteq \mathcal{L}(< k', l' > L) \subseteq \mathcal{L}(IL)$ *for any* $k, k', l, l' \in \mathbf{N}_0$, $k \leq k'$ *and* $l \leq l'$. $\square$

First we study the relations between families of Lindenmayer languages with interaction and languages of the Chomsky hierarchy.

Lemma 2.39 *For any recursively enumerable language* $L \subseteq T^*$ *there is a* $< 1, 1 >$*L system* G *such that* $L(G) \cap T^* = L$.

Proof. Let L be a recursively enumerable language. Then $L = L(H)$ for some grammar $H = (N, T, P, S)$ in Kuroda normal form (see Theorem 1.5). With any rule $p = AB \to CD \in P$ we associate the two new letters $A_{l,p}$ and $B_{r,p}$. We define

$$\begin{aligned}
N' &= \{A' \mid a \in N\},\\
N_l &= \{A_{l,p} \mid p = AB \to CD \in P\},\\
N_r &= \{B_{r,p} \mid p = AB \to CD \in P\},\\
V &= N \cup N' \cup N_l \cup N_r \cup T \cup \{F\},\\
V' &= V \cup \{\$\},\\
P_T &= \{(u, a, v) \to a \mid a \in T, u, v \in V'\},\\
P_N &= \{(u, A, v) \to \overline{A} \mid \overline{A} = A' \text{ or } \overline{A} = A_{r,p} \text{ or } \overline{A} = A_{l,q} \text{ for some } p, q \in P,\ u, v \in V'\},\\
P_{N'} &= \{(u, A', v) \to A \mid A \in N, u, v \in V'\} \cup \{(u, A', v) \to w \mid A \to w \in P, u, v \in V'\},\\
P_{r,l} &= \{(u, A_{l,p}, B_{r,p}) \to C \mid p = AB \to CD \in P, u \in V'\}\\
&\quad \cup\{(A_{l,p}, B_{r,p}, v) \to D \mid p = AB \to CD \in P, v \in V'\}\\
&\quad \cup\{(u, A_{l,p}, v) \to F \mid u \in V', v \in V' \setminus \{B_{r,p}\}\}\\
&\quad \cup\{(u, A_{l,p}, v) \to F \mid u \in V' \setminus \{A_{l,p}\}, v \in V'\},\\
P' &= P_T \cup P_N \cup P_{N'} \cup P_{r,l} \cup \{(u, F, v) \to F^2 \mid u, v \in V\}
\end{aligned}$$

and consider the $< 1, 1 >$L system $G = (V, \$, P', S)$.

Let w be a sentential form generated by H and assume that $w \in L(G)$ (note that these requirements hold for the axiom) and let $w \Longrightarrow_H w'$ by an application of the rule $p = AB \to CD \in P$. Then we replace the occurrences of A and B to which p is applied by $A_{l,p}$ and $B_{r,p}$, respectively, all remaining nonterminals E by the associated E' and any terminal a by a. This corresponds to a derivation step in G which yields a word w''. To any occurrence of a symbol E' in w'' we apply $(u, E', v) \to E$, to any terminal a in w'' we apply $(u, a, v) \to a$, and we apply $(u, A_{l,p}, B_{r,p}) \to C$ and $(A_{l,p}, B_{r,p}, v) \to D$. This leads to w'. Analogously, we can prove that derivation steps in H with an application of rules of the forms $A \to B$ or $A \to a$ or $A \to \lambda$ can be simulated in G. Thus any sentential form of H belongs to $L(G)$, too. Since $L(H)$ is the intersection of all sentential forms of H with T^*, we have $L(H) \subseteq L(G) \cap T^*$.

Conversely, by arguments as above, it is easy to see, that any sentential form of G is a sentential form of H or it contains the letter F. Thus $L(G) \cap T^* \subseteq L(H)$.

Therefore, $L(G) \cap T^* = L(H) = L$. □

Theorem 2.40 *The diagram of Figure 2.7 holds.*

Proof. i) $\mathcal{L}(REG) \subset \mathcal{L}(IL)$.
By Theorem 2.13, there exists a 0L language L which is not regular. Since $L \in \mathcal{L}(IL)$ by Corollary 2.38, we have a language in $\mathcal{L}(IL) \setminus cL(REG)$. Thus it is sufficient to prove the inclusion $\mathcal{L}(REG) \subseteq \mathcal{L}(IL)$.

Assume that $K \subset V^*$ is a regular language. Then K is accepted by a deterministic finite automaton $\mathcal{A} = (V, Z, z_0, F, \delta)$. Let $n = \#(Z)$.

We first note that K contains a word whose length is at most n. Assume the contrary, i.e., the shortest word w of K has a length $r \geq n + 1$. Let $w = a_1 a_2 \dots a_r$. We consider

$$\begin{tikzcd}
 & \mathcal{L}(RE) \\
 & \mathcal{L}(CS) \arrow[u] \\
\mathcal{L}(IL) \arrow[uur] & \mathcal{L}(CF) \arrow[u] \\
\mathcal{L}(0L) \arrow[u] \arrow[uur] & \mathcal{L}(REG) \arrow[u] \arrow[ul]
\end{tikzcd}$$

Figure 2.7: Relations between families of Lindenmayer languages with interaction and languages of the Chomsky hierarchy

the states $z_i = \delta(z_0, a_1a_2 \dots a_i)$ for $1 \le i \le r\}$. We have at least $n+1$ elements z_i, but only n states. Thus there are two numbers i and j, $1 \le i < j \le r$ such that $z_i = z_j$. By $w \in K$, we have

$$\delta(z_0, a_1 \dots a_r) = \delta(\delta(z_0, a_1 \dots a_j), a_{j+1} \dots a_r) = \delta(z_j, a_{j+1} \dots a_r) \in F\,.$$

Furthermore,

$$\begin{aligned}
\delta(z_0, a_1 \dots a_i a_{j+1} \dots a_r) &= \delta(\delta(z_0, a_1 \dots a_i), a_{j+1} \dots a_r) \\
&= \delta(z_i, a_{j+1} \dots a_r) \\
&= \delta(z_j, a_{j+1} \dots a_r) \in F\,.
\end{aligned}$$

Therefore $v = a_1a_2 \dots a_i a_{j+1} a_{j+2} \dots a_r \in K$ and $|v| = r - (j-i) < r$ which contradicts the choice of w as a shortest word in K.

Analogously, we prove that, for any state z, there is a word w of length at most n with $\delta(z, w) = z$ or there is no word v with $\delta(z, v) = z$.

Now we construct the $< n+1, n >$L system $H = (V, \$, P, \omega)$ where ω is one word in K with length at most n and P consists of all rules of the form

a1) $(\$^{n+1}, b_1, b_2b_3 \dots b_s\$^{n-s+1}) \to w$,
where $s \le n$ and w is a word of K of length at most $2n$,

a2) $(\$^{n-r+1}b_1b_2 \dots b_r, b_{r+1}, b_{r+2}b_{r+3} \dots b_s\$^{n-s+r}) \to \lambda$,
where $r+1 \le s \le n$

(by rules of these types we generate all words of K of length at most $2n$ from a word of length at most n),

b1) $(\$^{n+1}, a_0, a_1a_2 \dots a_n) \to w$,
where $a_i \in V$ for $0 \le i \le n$ and $w = a_0a_1 \dots a_t v a_{t+1}a_{t+2} \dots a_n$
for some $v \in V^*$ with $|v| \le n$, $\delta(z_0, a_0 \dots a_t) = \delta(z_0, a_0 \dots a_t v)$,

b2) $(\$^{n-r+1}c_1c_2 \dots c_r, a, d_1d_2 \dots d_n) \to \lambda$,
where $1 \le r \le n$, $c_i \in V$ for $1 \le i \le r$, $d_i \in V \cup \{\$\}$ for $1 \le i \le n$,
$d_1d_2 \dots d_n \in V^s\{\$\}^{n-s}$, $r + s \ge n$

b3) $(c_1c_2 \dots c_{n+1}, a, d_1d_2 \dots d_n) \to a$,
where $c_i \in V$ for $1 \le i \le n+1$, $d_i \in V \cup \{\$\}$ for $1 \le i \le n$,
$d_1d_2 \dots d_n \in V^s\{\$\}^{n-s}$, $s \ge 0$

(by these rules, for a word x of length at most $n+1$, i.e., $x = a_0a_1 \dots a_n x'$, we have a derivation

$$a_0a_1 \dots a_t a_{t+1}a_{t+2} \dots a_n x' \Longrightarrow a_0a_1 \dots a_t v a_{t+1}a_{t+2} \dots a_n x' \qquad (2.15)$$

where v is an arbitrary word with

$$\delta(z_0, a_0 \dots a_t) = \delta(z_0, a_0 \dots a_t v) \text{ and } |v| \le n\,. \qquad (2.16)$$

We now prove that $L(H) \subseteq K$. By definition, the start word belongs to K. Moreover, all words generated from the start word by an application of rules of type a1) and a2) yield a word of K, and rules of types b1), b2) and b3) cannot be applied to the start word. Further, if $x \in K$ and we apply rules of type b1), b2) and b3) to x, then

$$\delta(z_0, a_0a_1 \dots a_t a_{t+1}a_{t+2} \dots a_n x') = \delta(z_0, a_0a_1 \dots a_t v a_{t+1}a_{t+2} \dots a_n x')$$

which implies that the generated word $a_0a_1 \dots a_t v a_{t+1}a_{t+2} \dots a_n x'$ belongs to $T(\mathcal{A}) = K$, too. Thus we produce only words of K.

Conversely, $K \subseteq L(H)$ also holds. This can easily be proved by induction on the length of the words of K. If $w \in K$ has a length at most $2n$, then w can be produced by a1) and a2) applied to the start word. Thus the induction basis is satisfied. If $w \in K$ has a length r with $r > 2n$, i.e., $w = e_1e_2 \dots e_{n+1}v$, then there are integers i and j with $1 \le i < j \le n+1$ and $\delta(z_0, e_1e_2 \dots e_i) = \delta(z_0, e_1e_2 \dots e_j)$. Thus $w' = e_1e_2 \dots e_i e_{j+1}e_{j+2} \dots e_{n+1}v$ belongs to K. By induction hypothesis, $w' \in L(H)$. Now we are able to produce w from w' by an applications of rules of type b1), b2) and b3). Therefore $w \in L(H)$.

ii) $\mathcal{L}(0L) \subseteq \mathcal{L}(IL)$.
The inclusion holds by definition. Since, by Theorem **??**, there is a regular language R which is not in $\mathcal{L}(0L)$. By part i) of this proof $R \in \mathcal{L}(IL) \setminus \mathcal{L}(0L)$ holds. Thus the inclusion is proper.

iii) $\mathcal{L}(IL)$ and $\mathcal{L}(CF)$ are incomparable.
Since $\mathcal{L}(0L)$ contains a non-context-free language, it follows that $\mathcal{L}(IL)$ as a superset of $\mathcal{L}(0L)$ contains a non-context-free language.

On the other hand by Example 2.37 the context-free language

$$\{a^n b^{2n} \mid n \ge 1\} \cup \{a^{2n} b^n \mid n \ge 1\}$$

is not a $< k, l >$L language for any $k \in \mathbf{N}_0$ and $l \in \mathbf{N}_0$.

iv) $\mathcal{L}(IL) \subset \mathcal{L}(RE)$.
In analogy to the proof that any 0L language can be generated by a phrase structure

grammar, we can show that any $< k,l >$L language is in $\mathcal{L}(RE)$. Therefore $\mathcal{L}(IL) \subseteq \mathcal{L}(RE)$. The strictness of this inclusion follows from the Example 2.37.

v) $\mathcal{L}(IL)$ and $\mathcal{L}(CS)$ are incomparable.

The existence of a context-sensitive language which is not in $\mathcal{L}(IL)$ follows by Example 2.37.

Now let M be a set with $M \in \mathcal{L}(RE)$ and $M \notin \mathcal{L}(CS)$. Such a set exists by the proper inclusion of $\mathcal{L}(CS)$ in $\mathcal{L}(RE)$. By Lemma 2.39, there is a $< 1,1 >$L system G and a set T with $L(G) \cap T^* = M$. If $L(G)$ is context-sensitive, then $M \in \mathcal{L}(CS)$ by the known closure properties of $\mathcal{L}(CS)$ (see Chapter 1). Thus $L(G) \notin \mathcal{L}(CS)$. Therefore $\mathcal{L}(IL)$ contains a non-context-sensitive language. □

We now compare the families $\mathcal{L}(< k,l > L)$ with each other.

Lemma 2.41 *For any $k,k',l,l' \in \mathbf{N}$ with $k+l = k'+l'$, $\mathcal{L}(< k,l > L) = \mathcal{L}(< k',l' > L)$.*

Proof. We first prove $\mathcal{L}(< k,l > L) = \mathcal{L}(< k+1,l-1 > L)$ for $k \geq 1$ and $l \geq 2$. Let $G = (V,\$,P,\omega)$ be a $< k,l >$L system. Then we construct the $< k+1,l-1 >$L system $G' = (V,\$,P',\omega)$ where P' consists of all rules of the form
– $(\$^{k+1},a,v) \to \lambda$ where $|v| = l-1$ [2],
– $(ub,a,v) \to w$ where $(u,b,av) \to w \in P$, $|u| = k$, $|v| = l-1$, $v \neq \$^{l-1}$,
– $(cub,a,\$^{l-1}) \to w_1w_2$ where $(cu,b,a\$^{l-1}) \to w_1 \in P$, $(ub,a,\$^l) \to w_2 \in P$, $|c| = 1$, $|u| = k-1$.

Obviously, $z \Longrightarrow_G z'$ if and only if $z \Longrightarrow_{G'} z'$. The only difference is that in G' the first letter is replaced by λ, the i-th letter is replaced by w in G' iff the $(i-1)$-st letter is replaced by w in G, and the last letter is replaced by w_1w_2 in G' iff the last two letters are replaced by w_1 and w_2, respectively, in G. Therefore, $L(G) = L(G')$.

By an iterated application of equalities of this type, we get

$$\mathcal{L}(< k,1 > L) = \mathcal{L}(< k-1,2 > L) = \mathcal{L}(< k-2,3 > L) = \ldots = \mathcal{L}(< 1,k > L)\,.$$

□

For $k \geq 2$, we set $\mathcal{L}(kL) = \mathcal{L}(< 1,k-1 > L)$.

By Lemma 2.41, $\mathcal{L}(kL) = \mathcal{L}(< s,r > L)$ for any $s \in \mathbf{N}$ and $r \in \mathbf{N}$ with $s+r = k$.

Lemma 2.42 *For any $k,k',l,l' \in \mathbf{N}_0$ with $k \leq k'$, $l \leq l'$ and $k+l < k'+l'$,*

$$\mathcal{L}(< k,l > L) \subset \mathcal{L}(< k',l' > L).$$

Proof. For a proof of this lemma, we refer to [13]. □

The following theorem relates the families $\mathcal{L}(< k,l > L)$ to each other.

Theorem 2.43 *The diagram of Figure 2.8 holds.*

Proof. All inclusions and their strictnesses follow by Lemmas 2.41 and 2.42.

[2] We give here only the length of the words, their forms depend on the possibilities which are allowed by the rules in a $< k,l >$L system.

$$\begin{tikzcd}
 & \vdots & \\
\vdots & \mathcal{L}(4L) \arrow[u] & \vdots \\
\mathcal{L}(<3,0>L) \arrow[u] \arrow[ur] & \mathcal{L}(3L) \arrow[u] & \mathcal{L}(<0,3>L) \arrow[u] \arrow[ul] \\
\mathcal{L}(<2,0>L) \arrow[u] \arrow[ur] & \mathcal{L}(2L) \arrow[u] & \mathcal{L}(<0,2>L) \arrow[u] \arrow[ul] \\
\mathcal{L}(<1,0>L) \arrow[u] \arrow[ur] & & \mathcal{L}(<0,1>L) \arrow[u] \arrow[ul] \\
 & \mathcal{L}(0L) \arrow[ul] \arrow[ur] &
\end{tikzcd}$$

Figure 2.8: Relations between families of Lindenmayer languages with interaction

We now prove the existence of a language $L \in \mathcal{L}(<1,0>L)$ which is not contained in $\mathcal{L}(<0,l>L)$ for any $l \geq 1$. This shows that no family of the left chain is contained in some family of the right chain.

Let

$$L = \{c\} \cup \{a^{2^n} \mid n \geq 0\} \cup \{ba^{2^n+1} \mid n \geq 0\}.$$

By Example 2.34, $L = L(G_7)$ for the $<1,0>$L system G_7. Therefore $L \in \mathcal{L}(<1,0>L)$.

Now assume that $L \in \mathcal{L}(<0,l>L)$ for some $l \geq 1$. Let $G = (\{a,b,c\}, \$, P, \omega)$ be the $<0,l>$L system generating L. It is easy to see that $(a,v) \to w_{a,v} \in P$ and $(b,v) \to w_{b,v} \in P$ imply $w_{a,v} \in a^*$ and $w_{b,v} \in ba^*$ (otherwise, e.g., a^{2^n}, $n \geq l$, would derive a word with at least two occurrences of b). Moreover, for any v, $w_{a,v}$ and $w_{b,v}$ are uniquely determined. E.g., if $(a,a^l) \to w_1$ and $(a,a^l) \to w_2$, then we derive $w_1' = w_1 w$ and $w_2' = w_2 w$ from a^{2^n} with sufficiently large n where w originates from the last $2^n - 1$ letters. Since $||w_1'| - |w_2'|| = ||w_1| - |w_2||$ and the length between different words over $\{a\}$ in L grows unbounded, we obtain a contradiction.

Let $(a,a^l) \to a^r$. If $r = 0$, then we cannot generate words with an unbounded number of occurrences of a. If $r = 1$, then the increase of the length originates only from the first letter b and/or the last l letters such that the increase is bounded in contrast to the structure of the words of L.

Now assume that $a^{2^n} \Longrightarrow a^{2^m}$ with $m \geq n$ and $(b,a^l) \to ba^s$. Then $baa^{2^n} \Longrightarrow ba^s a^r a^{2^m} = ba^{2^m+r+s}$. Thus $r + s = 1$ which gives $r \leq 1$ which is impossible as shown above.

Hence in all cases we got a contradiction which shows $L \notin \mathcal{L}(<0,l>L)$.

Taking L^R, by analogous arguments one can show that $L^R \in \mathcal{L}(<0,1>L)$ and $L^R \notin \mathcal{L}(<k,0>L)$ for any $k \in \mathbf{N}$ which proves that no family of the right chain is contained in some family of the left chain.

We omit the proof of the incomparability of $\mathcal{L}(< k, 0 > L)$ with $\mathcal{L}(kL)$ and that of $\mathcal{L}(< 0, k > L)$ with $\mathcal{L}(kL)$. □

Finally, we present some results on topics which we studied in Sections 1.1.4, 1.1.5 and 1.1.6 for D0L systems. We omit the exact formal definitions of adult languages and growth functions of Lindenmayer systems with interaction. They can given by a straightforward translation from the concepts for (deterministic) 0L systems.

We start with a characterization of adult languages of L systems with interaction.

By $\mathcal{L}(AIL)$ we denote the family of all adult languages which can be generated by $< k, l >$L systems with $k \in \mathbf{N}_0$ and $l \in \mathbf{N}_0$.

Theorem 2.44 *$\mathcal{L}(AIL) = \mathcal{L}(RE)$.*

Proof. Let L be an arbitrary language of $\mathcal{L}(RE)$. We consider the $< 1, 1 >$L system constructed G constructed in the proof of Lemma 2.39. It is easy to see that $L_A(G) = L(G) \cap T^* = L$. Thus $\mathcal{L}(RE) \subset \mathcal{L}(AIL)$.

Let H be an arbitrary $< k, l >$L system. Then $L(H) \in \mathcal{L}(RE)$ by Theorem 2.40. We construct a Turing machine M which checks for a word w whether or not w derives only w according to the rules of H (as in the case of 0L system, if $w \Longrightarrow w$ is the only derivation from w, then there is exactly one rule for any letter and its context, and thus M has only to simulate the derivation and reject if there are more rules or one does not get w). Because Turing machines accept recursively enumerable languages, we have $T(M) \in \mathcal{L}(RE)$. Since $L_A(H) = L(H) \cap T(M)$ and $\mathcal{L}(RE)$ is closed under intersection, we get $L_A(H) \in \mathcal{L}(RE)$. Therefore $\mathcal{L}(AIL) \subseteq \mathcal{L}(RE)$. □

By Theorem 2.40 and Theorems 2.20 and 2.44, we know that 0L systems generate a smaller family of languages and a smaller family of adult languages than L systems with interaction. We now show that this also holds with respect to growth functions.

Theorem 2.45 *There is a deterministic $< 1, 0 >$L system G such that its growth function is not a growth function of a D0L system. More precisely, f_G is not bounded by a constant and, for any polynomial p with $p(m) \geq m$ for all $m \geq m_0$ for some $m_0 \in \mathbf{N}$,*

$$\lim_{m \to \infty} \frac{f_G(m)}{p(m)} = 0\,.$$

Proof. We consider the $< 1, 0 >$L system G_9 of Example 2.36. In Example 2.36, we have shown that $L(G_9)$ is infinite. Therefore f_{G_9} cannot be bounded by a constant.

Considering (2.11) and (2.12) we see that at least m derivation steps are necessary in order to get a length extension of a word of length m by one. Thus we need at least $1 + 2 + 3 + \ldots + m$ steps in order to obtain a word of length $m + 1$. Therefore we get

$$f_{G_9}(\frac{m(m+1)}{2}) \leq m + 1$$

or

$$f_{G_9}(m) \leq -\frac{1}{2} + \sqrt{\frac{1 + 8m}{4}} \leq \sqrt{2m}\,.$$

Therefore we get

$$\lim_{m\to\infty} \frac{f_{G_9}}{p(m)} \leq \lim_{m\to\infty} \frac{\sqrt{2m}}{m} = 0\,.$$

By Theorem 2.31, f_{G_9} grows slower than any unbounded growth function of a D0L system. Hence f_{G_9} is not a growth function of a D0L system. □

Chapter 3

DNA Molecules and Formal Languages

3.1 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 the evolution.

The nucleotides which form the DNA strands are molecules consist of a base, which is adenine, cytosin, guanine or thymine, a sugar group and a phosphate group. The left part of Figure 3.1 gives the nucleotide with the thymine base. The five carbons within the sugar group which are denoted by 1', 2', 3', 4' and 5' in the left part of Figure 3.1 are of special importance. Using this notation one can represent the whole molecule schematically as done in the right part of Figure 3.1. In the sequel we shall denote these molecules by A, C, G and T, depending on its base adenine, cytosine, guanine and thymine, respectively.

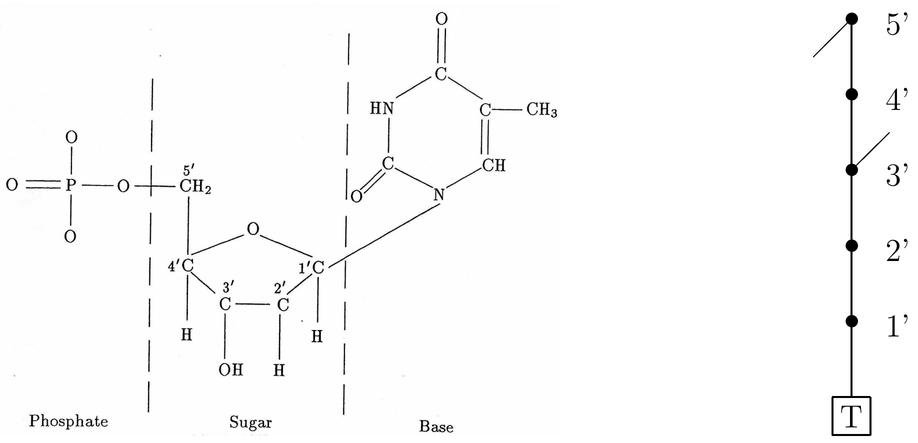


Figure 3.1: Molecule with thymine base.

The carbon group 5' of one nucleotide and the OH group of a carbon 3' of another nucleotide can join, producing a phosphodiester and water. We obtain the molecule shown in the right part of Figure 3.2. This forms a single strand. One can see that there is direction by the joins, which we denote by $5' \rightarrow 3'$. Finally, to get a double strand the basic groups are connected. However, not all combinations are possible. We can only combine adenine with thymine, thymine with adenine, guanine with cytosine and cytosine with guanine. This pairing is called the Watson-Crick complementarity. Moreover, the lower part has to have the opposite direction as the upper part. The middle part of Figure 3.2 gives a schematical description of a double stranded DNA molecule.

We note that Figure 3.2 is only an illustration. In nature, the double strands are twisted and in a three-dimensional space, i.e., they are far from the linear structure as given in the upper part of Figure 3.2.

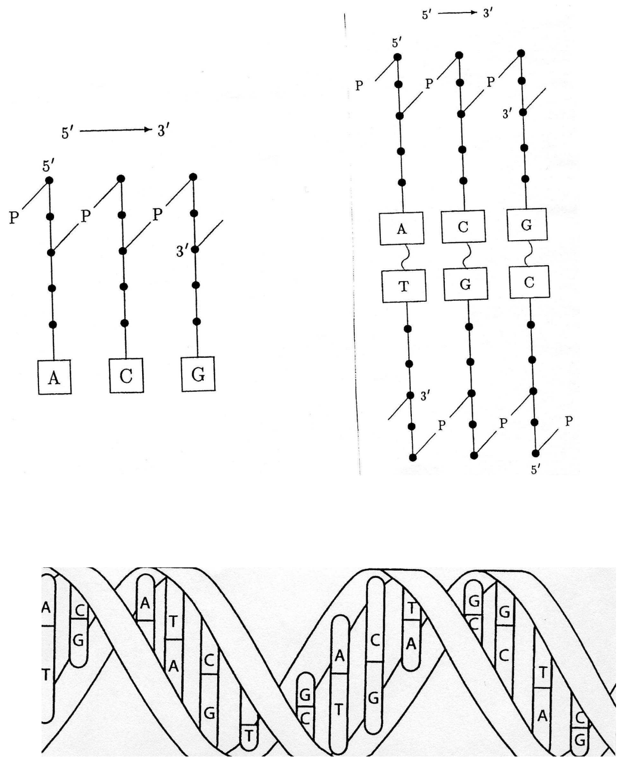


Figure 3.2: Structure of a DNA strand.

In order to describe a double stranded DNA molecule it is sufficient to give the basic parts, which are pairs

$$\begin{matrix} A & T & C & & G \\ T, & A, & G & \text{and} & C. \end{matrix}$$

However, since the upper part uniquely determines the lower part, in many cases it is sufficient to consider only the upper strand, which means that the DNA molecule can be represented as a word over the alphabet $\{A, C, G, T\}$.

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 (see left part of Figure 3.3). 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 right part of Figure 3.3).

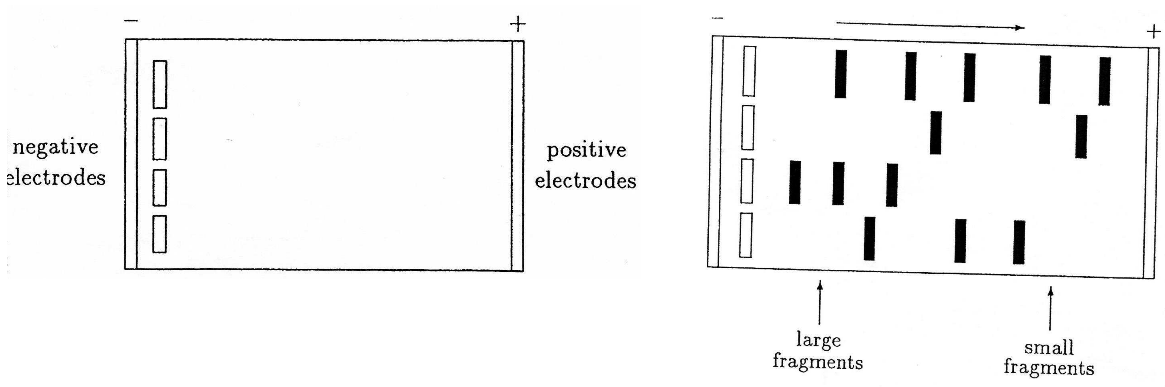


Figure 3.3: Measuring the length of DNA molecules by gel electrophoresis.

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

Figure 3.4 shows the polymerase, where in the direction from 5' to 3' we complete a partial double strand to a complete double strand, and the transferase, where we add in one strand in the direction from 5' to 3' further nucleotides.

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 3.5). Then we assume that in the solution are so-called primers which are obtained from the right end of the upper strand and the left end of the lower strand by the Watson-Crick complementarity. If we cool the solution, then the primer are connected with the corresponding ends (see the middle part of Figure 3.5). Finally, by a polymerase we can fill the missing parts and obtain two copies of the original DNA strand (see lower part of Figure 3.5).

This cycle can be iterated. After some cycles we have increased drastically the number of the strand we are interested in. Now there is a chance by some filtering to separate this strand from the others in the solution.

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 right part of Figure 3.6 this procedure is shown for the restriction enzyme *XmaI* which has the recognition site *CCCGGG* in the upper strand. If we take into consideration the direction, then the lower part is the same. The cut is performed after

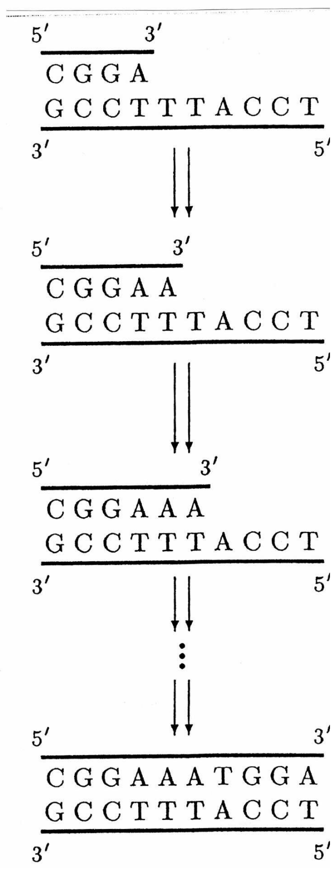


Figure 3.4: Polymerase

the first C in both strands, and moreover, the bonds between both strands of the molecule are separated between the cuts.

The left part of Figure 3.6 shows the same procedure for the restriction enzyme $EcoRI$ with the recognition site $GAATTC$. However, since the recognition site occurs two times in the DNA molecule the cut is formed at two places.

The ligase can be considered as the operation inverse to endonuclease. Here, in a first step the upper and lower part of the overhanging parts are connected, which is called a bonding. In a second step the ligase itself produces the connections between the free 5' and 3' molecules. Figure 3.7 illustrates the ligase.

The hybridization combines endonuclease and ligase. It is described in Figure 3.8. In the first step endonuclease with the enzyme $HpaII$ and the recognition site $rs = CCGG$ is done on the strands $\alpha_1 rs\beta_1$ and $\alpha_2 rs\beta_2$. By this operation we get four molecules and each of them has an overhanging part. In the second step by a bonding and ligase we paste these four molecules, however, we combine the α_1-part with the β_2-part and the α_2-part with the β_1-part. Thus we obtain the new molecules $\alpha_1 rs\beta_2$ and $\alpha_2 rs\beta_1$.

In a hybridization we use the same enzyme and the same recognition site at some places in the molecule. Obviously, one can take different enzymes with different recognition sites

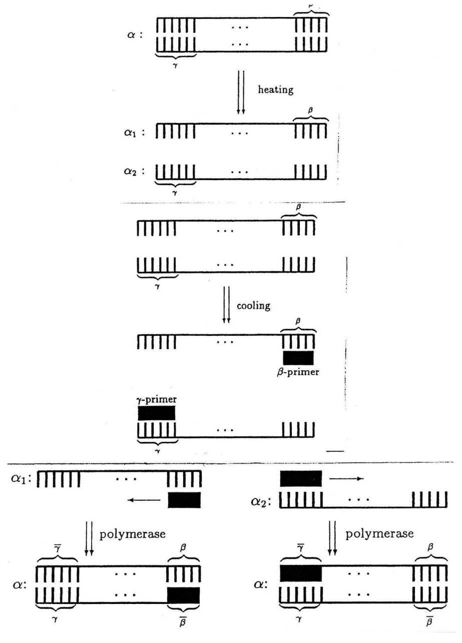


Figure 3.5: Polymerase chain reaction

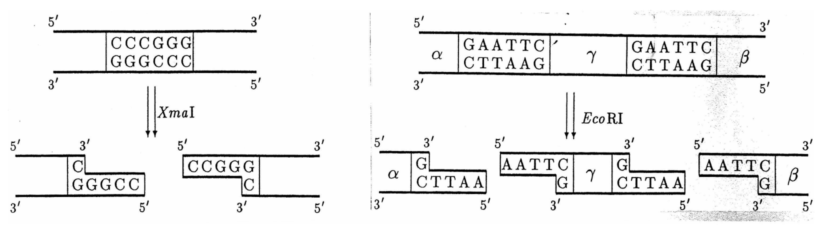


Figure 3.6: Endonuclease

and perform an operation analogous to the hybridization. This new operation is the splicing which will be intensively studied from the formal language point of view later.

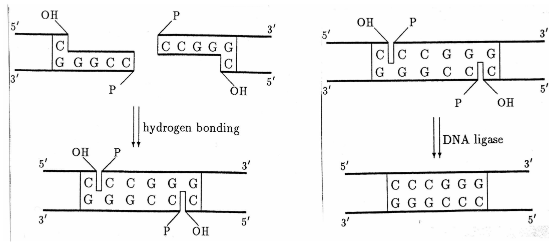


Figure 3.7: Bonding and Ligase.

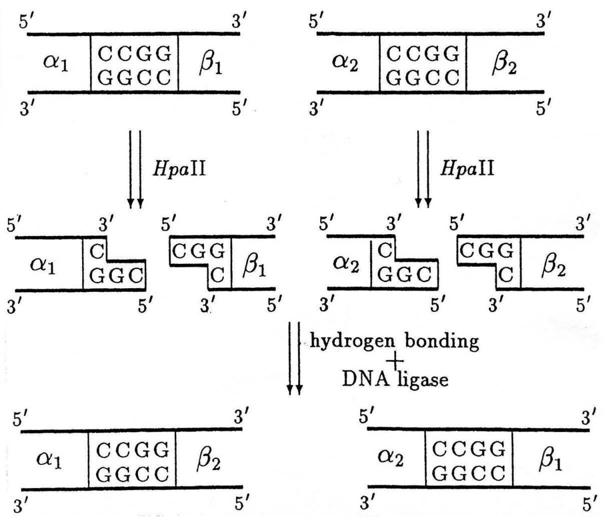


Figure 3.8: Hybridization.

Figure 3.9 presents an example of splicing with the recognition sites *TCGA* and *GCGC*.

```
ACTGCCGGTTTA                       GCACTCGTGATAT
TGAGGCCAAAT                        CGTGAGCACTATA
      ↓                                  ↓
ACTGC  CGGTTTA                     GCACT  CGTGATAT
TGACGGC  CAAAT                     CGTGAGC  ACTATA

                        ↓

 ACTGCCGTGATAT                       GCACTCGGTTTA
 TGACGGCACTATA                       CGTGAGCCAAAT
```

Figure 3.9: Splicing.

3.2 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 Adleman who was one of the first scientists solving a hard problem by easy calculations with DNA molecules.

We consider the well-known *Hamilton path problem*

Instance:	a graph and two nodes v_0 and v_1,
Answer:	Yes, if there is a path containing each node exactly once and starting in v_0 and ending in v_1

Let us consider the graph H shown in Figure 3.10. 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).

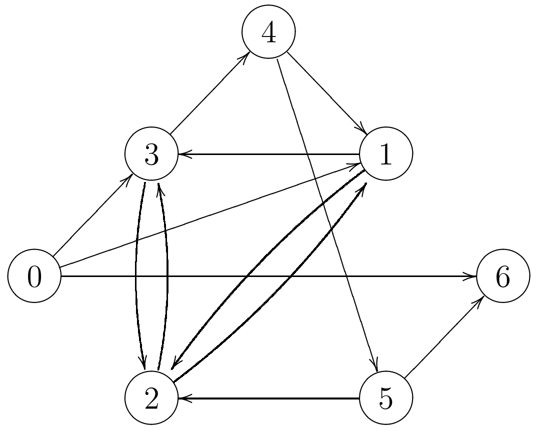


Figure 3.10: Graph whose Hamiltonian path problem is solved by DNA operations by Adleman

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 $GGCTAGGTACGAGCATGCTT$.

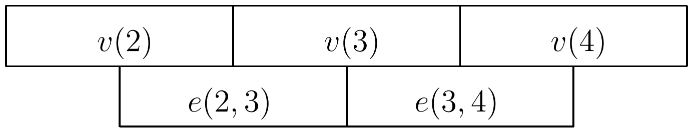
To model the edges we use single lower strands of length 20, too, in their 3'-5' orientation. Because we want to model edges we have to take into them information from the two nodes which are connected. One simple possibility is to take the Watson-Crick complementary of the second half of the strand 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 3.3).

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 which 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; Adleman needs hours to solve the Hamilton path problem for the graph H of Figure 3.10, 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[1], which means that we cannot expect an polynomial algorithm for this problem if we restrict to classical deterministic and sequential algorithms. Moreover, Lipton (see [18]) 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.

[1]For the basic concepts of complexity theory we refer to [6].