Contents

Chapter 5

Linear grammars

5.1 Linear case of ordinary grammars

Linear equations in mathematics: where a variable may only be multiplied by a constant. Linear language equations: a language variable may only be concatenated by a constant language.

Linear ordinary grammars: with rules $A \to uBv$ or $A \to u$.

Lemma 5.1 (The linear pumping lemma)**.** *For every linear ordinary language $L \subseteq \Sigma^*$ there exists a constant $p \geqslant 1$, such that for every string $w \in L$ with $|w| \geqslant p$ there exists a partition $w = xuyvz$, where $|uv| > 0$ and $|xuvz| \leqslant p$, such that $xu^iyv^iz \in L$ for all $i \geqslant 0$.*

Example 5.1. *The Dyck language is not generated by any linear ordinary grammar.*

Proof. Suppose it is. Then, by the linear pumping lemma, there is such a constant p, that the string $w = a^pb^pa^pb^p$ can be factorized as $w = xuyvz$, where $|uv| > 0$ and $|xuvz| \leqslant p$. Because of the latter condition, u must be in the first block a^p and v must be in the last block b^p. Then the pumping lemma asserts that the string $xyz = a^{p-|u|}b^pa^pb^{p-|v|}$ is in L, which is not true, since $|u| > 0$ or $|v| > 0$. ☐

Example 5.2. *The language $\{ a^mb^ma^nb^n \mid m, n \geqslant 0\}$ is not generated by any linear ordinary grammar.*

Therefore, not closed under concatenation.

Theorem 5.1. *The linear languages, as well as the linear unambiguous languages, are closed under intersection with regular languages.*

Proof. By the same construction as before; it preserves linearity of a grammar. ☐

Exercises

5.1.1. Prove that the language $\{ a^mb^{m+n}c^n \mid m, n \geqslant 0\}$ is not generated by any linear ordinary grammar.

5.2 Linear conjunctive grammars and trellis automata

5.2.1 Linear conjunctive grammars

Definition 5.1. *A conjunctive grammar $G = (\Sigma, N, R, S)$ is said to be linear conjunctive, if each rule in R is of one of the following forms.*

$$A \to u_1B_1v_1 \,\&\, \ldots \,\&\, u_mB_mv_m \qquad (m \geqslant 1;\ u_i, v_i \in \Sigma^*;\ B_i \in N)$$
$$A \to w \qquad (w \in \Sigma^*)$$

Example 5.3. *The conjunctive grammar for the language* $\{ a^n b^n c^n \mid n \geqslant 0\}$ *can be rewritten as a linear conjunctive grammar as follows.*

$$\begin{array}{lcl} S & \to & E \,\&\, F \\ E & \to & aE \mid B \\ B & \to & bBc \mid \varepsilon \\ F & \to & Fc \mid D \\ D & \to & aDb \mid \varepsilon \end{array}$$

Definition 5.2. *A linear conjunctive grammar* $G = (\Sigma, N, R, S)$ *is said to be in the linear normal form, if each rule in* R *is of one of the following forms:*

$$\begin{array}{ll} A \to bC_1 \,\&\, \ldots \,\&\, bC_m \,\&\, B_1c \,\&\, \ldots \,\&\, B_nc & (m+n \geqslant 1;\ b, c \in \Sigma;\ B_i, C_j \in N), \\ A \to a & (a \in \Sigma), \\ S \to \varepsilon & (\textit{only if } S \textit{ never occurs in the right-hand sides of any rules}). \end{array}$$

Theorem 5.2. *Every linear conjunctive grammar can be transformed to a grammar in the linear normal form that defines the same language.*

5.2.2 Linear Boolean grammars

Definition 5.3. *A linear Boolean grammar* $G = (\Sigma, N, R, S)$ *is said to be in the linear normal form, if each rule in* R *is of one of the following forms:*

$$A \to bC_1 \,\&\, \ldots \,\&\, bC_m \,\&\, B_1c \,\&\, \ldots \,\&\, B_nc \,\&\, \neg bE_1 \,\&\, \ldots \,\&\, \neg bE_k \,\&\, \neg D_1c \,\&\, \ldots \,\&\, \neg D_\ell c \qquad (m, n, k, \ell \geqslant 0;\ m+n \geqslant 1),$$

$$A \to a \qquad (a \in \Sigma),$$

$$S \to \varepsilon,$$

where the latter rule is allowed only if S *never occurs in the right-hand sides of any rules.*

Theorem 5.3 (Okhotin [7]). *Every linear Boolean grammar can be transformed to a grammar in the linear normal form that defines the same language.*

It shall be proved in the following that linear Boolean grammars define the same family of languages as linear conjunctive grammars.

5.2.3 Trellis automata

The family of languages defined by linear conjunctive grammars and linear Boolean grammars is exactly the same as the family recognized by one of the simplest types of cellular automata: the *one-way real-time cellular automata*, also known under the proper name of *trellis automata*, studied by Dyer [5], Čulík, Gruska and Salomaa [3, 4], Ibarra and Kim [6], and others.

A trellis automaton, defined as a quintuple $(\Sigma, Q, I, \delta, F)$, processes an input string of length $n \geqslant 1$ using a uniform triangular array of $\frac{n(n+1)}{2}$ processor nodes, connected as in Figure 5.1. Each node computes a value from a fixed finite set Q. The nodes in the bottom row obtain their values directly from the input symbols using a function $I \colon \Sigma \to Q$. The rest of the nodes compute the function $\delta \colon Q \times Q \to Q$ of the values in their predecessors. The string is accepted if and only if the value computed by the topmost node belongs to the set of accepting states $F \subseteq Q$.

Formally, the initial function I is extended to map a string of symbols to a string of states, as $I \colon \Sigma^+ \to Q^+$ with $I(a_1 \ldots a_n) = I(a_1) \ldots I(a_n)$, while the function δ is extended to define the outcome of a computation beginning with a string of states, as $\delta \colon Q^+ \to Q$ with $\delta(p\alpha q) = \delta(\delta(p\alpha), \delta(\alpha q))$ for all $p, q \in Q$ and $\alpha \in Q^*$. Then the language recognized by the automaton is defined as $L(M) = \{ w \in \Sigma^+ \mid \delta(I(w)) \in F\}$.

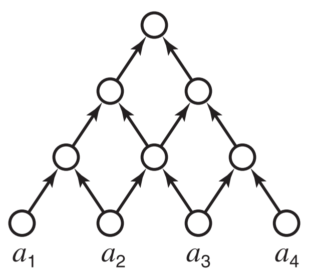


Figure 5.1: The form of a computation of a trellis automaton.

5.2.4 Computational equvalence of the models

Theorem 5.4 (Okhotin [8]). *Let Σ be any alphabet. For every language $L \subseteq \Sigma^+$, the following three statements are equivalent:*

(C). L is defined by a linear conjunctive grammar;

(B). L is defined by a linear Boolean grammar;

(T). L is recognized by a trellis automaton.

The implication ((C) $\Rightarrow$ (B)) is obvious, and it is sufficient to prove that (B) implies (T), and (T) implies (C). Each implication is proved by an effective construction.

Lemma 5.2. *Let $G = (\Sigma, N, R, S)$ be a linear Boolean grammar in the linear normal form that does not generate the empty string, in which the rules of the form*

$$A \to bC_1 \,\&\, \ldots \,\&\, bC_m \,\&\, B_1c \,\&\, \ldots \,\&\, B_nc \,\&\, \neg bE_1 \,\&\, \ldots \,\&\, \neg bE_k \,\&\, \neg D_1c \,\&\, \ldots \,\&\, \neg D_\ell c \tag{5.1a}$$
$$A \to a \tag{5.1b}$$

Construct a trellis automaton $M = (\Sigma, Q, I, \delta, F)$, where $Q = \Sigma \times 2^N \times \Sigma$ and

$$I(a) = (a, \{\, A \mid A \to a \in R\}, a), \tag{5.2a}$$
$$\delta\big((b, X, b'), (c', Y, c)\big) = (b, Z, c), \tag{5.2b}$$

where Z is the set of all such nonterminals $A \in N$, for which there exists a rule (5.1a), with $B_1, \ldots, B_n \in X$, $C_1, \ldots, C_m \in Y$, $D_1, \ldots, D_\ell \notin X$ and $E_1, \ldots, E_k \notin Y$;

$$F = \{\, (a, X, b) \mid X \subseteq N,\ S \in X,\ a, b \in \Sigma\}. \tag{5.2c}$$

Then, for every string $w \in \Sigma^+$, where b is its first symbol and c is its last symbol, the automaton computes the state $\delta(I(w)) = (b, \{\, A \in N \mid w \in L_G(A)\}, c)$, and accordingly, $L(M) = L(G)$.

Proof. Induction on the length of w.
TBW □

Lemma 5.3. *Let $M = (\Sigma, Q, I, \delta, F)$ be a trellis automaton. Construct a linear conjunctive grammar $G = (\Sigma, \{\, A_q \mid q \in Q\} \cup \{S\}, R, S)$, where R contains the following rules:*

$$S \to A_q \qquad \textit{(for all } q \in F\textit{)} \tag{5.3a}$$
$$A_{I(a)} \to a \qquad \textit{(for all } a \in \Sigma\textit{)} \tag{5.3b}$$
$$A_{\delta(q_1,q_2)} \to A_{q_1}c \& bA_{q_2} \qquad \textit{(for all } q_1, q_2 \in Q \textit{ and } b, c \in \Sigma\textit{)} \tag{5.3c}$$

Then, $L_G(A_q) = \{\, w \in \Sigma^+ \mid \delta(I(w)) = q\}$ for every state $q \in Q$, and accordingly, G is unambiguous and $L(G) = L(M)$.

5.3 Examples

Example 5.4 (Dyer [5, Thm. 3], Čulík et al. [3]). *The Dyck language over $\Sigma = \{a, b\}$ is recognized by the trellis automaton $M = (\Sigma, Q, I, \delta, F)$ with the set of states $Q = \{\nearrow, \nwarrow, X, -\}$, with the initial function given by $I(a) = \nearrow$ and $I(b) = \nwarrow$, and with the transition function defined in the following table:*

δ	$\nearrow$	$\nwarrow$	X	$-$
$\nearrow$	$\nearrow$	X	$\nearrow$	$\nearrow$
$\nwarrow$	$-$	$\nwarrow$		$-$
X		$\nwarrow$		$\nearrow$
$-$	$-$	$\nwarrow$	$\nwarrow$	$-$

No other pairs of states will ever be reached, so the rest of the transitions may be set arbitrarily. The set of accepting states is $F = \{X\}$.

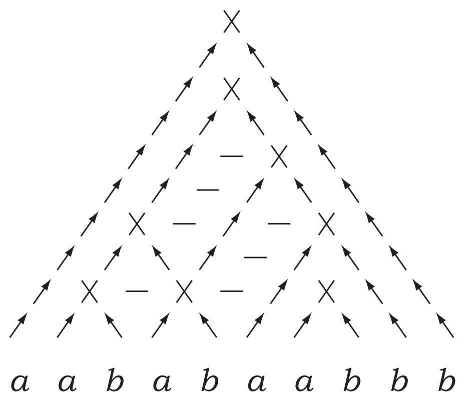


Figure 5.2: A sample computation of the trellis automaton for the Dyck language given in Example 5.4.

The transformation in Lemma 5.3, applied to this automaton, produces a linear conjunctive grammar with more than $|Q|^2 \cdot |\Sigma|^2 = 64$ rules.

Once the unused rules are removed, the grammar takes the following form.

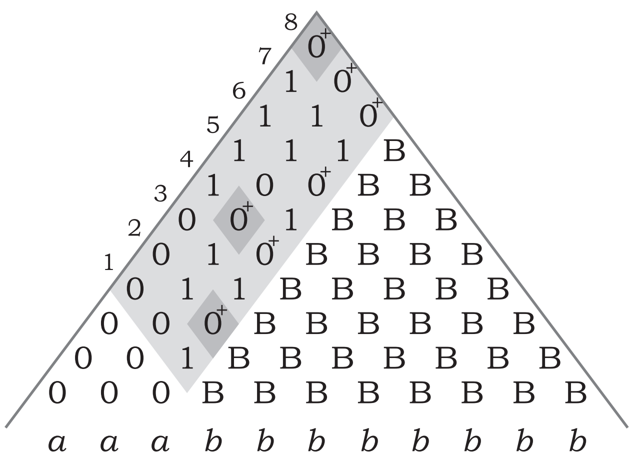
$$\begin{aligned}
&S' \to S \mid \varepsilon \\
&S \to Ab\&aB \\
&A \to Sa\&aX \mid Sb\&aX \mid Ab\&aS \mid Aa\&aA \mid Ab\&aA \mid Aa\&aX \mid Ab\&aX \mid a \\
&B \to Sb\&aB \mid Bb\&aB \mid Bb\&bB \mid Xb\&aB \mid Xb\&bB \mid Xb\&aS \mid Xb\&bS \mid b \\
&X \to Ba\&aA \mid Ba\&bA \mid Bb\&aA \mid Bb\&bA \mid Ba\&aX \mid Ba\&bX \mid \\
&\quad \mid Bb\&aX \mid Bb\&bX \mid Xa\&aA \mid Xa\&bA \mid Xb\&aA \mid Xb\&bA \mid \\
&\quad \mid Xa\&aX \mid Xa\&bX \mid Xb\&aX \mid Xb\&bX
\end{aligned}$$

The next example illustrates the ability of trellis automata to count in positional notation.

Example 5.5 (Ibarra, Kim [6, Ex. 2.1]). *The language $\{\, a^n b^{2^n} \mid n \geqslant 1\}$ is linear conjunctive.*

Before constructing the automaton for Example 5.5, consider another example.

Example 5.6. *The language $\{\, a^n b^{i \cdot 2^n} \mid i, n \geqslant 1\}$ is recognized by the trellis automaton $(\{a, b\}, Q, I, \delta, F)$, where $Q = \{0, 0^+, 1, B, D\}$, $I(a) = 0$, $I(b) = B$, the transition table is given in Figure 5.3 (with all undefined transitions leading to D), and the set of accepting states is $F = \{0^+\}$.*



	0	0^+	1	B
0	0	1	0	1
0^+	0		0	1
1	1	0^+	1	0^+
B				B

Figure 5.3: Trellis automaton for the language $\{\, a^n b^{i \cdot 2^n} \mid i, n \geqslant 1\}$ and its computation.

On a string $a^i b^j$ with $i, j \geqslant 1$, a trellis automaton recognizing this language computes the i-th bit of the base-2 representation of j. Some further states are used to represent the carry digit, and to keep track of whether j is less than, equal to, or greater than 2^i.

The above automaton can be modified to recognize the language $\{\, a^n b^{2^n} \mid n \geqslant 1\}$ as follows:

An actual trellis automaton can be obtained using the set of states $\{0, 1, 0^+\} \times \{t, f\}$, so that the state computed on a string $a^i b^j$ with $i, j \geqslant 1$ has the first component represent the i-th bit of the base-2 representation of j, while its second component determines whether j is less or equal than

Example 5.7. *Let $M = (\{a, b\}, Q, I, \delta, F)$ be the trellis automaton from the previous example and consider the trellis automaton $M' = (\{a, b\}, Q', I', \delta', F)$, in which: $Q' = (\{0, 0^+, 1\} \times \{true, false\}) \cup \{B, D\}$, $I(a) = (0, true)$, $I(b) = B$, $\delta'(B, B) = B$; $\delta'((0, x), B) = (1, x)$, $\delta'((1, x), B) = (0^+, x)$, $\delta'((0^+, x), B) = (1, false)$ for all $x \in \{true, false\}$; $\delta'((q, x), (q', x')) = (\delta(q, q'), x \wedge (q \neq 0^+))$ for all $q, q' \in \{0, 0^+, 1\}$ and $x, x' \in \{true, false\}$, all undefined transitions go to D, and $F = \{(0^+, true)\}$. Then $L(M') = \{\, a^n b^{2^n} \mid n \geqslant 1\}$.*

The first components of all states represent the state of M. The second component is used to remember whether none of the left predecessors of this state were 0^+. An accepting state is 0^+ with no occurrences of 0^+ among its left predecessors.

The language in the following example had once been proposed as a candidate language for having no trellis automaton [6]. Surprisingly, a sophisticated trellis automaton recognizing this language was constructed.

Example 5.8 (Čulík [2])**.** *The language $\{\, a^m b^{m+n} a^n \mid m, n \geqslant 1\}$ is linear conjunctive.*

The construction embeds a cellular automaton solving the *firing squad synchronization problem* (requires a minimum-time solution, as discovered by Goto and later improved by Waksman and by Balzer) into a trellis automaton.

5.4 Limitations of linear grammars

5.4.1 Terrier's lemma

The method of Terrier is based upon a special complexity function of a language, which reflects the amount of variation in the membership status of strings obtained by cutting a small number of symbols from the beginning and from the end of a single string. This complexity function is defined for any language as follows.

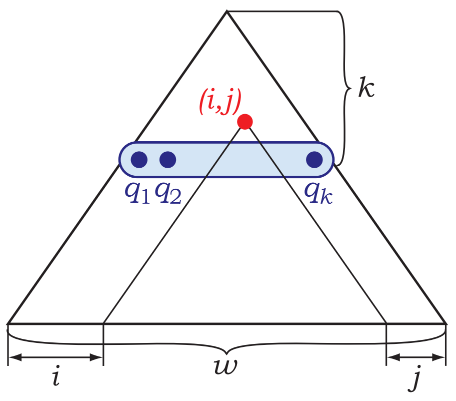


Figure 5.4: The last k rows of a computation of a trellis automaton, and a substring without first i symbols and last j symbols.

Definition 5.4. *Let $L \subseteq \Sigma^*$ be a language, let $k \geqslant 1$ and let $w = a_1 \ldots a_n$ be a string with $n \geqslant k$. Consider the set*

$$S_{L,k,w} = \{\, (i,j) \mid i,j \geqslant 0,\ i+j < k,\ a_{i+1} \ldots a_{n-j} \in L \},$$

which represents the membership in L of all substrings of w longer than $|w| - k$ symbols, where each pair of coordinates (i,j) refers to w without its first i symbols and its last j symbols. For each L and k, consider all possible sets of this form obtained for different strings $w \in \Sigma^{\geqslant k}$. The number of such sets is denoted by

$$f_L(k) = \big|\{\, S_{L,k,w} \mid w \in \Sigma^*,\ |w| \geqslant k \}\big|$$

where $f_L \colon \mathbb{N} \to \mathbb{N}$ is an integer function.

Each set $S_{L,k,w}$ has between 0 and $\frac{k(k+1)}{2}$ elements, and accordingly, the value of the function $f_L(k)$ is between 1 and $2^{\frac{k(k+1)}{2}}$. Note that this cardinality $f(k)$ equals 1 if L contains either all strings of length k or more, or none of these strings.

Assume that a language L is recognized by a trellis automaton, and consider the last k steps of the automaton's computation on any input string, illustrated in Figure 5.4. Then, the automaton is required to discrimitate between $f(k)$ different cases in the last k steps, and must do so on the basis of the information given in the states $q_1, \ldots, q_k$. Since the automaton has finitely many states, it may distinguish between at most $2^{O(k)}$ different situations. This reasoning yields the following upper bound on the growth rate of this function for linear conjunctive languages.

Lemma 5.4 (Terrier [9]). *If $L \in \Sigma^*$ is linear conjunctive, then its complexity measure $f_L(k)$ is bounded by an exponential function, that is, there exists such a number $p \geqslant 1$, that*

$$f_L(k) \leqslant p^k, \qquad \text{for all } k \geqslant 1.$$

Example of a language that maximizes this complexity measure, and therefore is recognized by no trellis automaton.

Example 5.9 (Terrier [9]). *The following language L has $f_L(k) = 2^{\frac{k(k+1)}{2}}$ and therefore is not linear conjunctive.*

$$L = \{\, a^{i_1} b^{j_1} \ldots a^{i_\ell} b^{j_\ell} a^{i_{\ell+1}} b^{j_{\ell+1}} \ldots a^{i_m} b^{j_m} \mid m \geqslant 2;\ i_t, j_t \geqslant 1,\ \exists \ell :\ i_1 = j_\ell \text{ and } i_{\ell+1} = j_m \}$$

This language is representable as a concatenation of a linear ordinary language

$$L_0 = \{\, a^n w b^n \mid n \geqslant 1;\ w \in b\{a,b\}^* a \text{ or } w = \varepsilon \}$$

with itself, as $L = L_0 \cdot L_0$.

Proof. To see that $f_L(k) = 2^{\frac{k(k+1)}{2}}$ for all $k \geqslant 1$, it has to be shown that every set $S \subseteq \{\, (i,j) | i,j \geqslant 0,\ i+j < k\}$ equals S_{L,k,w_S} for some string $w_S \in \Sigma^{\geqslant k}$. Such a string w_S is constructed as

$$w_S = a^k \Big(\prod_{(i,j) \in S} b^{k-i} a^{k-j} \Big) b^k,$$

where the concatenation $\prod_{(i,j)}$ can be taken in any order. Then $a^{k-s}\big(\prod_{(i,j)\in S} b^{k-i}a^{k-j}\big)b^{k-t} \in L$ if and only if $(s,t) \in S$, that is, $S_{L,k,w_S} = S$, which proves that $f_L(k) = 2^{\frac{k(k+1)}{2}}$. □

5.4.2 The case of a one-symbol alphabet

For a one-symbol alphabet, a trellis automaton degrades to a DFA and hence recognizes a regular language.

5.4.3 Periodic behaviour in sequences of states

Theorem 5.5 (Buchholz, Kutrib [1]). *Let $f \colon \mathbb{N} \to \mathbb{N}$ be a function, such that the language $L_f = \{\, a^n b^{f(n)} \mid n \geqslant 1\}$ is linear conjunctive. Then there exists such a constant $p \geqslant 1$, that $f(n) \leqslant p^{n+1}$.*

Proof. TBW. □

Example 5.10. *Languages such as $\{\, a^n b^{2^{2^n}} \mid n \geqslant 1\}$, $\{\, a^n b^{n!} \mid n \geqslant 1\}$, etc., are not linear conjunctive.*

Bibliography

[1] T. Buchholz, M. Kutrib, “On time computability of functions in one-way cellular automata”, *Acta Informatica*, 35:4 (1998), 329–352.

[2] K. Čulík II, “Variations of the firing squad problem and applications”, *Information Processing Letters*, 30:3 (1989), 152–157.

[3] K. Čulík II, J. Gruska, A. Salomaa, “Systolic trellis automata”, I and II, *International Journal of Computer Mathematics*, 15 (1984), 195–212, and 16 (1984), 3–22.

[4] K. Čulík II, J. Gruska, A. Salomaa, “Systolic trellis automata: stability, decidability and complexity”, *Information and Control*, 71 (1986) 218–230.

[5] C. Dyer, “One-way bounded cellular automata”, *Information and Control*, 44:3 (1980), 261–281.

[6] O. H. Ibarra, S. M. Kim, “Characterizations and computational complexity of systolic trellis automata”, *Theoretical Computer Science*, 29 (1984), 123–153.

[7] A. Okhotin, “Boolean grammars”, *Information and Computation*, 194:1 (2004), 19–48.

[8] A. Okhotin, “On the equivalence of linear conjunctive grammars to trellis automata”, *RAIRO Informatique Théorique et Applications*, 38:1 (2004), 69–88.

[9] V. Terrier, “On real-time one-way cellular array”, *Theoretical Computer Science*, 141:1–2 (1995), 331–335.

[10] V. Terrier, “Closure properties of cellular automata”, *Theoretical Computer Science*, 352:1–3 (2006), 97–107.

[11] S. Yu, “A property of real-time trellis automata”, *Discrete Applied Mathematics*, 15:1 (1986), 117–119.

Index