

Formal grammars

Lectures ??–??. Linear grammars and trellis automata

Alexander Okhotin

April 17, 2009

Abstract

The linear case of context-free, conjunctive and Boolean grammars. Trellis automata. Examples. Equivalence of grammars and automata.

1 Linear Boolean grammars

Definition 1. A Boolean grammar $G = (\Sigma, N, P, S)$ is said to be **linear** if every rule

$$A \rightarrow \alpha_1 \& \dots \& \alpha_m \& \neg \beta_1 \& \dots \& \neg \beta_n,$$

has $\alpha_i, \beta_j \in \Sigma^* \cup (\Sigma \cup N)^*$. If a linear Boolean grammar is conjunctive (context-free), it is called a linear conjunctive (linear context-free) grammar.

The following normal form similar to the binary normal form can be defined for linear Boolean grammars.

Definition 2. A linear Boolean grammar $G = (\Sigma, N, P, S)$ is said to be in the linear normal form if every rule in P is of the form

$$A \rightarrow bB_1 \& \dots \& bB_m \& C_1 c \& \dots \& C_n c \& \neg bD_1 \& \dots \& \neg bD_k \& \\ \& \neg E_1 c \& \dots \& \neg E_\ell c \quad (m, n, k, \ell \geq 0; m + n \geq 1),$$

$$A \rightarrow a$$

$$S \rightarrow \varepsilon \quad (\text{only if } S \text{ does not appear in right hand sides of rules})$$

Theorem 1. For every linear Boolean grammar there exists a linear Boolean grammar in the linear normal form generating the same language. If the grammar is linear conjunctive (linear context-free), then so is the resulting grammar in the linear normal form.

The construction is basically the same as used to transform a Boolean grammar to the binary normal form. Note that the given methods for eliminating ε conjuncts and unit conjuncts preserve linearity of a grammar.

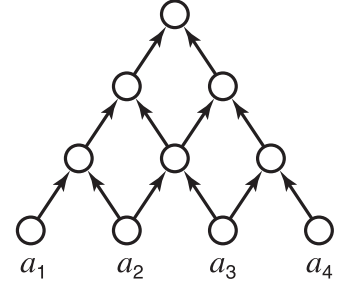
Motivation for TA: consider the basic cubic-time recognition algorithm for linear Boolean grammars, etc., etc.

2 Trellis automata

Trellis automata can be equally defined by their cellular automata semantics (using evolution of configurations) and through the trellis representing their computation. According to the latter approach, due to Culik et al. [3], a trellis automaton processes an input string of length $n \geq 1$ using a uniform triangular array of $\frac{n(n+1)}{2}$ processor nodes, as presented in the figure below. 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 $I : \Sigma \rightarrow Q$. The rest of the nodes compute the function $\delta : Q \times Q \rightarrow Q$ 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 $F \subseteq Q$. This is formalized in the following definition.

Definition 3. A trellis automaton is a quintuple $M = (\Sigma, Q, I, \delta, F)$, in which:

- Σ is the input alphabet,
- Q is a finite non-empty set of states,
- $I : \Sigma \rightarrow Q$ is a function that sets the initial states,
- $\delta : Q \times Q \rightarrow Q$ is the transition function, and
- $F \subseteq Q$ is the set of final states.



The result of the computation on a string $w \in \Sigma^+$ is denoted by $\Delta : \Sigma^+ \rightarrow Q$, which is defined inductively as $\Delta(a) = I(a)$ and $\Delta(awb) = \delta(\Delta(aw), \Delta(wb))$, for any $a, b \in \Sigma$ and $w \in \Sigma^*$. Then the language recognized by the automaton is $L(M) = \{w \mid \Delta(w) \in F\}$.

3 Equivalence

Let $G = (\Sigma, N, P, S)$ be a linear Boolean grammar in the linear normal form. Construct the trellis automaton $M = (\Sigma, Q, I, \delta, F)$, where $Q = \Sigma \times 2^N \times \Sigma$ and

$$\begin{aligned}
 I(a) &= (a, \{A \mid A \rightarrow a \in P\}, a), \\
 \delta((b, T_1, b'), (c', T_2, c)) &= (b, \{A \mid \text{there is a long rule } \text{, such that} \\
 &\quad B_i \in T_2, C_j \in T_1, D_s \notin T_2, E_t \notin T_1 \text{ for all } i, j, s, t\}, c), \\
 F &= \{(a, R, b) \mid S \in R\}.
 \end{aligned}$$

The correctness of this construction is stated in the following lemma:

Lemma 1. Let $w \in \Sigma^+$ and let $\delta(I(w)) = (b, T, c)$. Then the first symbol of w is b , the last symbol of w is c , and for each nonterminal $A \in Q$, $w \in L_G(A)$ if and only if $A \in T$.

The proof is by induction on the length of w . From this lemma it is easy to see that, for every $w \in \Sigma^+$, $\delta(I(w)) \in F$ if and only if $w \in L_G(S) = L(G)$.

Theorem 2. For every linear Boolean grammar G there exists and can be effectively constructed a trellis automaton M with $L(M) = L(G) \setminus \{\varepsilon\}$.

Conversely, every trellis automaton can be simulated by a linear conjunctive grammar as follows. Let $M = (\Sigma, Q, I, \delta, F)$ be an arbitrary trellis automaton. Construct the grammar $G = (\Sigma, N, P, S)$, where $N = \{A_q \mid q \in Q\} \cup \{S\}$ and P contains the following rules:

$$\begin{aligned} S &\rightarrow A_q \quad (q \in F) \\ A_{I(a)} &\rightarrow a \quad (a \in \Sigma) \\ A_{\delta(q_1, q_2)} &\rightarrow A_{q_1} c \&b A_{q_2} \quad (q_1, q_2 \in Q, b, c \in \Sigma) \end{aligned}$$

The following lemma states the correctness of the construction.

Lemma 2. *For every string $w \in \Sigma^+$ and for every state $q \in Q$, $w \in L_G(A_q)$ if and only if $\delta(I(w)) = q$.*

Proved by induction on $|w|$.

Theorem 3. *For every trellis automaton M there exists and can be effectively constructed a linear conjunctive grammar G with $L(G) = L(M)$.*

These two theorems, in particular, imply that linear conjunctive grammars and linear Boolean grammars generate the same family of languages.

4 Examples

Consider the *Dyck language* over $\{a, b\}$, representing the set of all strings with balanced brackets. It is generated by the context-free grammar $S \rightarrow SS \mid aSb \mid \varepsilon$, which is not linear.

Example 1 (cf. Dyer [2], Čulik 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 $Q = \{\nearrow, \searrow, X, _ \}$, with the initial function given by $I(a) = \nearrow$ and $I(b) = \searrow$, and with the transition function is defined as follows: $\delta(\nearrow, \searrow) = X$, $\delta(\nearrow, q) = \nearrow$ (for all $q \neq X$), $\delta(q, \searrow) = \searrow$ (for all $q \neq X$), $\delta(q, _) = \searrow$, $\delta(X, _) = \nearrow$, $\delta(_, X) = \searrow$. No other states will ever be reached, so the rest of the transitions may be set arbitrarily. The set of accepting states is $F = \{X\}$.*

The transition table of this automaton and a sample computation on the string $w = aabaababbaabb$ are depicted in Figure 1 (cf. Dyer [2, Fig.5]).

It is known from Ibarra and Kim [5] (**check**) that the language $\{a^n b^{2^n} \mid n \geq 1\}$ is recognized by a trellis automaton.

Example 2. *The language $\{a^n b^{i \cdot 2^n} \mid i, n \geq 1\}$ 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 2 (with all undefined transitions leading to D), and the set of accepting states is $F = \{0^+\}$.*

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

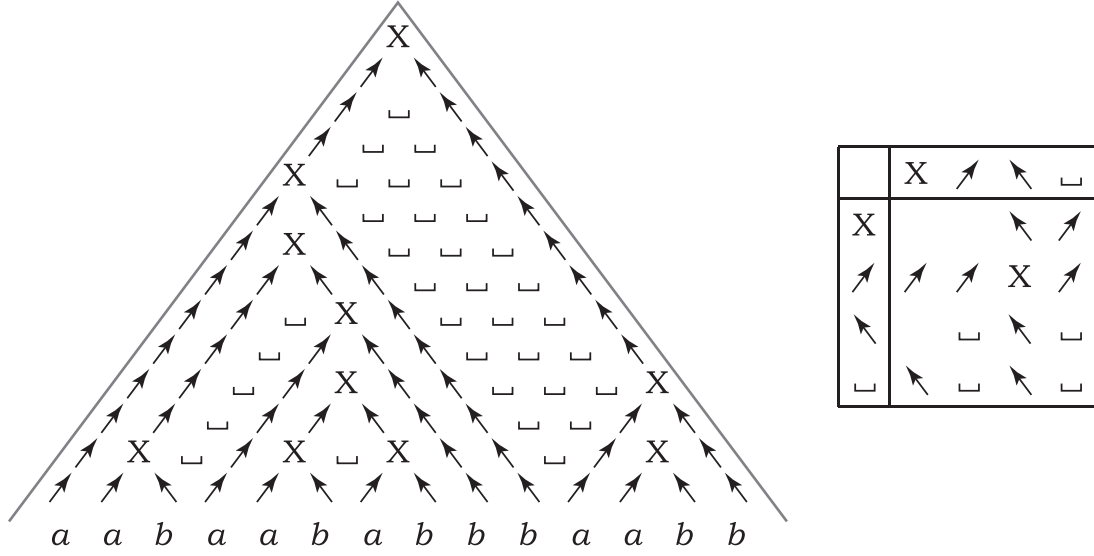


Figure 1: Trellis automaton for the Dyck language and its computation.

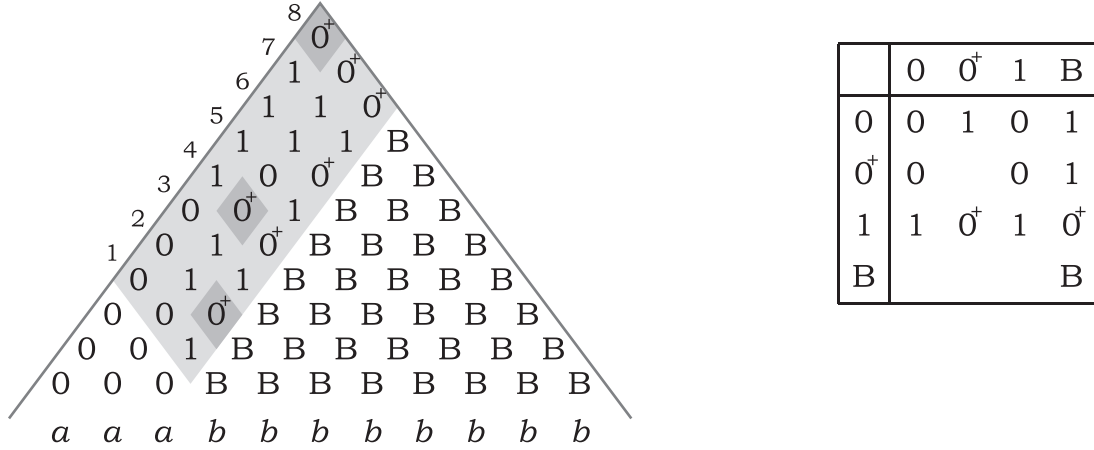


Figure 2: Trellis automaton for the language $\{a^n b^{i \cdot 2^n} \mid i, n \geq 1\}$ and its computation.

Example 3. 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: $Q' = (\{0, 0^+, 1\} \times \{\text{true}, \text{false}\}) \cup \{B, D\}$, $I(a) = (0, \text{false})$, $I(b) = B$, $\delta'(B, B) = B$; $\delta'((0, x), B) = (1, x)$, $\delta'((1, x), B) = (0^+, x)$, $\delta'((0^+, x), B) = (1, \text{true})$ for all $x \in \{\text{true}, \text{false}\}$; $\delta'((q, x), (q', x')) = (\delta(q, q'), x \vee (q = 0^+))$ for all $q, q' \in \{0, 0^+, 1\}$ and $x, x' \in \{\text{true}, \text{false}\}$, all undefined transitions go to D , and $F = \{(0^+, \text{false})\}$. Then $L(M') = \{a^n b^{2^n} \mid n \geq 1\}$.

The first components of all states represent the state of M . The second component is used to remember whether any of the left predecessors of this state was 0^+ . An accepting state is 0^+ with no occurrences of 0^+ among its left predecessors.

5 A trellis automaton for $\{a^m b^{m+n} a^n \mid m, n \geq 1\}$

The idea of Čulik [7] of embedding a cellular automaton solving the *firing squad problem* [8] into a trellis automaton. ***TBW***

6 Limitations of linear context-free grammars

Lemma 3 (Pumping lemma for linear context-free languages). *For every linear context-free language $L \subseteq \Sigma^*$ there exists a constant $p \geq 1$, such that for every string $w \in L$ with $|w| \geq p$ there exists a factorization $w = xuyvz$, where $|uv| > 0$ and $|xuvz| \leq p$, such that $xu^i y v^i z \in L$ for all $i \geq 0$.*

Example 4. *The Dyck language is not linear context-free.*

Suppose it is linear context-free. Then, by the pumping lemma, there is a constant p , such that the string $w = a^p b^p a^p b^p$ can be factorized as $w = xuyvz$, where $|uv| > 0$ and $|xuvz| \leq p$. 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 $xyz = a^{p-|u|} b^p a^p b^{p-|v|}$ is in L , which is not true, since $|u| > 0$ or $|v| > 0$.

Example 5. *The language $\{a^m b^{m+n} a^n \mid m, n \geq 1\}$ is not linear context-free.*

The same proof as above, using $w = a^p b^{2p} a^p$.

Theorem 4. *Linear context-free languages are closed under union. They are not closed under intersection, complementation, concatenation and star.*

Sketch of a proof. Closure under union: immediate, as in the general context-free case. Intersection: $\{a^n b^n c^n \mid n \geq 0\}$ is an intersection of two linear context-free languages. Complementation: $\{a^n b^n c^n \mid n \geq 0\}$ is a complement of a linear context-free language. Concatenation: $\{a^m b^{m+n} a^n \mid m, n \geq 1\} = \{a^m b^m \mid m \geq 1\} \cdot \{b^n a^n \mid n \geq 1\}$. Star: $\{a^n b^n \mid n \geq 1\}^*$ is not linear context-free. \square

7 Limitations of trellis automata

Example 6. *The language $\{a^n b^{i \cdot n} \mid n, i \geq 1\}$ is not linear conjunctive.*

Proof. Assume there is a trellis automaton recognizing this language, let Q be its set of states. For every $i \geq 0$, consider the following sequence of states:

$$\delta(I(a^i b)), \delta(I(a^i b^2)), \dots, \delta(I(a^i b^j)), \dots$$

The sequence corresponding to $i = 0$ is ultimately periodic with a period π_0 with $1 \leq \pi_0 \leq |Q|$. Every next sequence has a period $\pi_i = \pi_{i-1} \cdot k_i$ where $1 \leq k_i \leq |Q|$.

Now let p be any prime greater than $|Q|$. From the above, the sequence

$$\delta(I(a^p b)), \delta(I(a^p b^2)), \dots, \delta(I(a^p b^j)), \dots$$

is ultimately periodic with the period $\pi_p = \pi_0 \cdot k_1 \cdot \dots \cdot k_p$. On the other hand, since a state $\delta(I(a^p b^j))$ must be accepting if and only if $j = 0 \pmod{p}$, the period π_p must be divisible by p . Since p is a prime and all prime factors of π_p are smaller than p , this yields a contradiction. \square

Theorem 5. Let $f: \mathbb{N} \rightarrow \mathbb{N}$ be a function, such that the language $L_f = \{ a^n b^{f(n)} \mid n \geq 1 \}$ is linear conjunctive. Then there exists a constant $p \geq 1$, such that $f(n) \leq p^{n+1}$.

Proof. TBW. □

Example 7. Languages such as $\{ a^n b^{2^{2^n}} \mid n \geq 1 \}$, $\{ a^n b^{n!} \mid n \geq 1 \}$, etc. are not linear conjunctive.

Definition 4. Let $L \subseteq \Sigma^*$ be a language, let $k \geq 1$ and let $w = a_1 \dots a_n$ be a string with $n \geq k$. Define a set

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

which represents the membership in L of all substrings of w longer than $|w| - k$ symbols.

Next, define the set of all sets $S_{L,k,w}$ for all strings w :

$$\widehat{S}_{L,k} = \{ S_{L,k,w} \mid w \in \Sigma^*, |w| \geq k \}$$

The cardinality of this set is between 1 (if $L \cap \Sigma^+$ is \emptyset or Σ^+) and $2^{\frac{k(k+1)}{2}}$ (if $\widehat{S}_{L,k}$ contains all subsets of $\{ (i, j) \mid i, j \geq 0, i + j < k \}$).

Let $f_L(k) = |\widehat{S}_{L,k}|$.

Theorem 6. If $L \in \Sigma^*$ is linear conjunctive, then there exists a number p , such that

$$f_L(k) \leq p^k.$$

Proof. TBW. □

Example 8. Consider the linear context-free language

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

Then the language

$$L = L_0 \cdot L_0 = \{ a^{i_1} b^{j_1} \dots a^{i_m} b^{j_m} \mid m \geq 2; i_t, j_t \geq 1, \exists \ell : i_1 = j_\ell \text{ and } i_{\ell+1} = j_m \}$$

has $f_L(k) = 2^{\frac{k(k+1)}{2}}$ and therefore is not linear conjunctive.

Proof. To see that $f_L(k) = 2^{\frac{k(k+1)}{2}}$ for all $k \geq 1$, consider every set $S \subseteq \{ (i, j) \mid i, j \geq 0, i + j < k \}$ and construct the string

$$w_S = a^p \left(\prod_{(i,j) \in S} b^{p-i} a^{p-j} \right) b^p,$$

where the concatenation $\prod_{(i,j)}$ can be taken in any order. Then $a^{p-s} \left(\prod_{(i,j) \in S} b^{p-i} a^{p-j} \right) b^{p-t} \in L$ if and only if $(s, t) \in S$, that is, $S_{L,k,w_S} = S$, which proves that $\widehat{S}_{L,k} = 2^{\{(i,j) \mid i,j \geq 0, i+j < k\}}$. □

Theorem 7. Linear conjunctive languages are closed under all Boolean operations. They are not closed under concatenation and star.

Annotated bibliography

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

Linear Boolean grammars. Transformation to normal form. Simulation by trellis automata.

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

Idea of constructing a trellis automaton for the Dyck language.

- [3] K. Čulik 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.

Definition of trellis automata (equivalent to Dyer’s, obtained independently).

- [4] K. Čulik II, J. Gruska, A. Salomaa, “Systolic trellis automata: stability, decidability and complexity”, *Information and Control*, 71 (1984), 218–230.

...

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

Idea of constructing a trellis automaton for $\{a^n b^{2^n} \mid n \geq 1\}$.

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

Proof of equivalence, examples of conversion in both directions.

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

The idea of constructing a trellis automaton for $\{a^m b^{m+n} a^n \mid m + n \geq 1\}$ by embedding a cellular automaton for the firing squad problem.

- [8] R. Balzer, “An 8-state minimal time solution to the firing squad synchronization problem”, *Information and Control*, 10:1 (1967), 22–42.

A solution to the firing squad problem (one of possible solutions) that is embedded in a trellis automaton.

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

A proof that $\{a^n b^{i^n} \mid n, i \geq 1\}$ is not recognized by any trellis automaton.

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

A proof that if a language $\{a^n b^{f(n)} \mid n \geq 1\}$ is recognized by a trellis automaton, then f is bounded by an exponential function.

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

Theorem 6 and the proof that no trellis automaton recognizes $(\{a^n w b^n \mid n \geq 1; w \in b\{a, b\}^ a \text{ or } w = \varepsilon\})^2$.*