UMC205: Automata and Computability

Naman Mishra

January 2024

Contents

Lecture 12: Thu 08 Feb '24

Chapter 1

Context-Free Grammars

1.1 Introduction

The syntax of regular expressions over an alphabet $\{a, b\}$ is defined by

$$r ::= \varnothing \mid a \mid b \mid r + r \mid r \cdot r \mid r^*.$$

This is an example of a context-free grammar (CFG). Context-free grammars arise enaturally in syntax of programming language, parsing, compiling.

Examples.

- G_1 is the grammar given by the rules

$$S \to aX$$
$$X \to aX$$
$$X \to bX$$
$$X \to b$$

A string is derived from these rules as follows: Begin with S and keep rewriting the current string by replacing a non-terminal with the right-hand side in a production rule. For example,

$$S \to aX \to abX \to abb.$$

The language defined by a grammer G, written $L(G)$, is the set of all terminal strings that can be generated by G.
The language generated by G_1 is $a(a + b)^*b$.

- G_2 is the grammar given by the rules

$$S \to aSb$$
$$S \to \epsilon$$

An example string in this grammer is

$$S \to aSb \to aaSbb \to aaaSbbb \to aaabbb$$

and $L(G) = \{a^n b^n \mid n \geq 0\}$. Suppose we add the rule $S \to bSa$. We write this in short as

$$S \to aSb \mid bSa \mid \epsilon.$$

The language generated by this grammar is all even length "inverse" palindromes, *i.e.*, the mirror image of any alphabet about the midpoint inverts a's to b's and vice versa.

- Let G_3 be given be

$$S \to aSa \mid bSb \mid a \mid b \mid \epsilon.$$

Then $L(G_3)$ is all palindromes.

- Let G_4 be given by

$$S \to (S) \mid SS \mid \epsilon.$$

This gives the language of all balanced parentheses. Formally, a $w \in \{(,)\}$ is balanced if

  - $\#_(w) = \#_)(w)$, and
  - for each prefic u of w, $\#_(u) \geq \#_)(u)$.

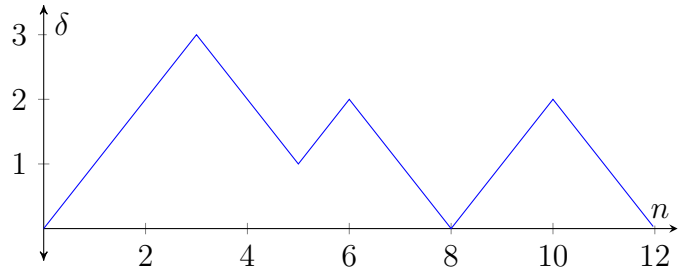
Exercise 1.1. Derive the string (((())())()()) from G_4.

Solution.

$$\begin{aligned} S &\to (S) \\ &\to (SS) \\ &\to ((S)S) \\ &\to ((SS)SS) \\ &\to (((S)(S))(S)(S)) \\ &\to (((())())()()) \end{aligned}$$

One can visualise any balanced string w as a graph of points (n, δ), where $0 \leq n \leq |w|$ and δ is the difference between the number of left

and right parentheses in the prefix of w of length n. The graph starts at $(0, 0)$ and ends at $(|w|, 0)$, and never goes below the x-axis.



Definition 1.2 (Context-free grammar)**.** A Context-Free Grammar (CFG) is a 4-tuple $G = (N, A, S, P)$, where

- N is a finite set of *non-terminal symbols*,
- A is a finite set of *terminal symbols* disjoint from N,
- $S \in N$ is the non-terminal *start symbol*, and
- P is a finite subset of $N \times (N \cup A)^*$, called the set of *productions* or *rules*. A production (X, α) is written as $X \to \alpha$.

We will denote letters with lower-case letters, non-terminals with upper-case letters, and strings of both letters and non-terminals with Greek letters.

Definition 1.3. Given a CFG $G = (N, A, S, P)$, we define the relation $\overset{n}{\Rightarrow}$ on $(N \cup A)^*$ inductively as follows:

- $\alpha \overset{0}{\Rightarrow} \alpha$;
- $\alpha \overset{1}{\Rightarrow} \beta$ if α is of the form $\alpha_1 X \alpha_2$ and $X \to \gamma$ is a production rule such that $\beta = \alpha_1 \gamma \alpha_2$; and
- $\alpha \overset{n+1}{\Longrightarrow} \beta$ if there exists a string γ such that $\alpha \overset{n}{\Rightarrow} \gamma$ and $\gamma \overset{1}{\Rightarrow} \beta$.

We further define $\Rightarrow_G^*$ as the union of all $\overset{n}{\Rightarrow}$ for $n \in \mathbb{N}$.

A *sentential form* of G is any $\alpha \in (N \cup A)^*$ such that $S \Rightarrow_G^* \alpha$.

The language defined by G is $L(G) = \{w \in A^* \mid S \Rightarrow_G^* w\}$.

1.2 Parse Trees

Each derivation of a string w from a CFG G can be represented by a parse tree, where each internal node is labelled by a non-terminal and each leaf is labelled by a terminal or ϵ. Each internal node has children corresponding to the right-hand side of a production rule for the non-terminal label of the node.

The string represented by a parse tree is the concatenation of the labels of the leaves read from left to right.

Exercise 1.4. Consider G_1 given in the examples above.

$$S \to aX$$
$$X \to aX \mid bX \mid b$$

Prove that $L(G_1) = a(a+b)^*b$.

Proof. Let $P(n)$ be that for any $w \in (N \cup A)^*$, $S \overset{n}{\Rightarrow} w$ if and only if

$$w \in a(a+b)^{n-1}X + a(a+b)^{n-2}b,$$

where we consider the second term to be $\varnothing$ if $n < 2$. The case $n = 1$ is direct.

Suppose $P(k)$ holds. Let $w \in (N \cup A)^*$. Then $S \overset{k+1}{\Longrightarrow} w$ iff there is some α such that $S \overset{k}{\Rightarrow} \alpha$ and $\alpha \overset{1}{\Rightarrow} w$. By the induction hypothesis, this is iff $\alpha \in a(a+b)^{k-1}X + a(a+b)^{k-2}b$ and $\alpha \overset{1}{\Rightarrow} w$. Since the second term has no non-terminals, this is equivalent to $\alpha \in a(a+b)^{k-1}X$ and $\alpha \overset{1}{\Rightarrow} w$ so

$$\begin{aligned} w &\in a(a+b)^{k-1}aX + a(a+b)^{k-1}bX + a(a+b)^{k-1}b \\ &= a(a+b)^kX + a(a+b)^{k-1}b \end{aligned}$$

Thus $P(k+1)$ holds.

By induction, $P(n)$ holds for all $n \in \mathbb{N}$. Then $L(G_1) = \{w \in A^* \mid S \Rightarrow_G^* w\}$ is the union of $a(a+b)^{n-2}b$ which is $a(a+b)^*b$. □

1.3 Chomsky Normal Form

Lecture 13: Tue 13 Feb '24

Definition 1.5 (Chomsky normal form)**.** A context-free grammar G is said to be in *Chomsky normal form* if all of its production rules are of the form

$$X \to YZ$$
$$X \to a$$

where Y and Z are non-terminals, and a is a terminal.

Example. Consider G_4 in the examples above, which generates balanced parentheses.

$$S \to (S) \mid SS \mid \epsilon$$

This can be converted into a CNF as follows

$$L \to ($$
$$R \to)$$
$$S \to LR \mid SS \mid LX$$
$$X \to SR$$

Why do we care about Chomsky normal form? Suppose we have a context-free grammar G and a string w. We want to know if $w \in L(G)$. This is hard to do in general, but it is trivial if G is in CNF. Any production rule applied to an intermediate string w cannot decrease the length of w, so we will know to terminate in finitely many steps.

Theorem 1.6. Every context-free grammar G can be converted into a Chomsky normal form grammar G' such that $L(G') = L(G) \setminus \{\epsilon\}$.

Choose any problematic production rule in G. If the RHS has more than two (say n) non-terminals, we can introduce a new non-terminal in place of $n-1$ of them, from which we generate those $n-1$ non-terminals in sequence.
If the RHS has more than one terminal, we can introduce a new non-terminal for each of those, and we have just shown how to deal with that case.
In fact, if the RHS is of length at least 2, we can replace its terminals with non-terminals.
Thus the only problematic case is when the RHS is either a single terminal, a single non-terminal, or the empty string.

Theorem 1.7. Let G be a context-free grammar. Then there is a context-free grammar G' such that $L(G') = L(G)$ and G' has no unit- or ϵ-productions.

We give an algorithm to achieve this, and will prove its correctness later.

Let $G = (N, A, S, P)$ be a context-free grammar. Create a new set of productions P' as follows: First add all the productions from P to P'. Then,

- if P has productions $X \to \alpha Y \beta$ and $Y \to \epsilon$, add the rule $X \to \alpha\beta$ to P'.
- if $X \to Y$ and $Y \to \gamma$, add $X \to \gamma$ to P'.

This gives us a new grammar $G' = (N, A, S, P')$. Finally, drop all unit- and ϵ-productions from P' to get P''. Then $G'' = (N, A, S, P'')$ is an "equivalent" grammar without unit- or ϵ-productions. Equivalence is in the sense that $L(G') = L(G) \setminus \{\epsilon\}$.

Example. We apply this to G_4 from above. The inital grammar is

$$S \to (S) \mid SS \mid \epsilon.$$

We can apply the first rule to add the production

$$S \to ().$$

There are no more productions to add, so we remove the ϵ-production to get the grammar

$$S \to () \mid (S) \mid SS.$$

Exercise 1.8. Put the following grammar into CNF.

$$S \to aSbb \mid T$$
$$T \to bTaa \mid S \mid \epsilon$$

Solution. By rule 2, we can add all productions of S to T, and vice versa. This gives the grammar

$$S, T \to aSbb \mid T \mid bTaa \mid S \mid \epsilon$$

By rule 1, we can add the productions

$$S \to abb \quad \text{since } S \to aSbb \text{ and } S \to \epsilon$$
$$S \to baa \quad \text{since } S \to bTaa \text{ and } T \to \epsilon$$

And add both of these to T as well. This gives the grammar

$$S, T \to aSbb \mid T \mid bTaa \mid S \mid \epsilon \mid abb \mid baa.$$

We cannot add any more productions, so we have our grammar G'. Dropping unit- and ϵ-productions gives us G'' as

$$S \to aSbb \mid bTaa \mid abb \mid baa$$
$$T \to aSbb \mid bTaa \mid abb \mid baa$$

We can obviously omit T and replace it with S to get

$$S \to aSbb \mid bSaa \mid abb \mid baa$$

We now prove that the algorithm terminates with a correct grammar.

Termination. The algorithm terminates because any new production added has an RHS that is a subsequence of the RHS of an original production. Only finitely many such subsequences exist. □

Correctness. We first show that $L(G') = L(G)$. Let G'_i be the grammar after the ith iteration of the algorithm. Clearly $L(G'_0) = L(G)$. It is easy to see that $L(G'_{i+1}) = L(G'_i)$. Thus $L(G') = L(G)$.

We need to show that $L(G'') = L(G') \setminus \{\epsilon\}$. We do this by first showing that for any $w \in L(G') \setminus \{\epsilon\}$, any minimal length derivation of $w \in G'$ does not use unit- or ϵ-productions.

Suppose the derivation uses an ϵ-production $Y \to \epsilon$. Since $w \neq \epsilon$, this cannot be the first step. So the derivation looks like

$$S \overset{l}{\Rightarrow} \alpha X \beta \overset{1}{\Rightarrow} \alpha\alpha' Y \beta'\beta \overset{m}{\Longrightarrow} \gamma Y \delta \overset{1}{\Rightarrow} \gamma\delta \overset{n}{\Rightarrow} w$$

where $\alpha\alpha' \rightsquigarrow \gamma$ and $\beta'\beta \rightsquigarrow \delta$. But then we can give a shorter derivation

$$S \overset{l}{\Rightarrow} \alpha X \beta \overset{1}{\Rightarrow} \alpha\alpha'\beta'\beta \overset{m}{\Longrightarrow} \gamma\delta \overset{n}{\Rightarrow} w.$$

Similarly, a derivation which contains a unit-production

$$S \overset{l}{\Rightarrow} \alpha X \beta \overset{1}{\Rightarrow} \alpha Y \beta \overset{m}{\Longrightarrow} \gamma Y \delta \overset{1}{\Rightarrow} \gamma\phi\delta \overset{n}{\Rightarrow} w$$

can be shortened to

$$S \overset{l}{\Rightarrow} \alpha X \beta \overset{1}{\Rightarrow} \alpha\phi\beta \overset{m}{\Longrightarrow} \gamma\phi\delta \overset{n}{\Rightarrow} w.$$

This proves that for any $w \in L(G')\setminus\{\epsilon\}$, $w \in L(G'')$. Thus $L(G')\setminus\epsilon \subseteq L(G'')$ and since $P'' \subseteq P'$, $L(G'') \subseteq L(G)$. All that remains to be shown is that

$L(G'')$ does not contain ϵ.

Since each production in P'' (weakly) increases the length of any intermediate string, no derivation in G'' can produce ϵ. □

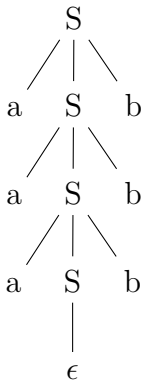
Lecture 14: Thu 15 Feb '24

1.4 Pumping Lemma

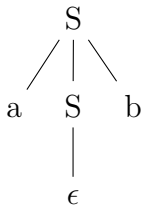
Theorem 1.9 (Pumping lemma for CFLs)**.** For every CFL L there is a constant $k \geq 0$ such that such that for any word z in L of length at least k, there are strings u, v, w, x, y such that

- $z = uvwxy$,
- $vx \neq \epsilon$,
- $|vwx| \leq k$, and
- for each $i \geq 0$, the string uv^iwx^iy belongs to L.

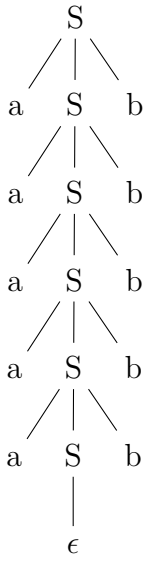
Consider a parse tree for any string. Note that subtrees hanging at the same non-terminal can be replaced by each other. For example, if



is a derivation, then so are



and



Proof. Let $G = (N, A, S, P)$ be a CNF grammar for L. A full binary tree of height h has 2^h leaves. A parse tree in G with height h has a terminal string of length at most 2^{h-1}, since a terminal node has no sibling. Thus a string of length 2^n or more, must have a parse tree of at least height $n+1$. Let $k = 2^{|N|}$. Consider a parse tree in G of a string z of length at least k. The longest path from the root to a leaf has length at least $|N|+1$, and thus has at least $|N|+2$ nodes, or $|N|+1$ non-terminal nodes. By the pigeonhole principle, two of these nodes must have the same label. Let X be the lowest repeated non-terminal, and let $X_\perp$ and $X^\top$ be the two lowest occurrences of X, with $X_\perp$ closer to the leaves.

Let w be the string of terminals derived from $X_\perp$, and let vwx be the string of terminals derived from $X^\top$. Let $z = uvwxy$. Since the longest path from $X^\top$ down to a leaf has length at most $|N|+1$, we have $|vwx| \leq 2^{|N|} = k$.

One of the strings v and x must be non-empty, since G is CNF ($X^\perp$ must have a sibling, which must lead to a terminal). We can then replace the subtree at $X^\top$ by the subtree at $X_\perp$, and obtain a parse tree for uwy. We can also replace the subtree at $X^\perp$ by the subtree at $X_\top$, and obtain a parse tree for uv^2wx^2y.

Continuing in this way, we can obtain a parse tree for uv^iwx^iy for any $i \geq 0$. □

	Closed?
Union	✓
Intersection	✗
Complement	✗
Concatenation	✓

Exercise 1.10. Show that the following languages are not context-free.

- $\{a^n b^n c^n \mid n \geq 0\}$.
- $\{ww \mid w \in \{a, b\}^*\}$.

Solution.

- Let k be the pumping lemma constant. Then $a^k b^k c^k$ is in the language, and so there are strings u, v, w, x, y such that $uvwxy = a^k b^k c^k$, $vx \neq \epsilon$, $|vwx| \leq k$, and uv^2wx^2y is in the language. Since $|vwx| \leq k$, it cannot contain all three letters. Since $vx \neq \epsilon$, uwy will have more of the letters that vwx does not contain, and less of the letters it does contain.
- Suppose it is. Let k be the pumping lemma constant. Then

  $$a^{k+1}b^{k+1}a^{k+1}b^{k+1}$$

  is in the language, and so there are strings u, v, w, x, y as above. Since $|vwx| \leq k$, it cannot overlap with 3 or 4 of the homogenous blocks.

  Thus uwx will still be of the form $a^p b^q a^r b^s$, with none of p, q, r, or s being zero. Since uwy is in the language, $(p, q) = (r, s)$. But removing v and x will change the number of as and bs in only one of the blocks, so $uwy \neq a^p b^q a^p b^q$.

1.5 Closure properties