Lec 09. Context-free language and Parse trees

Eunjung Kim

Derivation

Consider a CFG $G = (V, \Sigma, R, S)$, $u, v, w \in (\Sigma \cup V)^*$ (a string of terminals and nonterminals) and a variable (nonterminal) $A \in V$.

Yield, derive, derivation

- We say that uAv yields uwv, written $uAv \Rightarrow_G uwv$, if G has the rule $A \to w$; put another way, uwv is obtained by substituting a variable in the string uAv by the body of a rule whose head is the said variable.
- We say that u derives v, written $u \Rightarrow_G^* v$. if $u = v$ or there is a sequence $u_1, \ldots, u_k$ for some $k \geq 1$ such that

  $$u \Rightarrow_G u_1 \Rightarrow_G \cdots \Rightarrow_G u_k \Rightarrow_G v$$

  and the sequence is called a derivation.

We omit the subscript G in $\to_G$ and $\Rightarrow_G$ if the CFG under consideration is clear in the context.

Derivation by example

We want to describe, as CFG,

- all arithmetic expressions with $+$ and $\times$
- over the variables of the form $(a \cup b)(a \cup b \cup 0 \cup 1)^*$.

Consider the following CFG $G_{ari} = (\{E, I\}, \{a, b, 0, 1, +, \times, (,)\}, R, E)$, where R consists of the following rules

1. $E \to I$
2. $E \to E + E$
3. $E \to E \times E$
4. $E \to (E)$
5. $I \to a$
6. $I \to b$
7. $I \to Ia$
8. $I \to Ib$
9. $I \to I0$
10. $I \to I1$

Context-free language

For a CFG $G = (V, \Sigma, R, S)$, the language of G, denoted by $L(G)$ is the set of all strings consisting of terminals (only) that have derivations from the start symbol, i.e.

$$L(G) = \{w \in \Sigma^* \mid S \Rightarrow_G^* w\}.$$

A language is said to be context-free if it is the language of a context-free grammar. A context-free languages is often abbreviated as CFL.

Some remarks on CFG

- In general, the rule of a grammar has the form $u \to v$ with both u and v are strings of terminals and nonterminals.
- A grammar is context-free if the head u is a nonterminal (variable) in all the rules; we do not need to consider the context.
- Different restrictions on the grammar define the hierarchy of formal languages.

Class	Languages	Automaton	Rules	Word Problem	Example
type-0	recursively enumerable	Turing machine	no restriction	undecidable	Post's corresp. problem
type-1	context sensitive	linear-bounded TM	$\alpha \to \gamma$ $\lvert\alpha\rvert \leq \lvert\gamma\rvert$	PSPACE-complete	$a^n b^n c^n$
type-2	context free	pushdown automaton	$A \to \gamma$	cubic	$a^n b^n$
type-3	regular	NFA / DFA	$A \to a$ or $A \to aB$	linear time	$a^* b^*$

Figure 1: Chomsky Hierarchy

Figure 1, Lecture note on 15-411: Compiler Design, CMU, 2023

Grammar and meaning of the language

Let us show that $L(G_{pal})$ is precisely the set of palindromes consisting of 0's and 1's.

- ($\rightarrow$) We want to show that if $S \Rightarrow^* w$, then w is a palindrome. Induction on the length of derivation.
  1. Base: if length at most 1, then $w = \epsilon, 0$ or 1, which is trivially a palindrome.
  2. Induction: if the derivation has length $n + 1$, then it is of the form

     $$S \Rightarrow 0S0 \Rightarrow^* 0x0 = w$$

     (or 0 replaced by 1) where $S \Rightarrow^* x$ is a derivation of length n. By I.H. x is a palindrome, and thus $0x0$ is.

Grammar and meaning of the language

Let us show that $L(G_{pal})$ is precisely the set of palindromes consisting of 0's and 1's.

- ($\rightarrow$) We want to show that if $S \Rightarrow^* w$, then w is a palindrome. Induction on the length of derivation.
  1. Base: if length at most 1, then $w = \epsilon, 0$ or 1, which is trivially a palindrome.
  2. Induction: if the derivation has length $n + 1$, then it is of the form

     $$S \Rightarrow 0S0 \Rightarrow^* 0x0 = w$$

     (or 0 replaced by 1) where $S \Rightarrow^* x$ is a derivation of length n. By I.H. x is a palindrome, and thus $0x0$ is.
- ($\leftarrow$) We want to show that if w is a palindrome, then $S \Rightarrow^* w$. Induction on $|w|$.
  1. Base: $|w| \leq 1$, then $S \Rightarrow w$ by a single application of one of the rules $S \rightarrow \epsilon|0|1$.
  2. Induction: if w is a palindrome of length ≥ 2, it is of the form $0x0$ or $1x1$ for some palindrome x. By I.H., $S \Rightarrow^* x$. Therefore,

     $$S \Rightarrow 0S0 \Rightarrow^* 0x0 = w$$

Grammar and derivation

Consider CFG G with the following rules.

- $S \to XSX \quad | \quad R$
- $R \to aTb \quad | \quad bTa$
- $T \to XTX \quad | \quad X \quad | \quad \epsilon$
- $X \to a \quad | \quad b$

1. Variables? Terminals? Start Variable?
2. $T \Rightarrow^* T$?
3. $T \Rightarrow^* XXX$?
4. $XXX \Rightarrow^* aba$?
5. $R \Rightarrow^* \epsilon$?
6. $S \Rightarrow^* abaababbaaba$?

Grammar and meaning of the language

Consider CFG G with the following rules. Describe the language $L(G)$ in plain English.

1. $S \rightarrow aSb \mid aA \mid bB$
2. $A \rightarrow aA \mid \epsilon$
3. $B \rightarrow bB \mid \epsilon$

Designing a context-free grammar

1. $L = \{0^n 1^n : n \geq 1\}$.
2. $L = \{0^n 1^n : n \geq 1\} \cup \{0^n 1^n : n \geq 1\}$.
3. $L = \{a^i b^j : i > j\}$.
4. $L = \{a^i b^j c^k : i \neq j \text{ or } j \neq k\}$.
5. $L = \{$the set of all well-formed parentheses$\}$.
6. $L = \{$all strings with the same number of 0's and 1's$\}$.

How to design CFG

The design of CFG requires some ingenuity. Some useful tips here.

- Many CFLs are the union of simpler CFLs.
- It is convenient to think of a variable as something that represents a set of strings; those which can be derived from that variable.
- A CFG for a regular language is easy to construct.
- Sometimes you use some nice combinatorial property of the language.

$L = \{0^n 1^n : n \geq 1\}.$

$$L = \{0^n 1^n : n \geq 1\} \cup \{1^n 0^n : n \geq 1\}.$$

$L = \{a^i b^j : i > j\}.$

$L = \{a^i b^j c^k : i \neq j \textbf{ OR } j \neq k\}.$

$L = \{$THE SET OF ALL WELL-FORMED PARENTHESES$\}$.

$L = \{$THE SAME $\#$ OF 0'S AND 1'S$\}$.

Leftmost/rightmost derivation

Definition

A derivation is a leftmost derivation if a production rule is applied to the leftmost variable in each step. A rightmost derivation is defined similarly.

Example: a leftmost derivation of the string "$a \times (a + b00)$" in the CFG G_{ari}

1. $E \to I \mid E + E \mid E \times E \mid (E)$
2. $I \to a \mid b \mid Ia \mid Ib \mid I0 \mid I1$

Graphic representation of the derivation

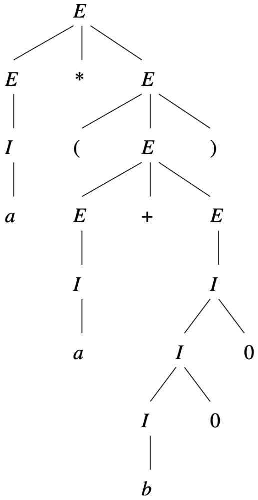

Figure 5.6, Hopcroft et. al. 2006

Parse tree

Definition

Let $G = (V, \Sigma, R, S)$ be a context-free grammar. A parse tree for the grammar G is a (rooted) tree satisfying the following.

1. Each internal node is labelled by a variable in V.
2. Each leaf is labelled by a member of $V \cup \Sigma \cup \{\epsilon\}$. If a leaf is labelled by ϵ, it is the only child of its parent.
3. If an internal node is labelled by A, and its children are labelled by

$$X_1, \dots, X_k$$

when read from the left to right, then there is a rule $A \to X_1 \cdots X_k$ in R.

Yield of a parse tree

Definition

Let $G = (V, \Sigma, R, S)$ be a context-free grammar. The yield of a parse tree is a string in $(V \cup \Sigma \cup \{\epsilon\})^*$ obtained by concatenating the labels on the leaves of the parse tree from left to right.

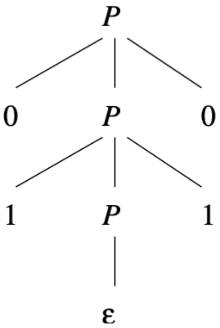

Figure 5.5, Hopcroft et. al. 2006

Equivalence of parse tree and derivation

Theorem

Let $G = (V, \Sigma, R, S)$ be a context-free grammar. The following are equivalent.

- $S \Rightarrow^* w$ *(i.e. $w \in L(G)$).*
- *There is a parse tree with root S and yield w.*
- $S \Rightarrow^*_{lm} w$.
- $S \Rightarrow^*_{rm} w$.