Parse Trees

CENG 280

Course outline

- Preliminaries: Alphabets and languages
- Regular languages
- Context-free languages
    - Context-free grammars
    - Parse trees
    - Push-down automaton
    - Push-down automaton - context-free languages
    - Languages that are and that are not context-free, Pumping lemma
- Turing-machines

Parse Trees

- Parse trees
- Derivations (leftmost-rightmost) and derivation similarity
- Ambiguity

Derivations

- Let $G = (V, \Sigma, R, S)$ be a context-free grammar.
- $L(G) = \{w \in \Sigma^\star \mid S \Rightarrow^\star w\}$

Derivations

- Let $G = (V, \Sigma, R, S)$ be a context-free grammar.
- $L(G) = \{w \in \Sigma^\star \mid S \Rightarrow^\star w\}$

Example

Consider $G = (V, \Sigma, R, S)$, $V = \{S, (,)\}$, $\Sigma = \{(,)\}$.
$R = \{S \to e, S \to SS, S \to (S)$. Write two derivations for ()().

Derivations

- Let $G = (V, \Sigma, R, S)$ be a context-free grammar.
- $L(G) = \{w \in \Sigma^\star \mid S \Rightarrow^\star w\}$

Example

Consider $G = (V, \Sigma, R, S)$, $V = \{S, (,)\}$, $\Sigma = \{(,)\}$.
$R = \{S \rightarrow e, S \rightarrow SS, S \rightarrow (S)$. Write two derivations for ()().

- A string $w \in L(G)$ of a context-free grammar can have different derivations.
- The derivations are considered the "same" when only the order of rule application changes, which is seen on a parse tree.

Derivations

- In a parse tree, each node is labelled with a symbol from V, and each leaf is labeled with a symbol from $\Sigma \cup \{e\}$
- By concatenating the labels of the leafs from left to right, we obtain the string generated by the derivation, which is called the yield of the parse tree.

Parse Trees

Definition

Given a context free grammar $G = (V, \Sigma, R, S)$, its parse trees are defined as follows.

1. $\circ\ a$ is a parse tree with yield "a" for each $a \in \Sigma$.
2. If $A \to e$ is a rule in R,
3. If $A \to A_1 \ldots A_n$ is a rule in R, and .....
4. Nothing else is a parse tree.

Example

Consider $G = (V, \Sigma, R, S)$, $V = \{S, (,)\}$, $\Sigma = \{(,)\}$.
$R = \{S \rightarrow e, S \rightarrow SS, S \rightarrow (S)$. Write two derivations for (())().

Parse trees represent equivalence classes of derivations suppressing the differences on the order of application rules.

Two derivations are said to be **similar** if one derivation can be obtained from another one by simply changing the order that the rules are applied. In other words, if one derivation can be obtained from another one via a series of *switchings*.

Parse Trees

- Two derivations are similar D, D', if one can be transformed into another by a series of switches in the order in which the rules are applied.
- The derivation similarity generates an equivalence relation.

Parse Trees

- Each equivalence class has a **leftmost derivation** : the left most non-terminal is replaced at each step of the derivation, i.e.

- A leftmost derivation of every parse tree exists and it is obtained as follows: Starting from the root of the tree, repeatedly replace the leftmost non-terminal in the derivation with the corresponding rule from the tree.

Parse Trees

- Similarly, each equivalence class (parse tree) has a rightmost derivation
- It is obtained by replacing the rightmost non-terminal (iteratively) in the derivation.

Parse Trees

Example 3.2.2 (continued): Parse trees capture exactly, via a natural isomorphism, the equivalence classes of the "similarity" equivalence relation between derivations of a string defined above. The equivalence class of the derivations of $(())()$ corresponding to the tree in Figure 3-4 contains the derivations D_1, D_2, D_3 shown above, and also these seven:

$$
\begin{aligned}
D_4 &= S \Rightarrow SS \Rightarrow (S)S \Rightarrow (S)(S) \Rightarrow ((S))(S) \Rightarrow (())(S) \Rightarrow (())() \\
D_5 &= S \Rightarrow SS \Rightarrow (S)S \Rightarrow (S)(S) \Rightarrow ((S))(S) \Rightarrow ((S))() \Rightarrow (())() \\
D_6 &= S \Rightarrow SS \Rightarrow (S)S \Rightarrow (S)(S) \Rightarrow (S)() \Rightarrow ((S))() \Rightarrow (())() \\
D_7 &= S \Rightarrow SS \Rightarrow S(S) \Rightarrow (S)(S) \Rightarrow ((S))(S) \Rightarrow (())(S) \Rightarrow (())() \\
D_8 &= S \Rightarrow SS \Rightarrow S(S) \Rightarrow (S)(S) \Rightarrow ((S))(S) \Rightarrow ((S))() \Rightarrow (())() \\
D_9 &= S \Rightarrow SS \Rightarrow S(S) \Rightarrow (S)(S) \Rightarrow (S)() \Rightarrow ((S))() \Rightarrow (())() \\
D_{10} &= S \Rightarrow SS \Rightarrow S(S) \Rightarrow S() \Rightarrow (S)() \Rightarrow ((S))() \Rightarrow (())()
\end{aligned}
$$

These ten derivations are related by $\prec$ as shown in Figure 3-5.

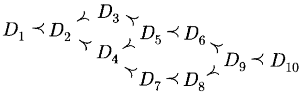

Figure 3-5

From "Elements of the theory of computation" by H.R. Lewis and C.H.

Theorem

Let $G = (V, \Sigma, R, S)$ be a context-free grammar, and let $A \in V \setminus \Sigma$ and $w \in \Sigma^\star$. Then the following statements are equivalent:

- $A \Rightarrow^\star w$
- *There is a parse tree with root A and yield w*
- *There is a leftmost derivation $A \overset{L}{\Rightarrow}^\star w$*
- *There is a rightmost derivation $A \overset{R}{\Rightarrow}^\star w$*

Ambiguity

Given a grammar $G = (V, \Sigma, R, S)$, if a string $w \in L(G)$ has two different parse trees (two different derivations that are not similar), then the grammar is called **ambiguous**.

Ambiguity

- Given a grammar $G = (V, \Sigma, R, S)$, if a string $w \in L(G)$ has two different parse trees (two different derivations that are not similar), then the grammar is called **ambiguous**.
- Assigning a parse tree to a string is called "parsing".
- It is an important concept since parsing allows us to understand why the string belongs to the grammar.
- In addition, it gives a "meaning" (or interpretation) to the string, which is particularly important for the programming languages.

Ambiguity

Example

$G = (V, \Sigma, R, E)$
$V = \{+, x, (,), id, E\}, \Sigma = \{+, x, (,), id, \}$,
$R = \{E \rightarrow E + E, E \rightarrow ExE, E \rightarrow (E), E \rightarrow id\}$
Write two different leftmost derivations for “id+id x id”.

Ambiguity

Example

Is $G = (V, \Sigma, R, S)$ ambiguous, where $V = \{a, S\}$, $R : S \rightarrow SS, S \rightarrow a$?
If yes, disambiguate the grammar.

Ambiguity

Example

Write a grammar for $L = \{a^n b^m c^k \mid n = m \text{ or } m = k\}$. Is the grammar ambiguous?

Ambiguity

A context-free language L is called **inherently ambiguous** if every grammar G that generate L, i.e. $L = L(G)$, is ambiguous.