Lec 08. Parse trees and ambiguity

Eunjung Kim

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

Graphic representation of the derivation

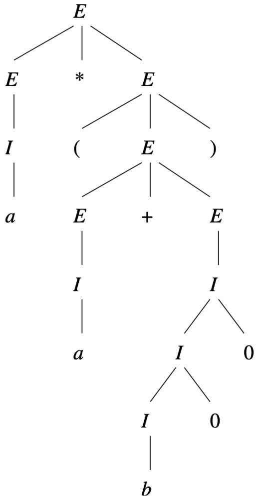

Figure 5.6, Hopcroft et. al. 2006

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

$$X_1, \ldots, X_k$$

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 obtained by concatenating the labels on the leaves of the parse tree from left to right.

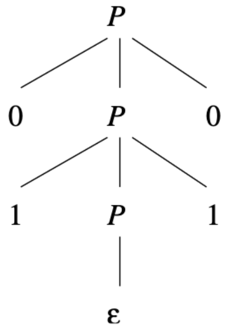

Figure 5.5, Hopcroft et. al. 2006

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.*
- $S \Rightarrow^*_{lm} w$.
- $S \Rightarrow^*_{rm} w$.

Ambiguity in grammars and languages

In the grammar G_{ari}

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

How many parse trees that yield the string "$E + E \times E$"?

How many parse trees that yield the string "$a + b$"?

Ambiguity in grammars and languages

Ambiguous grammar

- A grammar is ambiguous if there is a string $w \in \Sigma^*$ such that there are (at least two) parse trees, in each of which the root is labelled by the start variable S and w is the yield.
- Equivalently, a grammar is ambiguous if a string has two leftmost derivations.
- A grammar is unambiguous if every string has at most one parse tree in the grammar.

Inherent ambiguity

Inherently Ambiguous CFL

- A context-free language L is inherently ambiguous if any grammar G which generates L (i.e. $L(G) = L$) is ambiguous.
- A CFL L is unambiguous if there is an unambiguous grammar G which generates L.

Eliminating ambiguity

- There is no algorithm which decides whether a given CFG is ambiguous or not ("undecidable problem").
- There are inherently ambiguous languages: e.g. $\{a^n b^n c^m d^m : n, m \geq 1\} \cup \{a^n b^m c^m d^n : n, m \geq 1\}$
- Showing if a language is inherently ambiguous or unambiguous is not easy (in terms of proof...)
- BUT, many CFL's we care are unambiguous, and there are techniques to modify the grammar to eliminate ambiguity.

Eliminating ambiguity: example 1

The grammar G_{ari} is ambiguous: e.g. $a + a \times a$ and $a + a + a$

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

Goal: we want the grammar to

- respect the priority of operators, and
- generate a sequence of identical operations in a unique way, e.g. grouped from left to right.

Eliminating ambiguity: example 1

The grammar G_{ari} is ambiguous: e.g. $a + a \times a$ and $a + a + a$

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

Arithmetic expression with $+$, $\times$ are written like

$$(a0 + b1) \times a0 + b0 \times b \times b_0 + b11 \times (a1 + b0) = term + term + term$$

where each term is <u>factored</u> into

$$(a0 + b1) - a0,\ b_0\text{-}b\text{-}b_0 \text{ and } b11 - (a11 + b0).$$

Eliminating ambiguity: example 1

We introduce intermediary variables which represent the following.

- "Identifier": the existing variable I already represents them.
- "Factor": the operands of $\times$. Identifiers and an expression surrounded by () in the expressions of G_{ari}.
- "Term": those separated by $+$ in an expression.
- "Expression": any expression generated by G_{ari}.

Eliminating ambiguity: example 1

We introduce intermediary variables which represent the following.

- "Identifier": the existing variable I already represents them.
- "Factor": the operands of $\times$. Identifiers and an expression surrounded by () in the expressions of G_{ari}.
- "Term": those separated by $+$ in an expression.
- "Expression": any expression generated by G_{ari}.

Let us construct a CFG with the additional variables as above (Start E).

- $I \to a \quad | \quad b \quad | \quad Ia \quad | \quad Ib \quad | \quad I0 \quad | \quad I1$
- $F \to I \quad | \quad (E)$
- $T \to F \quad | \quad T \times F$
- $E \to E \quad | \quad E + T$

Eliminating ambiguity: example 1

New CFG generating $L(G_{ari})$ (Start E).

- $I \to a \quad | \quad b \quad | \quad Ia \quad | \quad Ib \quad | \quad I0 \quad | \quad I1$
- $F \to I \quad | \quad (E)$
- $T \to F \quad | \quad T \times F$
- $E \to E \quad | \quad E + T$

Parse tree for $a + a \times a$.

Eliminating ambiguity: example 1

New CFG generating $L(G_{ari})$ (Start E).

- $I \to a \quad | \quad b \quad | \quad Ia \quad | \quad Ib \quad | \quad I0 \quad | \quad I1$
- $F \to I \quad | \quad (E)$
- $T \to F \quad | \quad T \times F$
- $E \to E \quad | \quad E + T$

Parse tree for $a + a \times a$.

The new CFG is an unambiguous grammar generating the same language.

Eliminating ambiguity: example 2

CFG generating a well-formed parenthesis.

$$E \rightarrow EE \quad | \quad (E) \quad | \quad \epsilon$$

How many parse trees for ()()()?

Eliminating ambiguity: example 2

CFG generating a well-formed parenthesis.

$$E \to EE \quad | \quad (E) \quad | \quad \epsilon$$

How many parse trees for ()()()?

Ambiguity of the grammar arises from that a concatenation of well-formed parenthesis can be expressed by multiple parse trees.

We use the same principle for eliminating ambiguity as for the arithmetic expression.