CONTEXT FREE GRAMMARS

Submitted by: shabnam
Computer science department

CONTEXT-FREE GRAMMAR

Definition. A context-free grammar is a 4-tuple (Σ, NT, R, S), where:

- Σ is an alphabet (each character in Σ is called **terminal**)
- NT is a set (each element in NT is called **nonterminal**)
- R, the set of rules, is a subset of $NT \times (\Sigma \cup NT)^*$

If $(\alpha,\beta) \in R$, we write production $\alpha \rightarrow \beta$

β is called a **sentential form**

- S, the **start symbol**, is one of the symbols in NT

CFGs: Alternate Definition

many textbooks use different symbols and terms to describe CFG's

$$G = (V, \Sigma, P, S)$$

V = variables a finite set

Σ = alphabet or terminals a finite set

P = productions a finite set

S = start variable $S \in V$

Productions' form, where $A \in V$, $\alpha \in (V \cup \Sigma)^*$:

- $A \rightarrow \alpha$

DERIVATIONS

Definition. v is **one-step derivable** from u, written $u \Rightarrow v$, if:

- $u = x\alpha z$
- $v = x\beta z$
- $\alpha \rightarrow \beta$ in R

Definition. v is **derivable** from u, written $u \Rightarrow^* v$, if:
There is a chain of one-derivations of the form:

$$u \Rightarrow u_1 \Rightarrow u_2 \Rightarrow \ldots \Rightarrow v$$

CONTEXT-FREE LANGUAGES

Definition. Given a context-free grammar $G = (\Sigma, NT, R, S)$, the **language generated** or derived from G is the set:

$$L(G) = \{w : S \Rightarrow^* w \}$$

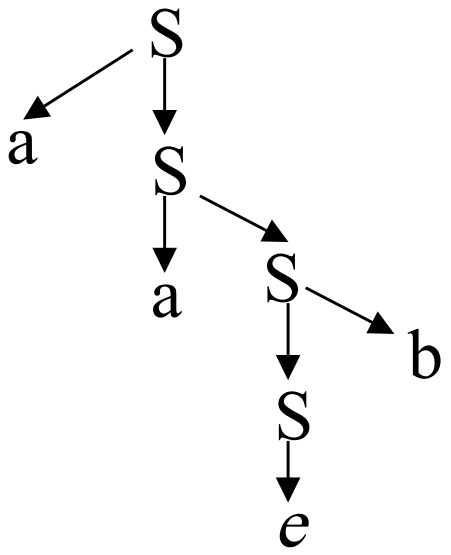
Definition. A language L is context-free if there is a context-free grammar $G = (\Sigma, NT, R, S)$, such that L is generated from G

PARSE TREE

A parse tree of a derivation is a tree in which:

- Each internal node is labeled with a nonterminal

- If a rule $A \rightarrow A_1A_2\ldots A_n$ occurs in the derivation then A is a parent node of nodes labeled $A_1, A_2, \ldots, A_n$



PARSE TREES

$S \rightarrow A \mid A\ B$

$A \rightarrow \varepsilon \mid \mathbf{a} \mid A\ \mathbf{b} \mid A\ A$

$B \rightarrow \mathbf{b} \mid \mathbf{b}\ \mathbf{c} \mid B\ \mathbf{c} \mid \mathbf{b}\ B$

Sample derivations:

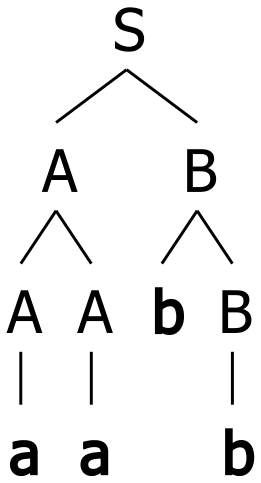
$S \Rightarrow AB \Rightarrow AAB \Rightarrow \mathbf{a}AB \Rightarrow \mathbf{aa}B \Rightarrow \mathbf{aab}B \Rightarrow \mathbf{aabb}$

$S \Rightarrow AB \Rightarrow A\mathbf{b}B \Rightarrow A\mathbf{bb} \Rightarrow AA\mathbf{bb} \Rightarrow A\mathbf{abb} \Rightarrow \mathbf{aabb}$

These two derivations use same productions, but in different orders.

This ordering difference is often uninteresting.

Derivation trees give way to abstract away ordering differences.



Root label = start node.

Each interior label = variable.

Each parent/child relation = derivation step.

Each leaf label = terminal or ε.

All leaf labels together = derived string = *yield*.

LEFTMOST, RIGHTMOST DERIVATIONS

Definition. A **left-most derivation** of a sentential form is one in which rules transforming the left-most nonterminal are always applied

Definition. A **right-most derivation** of a sentential form is one in which rules transforming the right-most nonterminal are always applied

LEFTMOST & RIGHTMOST DERIVATIONS

$S \rightarrow A \mid A\ B$

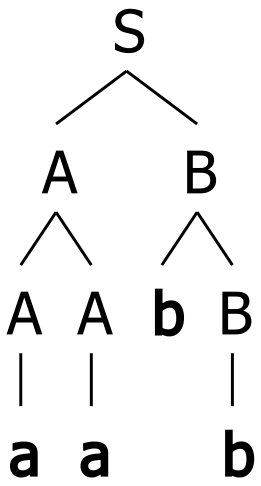
$A \rightarrow \varepsilon \mid \mathbf{a} \mid A\ \mathbf{b} \mid A\ A$

$B \rightarrow \mathbf{b} \mid \mathbf{b}\ \mathbf{c} \mid B\ \mathbf{c} \mid \mathbf{b}\ B$

Sample derivations:

$S \Rightarrow AB \Rightarrow AAB \Rightarrow \mathbf{a}AB \Rightarrow \mathbf{aa}B \Rightarrow \mathbf{aab}B \Rightarrow \mathbf{aabb}$

$S \Rightarrow AB \Rightarrow A\mathbf{b}B \Rightarrow A\mathbf{bb} \Rightarrow AA\mathbf{bb} \Rightarrow A\mathbf{abb} \Rightarrow \mathbf{aabb}$



These two derivations are special.

1st derivation is *leftmost*.
Always picks leftmost variable.

2nd derivation is *rightmost*.
Always picks rightmost variable.

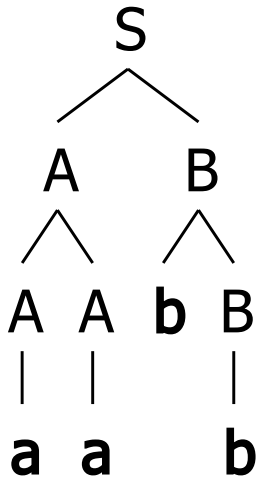
LEFT / RIGHTMOST DERIVATIONS

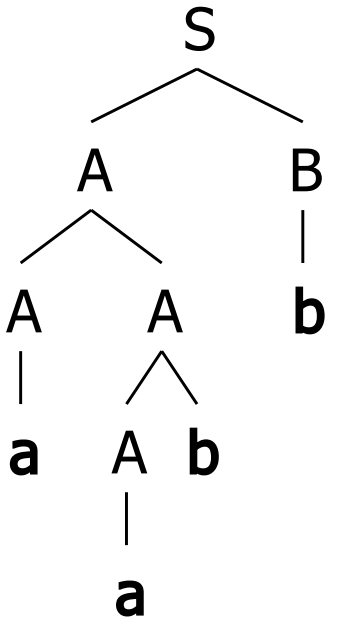
- In proofs…
  - Restrict attention to left- or rightmost derivations.
- In parsing algorithms…
  - Restrict attention to left- or rightmost derivations.
  - E.g., recursive descent uses leftmost; `yacc` uses rightmost.

DERIVATION TREES

S → A | A B
A → ε | **a** | A **b** | A A
B → **b** | **b c** | B **c** | **b** B

w = **aabb**

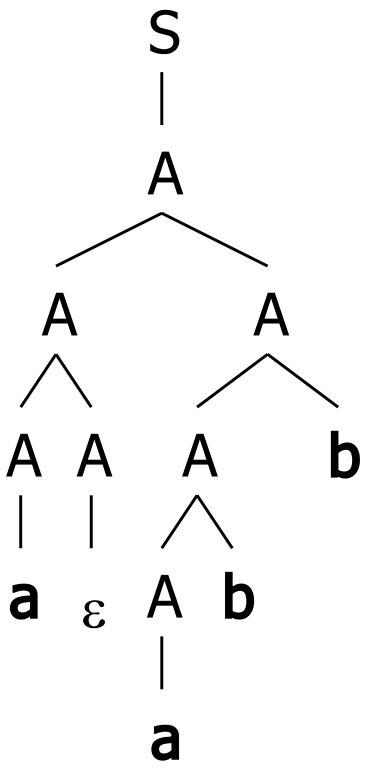




Other derivation trees for this string?

?

?



Infinitely many others possible.

AMBIGUOUS GRAMMAR

Definition. A grammar G is ambiguous if there is a word w $\in$ L(G) having are least two different parse trees

$S \rightarrow A$
$S \rightarrow B$
$S \rightarrow AB$
$A \rightarrow aA$
$B \rightarrow bB$
$A \rightarrow e$
$B \rightarrow e$

Notice that a has at least two left-most derivations

AMBIGUITY

CFG *ambiguous* $\Leftrightarrow$ any of following equivalent statements:

- $\exists$ string w with multiple derivation trees.
- $\exists$ string w with multiple leftmost derivations.
- $\exists$ string w with multiple rightmost derivations.

Defining ambiguity of grammar, not language.

AMBIGUITY & DISAMBIGUATION

Given an ambiguous grammar, would like an equivalent unambiguous grammar.

- Allows you to know more about structure of a given derivation.
- Simplifies inductive proofs on derivations.
- Can lead to more efficient parsing algorithms.
- In programming languages, want to impose a canonical structure on derivations. E.g., for **1+2×3**.

Strategy: Force an ordering on all derivations.

Thank you