Context free Grammar

- Context-free grammar is a 4-tuple $G = (N, T, P, S)$ where
  - T is a finite set of tokens (*terminal* symbols)
  - N is a finite set of *nonterminals*
  - P is a finite set of *productions* of the form

    $\alpha \to \beta$

    where $\alpha \in N$ and $\beta \in (N \cup T)^*$
  - $S \in N$ is a designated *start symbol*

Example Grammar

Context-free grammar for simple expressions:

G = <{*expr*,op,digit}, {**+**,**-**,*****,**/**,**0**,**1**,**2**,**3**,**4**,**5**,**6**,**7**,**8**,**9**,**)**,**(**},
P,expr>

with productions *P* =

expr→ *expr* op expr

expr →(expr)

expr → *digit*

digit → **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9**

op → **+** | **-** | ***** | **/**

Notational Conventions Used

- Terminals: Lower case letters, operator symbols, punctuation symbols, digits, bolface strings are all terminals

- Non Terminals: Upper case letters, lower case italic names are usually non terminals

- Greek letters such as α,β,γ represent strings of grammars symbols. Thus a generic production can be written as $A \rightarrow \alpha$

Example

- Design a CFG for the language

$$L(G)= \{0^n 1^m \mid n <> m\}$$

There are two cases:

- For n>m
- For n<m
- Write two separate set of rules and combine them

Example

- For n>m

$S1 \rightarrow AB$

$B \rightarrow 0B1 \mid \epsilon$

$A \rightarrow 0A \mid 0$

For n<m

$S2 \rightarrow XY$

$X \rightarrow 0X1 \mid \epsilon$

$Y \rightarrow 1Y \mid 1$

Combining both:

$S \rightarrow S1 \mid S2$

Examples

- Write a CFG that generates Equal number of a's, b's and c's?

Derivations

- The *one-step derivation* is defined by
  $\alpha\ A\ \beta \Rightarrow \alpha\ \gamma\ \beta$, where $A \rightarrow \gamma$ is a production in the grammar

- In addition, we define
  - $\Rightarrow$ is *leftmost* $\Rightarrow_{lm}$ if α does not contain a nonterminal
  - $\Rightarrow$ is *rightmost* $\Rightarrow_{rm}$ if β does not contain a nonterminal
  - Transitive closure $\Rightarrow^{*}$ (zero or more steps)
  - Positive closure $\Rightarrow^{+}$ (one or more steps)

- The *language generated by G* is defined by
  $L(G) = \{w \in T^{*} \mid S \Rightarrow^{+} w\}$

Derivation (Example)

Grammar $G = (\{E\}, \{+, *, (,), -, \mathbf{id}\}, P, E)$ with productions $P =$

$$
\begin{aligned}
E &\rightarrow E + E \\
E &\rightarrow E * E \\
E &\rightarrow (E) \\
E &\rightarrow - E \\
E &\rightarrow \mathbf{id}
\end{aligned}
$$

Example derivations: $E \Rightarrow - E \Rightarrow - \mathbf{id}$

$$E \Rightarrow_{rm} E + E \Rightarrow_{rm} E + \mathbf{id} \Rightarrow_{rm} \mathbf{id} + \mathbf{id}$$

$$E \Rightarrow^{*} E$$

$$E \Rightarrow^{*} \mathbf{id} + \mathbf{id}$$

$$E \Rightarrow^{+} \mathbf{id} * \mathbf{id} + \mathbf{id}$$

Derivation for the Example Grammar

$$
\begin{aligned}
&\underline{\mathit{list}} \\
&\Rightarrow \underline{\mathit{list}} + \mathit{digit} \\
&\Rightarrow \underline{\mathit{list}} - \mathit{digit} + \mathit{digit} \\
&\Rightarrow \underline{\mathit{digit}} - \mathit{digit} + \mathit{digit} \\
&\Rightarrow \mathbf{9} - \underline{\mathit{digit}} + \mathit{digit} \\
&\Rightarrow \mathbf{9} - \mathbf{5} + \underline{\mathit{digit}} \\
&\Rightarrow \mathbf{9} - \mathbf{5} + \mathbf{2}
\end{aligned}
$$

This is an example *leftmost derivation*, because we replaced the leftmost nonterminal (underlined) in each step. Likewise, a *rightmost derivation* replaces the rightmost nonterminal in each step

Chomsky Hierarchy: Language Classification

- A grammar *G* is said to be
  - *Regular* if it is *right linear* where each production is of the form

    $A \rightarrow w\,B$ or $A \rightarrow w$

    or *left linear* where each production is of the form

    $A \rightarrow B\,w$ or $A \rightarrow w$
  - *Context free* if each production is of the form

    $A \rightarrow \alpha$

    where $A \in N$ and $\alpha \in (N \cup T)^*$
  - *Context sensitive* if each production is of the form

    $\alpha\, A\, \beta \rightarrow \alpha\, \gamma\, \beta$

    where $A \in N$, $\alpha, \gamma, \beta \in (N \cup T)^*$, $|\gamma| > 0$
  - *Unrestricted*

Chomsky Hierarchy

L(*regular*) $\subset$ L(*context free*)
$\subset$ L(*context sensitive*) $\subset$ L(*unrestricted*)

Examples:

Every *finite language* is regular!
(construct a FSA for strings in $L(G)$)

$L_1 = \{ \mathbf{a}^n\mathbf{b}^n \mid n \geq 1 \}$ is context free

$L_2 = \{ \mathbf{a}^n\mathbf{b}^n\mathbf{c}^n \mid n \geq 1 \}$ is context sensitive

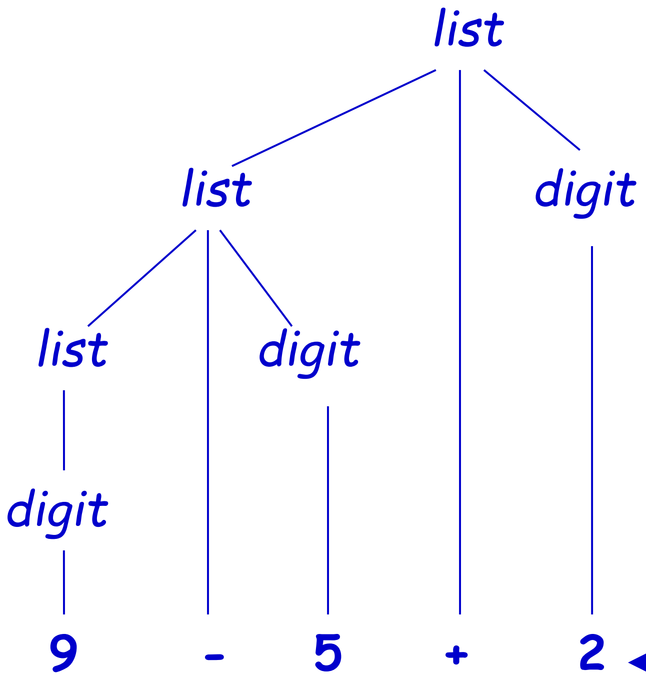
Parse Trees

- The *root* of the tree is labeled by the start symbol
- Each *leaf* of the tree is labeled by a terminal (=token) or ε
- Each *interior node* is labeled by a nonterminal

If $A \rightarrow X_1\ X_2\ \ldots\ X_n$ is a production, then node A has immediate *children* $X_1, X_2, \ldots, X_n$ where X_i is a (non)terminal or ε (ε denotes the *empty string*)

Parse Tree for the Example Grammar

Parse tree of the string **9-5+2** using grammar *G*



The sequence of leafs is called the *yield* of the parse tree

Example of Parse Tree

- Suppose we have the following grammar

  E → E + E
  E → E * E
  E → (E)
  E → - E
  E → **id**

Perform Left most derivation, right most derivation and construct a parse tree for the string

id+id*id

Two possible Parse Trees using Leftmost derivation

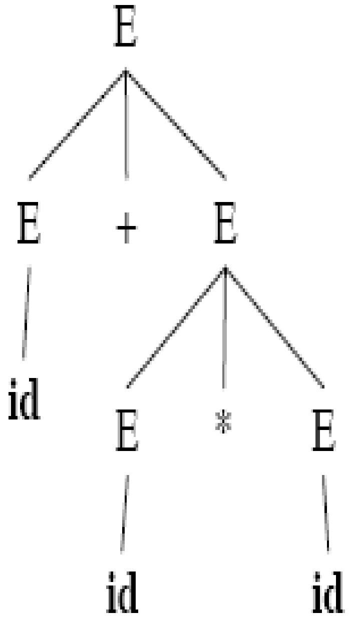
- $E \Rightarrow E + E$

$\Rightarrow id + E$

$\Rightarrow id + E * E$

$\Rightarrow id + id * E$

$\Rightarrow id + id * id$



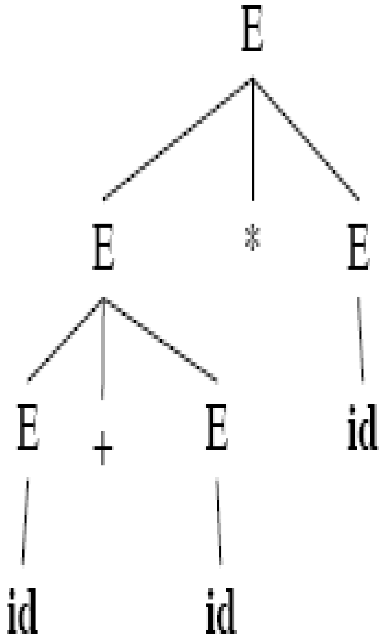
- $E \Rightarrow E * E$

$\Rightarrow E + E * E$

$\Rightarrow id + E * E$

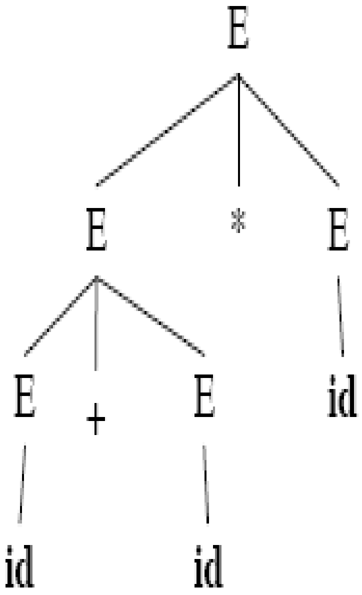
$\Rightarrow id + id * E$

$\Rightarrow id + id * id$



Parse Tree via Right most derivation

$$
\begin{aligned}
E &\Rightarrow E * E \\
&\Rightarrow E * id \\
&\Rightarrow E + E * id \\
&\Rightarrow E + id * id \\
&\Rightarrow id + id * id
\end{aligned}
$$



Ambiguity

- Grammar is ambiguous if more than one parse tree is possible for some string as shown in the previous example. If there are more than one left most derivations or more than one right most derivations.

- Ambiguity is not acceptable
  - Unfortunately, it's undecidable to check whether a given CFG is ambiguous
  - Some CFLs are inherently ambiguous (do not have an unambiguous CFG)

Ambiguity (cont'd)

Consider the following context-free grammar:

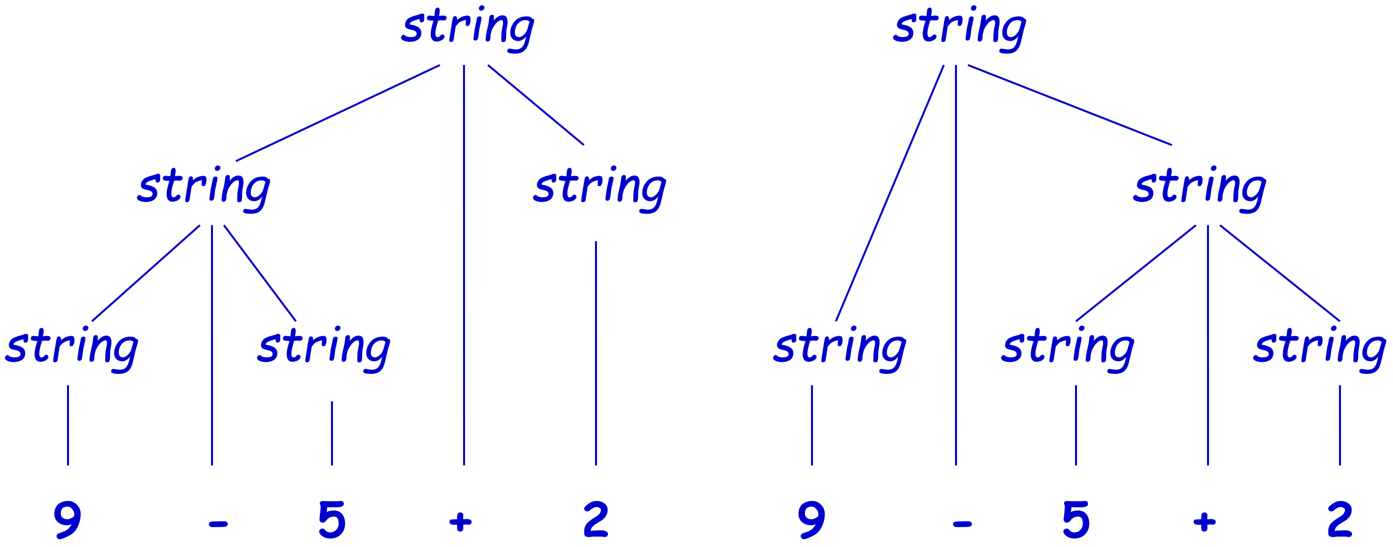
$G = \langle \{\mathit{string}\}, \{\mathbf{+}, \mathbf{-}, \mathbf{0}, \mathbf{1}, \mathbf{2}, \mathbf{3}, \mathbf{4}, \mathbf{5}, \mathbf{6}, \mathbf{7}, \mathbf{8}, \mathbf{9}\}, P, \mathit{string} \rangle$

with production P =

$\mathit{string} \to \mathit{string}\ \mathbf{+}\ \mathit{string} \mid \mathit{string}\ \mathbf{-}\ \mathit{string} \mid \mathbf{0} \mid \mathbf{1} \mid \ldots \mid \mathbf{9}$

This grammar is *ambiguous*, because more than one parse tree represents the string **9-5+2**

Two Parse Trees for the same string



Practice

- Show that the following grammar is ambiguous: (Find out strings and two parse trees)

1) $S \rightarrow AB \mid aaB$
   $A \rightarrow a \mid Aa$
   $B \rightarrow b$

2) $S \rightarrow a \mid abSb \mid aAb$
   $A \rightarrow bS \mid aAAb$

3) $S \rightarrow aSb \mid SS \mid \varepsilon$

Simplifications
of
Context-Free Grammars

A Substitution Rule

$$S \to aB$$
$$A \to aaA$$
$$A \to abBc$$
$$B \to aA$$
$$B \to b$$

Substitute

$$B \to b$$

Equivalent grammar

$$S \to aB \mid ab$$
$$A \to aaA$$
$$A \to abBc \mid abbc$$
$$B \to aA$$

A Substitution Rule

$$S \to aB \mid ab$$

$$A \to aaA$$

$$A \to abBc \mid abbc$$

$$B \to aA$$

Substitute

$$B \to aA$$

$$S \to \xcancel{aB} \mid ab \mid aaA$$

$$A \to aaA$$

$$A \to \xcancel{abBc} \mid abbc \mid abaAc$$

Equivalent grammar

In general:

$$A \to xBz$$

$$B \to y_1$$

Substitute

$$B \to y_1$$

$$A \to xBz \mid xy_1z$$

equivalent grammar

Nullable Variables

$\lambda-\text{production}:$ $A \to \lambda$

Nullable Variable: $A \Rightarrow \ldots \Rightarrow \lambda$

Removing Nullable Variables

Example Grammar:

$$S \to aMb$$

$$M \to aMb$$

$$M \to \lambda$$

Nullable variable

$S \to aMb$

$M \to aMb$

$\cancel{M \to \lambda}$

Substitute
$M \to \lambda$

Final Grammar

$S \to aMb$

$S \to ab$

$M \to aMb$

$M \to ab$

Unit-Productions

Unit Production: $A \rightarrow B$

(a single variable in both sides)

Removing Unit Productions

Observation:

$$A \to A$$

Is removed immediately

Example Grammar:

$$S \rightarrow aA$$

$$A \rightarrow a$$

$$A \rightarrow B$$

$$B \rightarrow A$$

$$B \rightarrow bb$$

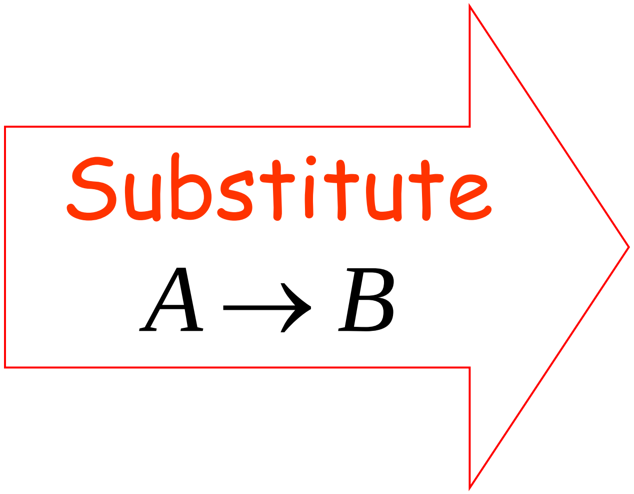
$$S \to aA$$

$$A \to a$$

$$\cancel{A \to B}$$

$$B \to A$$

$$B \to bb$$



Substitute

$$A \to B$$

$$S \to aA \mid aB$$

$$A \to a$$

$$B \to A \mid B$$

$$B \to bb$$

$$S \to aA \mid aB$$

$$A \to a$$

$$B \to A \mid \cancel{B}$$

$$B \to bb$$

Remove

$$B \to B$$

$$S \to aA \mid aB$$

$$A \to a$$

$$B \to A$$

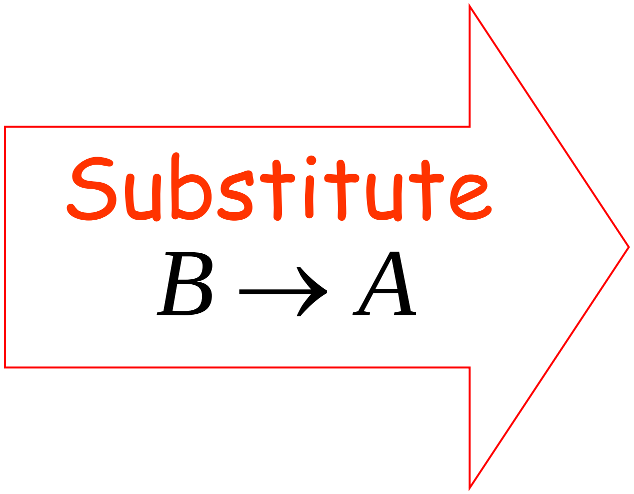
$$B \to bb$$

$$S \to aA \mid aB$$

$$A \to a$$

$$\xcancel{B \to A}$$

$$B \to bb$$



Substitute

$$B \to A$$

$$S \to aA \mid aB \mid aA$$

$$A \to a$$

$$B \to bb$$

Remove repeated productions

$$S \to aA \mid aB \mid \cancel{aA}$$
$$A \to a$$
$$B \to bb$$

⇒

Final grammar

$$S \to aA \mid aB$$
$$A \to a$$
$$B \to bb$$

Useless Productions

$$S \to aSb$$

$$S \to \lambda$$

$$S \to A$$

$$A \to aA$$ Useless Production

Some derivations never terminate...

$$S \Rightarrow A \Rightarrow aA \Rightarrow aaA \Rightarrow \ldots \Rightarrow aa\ldots aA \Rightarrow \ldots$$

Another grammar:

$$S \rightarrow A$$

$$A \rightarrow aA$$

$$A \rightarrow \lambda$$

$$B \rightarrow bA$$

Useless Production

Not reachable from S

In general:

contains only terminals

if $S \Rightarrow \ldots \Rightarrow xAy \Rightarrow \ldots \Rightarrow w$

$w \in L(G)$

then variable A is useful

otherwise, variable A is useless

A production $A \to x$ is useless
if any of its variables is useless

$S \to aSb$

$S \to \lambda$ Productions

Variables $S \to A$ useless

useless $A \to aA$ useless

useless $B \to C$ useless

useless $C \to D$ useless

Removing Useless Productions

Example Grammar:

$$S \rightarrow aS \mid A \mid C$$

$$A \rightarrow a$$

$$B \rightarrow aa$$

$$C \rightarrow aCb$$

First: find all variables that can produce strings with only terminals

$$S \to aS \mid A \mid C$$

$$A \to a$$

$$B \to aa$$

$$C \to aCb$$

Round 1: $\{A, B\}$

$$S \to A$$

Round 2: $\{A, B, S\}$

Keep only the variables that produce terminal symbols: $\{A, B, S\}$

(the rest variables are useless)

$$S \to aS \mid A \mid \cancel{C}$$

$$A \to a$$

$$B \to aa$$

$$\cancel{C \to aCb}$$

⇨

$$S \to aS \mid A$$

$$A \to a$$

$$B \to aa$$

Remove useless productions

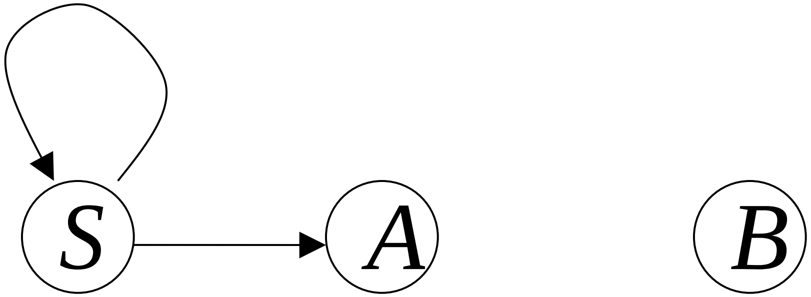
Second: Find all variables reachable from S

Use a Dependency Graph

$$S \to aS \mid A$$

$$A \to a$$

$$B \to aa$$



not reachable

Keep only the variables reachable from S

(the rest variables are useless)

$$S \to aS \mid A$$

$$A \to a$$

$$\xcancel{B \to aa}$$

$$\Longrightarrow$$

Final Grammar

$$S \to aS \mid A$$

$$A \to a$$

Remove useless productions

Removing All

Step 1: Remove Nullable Variables

Step 2: Remove Unit-Productions

Step 3: Remove Useless Variables

Do it yourself: Why in this order only??