## Context-Free Languages

Regular languages:

- keywords in a programming language
- names of identifiers
- integers
- all misc symbols: $=$;

Not Regular languages:

- $\left\{a^{n} c b^{n} \mid n>0\right\}$
- expressions - $((a+b)-c)$
- block structures (\{\} in Java/C++ and begin ... end in pascal)

Definition: A grammar $G=(V, T, S, P)$ is context-free if all productions are of the form

$$
\mathrm{A} \rightarrow \mathrm{x}
$$

Where $A \in V$ and $x \in(V \cup T)^{*}$.

Definition: $L$ is a context-free language (CFL) iff $\exists$ context-free grammar (CFG) G s.t. $L=L(G)$.

Example: G=(\{S\},\{a,b\},S,P)

$$
\mathrm{S} \rightarrow \mathrm{aSb} \mid \mathrm{ab}
$$

Derivation of aaabbb:

$$
\mathrm{S} \Rightarrow \mathrm{aSb} \Rightarrow \mathrm{aaSbb} \Rightarrow \mathrm{aaabbb}
$$

$\mathrm{L}(\mathrm{G})=$

Example: $\mathbf{G}=(\{\mathbf{S}\},\{\mathbf{a}, \mathbf{b}\}, \mathbf{S}, \mathbf{P})$

$$
\mathbf{S} \rightarrow \mathbf{a S a}|\mathbf{b S b}| \mathbf{a}|\mathbf{b}| \lambda
$$

Derivation of ababa:

$$
\mathbf{S} \Rightarrow \mathrm{aSa} \Rightarrow \mathrm{abSba} \Rightarrow \mathrm{ababa}
$$

$$
\Sigma=\{a, b\}, \mathbf{L}(\mathbf{G})=
$$

Example: $\mathbf{G}=(\{\mathbf{S}, \mathbf{A}, \mathbf{B}\},\{\mathbf{a}, \mathbf{b}, \mathbf{c}\}, \mathbf{S}, \mathbf{P})$

$$
\begin{aligned}
& \mathbf{S} \rightarrow \mathbf{A c B} \\
& \mathbf{A} \rightarrow \mathbf{a A a} \mid \lambda \\
& \mathbf{B} \rightarrow \mathbf{B b b} \mid \lambda
\end{aligned}
$$

$\mathrm{L}(\mathrm{G})=$

Derivations of aacbb:

1. $\mathrm{S} \Rightarrow \mathrm{AcB} \Rightarrow \mathrm{aAacB} \Rightarrow \mathrm{aacB} \Rightarrow$ aacBbb $\Rightarrow$ aacbb
2. $\mathrm{S} \Rightarrow \mathrm{AcB} \Rightarrow \mathrm{AcBbb} \Rightarrow \mathrm{Acbb} \Rightarrow$
aAacbb $\Rightarrow$ aacbb
Note: Next variable to be replaced is underlined.

Definition: Leftmost derivation - in each step of a derivation, replace the leftmost variable. (see derivation 1 above).

Definition: Rightmost derivation - in each step of a derivation, replace the rightmost variable. (see derivation 2 above).

Derivation Trees (also known as "parse trees")

A derivation tree represents a derivation but does not show the order productions were applied.

A derivation tree for $\mathrm{G}=(\mathrm{V}, \mathrm{T}, \mathrm{S}, \mathrm{P})$ :

- root is labeled S
- leaves labeled $\mathbf{x}$, where $\mathbf{x} \in \mathbf{T} \cup\{\lambda\}$
- nonleaf vertices labeled $\mathbf{A}, \mathrm{A} \in \mathrm{V}$
- For rule $\mathbf{A} \rightarrow a_{1} a_{2} a_{3} \ldots a_{n}$, where $\mathbf{A} \in \mathbf{V}, a_{i} \in(\mathbf{T} \cup \mathbf{V} \cup\{\lambda\})$,


Example: G=(\{S,A,B\},\{a,b,c\},S,P)

$$
\begin{aligned}
& \mathrm{S} \rightarrow \mathbf{A c B} \\
& \mathbf{A} \rightarrow \mathrm{aAa} \mid \lambda \\
& \mathrm{B} \rightarrow \mathbf{B b b} \mid \lambda
\end{aligned}
$$

Definitions Partial derivation tree subtree of derivation tree.

If partial derivation tree has root $S$ then it represents a sentential form.

Leaves from left to right in a derivation tree form the yield of the tree.

Yield (w) of derivation tree is such that $\mathbf{w} \in \mathrm{L}(\mathrm{G})$.
The yield for the example above is

## Example of partial derivation tree that has root S :

The yield of this example is which is a sentential form.

## Example of partial derivation tree that does not have root $S$ :

Membership Given CFG G and string $\mathbf{w} \in \Sigma^{*}$, is $\mathbf{w} \in \mathbf{L}(\mathbf{G}) ?$

If we can find a derivation of $w$, then we would know that $w$ is in $L(G)$.

Motivation
$G$ is grammar for Java w is Java program. Is w syntactically correct?

## Example

$$
\begin{aligned}
& \mathbf{G}=(\{\mathbf{S}\},\{\mathbf{a}, \mathbf{b}\}, \mathbf{S}, \mathbf{P}), \mathbf{P}= \\
& \quad \mathbf{S} \rightarrow \mathbf{S S}|\mathbf{a S a}| \mathbf{b} \mid \lambda \\
& L_{1}=\mathbf{L}(\mathbf{G})=
\end{aligned}
$$

## Is abbab $\in \mathbf{L}(\mathbf{G})$ ?

## Exhaustive Search Algorithm

For all $i=1,2,3, \ldots$
Examine all sentential forms yielded by i substitutions

Example: Is abbab $\in \mathbf{L}(\mathbf{G})$ ?

Theorem If CFG G does not contain rules of the form

$$
\begin{aligned}
& \mathbf{A} \rightarrow \lambda \\
& \mathbf{A} \rightarrow \mathbf{B}
\end{aligned}
$$

where $A, B \in V$, then we can determine if $w \in L(G)$ or if $w \notin L(G)$.

- Proof: Consider

1. length of sentential forms
2. number of terminal symbols in a sentential form

Example: Let $L_{2}=L_{1}-\{\lambda\} . L_{2}=\mathbf{L}(\mathbf{G})$ where G is:

$$
\mathrm{S} \rightarrow \mathrm{SS}|\mathrm{aa}| \mathrm{aSa} \mid \mathrm{b}
$$

Show baaba $\notin \mathrm{L}(\mathrm{G})$.

$$
\begin{array}{ll}
\mathrm{i}=1 & \text { 1. } \mathrm{S} \Rightarrow \mathrm{SS} \\
& \text { 2. } \mathrm{S} \Rightarrow \mathrm{aSa} \\
& \text { 3. } \mathrm{S} \Rightarrow \mathrm{aa} \\
& \text { 4. } \mathrm{S} \Rightarrow \mathrm{~b}
\end{array}
$$

$$
\mathrm{i}=2 \quad \text { 1. } \mathrm{S} \Rightarrow \mathrm{SS} \Rightarrow \mathrm{SSS}
$$

$$
\text { 2. } \mathrm{S} \Rightarrow \mathrm{SS} \Rightarrow \mathrm{aSaS}
$$

$$
\text { 3. } \mathrm{S} \Rightarrow \mathrm{SS} \Rightarrow \mathrm{aaS}
$$

$$
\text { 4. } \mathrm{S} \Rightarrow \mathrm{SS} \Rightarrow \mathrm{bS}
$$

$$
\text { 5. } \mathrm{S} \Rightarrow \mathrm{aSa} \Rightarrow \mathrm{aSSa}
$$

$$
\text { 6. } \mathrm{S} \Rightarrow \mathrm{aSa} \Rightarrow \mathrm{aaSaa}
$$

$$
\text { 7. } \mathrm{S} \Rightarrow \mathrm{aSa} \Rightarrow \text { aаaа }
$$

$$
\text { 8. } \mathrm{S} \Rightarrow \mathrm{aSa} \Rightarrow \mathrm{aba}
$$

Definition Simple grammar (or s-grammar) has all productions of the form:

$$
\mathrm{A} \rightarrow \mathrm{ax}
$$

where $A \in V, a \in T$, and $x \in V^{*}$ AND any pair ( $\mathrm{A}, \mathrm{a}$ ) can occur in at most one rule.

Ambiguity
Definition: A CFG G is ambiguous if $\exists$ some $w \in L(G)$ which has two distinct derivation trees.

Example Expression grammar
$\mathbf{G}=(\{\mathbf{E}, \mathbf{I}\},\{\mathbf{a}, \mathbf{b},+, *,()\},, \mathbf{E}, \mathbf{P}), \mathbf{P}=$

$$
\begin{aligned}
& \mathbf{E} \rightarrow \mathbf{E}+\mathbf{E}|\mathbf{E} * \mathbf{E}|(\mathbf{E}) \mid \mathbf{I} \\
& \mathbf{I} \rightarrow \mathbf{a} \mid \mathbf{b}
\end{aligned}
$$

Derivation of $a+b * a$ is:
$\mathbf{E} \Rightarrow \underline{\mathbf{E}}+\mathbf{E} \Rightarrow \underline{\mathrm{I}}+\mathbf{E} \Rightarrow \mathbf{a}+\underline{\mathbf{E}} \Rightarrow \mathbf{a}+\underline{\mathbf{E}} * \mathbf{E} \Rightarrow$
$\mathbf{a}+\mathbf{I} * \mathbf{E} \Rightarrow \mathbf{a}+\mathbf{b} * \mathbf{E} \Rightarrow \mathbf{a}+\mathbf{b} * \mathbf{I} \Rightarrow \mathbf{a}+\mathbf{b} * \mathbf{a}$
Corresponding derivation tree is:

Another derivation of $\mathbf{a}+\mathbf{b} * \mathbf{a}$ is:
$\mathbf{E} \Rightarrow \mathbf{E} * \mathbf{E} \Rightarrow \mathbf{E}+\mathbf{E} * \mathbf{E} \Rightarrow \mathbf{I}+\mathbf{E} * \mathbf{E} \Rightarrow$ $\mathbf{a}+\mathbf{E} * \mathbf{E} \Rightarrow \mathbf{a}+\mathbf{I} * \mathbf{E} \Rightarrow \mathbf{a}+\mathbf{b} * \mathbf{E} \Rightarrow \mathbf{a}+\mathbf{b} * \mathbf{I} \Rightarrow$ $a+b * a$

Corresponding derivation tree is:

Rewrite the grammar as an unambiguous grammar. (with meaning that multiplication has higher precedence than addition)

$$
\begin{aligned}
& \mathbf{E} \rightarrow \mathbf{E}+\mathbf{T} \mid \mathbf{T} \\
& \mathbf{T} \rightarrow \mathbf{T} * \mathbf{F} \mid \mathbf{F} \\
& \mathbf{F} \rightarrow \mathbf{I} \mid \mathbf{E}) \\
& \mathbf{I} \rightarrow \mathbf{a} \mid \mathbf{b}
\end{aligned}
$$

There is only one derivation tree for $\mathrm{a}+\mathrm{b} * \mathrm{a}$ :

Definition If $L$ is CFL and $G$ is an unambiguous CFG s.t. $L=L(G)$, then L is unambiguous.

Backus-Naur Form of a grammar:

- Nonterminals are enclosed in brackets <>
- For " $\rightarrow$ " use instead $":==$

Sample C++ Program:
main ()
\{
int a ; int b ; int sum;
$\mathrm{a}=40 ; \quad \mathrm{b}=6 ; \quad$ sum $=\mathrm{a}+\mathrm{b}$;
cout << "sum is "<< sum << endl;
\}
"Attempt" to write a CFG for C++ in BNF (Note: <program> is start symbol of grammar.)
$<$ program> ::= main () <block>
$<$ block $>\quad::=\{<$ stmt-list $>\}$
$<$ stmt-list $>:=<$ stmt $>\mid<$ stmt $><$ stmt-list $>$

$$
<\text { decl }>\mid<\text { decl }><\text { stmt-list }>
$$

$<$ decl $>\quad::=$ int $<$ id $>; \mid$ double $<$ id $>$;
$<$ stmt $>\quad::=<$ asgn-stmt $>\mid<$ cout-stmt $>$
$<$ asgn-stmt $>:=<$ id $>=<$ expr $>$;
$<$ expr $>\quad::=<$ expr $>+<$ expr $>$

$$
\begin{aligned}
& \mid<\text { expr }>*<\text { expr }> \\
& \mid(<\text { expr }>) \mid<\text { id }>
\end{aligned}
$$

<cout-stmt>: = cout <out-list> ;
etc., Must expand all nonterminals!

## So a derivation of the program test would look like:

$<$ program $>\Rightarrow$ main () $<$ block $>$ $\Rightarrow$ main () $\{<$ stmt-list $>\}$
$\Rightarrow$ main () $\{<$ decl $><$ stmt-list $>$
$\Rightarrow$ main () $\{$ int $<$ id $>;<$ stmt-list $\rangle$
$\Rightarrow$ main () $\{$ int a ; <stmt-list> $\}$
$\stackrel{*}{\Rightarrow}$ complete $C++$ program

More on CFG for C++
We can write a CFG G s.t.
$\mathrm{L}(\mathrm{G})=\{$ syntactically correct $\mathrm{C}++$ programs $\}$.

But note that \{semantically correct $\mathbf{C}++$ programs $\} \subset \mathbf{L}(\mathbf{G})$.

Can't recognize redeclared variables:

int x ;<br>double $x$;

Can't recognize if formal parameters match actual parameters in number and types:

## declar: int Sum(int $a$, int $b$, int $c$ ) ...

 call: $\quad$ newsum $=\operatorname{Sum}(x, y)$;