Context-Free Languages   Chapter 5

Regular languages:

- keywords in a programming language

- names of identifiers

- integers

- all misc symbols: $=$;

Not Regular languages:

- $\{a^n c b^n | n > 0\}$

- expressions - $((a+b)-c)$

- block structures ({} in Java/C++ and begin ... end in pascal)

1

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

$$A \rightarrow x$$

Where A∈V and x∈(V∪T)*.

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

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

$$S \to aSb \mid ab$$

Derivation of aaabbb:

$$S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow aaabbb$$

$$L(G) = \{a^n b^n \mid n > 0\}$$

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

$$\mathbf{S} \to \mathbf{aSa} \mid \mathbf{bSb} \mid \mathbf{a} \mid \mathbf{b} \mid \lambda$$

Derivation of ababa:

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

$$\Sigma = \{a, b\}, \ \mathbf{L(G)} = \{ w \in \Sigma^* \mid w = w^R \}$$

STOPPED!

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

$$S \rightarrow AcB$$
$$A \rightarrow aAa \mid \lambda$$
$$B \rightarrow Bbb \mid \lambda$$

L(G) =

Derivations of aacbb:

1. S $\Rightarrow$ $\underline{A}$cB $\Rightarrow$ a$\underline{A}$acB $\Rightarrow$ aac$\underline{B}$ $\Rightarrow$ aac$\underline{B}$bb $\Rightarrow$ aacbb

2. S $\Rightarrow$ Ac$\underline{B}$ $\Rightarrow$ Ac$\underline{B}$bb $\Rightarrow$ $\underline{A}$cbb $\Rightarrow$ a$\underline{A}$acbb $\Rightarrow$ 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 G=(V,T,S,P):

- root is labeled S

- leaves labeled x, where x$\in$T$\cup\{\lambda\}$

- nonleaf vertices labeled A, A$\in$V

- For rule A$\rightarrow a_1 a_2 a_3 \ldots a_n$, where A$\in$V, $a_i \in$(T$\cup$V$\cup\{\lambda\}$),
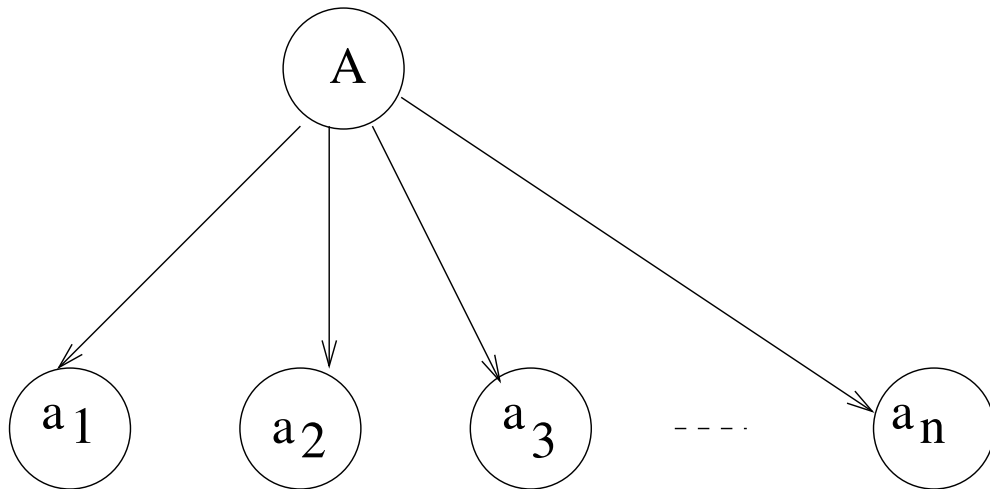
Example: G=({S,A,B},{a,b,c},S,P)

$$S \rightarrow AcB$$
$$A \rightarrow aAa \mid \lambda$$
$$B \rightarrow Bbb \mid \lambda$$

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 w∈L(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
w∈ Σ^*, is w∈L(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

G=({S}, {a,b}, S, P), P=

$$\mathbf{S} \rightarrow \mathbf{SS} \mid \mathbf{aSa} \mid \mathbf{b} \mid \lambda$$

$L_1=\mathbf{L(G)} =$

Is abbab $\in$ L(G)?

Exhaustive Search Algorithm

For all i=1,2,3,...
   Examine all sentential forms yielded
   by i substitutions

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

Theorem If CFG G does not contain rules of the form

$$A \to \lambda$$
$$A \to B$$

where A,B∈V, then we can determine if w∈ L(G) or if w∉ 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=L(G) where G is:

$$S \rightarrow SS \mid aa \mid aSa \mid b$$

Show baaba $\notin$ L(G).

i=1 1. S $\Rightarrow$ SS
     2. S $\Rightarrow$ aSa
     3. S $\Rightarrow$ aa
     4. S $\Rightarrow$ b

i=2 1. S $\Rightarrow$ SS $\Rightarrow$ SSS
     2. S $\Rightarrow$ SS $\Rightarrow$ aSaS
     3. S $\Rightarrow$ SS $\Rightarrow$ aaS
     4. S $\Rightarrow$ SS $\Rightarrow$ bS
     5. S $\Rightarrow$ aSa $\Rightarrow$ aSSa
     6. S $\Rightarrow$ aSa $\Rightarrow$ aaSaa
     7. S $\Rightarrow$ aSa $\Rightarrow$ aaaa
     8. S $\Rightarrow$ aSa $\Rightarrow$ aba

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

$$A \rightarrow ax$$

where $A \in V$, $a \in T$, and $x \in V^*$ AND any pair (A,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

G=({E,I}, {a,b,+,∗,(,)}, E, P), P=

$$E \rightarrow E+E \mid E*E \mid (E) \mid I$$
$$I \rightarrow a \mid b$$

Derivation of a+b∗a is:

E ⇒ E+E ⇒ I+E ⇒ a+E ⇒ a+E∗E ⇒ a+I∗E ⇒ a+b∗E ⇒ a+b∗I ⇒ a+b∗a

Corresponding derivation tree is:

Another derivation of a+b∗a is:

$$E \Rightarrow \underline{E}*E \Rightarrow \underline{E}+E*E \Rightarrow \underline{I}+E*E \Rightarrow$$
$$a+\underline{E}*E \Rightarrow a+\underline{I}*E \Rightarrow a+b*\underline{E} \Rightarrow a+b*\underline{I} \Rightarrow$$
$$a+b*a$$

Corresponding derivation tree is:

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

$$E \rightarrow E+T \mid T$$
$$T \rightarrow T*F \mid F$$
$$F \rightarrow I \mid (E)$$
$$I \rightarrow a \mid b$$

There is only one derivation tree for a+b*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;
   a = 40;     b = 6;    sum = a + 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>    ::= { <stmt-list> }
<stmt-list> ::= <stmt> | <stmt><stmt-list>
              <decl> | <decl><stmt-list>
<decl>     ::= int <id> ; | double <id> ;
<stmt>     ::= <asgn-stmt> | <cout-stmt>
<asgn-stmt>::= <id> = <expr> ;
<expr>     ::= <expr> + <expr>
              | <expr> * <expr>
              | ( <expr> ) | <id>
<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$>$
$\Rightarrow$ main () { int a ; $<$stmt-list$>$ }
$\overset{*}{\Rightarrow}$ *complete C++ program*

More on CFG for C++

We can write a CFG G s.t.
L(G)={syntactically correct C++
programs}.

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

Can't recognize redeclared variables:

int x;
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: newsum = Sum(x,y);