Part II

Pushdown Automata and Context-Free Languages

Chapter 1

Context-Free Grammars and Languages

(Lecture 19,20)

Bacus-Naur Form

- Regular Expr is too limited to describe practical programming languages.
- Bacus-Naur Form (BNF)
  - A popular formalism used to describe practical programming languages:
  - Example BNF: (p116)

<stmt>	::= <if-stmt> | <while-stmt> | <begin-stmt> | <assg-stmt>
<if-stmt>	::= if <bool-expr> then <stmt> else <stmt>
<while-stmt>	::= while <bool-expr> do <stmt>
<begin_stmt>	::= begin <stmt-list> end
<stmt-list>	::=<stmt> | <stmt> ; <stmt-list>
<assg-stmt>	::= <var> := <arith-expr>
<bool-expr>	::= <arith-expr> <comp-op> <arith-expr>
<comp-op>	::= < | > | <= | >= | NEQ
<arith-expr>	::= <var> | <const> | (<arith-expr> <arith-op> <arith-expr>)
<arith-op>	::= + | - | * | /
<const>	::= 0 | 1 | 2 | ... |8 | 9
<var>	::= a | b | c |... |x | y | z

Derivations

- Question: how to determine if the string

  while x > y do begin x := (x+1); y := (y-1) end

  belongs to the language represented by the above grammar?

Sol: Since the string can be derived from the grammar.

- <stmt>

<while-stmt>

while <bool-expr> do <stmt>

while <arith-expr><compare-op><arith-expr> do <stmt>

while <var><compare-op><arith-expr> do <stmt>

while <var> > <arith-expr> do <stmt>

while x > <var> do <stmt>

while x > y do <stmt>

….

while x > y do begin x := (x+1); y := (y-1) end

CFGs

- **Facts:**
  - 1. each nonterminal symbol can derive many different strings.
  - 2. Every string in a derivation is called a sentential form.
  - 3. Every sentential form containing no nonterminal symbols is called a sentence.
  - 4. The language L(G) generated by a CFG G is the set of sentences derivable from a distinguished nonterminal called the start symbol of G. (eg. <stmt>)
  - 5. A language is said to be *context free* (or a context free language (CFL)) if it can be generated by a CFG.
  - A sentence may have many different derivations; a grammar is called unambiguous if this cannot happen
  - (eg: previous grammar is unambiguous)

CFGs: related facts

- CFG are more expressive than FAs (and regular expressions) (i.e., all regular languages are context-free, but not vice versa.)
- Example CFLs which are not regular:
  - $\{a^n b^n \mid n \geq 0\}$
  - {Palindrome over {a,b}} = $\{x \in \{a,b\}^* \mid x = rev(x)\}$
  - {balanced strings of parentheses}
- Not all sets are CFLs:
  - Ex: $\{a^n b^n c^n \mid n \geq 0\}$ is not context-free.

CFGs and CFLs: a formal defintion

- a CFG is a quadruple $G = (N,\Sigma,P,S)$ where
  - N is a finite set (of nonterminal symbols)
  - Σ is a finite set (of terminal symbols) disjoint from N.
  - $S \in N$ is the start symbol.
  - P is a a finite subset of $N \times (N \cup \Sigma)^*$ (The productions)
- Conventions:
  - nonterminals: A,B,C,…
  - terminals: a,b,c,…
  - strings in $(N \cup \Sigma)^*$: $\alpha,\beta,\gamma,\ldots$
  - Each $(A,\alpha) \in P$ is called a production rule and is usually written as: $A \rightarrow \alpha$.
  - A set of rules with the same LHS:

    $A \rightarrow \alpha_1 \quad A \rightarrow \alpha_2 \quad A \rightarrow \alpha_3$ can be abbreviated as

    $A \rightarrow \alpha_1 \mid \alpha_2 \mid \alpha_3$.

Derivations

- Let $\alpha,\beta \in (N \cup \Sigma)^*$ we say β is derivable from α in one step, in symbols, $\alpha \rightarrow_G \beta$

  (G may be omitted if there is no ambiguity)

  if β can be obtained from α by replacing some occurrence of a nonterminal symbol A in α with γ, where $A \rightarrow \gamma \in P$; i.e.,

  if there exist $\alpha_1,\alpha_2 \in (N \cup \Sigma)^*$ and production $A \rightarrow \gamma$ s.t.

  $\alpha = \alpha_1 A \alpha_2$ and $\beta = \alpha_1 \gamma \alpha_2$.
- Let $\rightarrow^*_G$ be the reflexive and transitive closure of $\rightarrow_G$, i.e.,

  define $\alpha \rightarrow^0_G \alpha$ for any α

  $\alpha \rightarrow^{k+1}_G \beta$ iff there is γ s.t. $\alpha \rightarrow^k_G \gamma$ and $\gamma \rightarrow_G \beta$.

  Then $\alpha \rightarrow^*_G \beta$ iff $\exists$ $k \geq 0$ s.t. $\alpha \rightarrow^k_G \beta$.
- Any string in $(N \cup \Sigma)^*$ derivable from S (i.e., $S \rightarrow^*_G \alpha$) is called a **sentential form**, in particular, if α is a terminal string (i.e., $\alpha \in \Sigma^*$), α is called a **sentence**.

Language generated by a CFG

- The language generated by G, denoted L(G), is the set

$$L(G) =_{def} \{ x \in \Sigma^* \mid S \rightarrow^*_G x \}.$$

- A language $B \subseteq \Sigma^*$ is a context-free language (CFL) if B = L(G) for some CFG G.

Ex 19.1: The nonregular set $A = \{a^n b^n \mid n \geq 0 \}$ is a CFL. Since it can be generated by the grammar G:

$S \rightarrow \varepsilon \mid aSb$

or more precisely $G = (N,\Sigma,P,S)$ where

- $N = \{S\}$
- $\Sigma = \{a,b\}$
- $P = \{ S \rightarrow \varepsilon, S \rightarrow aSb \}$

- $a^3b^3 \in L(G)$ since $S \rightarrow aSb \rightarrow aaSbb \rightarrow aaaSbbb \rightarrow aaabbb$.

$\therefore S \rightarrow^4 aaabbb$ and $S \rightarrow^* aaabbb$

Techniques for show L = L(G)

- But how to show that $L(G) = A$ ($= \{a^n b^n \mid n \geq 0\}$) ?
- a consequence of the following lemmas:
  - Lem 1: $S \rightarrow^{n+1} a^n b^n$ for all $n \geq 0$.
  - Lemm2: If $S \rightarrow^* x$ ==> x is of the form $a^k S b^k$ or $a^k b^k$.

    (in particular, if x is a sentence => $x \in A$).
- Pf: of lem 1:
  - by ind. on n. $n = 0$ ==> $S \rightarrow e$. (ok)
  - $n = k+1 > 0$: By ind. hyp. $S \rightarrow^{k+1} a^k b^k$
  - $\therefore S \rightarrow aSb \rightarrow^{k+1} a^{k+1} b^{k+1}$. $\therefore S \rightarrow^{n+1} a^n b^n$.
- Pf of lem2: by ind. on k s.t. $S \rightarrow^k x$.
  - $k = 0 \Rightarrow S \rightarrow^0 S = a^0 S b^0$.
  - $k = t+1 > 0$. $S \rightarrow^t a^m S b^m \rightarrow a^m a S b b^m$ (ok) or
  - $\rightarrow a^m b^m$. (ok).

Balanced Parentheses

Ex 19.2: The set of palindromes P = { x ∈ {a,b}* | x = rev(x) }.
can be generated by the grammar G:
S → ε | a | b | aSa | bSb.

cf: The inductive definition of P
1. Initial condition: ε, a and b are palindromes.
2. recursive condition:
   If S is a palindrome, then so are aSa and bSb.

- Balanced Parentheses:

  Ex1: 2+3x5-4x6 ==> ((2+3)x(5-(4x6)))
  ==> (( ) (( )))
  --- balanced parentheses.

  Ex2: unbalanced parentheses:
  (( )) ((( ))))) --- no of "(" ≠ no of ")".
  (( ) (( )))) ((( )) --- unmatched ")" encountered.

Balanced Parentheses

- Formal definition:
- let $\Sigma \supseteq \{ [,] \}$. Define L,R: $\Sigma^* \rightarrow N$ as follows:
  - L(x) = number of "[" in x.
  - R(x) = number of "]" in x.
- a string $x \in \Sigma^*$ is said to be *balanced* iff
  (i) L(x) = R(x) -- equal # of left and right parentheses.
  (ii) for all prefix y of x, L(y) $\geq$ R(y).
  --- no dangling right parenthesis.
- Now define PAREN = $\{ x \in \{ [,] \}^* \mid x \text{ is balanced} \}$.
- Thm 20.1 : PAREN can be generated by the CFG G:

$$S \rightarrow \varepsilon \mid [S] \mid S\,S$$

pf: 1. L(G) $\subseteq$ PAREN.

Lem1: If $S \rightarrow^* x$ then x is balanced. In particular, if x contains no S => $x \in$ PAREN. $\therefore$ L(G) $\subseteq$ PAREN.

proof of theorem 20.1

pf of lem1 : by ind. on k s.t. $S \rightarrow^k x$.

$k = 0 \Rightarrow S \rightarrow^0 S = x$ is balanced.

$k = t + 1 > 0$:

$\Longrightarrow\ S \rightarrow^t ySz \rightarrow yz$ (1) or

$\rightarrow y[S]z$ (2) or

$\rightarrow ySSz$ (3).

By ind. hyp., ySz is balanced.

$\Rightarrow L(yz) = L(ySz) = R(ySz) = R(yz)$ and

if $y = wu \Rightarrow L(w) \geq R(w)$ since w is also a prefix of ySz.

if $z = wu \Rightarrow L(yw) = L(ySw) \geq R(ySw) = R(yw)$.

$\therefore$ yz is balanced.

Case (2) and (3) can be proved similarly.

Proof of theoreom 20.1 (cont'd)

Pf: PAREN $\subseteq$ L(G) (i.e., if x is balanced ==> S $\rightarrow$ * x.)

By ind. on |x|.

1. |x| = 0 ==> x = ε ==> S $\rightarrow$ ε (ok).
2. |x| > 0. Then either
   (a) $\exists$ a proper prefix y of x that is balanced or
   (b) No proper prefixes y of x are balanced.

- In case (a), we have x = y z with |y|,|z| < |x| for some z.

=> L(z) = L(x) - L(y) = R(x) - R(y) = R(z)

For all prefix w with z = w w': L(w) = L(yw) - L(y) $\geq$ R(yw) - R(y) = R(w)

==> both y and z are balanced ==> by ind. hyp., S $\rightarrow$* y and S $\rightarrow$* z

==> S $\rightarrow$ SS $\rightarrow$* yS $\rightarrow$*yz.

In case (b): x = [z] for some z (why ?)

Moreover it can be shown that z is balanced too.

Hence S $\rightarrow$* z. ==> S $\rightarrow$* [S] $\rightarrow$* [z] = x. QED

Pushdown Automata: a preview

- FAs recognize regular languages.
- What kinds of machines recognize CFLs ?

  ===> Pushdown automata (PDAs)
- PDA:
  - Like FAs but with an additional *stack* as working memory.
  - Actions of a PDA
    1. Move right one tape cell (as usual FAs)
    2. push a symbol onto stack
    3. pop a symbol from the stack.
  - Actions of a PDA depend on
    1. current state 2. currently scanned I/P symbol
    3. current top stack symbol.
  - A string x is accepted by a PDA if it can enter a final state (or clear all stack symbols) after scanning the entire input.
  - More details defer to later chapters.