CSCE 355
3/3/2025

Context-free languages

Used to describe programming language syntax.

Def: A context-free grammar (CFG) is a tuple $\langle V, T, S, P \rangle$ where

V - a finite set. Elements are called variables, nonterminals, syntactic categories

(also Σ) T - an ~~finite~~ alphabet (elements are called terminals or tokens

~~$V \cap T = \emptyset$~~

$S \in V$ (the start symbol)

P is a finite set of productions (or rewrite rules), each one has the form

$$\underbrace{A}_{\text{head}} \rightarrow \underbrace{\alpha}_{\text{body}}$$

where $A \in V$ and α is any string in $(V \cup T)^*$.

Idea:
- $A \rightarrow \alpha$ means "an A could be an α" or "A can be replaced by α"
- S — the broadest syntactic category.

6 productions that describe syntax of expressions over the arithmetic operators +, −, *, / & constants & parens

Ex:

$E \to E + E$
$E \to E - E$
$E \to E * E$
$E \to E / E$
$E \to (E)$
$E \to c$

Think E — "expression"

("constant")

string $(c + c) * c$

$E \Rightarrow E * E \Rightarrow (E) * E \Rightarrow (E + E) * E$

$\Rightarrow (c + E) * E \Rightarrow (c + c) * E \Rightarrow (c + c) * c$

So $(c + c) * c$ is a syntactically correct arithmetic expression.

Formally, the grammar is

$G = \langle \{E\}, \{+, -, *, /, '(', ')', c\}, E, \{E \to E + E, \ldots, E \to c\} \rangle$

Usually just give the list of productions:
- nonterminals are the left-hand sides (heads) of the productions
- terminals are any symbols appearing on the right-hand sides (bodies) that are not nonterminals
- start symbol is head of the first production in the list

Nonterminals — upper case roman letters

lang of statements (S) in a prog. lang

Ex: $S \to \{ L \}$ (L = "list of statements")

$S \to$ while E do S

$S \to$ if E then S

$S \to$ if E then S else S

$S \to$ assignment ;

as before

$E \to$

⋮

$E \to$

$L \to S\ L$

$L \to \varepsilon$ (ε is metasymbol denoting the empty string)

$L \Rightarrow S\ L \Rightarrow S\ S\ L \Rightarrow S\ S\ S\ L \Rightarrow S\ S\ S$

Def: Let $G = \langle V, \Sigma, S, P \rangle$ be a CFG.

A derivation in G of a string ~~of~~ $w \in \Sigma^*$ is a sequence ~~of strings~~

$$A_0 \Rightarrow A_1 \Rightarrow A_2 \Rightarrow \cdots \Rightarrow A_n$$

where all $A_i \in (V \cup \Sigma)^*$

- $A_0 = S$
- $A_n = w$
- For all i, $0 \le i < n$, A_{i+1} is obtained from A_i by replacing an occurrence

of some $B \in V$ with the body of a production in P, whose head is B.

Ex: $abBcd \Rightarrow abxyzcd$,

where $B \to xyz$ is a production in P.

$A_0, A_1, \ldots, A_n$ — sentential forms

$A_0 \Rightarrow \cdots \Rightarrow w$ is derivation of w

say that w is derivable from G

Def: Let G be a CFG with terminal alphabet Σ

$L(G) \subseteq \Sigma^*$:

$$L(G) = \{ w \in \Sigma^* : w \text{ is derivable from } G \}$$

~~Ambiguity~~ Def: A derivation is leftmost (resp. rightmost) if in each step, the leftmost (respectively rightmost) nonterminal of the sentential form is replaced.

Fact: If w is derivable, then it has a leftmost and a rightmost derivation.

Ambiguity: A CFG is ambiguous if there is some string in $L(G)$ that has more than one leftmost derivation.

Recall
$E \to E + E$
$E \to E * E$
$E \to (E)$
$E \to c$
} ambiguous

Let $w = c + c * c$

1 leftmost derivation: $E \to E + E \Rightarrow c + E$
$\Rightarrow c + E * E$
$\Rightarrow c + c * E$
$\Rightarrow c + c * c$

another

$E \Rightarrow E * E \Rightarrow E + E * E$
$\Rightarrow c + E * E \Rightarrow c + c * E \Rightarrow c + c * c$

Equivalent
Unambiguous grammars are preferable (if the exist)

An equiv. unambiguous grammar for arith exprs:

$E \to E + T$
$E \to T$
$T \to T * F$
$T \to F$
$F \to c$
$F \to (E)$

E = "expression"
T = "term"
F = "factor"

$E \Rightarrow E + T \Rightarrow E + T + T \Rightarrow T + T + T$
$T \Rightarrow T * F \Rightarrow T * F * F \Rightarrow F * F * F$

$c + c * c$ — has only 1 leftmost derivation;

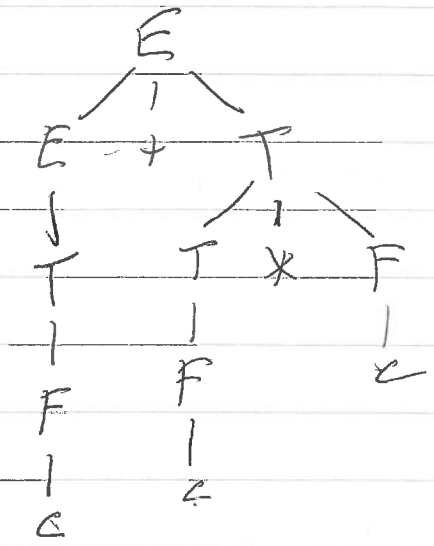
$E \Rightarrow E+T \Rightarrow T+T \Rightarrow F+T \Rightarrow c+T$

$\Rightarrow c+T*F \Rightarrow c+F*F$

$\Rightarrow c+c*F \Rightarrow c+c*c$

Parse Trees (alternative to derivation)

Ex: for $c+c*c$:



Def: Given a CFG G,

a parse tree of G

is a rooted ordered tree where

- internal nodes have nonterminal labels
- leaves have terminal labels (or ε)
- root labeled by the start symbol.
- the child of each internal node (label A, say) read left to right from the body of some production with head A)

The ~~yeald~~ yield of a parse tree T is the ~~the~~ in-order concatenation of its leaves.