Context-free languages

Used to describe programming language syntax:

- arith expressions
- assignments
- function calls & definitions
- structured statements
  - sequences
  - control-flow:
    - loops
    - if-then, if-then-else

A grammar (context-free grammar) is a finite set of "rewrite" rules, call productions (with some extra info)

Ex:

1. $\overbrace{S}^{\text{"head"}} \to \overbrace{0S1}^{\text{"body"}}$
2. $S \to \varepsilon$

Can replace S in a string with the right-hand side of one of these rules

Ex: $S \underset{(1)}{\Rightarrow} 0S1 \underset{(1)}{\Rightarrow} 00S11 \underset{(1)}{\Rightarrow} 000S111 \underset{(2)}{\Rightarrow} \underset{\text{stop}}{000111}$

Derives all strings (from S) of the form $0^n 1^n$ for some $n \geq 0$

$$S \underset{(2.)}{\Rightarrow} \varepsilon$$

Given a grammar, can at any time replace an occurrence of the *head* of some production by its *body* (in place)

Definition: A *context-free grammar* (*CFG*) is a 4-tuple $\langle V, \Sigma, S, P \rangle$ where

V is a finite set (elements are called *variables*, *nonterminals*, or *syntactic categories*) [V is an alphabet]

Σ is an alphabet (elements are called *terminals* or *tokens*)

Have $V \cap \Sigma = \emptyset$

$S \in V$ (the *start symbol*)

P is a finite set of *productions*,

A production is an expression of the form

$$A \to \alpha$$

~~where~~ for some $A \in V$ (the head) and some string $\alpha \in (V \cup \Sigma)^*$

[elements of $V \cup \Sigma$ are <u>grammar symbols</u>]

<u>Ex:</u> $\langle \{S\}, \{0,1\}, S, \{S \to 0S1, S \to \varepsilon\} \rangle$ from before.

<u>Ex:</u> $\Sigma = \{a, b, c\}$, start symbol S

we want to derive (from S) all strings of the form $a^m b^n c^m$ (for $m, n \geq 0$):

1. $S \to aSc$
2. $S \to bS$
3. $S \to \varepsilon$

Derive $aabcc$ $(m=2, n=1)$:

$$S \underset{(1.)}{\Rightarrow} aSc \underset{(1.)}{\Rightarrow} aaScc \underset{(2.)}{\Rightarrow} aabScc \underset{(3.)}{\Rightarrow} aabcc$$

Can derive strings not of this form:

$$S \overset{(2.)}{\Rightarrow} bS \overset{(1.)}{\Rightarrow} baSc \overset{(3.)}{\Rightarrow} bac$$

not the right form.

Want to derive $a^m b^n c^m$ but no other strings;

$V = \{S, T\}$

start symbol S

Productions

$S \to aSc$

$S \to T$

$T \to bT$

$T \to \varepsilon$

derive: aabcc:

$$S \Rightarrow aSc \Rightarrow aaScc \Rightarrow aaTcc \Rightarrow aabTcc \Rightarrow aabcc$$

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

A derivation of G is a sequence of the form,

$$\alpha_0 \Rightarrow \alpha_1 \Rightarrow \cdots \Rightarrow \alpha_n \quad (n \geq 0)$$

where —

each $\alpha_i \in (V \cup \Sigma)^*$ (string of grammar symbols)

$\forall i, 0 \leq i \leq n$

and

for each i, $0 \le i < n$, α_{i+1} is obtained from α_i by replacing a single occurrence of a variable in α_i with the body of a production with that variable as the head.
That is, for each i, there is a production $A \to \beta \in P$ and strings $\delta, \gamma \in (V \cup \Sigma)^*$ such that

$$\alpha_i = \delta \underline{A} \gamma \quad \text{and}$$

$$\alpha_{i+1} = \delta \underline{\beta} \gamma .$$

If $\alpha_0 = S$ (start symbol) and $\alpha_n \in \Sigma^*$, say this is a <u>complete derivation</u> (<u>ending</u> in α_n).

<u>Def</u>: $G = \langle V, \Sigma, S, P \rangle$ a CFG. The <u>language of G</u> is defined as

$$L(G) := \{ w \in \Sigma^* : \text{there is a complete derivation of } G \text{ ending in } w \}$$

In $\alpha_0 \Rightarrow \cdots \Rightarrow \alpha_n$, abbrev as $\alpha_0 \Rightarrow^* \alpha_n$

α_n is <u>derivable</u> from α_0

α_n is derivable from G means

α_n is " from G's start symbol. So

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

<u>Def</u>: A language $L \subseteq \Sigma^*$ is <u>context-free</u> (a <u>CFL</u>) if $L = L(G)$ for some CFG G with terminal alphabet Σ.

<u>Ex</u>: $\{0^n 1^n : n \geq 0\}$ is context-free (but not regular)

$\{a^m b^n c^n : m, n \geq 0\}$ is a CFL.

Some useful examples:

The lang. of all well-balanced parens

e.g.

$$\left.\begin{array}{ll} (()) & \checkmark \\ ()() & \checkmark \end{array}\right\} \text{well-balanced}$$

$$\left.\begin{array}{ll} ())) & \times \\ (() & \times \\ )(& \times \end{array}\right\} \text{not well-balanced}$$

CFG for well-balanced parens: $\langle \{S\}, \{'(', ')'\}, S, P \rangle$ 7

where P is

$$\boxed{S} \to (S)$$
$$S \to SS$$
$$S \to \varepsilon$$

$$S \Rightarrow (S) \Rightarrow ((S)) \Rightarrow (())$$

or

$$S \Rightarrow SS \Rightarrow (S)(S) \Rightarrow ()(S) \Rightarrow ()()$$

Arith exprs using $+, -, *, /, '(', ')'$

bin ops

over constants. Use 'c' to mean any constant

$E \to c$

~~$E \to E$~~

$E \to E + E$

$E \to E - E$

$E \to E * E$

$E \to E / E$

$E \to (E)$

shorthand

$E \to c \mid E+E \mid E-E \mid \cdots \mid (E)$

or

$S \to (S) \mid SS \mid \varepsilon$

etc.

Derive $c*(c+c)$:

$$E \Rightarrow \underline{E} * E \Rightarrow c * \underline{E} \Rightarrow c * (\underline{E}) \Rightarrow c*(\underline{E}+\underline{E}) \Rightarrow c*(\underline{c}+E)$$
$$\Rightarrow c*(c+c),$$

Leftmost derivation: always replace the leftmost occurrence of a variable in each string.

~~For every~~ Every string in $L(G)$ has a leftmost derivation.