Context-Free Grammars

Informal Comments

- A *context-free grammar* is a notation for describing languages.
- It is more powerful than finite automata or RE's, but still cannot define all possible languages.
- Useful for nested structures, e.g., parentheses in programming languages.

Informal Comments – (2)

- Basic idea is to use "variables" to stand for sets of strings (i.e., languages).
- These variables are defined recursively, in terms of one another.
- Recursive rules ("productions") involve only concatenation.
- Alternative rules for a variable allow union.

Example: CFG for { $0^n1^n \mid n \geq 1$ }

- Productions:

  S -> 01

  S -> 0S1
- Basis: 01 is in the language.
- Induction: if w is in the language, then so is 0w1.

Example: CFG for palindromes

- Productions:

  P -> ϵ

  P -> 0

  P -> 1

  P -> 0P0

  P -> 1P1

- Basis: ϵ, 0, 1 are palindromes.
- Induction: if w is in the language, then so are 0w0 and 1w1.

Shorthand Forms

- Several productions can be merged into a shorthand form using the "|" sign
- Productions for 0^n1^n:

  S -> 01 | 0S1
- Productions for palindromes:

  P -> ϵ | 0 | 1 | 0P0 | 1P1

CFG Formalism

- *Terminals* = symbols of the alphabet of the language being defined.
- *Variables* = *nonterminals* = a finite set of other symbols, each of which represents a language.
- *Start symbol* = the variable whose language is the one being defined.

Productions

- A *production* has the form variable -> string of variables and terminals.
- Convention:
  - A, B, C,… are variables.
  - a, b, c,… are terminals.
  - …, X, Y, Z are either terminals or variables.
  - …, w, x, y, z are strings of terminals only.
  - α, β, γ,… are strings of terminals and/or variables.

Example: Formal CFG

- Here is a formal CFG for $\{ 0^n1^n \mid n \geq 1\}$.
- Terminals = {0, 1}.
- Variables = {S}.
- Start symbol = S.
- Productions =

  S -> 01

  S -> 0S1

Derivations – Intuition

- We *derive* strings in the language of a CFG by starting with the start symbol, and repeatedly replacing some variable A by the right side of one of its productions.
  - That is, the "productions for A" are those that have A on the left side of the ->.

Derivations – Formalism

- We say $\alpha A\beta \Rightarrow \alpha\gamma\beta$ if $A \rightarrow \gamma$ is a production.
- Example: S -> 01; S -> 0S1.
- S => 0S1 => 00S11 => 000111.

Example: Iterated Derivation

=>* means "zero or more derivation steps."

- S -> 01; S -> 0S1.
- S => 0S1 => 00S11 => 000111.
- So S =>* S; S =>* 0S1; S =>* 00S11; S =>* 000111.

Sentential Forms

- Any string of variables and/or terminals derived from the start symbol is called a *sentential form*.
- Formally, α is a sentential form iff

$$S =>^* \alpha$$

Language of a Grammar

- If G is a CFG, then L(G), the *language of G*, is {w | S =>* w}.
  - Note: w must be a terminal string, S is the start symbol.
- Example: G has productions S -> ε and S -> 0S1.
- L(G) = $\{0^n1^n \mid n \geq 0\}$.

Note: ε is a legitimate right side.

Context-Free Languages

- A language that is defined by some CFG is called a *context-free language*.
- There are CFL's that are not regular languages, such as the example just given.
- But not all languages are CFL's.
- Intuitively: CFL's can count two things, not three.

Leftmost and Rightmost Derivations

- Derivations allow us to replace any of the variables in a string.
- Leads to many different derivations of the same string.
- By forcing the leftmost variable (or alternatively, the rightmost variable) to be replaced, we avoid these "distinctions without a difference."

Leftmost Derivations

- Say $wA\alpha \Rightarrow_{lm} w\beta\alpha$ if w is a string of terminals only and $A \rightarrow \beta$ is a production.
- Also, $\alpha \Rightarrow^*_{lm} \beta$ if α becomes β by a sequence of 0 or more $\Rightarrow_{lm}$ steps.

Example: Leftmost Derivations

- Balanced-parentheses grammmar:

  S -> SS | (S) | ()

- $S \Rightarrow_{lm} SS \Rightarrow_{lm} (S)S \Rightarrow_{lm} (())S \Rightarrow_{lm} (())()$
- Thus, $S \Rightarrow^*_{lm} (())()$
- S => SS => S() => (S)() => (())() is a derivation, but not a leftmost derivation.

Rightmost Derivations

- Say $\alpha Aw \Rightarrow_{rm} \alpha\beta w$ if w is a string of terminals only and A -> β is a production.
- Also, $\alpha \Rightarrow^*_{rm} \beta$ if α becomes β by a sequence of 0 or more $\Rightarrow_{rm}$ steps.

Example: Rightmost Derivations

- Balanced-parentheses grammmar:

  S -> SS | (S) | ()

- $S =>_{rm} SS =>_{rm} S() =>_{rm} (S)() =>_{rm} (())()$
- Thus, $S =>^*_{rm} (())()$
- S => SS => SSS => S()S => ()()S => ()()() is neither a rightmost nor a leftmost derivation.