Context-Free Grammars

Formalism

Derivations

Backus-Naur Form

Left- and Rightmost Derivations

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 $\{ 0^n 1^n \mid n \geq 1\}$

- Productions:

  S -> 01

  S -> 0S1

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

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 (*head*) -> string of variables and terminals (*body*).
- Convention:
  - A, B, C,… and also S 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 body of one of its productions.
  - That is, the "productions for A" are those that have head A.

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.

Iterated Derivation

- $=>^*$ means "zero or more derivation steps."
- Basis: $\alpha =>^* \alpha$ for any string α.
- Induction: if $\alpha =>^* \beta$ and $\beta => \gamma$, then $\alpha =>^* \gamma$.

Example: Iterated Derivation

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

Language of a Grammar

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

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.

BNF Notation

- Grammars for programming languages are often written in BNF (*Backus-Naur Form*).
- Variables are words in <…>; Example: <statement>.
- Terminals are often multicharacter strings indicated by boldface or underline; Example: **while** or <u>WHILE</u>.

BNF Notation – (2)

- Symbol ::= is often used for ->.
- Symbol | is used for "or."
  - A shorthand for a list of productions with the same left side.
- Example: S -> 0S1 | 01 is shorthand for S -> 0S1 and S -> 01.

BNF Notation – Kleene Closure

- Symbol … is used for “one or more.”
- Example: <digit> ::= 0|1|2|3|4|5|6|7|8|9
  <unsigned integer> ::= <digit>…
- Translation: Replace α… with a new variable A and productions A -> Aα | α.

Example: Kleene Closure

- Grammar for unsigned integers can be replaced by:

  U -> UD | D

  D -> 0|1|2|3|4|5|6|7|8|9

BNF Notation: Optional Elements

- Surround one or more symbols by [...] to make them optional.
- Example: <statement> ::= **if** <condition> **then** <statement> [; **else** <statement>]
- Translation: replace [α] by a new variable A with productions A -> α | ϵ.

Example: Optional Elements

- Grammar for if-then-else can be replaced by:

S -> iCtSA

A -> ;eS | ϵ

BNF Notation – Grouping

- Use {…} to surround a sequence of symbols that need to be treated as a unit.
  - Typically, they are followed by a … for "one or more."
- Example: <statement list> ::= <statement> [{;<statement>}…]

Translation: Grouping

- Create a new variable A for $\{\alpha\}$.
- One production for A: A -> α.
- Use A in place of $\{\alpha\}$.

Example: Grouping

L -> S [{;S}…]

- Replace by L -> S [A…]   A -> ;S
  - A stands for {;S}.
- Then by L -> SB   B -> A… | ϵ   A -> ;S
  - B stands for [A…] (zero or more A's).
- Finally by L -> SB   B -> C | ϵ
  C -> AC | A   A -> ;S
  - C stands for A… .

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

Rightmost Derivations

- Say $\alpha A w \Rightarrow_{rm} \alpha\beta w$ 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) | ()

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