EECS 2001N : Introduction to the Theory of Computation

Suprakash Datta
Office: LAS 3043

Course page: `http://www.eecs.yorku.ca/course/2001N`
Also on Moodle

Grammars

- Another model of languages
- Study languages recognized by different types of grammars

Regular Grammars and Regular Languages

A regular grammar $G = (V, \Sigma, R, S)$ is defined by

- V: a finite set of variables
- Σ: finite set terminals (with $V \cap \Sigma = \emptyset$)
- R: finite set of substitution rules $V \rightarrow (V \cup \Sigma \cup \Sigma V)$
  - $A \rightarrow \epsilon$, $A \in V$
  - $A \rightarrow a$, $a \in \Sigma, A \in V$
  - $A \rightarrow aB$, $a \in \Sigma, A, B \in V$
- S: start symbol $\in V$

Notation: Rules are combined as follows: $A \rightarrow a$, $A \rightarrow aB$ are written as $A \rightarrow a|aB$

Regular Grammar: Derivation

- A single step derivation $\Rightarrow$ consist of the substitution of a variable by a string according to a substitution rule
- Example: with the rules $A \rightarrow 0B, A \rightarrow 1A$, we can have the derivation $01A \Rightarrow 010B$
- A sequence of several derivations (or none) is indicated by "$\Rightarrow^*$"
- Same example: $0A \Rightarrow^* 11110B$
- define the language generated by G to be $L(G) = \{w | S \Rightarrow^* w, \text{ where } w \in \Sigma^*\}$

Regular Grammar: Example

- $\Sigma = \{0, 1\}$, $V = \{S, T\}$
  - $S \to \epsilon$
  - $S \to 0S$
  - $S \to T$
  - $T \to 1T$
  - $T \to \epsilon$
- Example derivation: $S \Rightarrow^* 0001111$
- This generates the language

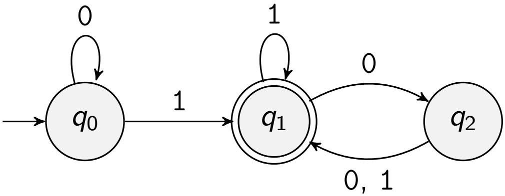
$$L = \{0^m 1^n | m, n \in \mathbb{Z}, m, n \geq 0\}$$

Regular Grammar from a DFA

Given a DFA $M = (Q, \Sigma, \delta, q_0, F)$, we construct a corresponding regular grammar $G_M = (V, \Sigma, R, S)$ with

- $V = Q$
- $S = q_0$
- Rules of G_M:
  - $q_i \to x\delta(q_i, x)$ for all $q_i \in V$ and all $x \in \Sigma$
  - $q_i \to \epsilon$ for all $q_i \in F$

Regular Grammar from a DFA - 2



- $\Sigma = \{0,1\}, V = \{q_0, q_1, q_2\}, S = q_0$
- Rules of G_M:
  - $q_0 \rightarrow 0q_0|1q_1$
  - $q_1 \rightarrow 0q_2|1q_1|\epsilon$
  - $q_2 \rightarrow 0q_1|1q_1$

Two Problems

- Given a grammar G, find the language it generates
- Given a grammar G and a word w, determine if $w \in L(G)$ [**Parsing**]
  Analogous problems:
  - Given an English sentence, determine if its is grammatically correct
  - Given a Java program, determine if it compiles

More Complex Languages

- Grammar view: how can we make the grammar richer?
  More complex rules, e.g. of the form
  - $A \to aAb$, $a, b \in \Sigma$, $A \in V$
  - $A \to aBC$, $a \in \Sigma$, $A, B, C \in V$

- Machine view: how can we augment a FA?
  Augment a FA with a stack
  – We will return to this view later

Do More Complex Grammars Help?

A Grammar for the Nonregular Language

$$L = \{0^n 1^n | n \in \mathbb{Z}, n \geq 0\}$$

- $S \rightarrow 0S1$
- $S \rightarrow \epsilon$

S yields $0^n 1^n$ according to the derivation:
$S \rightarrow 0S1 \rightarrow 00S11 \rightarrow \ldots \rightarrow 0^n S 1^n \rightarrow 0^n 1^n$

Context-free Languages

- Simplest grammar more complex than regular grammars
- Model for natural languages (Noam Chomsky)
- Specification of programming languages: "parsing of a computer program"
- "context free": Rules of the form $S \rightarrow aSbTbb$.
  The rule can be applied regardless of what is before or after S in an expression (**context**)
- "context sensitive": there is context in the left hand side of a rule, e.g. $bC \rightarrow bc$

Context-free Languages

- Human languages use context in word or phrase meanings
- Compilation of programs would be very difficult if programming languages were not context free
- Regular languages are context-free

Context-free Grammar (CFG)

A CFG $G = (V, \Sigma, R, S)$ is defined by

- V: a finite set of variables
- Σ: finite set terminals (with $V \cap \Sigma = \emptyset$)
- R: finite set of substitution rules $V \to (V \cup \Sigma)^*$
- S: start symbol $\in V$

Notation: Rules are combined using '$|$' like before

Examples of CFL's

- $L(G) = \{0^n 1^{2n} | n = 1, 2, \ldots\}$
- $L(G) = \{0^n 1^n | n = 1, 2, \ldots\} \cup \{1^n 0^n | n = 1, 2, \ldots\}$
- $L(G) = \{xx^R | x \text{ is a string over } \{a, b\}\}$
- Properly parenthesized expressions (Sec 3.2.1)
  Solution:
  $G = (V, \Sigma, R, S)$, where $V = \{S\}$, $\Sigma = \{(,)\}$, and
  $R = \{S \to \epsilon, S \to (S), S \to SS\}$

Harder Examples of CFL's

- $L(G) =$
  $\{x | x \text{ is a string over } \{0, 1\} \text{ with an equal number of 1's and 0's}\}$
  Solution:
  $G = (V, \Sigma, R, S)$, where $V = \{S\}$, $\Sigma = \{0, 1\}$, and
  $R = \{S \rightarrow \epsilon, S \rightarrow 0S1S, S \rightarrow 1S0S\}$

- Complement of a regular language (Sec 3.2.3) - only an exercise....what is a simpler solution?

Harder Examples of CFL's - 2

Verifying non-negative number addition (Sec 3.2.4)

$$L = \{a^n b^m c^{n+m} | n \geq 0, m \geq 0, m, n \in \mathbb{Z}\}$$

- Verify L is not regular
- Intuition: every time an 'a' or 'b' are added, a 'c' must be added
- Rules:
    - $S \rightarrow \epsilon | A$
    - $A \rightarrow \epsilon | aAc | B$
    - $B \rightarrow \epsilon | bBc$
- We can eliminate $S \rightarrow \epsilon$, $A \rightarrow \epsilon$