Discrete Mathematics

CS204: Spring, 2008

Jong C. Park
Computer Science Department
KAIST

Today's Topics

Sequential Circuits and Finite-State Machines

Finite-State Automata

Languages and Grammars

Nondeterministic Finite-State Automata

Relationships Between Languages and Automata

AUTOMATA, GRAMMARS, AND LANGUAGES

Sequential Circuits and Finite-State Machines

- Note
  - We assume that the state changes only at time t = 0, 1, …
- Definitions
  - A unit time delay accepts as input a bit x_t at time t and outputs x_{t-1}, the bit received as input at time $t - 1$.
  - The unit time delay is drawn as in Figure 12.1.1.
  - A serial adder accepts as input two binary numbers.
- Example
  - Serial-Adder Circuit

Sequential Circuits and Finite-State Machines

- Definition
  - A finite-state machine M consists of
    - (a) A finite set I of input symbols.
    - (b) A finite set O of output symbols.
    - (c) A finite set S of states.
    - (d) A next-state function f from $S \times I$ into S.
    - (e) An output function g from $S \times I$ into O.
    - (f) An initial state $\sigma \in S$.
  - We write $M = (I, O, S, f, g, \sigma)$.

Sequential Circuits and Finite-State Machines

- Definition
  - Let $M = (I, O, S, f, g, \sigma)$ be a finite-state machine.
  - The transition diagram of M is a digraph G whose vertices are the members of S.
    - An arrow designates the initial state σ.
    - A directed edge (σ_1, σ_2) exists in G if there exists an input i with $f(\sigma_1, i) = \sigma_2$. In this case, if $g(\sigma_1, i) = o$, the edge (σ_1, σ_2) is labeled $i\,/\,o$.

Sequential Circuits and Finite-State Machines

- Definition
  - Let $M = (I, O, S, f, g, \sigma)$ be a finite-state machine. An input string for M is a string over I. The string $y_1 \cdots y_n$ is the output string for M corresponding to the input string $\alpha = x_1 \cdots x_n$ if there exist states $\sigma_0, \ldots, \sigma_n \in S$ with

$$\sigma_0 = \sigma$$

$$\sigma_i = f(\sigma_{i-1}, x_i) \text{ for } i = 1, \ldots, n;$$

$$y_i = g(\sigma_{i-1}, x_i) \text{ for } i = 1, \ldots, n.$$

Sequential Circuits and Finite-State Machines

- Examples
  - A Serial-Adder Finite-State Machine
  - The SR Flip-Flop

Finite-State Automata

- Definition
  - A finite-state automaton
    $$A = (I, O, S, f, g, \sigma)$$
    is a finite-state machine in which the set of output symbols is {0, 1} and where the current state determines the last output.
  - Those states for which the last output was 1 are called accepting states.
- Note
  - The transition diagram of a finite-state automaton is usually drawn with the accepting states in double circles and the output symbols omitted.

Finite-State Automata

- Examples
  - Draw the transition diagram of the finite-state machine A defined by the table.
  - Draw the transition diagram of the finite-state automaton of Figure 12.2.3 as a transition diagram of a finite-state machine.

Finite-State Automata

- Note
  - As an alternative to the earlier definition, we can regard a finite-state automaton A as consisting of
    - (1) A finite set I of input symbols
    - (2) A finite set S of states
    - (3) A next-state function f from $S \times I$ into S
    - (4) A subset A of S of accepting states
    - (5) An initial state $\sigma \in S$.
  - If we use this characterization, we write

$$A = (I, S, f, A, \sigma).$$

Finite-State Automata

- Example
  - Draw the transition diagram of the finite-state automaton

$$A = (I, S, f, A, \sigma),$$

where

$$I = \{a, b\},$$

$$S = \{\sigma_0, \sigma_1, \sigma_2\},$$

$$A = \{\sigma_2\}\},$$

$$\sigma = \sigma_0,$$

with f given by the table.

Finite-State Automata

- Definition
  - Let $A = (I, S, f, A, \sigma)$ be a finite-state automaton.
  - Let $\alpha = x_1 \cdots x_n$ be a string over I.
  - If there exist states $\sigma_0, \cdots, \sigma_n$ satisfying

    (a) $\sigma_0 = \sigma$

    (b) $f(\sigma_{i-1}, x_i) = \sigma_i$ for $i = 1, \cdots, n$

    (c) $\sigma_n \in A$,

    we say that α is accepted by A. The null string is accepted if and only if $\sigma \in A$. We let $Ac(A)$ denote the set of strings accepted by A and we say that A accepts $Ac(A)$.
  - Let $\alpha = x_1 \cdots x_n$ be a string over I. Define states $\sigma_0, \cdots, \sigma_n$ by conditions (a) and (b) above. We call the (directed) path $(\sigma_0, \cdots, \sigma_n)$ the path representing α in A.

Finite-State Automata

- Examples
  - string acceptance
  - Design a finite-state automaton that accepts precisely those strings over $\{a, b\}$ that contain no a's.
  - Design a finite-state automaton that accepts precisely those strings over $\{a, b\}$ that contain an odd number of a's.

Finite-State Automata

Algorithm 12.2.10: Determining whether a string over $\{a, b\}$ is accepted by the finite-state automaton whose transition diagram is given in Figure 12.2.7.

Input: n, the length of the string ($n = 0$ designates the null string); $s_1 s_2 \cdots s_n$, the string

Output: "Accept" if the string is accpted
"Reject" if the string is not accepted

```
fsa(s, n) {
   state = 'E'
   for i = 1 to n {
      if (state == 'E' ∧ s_i == 'a')
         state = 'O'
      if (state == 'O' ∧ s_i == 'a')
         state = 'E'
   }
   if (state == 'O')
      return "Accept"
   else
      return "Reject"
}
```

Finite-State Automata

- Definition
  - The finite-state automata A and A' are equivalent if $Ac(A) = Ac(A')$.
- Example
  - Verify that the two finite-state automata of Figures 12.2.6 and 12.2.8 are equivalent.

Languages and Grammars

- Definition
  - Let A be a finite set. A (formal) language L over A is a subset of A^*, the set of all strings over A.
- Example
  - Let $A = \{a, b\}$. The set L of all strings over A containing an odd number of a's is a language over A. L is precisely the set of strings over A accepted by the finite-state automaton of Figure 12.2.7.

Languages and Grammars

- Definition
  - A phrase-structure grammar (or, simply, grammar) G consists of
    (a) A finite set N of nonterminal symbols
    (b) A finite set T of terminal symbols where $N \cap T = \varnothing$
    (c) A finite subset P of $[(N \cup T)^* - T^*] \times (N \cup T)^*$, called the set of productions
    (d) A starting symbol $\sigma \in N$.
  - We write $G = (N, T, P, \sigma)$.
- Note
  - A production is usually written $A \to B$.

Languages and Grammars

- Example
  - Let

$$N = \{\sigma, S\},$$
$$T = \{a, b\},$$
$$P = \{\sigma \to b\sigma, \sigma \to aS, S \to bS, S \to b\}.$$

  - Then $G = (N, T, P, \sigma)$ is a grammar.

Languages and Grammars

- Definition
  - Let $G = (N, T, P, \sigma)$ be a grammar.
  - If $\alpha \to \beta$ is a production and $x\alpha y \in (N \cup T)^*$, we say that $x\beta y$ is directly derivable from $x\alpha y$ and write $x\alpha y \Rightarrow x\beta y$.
  - If $\alpha_i \in (N \cup T)^*$ for $i = 1, \ldots, n$, and α_{i+1} is directly derivable from α_i for $i = 1, \ldots, n-1$, we say that α_n is derivable from α_1 and write $\alpha_1 \Rightarrow \alpha_n$.
  - We call $\alpha_1 \Rightarrow \alpha_2 \Rightarrow \cdots \Rightarrow \alpha_n$ the derivation of α_n (from α_1).
  - By convention, any element of $(N \cup T)^*$ is derivable from itself.
  - The language generated by G, written $L(G)$, consists of all strings over T derivable from σ.

Languages and Grammars

- Examples
  - Determine $L(G)$ where G is the grammar of the earlier example.
  - A Grammar for Integers
    - Backus normal form (or Backus-Naur form, BNF)
      - the nonterminal symbols typically begin with "<" and end with ">".
      - the production $S \rightarrow T$ is written $S ::= T$.
      - Productions of the form

$$S ::= T_1,$$
$$S ::= T_2,$$
$$\ldots,$$
$$S ::= T_n$$

may be combined as $S ::= T_1 \mid T_2 \mid \cdots \mid T_n$.

Languages and Grammars

- Definition
  - Let G be a grammar and let λ denote the null string.
    (a) If every production is of the form $\alpha A\beta \rightarrow \alpha\delta\beta$, where α, $\beta \in (N \cup T)^*$, $A \in N$, $\delta \in (N \cup T)^* - \{\lambda\}$, we call G a context-sensitive (or type 1) grammar.
    (b) If every production is of the form $A \rightarrow \delta$, where $A \in N$, $\delta \in (N \cup T)^*$, we call G a context-free (or type 2) grammar.
    (c) If every production is of the form $A \rightarrow a$ or $A \rightarrow aB$ or $A \rightarrow \lambda$, where A, $B \in N$, $a \in T$, we call G a regular (or type 3) grammar.

Languages and Grammars

- Definition
  - A language L is context-sensitive (respectively, context-free, regular) if there is a context-sensitive (respectively, context-free, regular) grammar G with $L = L(G)$.

- Definition
  - Grammars G and G are equivalent if $L(G) = L(G)$.

Languages and Grammars

- Definition
  - A context-free interactive Lindenmayer grammar consists of
    - (a) A finite set N of nonterminal symbols
    - (b) A finite set T of terminal symbols where $N \cap T = \varnothing$
    - (c) A finite set P of productions $A \to B$, where $A \in N \cup T$ and $B \in (N \cup T)^*$
    - (d) A starting symbol $\sigma \in N$.
- Note
  - In a context-free interactive Lindenmayer grammar, to derive the string β from the string α, all symbols in α must be replaced simultaneously.

Summary

- Sequential Circuits and Finite-State Machines
- Finite-State Automata
- Languages and Grammars
- Nondeterministic Finite-State Automata
- Relationships Between Languages and Automata