CS 321

Theory of Computation

Part I: Finite Automata and Regular Languages

Instructor: Liang Huang

(some slides courtesy of J. Ullman and H. Potter)
Finite Automata Example

- in Tennis, match = 3-5 sets; set = 6 or more games
- in one game, one person serves throughout
- to win, you must win by at least 2 points
Finite Automata Example

- **states**: encode status info.  **finite**: only knows current state

- state changes in response to **inputs** (s, o in this example)

- **transitions**: rules for state change.  *at state x, on input a, goto state y*

  ![Finite Automaton Diagram]

  *trap state not shown*
Acceptance/Rejection of Inputs

- given an input string, start at the start state and follow transitions.
- input is accepted if you wind up in a final (accepting) state after all inputs have been read; otherwise rejected.
Example: Processing a String
Example: Processing a String
Example: Processing a String

s o s o s o s o s o s o s o s
Example: Processing a String

s o s o s o s o s o s s

Start

Love

Love-15

15-Love

30-Love

40-Love

Server Wins

Opp’n’t Wins

Ad-out

Ad-in

deuce

30-40

30-all

30-15

40-15

40-30

15-30

15-40

Love-30

Love-40
Example: Processing a String

s o s o s o s o s o s s

Start

Love

Love-15

Love-30

Love-40

Opp’nt Wins

Server Wins

40-Love

30-Love

40-15

30-15

40-30

30-40

deuce

Ad-in

Ad-out

s o s o s o s o s o s s
Example: Processing a String
Example: Processing a String

s o s o s o s o s o s o s
Example: Processing a String

s o s o s o s o s o s
Example: Processing a String

s o s o s o s o s o s s

Start 15-Love 30-Love 40-Love Server Wins
Love 15-all 30-15 40-15 Ad-in
Love-15 15-30 30-all 40-30 deuce
Love-30 40-15 30-40 Ad-out
Love-40 Opp’n’t Wins

*
Example: Processing a String

s o s o s o s o s o s
Example: Processing a String

sosoososososs
Example: Processing a String

s o s o s o s o s o s o s o s o s
Example: Processing a String

s o s o s o s o s o s s
Language of an Automaton

- the set of strings accepted by an automaton $A$ is the language of $A$, denoted $L(A)$
- $L(\text{Tennis}) =$ strings of $\{s, o\}$ that determine the winner
"A language is a set of strings."

\[
L_1 = \{ab, bc, ac, dd\}
\]
\[
L_2 = \{\varepsilon, ab, abab, ababab, \ldots\}
\]

The empty string:
\[
\varepsilon
\]

The empty language:
\[
\{\} = \emptyset
\]

The language containing only \(\varepsilon\):
\[
L_3 = \{\varepsilon\}
\]
Construct Finite Automata for ...

- \( L_1 = \{ \text{aa, ab, ac, ..., ba, bb, ..., zz} \} \) (finite)
  - start state, final states

- \( L_2 = \text{all letter sequences} \) (infinite)
  - (cycle)

- \( L_3 = \text{all alphanumeric strings that start with a letter} \)
Construct Finite Automata for ...

- $L_4 = \text{all letter strings with at least a vowel}$

- $L_5 = \text{all letter strings with vowels in order}$

- $L_6 = \text{all 01 strings with even number of 0's and even number of 1's}$
Formal Definitions

Described by a 5-tuple:

\[ M = (Q, \Sigma, \delta, q_0, F) \]

- **Q** = Set of States  
  - Finite Number of states
- **\Sigma** = Alphabet, a Finite Set of Symbols
- **\delta** = The TRANSITION FUNCTION  
  \[ \delta: Q \times \Sigma \rightarrow Q \]
- **q_0** = The STARTING STATE  
  \[ q_0 \in Q \]  
  (or “INITIAL” STATE)
- **F** = The set of ACCEPT states  
  (or “FINAL” STATES)  
  \[ F \subseteq Q \]
Language, String, Machine

The language that $M$ accepts is $A$.

"$M$ recognizes $A$."

"$M$ accepts $A$."

M accepts strings

M rejects strings

The empty string $\epsilon$ (epsilon, $\epsilon$)

The empty language $\emptyset = \{\epsilon\}$

Note: $\emptyset \neq \emptyset$

$\epsilon \neq \emptyset$

If a machine accepts NO strings then it recognizes the EMPTY LANGUAGE

does $M$ accept any string?

does $M$ accept empty string?

does $M$ recognizes the empty language?
What’s the language of ...
Construct Finite Automaton for ... 

- any string that does not contain 001 in it
- try simpler problem: a string that does contain 001
Construct Finite Automaton for ...

- any string that does not contain 0011 in it
- try simpler problem: a string that does contain 0011
Compliment Language

Notation

\[ \text{L}(M_1) = \text{The language that } M_1 \text{ recognizes.} \]
\[ = \text{The set of strings over } \{0, 1\}^* \text{ that contain 0011 as a substring.} \]
\[ \text{L}(M_2) = \text{The set of strings over } \{0, 1\}^* \text{ that do not contain 0011.} \]

Complimenting a language

They are sets, after all.

\[ \text{L}(M_1) = \overline{\text{L}(M_2)} \]

The “Universe”

All possible strings made with symbols from the alphabet.

\[ \Sigma = \{0, 1\} \quad \text{Universe} = \{0, 1\}^* \]

Set complement is always relative to some universe; (implicitly).
Dead States Formally Defined

What does this F.S.M. recognize?

Recognizes 10
Also 01, 001, 0001, ... 0^+1

L = \{ \omega \mid \omega \text{ is either } 10 \text{ or a string of at least one } 0 \text{ followed by a single } 1 \}

What about

111 \text{ } ? \text{ What happens?}
1010

\delta \text{ is a function}
FORMALLY, must be defined
\delta(c, 1) = ?

If some transitions are missing, add a dead state.
(Often, prefer not to show dead state.)

DEAD STATES
Formal Definition of Computation

Let $M = (Q, \Sigma, \delta, q_0, F)$

Let $w = w_1w_2 \ldots w_n$ be a string
where $w_i \in \Sigma$

$M$ accepts $w$ if there is a sequence of states
$r_0, r_1, r_2, \ldots, r_n$ in $Q$
such that
$r_0 = q_0$
$\delta(r_i, w_{i+1}) = r_{i+1}$ for $0 \leq i \leq n$
$r_n \in F$

We say...

$M$ "recognizes" Language $A$
if $A = \{ w \mid M$ accepts $w \}$