CSC236 *Intro. to* the Theory of Computation

Lecture 9: Finite State Automata

Amir H. Chinaei, Fall 2016

Office Hours: W 2-4 BA4222

ahchinaei@cs.toronto.edu

http://www.cs.toronto.edu/~ahchinaei/

Course page:

http://www.cdf.toronto.edu/~csc236h/fall/index.html

Section page:

http://www.cdf.toronto.edu/~csc236h/fall/amir_lectures.html

review

- so far
  - different flavour of proofs and their application in cs
- in particular, recently

  we saw tools useful toward
    - **proof**: if a program is **semantically** correct

  let's reword it:
    - **recognize**: if a program is **semantically** correct
- next: finite state machines/automata
  - tools useful to **recognize** if a program is **syntactically** correct
  - **and …**

Examples 83, 84

- identifiers
  - e.g., a *letter* followed by a *digit*
  - more practical ones

Example 85

- Python-like float

Example 86

- strings with an odd number of a's (and any number of b's)

Finite State Automaton definition

- is a 5-tuple $M = (Q, \Sigma, \delta, q_0, F)$
  - Q is the set of states, which is finite & non-empty
  - Σ is the alphabet, which is finite & non-empty
  - $\delta : Q \times \Sigma \to Q$ is the transition function
  - $q_0 \in Q$ is the start state
  - $F \subseteq Q$ is the set of accept states
- Then, $L(M)$ is a language
  - that machine M accepts,
  - i.e., set of all strings that machine M accepts

Example 86 revisited

- devise a machine that only accepts strings with an odd number of a's. $\Sigma = \{a, b\}$

Examples 85, 84, 83 revisited

- **85.** devise a *machine* that accepts strings representing a float number a. $\Sigma = \{0..9, +, -, .\}$

- **84.** devise a *machine* that accepts identifiers $\Sigma = \{0..9, a..z, \_\}$

- **83.** devise a *machine* that accepts simple identifiers (length 2, first character a letter). $\Sigma = \{0..9, a..z\}$

notes

notes