CSC236 *Intro. to* the Theory of Computation

Lecture 11: fsa and regular expressions

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 of FSA

- last week
  - FSA and regular languages.
- this week:
  - FSA and regular expressions

notation

- Σ: finite non-empty set of symbols, e.g., $\{a, b\}$
- Σ^k: concatenation of symbols of Σ, k times, ≥ 0
  - e.g., Σ^0: $\{\varepsilon\}$, Σ^1: $\{a,b\}$, Σ^2: $\{aa, bb, ab, ba\}, \dots$
- $\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \dots$
- string $\in \Sigma^*$, e.g., $abbaba$
- |string|: length of string, e.g., $|abbaba| = 6$
- s^R: reversal of string s, e.g., $abbaba^R = ababba$
- $s.t$: concatenation of strings s and t
- language $\subseteq \Sigma^*$, e.g., $L_{86} = \{\omega \in \Sigma^* | \omega$ has odd # of $a\}$

language operations

- $L_1 \cup L_2 = \{\omega \in \Sigma^* |\ \omega \in L_1$ or $\omega \in L_2\}$
  - e.g., $L_{86} \cup L_{88} = \{\omega \in \Sigma^* |\ \omega$ has odd number of a's or ω does not end with $a\}$
- $L_1 . L_2 = \{\omega = s.t \in \Sigma^* |\ s \in L_1, t \in L_2\}$
  - e.g., $L_{86} . L_{88} = \{\omega \in \Sigma^* |\ \omega$ has odd number of a's followed by a string that does not end with $a\}$
- $L_1{}^* = \{\varepsilon\} \cup \{\omega \in \Sigma^* |\ \exists s_1, s_2, \dots, s_n \in L_1$ such that $\omega = s_1 . s_2 . \dots . s_n$ for some $n\}$
  - e.g., $L_{86}{}^* = \{\omega \in \Sigma^* |\ \omega$ concatenation of any number strings that have odd number of $a\}$

language operations

- $L_1 \cap L_2 = \{\omega \in \Sigma^* |\ \omega \in L_1$ and $\omega \in L_2\}$
  - e.g., $L_{86} \cap L_{88} = \{\omega \in \Sigma^* |\ \omega$ has odd number of a's and does not end with $a\}$
- $L_1 - L_2 = \{\omega \in \Sigma^* |\ \omega \in L_1$ and $\omega \notin L_2\}$
  - e.g., $L_{86} - L_{88} = \{\omega \in \Sigma^* |\ \omega$ has odd number of a's and ends with $a\}$
- $\overline{L_1} = \{\omega \in \Sigma^* |\ \omega \notin L_1\}$
  - e.g., $\overline{L_{88}} = \{\omega \in \Sigma^* |\ \omega$ ends with $a\}$
- $L_1{}^R = \{\omega \in \Sigma^* |\ \omega^R \in L_1\}$
  - e.g., $L_{88}{}^R = \{\omega \in \Sigma^* |\ \omega$ do not start with $a\}$

regex

- so far, we have explicitly seen
  - RL can be shown by **FSA**
  - RL can be shown by **set description**
- another way to define RL is by:
  - **Regular Expressions**
    - aka **regex**, **RE**

RL: formal definition (revisit)

- let Σ be the alphabet:
  - the empty set, $\emptyset$, is a RL
  - the set $\{\varepsilon\}$ is a RL
  - for each a $\in \Sigma$, the set $\{a\}$ is a RL
  - If L_1 and L_2 are regular languages, then
    - union: $L_1 \cup L_2$ is RL
    - concatenation: $L_1 . L_2$ is a RL
    - Kleene star: $L_1{}^*$ is a RL
- no other RL over Σ exists.

$L(r)$ is defined by structural induction

- **basis step:**
  - if r is a $regex$ defined by basis step of the definition,
    - $L(\emptyset)$ is a RL
    - $L(\varepsilon)$ is a RL
    - $L(a)$, for any a $\in \Sigma$, is a RL
- **inductive step**:
  - if r_1, r_2 are $regex$'s defined by ind step of the definition,
    - $L(r_1 + r_2) = L_1(r_1) \cup L_2(r_2)$ is a RL
    - $L(r_1 . r_2) = L_1(r_1) . L_2(r_2)$ is a RL
    - $L(r_1{}^*) = L_1(r_1)^*$ is a RL

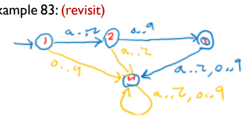
$regex$ examples (96)

- assume $\Sigma = \{0,1\}$
  - $\emptyset$, ε, 0, 1, 0+1, 00, 01, 10, 11, 000, 111,
  - $L((0+1)^*)$
  - $L(0^*)$
  - $L((10)^*)$
  - $L(10^*)$

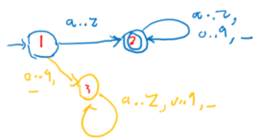
$regex$ examples

notes

Example 83: (revisit)

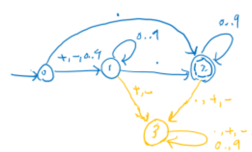


Example 84: (revisit)



notes

Example 85: (revisit)



Example 97

- Prove $L_{86} = L(r_{86})$ where
  - L_{86}={$\omega \in \{0,1\}^*$| ω starts and ends with different bits}
  - r_{86}=$0.(0+1)^*.1+1.(0+1)^*.0$

Example 97

regex identities

- communitativity of union:
- associativity of union:
- associativity of concatenation:
- left distributivity:
- right distributivity:
- identity for union:
- identity for concatenation:
- annihilator for concatenation:
- idempotence of Kleene star:

$NFA, DFA, regex$

$NFA, DFA, regex$

notes