CSC236 fall 2012
regular expressions

Danny Heap
heap@cs.toronto.edu
BA4270 (behind elevators)
http://www.cdf.toronto.edu/~heap/236/F12/
416-978-5899

Using Introduction to the Theory of Computation,
Chapter 7
Outline

regular expressions

product, non-deterministic FSAs

regular languages

notes
another way to define languages
In addition to the language accepted by DFSA $L(M)$ and set description $L = \{\ldots\}$.

Definition: The regular expressions (regexps or REs) over alphabet $\Sigma$ is the smallest set such that:

1. $\emptyset$, $\epsilon$, and $a$, for every $a \in \Sigma$ are REs over $\Sigma$
2. if $T$ and $S$ are REs over $\Sigma$, then so are:
   - $T + S$ (union) — lowest precedence operator
   - $TS$ (concatenation) — middle precedence operator
   - $T*$ (star) — highest precedence
The $L(R)$, the language denoted (or described) by $R$ is defined by structural induction:

**Basis:** If $R$ is a regular expression by the basis of the definition of regular expressions, then define $L(R)$:

- $L(\emptyset) = \emptyset$ (the empty language)
- $L(\varepsilon) = \{\varepsilon\}$ (the language consisting of just the empty string)
- $L(a) = \{a\}$ (the language consisting of the one-symbol string)

**Induction step:** If $R$ is a regular expression by the induction step of the definition, then define $L(R)$:

- $L(S + T) = L(S) \cup L(T)$
- $L(ST) = L(S)L(T)$
- $L(T^*) = L(T)^*$
regexp examples

- \( L(0 + 1) = \{0, 1\} \)

- \( L((0 + 1)^*) \) All binary strings over \( \{0, 1\} \)

- \( L((01)^*) = \{\varepsilon, 01, 0101, 010101, \ldots\} \)

- \( L(0^*1^*) \) 0 or more 0s followed by 0 or more 1s.

- \( L(0^* + 1^*) \) 0 or more 0s or 0 or more 1s.

- \( L((0 + 1)(0 + 1)^*) \) Non-empty binary strings over \( \{0, 1\} \).
example

$L = \{ x \in \{0, 1\}^* \mid x \text{ begins and ends with a different bit} \}$
RE identities

some of these follow from set properties...
others require some proof (see 7.2.5 example)

- communitativity of union: $R + S \equiv S + R$
- associativity of union: $(R + S) + T \equiv R + (S + T)$
- associativity of concatenation: $(RS)T \equiv R(ST)$
- left distributivity: $R(S + T) \equiv RS + RT$
- right distributivity: $(S + T)R \equiv SR + TR$
- identity for union: $R + \emptyset \equiv R$
- identity for concatenation: $R\epsilon \equiv R \equiv \epsilon R$
- annihilator for concatenation: $\emptyset R \equiv \emptyset \equiv R\emptyset$
- idempotence of Kleene star: $(R^*)^* \equiv R^*$

$L(R^*) = \bigcup_{k \in \mathbb{N}} L(R)^k$
product construction

$L$ is the language of binary strings over $\{0, 1\}^*$ with two 1s in a row and an even number of 0s.

Idea: $\delta((q_i, q_j), a) = (\delta(q_i, a), \delta(q_j, a))$

\[ L = L(M) \]
non-deterministic FSA (NFSA) example

FSA that accepts $L((010 + 01)^*)$

$\delta^*(q_0, 0101) \rightarrow \{q_0, q_3\}$

0101 $\in L$
they’re equivalent:

\[ L = L(M) \text{ for some DFSA } M \Leftrightarrow L = L(M') \text{ for some NFSA } M' \Leftrightarrow L = R(R) \text{ for some regular expression } R \]

\[ M_1, M_2 - \text{FSA} \]

and \( L(M_1) = L(M_2) \)

Want \( M_3 \) s.t. \( L(M_3) = L(M_1) \cup L(M_2) \)