both test - posted
q1 - q4
11 10 9 8 7 6 5 4

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 Σ is the **smallest** set such that:

$\emptyset$ 1. $\{\}$, $\overset{\varepsilon}{\epsilon}$, and a, for every $a \in \Sigma$ are REs over Σ
2. if T and S are REs over Σ, then so are:
   - $T + S$ (union) — lowest precedence operator
   - TS (concatenation) — middle precedence operator
   - T^* (star) — highest precedence

regular expression to language:

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

- $\left(L(0+1)\right)^*$ = $\{0, 1\}$
- $L((0+1)^*)$ All binary strings over $\{0, 1\}$
- $L((01)^*) = \{\varepsilon, 01, 0101, 010101, \ldots\}$ $L((0^*1^*)^*)$
- $L(0^*1^*)$ 0 or more 0s followed by 0 or more 1s.

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

  $\{\varepsilon, 1, 0, 11, 00, \ldots\}$
- $L((0+1)(0+1)^*)$ Non-empty binary strings over $\{0, 1\}$.

example

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

$R = \left(0(0+1)^*1 + 1(1+0)^*0\right)$ OR $\cancel{\left(0^*1^* + 1^*0^*\right)}$

0101

$\varepsilon \in$

$\cancel{1(1+0)^*0}$

<u>prove</u> $L = L(R)$

$L \subseteq L(R) \wedge L(R) \subseteq L$

<u>Proof</u> (not really): to show $\underline{L(R) \subseteq L}$

assume s is an arbitrary element of $L(R)$.

Then, WLOG (switch 0s and 1s otherwise)

assume $s \in L(0(0+1)^*1)$. Then, s has

the form tuv, where $t \in L(0)$, $u \in L((0+1)^*)$,

$v \in L(1)$

without loss of generality.

RE identities

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

$L(R) \cup L(S) \equiv L(S) + L(R)$

- 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: $R\varepsilon \equiv R \equiv \varepsilon R$
- annihilator for concatenation: $\emptyset R \equiv \emptyset \equiv R\emptyset$
- idempotence of Kleene star: $(R^*)^* \equiv R^*$

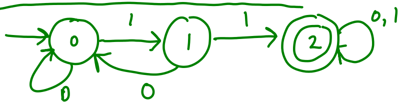
$((R^*)^*)^* \equiv (R^*)^* \equiv R^*$

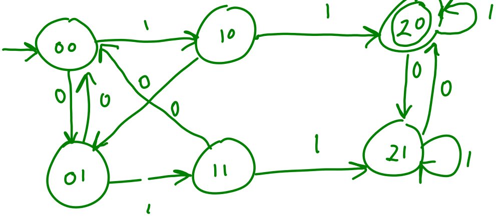
Computer Science
UNIVERSITY OF TORONTO

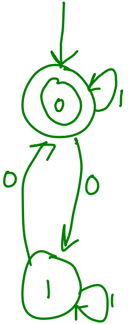
product construction

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

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



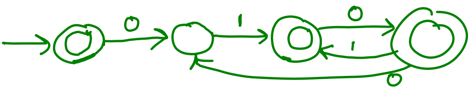




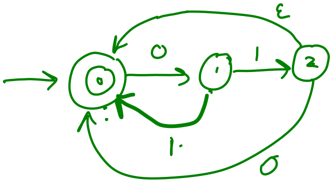
0 1 0 11

non-deterministic FSA (NFSA) example

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



construct a corresponding DFSA



010

Blue added after lecture.

$F = \{q_0\}$

$\delta^*(q_0, 010) = \{q_0, q_1\}$

non-empty intersection with F → accepted.

transition function is between sets of states and of string drives our machine to a set that includes an accepting state, said string is accepted.

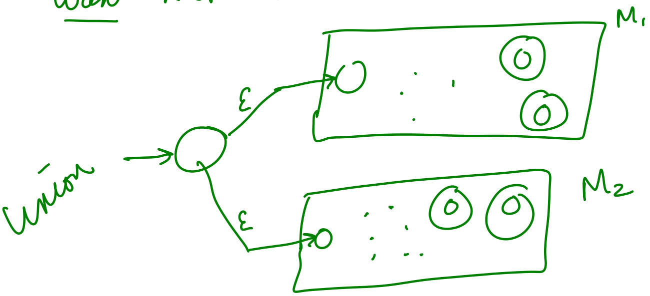
subset contruction

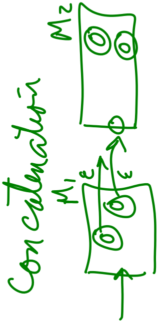
$\hat{M} = \{Q, \delta, s, F, \Sigma\}$; $M = \{Q = \mathcal{P}(Q), \delta', \{s\}, F, \Sigma\}$

they're equivalent:

$L = L(M)$ for some DFSA M $\Leftrightarrow$ $L = L(M')$ for some NFSA M' $\Leftrightarrow$
$L = L(R)$ for some regular expression R

$$L(R) = L(M_1)$$
$$L(S) = L(M_2)$$
$$\underline{\text{want}}\quad L(R+S) = L(M_1) \cup L(M_2)$$





notes