Office hours tomorrow: 4–6, BA 3201
Next Thurs noon - 2, BA 3201

CSC236 fall 2016

regular languages, regular expressions

Danny Heap
heap@cs.toronto.edu / BA4270 (behind elevators)
http://www.cdf.toronto.edu/~csc236h/fall/
416-978-5899

Using Introduction to the Theory of Computation, Chapter 7

Outline

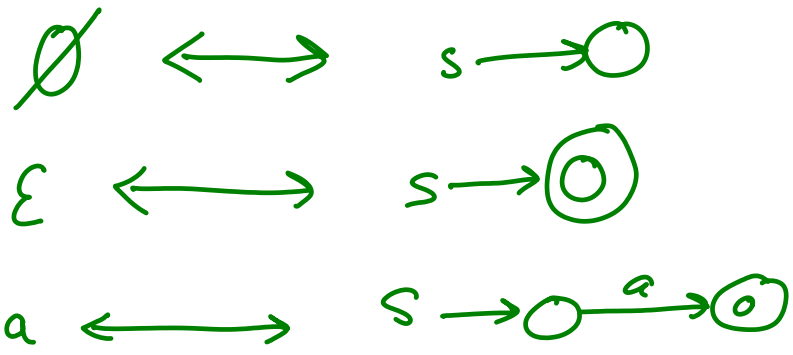
regular expressions, regular languages

notes

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
step 1: convert $L(R)$ to $L(M')$

start with $\emptyset$, ε, $a \in \Sigma$



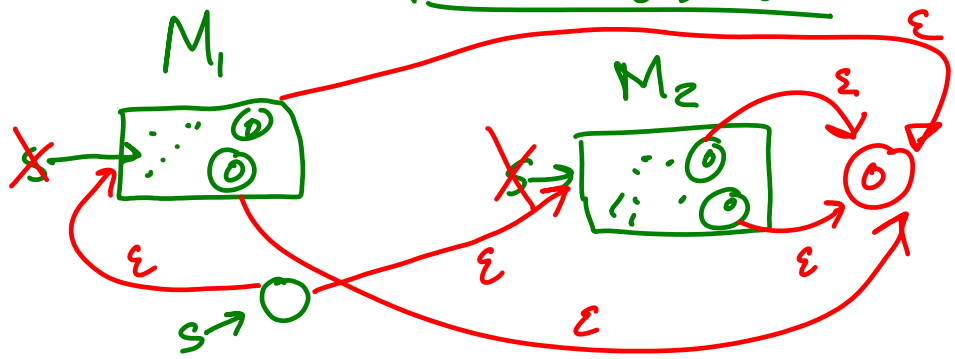
equivalence...

step 1.5: convert $L(R)$ to $L(M')$:
union, concatenation, stars

$L(r_1 + r_2) \leftrightarrow L(r_1) \cup L(r_2) \leftrightarrow L(M_1) \cup L(M_2)$

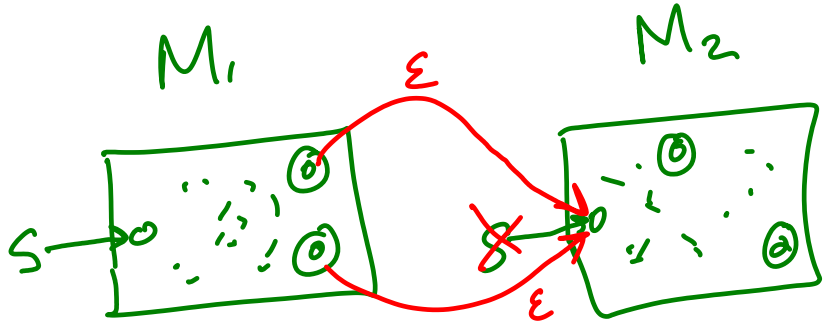
$\leftrightarrow L(M)$ where M is union of M_1, M_2 machines

product construct



$L(r_1 r_2) \leftrightarrow L(r_1)L(r_2) \leftrightarrow L(M_1)L(M_2)$

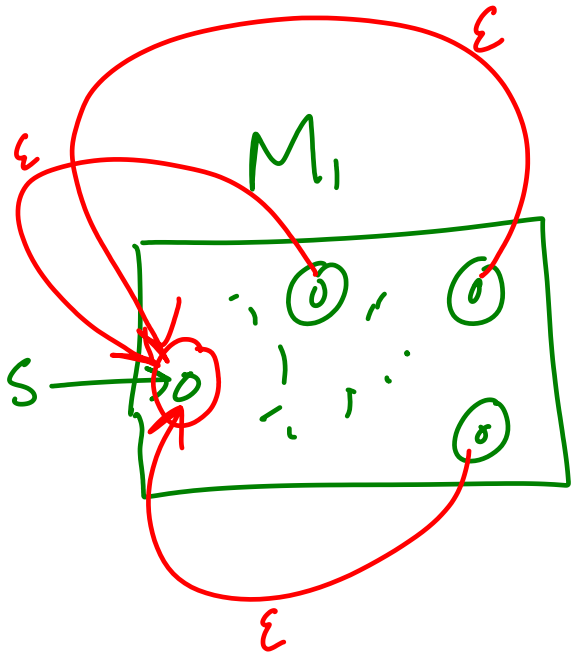
M



equivalence...

step 1.5: convert $L(R)$ to $L(M')$:
union, concatenation, stars

$L(N)^* \longleftrightarrow L(M)$, M



$(aa)^*$

$aaaa$

equivalence...

step 2: convert $L(M')$ to $L(M)$

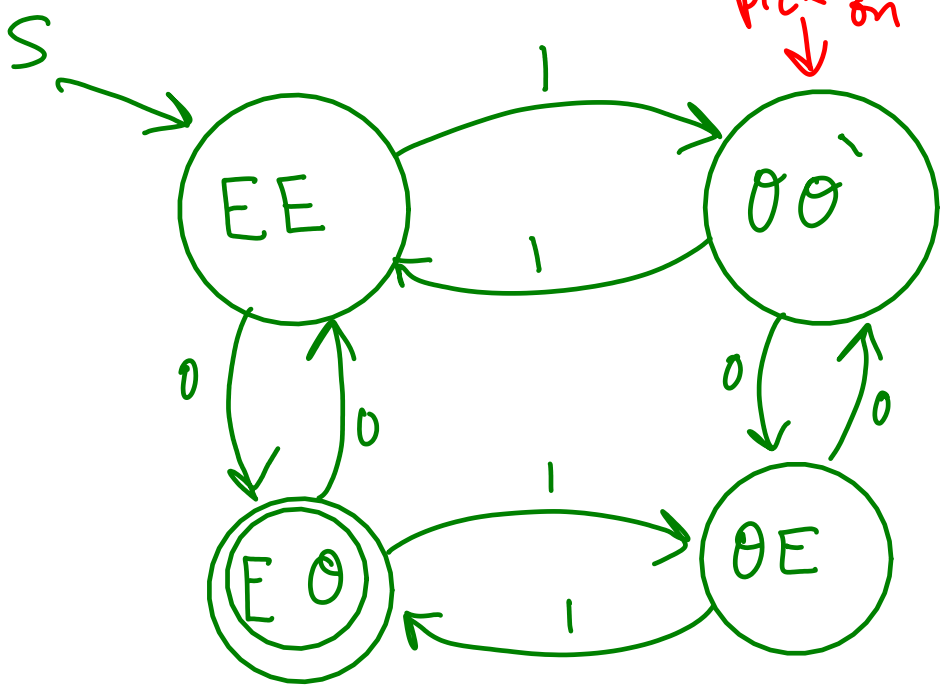
use subset construction (also see example in week 11)

there could be $2^{|Q|}$ subsets to consider..

they're equivalent:

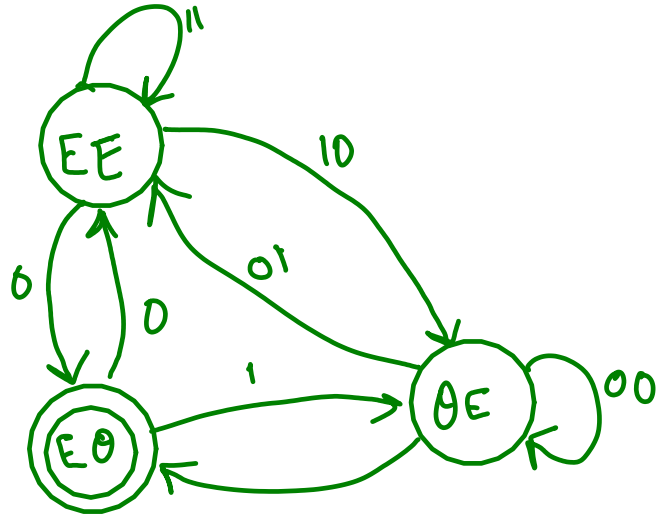
$L = L(M)$ for some DFSA $M \Leftrightarrow L = L(M')$ for some NFSA $M' \Leftrightarrow$
$L = L(R)$ for some regular expression R

step 3: convert $L(M)$ to $L(R)$, eliminate states



Even #'s, odd length

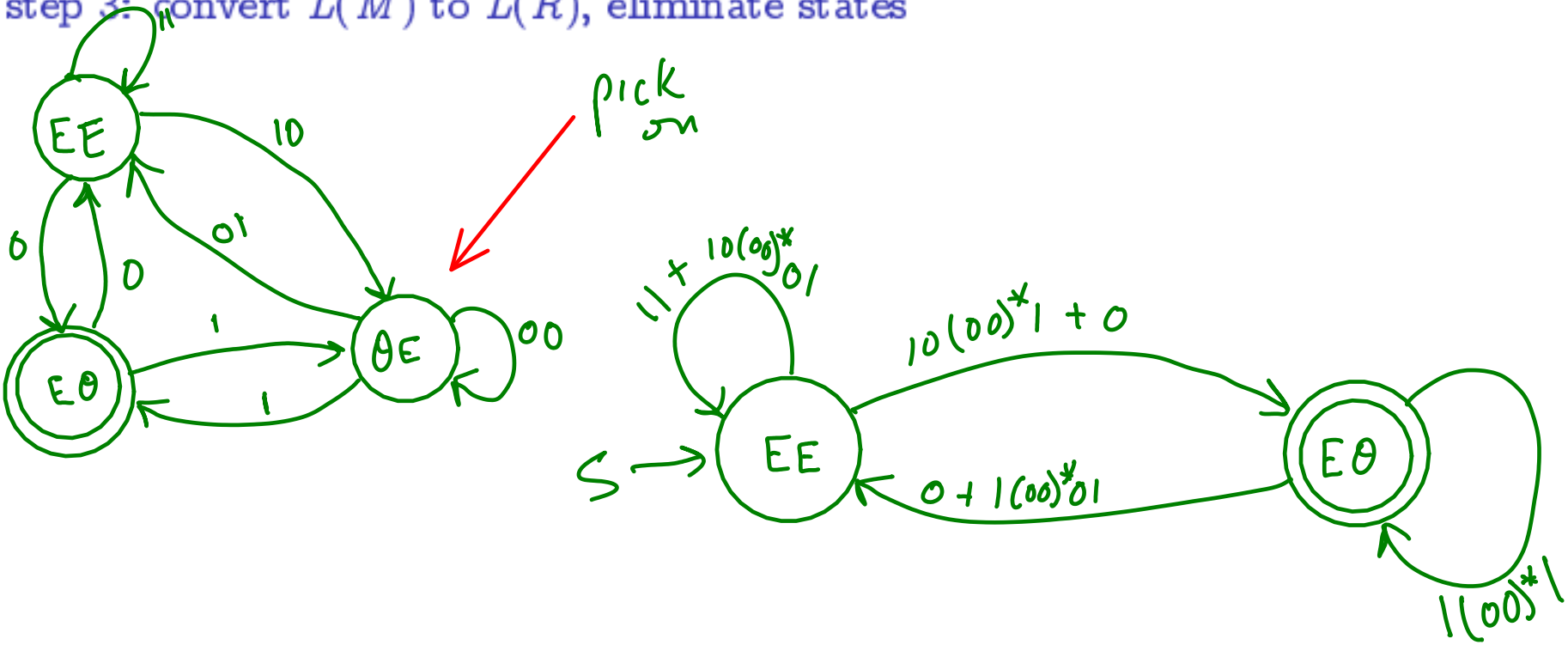
label transitions with REs



they're equivalent:

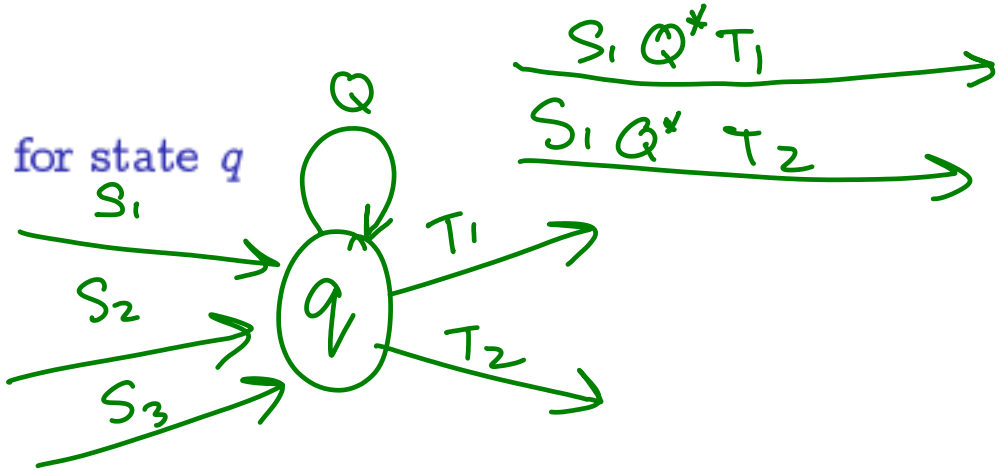
$L = L(M)$ for some DFSA $M \Leftrightarrow L = L(M')$ for some NFSA $M' \Leftrightarrow$
$L = L(R)$ for some regular expression R

step 3: convert $L(M)$ to $L(R)$, eliminate states



equivalence...

state elimination recipe for state q



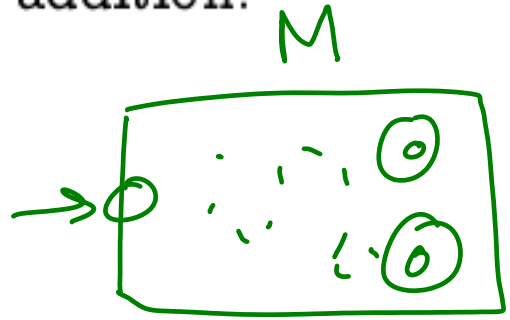
1. $s_1 \ldots s_m$ are states with transitions to q, with labels $S_1 \ldots S_m$
2. $t_1 \ldots t_n$ are states with transitions from q, with labels $T_1 \ldots T_n$
3. Q is any self-loop on q
4. Eliminate q, and add (union) transition label $S_i Q^* T_j$ from s_i to t_j.

regular languages closure

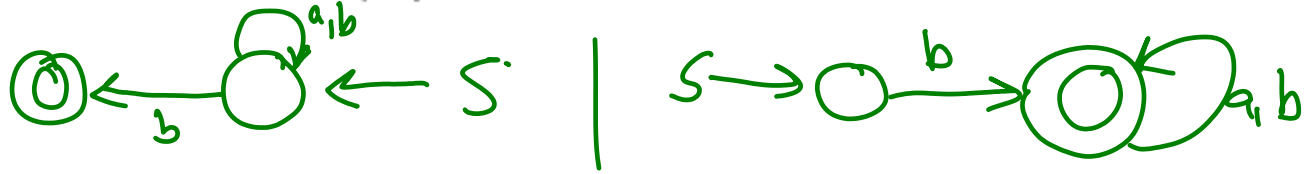
Regular languages are those that can be denoted by a regular expression or accept by an FSA. In addition:

- L regular $\Rightarrow$ $\overline{L}$ regular

swap accepting/non accepting



- L regular $\Rightarrow$ $Rev(L)$ regular



pumping lemma (see course notes, page 234)

number of states in FSA for L

If $L \subseteq \Sigma^*$ is a regular language, then there is some $n_L \in \mathbb{N}$ (n_L depends on L) such that if $x \in L$ and $|x| \geq n_L$ then:

- $\exists u, v, w \in \Sigma^*, x = uvw$
- $|v| > 0$
- $|uv| \leq n_L$
- $uv^k w \in L \forall k \in \mathbb{N}$

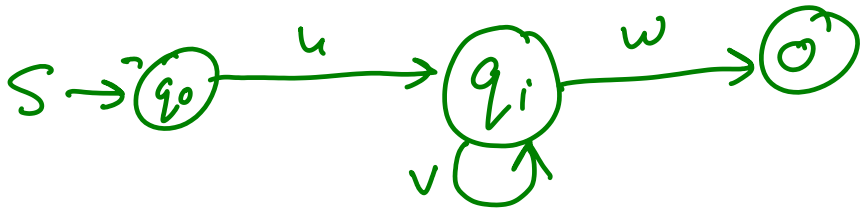
$|x| \geq n$, # states in $L(M)$

$q_0 \to q_1 \cdots q_n \to \cdots q_{|x|}$

by pigeon-hole some state is visited twice! say q_i

$q_0 \to q_1 \cdots q_i \to q_i \to q_n \to q_{|x|}$

u v w



consequences of regularity

How about $L = \{1^n 0^n | n \in \mathbb{N}\}$

Proof-ish (Contradiction)

Suppose L is regular. Then by PL $\exists\, k \in \mathbb{N}^+$ s.t. if $x \in L$ and $|x| \geq k$, then $x = uvw$. Choose $x = 1^k 0^k$. then $x = uvw$ and $|uv| \leq k$

$\Rightarrow uv = 1^p$, $p+q \leq k$, $q > 0$. $v = 1^q + |v| > 0$

$x = uvw \in L \Longrightarrow uv^2w \in L$

But uv^2w has $|v|$ too many 1s, ie $uv^2w = 1^{k+q}0^k$

$\longrightarrow \longleftarrow$ $k+q \neq k$.

Let L be language of strings with balanced parentheses. Since $(^k)^k \in L$

How about $L = \{w \in \Sigma^* \mid |w| = p, p \text{ is prime}\}$

- choose $|\chi| = p > n_L$
- $\chi = uvw$ (as before).

$uv^{p+1}w \in L$

$|uv^{p+1}w| = \underbrace{|uvw|}_{p} + \underbrace{|v^p|}_{|v| \cdot p} = p(1+|v|)$

not prime

notes