spoiler alert: we've back, forth, back again between FSAs and regexes. Punchline: they accept/denote the same set of languages. Second punchline: some languages are neither accepted/denoted

CSC236 fall 2018
machines, expressions: equivalence

Danny Heap
heap@cs.toronto.edu / BA4270 (behind elevators)
http://www.teach.cs.toronto.edu/~heap/236/F18/ 416-978-5899

Using Introduction to the Theory of Computation, Chapter 7
Outline

regular expressions, regular languages

notes
non-deterministic FSA (NFSA) example
FSA that accepts $L((010 + 01)^*)$

from start, diagram transitions to *sets of states* that could be reached. Any set of states that contains at least one accepting state becomes an accepting state. The new machine is deterministic --- DFSA.
NFSAs are real

...you can always convert them to DFSA

Use subset construction, notes page 219 if $\Sigma = \{0, 1\}$, the construction is, roughly

- start at the start state combined with any states reachable from start with $\varepsilon$-transitions
- if there are any 1-transitions from this new combined start state, combine them into a new state
- there are any 0-transitions from this new combined start, combine them into a new state
- repeat for every state reachable from the start...
NFSA that accepts $L((0 + 10)(0 + 10)^*)$

construct the corresponding DFSA...
NFSA that accepts $\text{Rev}(L((0 + 10)(0 + 10)^*))$

construct the corresponding DFSA...

1. swap start and accepting state (epsilon if multiple starts)
2. reverse all transitions

I re-named the states to D (was A|B) E (was A) and F (was C) to reduce confusing notation during the subset construction...
FSAs, regexes are equivalent:

\[ L = L(M) \] for some DFSA \( M \) \iff \[ L = L(M') \] for some NFSA \( M' \) \iff 
\[ L = L(R) \] for some regular expression \( R \)

step 1.0: convert \( L(R) \) to \( L(M') \)

for concreteness, let's say \( \Sigma = \{0, 1\} \)

start with \( \emptyset, \varepsilon, a \in \Sigma \)

\( L(\emptyset) = L(M) \), where \( M = \)

\( L(\varepsilon) = L(M) \), where \( M = \)

\( L(0) = L(M) \), where \( M = \)
equivalence...

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

Assume $r_1, r_2 \in \text{RE}$, and that $L(r_1) = L(M_1), L(r_2) = L(M_2)$, where $M_1, M_2$ are FSA

$L(r_1 + r_2) = L(r_1) \cup L(r_2) = L(M)$ either uses the product construction OR

$L(r_1r_2) = L(r_1)L(r_2) = L(M)$

$L(r_1^*) = L(r_1)^* = L(M)$, where $M$
equivalence...

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

use subset construction

there could be $2^{|Q|}$ subsets to consider, but often many are unreachable and may be ignored...
FSAs, regexes are 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

accepts:
""
0000000
11
1111
1001
110

eliminate state 1

eliminate state 2

our regex:
$(0+11+10(1+00)^*01)^*$
equivalence...

state elimination recipe for state \( q \)

label transitions with *regexes* rather than symbols

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

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

$L(M) = L(1)$, so $M$:

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

We did an example earlier...
pumping lemma (see course notes, page 234)

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:

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

idea: if machine \( M(L) \) has \( |Q| = n_L \), \( x \in L \land |x| \geq n_L \), denote \( q_i = \delta^*(q_0, x[\cdot:i]) \), so \( x \) “visits” \( q_0, q_1, \ldots, q_L \) with the first \( n_L + 1 \) prefixes of \( x \ldots \) so there is at least one state that \( x \) “visits” twice (pigeonhole principle)

if it accepts \( uvw \), it also accepts \( uw, uvvw, uvvvvvwwv, \) etc.
consequences of regularity

How about $L = \{1^n0^n \mid n \in \mathbb{N}\}$
How about $L = \{w \in \Sigma^* \mid |w| = p \land p \text{ is prime}\}$
notes