T2: back before 8 pm, average B
A2: some sections done released soon.
A3: up tomorrow night.
Tutorials changed polarity due to fall break ... so suddenly evening section leads off on formal languages...

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

formal languages

FSAs

notes
some definitions

alphabet: finite, non-empty set of symbols, e.g. \{a, b\} or \{0, 1, \text{--}1\}.Conventionally denoted \(\Sigma\).

\[\Sigma_0,1^* = \{\varepsilon, 0,1,00,01,10,11, \ldots\}\]

string: finite (including empty) sequence of symbols over an alphabet: abba is a string over \{a, b\}.
Convention: \(\varepsilon\) is the empty string, never an allowed symbol, \(\Sigma^*\) is set of all strings over \(\Sigma\).

language: Subset of \(\Sigma^*\) for some alphabet \(\Sigma\). Possibly empty, possibly infinite subset. E.g. \{}, \(\emptyset\), \{aa, aaa, aaaa, \ldots\}.

N.B.: \(\{} \neq \{\varepsilon\}.\)
Many problems can be reduced to languages: logical formulas, identifiers for compilation, natural language processing. Key question is recognition:

Given language $L$ and string $s$, is $s \in L$?

Languages may be described either by descriptive generators (for example, regular expressions) or procedurally (e.g. finite state automata)
more notation

string length: denoted $|s|$, is the number of symbols in $s$, e.g. $|bba| = 3$.

$s = t$: if and only if $|s| = |t|$, and $s_i = t_i$ for $1 \leq i \leq |s|$.

$s^R$: reversal of $s$ is obtained by reversing symbols of $s$, e.g. $1011^R = 1101$.

$st$ or $s \circ t$: concatenation of $s$ and $t$ — all characters of $s$ followed by all those of $t$, e.g. $bba \circ bb = bbabb$.

$s^k$: denotes $s$ concatenated with itself $k$ times. E.g., $ab^3 = ababab$, $101^0 = \varepsilon$.

$\Sigma^n$: all strings of length $n$ over $\Sigma$, $\Sigma^*$ denotes all strings over $\Sigma$. 
language operations

\( \overline{L} \): Complement of \( L \), i.e. \( \Sigma^* - L \). If \( L \) is language of strings over \( \{0, 1\} \) that start with 0, then \( \overline{L} \) is the language of strings that begin with 1 plus the empty string.

\( L \cup L' \): union

\( L \cap L' \): intersection

\( L - L' \): difference

Rev(\( L \)): \( = \{s^R : s \in L\} \)

concatenation: \( LL' \) or \( L \cdot L' = \{rt \mid r \in L, t \in L'\} \). Special cases

\( L\{\varepsilon\} = L = \{\varepsilon\}L \), and \( L\{\} = \{\} = \{\}L \).
more language operations

\[ L^k \text{ is concatenation of } L \text{ } k \text{ times. Special case, } L^0 = \{\varepsilon\}, \text{ including } L = \{\}\].

\[ \Sigma^* = \Sigma a \Sigma^* \]

Kleene star: \( L^* = L^0 \cup L^1 \cup L^2 \cup \ldots. \)
states needed to classify a string

what state is a stingy vending machine in based on coins?
accepts only nickles (a), dimes (b), and quarters (c),
no change given, and everything costs 30 cents
useful toy (you’ll need JRE)

\[ \Sigma = \{ n, d, q \} \]

<table>
<thead>
<tr>
<th>( \delta )</th>
<th>0</th>
<th>5</th>
<th>10</th>
<th>15</th>
<th>20</th>
<th>25</th>
<th>( \geq 30 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>n</td>
<td>5</td>
<td>10</td>
<td>15</td>
<td>20</td>
<td>25</td>
<td>( \geq 30 )</td>
<td>( \geq 30 )</td>
</tr>
<tr>
<td>d</td>
<td>10</td>
<td>15</td>
<td>20</td>
<td>25</td>
<td>( \geq 30 )</td>
<td>( \geq 30 )</td>
<td>( \geq 30 )</td>
</tr>
<tr>
<td>q</td>
<td>25</td>
<td>( \geq 30 )</td>
<td>( \geq 30 )</td>
<td>( \geq 30 )</td>
<td>( \geq 30 )</td>
<td>( \geq 30 )</td>
<td>( \geq 30 )</td>
</tr>
</tbody>
</table>
build an automaton with formalities...

quintuple: \((Q, \Sigma, q_0, F, \delta)\)

- \(Q\) is set of states, \(\Sigma\) is finite, non-empty alphabet, \(q_0\) is start state
- \(F\) is set of accepting states, and \(\delta : Q \times \Sigma \mapsto Q\) is transition function

We can extend \(\delta : Q \times \Sigma \mapsto Q\) to a transition function that tells us what state a string \(s\) takes the automaton to:

\[
\delta^*(q_0, s) = \begin{cases} 
q & \text{if } s = \varepsilon \\
\delta(\delta^*(q_0, s'), a) & \text{if } s' \in \Sigma^*, a \in \Sigma, s = s' a 
\end{cases}
\]

String \(s\) is accepted if and only if \(\delta^*(q_0, s) \in F\), it is rejected otherwise.
example — an odd machine
device a machine that accepts strings over \( \{a, b\} \) with an odd number of \( a \)s

\[
\Sigma = \{0, 1, \overline{3}\}
\]

\( \ell \) has even # of \( a \)s + odd length

Formal proof requires inductive proof of invariant:

\[
\delta^*(E, s) = \begin{cases} 
E & \text{if } s \text{ has even number of } a \text{s} \\
O & \text{if } s \text{ has odd number of } a \text{s}
\end{cases}
\]

for all \( k \in \mathbb{N} \), \( s \in \Sigma^k \), then

\[\delta^*(E, s)\] is true, work to conclude that

\( M \) accepts \( s \) if \( s \) has odd # of \( a \)s.
float machine

\[ L_1 = \{0, \ldots, 9\} \]
\[ L_2 = \{+, -\}, L_3 = \{.\} \]
\[ L_F = \{s \in L_2^j L_1^m L_3^k L_1^n \mid j, k \leq 1, m, n \geq 1\} \]

Devise a machine that accepts \( L_F \)