A2 - posted by section (Lila, Colin, ...).

CSC236 fall 2012
automata and languages
FSA

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, -1\}. Conventionally denoted \(\Sigma\).

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 \(\Sigma\).

\[\Sigma = \{0, 1\}, \quad \Sigma^* = \{\varepsilon, 0, 1, 01, 10, 010, 001\}\]

language: Subset of \(\Sigma^*\) for some alphabet \(\Sigma\). Possibly empty, possibly infinite subset. E.g. \(\{\}\), \(\{aa, B's\}\).

N.B.: \(\{\} \neq \{\varepsilon\}\).
Many problems can be reduced to languages: logical formulas, identifiers for compilation, natural language recognition. 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)
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. 
\(1011^R = 1101\). 
\(\varepsilon^R = \varepsilon\) 
\(|1^R| = |1|\)

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

\(s^k\): 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 \(\Sigma\).

\(\Sigma^0 = \{ \varepsilon \}\)
language operations

\[ \Sigma \]

\[ \Sigma^* \]

\[ L = \{0, 1, 01, 11, 10, 00\} \]

\[ \overline{L} \]: 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 \[ \{a, b, ba\} = L' \]

\[ L \cap L' \]: intersection

\[ L - L' \]: difference

\[ L \setminus L' \]
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)

<table>
<thead>
<tr>
<th>δ</th>
<th>0</th>
<th>5</th>
<th>10</th>
<th>15</th>
<th>20</th>
<th>25</th>
<th>≥ 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>≥ 30</td>
<td>≥ 30</td>
</tr>
<tr>
<td>d</td>
<td>10</td>
<td>15</td>
<td>20</td>
<td>25</td>
<td>≥ 30</td>
<td>≥ 30</td>
<td>≥ 30</td>
</tr>
<tr>
<td>q</td>
<td>25</td>
<td>≥ 30</td>
<td>≥ 30</td>
<td>≥ 30</td>
<td>≥ 30</td>
<td>≥ 30</td>
<td>≥ 30</td>
</tr>
</tbody>
</table>

Diagram:

```
\text{h} \quad \text{d} \quad \text{n} \quad \text{n}
```
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 \( \delta : Q \times \Sigma \rightarrow Q \) is transition function

\[ \{ 0 \$, 5 \$, 10 \$, 15 \$, 20 \$, 25 \$, \geq 30 \$ \} \]

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

\[
\delta^*(5, \text{nd } n) = \delta(\delta^*(5, \text{nd}), n) \\
\delta(\delta(\delta^*(5, \varepsilon), n), d), n) \\
\delta(\delta(\delta^*(5, \varepsilon), n), d), n)
\]

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

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

Formal proof requires inductive proof of invariant:

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