starting formal languages

CSC236 fall 2018
automata and languages

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

FSAs (finite state automata)

notes
**turnstile finite-state machine**

**what are the rules for turnstiles?**

**FSA aka FSM**

alphabet $\Sigma = \{p, t, b\}$
- $p$: push
- $t$: token
- $b$: bicycle

states, $Q = \{U, L, D\}$
- $U$: unlocked
- $L$: locked
- $D$: dead

$\Sigma^*$ all strings over $\Sigma$

pre-Presto TTC turnstile

\[ \text{tttpt is accepted} \quad \text{tp is rejected} \]
float machine

which strings are floats in Python?

2.1, 1.0, 15973.286551, also no numerals before . or no numerals after ., but not both missing, exactly one period,
Also -2.1 or +15973.28555

should reject: 2..1
++2.1
states needed to classify a string
what state is a stingy vending machine in, based on coins?
accepts only nickles, dimes, and quarters,
no change given, and everything costs 30 cents...
here’s a useful toy

<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>

accept state: >= 30
start state: 0
transitions: see table
\( \sigma = \{n, d, q\} \)
nnddd yes
nnd no
integer multiples of 3

\( \Sigma = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\} \)

accept state: M
build an automaton with formalities...

quintuple: \((Q, \Sigma, q_0, F, \delta)\)  
drawing, \(\delta\) is an arrow, labelled, here it's a function  
\(Q\) 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:

\[
\epsilon = "\"
\]

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

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 as

Formal proof requires inductive proof of invariant:

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

Prove \( \forall s \in \Sigma^*, P(s) \).

basis \( \delta^*(E, \varepsilon) = \begin{cases} 
  E & \varepsilon \text{ has even } \# \text{ of } a\text{s} \\
  O & \varepsilon \text{ has odd } \# \text{ of } a\text{s}
\end{cases} \)
example — an odd machine

device a machine that accepts strings over \( \{a, b\} \) with an odd number of \( as \)

Formal proof requires inductive proof of invariant:

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

Let \( s \in \Sigma^* \), assume \( P(s) \). Must show that \( P(sa) \) and

\( P(sb) \) follow

Case sa:

\[
\delta^*(E, sa) = \delta(\delta^*(E, s), a) = \begin{cases} 
\delta(E) & \Rightarrow s \text{ has even } \#a_s \\
\delta(O, a) & \Rightarrow s \text{ has odd } \#a_s
\end{cases}
\]

\[
\Rightarrow \quad \begin{cases} 
O & \Rightarrow \quad s_{a} \text{ has odd } \#a_s \\
E & \Rightarrow \quad s_{a} \text{ has even } \#a_s
\end{cases}
\]

Case sb: exercise
more odd/even: intersection

$L$ is the language of binary strings
with an odd number of $a$s, and at least one $b$
Devise a machine for $L$

Intersection procedure

For union, just add accepting states which?
more odd/even: union

$L$ is the language of binary strings with an odd number of $a$s, or at least one $b$.

Devise a machine that accepts $L$. 

union
more odd/even

$L$ is the language of binary strings with an odd number of $a$s, but even length

Devise a machine for $L$