Wednesday office hour: cancelled, due to me attending Faculty Council.
Sara R will host hours Wednesday 2--4...

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?

language: set of strings
FSM aka FSA: machines that model computation
on languages: which strings belong

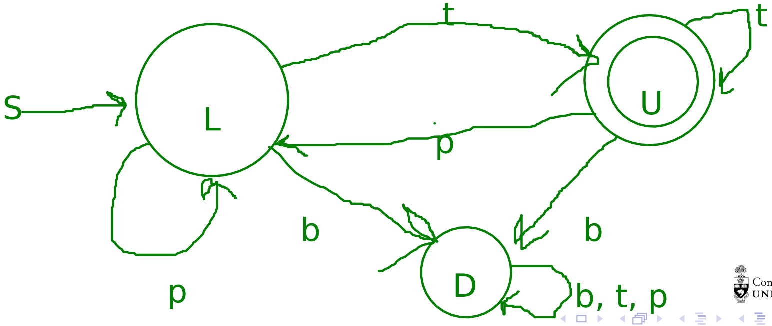
chomskian hierarchy of grammars (languages)

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

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

tttttttttb not accepted!

pt accepted
tp not accepted
tbbp not accepted



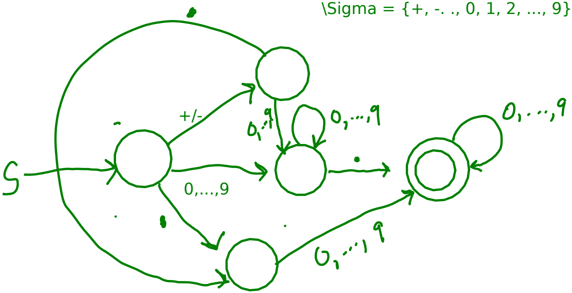
float machine

7 is not a float 7..3 not a float

which strings are floats in Python?

7.0 is a float, what about .0 is a float what about 7.
7.356.8 is not float
+7.356 or -7.357 are floats

\Sigma = {+, -. ., 0, 1, 2, ..., 9}



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

\Sigma = {n, d, q}
Q = {0, 5, 10, 15, 20, 25, >= 30}

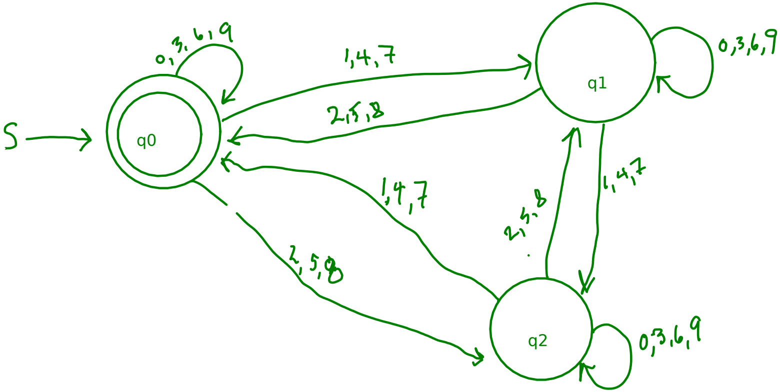
δ	0	5	10	15	20	25	≥ 30
n	5	10	15	20	25	≥ 30	≥ 30
d	10	15	20	25	≥ 30	≥ 30	≥ 30
q	25	≥ 30	≥ 30	≥ 30	≥ 30	≥ 30	≥ 30

= \delta

integer multiples of 3

"" – ε

alphabet \Sigma = {0, 1, 2, ..., 9}, states Q ={q0, q1, q2}, S = F = {q0}



build an automaton with formalities...

quintuple: $(Q, \Sigma, q_0, F, \delta)$ e.g. \Sigma = {0, 1} and Q = {q0, q1, q2}

Q is set of states, Σ 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:

called the extended transition function

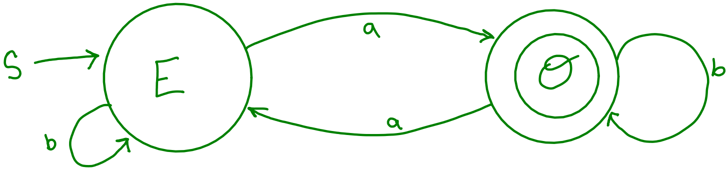
we use variable \epsilon to stand for ""

$$\delta^* : Q \times \Sigma^* \mapsto Q \qquad \delta^*(q, s) = \begin{cases} q & \text{if } s = \varepsilon = \text{""} \\ \delta(\delta^*(q, s'), a) & \text{if } s' \in \Sigma^*, \\ & \quad a \in \Sigma, s = s'a \end{cases}$$

String s is accepted if and only if $\delta^*(q_0, s) \in F$, it is rejected otherwise.

example — an odd machine

devise a machine that accepts strings over $\{a, b\}$ with an odd number of as



Formal proof requires inductive proof of invariant:

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

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

by structural induction

define Σ^* as smallest set such that

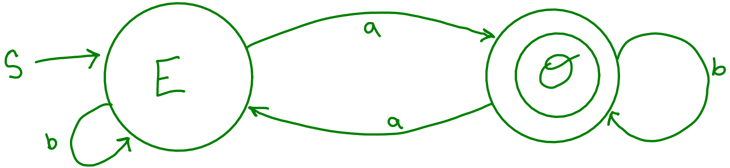
① $\varepsilon \in \Sigma^*$

② $s \in \Sigma^* \Rightarrow sa \in \Sigma^* \wedge sb \in \Sigma^*$

base case: $\delta^*(E, \varepsilon) = \begin{cases} E & \text{if } \varepsilon \text{ has even \# } a\text{s} \checkmark \\ O & \text{if } \varepsilon \text{ has odd \# } a\text{s [vacuously true]} \checkmark \end{cases}$

example — an odd machine

devise a machine that accepts strings over $\{a, b\}$ with an odd number of as



Formal proof requires inductive proof of invariant:

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

inductive step: Let s \in \Sigma^* and assume P(s). I must show that P(sa) and P(sb) follow.

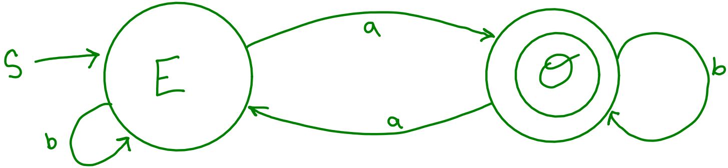
case sa: $\delta^*(E, sa) = \delta(\delta^*(E,s), a) = \begin{cases} \delta(E,a) & \text{if } s \text{ has even \# as} \\ \delta(O,a) & \text{if } s \text{ has odd \# as} \end{cases}$

by I.H.

$= \begin{cases} O & \text{if } sa \text{ has odd \# as} \\ E & \text{if } sa \text{ has even \# as} \end{cases}$

example — an odd machine

devise a machine that accepts strings over $\{a, b\}$ with an odd number of as



Formal proof requires inductive proof of invariant:

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

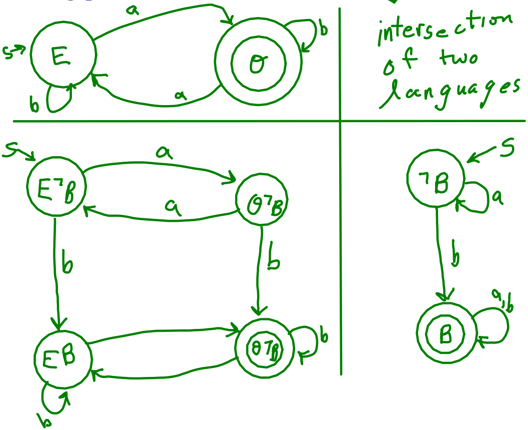
case sb: left as exercise...

Notice that, if the proof succeeds, we have proved that if s has odd number of as, then \delta^*(E, s) takes this machine to O. We also need the converse, and we do this by taking the contrapositive of the other invariant: \neg(E) implies \neg(s has even number of as), which evaluates to \delta^*(E, S) takes this machine to O if s has an odd number of as.

Notice that we need invariants about all states to do this sort of thing... probably including dead states.

more odd/even: intersection

L is the language of binary strings
with an odd number of as, and at least one b
Devise a machine for L
using product construction



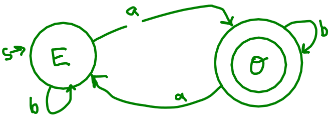
each state in product machine corresponds to a pair of states in factor machines...

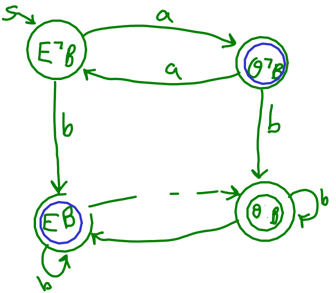
more odd/even: union

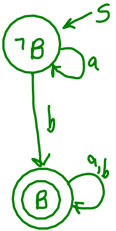
L is the language of binary strings
with an odd number of as, or at least one b
Devise a machine that accepts L
using product construction

union
~~intersection~~
of two languages

add accepting states







more odd/even

L is the language of binary strings
with an odd number of as, but even length
Devise a machine for L
using product construction

$\equiv$ and

exercise
to reader...

notes