MA/CSSE 474

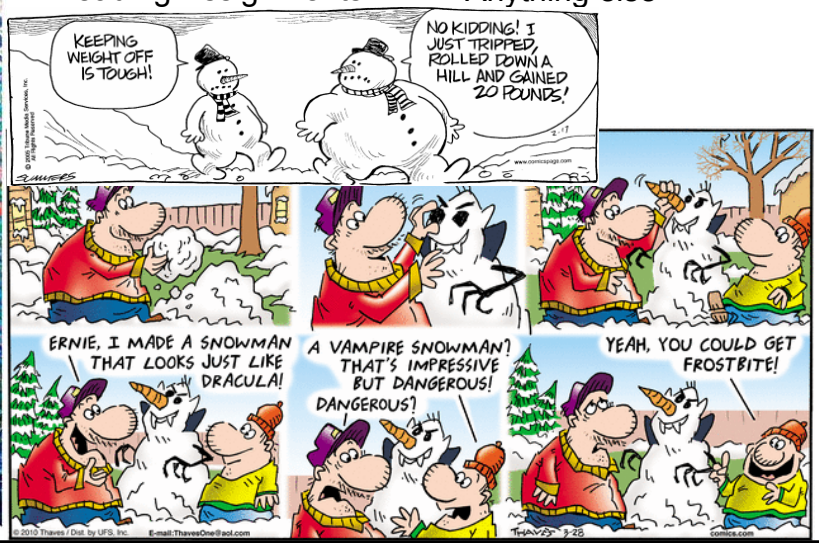
Theory of Computation

Nondeterminism

NFSMs

Your Questions?

- Previous class days' material
- Reading Assignments
- HW1 solutions
- HW3 or HW4 problems
- Anything else

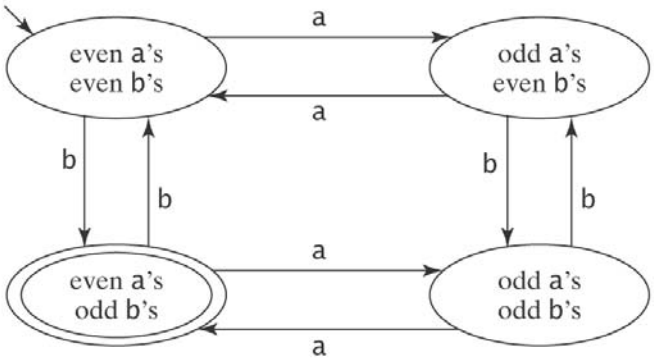


MORE DFSM EXAMPLES

Examples: Programming FSMs

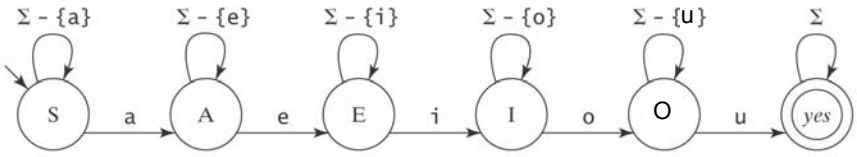
Cluster strings that share a “future”.

$L = \{w \in \{a, b\}^* : w$ contains an even number of a’s and an odd number of b’s$\}$



Vowels in Alphabetical Order

L = {$w \in$ {a - z}* : all five vowels, a, e, i, o, and u, occur in w in alphabetical order}.

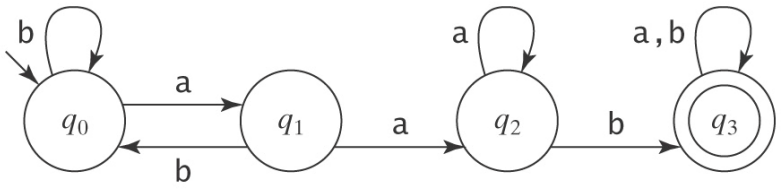


THIS EXAMPLE MAY BE facetious!

Negate the condition, then …

L = {$w \in$ {a, b}* : w does not contain the substring aab}.

Start with a machine for the complement of L:



How should we change it?

The Missing Letter Language

Let Σ = {a, b, c, d}.

Let $L_{Missing}$ =
{$w \in \Sigma^*$: there is a symbol $a_i \in \Sigma$ that does not appear in w}.

Try to make a DFSM for $L_{Missing}$:

Expressed in first-order logic,
$L_{Missing}$ = $\{w \in \Sigma^* : \exists a \in \Sigma\ (\forall x,y \in \Sigma^* (w \neq xay))\}$

How many states are needed?

NONDETERMINISM

Nondeterminism

- A **nondeterministic** machine M,
  when it is in a given state,
  looking at a given symbol
  (and with a given symbol on top of the stack if it is a PDA),
  may have a choice of several possible moves that it can make.
- If there is a move that leads toward eventual acceptance, M makes that move.

Necessary Nondeterminism?

- As you saw in the homework, a PDA is a FSM plus a stack.
- Given a string in {a, b}*, is it in
  PalEven = $\{ww^R : w \in \{a,b\}^*\}$?
- PDA
- **Choice:** Continue pushing, or start popping?
- This language can be accepted by a nondeterministic PDA but not by any deterministic one.

Nondeterministic value-added?

- Ability to recognize additional languages?
  - FSM: no
  - PDA : yes
  - TM: no

We will prove these later

- Sometimes easier to design a nondeterministic machine for a particular language?
  - Yes for all three machine types

A Way to Think About Nondeterministic Computation

1. *choose* (action 1;
   action 2;
   …
   action *n*)

First case: Each action will return True, return False, or run forever.

If any of the actions returns TRUE, *choose* returns TRUE.
If all of the actions return FALSE, *choose* returns FALSE.
If none of the actions return TRUE, and some do not halt, *choose* does not halt.

2. *choose*(*x* from *S*: *P*(*x*))

Second case: S may be finite, or infinite with a generator (enumerator).

If P returns TRUE on some x, so does *choose*
If it can be determined that P(x) is FALSE for all x in P, *choose* returns FALSE.
Otherwise, *choose* fails to halt.

NONDETERMINISTIC FSM

Definition of an NDFSM

$M = (K, \Sigma, \Delta, s, A)$, where:

K is a finite set of states

Σ is an alphabet

$s \in K$ is the initial state

$A \subseteq K$ is the set of accepting states, and

Δ is the **transition relation**. It is a finite subset of

$$(K \times (\Sigma \cup \{\varepsilon\})) \times K$$

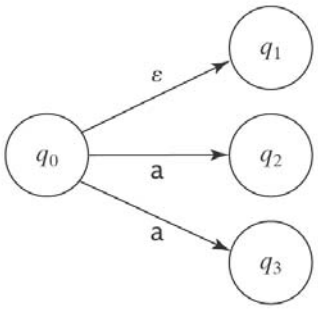
Another way to present it: $\Delta : (K \times (\Sigma \cup \{\varepsilon\})) \rightarrow 2^K$

Acceptance by an NDFSM

An NDFSM M **accepts** a string *w* iff there exists ***some*** *path* along which *w* can take *M* to some element of *A*.

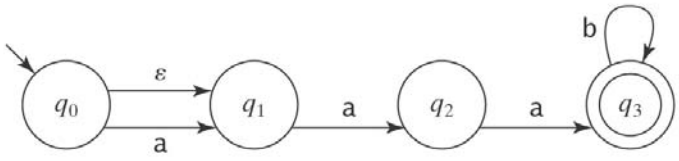
The language accepted by *M*, denoted *L*(*M*), is the set of all strings accepted by *M*.

Sources of Nondeterminism



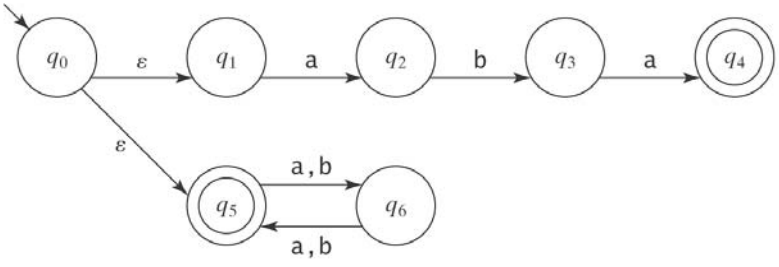
Optional Substrings

$L = \{w \in \{a, b\}^* : w$ is made up of an optional a followed by aa followed by zero or more b's$\}$.



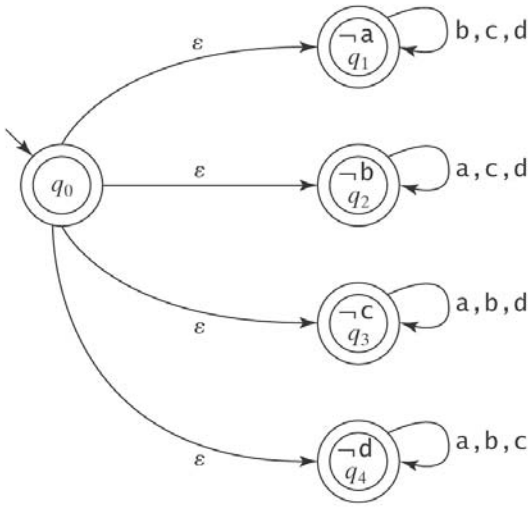
Multiple Sublanguages

$L = \{w \in \{a, b\}^* : w =$ aba or $|w|$ is even$\}$.



The Missing Letter Language

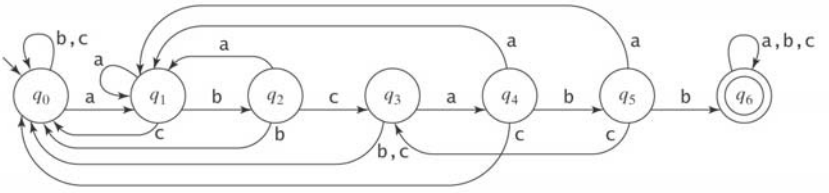
Let Σ = {a, b, c, d}. Let $L_{Missing}$ = {w : there is a symbol $a_i \in \Sigma$ that does not appear in w}



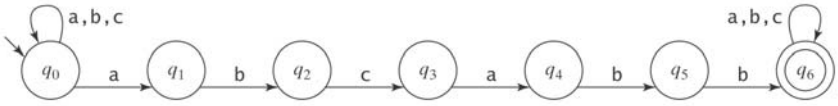
Pattern Matching

L = {$w \in$ {a, b, c}* : $\exists x, y \in$ {a, b, c}* ($w = x$ abcabb y)}.

A DFSM:



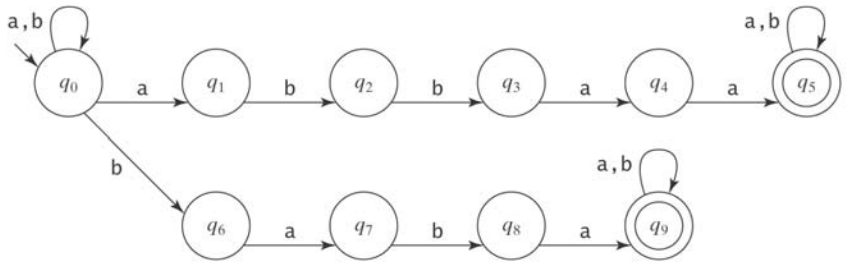
An NDFSM:



Pattern Matching: Multiple Keywords

$L = \{w \in \{a, b\}^* : \exists x, y \in \{a, b\}^*$

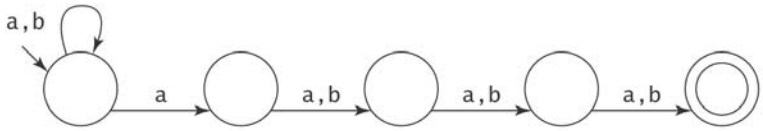
$((w = x\ abbaa\ y) \vee (w = x\ baba\ y))\}$.



Checking from the End

$L = \{w \in \{a, b\}^*$:

the fourth character from the end is a}



Another Pattern Matching Example

L = {$w \in$ {0, 1}* : w is the binary encoding of a positive integer that is divisible by 16 or is odd}

Another NDFSM

L_1= {$w \in$ {a, b}*: aa occurs in w}
L_2= {$x \in$ {a, b}*: bb occurs in x}
L_3= $L_1 \cup L_2$

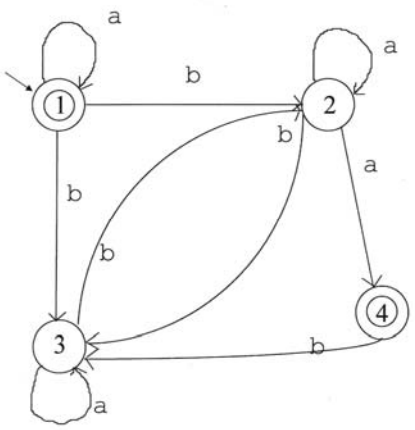
M_1 =

This is a good example for practice later

M_2=

M_3=

Analyzing Nondeterministic FSMs



Does this FSM accept:

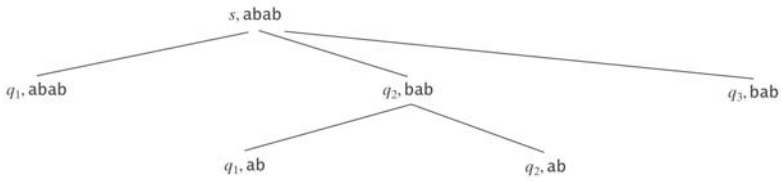
baaba

Remember: we just have to find one accepting path.

Simulating Nondeterministic FSMs

Two approaches:

• Explore a search tree:



• Follow all paths in parallel

Dealing with ε Transitions

The epsilon closure of a state:

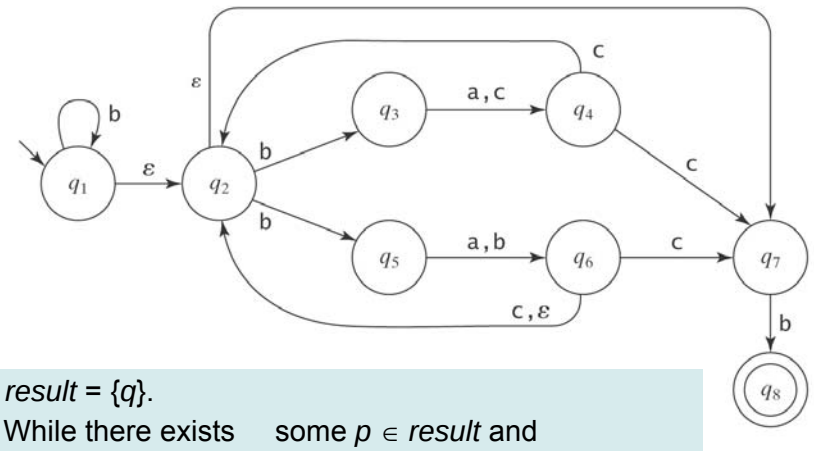
$eps(q) = \{p \in K : (q, w) \vdash^*_M (p, w)\}$.

eps(*q*) is the closure of {*q*} under the relation
{(*p*, *r*) : there is a transition (*p*, ε, *r*) ∈ Δ}.

Algorithm for computing eps(q):

result = {*q*}
while there exists some *p* ∈ *result* and
some *r* ∉ *result* and
some transition (*p*, ε, *r*) ∈ Δ do:
Insert *r* into *result*
return *result*

Calculate eps(q) for each state q



result = {*q*}.
While there exists some *p* ∈ *result* and
some *r* ∉ *result* and
some transition (*p*, ε, *r*) ∈ Δ do:
result = *result* ∪ { *r* }
Return *result*.

Simulating a NDFSM

ndfsmsimulate(*M*: NDFSM, *w*: string) =

1. *current-state* = *eps*(*s*).
2. While any input symbols in *w* remain to be read do:
   1. *c* = get-next-symbol(*w*).
   2. *next-state* = $\varnothing$.
   3. For each state *q* in *current-state* do:
      For each state *p* such that (*q, c, p*) $\in \Delta$ do:
      next-state = *next-state* $\cup$ *eps*(*p*).
   4. *current-state* = *next-state*.
3. If *current-state* contains any states in *A*, accept. Else reject.

Do it for the previous 8-state machine, with input `bbabc`