CS 3313
Foundations of Computing:

NFA and Regular Expessions Review

Extending the NFA model: NFA's With λ-Transitions

- We allow state-to-state transitions on empty string input λ (also denoted ϵ) .
- These transitions are done spontaneously, without looking at the input string.
- A convenience at times, but still only regular languages are accepted.
  - Allowing λ-transitions can make it easier to define and build the automaton

- Analogous to program going to several next states before reading the next input

Any advantage to NFA model with empty string input ?

- w is a string in L1 or L2   (property A or property B)
  - Construct M1 for L1 and M2 for L2, then on λ –transition (no input) from start go to both M1 and M2

- w is a string x.y where x is in L1 and y is in L2; property A followed by property B
  - Construct M1 for L1 and M2 for L2, start in M1 and if it goes to final state then start M2.

- What are we doing here…..simplification of the language/problem

NFA Exercise 2: work in groups

- Provide an NFA M (with λ moves) that accepts the language L over alphabet {0,1,2} where $L = \{ w \mid$ (a) $w=x$ and x has two consecutive 0's or (b) $w=y$ and y has substring 101 and ends with two 2's $\}$

  Ex: 0120012 is in L   0102101222 is in L

  02010220 is not in L

Property (a): build NFA M1 that recognizes substring 00

Property (b): build NFA M2 that recognizes two properties in sequence – substring 101 and then ends with two 2's.

To design NFA M, start M and then go and start both M1 and M2.

Questions on NFAs ?

Languages Associated with Regular Expressions

- A regular expression (RE) **r** denotes a language **L(r)**
- Basis: Assuming that r_1 and r_2 are regular expressions:
  1. The regular expression $\varnothing$ denotes the empty set
  2. The regular expression λ denotes the set $\{ \lambda \}$
  3. For any a in the alphabet, the regular expression **a** denotes the set $\{ a \}$
- Inductive step: if r_1 and r_2 are regular expressions, denoting languages $L(r_1)$ and $L(r_2)$ respectively, then
  1. $r_1 + r_2$ is a RE denoting the language $L(r_1) \cup L(r_2)$
  2. $r_1 \cdot r_2$ is a RE denoting the language $L(r_1). L(r_2)$
  3. (r_1) is a RE denoting the language $L(r_1)$
  4. $r_1{}^*$ is a RE denoting the language $(L(r_1))^*$

Deriving Regular Expressions

- ”map” property in the language to a Reg.Expr. Pattern
- Break down the properties into union, concatenation, star
- Start with smallest reg expression (simplest property)

- Ex: all strings in alphabet {a,b} = (a+b)*
- Two consecutive a’s = aa
- Ends with a pattern aba: (a+b)* aba
- ….

Regular Expressions - Examples

1. L_1= { all strings over alphabet {a,b,c} that contain no more than three a's }
2. L_2 = { all binary strings ending in 01 }

Regular Expressions – Exercise ?

L_3 = { all binary strings that do not end in 01 }

- Hint: you can have strings of length 0 or length 1 – what are they ?
- If string has length two or more, then what substrings can it end in (i.e., what can the rightmost two symbols be ?)
  - It cannot end in 01

Questions on Reg. Expressions ?

DFA/NFA to Regular Expression

- we will outline a procedure in the lecture – works smoothly if you code the algorithm
  - Can be tedious to do by hand for a small-ish DFA/NFA

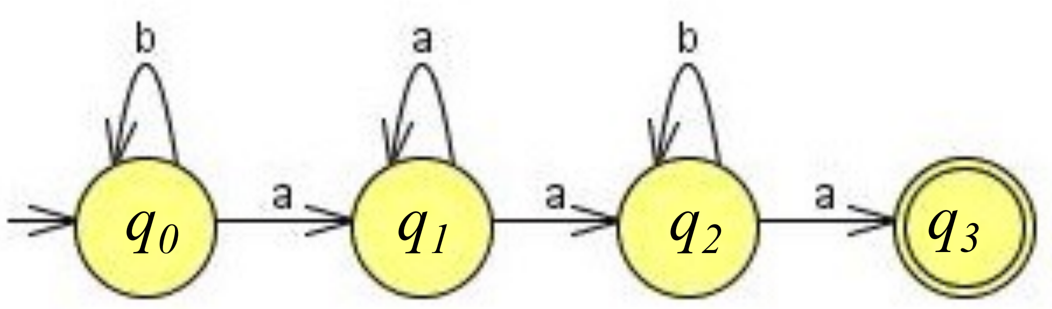
- Alternate approach: by examining the automaton and figuring out the expressions for paths to a final state

DFA/NFA to Regular Expression

- language accepted by a DFA/NFA = { w | there is a path labelled w from start state to a final state}
- To find regular expression for the language accepted by a DFA/NFA, find the labels (and reg. expr.) of the paths from start state to each final state
  - Concatenate labels on the path – the label is the regular expression
    - Concatenate labels on the subpaths
  - If we have two choices of paths with labels w_1 and w_2 then "or" the paths to get w_1+w_2
  - If there is a cycle, with path labelled w, then w^*

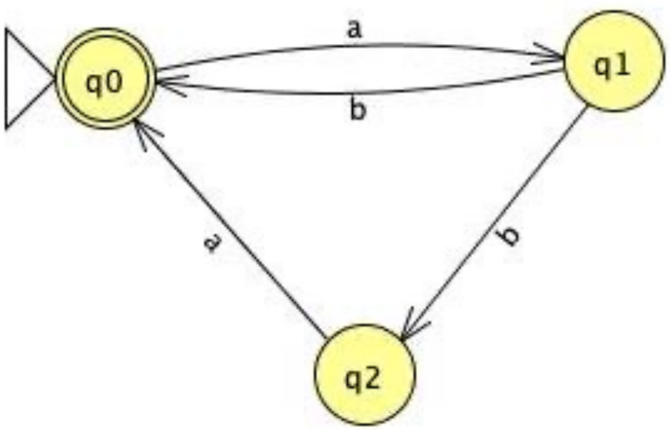
DFA to Reg. Expression – Example 1

- Find expression for paths from q_0 to q_3:
  - Paths from q_0 to q_1 followed by q_1 to q_2 followed by q_2 to q_3
- b* a followed by a*a followed by b*a
- Reg expr= $b^*a\ a^*\ a\ b^*\ a$



Automaton to Reg. Expression – Example 2

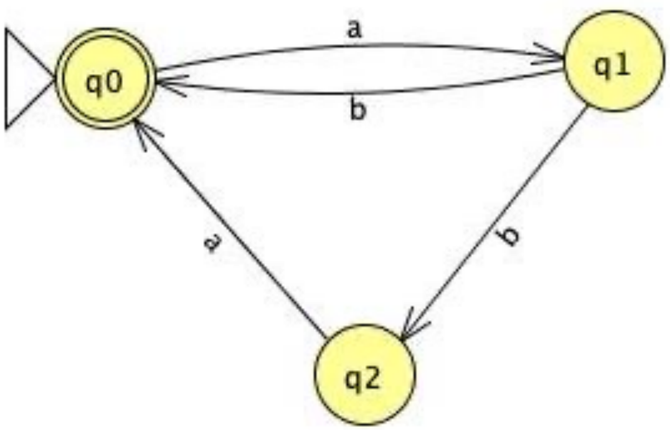
- Find expression for all paths from start state to a final state
- Example: paths from q_0 to q_0
  - q_0 to q_1 to q_0 =
  - q_0 to q_1 to q_2 to q_0 =
  - But: can repeat cycle from q_0 to q_0
  - q_0 to itself on empty string λ
- Therefore: *Reg. Exp.* =



Automaton to Reg. Expression – Example 2

- Find expression for all paths from start state to a final state
- Example: paths from q_0 to q_0
  - q_0 to q_1 to q_0 = (ab)
  - q_0 to q_1 to q_2 to q_0 = (aba)
  - But: can repeat cycle from q_0 to q_0
  - q_0 to itself on empty string λ
- Therefore: *Reg. Exp.* $= (ab + aba)^*$

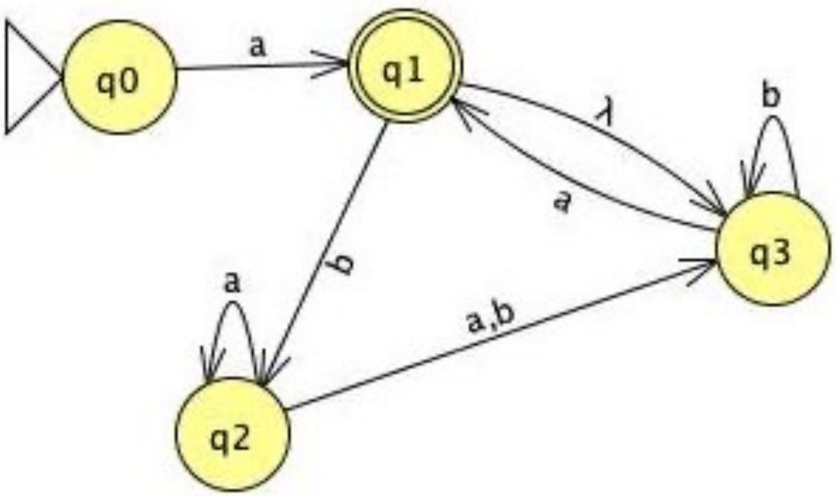
$$= (a.(b + ba))^*$$



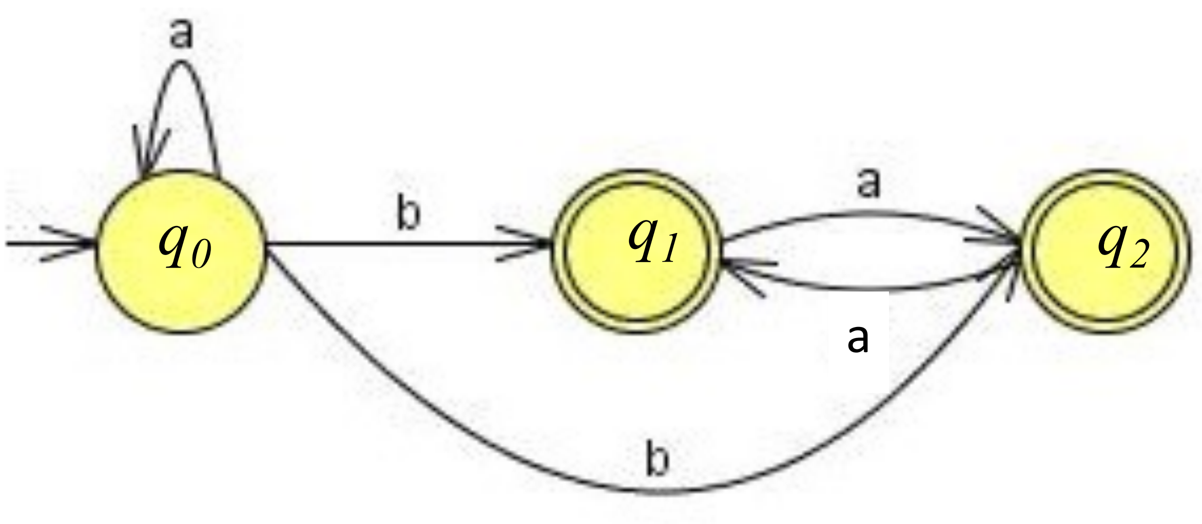
NFA to Reg. Expression – Example 3

- Direct edge label a from start to the final state q_1
- Cycles/path from q_1 to q_1 : consider the two paths –
  - either utilization the λ : $\lambda\ b^*\ a = (b^*a)$
  - or not: $(\ b\ a^*\ (a+b)\ b^*\ a)$
- Therefore, cycle is: $(\ (\ b\ a^*\ (a+b)\ b^*\ a) + \ (b^*a)\ )^*$
- Therefore reg. expr. Is

$a\ (\ (\ b\ a^*\ (a+b)\ b^*\ a) + \ (b^*a)\ )^*$



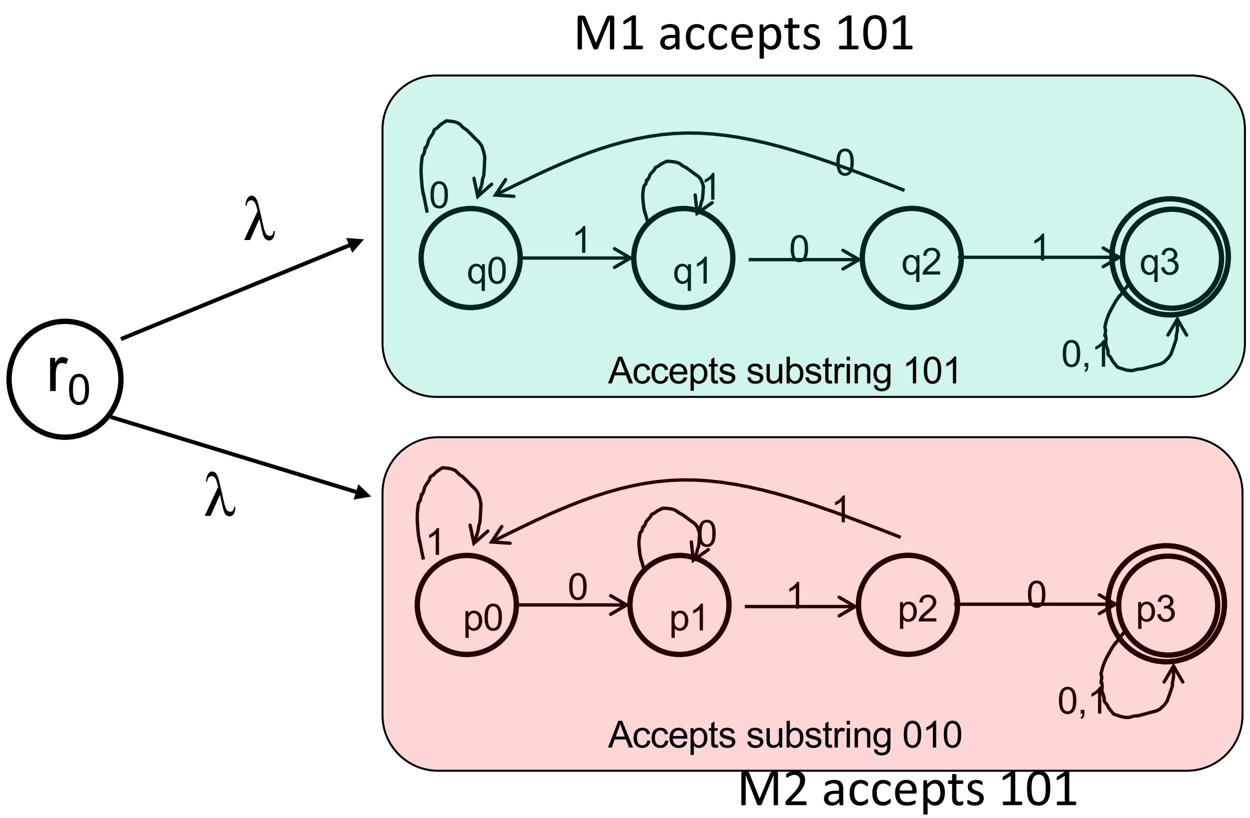
DFA to Reg. Exercise ?



Questions ?

Extra Slides/Examples

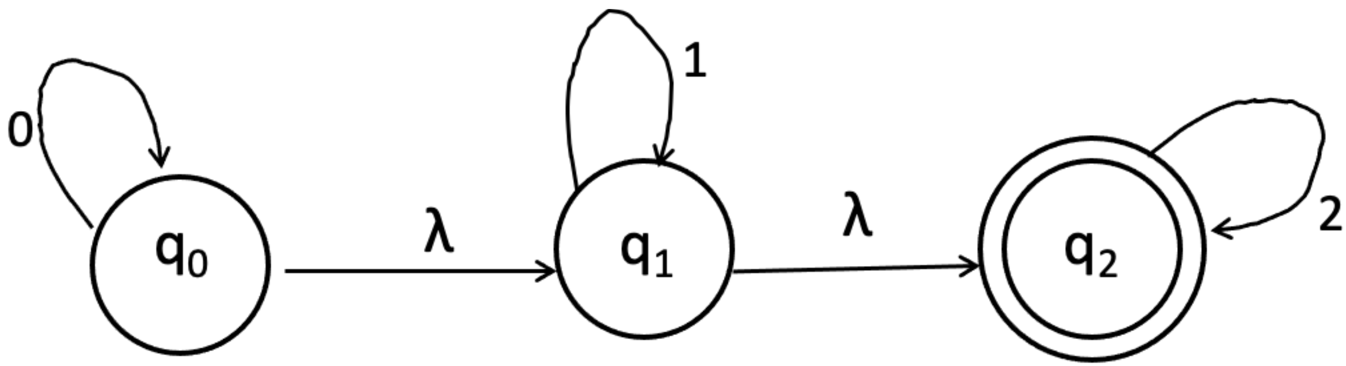
NFA Example 1: L = { w | w has (a) Substring 101 or (b) substring 010}



From new start state r0,
Start both automaton M1 and M2 (green and pink) at the same time
If one of them goes to a final state then accept input

NFA Example 2:L = {w | w has 0's followed by 1's followed by 2's}

- L = {w | w has 0's followed by 1's followed by 2's}
- think of solution as three machines in sequence:
- M1 only accepts 0's, M2 only accepts 1's, M3 only 2's
- Start M1, after it finished start M2, after it finished start M3



DFA to Reg. Expression – Example 4

- final state = q_0
- set of strings (reg. expr.) from q_0 to q_1 to q_0 = ? $(ab)^+$
- set of strings (reg. expr.) from q_0 to q_2 = ? (aa + b)
- Set of strings from q_0 to q_2 to q_1 = ? (aa+b).b
- Set of strings from q_0 to q_2 to q_1 to q_0 = ? ((aa+b).bb)
- Set of strings from q_2 to q_2 = ? (ba)*
- Putting it all together….. $\lambda + (ab)^+ + ((aa+b)(ba)^*\, bb)^*$