CS 3313
Foundations of Computing:

Lab 3: NFA and Regular Expessions Review

Outline

- Building NFAs
- NFA to DFA Conversion
- Regular Expressions
- NFA to Regular Expressions Conversion

NFA vs. DFA

NFAs have 2 critical differences from DFAs

- Allow state-to-state transitions on empty string input ϵ
  - These transitions are done spontaneously, without looking at the input string.
- Allow more than one outgoing transition on same symbol
  - Allows the NFA to choose which path to take
  - Still need to verify that path taken leads to an accept state

Important: Still need to make sure that only strings in desired language are accepted

Exercise 1: work in groups

- Provide an NFA M that accepts the language L over alphabet {0,1,2} where L = { w | (a) w has two consecutive 0's or (b) w has a 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.

Outline

- Building NFAs
- NFA to DFA Conversion
- Regular Expressions
- NFA to Regular Expressions Conversion

Converting NFA to DFA

- We proved in class that NFAs and DFAs recognize the same languages
- So, for every NFA N, we can construct equivalent DFA M
- In class, we gave a procedure for converting an NFA to an equivalent DFA

Let’s review

NFA $N=(Q,\Sigma,\delta,q_0,F)$

- Q – set of states
- Σ – alphabet
- q_0 – start state
- F – accept state(s)
- δ – transition function

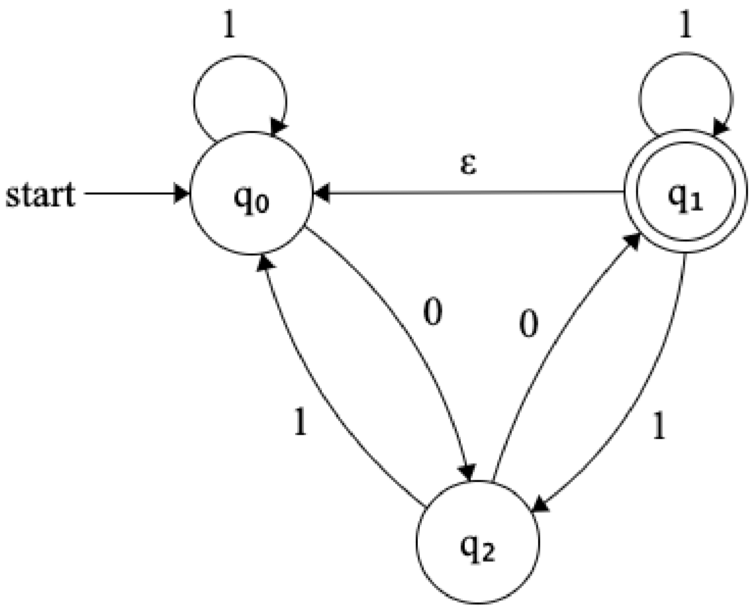
DFA $M= (Q',\Sigma',\delta',q_0',F')$

- Q' = P(Q) – powerset of Q
- $\Sigma' = \Sigma$
- q_0' = E(q) – set of states reachable from q via ϵ edges
- F' = Set of nodes in Q' that contain an accept state from Q
- δ' = Use the "finger trick":

  i.e., set of all possible states that can be reached from current set $q' \in Q'$

Exercise 2: NFA to DFA – Work in groups

Construct a DFA equivalent to the following NFA



Outline

- Building NFAs
- NFA to DFA Conversion
- Regular Expressions
- NFA to Regular Expressions Conversion

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 Ø denotes the empty set
  2. The regular expression ϵ denotes the set { ϵ }
  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. $r_1 \cup r_2$ is a RE denoting the language $L(r_1) \cup L(r_2)$
  2. $r_1 r_2$ is a RE denoting the language $L(r_1) \circ L(r_2)$
  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} = $(a \cup b)^*$
- Two consecutive a’s = aa
- Ends with a pattern aba: $(a \cup b)^* aba$
- ….

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 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 Examples

1. L_1= { all strings over alphabet {a,b,c} that contain no more than three a's }

- Can contain zero a's or 1 a or 2 a's or 3 a's; and can have any number of b,c before and after
- $=(b \cup c)^* \cup \left((b \cup c)^* a (b \cup c)^*\right) \cup \left((b \cup c)^* a (b \cup c)^* a (b \cup c)^*\right) \cup ((b \cup c)^* a (b \cup c)^* a (b \cup c)^* a (b \cup c)^*)$

2. L_2 = = { all binary strings ending in 01 }

Regular expressions Examples

1. L_1= { all strings over alphabet {a,b,c} that contain no more than three a's }

- Can contain zero a's or 1 a or 2 a's or 3 a's; and can have any number of b,c before and after
- =$(b \cup c)^* \cup \left((b \cup c)^* a (b \cup c)^*\right) \cup \left((b \cup c)^* a (b \cup c)^* a (b \cup c)^*\right) \cup ((b \cup c)^* a (b \cup c)^* a (b \cup c)^* a (b \cup c)^*)$

2. L_2 = = { all binary strings ending in 01 }
   - Any string w in {0,1}* followed by 01 = $(0 \cup 1)^* 01$

Exercise 3: Regular Expressions – Work in groups

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

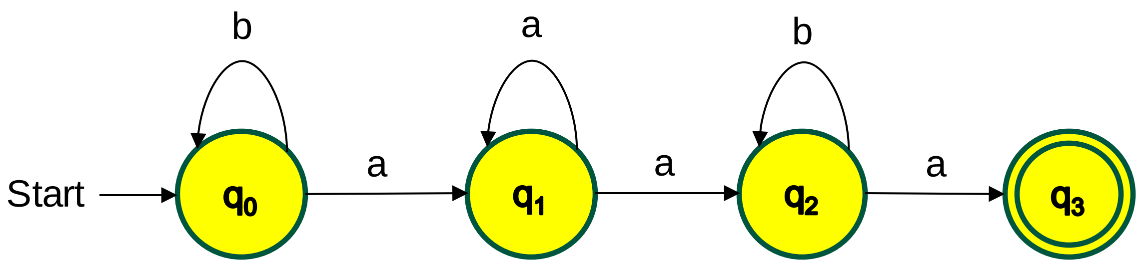
Outline

- Building NFAs
- NFA to DFA Conversion
- Regular Expressions
- NFA to Regular Expressions Conversion

DFA/NFA to Regular Expression

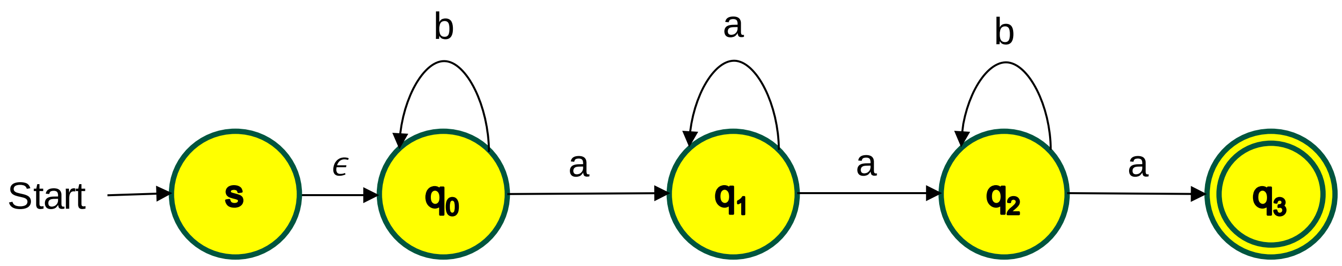
- We outlined a procedure in the lecture based on state elimination:
  1. Start with generalized NFA
     a. Start state has no incoming edges
     b. Only one accept state with no outgoing edges
     c. Edges labeled by regular expressions
  2. One-by-one remove the states
  3. When removing a state, add edges with regular expressions corresponding to all paths through that state
  4. When down to 2 states, we are done

NFA to Reg.Expression – Example 1



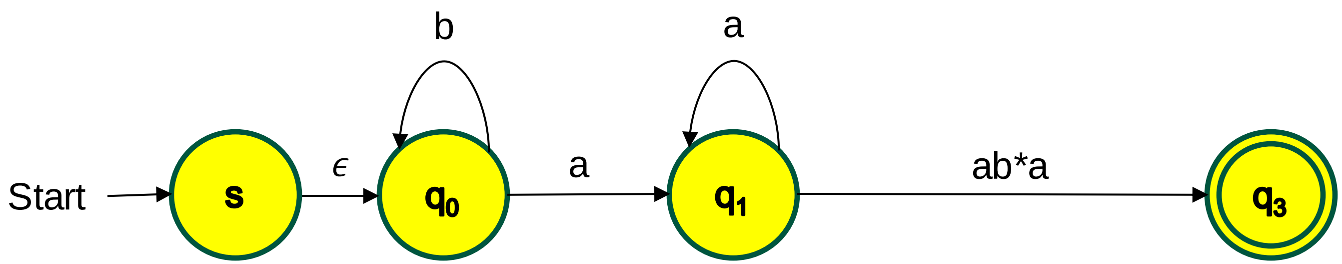
1. Convert to generalized NFA

NFA to Reg.Expression – Example 1



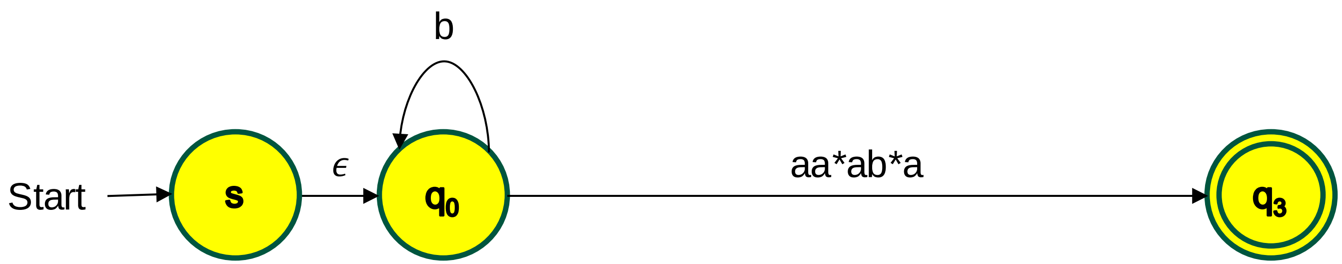
1. Convert to generalized NFA
2. Eliminate q_2

NFA to Reg.Expression – Example 1



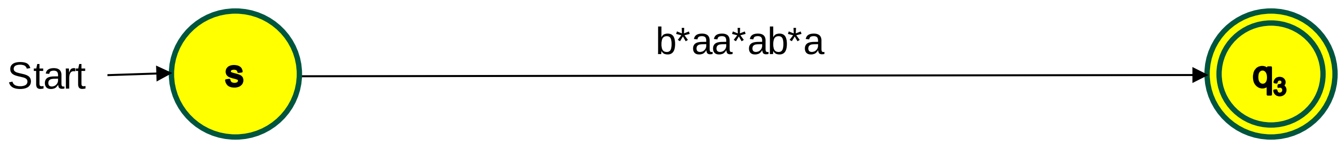
1. Convert to generalized NFA
2. Eliminate q_2
3. Eliminate q_1

NFA to Reg.Expression – Example 1



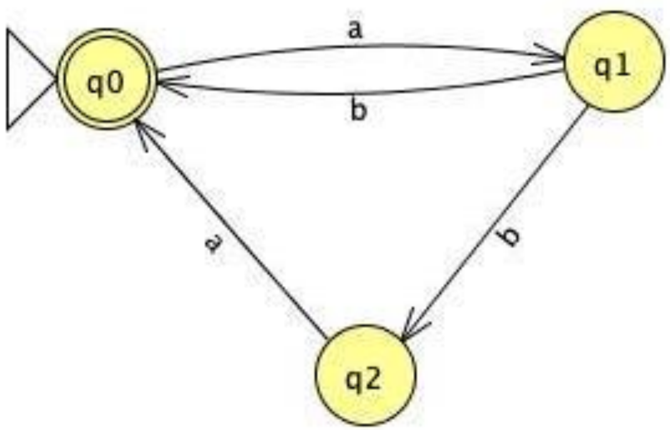
1. Convert to generalized NFA
2. Eliminate q_2
3. Eliminate q_1
4. Eliminate q_0
5. Regular Expression =

NFA to Reg.Expression – Example 1



1. Convert to generalized NFA
2. Eliminate q_2
3. Eliminate q_1
4. Eliminate q_0
5. Regular Expression = b*aa*ab*a

Automaton to Reg. Expression – Example 2



1. Convert to generalized NFA
2. Eliminate q_2
3. Eliminate q_1
4. Eliminate q_0
5. Regular Expression = $(a(b \cup ba))^*$

Exercise 4: NFA to Reg. Exp. – Work in groups