CS 3313
Foundations of Computing:

Lab 4: Regular Expessions Review and the Pumping Lemma

Outline

- Regular Expressions
- NFA to Regular Expressions Conversion
- NFA/DFA Pumping Lemma

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 }
   - 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
   - $=(b \cup c)^* \cup \left((b \cup c)^* a (b \cup c)^*\right) \cup \left((b \cup c)^* a (b \cup c)^* a (b \cup c)^*\right) \cup \left((b \cup c)^* a (b \cup c)^* a (b \cup c)^* a (b \cup c)^*\right)$

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
   - =$(b \cup c)^* \cup \left((b \cup c)^* a (b \cup c)^*\right) \cup \left((b \cup c)^* a (b \cup c)^* a (b \cup c)^*\right) \cup \left((b \cup c)^* a (b \cup c)^* a (b \cup c)^* a (b \cup c)^*\right)$

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

Exercise 1: 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

- Regular Expressions
- NFA to Regular Expressions Conversion
- NFA/DFA Pumping Lemma

DFA/NFA to Regular Expression

1. State Elimination
   - We outlined a procedure in the lecture based on state elimination
   - You will need to do this on the homework

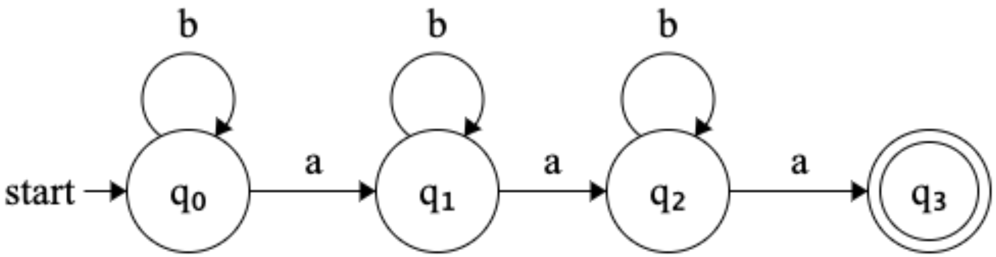
2. Alternate approach
   - Examine the automaton and figure out the expressions for paths from start to a final state
   - This works well for simple DFA/NFA, but may be hard for more complicated examples

DFA/NFA to Regular Expression – Alternate Approach

- 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 $(w_1 \cup w_2)$
  - If there is a cycle, with path labelled w, then w^*

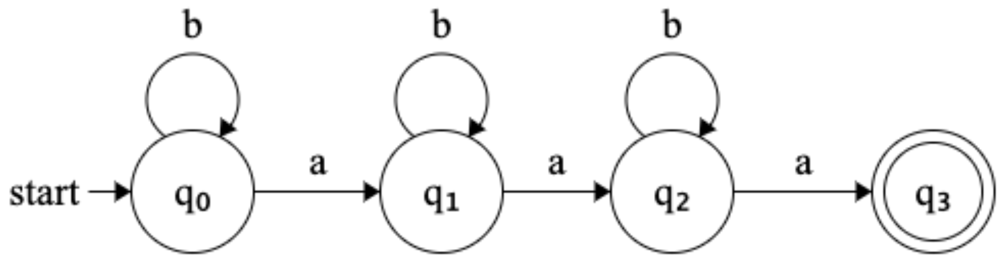
NFA to Regular Expression – Example 1

- Find a regular expression corresponding to below NFA:

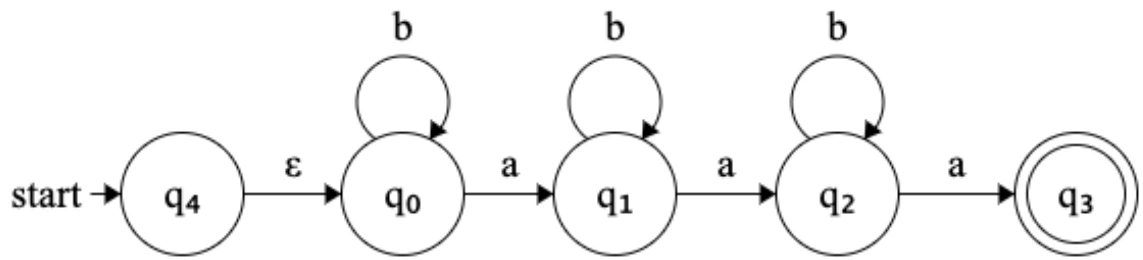


Example 1 by Node Elimination

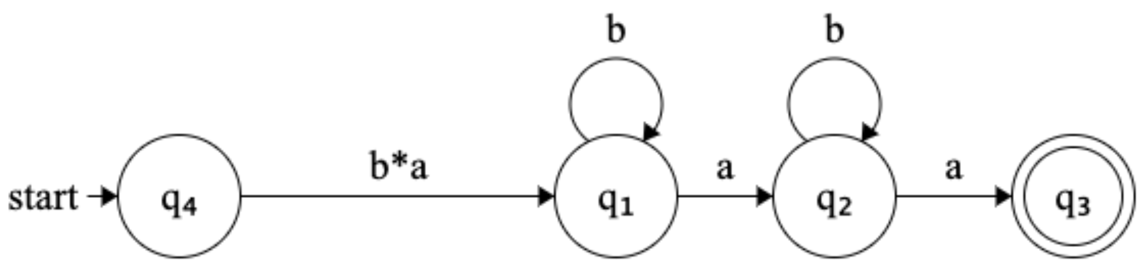
Original DFA



1. Add start state to avoid incoming edges

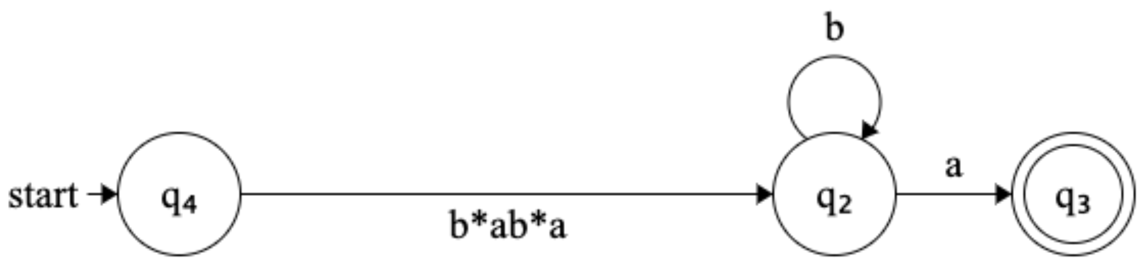


2. Remove q_0

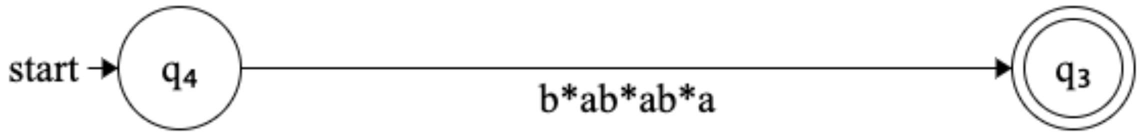


Example 1 by Node Elimination

3. Remove q_1



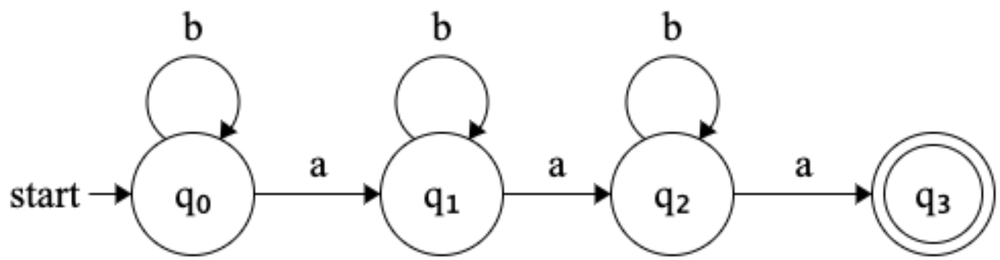
4. Remove q_2



5. Read off answer

L=b*ab*ab*a

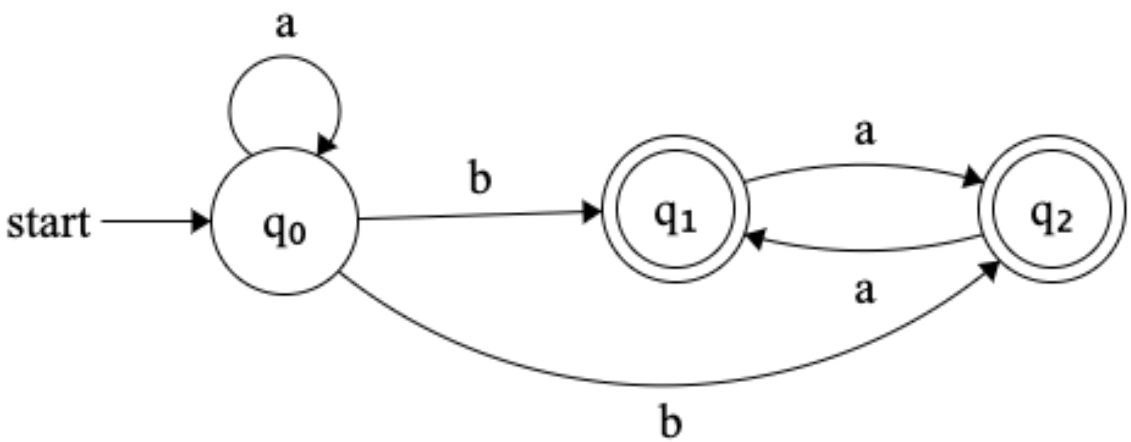
NFA to Regular Expression – Alternate Approach



- 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 b*a followed by b*a
- Reg expr= $b^*a\ b^*\ a\ b^*\ a$

Exercise 2: NFA to Reg. Exp. – Work in groups

- You can use either approach here, but on the homework must use node elimination



Outline

- Regular Expressions
- NFA to Regular Expressions Conversion
- NFA/DFA Pumping Lemma

How to prove a language is not regular…
The Pumping Lemma for Regular Languages

For every regular language L

There is an integer p, such that (note; you cannot fix p)

For every string w in L of length $\geq p$ *(you can choose w)*

We can write $w = xyz$ such that:

1. $|xy| \leq p$ *(this lets you focus on pumping within first p symbols)*
2. $|y| > 0$ *(y cannot be empty)*
3. *For all* $i \geq 0$, xy^iz *is in L.* *(to get contradiction find one value of i where pumped string is not in L)*

Pumping Lemma as an Adversarial Game

1. Player 1 (me) picks language L to be proved nonregular
   - Prove $L = \{ss^R \mid s \in \{a, b\}^*\}$ is not regular.

2. Player 2 picks p, but doesn't tell me what p is, player 1 must win for all values of p
3. Player 1 picks a string w, which may depend on p, and must be of <u>length at least p</u>

   - Assume L is regular. Let w $= a^p b^1 b^1 a^p \in L$, i.e., s $= a^p b^1$; as well as $|s| \geq p$.

<u>Note</u>: Words in purple are the example wordings we use in this type of proofs.

Pumping Lemma as an Adversarial Game

4. Player 2 divides w into xyz s.t. |y|>0 and |xy|<=p
    - He does not tell player 1 this division, player 1's strategy must work for all choices

➢ Then by the Pumping Lemma, w can be divided into three parts $w = xyz$, such that $x = a^{\alpha}, y = a^{\beta}, z = a^{p-\alpha-\beta}b^1b^1a^p$, where $\beta \geq 1, (\alpha + \beta) \leq p$.

5. Player 1 "wins" by picking an integer k>=0, which may be a function of p,x,y, and z, such that $xy^kz \notin L$

➢ Now, consider $k = 0$. Then the string after the pumping becomes $w' = xy^0z = xz = a^{p-\beta}b^1b^1a^p$. Note that since $\beta \geq 1$, there's no way for w' to be in the form of a string followed by its reverse; hence $w' \notin L$. *Contradiction*. $\Longrightarrow L$ not regular.

Pumping Lemma Remarks

- How do we know what string we need to choose?
  - **Trial and Error** and some eureka
  - $L = \{ww^R \mid w \in \{a, b\}^*\}$, if we'd chosen $s = a^n a^n$, then for $s' = a^{n-\beta} a^n$, then adversary can just choose $\beta \geq 1$ to be of even length, such that $s' = w' w'^R$. So, choosing such an s has no use for us.
  - $L = \{a^n b^m \mid m \neq n,\ n, m \geq 1\}$, by choose $s = a^p b^{p+1}$ or $s = a^p b^{2p}$, can we find some integer k such for $s' = xy^k z$, number of a's equals to number of b's. [We saw this in class]

Exercise 3: Pumping Lemma

Exercise: Prove that $L = \{a^m b^n \mid m < n\}$ is not regular.

1. What string s should we choose?
2. What does the pumping lemma tell us?
3. How to complete the proof?