Lecture 5

Announcements:

- HW1 OUT. New due date: Tues Oct 3, 11:59 pm
- My office hours this week: Fri 6-7 pm (by zoom)
  SUN 8-9 pm

Recap from last class

→ we showed regular languages are closed under operations $+, \circ, *$

→ Defined regular expressions, and the language associated with a regular expression

Closure Properties of Regular Languages

1. If $L \subseteq \Sigma^*$ is regular, then $\overline{L} = \{ w \in \Sigma^* \mid w \notin L \}$ is also regular

2. If L is regular, then $L^* = \{ w \mid w = v_1 \cdot v_2 \cdot \ldots \cdot v_k \mid v_1, \ldots, v_k \in L \}$ is regular

3. If L_1 and L_2 are regular, then $L_1 + L_2 = \{ w \mid w \in L_1 \text{ or } w \in L_2 \}$ is regular

4. If L_1 and L_2 are regular, then $L_1 \circ L_2 = \{ w \mid$ w can be written as $u \cdot v$ where $u \in L_1$ and $v \in L_2 \}$

Formal Definition of a Regular Expression

Let Σ be a finite alphabet

R is a regular expression over Σ if:

1. $R = a$ for some $a \in \Sigma$
2. $R = \varepsilon$
3. $R = \phi$

$\Bigg\}$ base cases

4. $R = R_1 + R_2$ where R_1, R_2 are regular expressions over Σ
5. $R = R_1 \cdot R_2$ where R_1, R_2 are regular expressions over Σ
6. $R = (R_1)^*$ where R_1 is a regular expression over Σ

$\Bigg\}$ inductive cases

* Note: in book $+$ is $\cup$ (union)

$\cdot$ is $\circ$ (concatenation)

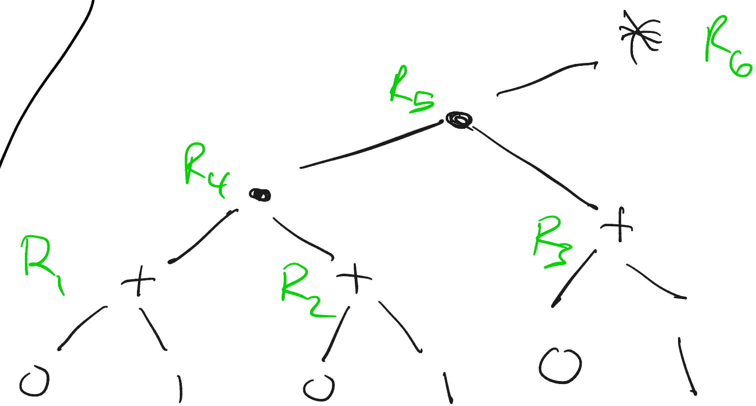
More Examples

1. $((0+1)\cdot(0+1)\cdot(0+1))^*$ $\Rightarrow$

$$\overbrace{\big(\overbrace{\underbrace{(\underbrace{(0+1)}_{R_1}\cdot\underbrace{(0+1)}_{R_2})}_{R_4}\underbrace{(0+1)}_{R_3}}^{R_5}\big)^*}^{R_6}$$

2. $(0+1)^*\cdot 1\cdot 1\cdot 1\cdot(0+1)^*$

3. $1^*\cdot 0\cdot 1^*$



Theorem Let Σ be a finite alphabet.
The class of languages over Σ that are regular is equal to the class of languages that are described by regular expressions

Proof has 2 directions:

(i) L has a regular expression $\rightarrow$ L has an NFA (and therefore L has a DFA so L is regular)

(ii) L has a DFA (or NFA) $\rightarrow$ L has a regular expression

Theorem Let Σ be a finite alphabet.
The class of languages over Σ that are regular is equal to the class of languages that are described by regular expressions

Proof has 2 directions:

(i) L has a regular expression $\longrightarrow$ L has an NFA
(proof uses closure properties!)

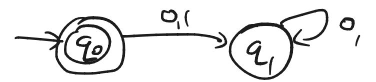
(ii) L has a DFA (or NFA) $\longrightarrow$ L has a regular expression
(harder)

(i) L has a regular expression $\rightarrow$ L has an NFA

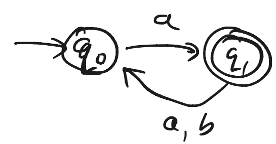
Proof by induction on length of regular expression for L.

Base cases: $L = \phi$

$L = \varepsilon$



$L = \{a\}$, $a \in \Sigma$



(i) L has a regular expression $\rightarrow$ L has an NFA

Proof by induction on length of regular expression for L.

Inductive step:

IND hyp: For any L described by a regular expression involving at most K operations $*, +, \bullet$, L has an NFA

show: any L described by reg expression with K operations has an NFA

(i) L has a regular expression → L has an NFA

Inductive step:

IND hyp: For any L described by a regular expression involving at most K operations $*, +, \cdot$, L has an NFA

show: any L described by reg expression with K operations has an NFA

3 cases: (i) $L = (L_1)^*$

(ii) $L = L_1 + L_2$

(iii) $L = L_1 \cdot L_2$

where L_1, L_2 described by regular expressions using $\leq K$ operations

(i) follows by closure property 2
(ii) " " " " 3
(iii) " " " " 4

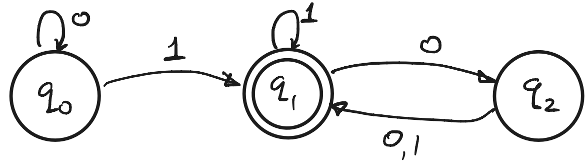
see first slide from this lecture

(ii) L has a DFA (or NFA) $\Rightarrow$ L has a regular expression

To prove this direction we will give an <u>algorithm</u> that takes as input a DFA or NFA M produces a regular expression such that the language accepted by M corresponds to the regular expression

* Our algorithm different (and easier I think) than Sipser. See supplemental material for more info on the algorithm we give today.

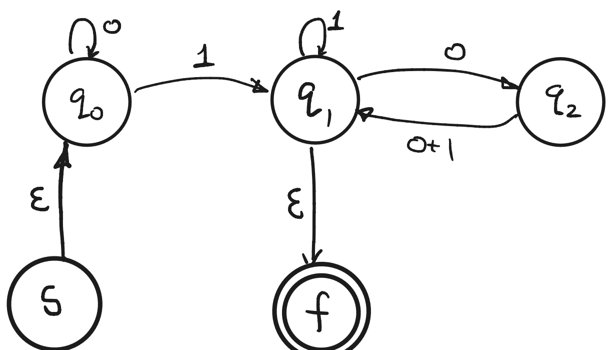
Example 1 : constructing Regular Expression from an NFA



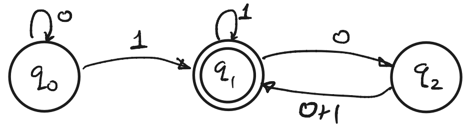
From Lecture 1 !

step 1

- New start state s with ε-transition to original start state
- New single accept state f with ε-transitions from old accept states to f

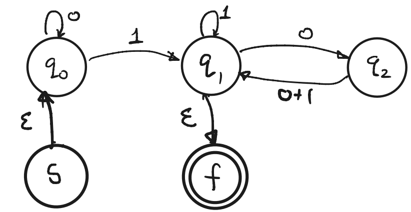


Step 0

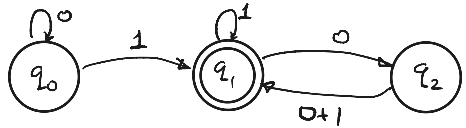


step 1

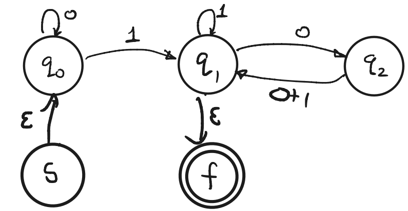
- New start state s with ε-transition to original start state
- New single accept state f with ε-transitions from old accept states to f



Step 0

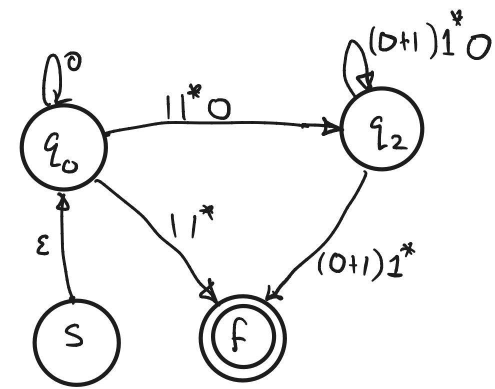


Step 1



- New start state s with ε-transition to original start state
- New single accept state f with ε-transitions from old accept states to f

Step 2 (remove q_1)



- Consider all pairs of edges $(q \to q_1,\ q_1 \to q')$, $q, q' \neq q_1$

$q_0 \to q_1 \quad q_1 \to q_2$: $1 1^* 0$

$q_0 \to q_1 \quad q_1 \to f$: $1 1^*$

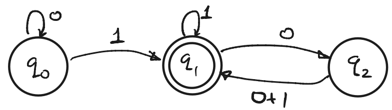
$q_2 \to q_1 \quad q_1 \to f$: $(0+1)1^*$

$q_2 \to q_1 \quad q_1 \to q_2$: $(0+1)1^*0$

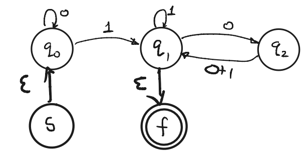
- Remove q_1.

- For all pairs $q \to q_1 \quad q_1 \to q'$ add the corresponding regular expression to edge $q \to q'$

Step 0

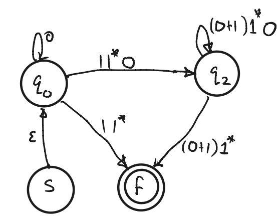


Step 1



- New start state s with ε-transition to original start state
- New single accept state f with ε-transitions from old accept states to f

Step 2 (remove q_1)



- Consider all pairs of edges $(q \to q_1,\ q_1 \to q')$, $q, q' \neq q_1$

$q_0 \to q_1 \quad q_1 \to q_2 : \; 1\,1^*\,0$

$q_0 \to q_1 \quad q_1 \to f : \; 1\,1^*$

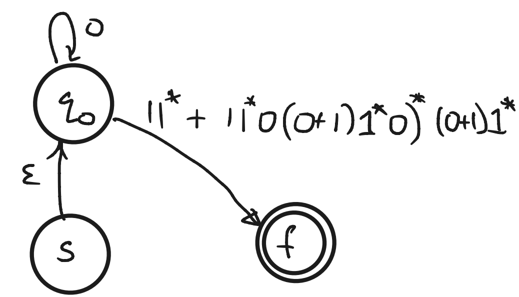
$q_2 \to q_1 \quad q_1 \to f : \; (0+1)1^*$

$q_2 \to q_1 \quad q_1 \to q_2 : \; (0+1)1^*0$

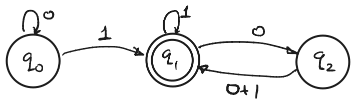
- Remove q_1.
- For all pairs $q \to q_1 \quad q_1 \to q'$ add the corresponding regular expression to edge $q \to q'$

Step 3 (remove q_2)

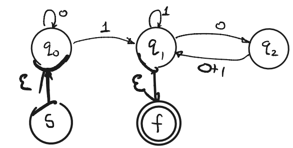
$q_0 \to q_2 \quad q_2 \to f : \; 11^*0\,((0+1)1^*0)^*\,(0+1)1^*$



Step 0

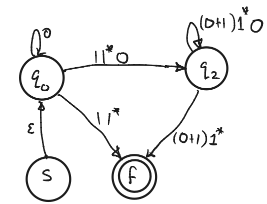


Step 1



- New start state s with ε-transition to original start state
- New single accept state f with ε-transitions from old accept states to f

Step 2 (remove q_1)



- Consider all pairs of edges $(q \to q_1,\ q_1 \to q')$, $q, q' \neq q_1$

$q_0 \to q_1 \quad q_1 \to q_2$: $1\,1^*\,0$

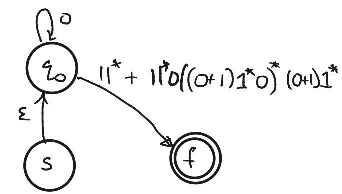
$q_0 \to q_1 \quad q_1 \to f$: $1\,1^*$

$q_2 \to q_1 \quad q_1 \to f$: $(0+1)1^*$

$q_2 \to q_1 \quad q_1 \to q_2$: $(0+1)\,1^*\,0$

- Remove q_1.
- For all pairs $q \to q_1$ $q_1 \to q'$ add the corresponding regular expression to edge $q \to q'$

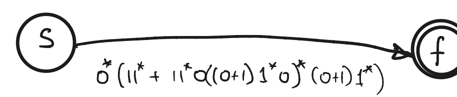
Step 3 (remove q_2)



$q_0 \to q_2 \quad q_2 \to f$: $11^*0\,((0+1)1^*0)^*\,(0+1)1^*$

Step 4 (remove q_0)

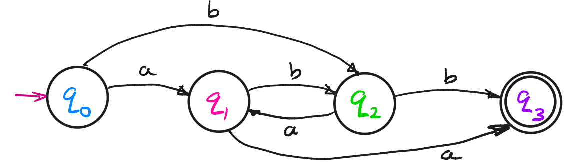
$s \to q_0 \to f$ $\quad 0^*(11^* + 11^*0((0+1)1^*0)^*(0+1)1^*)$



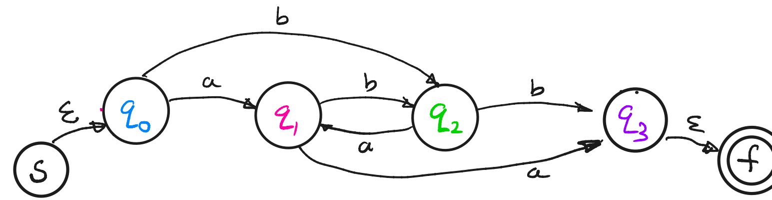
NFA

$01 = 0\varepsilon 1$

Example 2



$\boxed{\text{Step 1}}$



$\boxed{\text{Step 2}}$

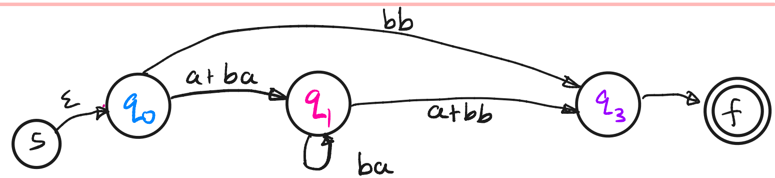
Remove q_2

$q_1 \to q_2 \to q_1$: ba

$q_1 \to q_2 \to q_3$: bb

$q_0 \to q_2 \to q_1$: ba

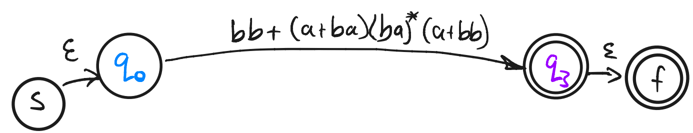
$q_0 \to q_2 \to q_3$: bb



$\boxed{\text{Step 3}}$

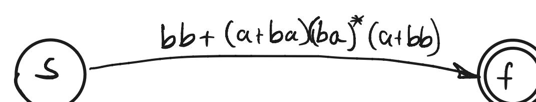
Remove q_1

$q_0 \to q_1 \to q_3$: $(a+ba)(ba)^*(a+bb)$



$\boxed{\text{Steps 4-5}}$

Remove q_0, q_3



Recap so far

1. DFAs and Regular Languages
2. NFAs, and equivalence with DFAs
3. Closure Properties of Regular Languages
4. Regular Expressions and Equivalence with Regular Languages

Recap so far

1. DFAs and Regular Languages

2. NFAs, and equivalence with DFAs

3. Closure Properties of Regular Languages

4. Regular Expressions and Equivalence with Regular Languages

Next:

5. Proving that a language is <u>Not</u> regular: Pumping Lemma

6. DFA state minimization

Nonregular Languages & Pumping Lemma

Warmup: Which of these Languages is regular?

$A = \{ 0^n 1^n \mid n \geq 0 \}$

$B = \{ w \in \{0,1\}^* \mid$ w has equal number of 0's & 1's $\}$

$C = \{ w \in \{0,1\}^* \mid$ w has equal number of occurrences of '01' and '10' $\}$

Nonregular Languages & Pumping Lemma

Warmup: Which of these Languages is regular?

$L_1 = \{w \in \{0,1\}^* \mid$ the number of 0's in w is equal to the number of 1's in $w\}$

$L_2 = \{w \in \{0,1\}^* \mid$ the number of occurrences of '01' in w is equal to the number of occurrences of '10' $\}$

Lower Bounds: How to prove that a language is not regular?

$L = \{0^n 1^n \mid n \geq 0\}$

Tricky since we need to show that every DFA M has to make a mistake with respect to L

(Show: either $\exists w \in L$ not accepted by M, or $\exists w$ accepted by M and not in L)

And there are an infinite number of DFAs!

Lower Bounds: How to prove that a Language is not regular?

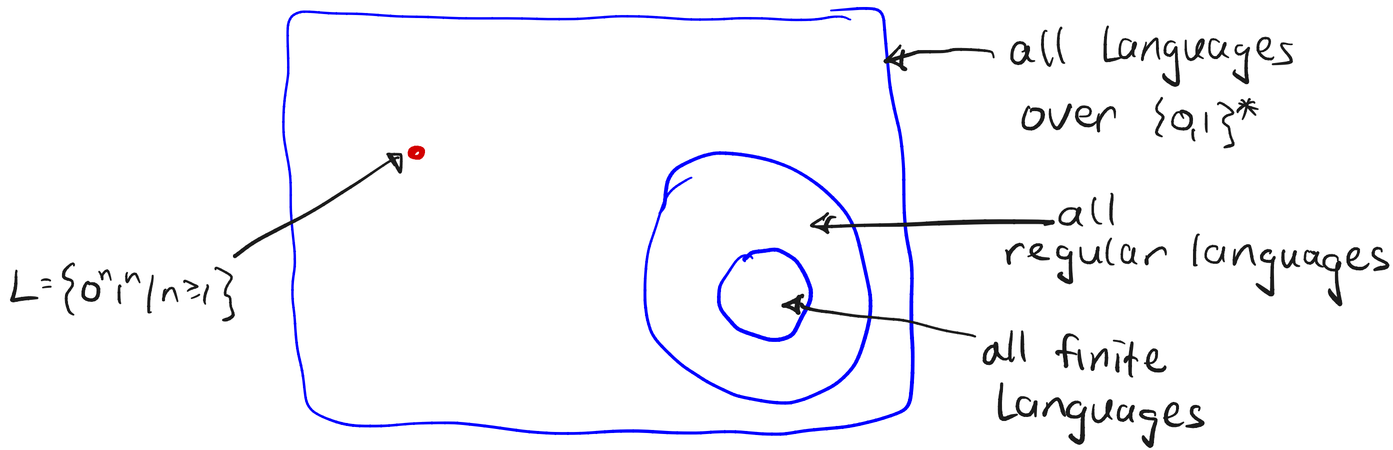
$L = \{0^n 1^n \mid n \geq 0\}$

- Not enough to show that the obvious or natural DFAs don't accept L
- Avoid a common trap:

  L may be defined by some property.
  But we can't assume that a DFA for L needs to be able to recognize/compute that property

Example: $L_2 = \{w \in \{0,1\}^* \mid$ the number of occurrences of '01' in w is equal to the number of occurrences of '10' $\}$

Lower Bounds: How to prove that a Language is not regular?



Proof by contradiction: Assume that L is regular, so some DFA, M, accepts A.

Find some property that all regular Languages have that L doesn't have to get a contradiction.

WARMUP: A language L' is finite if $\exists c \geq 0$ such that $|L| \leq c$

Let's show: $L = \{0^n 1^n \mid n \geq 0\}$ is not a finite language.

Property: A language L' is length-bounded if $\exists k \geq 0$ such that every $w \in L'$ has length $\leq k$

Claim All finite languages are length-bounded.

Proof that $L = \{0^n 1^n \mid n \geq 0\}$ is not finite:

- Assume for sake of contradiction that L is finite
- By claim, $\exists k \geq 0$ such that every $w \in L$ has length $\leq k$.
- But $w = 0^k 1^k \in L$ and $|w| > k$. $\therefore L' \neq L$. So L is not finite

Now we will show that $L = \{0^n 1^n \mid n \geq 0\}$ is Not regular

Main tool: Pumping Lemma.

Key Idea Every DFA has a finite number of states.
Therefore for any DFA M (allegedly accepting language L)
for any sufficiently large $w \in \Sigma^*$, M's computation
on w will Loop.

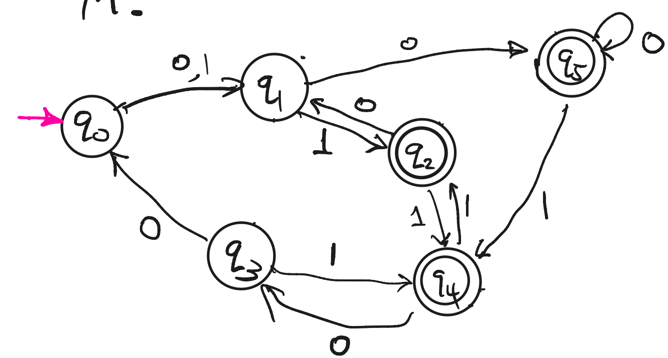
For example, suppose M has K states. Then for every $w \in \Sigma^*$ of length $\geq K$ ($|w| \geq K$), M will loop on w.

Key Idea Every DFA has a finite number of states.
Therefore for any DFA M (allegedly accepting language L) for any sufficiently large $w \in \Sigma^*$, M's computation on w will loop.

For example, suppose M has K states. Then for every $w \in \Sigma^*$ of length $\geq K$ ($|w| \geq K$), M will loop on w.

Example

M:



M has $K = 6$ states

So any string w of length ≥ 6 will loop (repeat a state)

$w = 1011011$ $\quad q_0\ q_1\ q_5\ q_4\ q_2\ q_1\ q_2\ q_4$

$w = 111001$ $\quad q_0\ q_1\ q_2\ q_4\ q_3\ q_0\ q_1$

$w = 1001111$ $\quad q_0\ q_1\ q_5\ q_4\ q_2\ q_4\ q_2$

Proof that $L = \{0^n 1^n \mid n \geq 0\}$ is not regular

Property: For any DFA M, $\exists K \geq 0$ such that for every $w \in \Sigma^*$, $|w| \geq K$, M on w repeats a state.

That is, $\forall w$, $|w| \geq K$, $\exists$ a state q^* satisfying:

we can write $w = xyz$, $|y| > 0$, $|xy| \leq K$ satisfying:

M is in state q^* after reading x, and again is in state q^* after reading xy

Therefore for every $i \geq 0$, the string $w' = xy^i z$ behaves the same as w on M. That is:

M accepts w' if and only if M accepts w

Property: For any DFA M, $\exists K \geq 0$ such that for every $w \in \Sigma^*$, $|w| \geq K$, M on w repeats a state. That is, $\forall w$, $|w| \geq K$, $\exists$ a state q^* satisfying:

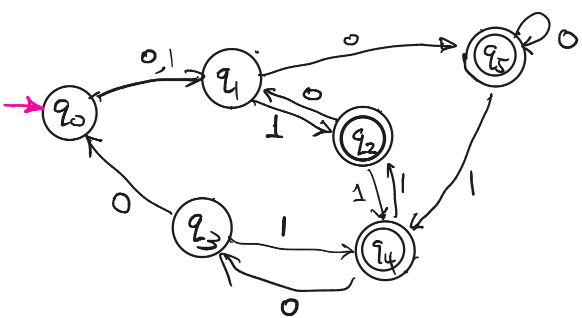
We can write $w = xyz$, $|y| > 0$, $|xy| \leq K$

M is in state q^* after reading x, and again is in state q^* after reading xy

Therefore for every $i \geq 0$, the string $w' = xy^iz$ behaves the same as w on M. That is: M accepts w' if and only if M accepts w

Example

M: $K = 6$



$w = 1011011$ $\quad \underbrace{q_0}_{x}\ \underbrace{q_1\ q_5\ q_4\ q_2\ q_1}_{y}\ \underbrace{q_2\ q_4}_{z}$

$w = 111001$ $\quad \underbrace{\varepsilon}_{x}\ \underbrace{q_0\ q_1\ q_2\ q_4\ q_3\ q_0}_{y}\ \underbrace{q_1}_{z}$

$w = 1001111$ $\quad \underbrace{q_0\ q_1\ q_5}_{x}\ \underbrace{q_4\ q_2\ q_4}_{y}\ \underbrace{q_2}_{z}$

Property: For any DFA M, $\exists k \geq 0$ such that for every $w \in \Sigma^*$, $|w| \geq k$, M on w repeats a state. That is, $\forall w$, $|w| \geq k$, $\exists$ a state q^* satisfying:

We can write $w = xyz$, $|y| > 0$ such that

M is in state q^* after reading x, and again is in state q^* after reading xy

Therefore for every $i \geq 0$, the string $w' = xy^i z$ behaves the same as w on M. That is: M accepts w' if and only if M accepts w

Proof that $L = \{0^n 1^n \mid n \geq 0\}$ is not regular:

Assume that L is regular & Let M be a DFA accepting L, where M has k states

Consider the input $w = 0^k 1^k$. Since $w \in L$, M should accept w (otherwise we reach contradiction)

By above property, we can write $w = xyz$, $|y| > 0$, $|xy| \leq k$

such that $\forall i \geq 0$ $xy^i z$ is also accepted by M (since M accepts w)

Since $|xy| \leq k$, $|y| > 0$, $w = xyz = \underbrace{0^a}_{x} \underbrace{0^b}_{y} \underbrace{0^{k-a-b} 1^k}_{z}$ $b > 0$

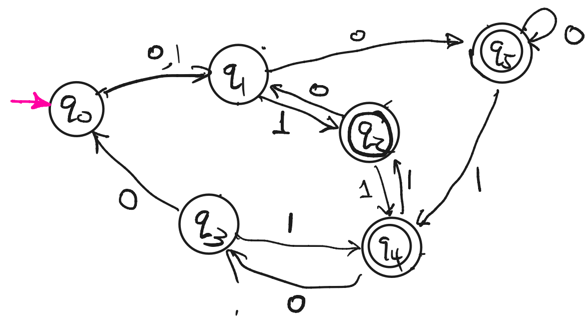
Then the string $xy^2 z = 0^a 0^{2b} 0^{k-a-b} 1^k = 0^{k+b} 1^k$ is accepted by M

But $xy^2 z \notin L$. Contradiction. $\therefore L$ is not regular.

Proof that $L = \{0^n 1^n \mid n \geq 0\}$ is not regular, cont'd

For example:

M:



our string $w = 0^k 1^k = 0^6 1^6 = 0\ 0\ 0\ 0\ 0\ 0\ 1\ 1\ 1\ 1\ 1\ 1$

M on w accepts: $\underbrace{q_0\ q_1}_{x}\ \underbrace{q_5}_{y}\ \underbrace{q_5\ q_5\ q_5\ q_5\ q_4\ q_2\ q_4\ q_2\ q_4\ q_2}_{z}$

M on $\underbrace{xy^2z}_{\text{"pumped" string}}$: also accepts, but $xy^2z \notin L$