CSC 236 Lecture 11: Formal Language Theory 3

Harry Sha

August 2, 2023

Today

- Recap
- NFA to Regex
- Non-regular Languages
- Myhill-Nerode Theorem
  - Statement and Proof
  - Applying the Theorem
- Pumping Lemma
  - Pumping Lemma vs. Myhill Nerode

Regular Languages

The following are equivalent

- A is regular
- There is a DFA M such that $L(M) = A$
- There is a NFA N such that $L(N) = A$
- There is a regular expression R such that $L(R) = A$

Closure

If A, B are regular, so are

- $\overline{A}$
- $A \cup B$
- $A \cap B$
- AB
- A^n
- A^*

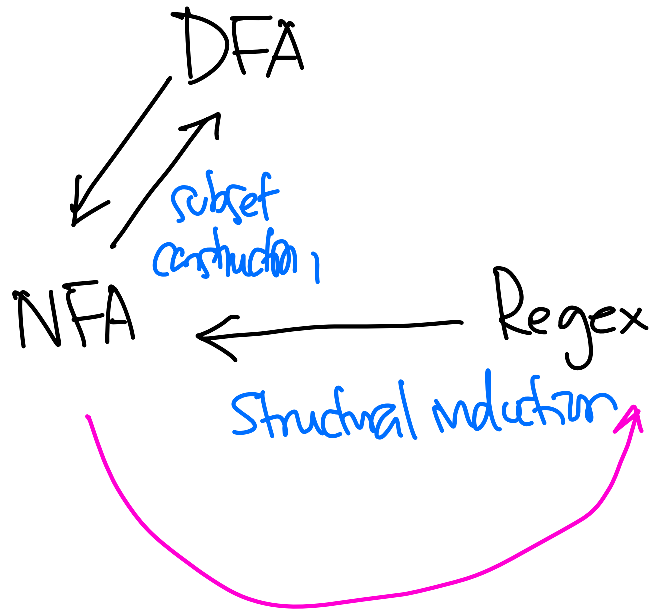
Recap

NFA to Regex

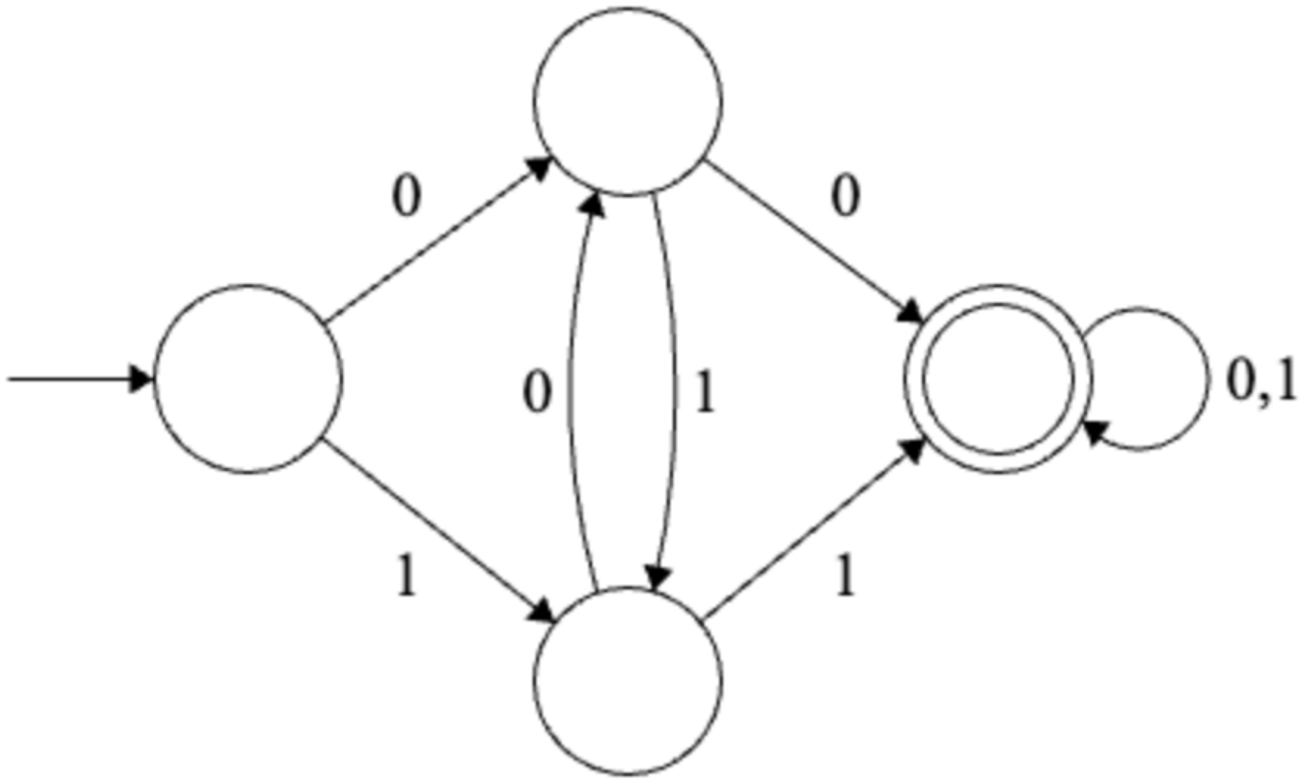
Non-regular Languages

Myhill-Nerode Theorem
- Statement and Proof
- Applying the Theorem

Pumping Lemma
- Pumping Lemma vs. Myhill Nerode

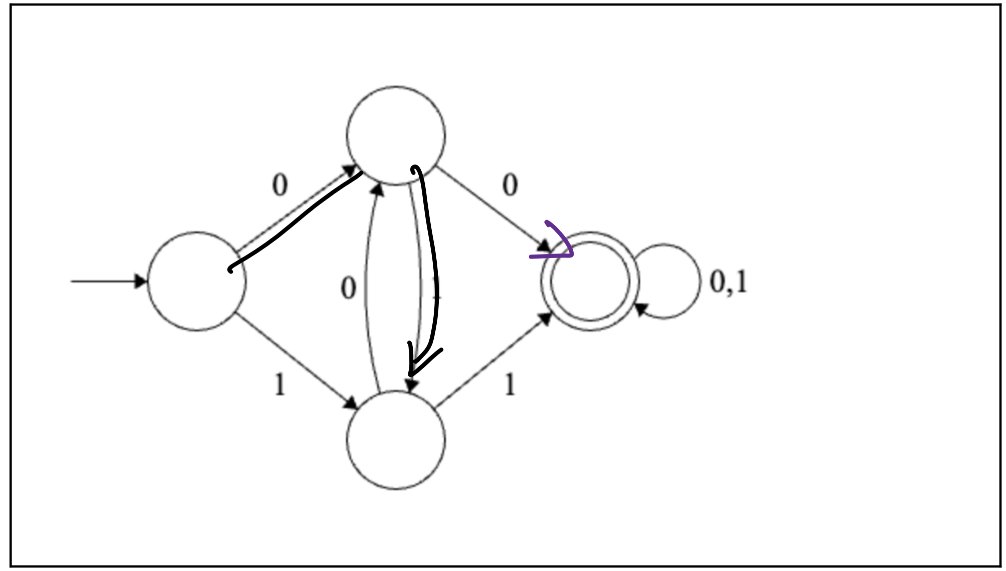


Example [1]

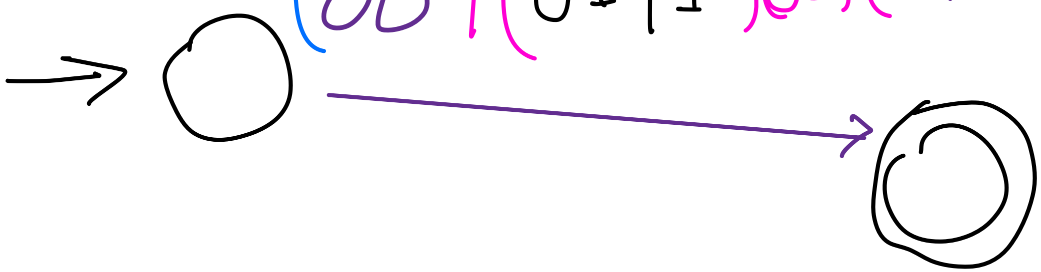


[1] Reference: CSC236 2022 Fall

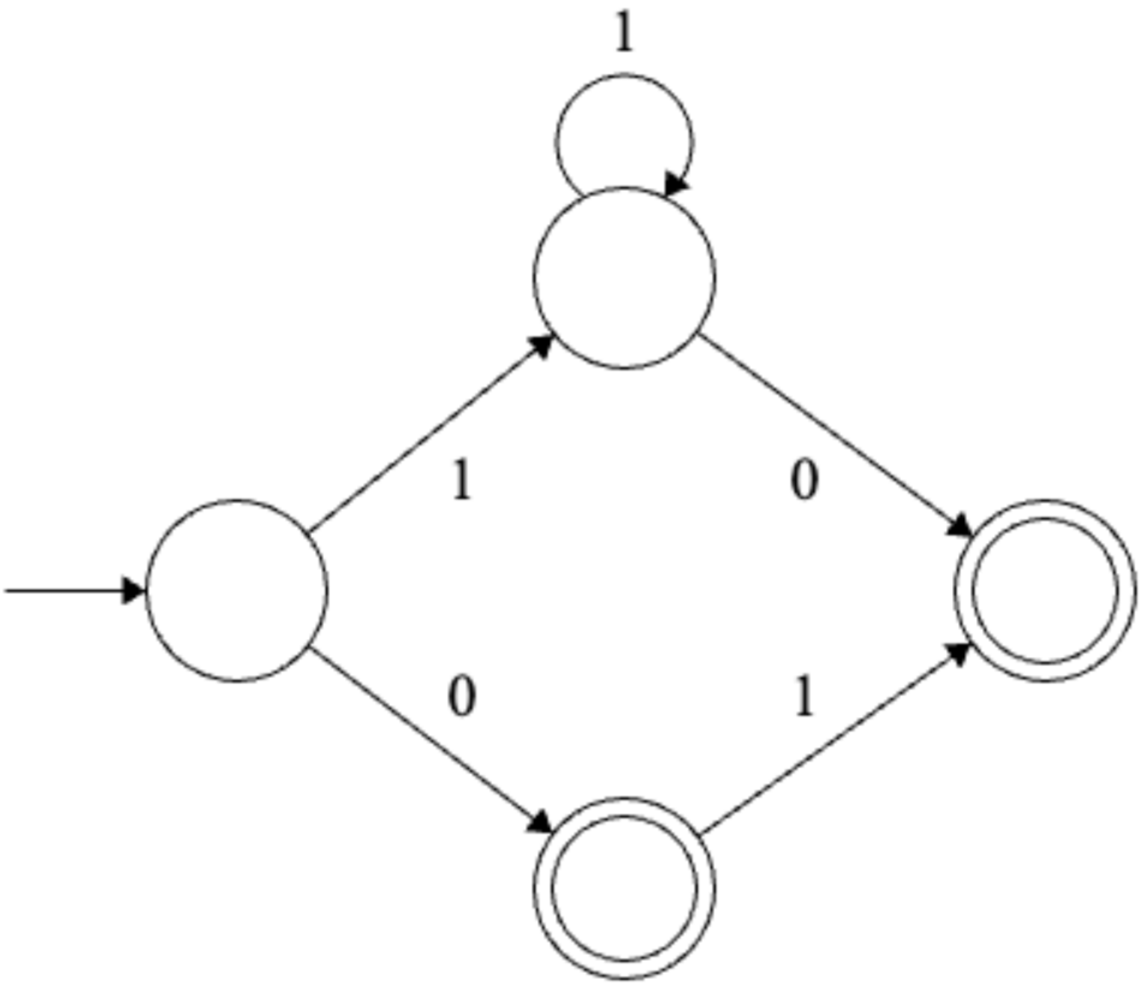
Example



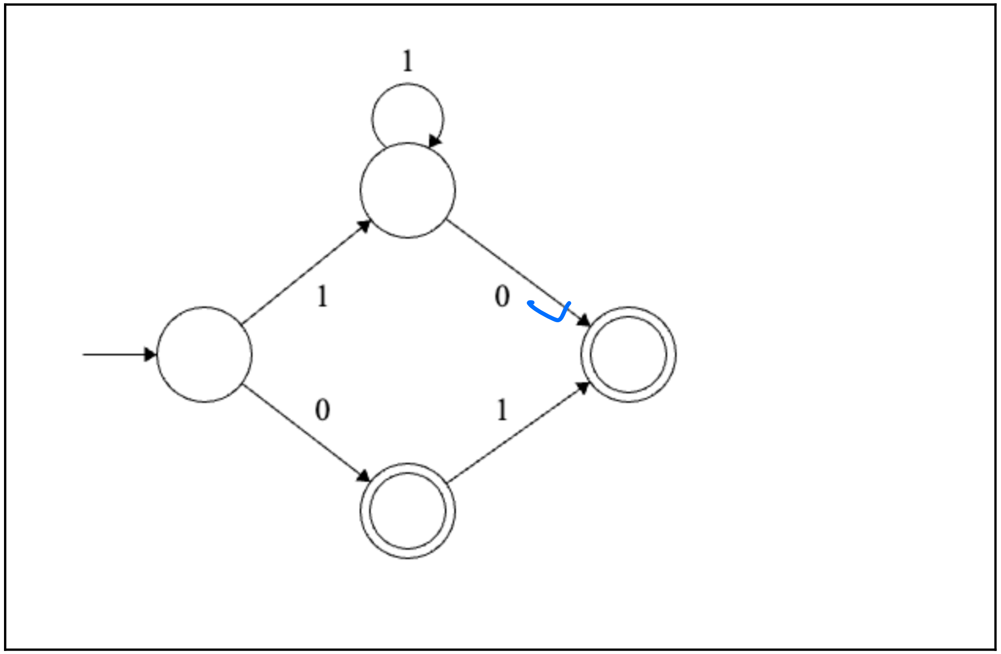
$(00 \mid (01 \mid 1)(01)^*(1 \mid 00))(0 \mid 1)^*$

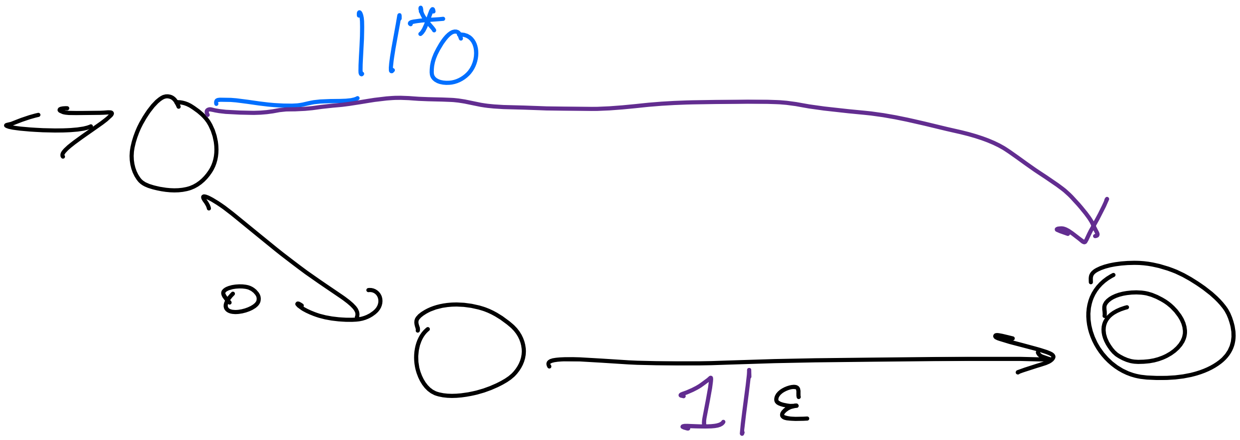


Example 2



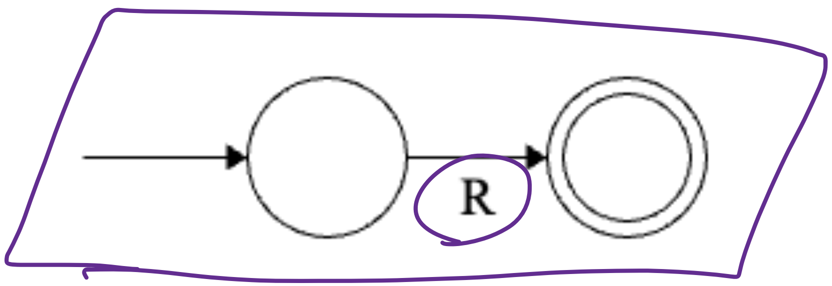
Example 2





Sketch

- Alter the NFA so there's just one accepting state (using ϵ transitions).
- Iteratively rip out states, replacing transitions with regular expressions until you have something that looks like



R is the equivalent regular expression.

“Ripping” out states

For two states q_1, q_2 with a transition between them, let $f(q_1, q_2)$ be the regular expression labelling the transition.

Here are the steps to rip out a state q.

1. **Remove the loop**: If there is a self loop on state q, for each state s with a transition into q, update the transition $f(s, q) = f(s, q)f(q, q)^*$.
2. **Bypass** q: for each path (s, q, t) of length 2 through q, update $f(s, t) = f(s, t)|f(s, q)f(q, t)$. Note that it is possible that $s = t$, in which case this step adds a loop.
3. Remove q.

We showed a bunch of languages were regular...

However, from lecture 1, we know that there are some problems that computers can't solve...

... so what do non-regular languages look like?

What are some limitations for DFAs and NFAs?

Regular languages KEY intuition

D**F**As has a finite number of states.

Regular languages KEY intuition

D**F**As has a finite number of states.

States correspond to memory.

Thus, DFAs can compute languages that only need a finite amount of memory (and read the input once left to right).

In particular, a DFA has a fixed amount of memory, no matter how large the input is.

Example

Even is regular because no matter how large the input is, I only need to store one bit corresponding to whether or not the input has an even number of 1s so far.

Infinite Memory Required

What are some things you can't do with a fixed amount of memory?

Example

Here's an example of a language that can't be computed using finite memory.

$$\{a^n b^n : n \in \mathbb{N}\}$$

Why?

Example

Here's an example of a language that can't be computed using finite memory.

$$\{a^n b^n : n \in \mathbb{N}\}$$

Why?

I don't know ahead of time how many *a*s there are, and I need to keep track of them to see how many *b*s I should expect.

Proving not regular

Intuitively,

$$X = \{a^n b^n : n \in \mathbb{N}\}$$

requires infinite memory so is not regular. However, this doesn't prove that it is not regular.

Proving not regular

Intuitively,

$$X = \{a^n b^n : n \in \mathbb{N}\}$$

requires infinite memory so is not regular. However, this doesn't prove that it is not regular.

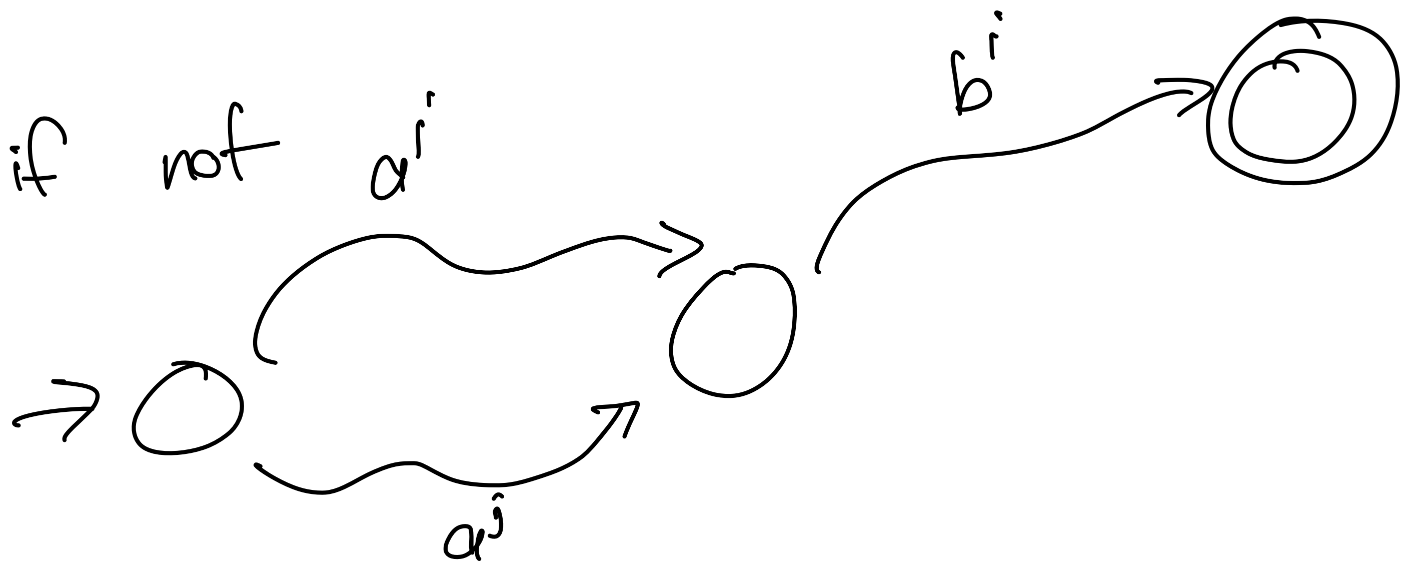
To show X is not regular, we need to show **that there does not exist** a DFA M such that $L(M) = X$.

$X = \{a^n b^n : n \in \mathbb{N}\}$ is not regular

By contradiction, suppose X is regular, then $\exists$ a DFA M s.t. $L(M) = X$.

Claim: for any $i, j \in \mathbb{N}$, s.t. $i \neq j$, let q_i, q_j be the states reached when reading a^i and a^j respectively. $q_i \neq q_j$

Proof of claim: if not



$X = \{a^n b^n : n \in \mathbb{N}\}$ is not regular

Claim: for any $i, j \in \mathbb{N}$, s.t. $i \neq j$, let q_i, q_j be the states reached when reading a^i and a^j respectively.

$q_i \neq q_j$

```
c = 0
for char in w:
    if
```

ε, a, aa, aaa, ...

each of these needs its own unique state.

$q_0 \neq q_1 \neq q_2$ .... $\Rightarrow$ need at least 1 state

for each a^i, for each $i \in \mathbb{N}$.

$\Rightarrow$ number of states is infinite which is a contradiction!

$$X = \{a^n b^n : n \in \mathbb{N}\} = \bigcup_{i \in \mathbb{N}} \left(\{a^i b^i\} \right)$$

...

can't take infinite union.

Claim $L(a^* b^*) = X$

aabbb $\quad \{2n : n \in \mathbb{N}.\}$

a, aa, aaa, $\quad$ aaaa

0 $\quad$ 0 $\quad$ 0

Key Insights

- Same state $\implies$ same fate. If two strings x, y led the DFA to the same state. **No matter what string** w **was read after**, either xw and yw both get accepted or yw both get rejected.

Key Insights

- Same state $\implies$ same fate. If two strings x, y led the DFA to the same state. **No matter what string** w **was read after**, either xw and yw both get accepted or yw both get rejected.
- The language $\{a^n b^n : n \in \mathbb{N}\}$ had infinitely many strings that do NOT share the same fate (and hence must have distinct states).

ε, a, aa, aaa, ...

"Same state same fate" but more formal

Language = Even # of 1, 1 111

Let A be any language and $x, y \in \Sigma^*$. Call x and y distinguishable relative to A if there exists w such that one of xw and yw are in A and the other is not. If x and y are not distinguishable, call them indistinguishable relative to A[2].

[2] If the language A is evident from the context, you can omit the "relative to A" part.

“Same state same fate” but more formal

Let A be any language and $x, y \in \Sigma^*$. Call x and y distinguishable relative to A if there exists w such that one of xw and yw are in A and the other is not. If x and y are not distinguishable, call them indistinguishable relative to A[2].

Lemma (Same state same fate)

Suppose M is a DFA such that $L(M) = A$, and let q_x and q_y be the states reached after reading x and y, respectively. If $q_x = q_y$, then x and y are indistinguishable relative to A.

[2] If the language A is evident from the context, you can omit the “relative to A” part.

Proof (informal)

DFAs are deterministic.

Given a state and the character read, the next state is determined.

Myhill-Nerode Theorem (corollary)

Theorem

Let A be a language over Σ. Suppose there exists a set $S \subseteq \Sigma^*$ with the following properties

- *(Infinite). S is infinite*
- *(Pairwise distinguishable). $\forall x, y \in S$, with $x \neq y$. x, and y are distinguishable relative to A.*

Then A is not regular.

Proof

Let A be a language, S is infinite & pairwise distinguishable $=\{a^i : i \in \mathbb{N}\}$.

WTS, A is not reg.

By contr. suppose A is regular w/ DFA M, s.t. $L(M) = A$.

Let $q : S \to Q$, set of states $q(x)$ is the state reached after reading x. Then claim: q is injective.

if not, $q(x) = q(y)$ for some $x \neq y$. $\Rightarrow$ x and y are indistinguishable b/c of the same state same fate lemma. but x, y are distinguishable b/c of pairwise distinguishability.

$$|S| \leq |Q|$$

infinite

set of state

$\Longrightarrow$ there are an infinite # of states.

Using The Myhill Nerode Theorem

By The Myhill-Nerode Theorem, it suffices to find a set of strings, S, such that S is infinite and pairwise distinguishable relative to A.

Proof: $X = \{a^n b^n : n \in \mathbb{N}\}$ is not regular

By the Myhill Nerode theorem, it suffices to find an infinite set of pairwise distinguishable strings. claim: $S = \{a^i : i \in \mathbb{N}\}$ works

1.) S is infinite: S has one string for each natural number, and there are an infinite number of natural numbers.

2.) S is pairwise distinguishable: Let $x, y \in S$ where $x \neq y$. $\Rightarrow$ $x = a^i$, $y = a^j$ for some $i, j \in \mathbb{N}$, $i \neq j$.

Then, $a^i b^i \in X$, $a^j b^i \notin X$, $\Rightarrow$ b^i distinguishes x, y.

$\Rightarrow$ x and y are distinguishable. ∎

An alternate proof

$$X = \{a^n b^n : n \in \mathbb{N}\}$$

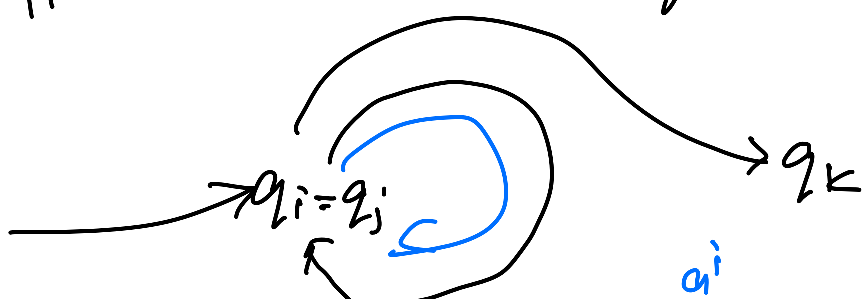
By contr. suppose regular, let M be a DFA s.t. L(M) = X. → k state

consider what happens when I read a^k

$$\underbrace{\to q_0 \xrightarrow{a} q_1 \xrightarrow{a} q_2 \cdots\cdots q_k}$$

$(q_0, \ldots q_k)$ is the sequence of states reached when reading a^k.

by the PP since there are $k+1$ states in the seq, and k states in the DFA, some state appears twice in the sequence. i.e. $\exists\, i, j$ s.t. $q_i = q_j$.



$$q_0 \xrightarrow{a^i} q_i \xrightarrow{a^{j-i}} q_i \xrightarrow{a^{k-j}} q_k$$

An alternate proof

$$\boxed{X = \{a^n b^n : n \in \mathbb{N}\}}$$

This means it reach the same state after

reading $a^i a^{j-i} a^{k-j}$ or reading $a^i a^{j-i} a^{j-i} a^{k-j}$

$$\begin{array}{ccc} a^i a^{j-i} a^{k-j} & & \\ \| & & \\ a^k & \neq & a^{k+j-i} \end{array}$$

since they end up at the same state they must be indistinguishable <u>but</u> they're not. in particular, b^k distinguishes them: ($a^k b^k \in X$, $a^{k+j-i} b^k \notin X$).

contr.

The Pumping Lemma

finite languages are regular.

$\{a_1, a_2, \ldots, a_n\}$.

$a_1 | a_2 | a_3 | \ldots | a_n$.

$a^* b^* \supseteq \{a^n b^n : n \in \mathbb{N}\}$.

not all superset of non-regular languages are non-regular.

Suppose A is a regular language. Then there exists $k \in \mathbb{N}$ such that for all $w \in A$ with $|w| \geq k$, we can write w as xyz such that

1. $|xy| \leq k$
2. $|y| > 0$
3. For $i \in \mathbb{N}$, $xy^i z \in A$.

pumping length

The Pumping Lemma explained

Suppose A is a regular language. Then there exists $k \in \mathbb{N}$ such that for all $w \in A$ with $|w| \geq k$, we can write w as xyz such that

1. $|xy| \leq k$
2. $|y| > 0$
3. For $i \in \mathbb{N}$, $xy^i z \in A$.

- Think of k as the number of states in the DFA.
- $w = xyz$. x is the part of the string that takes us to the start of the loop. y is the string that takes us in a loop. z is the remainder of the string.
- The three conditions mean the following.
  1. The loop occurs within the first k steps.
  2. The length of the looping string is non-zero. I.e., it's actually a loop.
  3. You can take the loop as many times as you like.

Using The Pumping Lemma to prove a language is not regular

Template:

By contradiction, suppose A is regular. Then, by the pumping lemma, there exists a pumping length $k \in \mathbb{N}$.

[find a string $w \in A$ with $|w| \geq k$.]
Thus, we can write $w = xyz$ satisfying the conditions of the pumping lemma.

[use conditions 1, 2 to argue something about what y looks like]

[use condition 3 to find another string in A of the form xy^iz for some $i \in \mathbb{N}$ which should actually NOT be in A.]

Example

$X = \{a^n b^n : n \in \mathbb{N}\}$

By contradiction, suppose X is reg. then $\exists$ a pumping length k,

consider $w = a^k b^k$, $|w| = 2k \geq k$ $\Rightarrow$ we can write

$w = xyz$ satisfying the conditions of the pumping lemma,

- since $|xy| \leq k$, $\Rightarrow$ $y = a^i$ for some $i \in \mathbb{N}$.

- since $|y| \neq 0$, $i > 0$.

- by condition 3, we have $xy^0z = xz \in X$

$$xz = a^{k-i} b^k \notin X$$

contradiction!

Pumping Lemma vs. Myhill Nerode for showing a language is not regular.

I prefer The Myhill Nerode Theorem - I find the arguments easier and harder to mess up.

The Pumping Lemma is an excellent backup to know.

You should try all the problems that require you to show a language is not regular using both methods and see if you develop a preference!