Theory of Automata and Languages

Finite Automata and Regular Languages - 3

Fall 2024

Sharif University of Technology

Mehran Moeini Jam

Overview

- The Myhill-Nerode Theorem
- Proof of the Myhill-Nerode Theorem
- DFA Minimization Technique
- Non-regular Languages
- Using Pumping Lemma to Prove Non-regularity
- Using Closure Properties to Prove Non-regularity
- Beyond Simple Finite Systems

Equivalence Relations

- We call a binary relation R on a set S an equivalence relation when for every a, b, c ∈ S the following three properties hold:
  - Reflexive Property: a R a ((a, a) ∈ R)
  - Symmetric Property: a R b iff b R a
  - Transitive Property: a R b and b R c then a R c
- Each equivalence relation provides a partition of the underlying set into disjoint equivalence classes. The number of classes is called the index.

Right-Invariant Equivalence Relations

- We call an equivalence relation R on a set S a right-invariant equivalence relation with respect to a binary operator X when for every a, b, c ∈ S the following holds:
  - a R b then a **X** c R **b X** c

- Now consider S to be Σ^* and X to be the concatenation operator, then the property will be as follows where a, b, c are arbitrary strings:
  - a R b then a . c R **b .** c

The Myhill-Nerode Theorem: Background

- The Myhill-Nerode Theorem is considered as one of the most fundamental results in the theory of regular languages.

Anil Nerode

John R. Myhill Sr.

The Myhill-Nerode Theorem: Background

- The Myhill-Nerode Theorem is considered as one of the most fundamental results in the theory of regular languages.

- It can be directly leveraged to solve the following problems, among many others:

1. Determining whether or not a language L is regular
2. Determining the minimal number of states for a DFA that recognizes a regular language L

The Myhill-Nerode Theorem: Background

- The Myhill-Nerode Theorem is considered as one of the most fundamental results in the theory of regular languages.

- It can be directly leveraged to solve the following problems, among many others:

1. Determining whether or not a language L is regular
2. Determining the minimal number of states for a DFA that recognizes a regular language L

The Myhill-Nerode Theorem

- These three statements are equivalent:

1. The language L is accepted by some finite automaton (L is regular)
2. There exists a right-invariant equivalence relation (with respect to the concatenation operator) of finite index that L can be defined as the union of some of its equivalence classes.
3. Let R_L be the following equivalence relation for L : $x \, R_L \, y$ iff $\forall z . \; xz \in L \Leftrightarrow yz \in L$

   R_L is of finite index.

Proving Equivalence

- In propositional calculus, considered as a formal system, the following inference rule is valid to prove equivalence between n statements, numbered P_1 to P_n :

$$
\begin{array}{l}
P_1 \rightarrow P_2 \\
P_2 \rightarrow P_3 \\
\quad \ldots \\
P_{n-1} \rightarrow P_n \\
P_n \rightarrow P_1 \\
\hline
\therefore P_1 \leftrightarrow P_2 \leftrightarrow \ldots \leftrightarrow P_n
\end{array}
$$

Proving the Myhill-Nerode Theorem – Part 1

1. The language L is accepted by some finite automaton (it is regular)

$(1) \rightarrow (2)$

2. There exists a right-invariant equivalence relation of finite index that L can be defined as the union of some of its equivalence classes

Consider M = $(Q, \Sigma, \delta, q_0, F)$ to be the DFA that accepts the language L. We construct the equivalence relation R_M as follows:

R_M: $\forall\, x, y \in \Sigma^* .\ x R_M y$ iff $\delta^*(q_0, x) = \delta^*(q_0, y)$

It can be verified that it satisfies all four conditions.

Proving the Myhill-Nerode Theorem – Part 1

R_M: $\forall$ x, y $\in \Sigma^*$. x R_M y iff $\delta^*(q_0, x) = \delta^*(q_0, y)$

It can be verified that it satisfies all the conditions.

1. The relation is an equivalence relation. It's reflexive, symmetric, transitive.
2. The relation is right-invariant. For every postfix, M starts from the same location and because of the deterministic nature, reaches the same location for both strings x and y.
3. The relation is of finite index, because the number of states is finite.

Formal Proof of Right-invariancy

R_M: $\forall\, x, y \in \Sigma^* .\ x\, R_M\, y$ iff $\delta^*(q_0, x) = \delta^*(q_0, y)$

Theorem: The relation is right-invariant.

Proof:

$$\forall\, z \in \Sigma^*.\ \delta^*(q_0, xz)$$

Formal Proof of Right-invariancy

R_M: $\forall\, x, y \in \Sigma^* .\ x\, R_M\, y$ iff $\delta^*(q_0, x) = \delta^*(q_0, y)$

Theorem: The relation is right-invariant.

Proof:

$$\forall\, z \in \Sigma^*.\ \delta^*(q_0, xz) = \delta^*(\delta^*(q_0, x), z)$$

$$\delta^*(q, \omega_1\omega_2) = \delta^*(\delta^*(q, \omega_1), \omega_2)$$

Claim, to be proved later

Formal Proof of Right-invariancy

R_M: $\forall\, x, y \in \Sigma^*$. $x\, R_M\, y$ iff $\delta^*(q_0, x) = \delta^*(q_0, y)$

Theorem: The relation is right-invariant.

Proof:

$$\forall\, z \in \Sigma^*.\ \delta^*(q_0, xz) = \delta^*(\delta^*(q_0, x), z)$$

$$= \delta^*(\delta^*(q_0, y), z)$$

Formal Proof of Right-invariancy

R_M: $\forall\, x, y \in \Sigma^* .\ x\, R_M\, y$ iff $\delta^*(q_0, x) = \delta^*(q_0, y)$

Theorem: The relation is right-invariant.

Proof:

$$\forall\, z \in \Sigma^*.\ \delta^*(q_0, xz) = \delta^*(\delta^*(q_0, x), z)$$

$$= \delta^*(\delta^*(q_0, y), z)$$

$$= \delta^*(q_0, yz)$$

$\delta^*(q, \omega_1\omega_2) = \delta^*(\delta^*(q, \omega_1), \omega_2)$

It can be concluded that $xz \quad R_M \quad yz$

Proof of the Lemma

We claimed that $\forall\ \omega_1, \omega_2 \in \Sigma^*\ ,\ q \in Q\ .\ \delta^*(q, \omega_1\omega_2) = \delta^*(\delta^*(q, \omega_1), \omega_2)$

Proof:

- Base case :

$$\delta^*(q, \omega_1\varepsilon) = \delta^*(q, \omega_1) = \delta^*(\delta^*(q, \omega_1), \varepsilon)$$

$\forall\ q \in Q\ .\ q = \delta^*(q, \varepsilon)$

Recall from lecture 1 slides

Proof of the Lemma

We claimed that $\forall\ \omega_1, \omega_2 \in \Sigma^*\ ,\ q \in Q \quad . \quad \delta^*(q, \omega_1\omega_2) = \delta^*(\delta^*(q, \omega_1), \omega_2)$

Proof:

- Inductive Step:

  $\delta^*(q, \omega_1\omega_2) = \delta^*(\delta^*(q, \omega_1), \omega_2)$ is assumed to be true, considering $|\omega_2| = n$

  We want to show that $\delta^*(q, \omega_1\omega_2 a) = \delta^*(\delta^*(q, \omega_1), \omega_2 a)$

$$\delta^*(q, \omega_1\omega_2 a) = \delta\ (\ \delta^*(q, \omega_1\omega_2), a\ ) = \delta\ (\ \delta^*(\delta^*(q, \omega_1), \omega_2), a) = \delta^*(\delta^*(q, \omega_1), \omega_2 a)$$

Proving the Myhill-Nerode Theorem – Part 1

R_M: $\forall\, x, y \in \Sigma^* .\ x\, R_M\, y$ iff $\delta^*(q_0, x) = \delta^*(q_0, y)$

Now for the forth condition:

4. The language L is the union of those equivalence classes that correspond to the final states, i.e. $L = \{\, [x] \mid \delta^*(q_0, x) \in F \,\}$

Proving the Myhill-Nerode Theorem – Part 2

2. There exists a right-invariant equivalence relation of finite index that L can be defined as the union of some of its equivalence classes

$(2) \rightarrow (3)$

3. Let R_L be the following equivalence relation, considering L :

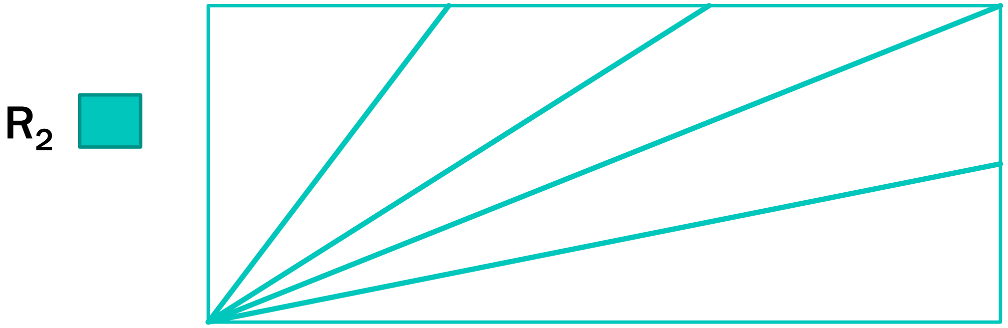
$x \, R_L \, y$ iff $\forall z \in \Sigma^* . \; xz \in L \Leftrightarrow yz \in L$

R_L is of finite index

We First Need Some More Definitions

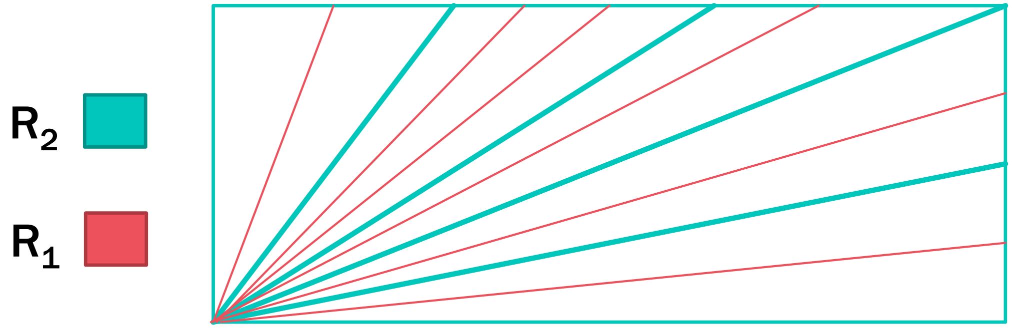
Refinement of an Equivalence Class

- We say that an equivalence relation R_1 is a refinement of an equivalence relation R_2 when every equivalence class of R_1 is entirely contained in some equivalence class of R_2



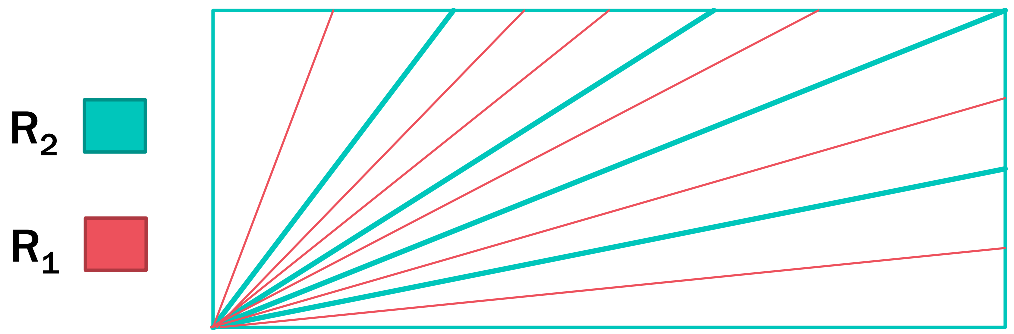
Refinement of an Equivalence Class

- We say that an equivalence relation R_1 is a refinement of an equivalence relation R_2 when every equivalence class of R_1 is entirely contained in some equivalence class of R_2



Refinement of an Equivalence Class

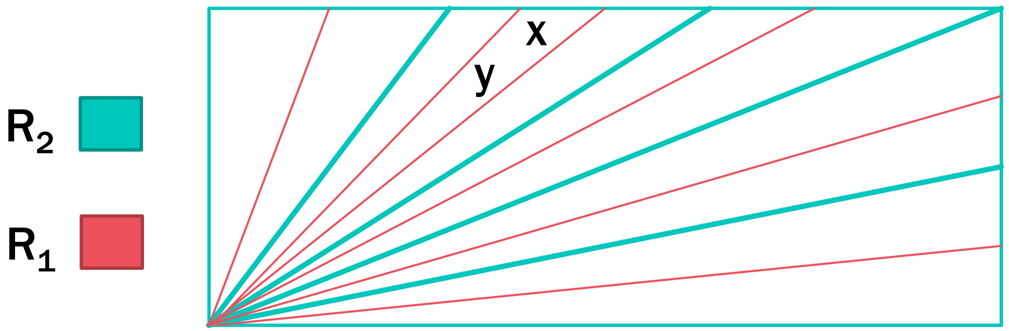
- We say that an equivalence relation R_1 is a refinement of an equivalence relation R_2 when every equivalence class of R_1 is entirely contained in some equivalence class of R_2



- It can be concluded from the above statement that the index of R_2 is less than or equal the index of R_1

Refinement of an Equivalence Class

- To show that an equivalence relation R_1 on a set S is a refinement of an equivalence relation R_2 defined on the same set, it suffices to show that for every $x,y \in S$, whenever $x\ R_1\ y$ then $x\ R_2\ y$; i.e. each equivalence class of R_1 is entirely contained in an equivalence class of R_2 .



Proving the Myhill-Nerode Theorem – Part 2

2. There exists a right-invariant equivalence relation of finite index that L can be defined as the union of some of its equivalence classes

$(2) \rightarrow (3)$

3. Let R_L be the following equivalence relation, considering L :

$x \, R_L \, y$ iff $\forall z \in \Sigma^* . \; xz \in L \Leftrightarrow yz \in L$

R_L is of finite index

Consider E to be the finite index equivalence relation in statement (2) and $x \, E \, y$ for some x and y. then since E is right invariant, for each $z \in \Sigma^*$, $xz \, E \, yz$ and as L can be defined as the union of some of its equivalence classes, $xz \in L \Leftrightarrow yz \in L$. Thus it can be concluded that $x \, R_L \, y$. So E is a refinement of R_L and because E is finite, R_L is also finite.

Proving the Myhill-Nerode Theorem – Part 3

3. Let R_L be the following equivalence relation, considering L :

$x \; R_L \; y$ iff $\forall z \in \Sigma^* . \; xz \in L \Leftrightarrow yz \in L$

R_L is of finite index

(3) → (1)

1. The language L is accepted by some finite automaton (it is regular)

We must first show that R_L is right-invariant. Suppose that $x \; R_L \; y$ for some x and y, so we know that $\forall z \in \Sigma^*$, $xz \in L \Leftrightarrow yz \in L$ is valid. Now to prove the right-invariancy, if suffices to show that $\forall z' \in \Sigma^*$, $xz' \; R_L \; yz'$ is valid. This second statement is equivalent to $\forall z'' \in \Sigma^* . \; xz'z'' \in L \Leftrightarrow yz'z'' \in L$ which is already true, considering z'z'' to be z.

Proving the Myhill-Nerode Theorem – Part 3

3. Let R_L be the following equivalence relation, considering L :

$x \; R_L \; y$ iff $\forall z \in \Sigma^* . \; xz \in L \Leftrightarrow yz \in L$

R_L is of finite index

(3) → (1)

1. The language L is accepted by some finite automaton (it is regular)

Now, we want to construct the DFA $M' = (Q', \Sigma, \delta', q_0', F')$ in order to prove the above entailment. Let Q' be the finite set of classes of R_L and for every $x \in \Sigma^*$ consider [x] to represent the equivalence class containing x, then we can define $\delta'([x], a) = [xa]$ to be the transition relation, $q_0' = [\varepsilon]$ as the initial state and $F' = \{ [x] \mid x \in L \}$ as final states.

DFA Minimization

- For every deterministic finite automaton representing a regular language L, we can define a right-invariant equivalence relation that each one of its classes corresponds to one of the states in the DFA.

- This equivalence relation was proved to be a refinement of the following equivalence relation R_L which can be used to construct a new finite automaton that **may** have a smaller number of states: $x \, R_L \, y \text{ iff } \forall z \in \Sigma^* .\ xz \in L \Leftrightarrow yz \in L$

DFA Minimization

- This equivalence relation was proved to be a refinement of the following equivalence relation R_L which can be used to construct a new finite automaton that **may** have a smaller number of states:

$$x \, R_L \, y \text{ iff } \forall z \in \Sigma^* . \; xz \in L \Leftrightarrow yz \in L$$

- The number of states in the DFA constructed from the above relation is minimal, as any other DFA defines an equivalence relation that is a refinement of this relation.

Minimum-state DFA is Unique

- We say that two DFAs are isomorphic when there exists a one-to-one correspondence between their states, such that for every two states q and q' in these two automata (respectively), $q \mapsto q'$ only if for every $a \in \Sigma$, $\delta(q, a) \mapsto \delta(q', a)$ and they are either both initial (or final) or none is.

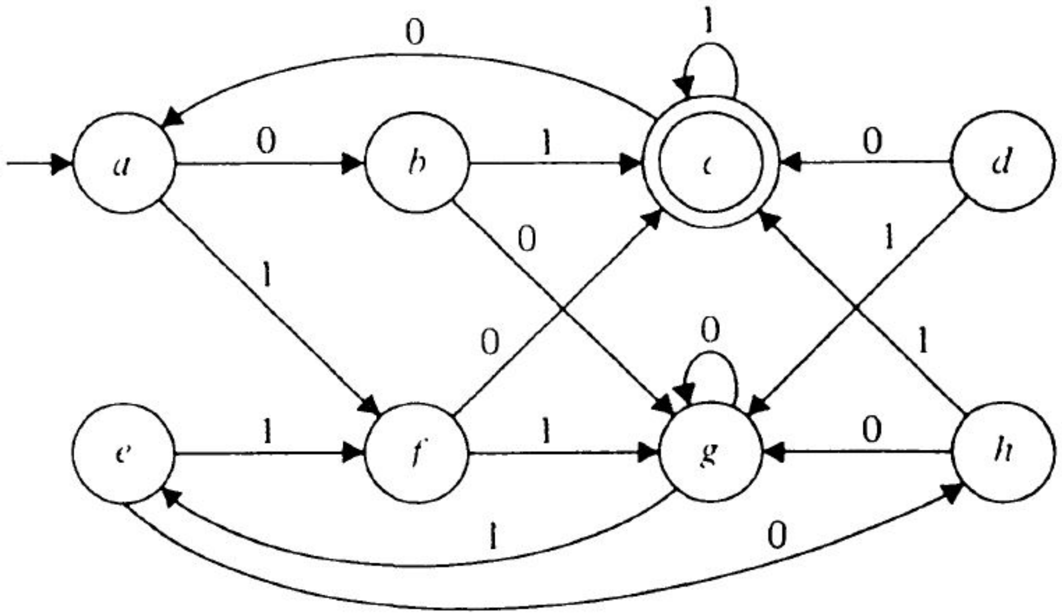
- The minimum-state DFA is unique, up to the isomorphism, i.e. every other DFA with the same number of states is isomorphic to this DFA.

DFA Minimization Algorithm

- We construct a table with an entry for each pair of states. The goal is to identify states that are indistinguishable, i.e., they represent two equivalent classes that can be merged to form the partition defined in previous slides.

- We start by marking each entry that corresponds to one final state and one non-final state. These are distinguishable states. Then we continue by marking pair of states p and q when there exist a character a that $\delta(p, a)$ and $\delta(q, a)$ are distinguishable.

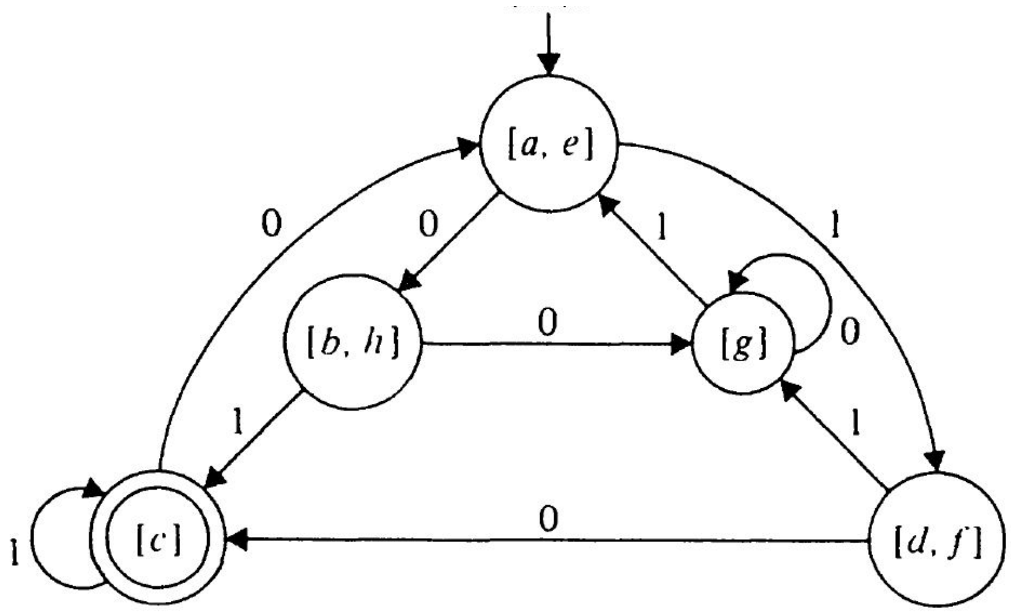
Example



	a	b	c	d	e	f	g
b	X						
c	X	X					
d	X	X	X				
e		X	X	X			
f	X	X	X		X		
g	X	X	X	X	X	X	
h	X		X	X	X	X	X

Example

	a	b	c	d	e	f	g
b	X						
c	X	X					
d	X	X	X				
e		X	X	X			
f	X	X	X		X		
g	X	X	X	X	X	X	
h	X		X	X	X	X	X



Non-regular Languages

- There are many languages that cannot be recognized by some finite automata. For now, we call these languages non-regular.

- Consider the following language

$$L = \{a^n b^n \mid n \geq 0\}$$

- There is no finite automaton that can distinguish between the strings that belong to the language and the strings that do not.

The Pumping Lemma for Regular Languages

- The Pumping Lemma for regular languages is powerful tool that can be used to prove that certain languages are not regular.
- Every regular language must satisfy the pumping lemma for regular languages. In other words, if a language doesn't satisfy the pumping lemma, it's not regular.

Statement

L is regular $\rightarrow$ L satisfies the pumping lemma

Contrapositives Statement

L doesn't satisfy the pumping lemma $\rightarrow$ L is not regular

Pitfall

- It's important to note that not every language that satisfies the pumping lemma is necessarily a regular language.

Statement	L is regular → L satisfies the pumping lemma	✓
Contrapositives Statement	L doesn't satisfy the pumping lemma → L is not regular	✓
Converse Statement	L satisfies the pumping lemma → L is regular	X

The Pumping Lemma for Regular Languages

- The formal definition is as follows.

Consider **Regular** to be the class of regular languages.

$$(\forall \mathbf{L} \in \text{Regular})\ (\exists\ n \in \mathbb{N})(\forall\ \omega \in \mathbf{L}),$$

$$(\ (\ |\omega| \geq n\ ) \Rightarrow (\ (\exists\ x, y, z \in \Sigma^*), (\omega = xyz,\ |xy| \leq n,\ |y| \geq 1, (\forall i \in \mathbb{N}), (xy^iz \in \mathbf{L})\ )\ )\ )$$

The Pumping Lemma for Regular Languages

- The formal definition is as follows.

Consider **Regular** to be the class of regular languages.

$$(\forall\ L \in \text{Regular})\ (\exists\ n \in \mathbb{N})(\forall\ \omega \in L),$$

$$(\ (\ |\omega| \geq n\ ) \Rightarrow (\ (\exists\ x, y, z \in \Sigma^*), (\omega = xyz,\ |xy| \leq n,\ |y| \geq 1, (\forall i \in \mathbb{N}), (xy^iz \in L)\ )\ )\ )$$

(1) (2) (3)

The Pumping Lemma for Regular Languages

- Consider the regular language $L = c\,(ab)^*$

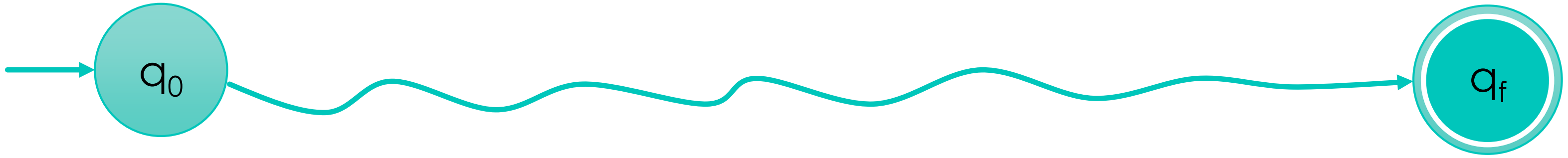
- Try to break the string cabab in a way that satisfies the conditions, assuming the length of the string is greater than the intended number n.

$$(\forall\, L \in \text{Regular})\ (\exists\, n \in \mathbb{N})(\forall\, \omega \in L),$$

$$(\,(\,|\omega| \geq n\,) \Rightarrow (\,(\exists\, x, y, z \in \Sigma^*), (\omega = xyz,\ |xy| \leq n,\ |y| \geq 1,\ (\forall i \in \mathbb{N}),\ (xy^iz \in L)\,)\,)\,)$$

(1) (2) (3)

Proof Scheme

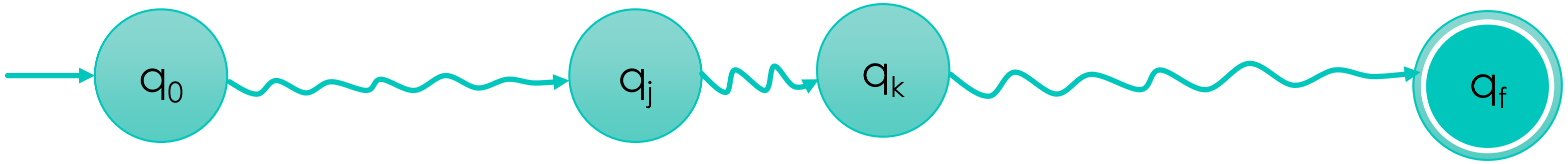


$(\forall$ **L** $\in$ **Regular**$)$ $(\exists$ **n** $\in \mathbb{N})(\forall$ **ω** $\in$ **L**$)$,

$((\ |\omega|\ \geq n\ ) \Rightarrow (\ (\exists\ x, y, z \in \Sigma^*), (\omega = xyz,\ |xy| \leq n,\ |y| \geq 1, (\forall i \in \mathbb{N}), (xy^iz \in L)\ )\ )\ )$

(1) (2) (3)

Proof Scheme

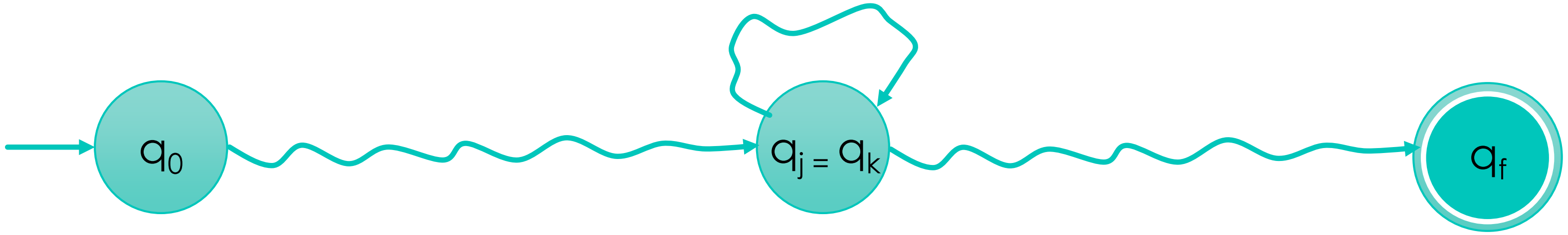


$(\forall\ L \in \text{Regular})\ (\exists\ n \in \mathbb{N})(\forall\ \omega \in L),$

$((\ |\omega| \geq n\ ) \Rightarrow (\ (\exists\ x, y, z \in \Sigma^*), (\omega = xyz,\ |xy| \leq n,\ |y| \geq 1, (\forall i \in \mathbb{N}), (xy^iz \in L)\ )\ )\ )$

(1) (2) (3)

Proof Scheme

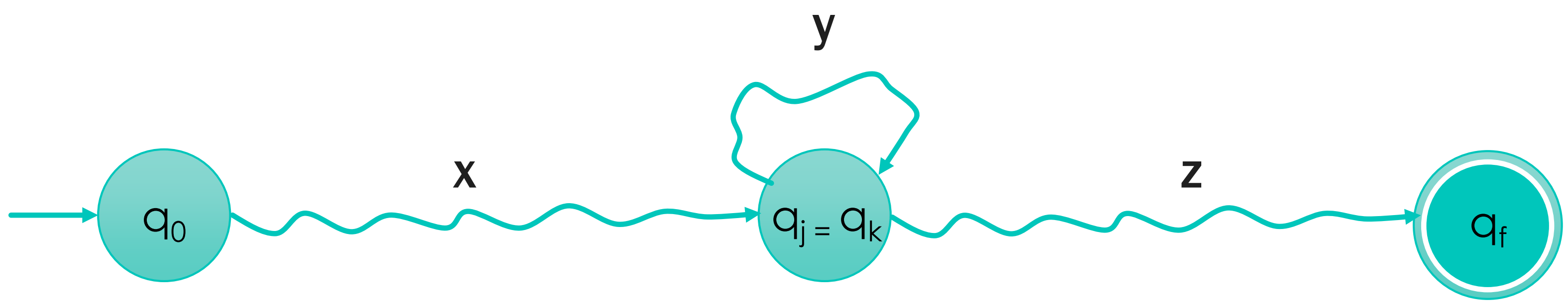


$$(\forall\ L \in \text{Regular})\ (\exists\ n \in \mathbb{N})(\forall\ \omega \in L),$$

$$(\ (\ |\omega| \geq n\ ) \Rightarrow (\ (\exists\ x, y, z \in \Sigma^*), (\omega = xyz,\ |xy| \leq n,\ |y| \geq 1, (\forall i \in \mathbb{N}), (xy^iz \in L)\ )\ )\ )$$

(1) (2) (3)

Proof Scheme



$$(\forall\ L \in \text{Regular})\ (\exists\ n \in \mathbb{N})(\forall\ \omega \in L),$$

$$(\ (\ |\omega| \geq n\ ) \Rightarrow (\ (\exists\ x, y, z \in \Sigma^*), (\omega = xyz,\ |xy| \leq n,\ |y| \geq 1,\ (\forall i \in \mathbb{N}), (xy^iz \in L)\ )\ )\ )$$

(1) (2) (3)

Using Pumping Lemma to Prove Non-regularity

1. Select the language you wish to prove non-regular (irregular).
2. The "adversary" picks the constant n.
3. Select a string ω in L, your choice may depend on the value of n.
4. The "adversary" breaks ω into x, y, z, subject to the first two constraints.
5. You achieve a contradiction by showing that there exists an $i \in \mathbb{N}$ for which $xy^iz \notin L$

$$(\forall\ L \in \text{Regular})\ (\exists\ n \in \mathbb{N})(\forall\ \omega \in L),$$

$$(\ (\ |\omega| \geq n\ ) \Rightarrow (\ (\exists\ x, y, z \in \Sigma^*), (\omega = xyz,\ |xy| \leq n,\ |y| \geq 1, (\forall i \in \mathbb{N}), (xy^iz \in L)\ )\ )\ )$$

Using Pumping Lemma to Prove Non-regularity

1. Select the language you wish to prove non-regular (irregular).
2. The “adversary” picks the constant n.
3. Select a string ω in L, your choice may depend on the value of n.
4. The “adversary” breaks ω into x, y, z, subject to the first two constraints.
5. You achieve a contradiction by showing that there exists an $i \in \mathbb{N}$ for which $xy^iz \notin L$

$$(\forall\ L \in \text{Regular})\ (\exists\ n \in \mathbb{N})(\forall\ \omega \in L),$$

$$(\ (\ |\omega| \geq n\ ) \Rightarrow (\ (\exists\ x, y, z \in \Sigma^*), (\omega = xyz,\ |xy| \leq n,\ |y| \geq 1, (\forall i \in \mathbb{N}), (xy^iz \in L)\ )\ )\ )$$

Pitfall

1. Select the language you wish to prove non-regular (irregular).
2. The “adversary” picks the constant n.
3. Select a string ω in L, your choice may depend on the value of n.
4. The “adversary” breaks ω into x, y, z, subject to the first two constraints.
5. You achieve a contradiction by showing that there exists an $i \in \mathbb{N}$ for which $xy^iz \notin L$

You should consider all possible such breaks

$$(\forall L \in Regular)\ (\exists n \in \mathbb{N})(\forall \omega \in L),$$

$$((\ |\omega| \geq n\ ) \Rightarrow (\ (\exists\ x, y, z \in \Sigma^*), (\omega = xyz,\ |xy| \leq n,\ |y| \geq 1, (\forall i \in \mathbb{N}), (xy^iz \in L)\ )\ )\ )$$

The Winning Strategy

- Your objective in the game is to reach a contradiction, which your opponent will do everything in their power to prevent.

- Proving that a language is non-regular by pumping lemma is equivalent to identifying a winning strategy in the described game — a strategy that guarantees victory regardless of how "the adversary" plays.

Example

- $L = \{a^k b^k \mid k \geq 0\}$

- Consider $\omega = a^n b^n$ as the string, then $|\omega| = 2n$

- Let $i = 2$, the new string ω' becomes xy^2z for any given x, y and z

- So he string is equal to $a^n a^{|y|} b^n$ where $1 \leq |y| \leq n$ ※

$$(\forall L \in Regular)(\exists n \in \mathbb{N})(\forall \omega \in L),$$

$$((|\omega| \geq n) \Rightarrow ((\exists x, y, z \in \Sigma^*), (\omega = xyz, |xy| \leq n, |y| \geq 1, (\forall i \in \mathbb{N}), (xy^iz \in L))))$$

Example

- $L = \{0^{k^2} \mid k \geq 1\}$

- Consider $\omega = 0^{n^2}$ as the string, then $|\omega| = n^2$

- Let $i = 2$, the new string ω' becomes xy^2z for any given x, y and z

- $n^2 < n^2 + 1 \leq |\omega'| \leq n^2 + n < (n + 1)^2$ ※

$$(\forall L \in \text{Regular})(\exists n \in \mathbb{N})(\forall \omega \in L),$$

$$((|\omega| \geq n) \Rightarrow ((\exists x, y, z \in \Sigma^*), (\omega = xyz, |xy| \leq n, |y| \geq 1, (\forall i \in \mathbb{N}), (xy^iz \in L))))$$

Example

- $L = \{a^k b a^{2k} \mid k \geq 0\}$

- Consider $\omega = a^n b a^{2n}$ as the string, then $|\omega| = 3n + 1$

- Let $i \geq 2$, the new string ω' becomes xy^iz for any given x, y and z

- Here, y must be part of the n first characters, so the new string is $a^n a^{(i-1)|y|} b a^{2n}$ ※

$$(\forall L \in \text{Regular})(\exists n \in \mathbb{N})(\forall \omega \in L),$$

$$((|\omega| \geq n) \Rightarrow ((\exists x, y, z \in \Sigma^*), (\omega = xyz, |xy| \leq n, |y| \geq 1, (\forall i \in \mathbb{N}), (xy^iz \in L))))$$

Example

- $L = \{a^p \mid p \text{ is a prime number}\}$
- Consider $\omega = a^{n'}$ as the string, where n' is the first prime number greater than n
- Let $i = n' + 1$, the new string ω' becomes $xy^{n'+1}z$ for any given x, y and z
- We can move the new a's in ω' to the end, so the string is equal to $xyzy^{n'} = a^{n'}a^{|y|n'}$
- It can be concluded that $\omega' = a^{n'(|y|+1)}$ ※

$$(\forall L \in Regular)(\exists n \in \mathbb{N})(\forall \omega \in L),$$

$$((|\omega| \geq n) \Rightarrow ((\exists x, y, z \in \Sigma^*), (\omega = xyz, |xy| \leq n, |y| \geq 1, (\forall i \in \mathbb{N}), (xy^iz \in L))))$$

Using Closure Properties to Prove Non-regularity

- Consider the following language

  L = { r ∈ {a, b}* | r has equal number of a’s and b’s}

- We are aware that the class of regular languages is closed under intersection. So if L is regular, then L’ = L ∩ a*b* must also be regular.

- The language L’ is { a^kb^k | k ≥ 0 } which was previously proved to be non-regular ※

Beyond Simple Finite Systems

- Cellular Automata
- Probabilistic Automata
- Markov Chains
- Timed Automata
- Quantum Finite Automata