Fooling Sets and Introduction to Non-deterministic Finite Automata

Lecture 5
Proving that a language is not regular

• Given a language, we saw how to prove it is regular (union, intersection, concatenation, complement, reversal…)

• How to prove it is not regular?
Proving that a language is not regular

• Pick your favorite language $L$ (= let $L$ be an arbitrary language)

• For any strings $x,y$ ($x,y$ not necessarily in $L$) we define the following equivalence:

$$x \equiv_L y$$

• Means for EVERY string $z \in \Sigma^*$ we have

$$xz \in L \text{ if and only if } yz \in L$$
Proving that a language is not regular

• Conversely,

\[ x \not\in_L y \]

• Means for SOME string \( z \in \Sigma^* \) we have

either \( xz \in L \) and \( yz \in L \)

or \( xz \notin L \) and \( yz \in L \)

We say \( z \) distinguishes \( x \) from \( y \) (take \( z \), glue it to \( x \) and see what belongs to \( L \))
Example

- Pick your favorite language
- e.g. $L = \{\text{strings with even zeroes and odd ones}\}$
- Pick $x = 0011$ and $y = 01$. None of them in $L$!
- Can we find distinguishing suffix $z$?

$z = 1$:
- $xz = 00111 \text{ in } L$
- $yz = 011 \text{ not in } L$

$z = 0$:
- $xz = 00110 \text{ not in } L$
- $yz = 010 \text{ in } L$

$z = \varepsilon$:
- $xz = 0011 \text{ not in } L$
- $yz = 01 \text{ not in } L$
Example

- $L = \{\text{strings with even zeroes and odd ones}\}$
- Pick $x = 0011$ and $y = 01$. None of them in $L$!
- Can we find distinguishing suffix $z$?

- $z = 1$: $xz = 00111$ in $L$, $yz = 011$ not in $L$
- $z = 0$: $xz = 00110$ not in $L$, $yz = 010$ in $L$
- $z = \varepsilon$: $xz = 0011$ not in $L$, $yz = 01$ not in $L$

Bad choice for $z$!
Why do I care?

• I can learn something about the equivalence relation by looking at every DFA that accepts L.

• Assume that after the DFA reads x and y it ends up at the same state:

\[ \delta^*(s, x) = \delta^*(s, y) \Rightarrow x \equiv_L y \]

Proof: For any z,

\[ \delta^*(s, xz) = \delta^*(s, yz) \Rightarrow \delta^*(s, xz) \in A \Leftrightarrow \delta^*(s, yz) \in A \]
Why do I care?

- This implication can be turned around:

\[ x \not\equiv y \Rightarrow \delta^*(s, x) \neq \delta^*(s, y) \]

In ANY DFA for L

\[ \Rightarrow |Q| \geq 2 \]

- For the example before, we found two strings not equivalent. Any DFA for the language has AT LEAST two distinct states!
- Kind of trivial, cause what DFA has only one state?
Why do I care?

• Pushing it further:

If we can find $k$ strings $x_1, \ldots, x_k$ such that

$$x_i \neq x_j \quad \forall i \neq j$$

Then, any DFA for $L$ has at least $k$ states

A way of formally proving how “complicated” a language is if it is regular
Our Example

• $L = \{\text{strings with even zeroes and odd ones}\}$

$x_1=00$

$x_2=01$

$x_3=001$

$x_4=000$
Our Example

- $L = \{\text{strings with even zeroes and odd ones}\}$

- $x_1 = 00$

- $x_2 = 01$

- $x_3 = 001$

- $x_4 = 000$

- $x_1 z = 0001$ not in $L$

- $x_2 z = 00001$ in $L$
Our Example

• $L = \{\text{strings with even zeroes and odd ones}\}$

$x_1=00$
$x_2=01$
$x_3=001$
$x_4=000$

$z=?$
Our Example

- $L = \{\text{strings with even zeroes and odd ones}\}$

$x_1 = 00$
$x_2 = 01$
$x_3 = 001$
$x_4 = 000$

$z = 1$
Our Example

- $L = \{\text{strings with even zeroes and odd ones}\}$

Any DFA for $L$ has AT LEAST 4 states!

What is a DFA for $L$?

$x_1 = 00$

$x_2 = 01$

$x_3 = 001$

$x_4 = 000$
Our Example

• $L = \{\text{strings with even zeroes and odd ones}\}$

$x_1=00$
$x_2=01$
$x_3=001$
$x_4=000$

We proved that this (obvious) DFA is the minimal one!!!
Our Example

- \( L = \{ \text{strings with even zeroes and odd ones} \} \)

- Fooling set.
Proving that a language is not regular

• Suppose I can find an infinite fooling set for L.

• Infinite set of strings \{x_1, x_2, \ldots\} such that

\[ x_i \neq x_j \quad \forall i \neq j \]

• Then every DFA for L has at least infinite number of distinct states

• L not regular!
Proving that a language is not regular

• Example: \( L = \{0^n1^n \mid n \geq 0 \} = \{\varepsilon, 01, 0011, \ldots\} \)

• Claim: This is a fooling set: \( F = \{0^n \mid n \geq 0 \} \)

Proof: Let \( x, y \) two arbitrary different strings in \( F \).

Therefore \( x \neq y \).
Proving that a language is not regular

- Example: \( L = \{0^n1^n \mid n \geq 0 \} = \{\varepsilon, 01, 0011, \ldots\} \)

- Claim: This is a fooling set: \( F = \{0^n \mid n \geq 0 \} \)

Proof: Let \( x, y \) two arbitrary different strings in \( F \).

\[
x = 0^i \text{ for some integer } i \\
y = 0^j \text{ for some different integer } j \\
z = 1^i
\]

Therefore \( x \not\equiv y \).
Proving that a language is not regular

• Example: \( L = \{0^n1^n \mid n \geq 0 \} = \{\varepsilon, 01, 0011, \ldots \} \)

• Claim: This is a fooling set: \( F = \{0^n \mid n \geq 0 \} \)

Proof: Let \( x, y \) two arbitrary different strings in \( F \).

\[
\begin{align*}
x &= 0^i \text{ for some integer } i \\
y &= 0^j \text{ for some different integer } j \\
z &= 1^i
\end{align*}
\]

\( xz = 0^i \ 1^i \) in \( L \)

\( yz = 0^j \ 1^i \) not in \( L \)

Therefore \( x \neq y \).
Proving that a language is not regular

- To prove that $L$ is not Regular:
  - Find some infinite set $F$
  - Prove for any two strings $x$ and $y$ in $F$ there is a string $z$ such that $xz$ is in $L$ XOR $yz$ is in $L$.

- How to come up with those fooling sets?

- Be clever :)

- Think of what information you have to keep track of in a DFA for $L$. 
What to keep track of?

• Example: \( L = \{0^n1^n\} = \{\varepsilon, 01, 0011, \ldots\} \)

• Is a string in \( L \)? What do I have to keep track of?

• I need to keep track of the number of zeroes.

• So, every **number of zeroes** is intuitively a different state (different equivalence class).

• Fooling set is a set of strings that exercises all possible values that I need to keep track in my head.

• Sometimes easier to narrow it down.
What to keep track of?

- Another Example: \( L = \{ww^R|w \in \Sigma^* \} \) = even length palindromes

- What is a fooling set?

- I have to remember the whole string \( w \).

**Attempt 1:**

\[
F = \Sigma^* \\
x = 0000 \\
y = 00
\]

**Attempt 2:**

\[
F = \{?\}
\]
What to keep track of?

• Another Example: \( L = \{ww^R \mid w \in \Sigma^* \} \) = even length palindromes

• What is a fooling set?

• I have to remember the whole string \( w \).

**Attempt 1:**

\[
F = \Sigma^* \\
x = 0000 \\
y = 00
\]

**Attempt 2:**

\[
F = 0^*1 \\
x = 0111 \\
y = 0011
\]
What to keep track of?

- Another Example: $L = \{ww^R | w \in \Sigma^* \} = \text{even length palindromes}$

- What is a fooling set?

- I have to remember the whole string $w$.

  $F = 0^*1$

  $x = 0^i1$

  $y = 0^j1$

  What $z$ (exercise)?
What to keep track of?

• Another Example: \( L = \{ w w^R \mid w \in \Sigma^* \} = \text{even length palindromes} \)

• What is a fooling set?

• I have to remember the whole string \( w \).

\[
F = 0^*1 \\
x = 0^i1 \\
y = 0^i1 \\
z = 10^i
\]
What to keep track of?

- Another Example: \( L = \{ w | w = w^R \} \) = all palindromes

- What is a fooling set?

\[
F = 0^*1
\]
\[
x = 0^i1
\]
\[
y = 0^j1
\]
\[
z = 10^i
\]
What to keep track of?

- Another Example: \( L = \{ w \mid w = w^R \} = \text{all palindromes} \)
- What is a fooling set : SAME!

\[
F = 0^*1 \\
x = 0^11 \\
y = 0^11 \\
z = 10^i
\]
What to keep track of?

• Another Example: \( L = \{ w | w = w^R \} = \) all palindromes over the alphabet \( \{0,1,a,b,c,d,e,f\} \)

• What is a fooling set : SAME!

\[
F = 0^*1 \\
x = 0^i1 \\
y = 0^j1 \\
z = 10^i
\]
Proving that a language is not regular

Language is regular if and only if there is no infinite fooling set.
Nondeterminism

• Aka Magic.
Tracking Computation

A computation’s configuration evolves in each time-step on input 1010.
Deterministic Computation

Deterministic: Each step is fully determined by the configuration of the previous step and the transition function. If you do it again, exactly the same thing will happen.
Nondeterminism

• Determinism: opposite of free will

• Nondeterminism: you suddenly have choices!
Non-Deterministic FA

What can be non-deterministic about an FA?

1. At a given state, on a given input, a set of "next-states" set could be empty, could be all states…

What language?
NFA : Formally

DFA : \( M = (\Sigma, Q, \delta, s, A) \)

- \( \Sigma \): alphabet
- \( Q \): state space
- \( s \): start state
- \( A \): set of accepting states

\[
\delta : Q \times \Sigma \rightarrow Q
\]

\[
\delta(q, a) = \text{a state}
\]

NFA : \( N = (\Sigma, Q, \delta, s, A) \)

\[
\delta : Q \times \Sigma \rightarrow 2^Q = \mathcal{P}(Q)
\]

\[
\delta(q, a) = \{ \text{a set of states} \}
\]
• Input = 1001

• $L = \{\text{contains either 00 or 11}\}$
One of the states are accepting. There needs to be AT LEAST one accepting state
Nondeterminism

• What is non determinism?
• Magic?
• Parallelism?
• Advice?
Nondeterminism

• What is nondeterminism?

• Suppose I wanted to prove to you that the string 1001 is in L = \{contains either 00 or 11\}

• We built a DFA with product last time.

• Proof is an accepting computation
NFA
Nondeterminism

- What is non determinism?

- Suppose I wanted to prove to you that the string 1001 is in \( L = \{ \text{contains either 00 or 11} \} \)

- We built a DFA with product last time.

- Proof is an accepting computation: guide for the reader to how to follow the steps to a given conclusion.
Nondeterminism

• P vs. NP

• Are they the same?

• Easier to give the proof than come up with the proof! (?)
Non-determinism

• For FSM, non-determinism does not give you more expressive power!

• Any language that can be accepted by an NFAs can also be accepted by a DFA.

• It is more efficient, last example had 4 states but product construction had 8!
DFA for $L = \{w: w$ contains 00 or 11$\}$
NFA for $L = \{w: w \text{ contains } 00 \text{ or } 11\}$
NFA : More efficient

Design an NFA to recognize
\[ L(M) = \{ w \mid w : 7\text{th character from the end is a } 1 \} \]

- Minimum DFA for this language would have \( 2^7 \) states at least!
- need to remember the last 7 symbols.
NFA : Formally

- NFA has 5 parts, similar to a DFA: \( N = (\Sigma, Q, \delta, s, A) \)
  - \( \Sigma \): alphabet
  - \( Q \): state space
  - \( s \): start state
  - \( A \): set of accepting states
  - \( \delta : Q \times \Sigma \rightarrow \mathcal{P}(Q) = 2^Q \) transition function

- Define extended transition function:
  \( \delta^* : Q \times \Sigma \rightarrow \mathcal{P}(Q) = 2^Q \)
  
  \[ \delta^*(q, w) = \]
  - \( \ldots \ldots \) if \( w = \varepsilon \)
  - \( \ldots \ldots \) if \( w = ax \)
NFA: Formally

- NFA has 5 parts, similar to a DFA: \( N = (\Sigma, Q, \delta, s, A) \)
  - \( \Sigma \): alphabet
  - \( Q \): state space
  - \( s \): start state
  - \( A \): set of accepting states
  - \( \delta \): \( Q \times \Sigma \rightarrow 2^Q \) transition function

- Define extended transition function:

\[
\delta^*: Q \times \Sigma^* \rightarrow 2^Q
\]

\[
\delta^*(q, w) = \begin{cases} 
\{q\} & \text{if } w = \varepsilon \\
\bigcup_{p \in \delta(q, a)} \delta^*(p, x) & \text{if } w = ax
\end{cases}
\]
NFA : When does it accept?

NFA accepts a string $w$ if and only if

$$\delta^*(s, w) \cap A \neq \emptyset$$