

Lecture 5: Announcements later

Small red text in the top right corner, likely a page number or document identifier.

how to use it to
prove lang non-regular

Last lecture: Myhill - Nerode

All in lecture notes on the webpage!

Def: for a language $L \subseteq \Sigma^*$, we say x, y are distinguishable (for L) if there exists $z \in \Sigma^*$ such that exactly one of xz or $yz \in L$

x and y are indistinguishable if for all z we have $xz \in L \Leftrightarrow yz \in L$.

$\sim_L$ is the equivalence relation where $x \sim_L y$ if and only if x and y are indistinguishable for L

Ex: $L = \{a^n \mid n \text{ is even}\}$
 $x = aa, y = aaaa$
 $x \sim_L y$

$\sim_L$ partitions Σ^* into equivalence classes,

where each class $[x] = \{y \mid y \sim_L x\}$

$[x] = [y]$ iff $x \sim_L y$

Lemma 2: Let $L \subseteq \Sigma^*$ be a language. Suppose that $\exists$ a set $S = \{x_1, \dots, x_k\}$ ($|S| = k$) such that for all $i \neq j$ we have x_i and x_j are distinguishable for L .

Then there is no DFA with $< k$ states that recognizes L .

Myhill - Nerode theorem

A language L is regular iff $\sim_L$ has finitely many equivalence classes

Moreover if L is regular, the # of equivalence classes in $\sim_L$ is the minimal # of states in a DFA for L

Proof: (sketch, see lecture notes)

(1) If L is regular, then $\sim_L$ has finitely many equivalence classes

We prove the contra positive: Suppose $\sim_L$ has ∞ many equivalence classes. Then there exists a set $S = \{x_1, x_2, \dots\}$ which has ∞ many strings and for every $x_i, x_j \in S$ x_i and x_j are distinguishable for L . (take 1 string per equivalence class)

So for any number k , there can't exist a DFA with k states that recognizes L . So L isn't regular

↳ by Lemma 2

(2) If $\sim_L$ has finitely many equivalence classes, then L is regular.

We construct a DFA for L .

Recall that $[x]$ is the equivalence class of x ; $[x] = \{y \mid x \sim_L y\}$

$D = (Q, \Sigma, \delta, q_0, F)$ where

* $Q = \{[x] \mid x \in \Sigma^*\}$: one state per equivalence class
(hence Q is finite)

Idea: after reading x : the current state is $[x]$

* $q_0 = [\epsilon]$

$x_1 x_2 \dots x_n \Rightarrow [\epsilon] \xrightarrow{x_1} [x_1] \xrightarrow{x_2} [x_1 x_2] \downarrow \vdots$

* $\delta([x], a) = [xa]$



well defined?

need to show: if $[x] = [y]$ then $[xa] = [ya]$

* $F = \{[x] \mid x \in L\}$ make accept states equivalence classes where strings are in L

well defined?

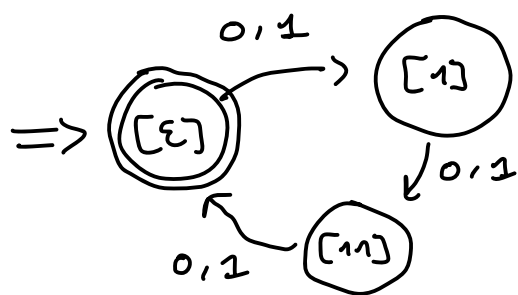
need to prove that if $[x] = [y]$ then $x \in L \Leftrightarrow y \in L$

Since x ends in state $[x]$, D accepts $\Leftrightarrow x \in L$
 $[x] \in F$

Example: $L = \{x \mid |x| \equiv 0 \pmod 3\}$ $\epsilon \in L$

We saw the equivalence classes are $[\epsilon] = \{x \mid |x| \equiv 0 \pmod 3\}$

$[1] = [0] = \{x \mid |x| \equiv 1 \pmod 3\}$
(so $0 \notin L$)

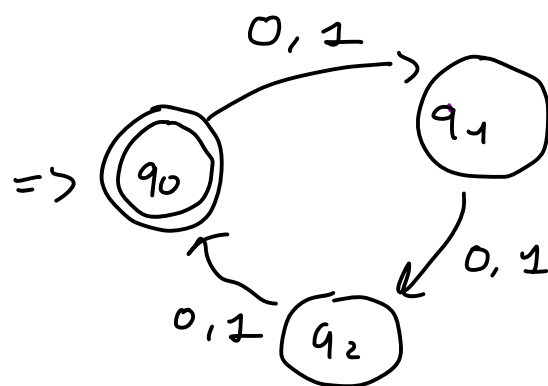


$$[00] = \{x \mid |x| \equiv 2 \pmod 3\}$$

$\uparrow$ $00 \notin 2$

why? $[10] = [11]$

Same as the DFA we saw $\Rightarrow$



(3) L is regular $\Rightarrow$ # of states in minimal DFA recognizing L is exactly the # of equivalence classes

The above DFA has # of states = to # of equivalence classes

we had a state $[x]$ for each equivalence class

We also know we can't have a DFA with an # of states strictly smaller than # of equivalence classes (easy to prove using lemma 2)

If you have k equivalence classes, there is

a set $S = \{x_1, \dots, x_k\}$ where each x_i, x_j are distinguishable.

How to use MN?

* come up with a set S such that.

- S has ∞ many strings

- $\forall x, y \in S$ where $x \neq y$

show x and y are distinguishable

- conclude $\sim_L$ has ∞ many equivalence classes. So by MN, L is not regular

Example: $L = \{w \mid w \text{ has the same number of } a\text{'s and } b\text{'s}\} \quad (\Sigma = \{a, b\})$

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

Let $x, y \in S$. $x = a^n b^n$, $y = a^m b^m$ $n \neq m$

Take $z = a^n b^n$. Then $xz = a^n b^n a^n b^n \in L$

Then $yz = \underbrace{a^m b^m a^n b^n}_{\substack{\in L \quad n+m \text{ } a\text{'s and } b\text{'s}}}$: since $\therefore$ this doesn't work

$$S = \{a^n b \mid n \geq 1\} \quad / \quad S = \{a^n \mid n \geq 0\}$$

Let $x, y \in S$. $x = a^n b$, $y = a^m b$ $n \neq m$

Take $z = b^{n-1}$. Then $xz = a^n b b^{n-1} \in L$

Then $yz = a^m b b^{n-1} \notin L$

because yz has m a 's and n b 's but $n \neq m$.

Another proof: Use closure properties

We know $L' = \{a^n b^n \mid n \geq 0\}$ isn't regular

We know $L'' = \{a^i b^j \mid i, j \geq 0\}$ is regular

$L = \{w \mid w \text{ has the same \# of } a\text{'s as \# of } b\text{'s}\}$

$L'' \cap L = \{a^i b^j \mid i = j\} = \{a^n b^n \mid n \geq 0\} = L'$

Assume for contradiction that L is regular.

by closure under $\cap$, we have $L'' \cap L = L'$ is regular

That's a contradiction! L isn't regular

Example: if $L_1 \cap L_2$ is regular then L_1, L_2 are regular

Ex: $L_2 = \{a^{2^n} \mid n \geq 0\}$ ex: $\{a, aa, aaaa \dots\}$
 $\Sigma = \{a\}$ $\uparrow$
power of 2

Let $S = \{a^{2^n} \mid n \geq 0\} = L_2$. S is infinite.

Let $x = a^{2^i}$, $y = a^{2^j}$ where $i \neq j$.

We must find a distinguishing extension of x, y :

Let $z = a^{2^i}$. $xz = a^{2^i} a^{2^i} = a^{2^i + 2^i} = a^{2^{i+1}}$
so $xz \in L$
a power of 2

$yz = a^{2^j} a^{2^i} = a^{2^j + 2^i}$ but $2^i + 2^j$ can't be a power of 2 when $i \neq j$. Assume $i < j$. Then

$$2^i + 2^j = 2^i (1 + 2^{j-i}) \Rightarrow \text{odd because } 2^{j-i} \text{ is even } (i \neq j)$$

So $y \notin L$.

So $\sim_{L_2}$ has ∞ many equivalence classes. So L_2 is not regular.

Challenge: $L = \{w \in \{0,1\}^* \mid w = xy, x \text{ has an even \# of 1s}$
 $y \text{ has an even \# of 0s}$
 $|x| = |y|\}$

We know $L' = \{a^n b^n \mid n \geq 0\}$ is not regular

We know $L'' = \{a^i b^j \mid i, j \geq 0\}$ is regular

$$L'' \cap L = \{a^i b^j \mid i=j\} = L'$$

So if L was regular, by closure under $\cap$ we'd have L' is regular. So L is not regular.

$$L_2 = \{a^{2^n} \mid n \geq 0\} \quad \text{ex: } \{a, aa, aaaa, \dots\}$$

↑
power of 2
many a's

Take $S = \{a^{2^n} \mid n \geq 0\}$. S is infinite.

Let $x = a^{2^i}$, $y = a^{2^j}$ ($i > j$) in S . We show they are distinguishable. Let $z = a^{2^i}$ then

$$xz = a^{2^i} a^{2^j} = a^{2^i + 2^j} = a^{2^{i+1}} \in S$$

$$yz = a^{2^j} a^{2^i} = a^{2^j + 2^i} \quad \text{but for any } i, j > 0$$

$2^i + 2^j = 2^i(1 + 2^{i-j})$ ← power of 2 * odd number.

$2^i + 2^j$ is a power of 2 iff $i=j$

So $y \notin L$

Challenge: $L = \{w \mid w = xy, |x| = |y| \text{ and}$

x has even # of 1's

y has even # of 0's }

End of material for HW1 + Quiz 1

Announcements:

- * This is the end of material for HW 1 and Quiz 1
 - only material so far (class, handout, checkup, etc.) can be used on HW 1 and quiz 1

NOT techniques we will study next,
nor things from textbook that we didn't cover (eg no NFAs or regular expression)

- * HW 1 : due Mon 11:59 pm

- see webpage for policies:

lateness, collaboration, AI policy
not allowed!

- * check webpage again (many updates)

- * checkup 1 => last attempt due tonight

- * checkup 2 => first attempt due Thursday 11:59pm

last attempt : next Thursday

* Handout 1: coming out soon

↳ examples + solutions

* Quiz 1: October 6th in class

- come at time you're registered for
(unless DS / cards / special agreement)

* ~ 30 minutes: closed book, closed notes, ...

* Quiz 1 content:

very much

like

HW1

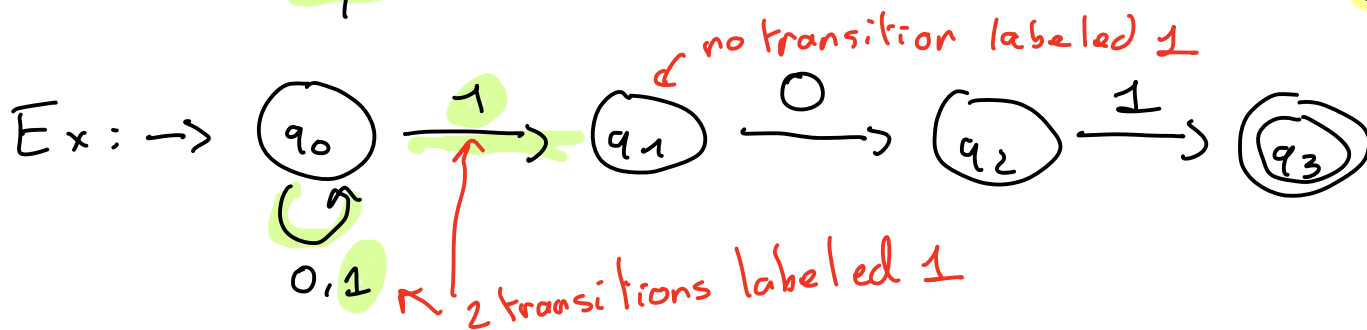
- Here's a DFA, describe its language
- Here's a language, construct a DFA
- Here's a language, prove non-regular

- Some short questions with no justification

NFA: nondeterministic finite automaton : our 2nd model of computation

Informally: NFAs are like DFAs but we can have

0 or more transitions for each state and symbol. Some transitions are labelled ϵ .



On a string w : an NFA can have 0 or more computations

Ex: On input 101 we have the following:

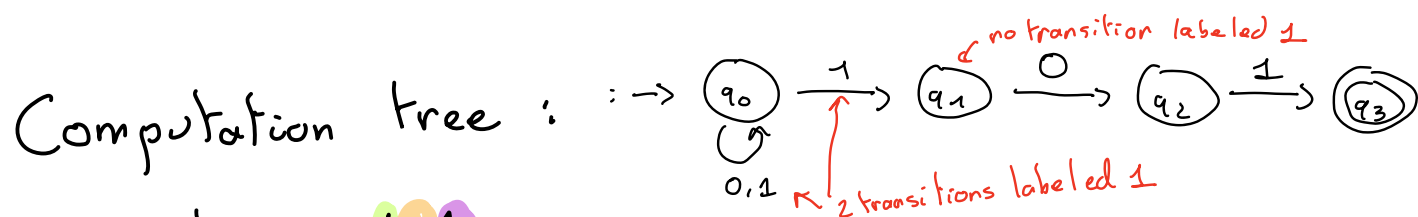
$\rightarrow q_0 \xrightarrow{1} q_0 \xrightarrow{0} q_0 \xrightarrow{1} q_0$) doesn't end in accept
 $\rightarrow q_0 \xrightarrow{1} q_1 \xrightarrow{0} q_2 \xrightarrow{1} q_3$) ends in accept state
 $\rightarrow q_0 \xrightarrow{1} q_0 \xrightarrow{0} q_0 \xrightarrow{1} q_1$) doesn't end in accept

An NFA accepts a string w if at least one computation ends in an accept state. So 101 is accepted by the above NFA.

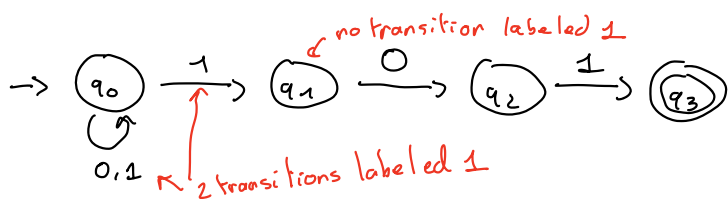
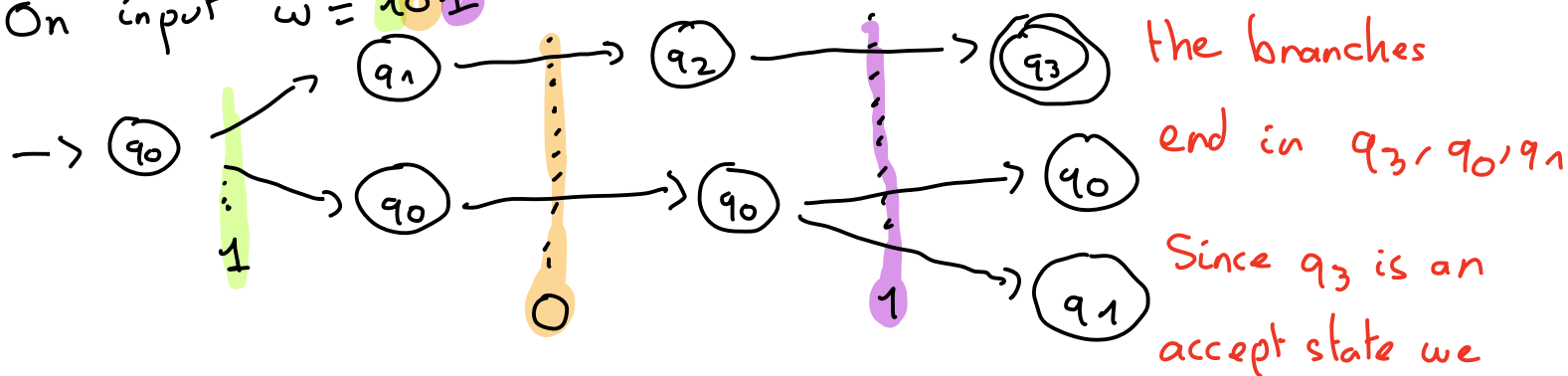
2 views of computation:

- try all possible computations in parallel, accept if at least one ends in an accept state.

- when there is more than 1 option, you can choose (and hope for the best). If some choices can lead to accept, then the string is accepted



On input $w = 101$



10110 is not accepted : the branches end in q_0, q_2 and these are not accept states.

What's the language of the NFA?

$$L(N) = \{w \in \{0,1\}^* \mid w \text{ ends in } 101\}$$

