

(Announcements: a little later in class.)

Last time: MN theorem and how to use it to prove lang non-regular

today: will go over proof and a couple more examples

(all in lecture notes we uploaded)

Recap:

Let $L \subseteq \Sigma^*$ be any language, $x, y \in \Sigma^*$ be strings.

x, y are distinguishable by L if $\exists z$ s.t. exactly one of $xz, yz \in L$

x, y are indistinguishable by L if $\forall z \quad xz \in L \Leftrightarrow yz \in L$
($x \sim_L y$) (both in L or both $\notin L$)

$\sim_L$ is an equivalence relation: partitions Σ^* into equivalence classes, where each class is $[x] = \{y \mid x \sim_L y\}$
($[x] = [y]$ iff $x \sim_L y$)

Lemma 1: Let $L \subseteq \Sigma^*$ be a regular language, and let D be a DFA for L . If x, y are distinguishable by L , then the computations of D on x and on y must end in different states.

$\Rightarrow$ Lemma 2: Let $L \subseteq \Sigma^*$ be any language, $k > 0$ integer.

Suppose $\exists$ a set of k distinct strings $S = \{x_1, \dots, x_k\}$ s.t. $\forall x_i, x_j \in S$ where $i \neq j$, x_i, x_j are distinguishable by L .

Then there does not exist a DFA for L with $\leq k$ states

Myhill-Nerode theorem:

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

Moreover, if L is regular, the number of equivalence classes is the number of states in the minimal DFA for L

Next: sketch the proof (using Lemma 2) and afterwards recap how to use MN to prove non regularity and show examples.

proof: (1) L regular $\Rightarrow \simeq_L$ has finitely many equivalence classes:

Suppose $\simeq_L$ had infinitely many equivalence classes.

Then there is a set $S = \{x_1, x_2, \dots\}$ with infinitely many strings, such that for every $x_i, x_j \in S$ ($i \neq j$), x_i, x_j are distinguishable by L .

(just take one from each equivalence class)

Thus, for any integer k , there does not exist a DFA for L with $\leq k$ states (by Lemma 2) $\Rightarrow L$ non-regular.

(2) $\sim$ has finitely many equivalence classes $\Rightarrow L$ regular

Idea: Construct a DFA with a state for each equivalence class, s.t. the computation on any string x ends in state $[x]$.

Construct a DFA for L $D = (Q, \Sigma, \delta, q_0, F)$ as follows:

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

- δ : $\forall [x] \in Q, a \in \Sigma$, define $\delta([x], a) = [xa]$

is this well defined? $\swarrow$

yes! Can show that if $[x] = [y]$ then $[xa] = [ya]$

- $q_0 = [\epsilon]$

- $F = \{[x] \mid x \in L\}$

$\searrow$ is this well defined?

yes! can show that if $[x] = [y]$ then $x \in L \Leftrightarrow y \in L$

(i.e., every equivalence class has all its strings in L , or all its strings not in L).

The computation of D on any string x ends in $[x]$,
so D accepts $x \Leftrightarrow x \in L$. Thus, D recognizes L ,
and L is regular. 

Revisiting example from last time:

$$L = \{x \in \{0,1\}^* \mid |x| \text{ is divisible by } 3\}$$

The equivalence classes are $\{[\epsilon], [0], [00]\}$:

we can prove that $[\epsilon] = \{x \mid |x| = 0 \bmod 3\}$

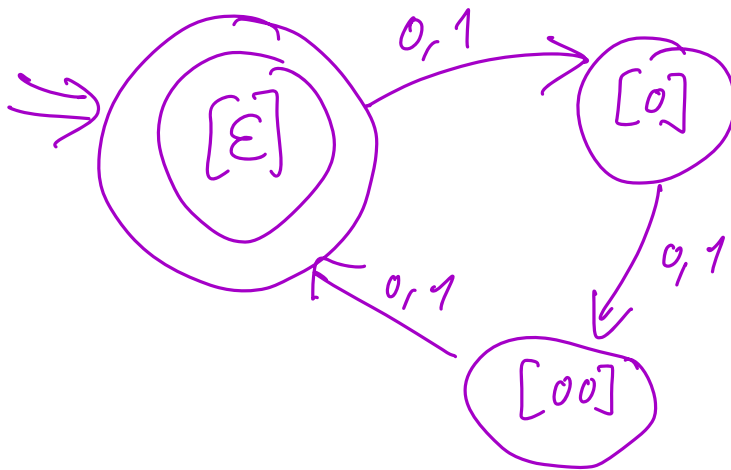
$$[0] = \{x \mid |x| = 1 \bmod 3\}$$

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

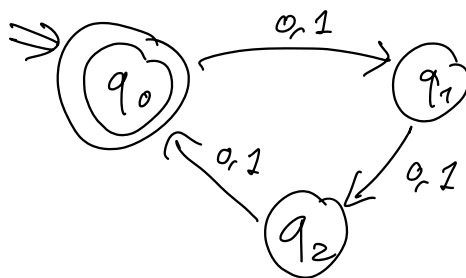
every string
is in one of
 $[\epsilon], [0], [00]$

and that these are all different ($\{\epsilon, 0, 00\}$ pairwise distinguishable)

So, the DFA we get from proof above is



which is exactly the same DFA we previously constructed for L :



To finish proving MN:

③ L regular $\Rightarrow$ #states in minimal DFA for L is exactly #equivalence classes

In ② we constructed a DFA for L with this # of states, so such a DFA exists.

A DFA with fewer states does not exist; this follows from Lemma 2

(if there are k equivalence classes, there are k strings that are pairwise distinguishable)

Thus, this DFA has the minimal # of states



We proved MN theorem.

How do we use it to prove a lang is non-regular?

same idea as in (1) above / saw last time!

recap

Come up with a set S s.t.

- S has infinitely many strings

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

x and y are distinguishable by L

$\Rightarrow L$ has ∞ -many equiv classes $\Rightarrow L$ is non-regular.

More examples:

- $L = \{w \in \{a,b\}^* \mid w \text{ has the same number of a's and b's}\}$ is non-regular

proof: (using MN):

That's what we used in section 1.

In section 2 we used $S = \{a^i \mid i \geq 0\}$ which also worked.

$S = \{bba^ib \mid i \geq 0\}$ also works for this L , but $S = \{a^ib^i \mid i \geq 0\}$ doesn't.

Choose $S = \{a^ib \mid i \geq 0\}$. S is infinite.

Let $x, y \in S$ s.t. $x \neq y$, so $x = a^ib$ $y = a^jb$ $i \neq j$

need to show x, y are distinguishable by L

Choose $z = b^{i-1}$ $xz = a^ib^i \in L$, $yz = a^jb^i \notin L$ (since $i \neq j$)

Alternative proof: using closure $\rightarrow$ recall we proved regular lang closed under complement, $\cap$, $\cup$

• We already proved that $L' = \{a^n b^n \mid n \geq 0\}$ not-regular

• We also proved that $L'' = \{a^i b^j \mid i, j \geq 0\}$ is regular

But note that $\underbrace{L}_{?} \cap \underbrace{L''}_{\text{regular}} = \{a^i b^j \mid i=j\} = \underbrace{L'}_{\text{non-regular}}$

If L was regular, then by closure under intersection,

$L \cap L'' = L'$ would be regular - contradiction.

$\Rightarrow L$ is not regular.

$$L = \{a^{2^n} \mid n \geq 0\} = \{a, aa, aaaa, aaaaaaaaaa, \dots\}$$

is non-regular.

alphabet is $\{a\}$

L = set of strings of length
a power of 2.

proof:

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

Let $x, y \in S$ s.t. $x \neq y$, wlog assume x is shorter

without loss
of generality

$$\text{so } x = a^{2^i} \quad y = a^{2^j} \quad \text{as } i < j$$

choose $z = a^{2^i}$

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

$$yz = a^{2^j} a^{2^i} = a^{2^j + 2^i} \notin L \quad \text{because for } i < j$$

$$2^i + 2^j = 2^i (1 + 2^{j-i}) \quad \text{not a power of 2.}$$

$\underbrace{1 + 2^{j-i}}_{\substack{\text{odd} \\ \geq 3}}$

$\Rightarrow L$ is not regular

challenge: prove this L is non-regular

$$L = \{w \in \{0,1\}^* \mid w = xy, |x| = |y|, \text{ and}$$

x has an even # of 1's

y has an even # of 0's $\}$

This is the end of the first unit of class!

Announcements

* This is the end of material for HW1 and Quiz 1!

- only material covered so far (class, handout, checkup, etc)
can be used on HW1 and Quiz 1

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

* HW1: due Mon 11:59 pm

- See webpage for policies:

lateness, collaboration, AI policy ^{prohibited!} (but read why!)

* Check webpage again (many updates)

* checkup #1 → last attempt due tonight

* checkup #2 → first attempt due Thur night
last attempt a week later

* Handout 1: Coming out soon (will have review
and solved examples for this unit)

* Quiz 1: October 6 in class

- come at the time you are enrolled (unless DS/CARDS/
special agreement)
- about 30 minutes, closed book, closed notes, no cheatsheet.
- Quiz 1 Content:
 - Here's a DFA, describe its language
 - Here's a language, construct a DFA
 - Here's a language, prove non-regular
 - Additional short questions with no justification.

Our second model of Computation (formal defs next time):

NFA: non-deterministic Finite Automaton

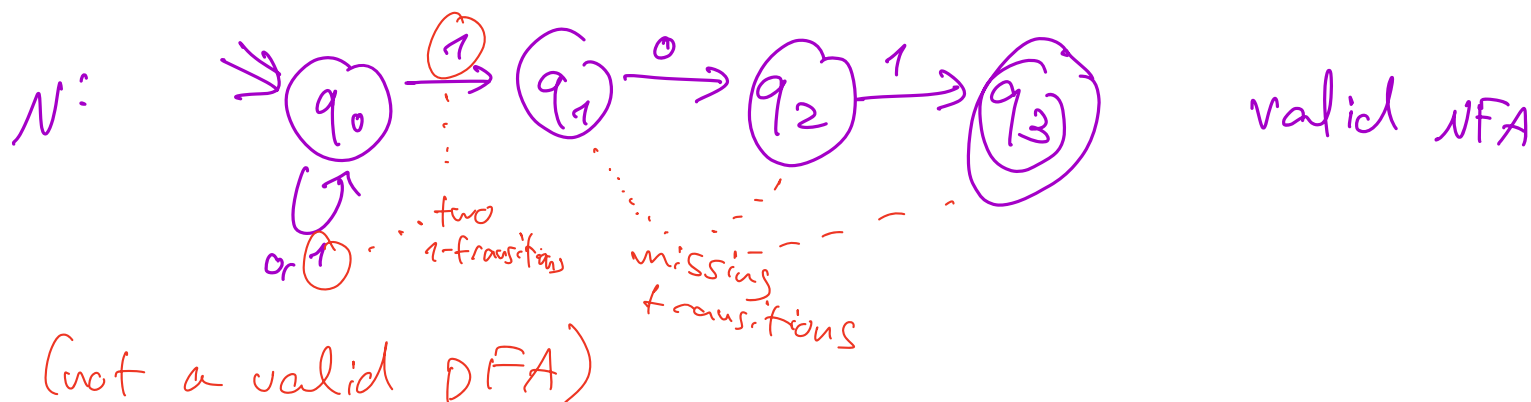
Informally, NFA is like a DFA, except that we can have 0 or more transitions from each state $\neq$ symbol,

$\nwarrow$
DFA: exactly one

transitions can be labeled by ϵ (or by a symbol $\in \Sigma$)

$\nwarrow$
DFA: labeled by symbol

Example:



NFA can have 0 or more Computations on any given String.

$\nwarrow$ DFA: exactly 1

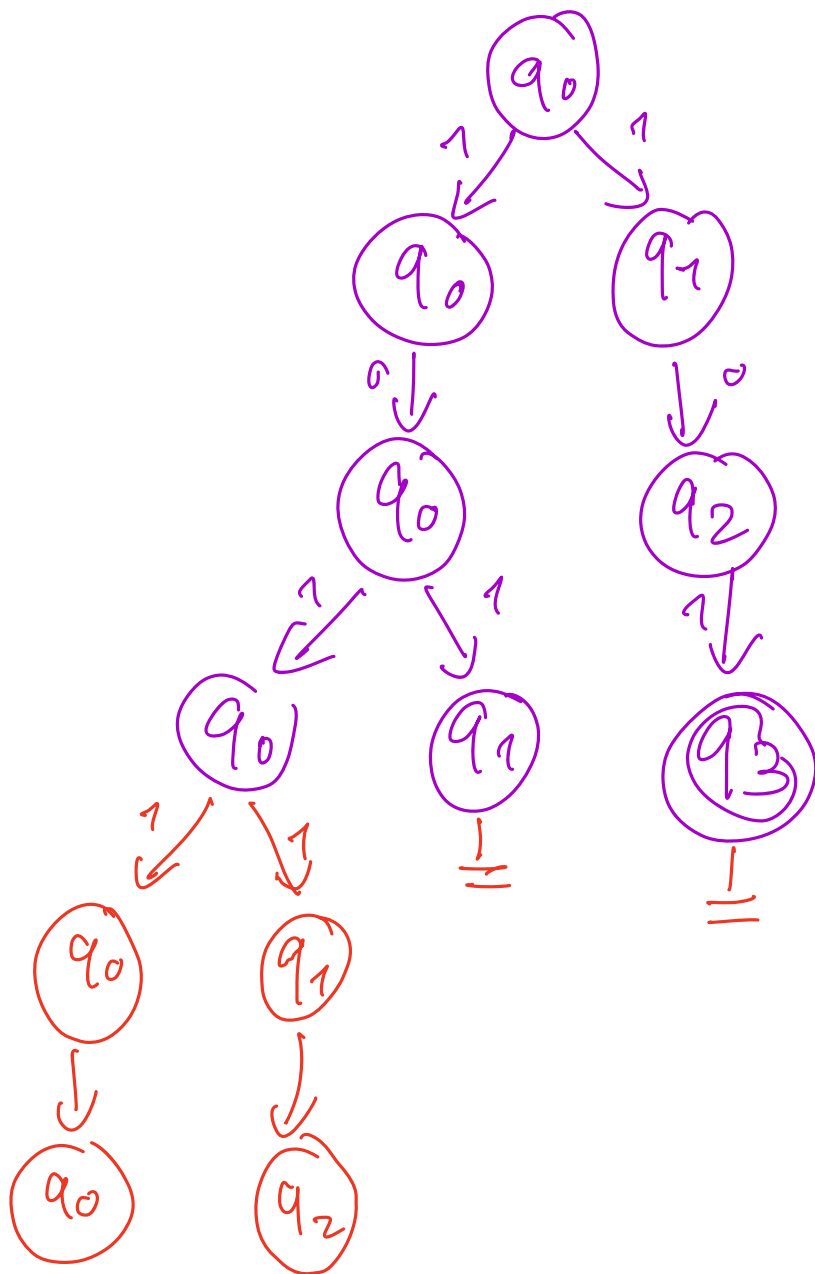
NFA accepts a string if there exists at least one accepting computation.

eg Computations of N on 101:

$q_0 \xrightarrow{1} q_0 \xrightarrow{0} q_0 \xrightarrow{1} q_0$	reject	} $\Rightarrow$ 101 is <u>accepted</u> by N
$q_0 \xrightarrow{1} q_1 \xrightarrow{0} q_2 \xrightarrow{1} q_3$	accept	
$q_0 \xrightarrow{1} q_0 \xrightarrow{0} q_0 \xrightarrow{1} q_1$	reject	

Computation tree for NFA on string

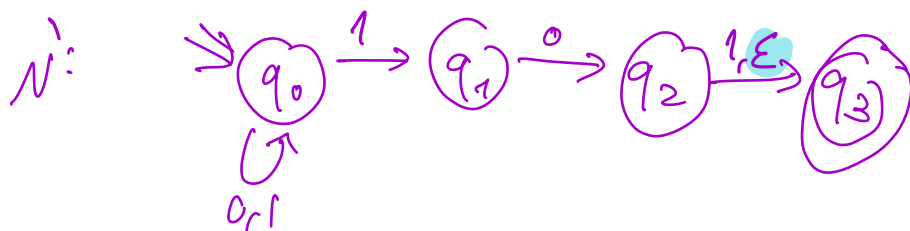
11
10110


$$\Rightarrow 101 \in L(N)$$
$$\Rightarrow 10110 \notin L(N)$$

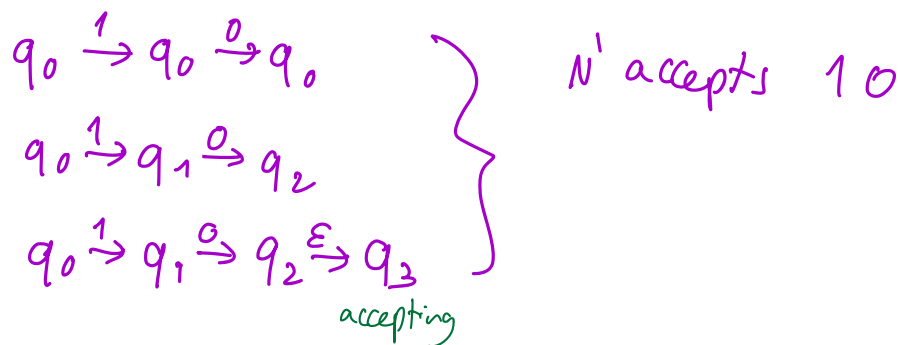
$$L(N) = \{w \mid w \text{ ends in } 101\}$$

When there's an ϵ -transition, computation can take it or not
(without reading another character)

Ex:



Computations of N' on string 10:



$$L(N') = \{w \mid w \text{ ends in } 10 \text{ or } w \text{ ends in } 101\}$$

Another example:



$$L(N'') = \{\epsilon\}$$

(for any other string, the computation dies,
as there is no transition coming out of start state
 $\Rightarrow$ only string that can be accepted is ϵ .)

[Note: in section 1 we didn't cover as much
of the examples, will go over them next time]

