

Lecture 4: waitlists: sec 1: 0 sec 3: 6
sec 2: 28

* Checkup 1 $\Rightarrow$ you can resubmit
(2nd due date Tuesday 11:59 pm)

* HW 1 = out today / tomorrow

* Handout 1 $\Rightarrow$ to help on HW + Quiz

* Checkup 2 : out soon

On the webpage (under "learning objectives")
 $\hookrightarrow$ tips

Last time: more DFA examples

Thm: If L is finite then L is regular

"closure properties"

Thm: If L_1, L_2 are regular then:

* $\overline{L_1}$ is regular

* $L_1 \cup L_2$ is regular

* $L_1 \cap L_2$ is regular

$$\overline{L_1} = \Sigma^* \setminus L_1 = \{w \mid w \text{ is not in } L_1\}$$

$$L_1 \cup L_2 = \{w \mid w \text{ is in } L_1 \text{ or } w \text{ is in } L_2\}$$

$$L_1 \cap L_2 = \{w \mid w \text{ is in } L_1 \text{ and } w \text{ is in } L_2\}$$

Today : Myhill-Nerode $\rightarrow$ not in the book
 $\rightarrow$ we'll post lecture notes

$$L = \{a^n b^n \mid n \geq 0\} = \{\epsilon, ab, aabb, \dots\}$$

$\uparrow$
same number of a's & b's

Is this regular? How to build a DFA for it?

- How do we prove a language L is not regular?

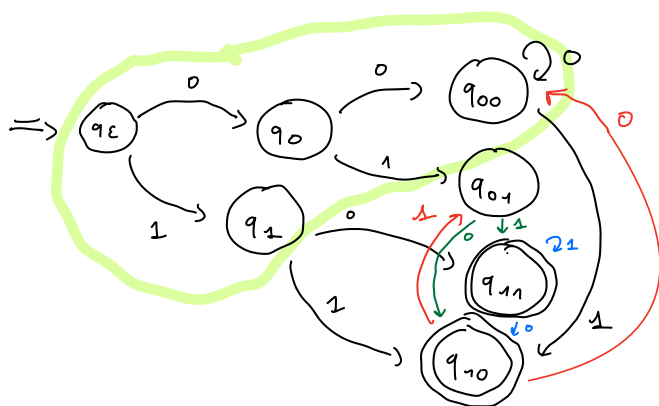
Intuitively, we want to say that a DFA with any fixed number k of states does NOT have enough memory to remember what it needs :

The input strings can be arbitrarily long...

But how to prove this?

What if there is a super efficient algorithm that exists?

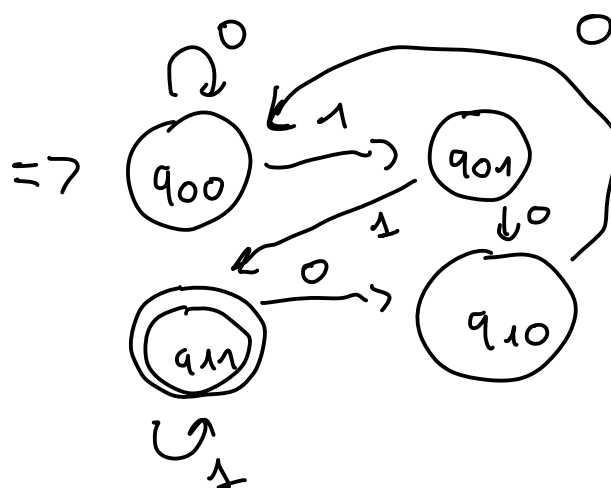
$L_1 = \{w \in \{0,1\}^* \mid \text{the before last symbol in } w \text{ is } 1, |w| \geq 2\}$



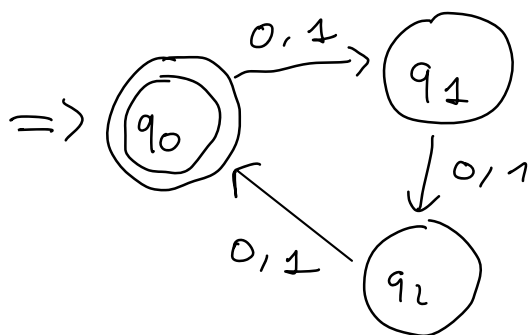
$\epsilon \notin L$ $0 \notin L$

$10 \in L$

$110 \in L$



But what about $L_3 = \{w \in \{0,1\}^* \mid |w| \text{ is divisible by } 3\}$



Can we use 2 states?

Intuitively we must use 3 states to remember the length mod 3

* M.N. is a theorem that allows us to prove:

* how many states you need to build a DFA for a language L .

* or that no finite amount of states
is sufficient

$\hookrightarrow L$ isn't regular (we can't build
a DFA)

Def: Let $L \subseteq \Sigma^*$ be a language. For any
 $x, y \in \Sigma^*$ we say that x and y are
distinguishable (dist.) (with respect to L)
If there exists a string $z \in \Sigma^*$ such that
exactly one of xz or yz is in L
 $\underbrace{\hspace{1cm}}_{L, \text{ concatenation}}$

In this case, we call z a distinguishing
extension of x and y (with respect to L)

If x and y are not distinguishable by L , we
say they are indistinguishable and denote

$x \sim_L y$

$\hookrightarrow$ this is a relation $\hookrightarrow \forall z \in \Sigma^*$
 $xz \in L \iff yz \in L$

Example: $L_1 = \{w \in \{0,1\}^* \mid |w| \geq 2 \text{ and the}$

before last symbol is 1?

$x = \epsilon$, $y = 1$ do we have $x \sim_L y$? **NO**

take $z = 0$ we have $xz = \epsilon \cdot 0 = 0 \notin L$

$$yz = 1 \cdot 0 = 10 \in L$$

so x, y are distinguishable

$x = 10$ $y = 11$, $z = 0$ we have $xz = 100 \notin L$
 $yz = 110 \in L$

$\forall i, j$ we have $0^i \sim_L 0^j$ $0^l = \underbrace{0 \dots 0}_{l \text{ times}}$

why: $\forall i, j$ and $z \in \{0, 1\}^*$: $0^i \cdot z \in L \Leftrightarrow$ the before last symbol of $z = 1$

$$0^i \sim_L 0^j$$

$$\Leftrightarrow 0^j \cdot z \in L$$

Example: $L = \{w \in \{0, 1\}^* \mid |w| = 0 \pmod 3\}$

* give 2 strings that are distinguishable

* give 2 strings that are indist.

$x = 0$, $y = 01$, $z = 1$

$$xz = 01 \notin L$$

$$yz = 011 \in L$$

$$\hookrightarrow x=0, y=1 \quad \forall z \in \{0,1\}^*$$

$$xz = 0z \in L \Leftrightarrow z \text{ has length } 2 \bmod 3$$

$$\Leftrightarrow 1z \in L \Leftrightarrow yz \in L$$

In words: xz and yz always have the same length

Lemma 1: Let $L \subseteq \Sigma^*$ be a regular language.

Let $D = (Q, \Sigma, \delta, q_0, F)$ be a DFA recognizing L
 $L, L(D) = L$

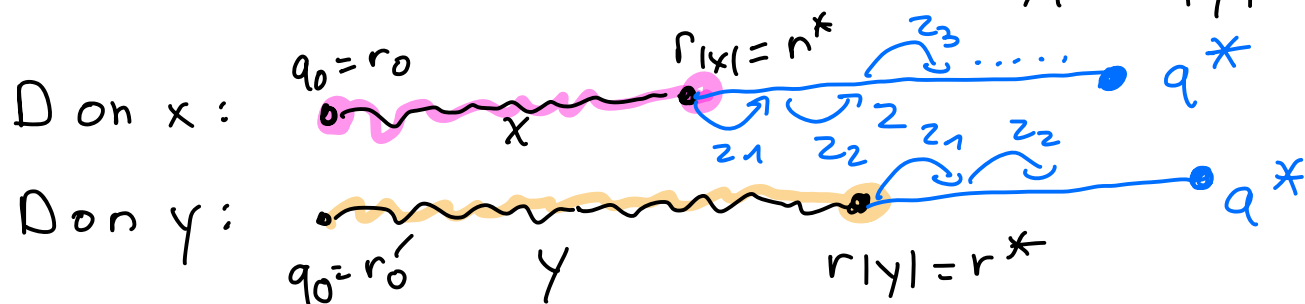
If x, y are distinguishable by L , then the computation of D on x and y can't end in the same state

Pf: Say z is a distinguishing extension of x, y

Let $(r_0, \dots, r_{|x|})$ be the computation of D on x

Let $(r'_0, \dots, r_{|y|})$ be the computation of D on y

Assume for contradiction that $r_{|x|} = r_{|y|} = r^*$



What is the computation of D on xz and yz

So xz and yz end in the same state q^* .

But exactly one of xz and yz is in L .

If q^* is in F (accept state) we accept both xz and yz $\&$

If q^* is not in F we reject both strings xz and yz $\&$

So contradiction: the computation of D on x and y must end in different states.

Lemma 2: Let $L \subseteq \Sigma^*$ be a language. Suppose $\exists$ a set $S = \{x_1, \dots, x_k\}$ (so $|S| = k$) such that $\forall i \neq j$ we have that x_i and x_j are distinguishable by L . Then there is no DFA for L with $< k$ states.

Proof: sketch. We have k strings that are pairwise distinguishable. Assume for contradiction we had a DFA for L using $< k$ states.

By pigeonhole principle, there exists $\overbrace{x, y}^{x \neq y} \in S$ such that the computation of D on x, y end in the same state.

By lemma 1 $\Rightarrow$ this is a contradiction (because x, y are distinguishable)

Back to $L = \{w \in \{0,1\}^* \mid |w| \equiv 0 \pmod{3}\}$

We have a DFA with 3 states. Is that optimal?

$$S = \{\epsilon, 0, 00\}$$

x	y	what z do I pick?		
ϵ	0	$z = \epsilon$	$\epsilon \cdot \epsilon \in L$	$0 \cdot \epsilon \notin L$
0	00	$z = 0$	Then $xz = 00 \notin L$	$yz = 000 \in L$
ϵ	00	$z = \epsilon$	$xz = \epsilon \cdot \epsilon \in L$	$yz = 00 \cdot \epsilon \notin L$

So by lemma 2, L needs ≥ 3 states for a DFA

Claim: the "indistinguishability relation" $\sim_L$ is an equivalence relation

- symmetric, reflexive, transitive

Proof: exercise

Importantly: $\sim_L$ partitions Σ^* into equivalence classes

For a string x : we let $[x] = \{y \mid x \sim_L y\}$
be the equivalence class of x

Myhill-Nerode: Let $L \subseteq \Sigma^*$ be a language

L is regular if and only if the number of equivalence classes of $\sim_L$ is finite

Moreover if L is regular, the number of states in the smallest DFA recognizing L is the number of equivalence classes of $\sim_L$.

If you can determine what the equivalence classes of $\sim_L$ are then:

- (1) you can tell if L is regular
- (2) if L is regular, you can build the minimal DFA recognizing L

Our focus is on proving L is not regular.

↳ to do this we only need to prove $\sim_L$ has infinitely many equivalence classes.

this proves $\sim_L$ has ∞ many equivalence classes $\Leftarrow$

To show L is not-regular, we want to find a set S of strings such that

- (1) S has infinitely many strings
- (2) $\forall x, y \in S$ ($x \neq y$) x and y are distinguishable by L

Example: $L = \{a^n b^n \mid n \geq 0\}$ is not regular

Proof: Let $S = \{ a^l \mid l \geq 0 \} = \{ \epsilon, a, aa, aaa, \dots \}$

(1) S has infinitely many strings

(2) let $x, y \in S$ where $x \neq y$. We have

$$x = a^i, y = a^j \text{ where } i \neq j$$

(choose: z such that z is a distinguishing extension)
for x and y

$$\text{Let } z = b^i. \text{ We have } \begin{aligned} xz &= a^i b^i \in L \\ yz &= a^j b^i \notin L \quad (i \neq j) \end{aligned}$$

x and y are distinguishable by

So $\sim_L$ has ∞ many equivalence classes $\Rightarrow$ by MN
 L is not regular