CPSC 421/501 Oct 26

Today: Turing Machines (Ch. 3 [Sip])

- single tape (simplest)
- multi-tape (needed for "better" notion of time complexity)
- non-deterministic (for P vs. NP)
- Turing machines with oracle calls (Baker-Gill-Solovay Thm)

(how not to solve P. vs NP)

Good news:

- Turing machines much more realistic than DFA's for "time complexity"

"time complexity" = running time
of an algorithm

— poly time in Turing machine

= " " in any reasonable
sense (deterministic, no randomness,
no quantum, ~~no~~
up to poly # of threads)

⋮

Look at

C_k, $\{0^n 1^n \mid n \in \mathbb{N}\}$, ...

=

Complexity of

$\{0^n 1^n \mid n \in \mathbb{N}\}$ w DFA's
NFA's

= ?? ∞

∞ # of states ...

Turing machines!

single tape machine: roughly finite # states

$O(n^2)$ time

2-tape machine: easy: finite # states

time $O(n)$

Start today with

$$C_k = \{ w \in \{a,b\}^* \mid \text{the } k^{th} \text{ to last letter of } w \text{ is "a"} \}$$

Turing machine:

Textbook [Sip]

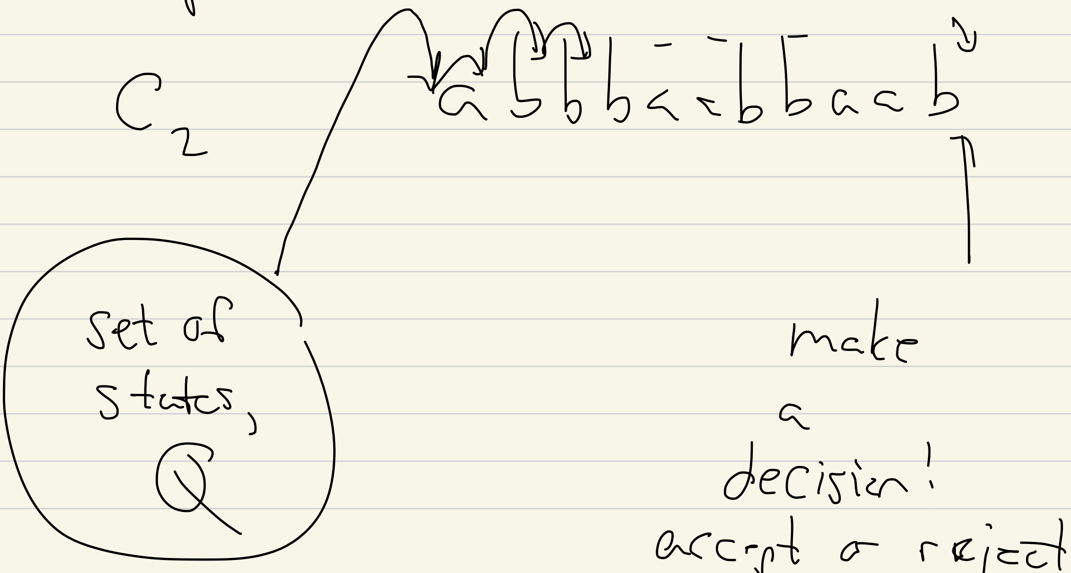
Turing machine = DFA + a bit

more power

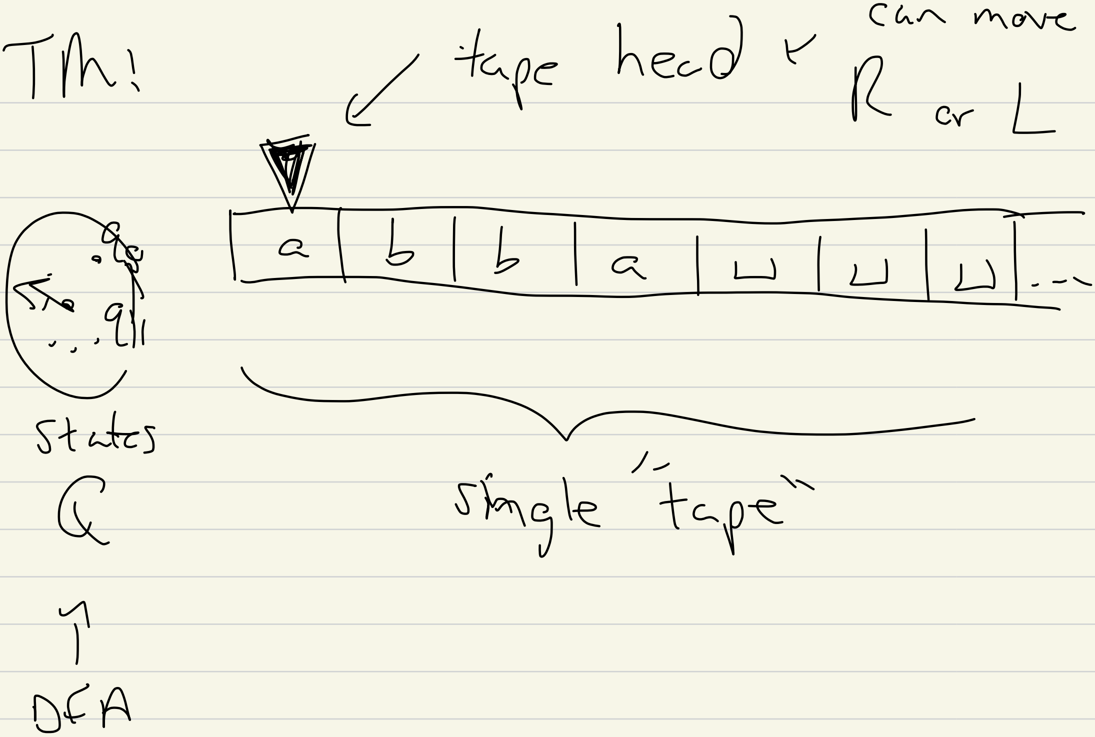
DFA: for C_1

input

C_2



TM:



So here:

- tape head moves <u>left</u> or right
- you can write on the tape

write/read: Σ = alphabet of input

$\sqcup$ = blank cell indicator

- tape alphabet Γ contains Σ, $\sqcup$

Formally: Turing machine:

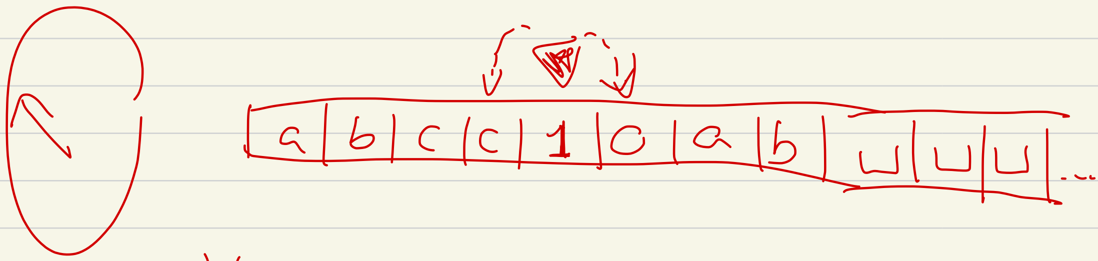
$$M = (Q, \Sigma, \underbrace{\Gamma}_{\text{tape alphabet}}, \delta, q_0, \underbrace{q_{acc}, q_{rej}}_{\text{you accept or reject the input + stop the computation}})$$

"time" that M takes a input w

= # of steps it takes to reach q_{acc}, q_{rej}

reaching q_{acc}, q_{rej} "halting"

What is δ ?



at some state

say $\Sigma = \{a, b\}$

$\Gamma = \{a, b, \sqcup, 0, 1, c\}$

$\delta : Q \times \Gamma \longrightarrow Q \times \Gamma \times \{L, R\}$

(for DFA: $\delta : Q \times \Sigma \longrightarrow Q$

for NFA: $\delta : Q \times \Sigma \longrightarrow \text{Power}(Q)$)

An algorithm on a TM (single-tape) for

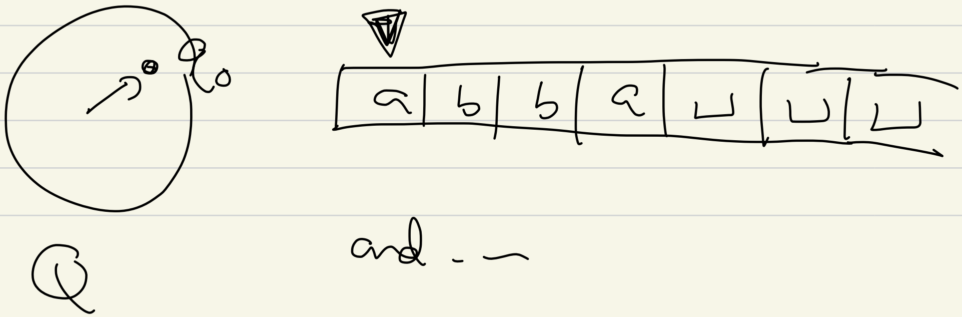
$$C_2 = \{ w \in \{a,b\}^* \mid \text{the } 2^{nd} \text{ to last character/symbol of } w \text{ is } a \}$$

$$= \{a,b\}^* \circ \{a\} \circ \{a,b\}$$

$$= \Sigma^* a \Sigma, \quad \Sigma = \{a,b\}.$$

initial situation

input is abba



and . . .

$\Gamma \supseteq \{a, b, \sqcup, \ldots\}$ tape alphabet

$\Sigma = \{a, b\}$

initial state q_0

=

break 10:14 – 10:19

=

Goal:

- To convince you that

(1) poly time on TM = any other notion of classical poly time

(2) you can build a "universal Turing machine"

Question: you have $w \in \{a,b\}^*$,

$|w| = n$, you want an algorithm

for C_2 (C_k, $\{0^n 1^n\}$, etc.)

s.t. # states is not too big

states ↔ your C

program,

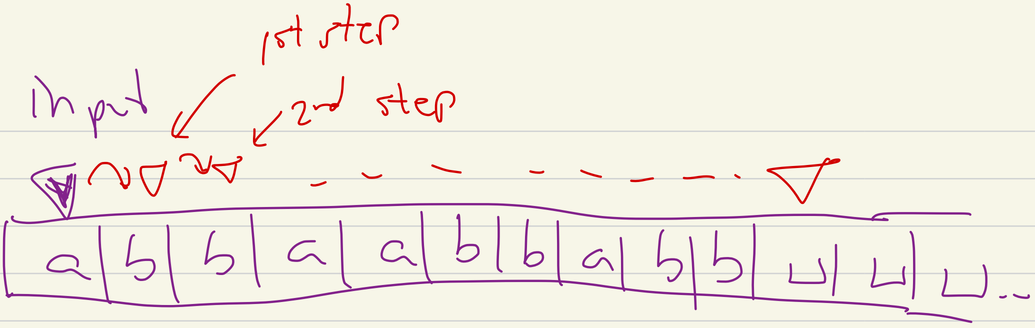
Javascript

program, ~

time it takes to run the

computation = # steps

is small function of n



high-level descriptions of TM

implementation-level " " "

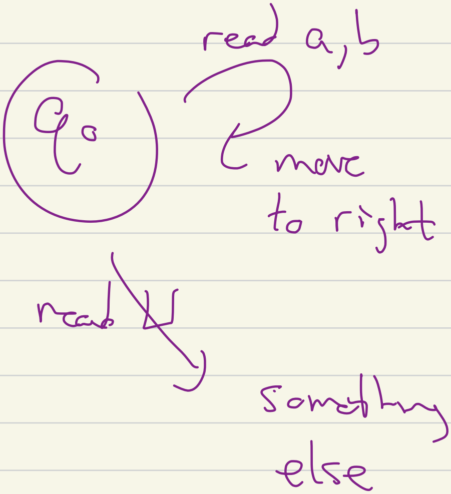
formal " " "

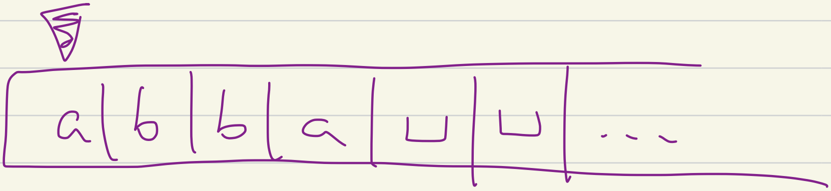
you specify δ

and ...

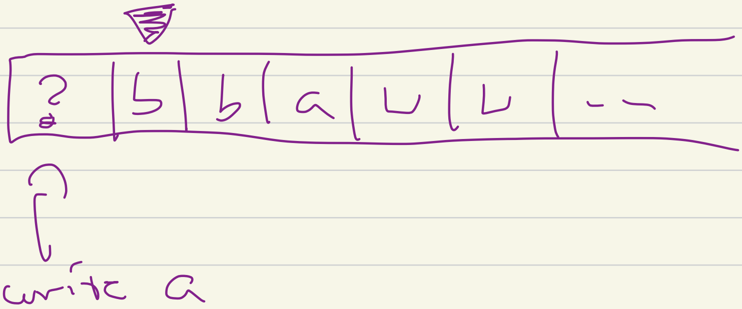
High-level: go to end of word, jump back two steps to the left after reading the ⊔ symbol

Idee



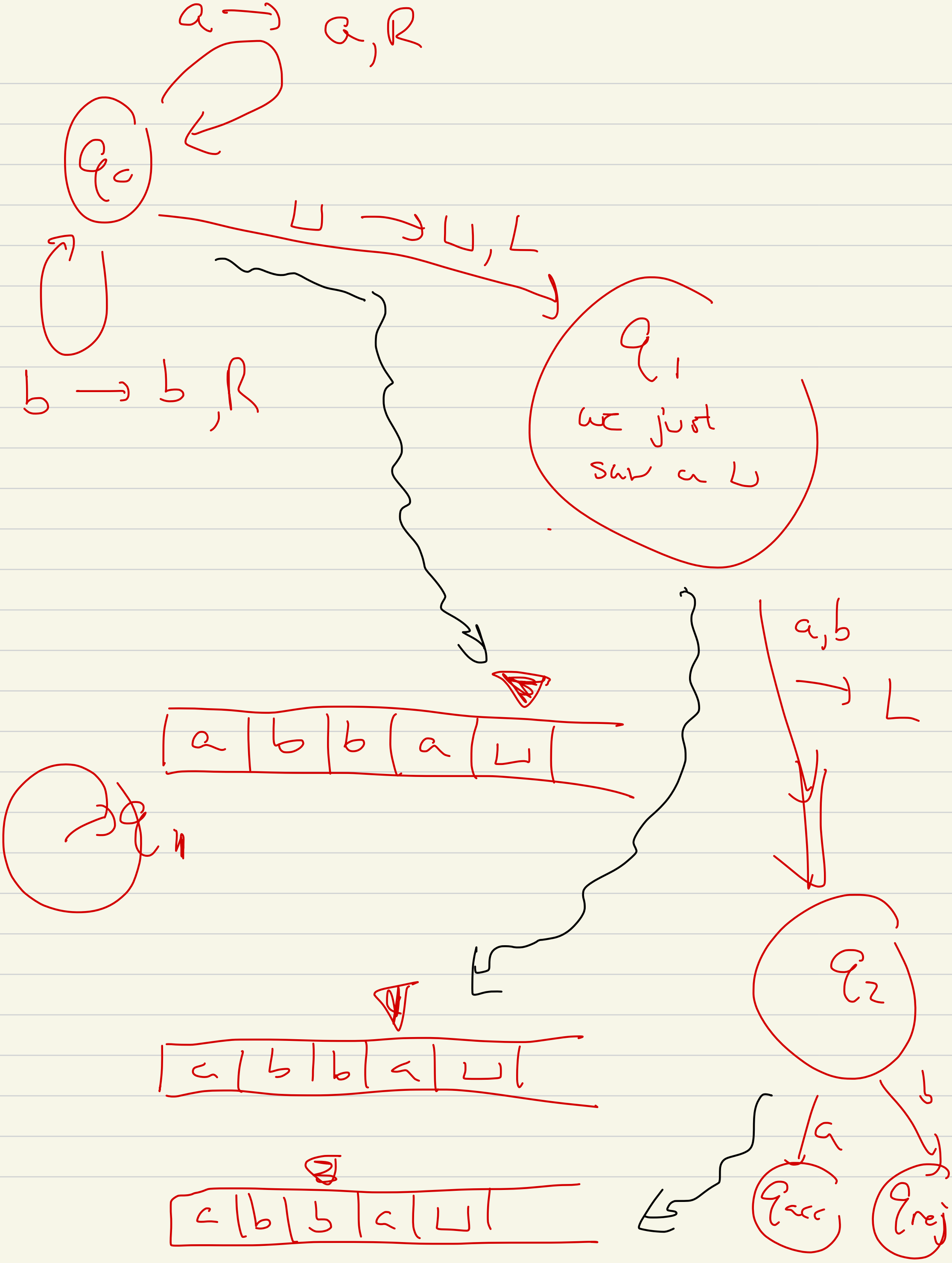


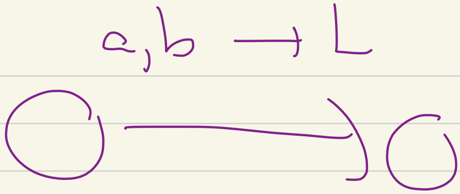
initial



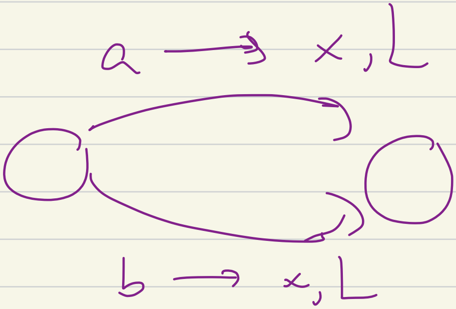
$$\delta(q_0, a) = (q_0, a, R)$$

$$\delta(q_0, b) = (q_0, b, R)$$


a → a, R
q0
⊔ → ⊔, L
b → b, R
q1
we just saw a ⊔
a, b → L
a | b | b | a | ⊔
q4
a | b | b | a | ⊔
a | b | b | a | ⊔
q2
a
b
qacc
qrej



shorthand for



x = anything

then

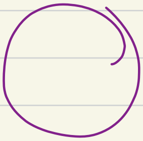
doesn't matter

$$\delta(q_2, a) = (q_{acc}, \quad , \quad)$$

$$\delta(q_2, b) = (q_{rej}, \quad , \quad)$$

Clear! If $|w| \geq 2$, then this algorithm accepts $w \in C_2$, rejects $w \notin C_2$

Convention: if



no arrow out
on some symbol
in Γ

means: this will never happen

OK for TM, never for DFA's, but OK for NFA's

To be continued on Thursday

- Make sure this or a variant of this works on input "⊔", "a", and "b"

- Write a TM for $\{0^n1^n\}$