CS 121: Lecture 10
Turing Machines

Madhu Sudan

https://madhu.seas.Harvard.edu/courses/Fall2020

Book: https://introtcs.org

How to contact us
- The whole staff (faster response): CS 121 Piazza
- Only the course heads (slower): cs121.fall2020.course.heads@gmail.com

Announcements:

- Midterm 1 next week:
  - Logistics announcement by Thursday
  - Prep Material: Canvas → Files → Midterm Prep
  - ~~Likely~~ Now: 2 pages of typset cheatsheet allowed. No other external refs.
- Homework 3 due Thursday

- Advanced Sections: Christina Ilvento on Differential Privacy!

Thursdays at 4:30.

Where we are:

Part I: Circuits:
Finite computation, quantitative study

Part II: Automata:
Infinite restricted computation, quantitative study

Part III: Turing Machines:
Infinite computation, qualitative study

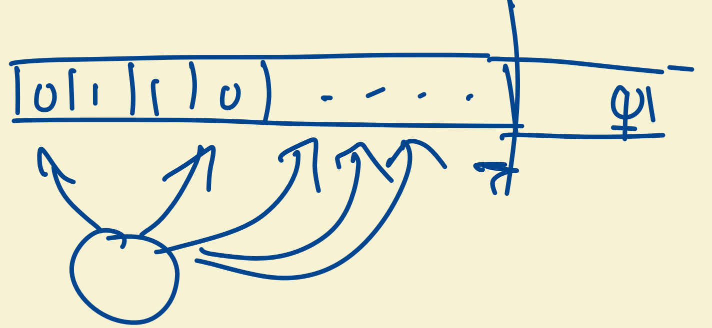
Part IV: Efficient Computation:
Infinite computation, quantitative study

Part V: Randomized computation:
Extending studies to non-classical algorithms

Today:

- Definition of Turing Machines
- A function F not computable by DFA or Circuits
- Computing F with Turing Machine

Definition of Turing Machine (TM)



- Recall: DFA = Finite state control + input on tape + move right on each step.
- In a nutshell: TM = DFA + “Write” + “Move left+right on tape”
  - (Either “Write” / “Move left+right” on its own insufficient)

DFA: $T : [C] \times \{0,1\} \to [C]$

Write

$\delta : [k] \times \Sigma \to [k] \times \Sigma \times \{L, R, S, H\}$

- TM: Main change:
  - More Involved Transition function: T (now δ):
    - δ: (current state, read symbol) $\mapsto$ (new state, write symbol, direction of move/halt)
  - Explicit halting (don’t just end after reading last input bit)
    - Computes functions: output = concatenation of $\{0,1\}$ symbols on tape.

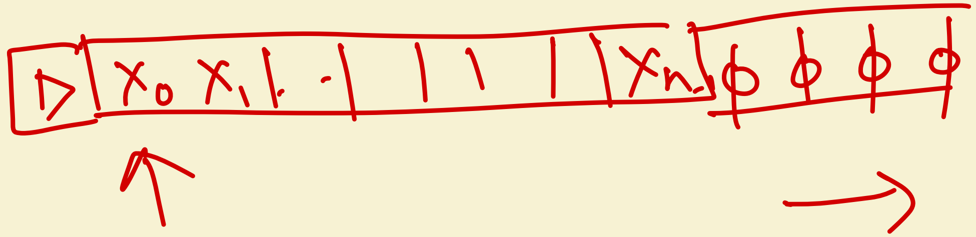
Formal Definition

States $= \{0, \ldots, k-1\}$

start

- (Barak, Definition 7.1):
- TM on k states and alphabet $\Sigma \supseteq \{0,1,\triangleright,\phi\}$ is given by $\delta: [k] \times \Sigma \to [k] \times \Sigma \times \mathrm{Action}$, where $\mathrm{Action} = \{L, R, S, H\}$
  - L=Left, R=Right, S=Stay (don't move), H=Halt (done!!)
- Operation:
  - Start in state 0, Tape $T = \triangleright x_0 \ldots x_{n-1}\phi\phi\phi \ldots$, Head (i) at x_0
  - General step: current state q ; input symbol σ:

    Let $\delta(q,\sigma) = (r,\tau,X) \Rightarrow$ Write τ on tape (overwriting σ) ; Move to state r;
    Move Head left ($i \leftarrow i-1$) if $X = L$; right if $X = R$; don't move if $X = S$.
  - Repeat General step until $X = H$



$x_0 \ldots x_{n-1} \in \{0,1\}$

$\Downarrow$

input $\in \{0,1\}^*$

TM Example

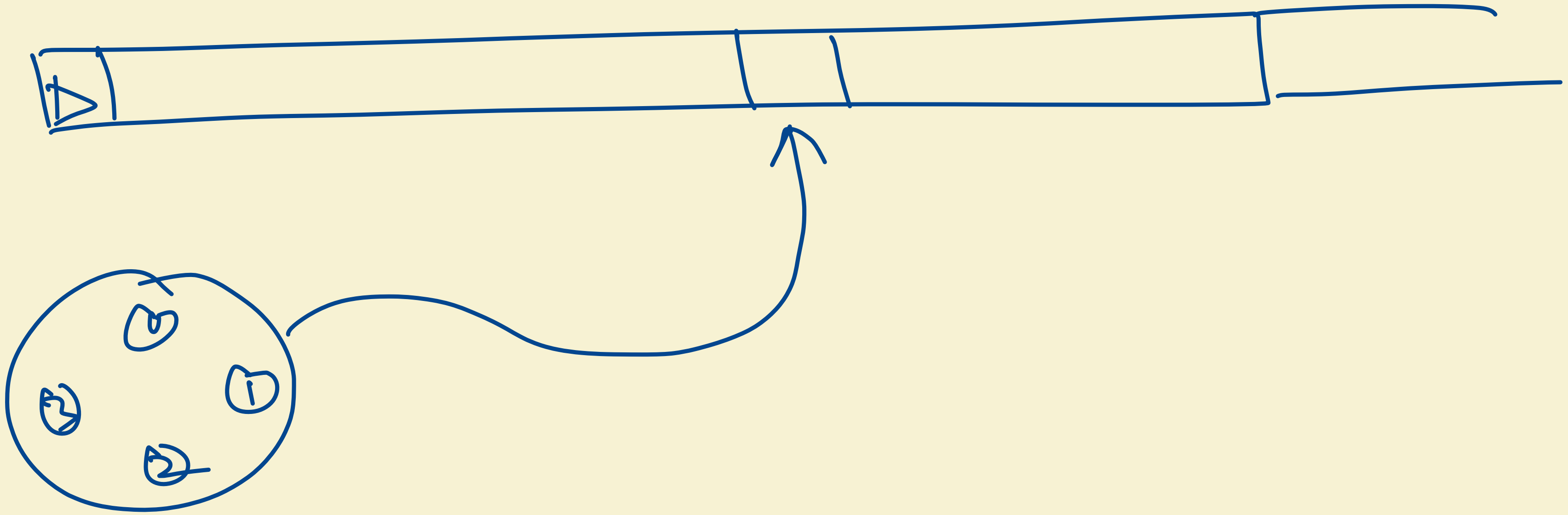
nextState   Symbol we write   Action of Tape Head

End/Halt

- Example: $k = 1; \Sigma = \{0,1,\triangleright,\phi\}\,; \delta(0,\sigma) = \begin{cases} (0,0,R) \text{ if } \sigma \in \{0,1\} \\ (0,\phi,H) \text{ if } \sigma \notin \{0,1\} \end{cases}$

- What does TM output on $\triangleright 101\phi$ … (in future, we won't write $\triangleright$ or ϕ)

Output = 000



TM Example

- Example: $k = 1$; $\Sigma = \{0,1,\triangleright,\phi\}$; $\delta(0,\sigma) = \begin{cases} (0,0,R) \text{ if } \sigma \in \{0,1\} \\ (0,\phi,H) \text{ if } \sigma \notin \{0,1\} \end{cases}$
- What does TM output on $\triangleright 101\phi$ … (in future, we won't write $\triangleright$ or ϕ)
- What function does TM compute?

0 1 A 1 1 B 1 0 => 0 1 1 1 0

Def: Output of Turing Machine = concatenation of 0s & 1s on tape.

A "hard" function

- $f: \{0,1\}^* \to \{0,1\}$, $f(x) = 1 \Leftrightarrow x = 1^n$ for $n = 2^t$ for integer t

$f(\text{""}) = 0$

$f(1) = 1$

$f(1^2) = 1$

$f(1^{256}) = 1$

$f(1^{200}) = 0$

. . .

Exercise Break 1

- $f:\{0,1\}^* \rightarrow \{0,1\},\ f(x) = 1 \Leftrightarrow x = 1^n$ for $n = 2^t$ for integer t
- (30 sec) Prove that no circuit computes f
- (4 min 30 sec) Prove no DFA computes f
  - Part 1: Focus on big idea; defer calculations/parameter settings.
  - Part 2: Get your hands dirty; do calculations+parameter settings.

if length of input is 2^t & reg. exp e has length q

then $\Rightarrow$ must be a star in e;

$$[\, q < 2^t \,]$$

$$\Rightarrow \underbrace{1^a}\ \underbrace{1^b}_{\text{matches } \star}\ \underbrace{1^c} \Rightarrow \underbrace{1^a\ 1^{2b}\ 1^c}$$

$a+b+c = 2^t \Rightarrow f(1^{a+b+c}) = 1$ $\qquad f(1^{a+2b+c}) = 1$

$1 \to q_1$

$1^2 \to q_2$

$1^3 \to$

$\vdots$

$1^x \to q_x$

$y > x$ $\quad 1^y \to q_y$

$1^{c+1} \to$

DFA has c states

if

$\exists\, x \;\&\; y$ s.t.

$q_x = q_y$

$1^y \cdot 1^z = 1^{2^t}$

$1^x \cdot 1^z = 1^{\underbrace{2^t + x - y}_{?}}$

Hint: $t = 2^y - y$

can prove that $2^y + x - y$ is not a power of 2.

F is computable by a Turing Machine

Main Idea: Loop many times:

Scan string left to right

Replace every alternate 1 by 0;

reject if number of 1s is odd and greater than 2.

1 1 1 1 1 1 1 1 1 1

1 # 1 # 1 # 1 # 1 #

1 # # # 1 # # # 1 # ?

- Say $\exists$ DFA with C states computing f.
- $q_i \doteq$ state after input 1^i, $i = 0, 1, 2, \dots, C+1, \dots$

$\uparrow x$ $\uparrow y$

$\exists\ 1 \le x < y \le k+1$ s.t.

$q_x = q_y$

$z = 2^y - y$

$$\boxed{\begin{array}{c} 2^y - y + x + 2^t \\ \text{for any } t \end{array}}$$

$$\begin{cases} 1^x \cdot 1^z \Longrightarrow \text{must reject} \\ 1^y \cdot 1^z = 1^{2^y} \Rightarrow \text{must accept.} \end{cases}$$

More details: Alphabet & States

Alphabet $\Sigma = \{0, 1, \triangleright, \phi, \#\}$

▷ 1 1 1 1 1 1 1 1 1 1 ȸ ȸ . . . ȸ

q

0. Start/Not seen any ones
1. Move Right first one ← after seeing one 1
2. Move Right even # of ones ← after seeing even # of 1s
3. Move Right odd # of ones ← after seeing an odd > 2 # of 1s.
4. Move Left ←
5. Clean Right and Reject
6. Clean Left and Reject

Input $\triangleright\ 1\ 1\ 0\ 1\ 1\ 1\ 1\ 1$

$\uparrow \qquad q$

$\rightarrow$ State $0, 1, 2$ $\rightarrow$ clean right & reject

$\triangleright\ 1\ 1\ 0\ \#\ \#\ \#\ \#\ \#\ \#\ \triangleright$

q

$\triangleright\ \#\ \#\ \#\ \#\ \#\ \#\ \#\ \#\ \#\ \#\ 0$

halt

clean left & reject

State/Input	▷	0	1	ϕ	#
0	invalid	(5,#,R)	(1,1,R)	(6,0,L)	(0,#,R)
1	invalid				
2	invalid				
3	invalid				
4	(0,▷,R)				
5					
6	(–,▷,H)				

Alphabet $\Sigma = \{0,1,\triangleright,\phi,\#\}$

0. Start/Not seen any ones
1. Move Right first one
2. Move Right even # of ones
3. Move Right odd # of ones
4. Move Left
5. Clean Right and Reject
6. Clean Left and Reject

$f\colon \{0,1\}^* \to \{0,1\},\ f(x) = 1 \Leftrightarrow x = 1^n$ for $n = 2^t$ for integer t

State/Input	$\triangleright$	0	1	ϕ	#
0					
1					
2					
3					
4					
5					
6					

Alphabet $\Sigma = \{0,1,\triangleright,\phi,\#\}$

0. Start/Not seen any ones
1. Move Right first one
2. Move Right even # of ones
3. Move Right odd # of ones
4. Move Left
5. Clean Right and Reject
6. Clean Left and Reject

Exercise Break 2:

Fill in rows for states 2 & 4

Keep answer ready (5 triples) to type into chat. Use D for $\triangleright$

State/Input	$\triangleright$	0	1	ϕ	#
0	invalid	(5,#,R)	(1,1,R)	(-,0,H)	(0,#,R)
1	invalid	(5,#,R)	(2,#,R)	(-,#,H)	(1,#,R)
2	invalid	(5,#,R)	~~(3,#,R)~~ (3, 1, R)	(4,ϕ,L)	(2,#,R)
3	Invalid	(5,#,R)	(2,1,R) (2, #, R)	(6,0,L)	(3,#,R)
4	(0,$\triangleright$,R)	invalid	(4,1,L)	invalid	(4,#,L)
5	invalid	(5,#,R)	(5,#,R)	(6,0,L)	(5,#,R)
6	(-, $\triangleright$,H)	invalid	(6,#,L)	invalid	(6,#,L)

Alphabet $\Sigma = \{0, 1, \triangleright, \phi, \#\}$

0. Start/Not seen any ones
1. Move Right first one
2. Move Right even # of ones
3. Move Right odd # of ones
4. Move Left
5. Clean Right and Reject
6. Clean Left and Reject

Summary & Next

- Achieved today:
  - Defined TM
  - Shown it computes one function that DFA and circuits can’t
- Next Lecture:
  - More examples.
  - Towards equivalence with (all) programs