BU CS 332 – Theory of Computation

Lecture 11:

- TM Variants
- Closure Properties

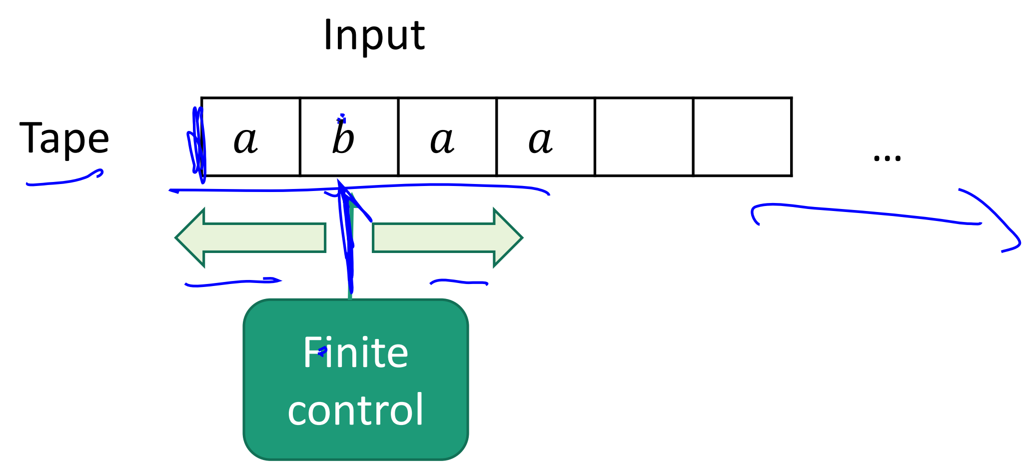
Reading:

Sipser Ch 3.2

Ran Canetti

October 15, 2020

The Basic Turing Machine (TM)



- Input is written on an infinitely long tape
- Head can both read and write, and move in both directions
- Computation halts when control reaches "accept" or "reject" state

Formal Definition of a TM

A TM is a 7-tuple $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}})$

- Q is a finite set of states
- Σ is the input alphabet (does **not** include ⊔)
- Γ is the tape alphabet (contains ⊔ and Σ)
- $q_0 \in Q$ is the start state
- $q_{\text{accept}} \in Q$ is the accept state
- $q_{\text{reject}} \in Q$ is the reject state ($q_{\text{reject}} \neq q_{\text{accept}}$)
- δ is the transition function

TM Transition Function

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

L means "move left" and R means "move right"

$\delta(p, a) = (q, b, R)$ means:

- Replace a with b in current cell
- Transition from state p to state q
- Move tape head right

TM Transition Function

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

L means "move left" and R means "move right"

$\delta(p, a) = (q, b, R)$ means:

- Replace a with b in current cell
- Transition from state p to state q
- Move tape head right

$\delta(p, a) = (q, b, L)$ means:

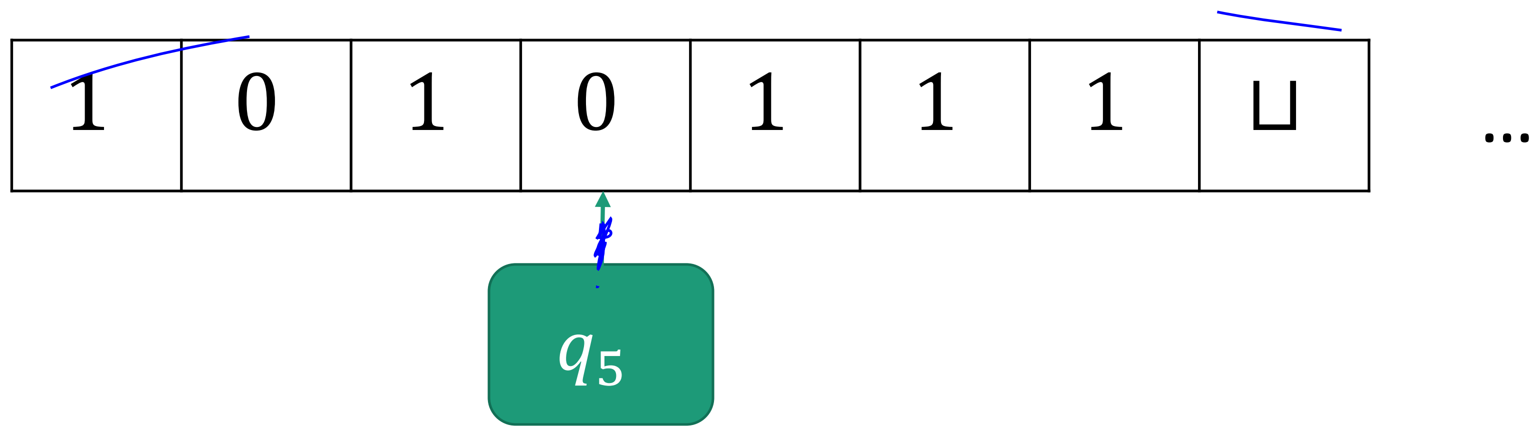
- Replace a with b in current cell
- Transition from state p to state q
- Move tape head left UNLESS we are at left end of tape, in which case don't move

Configuration of a TM

A string with captures a "a snapshot of the computation" (should suffice to continue the computation)

what information is needed

in a configuration?

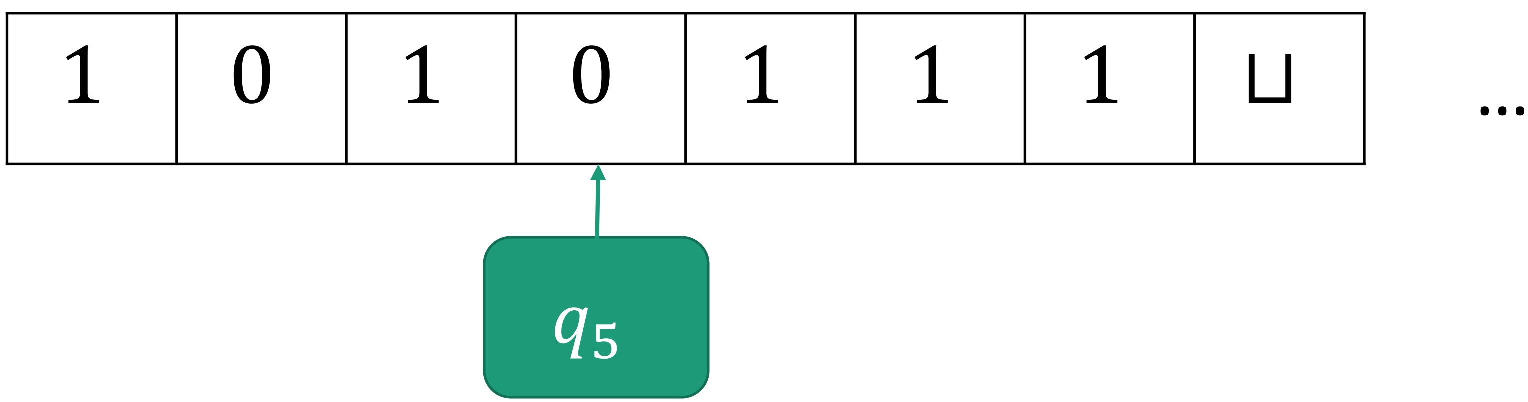


Configuration of a TM: Formally

A configuration is a string uqv where $q \in Q$ and $u, v \in \Gamma^*$

- Tape contents = uv (followed by blanks ⊔)
- Current state = q
- Tape head on first symbol of v

$101q_5 0111$



How a TM Computes

Start configuration: $q_0 w$

One step of computation:

- $ua\ q\ bv$ yields $uac\ q'\ v$ if $\delta(q, b) = (q', c, R)$
- $ua\ q\ bv$ yields $u\ q'\ acv$ if $\delta(q, b) = (q', c, L)$
- $q\ bv$ yields $q'\ cv$ if $\delta(q, b) = (q', c, L)$

Accepting configuration: $q = q_{\text{accept}}$

Rejecting configuration: $q = q_{\text{reject}}$

How a TM Computes

M accepts input w if there is a sequence of configurations $C_1, \dots, C_k$ such that:

- $C_1 = q_0 w$
- C_i yields C_{i+1} for every i
- C_k is an accepting configuration

$L(M)$ $=$ the set of all strings w which M accepts

A is Turing-recognizable if $A = L(M)$ for some TM M

How a TM Computes

M accepts input w if there is a sequence of configurations $C_1, \ldots, C_k$ such that:

- $C_1 = q_0 w$
- C_i yields C_{i+1} for every i
- C_k is an accepting configuration

$L(M) =$ the set of all strings w which M accepts

A is Turing-recognizable if $A = L(M)$ for some TM M:

- $w \in A \implies M$ halts on w in state q_{accept}
- $w \notin A \implies M$ halts on w in state q_{reject}

How a TM Computes

M accepts input w if there is a sequence of configurations $C_1, \ldots, C_k$ such that:

- $C_1 = q_0 w$
- C_i yields C_{i+1} for every i
- C_k is an accepting configuration

$L(M) =$ the set of all strings w which M accepts

A is Turing-recognizable if $A = L(M)$ for some TM M:

- $w \in A \implies M$ halts on w in state q_{accept}
- $w \notin A \implies M$ halts on w in state q_{reject}
  or M runs forever

Recognizers vs. Deciders

- A TM is a decider if it halts (i.e., either accepts or rejects) on all inputs.

A is (Turing-)decidable if $A = L(M)$ for some TM M which is a decider

Recognizers vs. Deciders

- A TM is a decider if it halts (i.e., either accepts or rejects) on all inputs.

A is (Turing-)decidable if $A = L(M)$ for some TM M

which is a decider:

- $w \in A \implies M$ halts on w in state q_{accept}
- $w \notin A \implies M$ halts on w in state q_{reject}

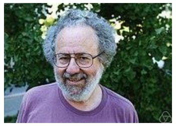
Back to Hilbert's Tenth Problem

Computational Problem: Given a Diophantine equation, does it have a solution over the integers?

$L =$

- L is Turing-recognizable

- L is not decidable (1949-70)




Back to Hilbert's Tenth Problem

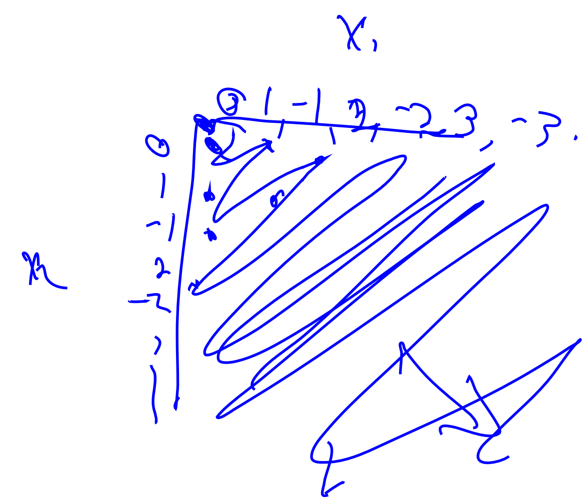
Computational Problem: Given a Diophantine equation, does it have a solution over the integers?

$$L = \{\mathrm{p} = \mathrm{a}_1 \mathrm{x}_1^{\mathrm{c1}} + \cdots + \mathrm{a}_n \mathrm{x}_n^{\mathrm{cn}} \mid \exists v_1 \ldots v_n \in N \; s.t.\, p(x_1 \ldots x_n) = 0\}$$

$c_i, C_i \in N$

- L is Turing-recognizable

$p(x_1 \ldots x_n)$



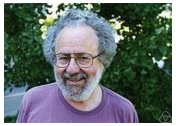
Back to Hilbert's Tenth Problem

Computational Problem: Given a Diophantine equation, does it have a solution over the integers?

$$L = \{\mathrm{p} = \mathrm{a}_1\mathrm{x}_1^{\mathrm{c}1} + \cdots + \mathrm{a}_\mathrm{n}\mathrm{x}_\mathrm{n}^{\mathrm{cn}} | \exists v_1 \ldots v_n \in N \; s.t. \, p(x_1 \ldots x_n) = 0\}$$

- L is Turing-recognizable

- L is not decidable (1949-70)




TM Variants

How Robust is the TM Model?

Does changing the model result in different languages being recognizable / decidable?

Short answer: No….

Longer answer:

Extensions that do not increase the power of the TM model

- TMs that are allowed to "stay put" instead of moving left or right

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

Proof that TMs with "stay put" are no more powerful:

Simulation: Convert any TM M with "stay put" into an equivalent TM M' without

$L(M) = L(M')$

Replace every "stay put" instruction in M with a move right instruction, followed by a move left instruction in M'

more formally:

$M = (Q, q_0, q_a, q_r, \Gamma, \Sigma, \delta)$

$M' = (Q', q_0', q_a', q_r', \Gamma, \Sigma, \delta')$

$Q' = Q \times \{0, 1\}$

$q \to (q, 0), (q, 1)$

$\delta(q, s) \to (p, s', L/R)$

$\delta'(q_0, s) \to (p, s', L/R)$

If $\delta(q, s) = (p, s', S)$

$\to \begin{cases} \delta'(q_0, s) = (p_1, s', L) \\ \delta'(p_1, s) = (p_0, s, R) \end{cases}$

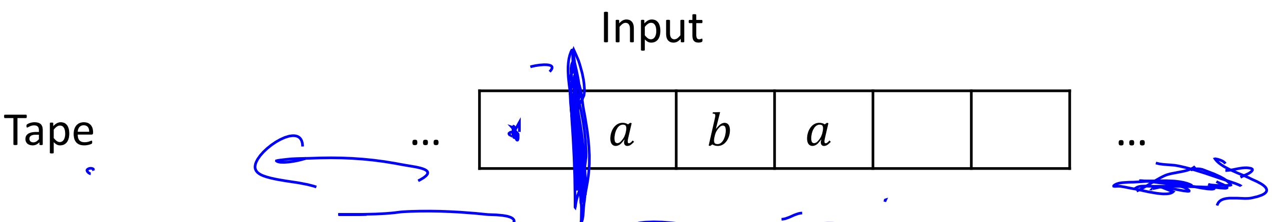
Proof of correctness:

- $x \in L(M) \Rightarrow x \in L(M')$

  $x \in L(M') \Rightarrow x \in L(M)$

Extensions that do not increase the power of the TM model

- TMs with a 2-way infinite tape, unbounded left to right

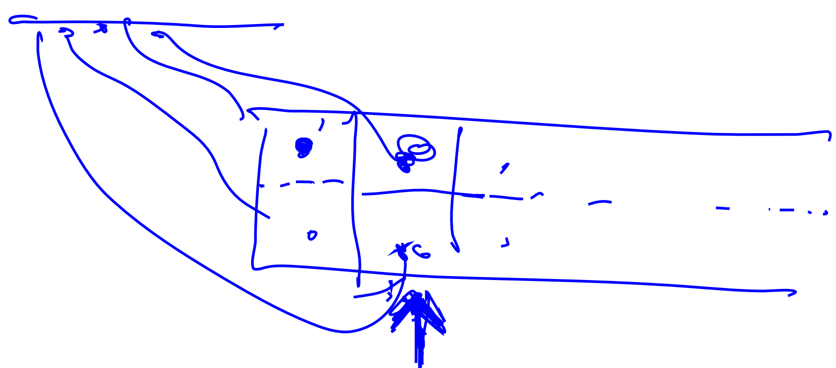


Proof that TMs with 2-way infinite tapes are no more powerful:

Simulation: Convert any TM M with 2-way infinite tape into a 1-way infinite TM M' with a "two-track tape"

$\Gamma' = \Gamma \times \Gamma$

$Q' = Q \times \{L, R\}$



Formalizing the Simulation

$M' = (Q', \Sigma, \Gamma', \delta', q_0', q'_{\text{accept}}, q'_{\text{reject}})$

New tape alphabet: $\Gamma' = (\Gamma \times \Gamma) \cup \{\$\}$

New state set: $Q' = Q \times \{+, -\}$

$(q, -)$ means "q, working on upper track"

$(q, +)$ means "q, working on lower track"

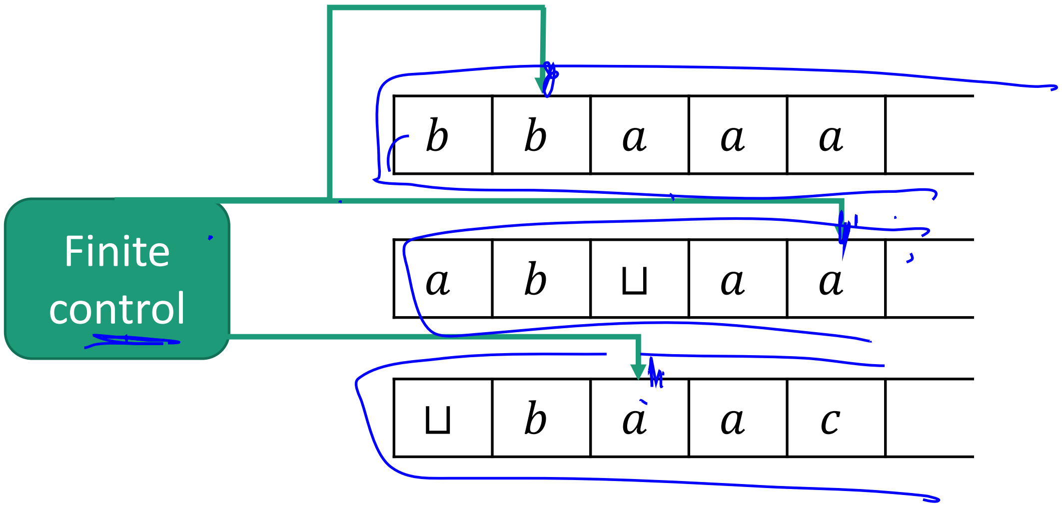
New transitions:

If $\delta(p, a_-) = (q, b, L)$, let $\delta'((p, -), (a_-, a_+)) = ((q, -), (b, a_+), R)$

Also need new transitions for moving right, lower track, hitting \$,

initializing input into 2-track format

Multi-Tape TMs

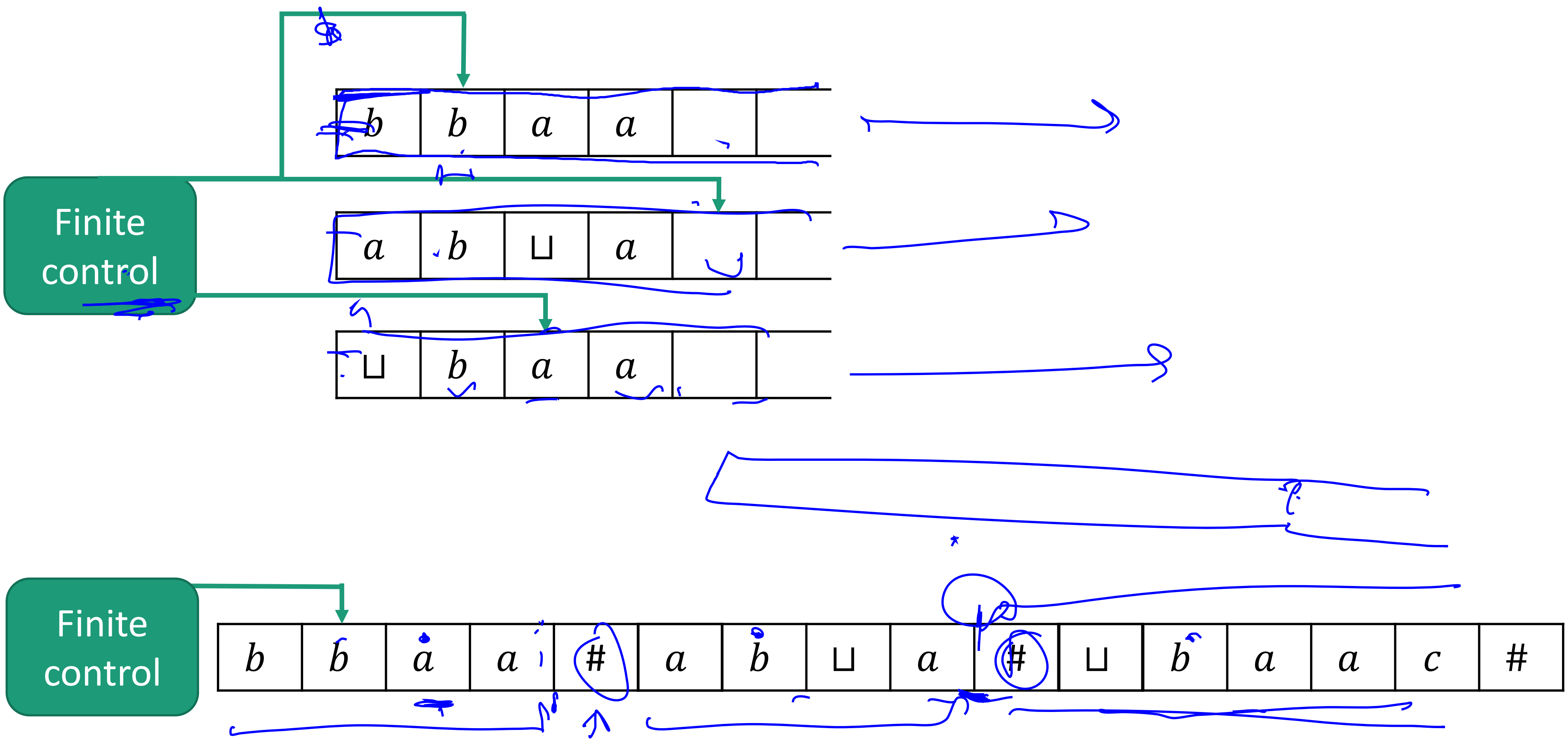


Fixed number of tapes k (can't change during computation)

Transition function $\delta : Q \times \Gamma^k \rightarrow Q \times \Gamma^k \times \{L, R, S\}^k$

Multi-Tape TMs are Equivalent to Single-Tape TMs

Theorem: Every k-tape TM M with can be simulated by an equivalent single-tape TM M'



$\Gamma' = \Gamma \times \{$no hd, yes hd$\}$ tape

Expen

Simulating Multiple Tapes

Implementation-Level Description

On input $w = w_1 w_2 \dots w_n$

1. Format tape into $\# \dot{w}_1 w_2 \dots w_n \# \dot{\sqcup} \# \dot{\sqcup} \# \dots \#$
2. For each move of M:

    Scan left-to-right, finding current symbols

    Scan left-to-right, writing new symbols,

    Scan left-to-right, moving each tape head

    If a tape head goes off the right end, insert blank

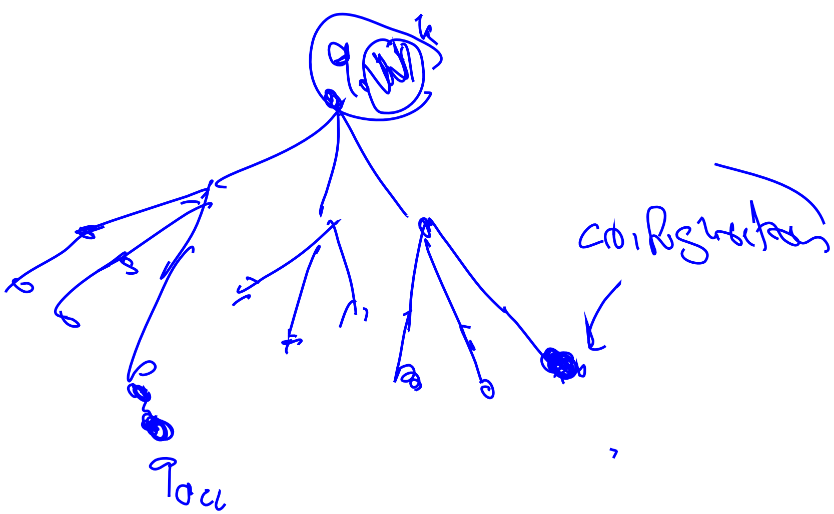
    If a tape head goes off left end, move back right

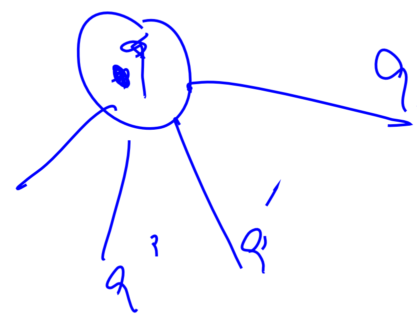
Non-deterministic TMs

At any point in computation, may non-deterministically branch. Accepts iff there exists an accepting branch.

Transition function $\delta : Q \times \Gamma \rightarrow P(Q \times \Gamma \times \{L, R, S\})$

Ex. NTM for $\{w \mid w$ is a binary number representing the product of two positive integers $a, b\}$

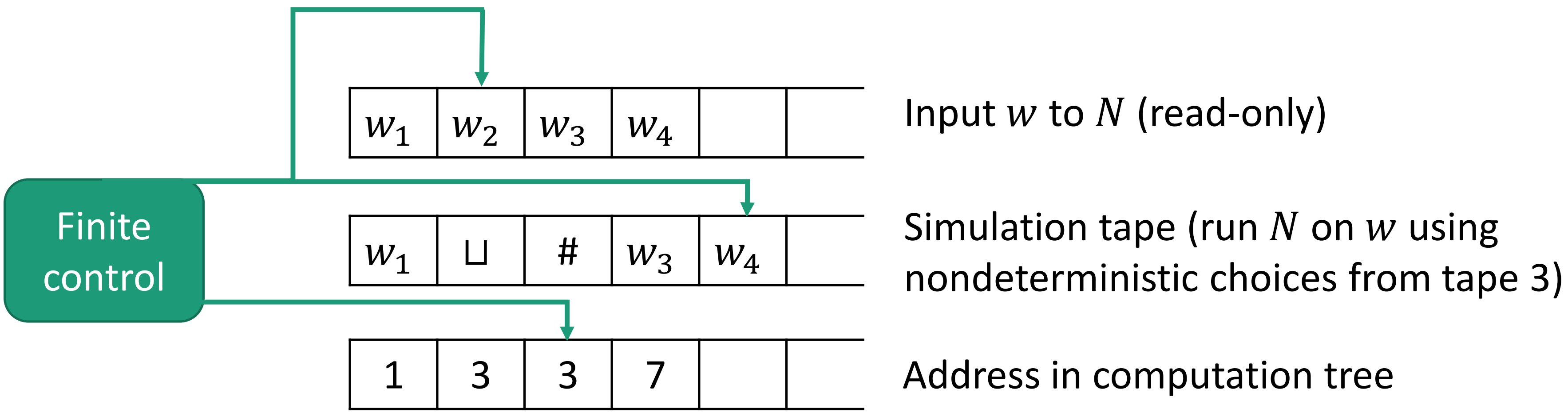




Non-deterministic TMs

Theorem: Every nondeterministic TM has an equivalent deterministic TM

Proof idea: Simulate an NTM N using a 3-tape TM



TMs are equivalent to…

- TMs with "stay put"
- TMs with 2-way infinite tapes
- Multi-tape TMs
- Nondeterministic TMs
- Random access TMs
- Enumerators
- Finite automata with access to an unbounded queue = 2-stack PDAs
- Primitive recursive functions
- Cellular automata

…

Church-Turing Thesis

The equivalence of these models is a mathematical theorem

Church-Turing *Thesis*: Each of these models captures our intuitive notion of algorithms

The Church-Turing Thesis is not a mathematical statement!