Turing Machines

Frank

January 19, 2022

1 Turing Machines (Informal)

1.1 Definition of a Problem

- Define input $\rightarrow$ output as a "problem", for example
- Input: $n \in \mathbb{N}$

  Output: $n^2 \in \mathbb{N}$
- Input: undirected connected weighted graph

  Output: a minimum spanning tree of the graph
- Input: $n \in \mathbb{N}$

  Output: yes if n is prime, no otherwise
- Thus a **problem** is an input-output relation
- An **instance** of a problem is a particular input of that problem
- A **decision problem** is a problem where output $\in$ {yes, no} || {true, false} || {0, 1}

  Then we define **yes-instances** and **no-instances** to be the different outputs
- Then computing a problem is essentially evaluating a function

1.2 Turing Machine

- A turing machine, TM is a mathematically model that represents solving of a problem "mechanically".

  Note that we cannot give a machine a number, but rather we can represent the number with something else (such as binary)

1.2.1 Decision Problems & Formal Languages

- A decision problem is the set of representation of yes-instances of a problem

  Consider the problem: is n a prime?

  Then the set of yes-instances of the problem is $L_p = \{10, 11, 101, 111, \ldots\}$ in binary
- A formal language is a set of strings over an alphabet Σ
- You can turn any problem into a series of decision problems (yes no questions)

1.2.2 Solving a Problem "Mechanically"

- A turing machine:

  Given a representation of input, we want to produce a representation of the appropriate output

  By following a finite sequence of steps, each of which can be obviously carried out

  This is what a modern computer or a turing machine does

1.2.3 What does it look like?

- One way **infinite tape** divided into cells, plus a **finite control** (finite number of states) that controls a **tape head** that allows read/write on the cells of the tape.

  Each cell of the tape contains one symbol from a finite alphabet Σ including a special symbol "blank" ($\sqcup$)

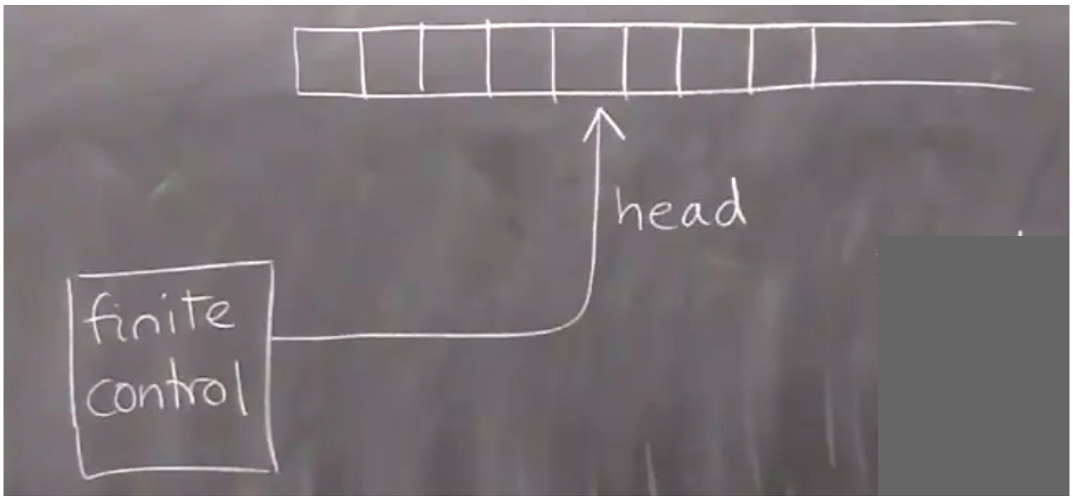


Figure 1: l21

  At any point in time, the head scans exactly one cell.

- At any state, the TM, given the present state, the present symbol scanned by its head may change its state, change the present symbol, and must move either right or left

  On the leftmost cell, if the instruction is to move left, the head will stay in the same place

1.2.4 Computation of a Turing Machine

- Start with some input string x loaded onto the tape, followed by "blanks". Note that the EOF is indicated by the blank, thus the input must **not** contain any blanks.

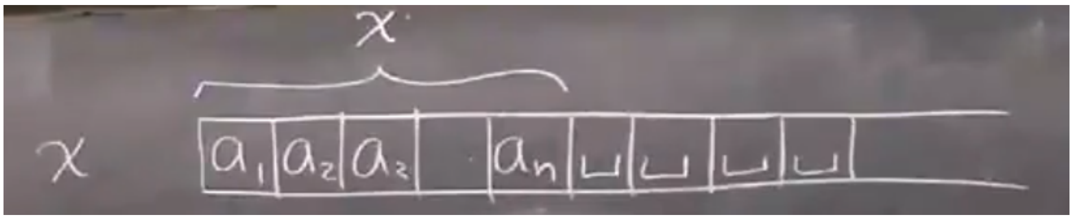


Figure 2: l22

- Then the control will progress between states (start at initial state q_0, with head at the leftmost cell), $q_1, q_2, q_3, \ldots$

  The control will keep switching between states until we enter a special state q_{accept} or q_{reject}, when we either accept the input string or reject the input string

  Note that we don't even have to read the entire input string to go into an accept or reject state

- The machine **halts** when we go eventually go into an accepting state or rejecting state. The machine **loops** if it never halts

- Then the **Language of the TM** is the set of input strings that leads to the accepting state

2 Turing Machines (Formal)

2.1 Definition

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

- Q = finite set of states
- Σ = alphabet, finite set of symbols that does not include $\sqcup$ (input alphabet)
- $\Gamma \supseteq \Sigma$ = alphabet, including $\sqcup$ (tape alphabet)
- $\delta : (Q - \{q_a, q_r\} \times \Gamma) \to (Q \times \Gamma \times \{L, R\})$ (transition function)

  $\delta(q, a) = (p, b, L/R)$

  if M is in state q and its head scanning cell with symbol a, then enter state p, write symbol b on tape cell, and move head one cell to the left or to the right

  Note that δ is also finite by definition. It has to be finite because it is a set of instructions that defines what the machine should do at every configuration (state, tape content, and head location).

- $q_0 \in Q$ Initial state
- $q_a \in Q$ Accept state
- $q_r \in Q - \{q_a\}$ Reject state

2.2 Computation (Formally)

- Define **configuration** ("instantaneous description" ID)
  - Current state
  - Contents of the tape (except the infinitely many trailing $\sqcup$)
  - Position of head (first symbol of y)

  Formally, a configuration is a string of the form

  $$xqy$$

  Where x and y are members of $\Gamma^\star$, s.t. y does not end with $\sqcup$ and $q \in Q$

  Note that y could be empty so that the head points to an empty string (end of the input)

- Define **relation** $\vdash_M$ between configurations of M

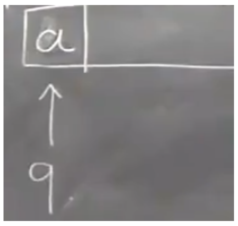
  $C \vdash_M C'$ ["yields"],
  then M moves from configuration C to configuration C' in <u>one step</u>

2.2.1 Aside: Notational Conventions

- $p, q \sim$ states
- $a, b, c, \ldots \in \Gamma$
- $v, w, x, y, z \in \Gamma^\star$
- $\sqcup$ blank

2.2.2 Aside: Formal Definition of Yield

- Suppose $C = xqy$
  - If $y = ay'$ (i.e. $y \neq \epsilon$) and $\delta(q, a) = (p, b, R)$
    Then $C \vdash C' \iff C' = xbpy'$
  - If $y = ay'$ (i.e. $y \neq \epsilon$) and $\delta(q, a) = (p, b, L)$ and $x = x'c$ (i.e. $x \neq \epsilon$)
    Then $C \vdash C' \iff C' = x'pcby'$
  - If $y = ay'$ (i.e. $y \neq \epsilon$) and $\delta(q, a) = (p, b, L)$ and $x = \epsilon$



    Then $C \vdash C' \iff C' = pby'$
  - If $y = \epsilon$ and $\delta(q, a) = (p, b, R)$
    Then $C \vdash C' \iff C' = xbp$
  - If $y = \epsilon$ and $\delta(q, a) = (p, b, L)$ and $x = x'c$ (i.e. $x \neq \epsilon$)
    Then $C \vdash C' \iff C' = x'pcb$

  - Define $\vdash_M^\star$ to be the transitive closure of $\vdash_M$, then

    $C \vdash^\star C'$ iff $C' = C$ or $\exists$ finite sequence of configurations $C_1, C_2, \ldots, C_k$ s.t. $C_1 = C$, $C_k = C'$, and $C_i \vdash C_{i+1}$ for all $1 \leq i < k$

Back to Computation...

- M **accepts** the string $x \in \Sigma^\star \iff q_0x$ (initial configuration) $\vdash^\star yq_az$ (accepting configuration)
- M **rejects** the string $x \in \Sigma^\star \iff q_0x$ (initial configuration) $\vdash^\star yq_rz$ (rejecting configuration)
- If M either accepts x or rejects x, then M **halts** on x.

  M **loops** on $x \iff M$ does **not** halt on x. i.e. $\exists$ infinite number of configurations $C_1, C_2, \ldots$ s.t. $q_0x \vdash C_1 \vdash C_2 \vdash \ldots$
- The language $L(M)$ <u>accepted</u> (or recognized) by TM M:

  $$L(M) = \{x \in \Sigma^\star : M \text{ accepts } x\}$$

  This language is the description of a decision problem, where every single string in $L(M)$ encodes a "yes" instance of the problem

  For example, if we have a TM that identifies whether a number is prime or not, then $L(M)$ is the set of prime numbers.
- Note that M does not accept $x \not\Longrightarrow M$ rejects x

  But rather $\implies M$ rejects x or M loops on x
- Thus, a "no" instance to our problem might indicate rejection or looping.
- Language L is **recognizable** iff $\exists$ TM M (**recognizer**) s.t. $L(M) = L$
- Language L is **decidable** iff $\exists$ TM M (**decider**) s.t. $L(M) = L$ and M halts on <u>every</u> input
- L decidable $\implies L$ recognizable by def, but the converse is false

2.2.3 Aside: Alternative Terminology

- Decidable - Recursive
- Recognizable - Semi-decidable - Recursively enumerable (RE)
- Also say <u>set/decision problem</u> is decidable/recognizable

2.3 Turing Machine Examples

- Example: TM that decides

  $$L = \{x \in \Sigma^\star : x \text{ is an even-length palindrome}\}$$

  For $\Sigma = \{0, 1, 2\}$

  We should go from end to end, checking that we have the same symbol on either end, replacing each symbol with a blank as we check it off. Continue this process until we find a mismatch or we have an odd number of symbols.

2.3.1 Procedure

1. If symbol under head is $\sqcup$, then accept (empty string is a palindrome, and detects even palindromes only). Otherwise, "remember" that symbol, replace it by $\sqcup$, and move R.
2. While symbol scanned $\neq \sqcup$, move R
3. Move one cell to the L (we overshot)
4. If symbol under head is not the same as the one "remembered", then reject. Otherwise, replace it by $\sqcup$ and move to L
5. While symbol scanned $\neq \sqcup$, move L
6. Move one cell to the R (we overshot again) and go to stage 1

2.3.2 Formal Description

- States of M:
  - q_0 initial state
  - q_a, q_r accept/reject states
  - $[q_1, a]$ for all $a \in \Sigma$ (q_1 = Move R, first symbol was a)
  - $[q_2, a]$ for all $a \in \Sigma$ (q_2 = reached right end, and first symbol seen was a)
  - q_3 (keep moving L)
- $Q = \{q_0, q_a, q_r, q_3\} \cup \{[q_1, a] : a \in \Sigma\} \cup \{[q_2, a] : a \in \Sigma\}$

  Note that we need states to be finite, thus we cannot do something like $\{[q_1, n] : n \in \mathbb{N}\}$, which would be infinite. The set Σ is finite, but $\mathbb{N}$ is infinite
- $\Sigma = \{0, 1, 2\}$
- $\Gamma = \{0, 1, 2, \sqcup\}$
- δ:

  $$\delta(q_0, a) = \begin{cases} (q_a, \sqcup, R) & \text{if } a = \sqcup \\ ([q_1, a], \sqcup, R) & \text{otherwise} \end{cases}$$

  Initially, if we see an empty symbol, then the input string must be empty, so we accept.

  However, if we do not see an empty symbol, then we go into q_1, which is to move right. We remember the symbol we saw, which is a, and we replace it by $\sqcup$, then we move right.

  $$\delta([q_1, b], a) = \begin{cases} ([q_1, b], a, R) & \text{if } a \neq \sqcup \\ ([q_2, b], \sqcup, L) & \text{if } a = \sqcup \end{cases}$$

  While we are moving right (remembering some symbol b on the leftmost end of the tape), if we do not see an $\sqcup$, then we keep moving right, not changing anything (the symbol a under our head still remains a).

  If we do see an $\sqcup$, that means we've reached the rightmost end of the tape, so we enter q_2 (reached right end), leave the $\sqcup$ alone, and move left so our head points to the rightmost symbol on the tape.

$$\delta([q_2, b], a) = \begin{cases} (q_r, \sqcup, L) & \text{if } a \neq b \\ (q_3, \sqcup, L) & \text{if } a = b \end{cases}$$

If we find that the rightmost symbol is not equal to the symbol we remembered, $(a \neq b)$, then we reject because the string cannot be a palindrome.

Otherwise, we confirm that the symbols on the two ends match, and we replace the rightmost symbol with $\sqcup$, and go into q_3 (move left)

$$\delta(q_3, a) = \begin{cases} (q_3, a, L) & \text{if } a \neq \sqcup \\ (q_0, \sqcup, R) & \text{if } a = \sqcup \end{cases}$$

Note that we do not need to remember anything while moving left because we are simply returning to the starting position. So we just simply move left and not change anything until we reach $\sqcup$, then we move right to end up at the leftmost position.

2.3.3 Computation of TM on 0110

- According to the δ we defined above, we can trace the TM as follows

$$\begin{aligned} q_0 0110 &\vdash \sqcup[q_1,0]110 \\ &\vdash \sqcup 1[q_1,0]10 \\ &\vdash \sqcup 11[q_1,0]0 \\ &\vdash \sqcup 110[q_1,0] \\ &\vdash \sqcup 11[q_2,0]0 \\ &\vdash \sqcup 1 q_3 1 \\ &\vdash \sqcup q_3 11 \\ &\vdash q_3 \sqcup 11 \\ &\vdash \sqcup q_0 11 \\ &\vdash \sqcup\sqcup[q_1,1]1 \\ &\vdash \sqcup\sqcup 1[q_1,1] \\ &\vdash \sqcup\sqcup[q_2,1]1 \\ &\vdash \sqcup q_3 \\ &\vdash \sqcup\sqcup q_0 \\ &\vdash \sqcup\sqcup\sqcup q_a \end{aligned}$$

2.3.4 Computation of TM on 010

- According to the δ we defined above, we can trace the TM as follows

$$\begin{aligned} q_0 010 &\vdash \sqcup[q_1,0]10 \\ &\vdash \sqcup 1[q_1,0]0 \\ &\vdash \sqcup 10[q_1,0] \\ &\vdash \sqcup 1[q_2,0]0 \\ &\vdash \sqcup q_3 1 \\ &\vdash q_3 \sqcup 1 \\ &\vdash \sqcup q_0 1 \\ &\vdash \sqcup\sqcup[q_1,1] \\ &\vdash \sqcup[q_2,1] \\ &\vdash q_r \end{aligned}$$

3 Variants of Turing Machines (Equivalencies)

- We can extend the TM in many ways, including having multiple tapes, a 2-d tape, etc. But note that **none** of these extensions of the TM are <u>more powerful</u> than the most basic TM. However, the extension of the TM may prove useful in certain problems.

3.1 Multitape TMs

- Consider the diagram sketching the TM M:

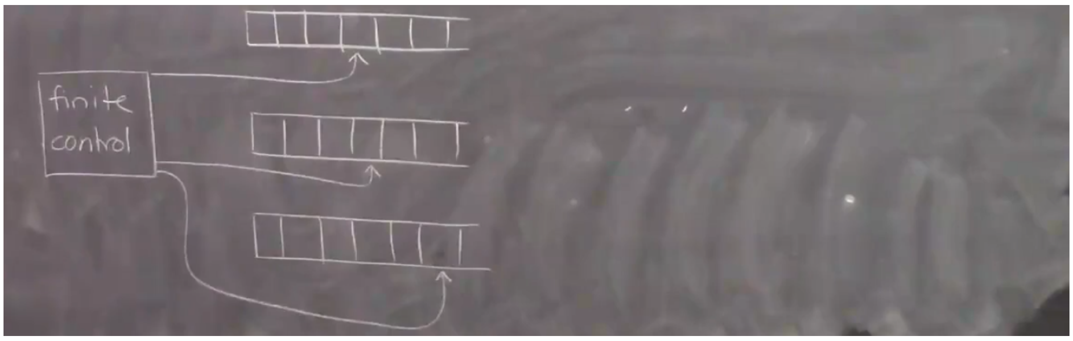


Figure 3: l24

Then we use the present state plus the vector of symbols on the k tapes to define δ. Then we can control each head independently.

When we load the input string, load as usual onto the first tape, then leave all empty spaces on every other tape.

- **Thm 2.1**: If M is a Multitape TM, $\exists$ a TM M' s.t. $L(M') = L(M)$
- Proof: Consider a diagram of M'

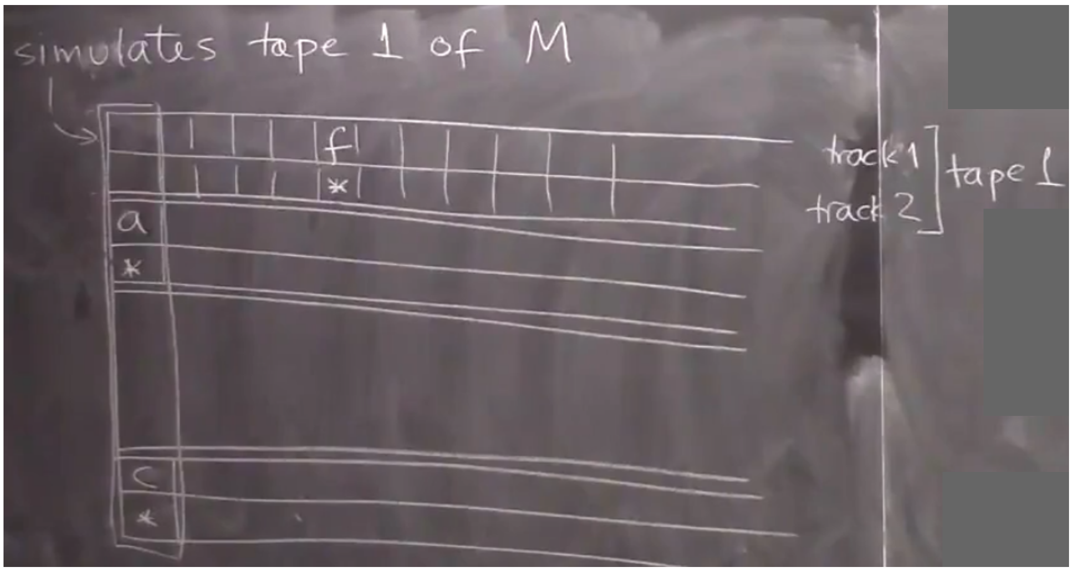


Figure 4: l25

We have 1 tape with many tracks (exactly $2k$ tracks. 2 tracks per tape of M). Each track is divided into cells as a usual tape. The first track simulates the corresponding tape in M, the second track will be mostly empty, except it has a $\star$ that indicates the head of the first tape in M is on this corresponding cell.

Think of this as an array on a really fat tape. This is possible because a TM can have a tape with infinitely many cells. The contents of the cell doesn't have to be a simple symbol. In this case, it is actually a vector of size $2k$. This means our head can still scan over this entire vector in one go.

We want to start with an empty vector of size k, then as we sweep across the tape, we want to "fill up" the vector with content of cells that M is currently pointing to with each head (the head locations is indicated by a $\star$). Our purpose is to fill up this vector, so we know the exact state of M

To reflect changes on M, we do a reverse sweep and for each pair of tracks, we reflect the appropriate changes on the cells and the heads of M.

After M' is finished with the above, it has simulated 1 move of M, so we can keep repeating this double sweeping action.

3.1.1 Simulation of one move of M by M'

- Remember M's state
- Head starts from leftmost cell of M'
- Scan the multitrack tape of M' to the R, remembering in the state of M' all symbols above the $\star$'s until you have seen all k $\star$'s

- Determine the symbols to be written on the cell of each tape under its head (use state transition function of M). Remember that in our state (using another vector of length k like the "fill up" vector we used above).
- M' scans its multitrack (single) tape to the L, performing the appropriate action as it encounters every $\star$. Accept or reject if M does.
- **Thm 2.1**: $L(M') = L(M)$ If k-tape TM M accepts/rejects x in m moves, the single tape TM M' accepts/rejects x in $O(m^2)$ moves

  If M makes m moves, its heads are $\leq m$ apart.

  The R moves of M' to simulate one move of M take $\leq m$ steps.

  The L moves of M' to simulate one move of M take $\leq m + 2k$ steps, because we might move the head back and forth to simulate changing head position and changing symbol on M (we change at most $2k$ times)
- Thus, one move of $M = 2(m + k)$ moves of $M' \implies m$ moves of $M = O(m^2)$ moves of M'

3.2 Nondeterministic TMs (NTM)

- To be continued...