Turing Machines

Reading: Chapter 8

Turing Machines are…

- Very powerful (abstract) machines that could simulate any modern day computer (although very, very slowly!)

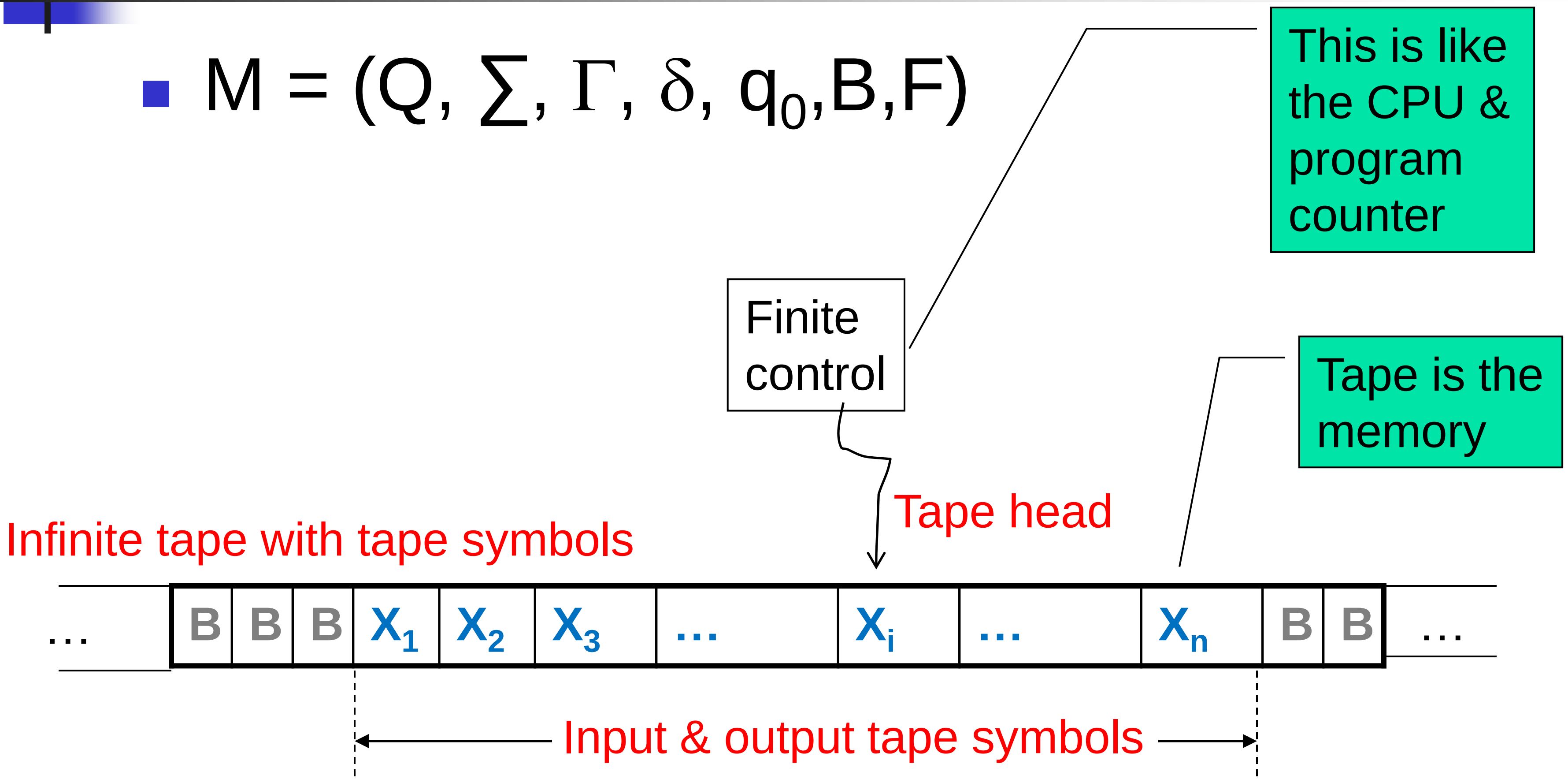
- Why design such a machine?
  - If a problem cannot be "solved" even using a TM, then it implies that the problem is ***undecidable***

For every input, answer YES or NO

- Computability vs. Decidability

A Turing Machine (TM)

- $M = (Q, \sum, \Gamma, \delta, q_0,B,F)$



B: blank symbol (special symbol reserved to indicate data boundary)

You can also use:

→ for R

← for L

Transition function

- One move (denoted by |---) in a TM does the following:
  - $\delta(q,X) = (p,Y,D)$
    - q is the current state
    - X is the current tape symbol pointed by tape head
    - State changes from q to p
    - <u>After the move:</u>
      - X is replaced with symbol Y
      - If D="L", the tape head moves "left" by one position.
        Alternatively, if D="R" the tape head moves "right" by one position.

ID of a TM

- Instantaneous Description or ID :
  - $X_1X_2\ldots X_{i-1}qX_iX_{i+1}\ldots X_n$
    means:
    - q is the current state
    - Tape head is pointing to X_i
    - $X_1X_2\ldots X_{i-1}X_iX_{i+1}\ldots X_n$ are the current tape symbols

- $\delta(q,X_i) = (p,Y,R)$ is same as:
  $X_1\ldots X_{i-1}qX_i\ldots X_n$ |---- $X_1\ldots X_{i-1}YpX_{i+1}\ldots X_n$
- $\delta(q,X_i) = (p,Y,L)$ is same as:
  $X_1\ldots X_{i-1}qX_i\ldots X_n$ |---- $X_1\ldots pX_{i-1}YX_{i+1}\ldots X_n$

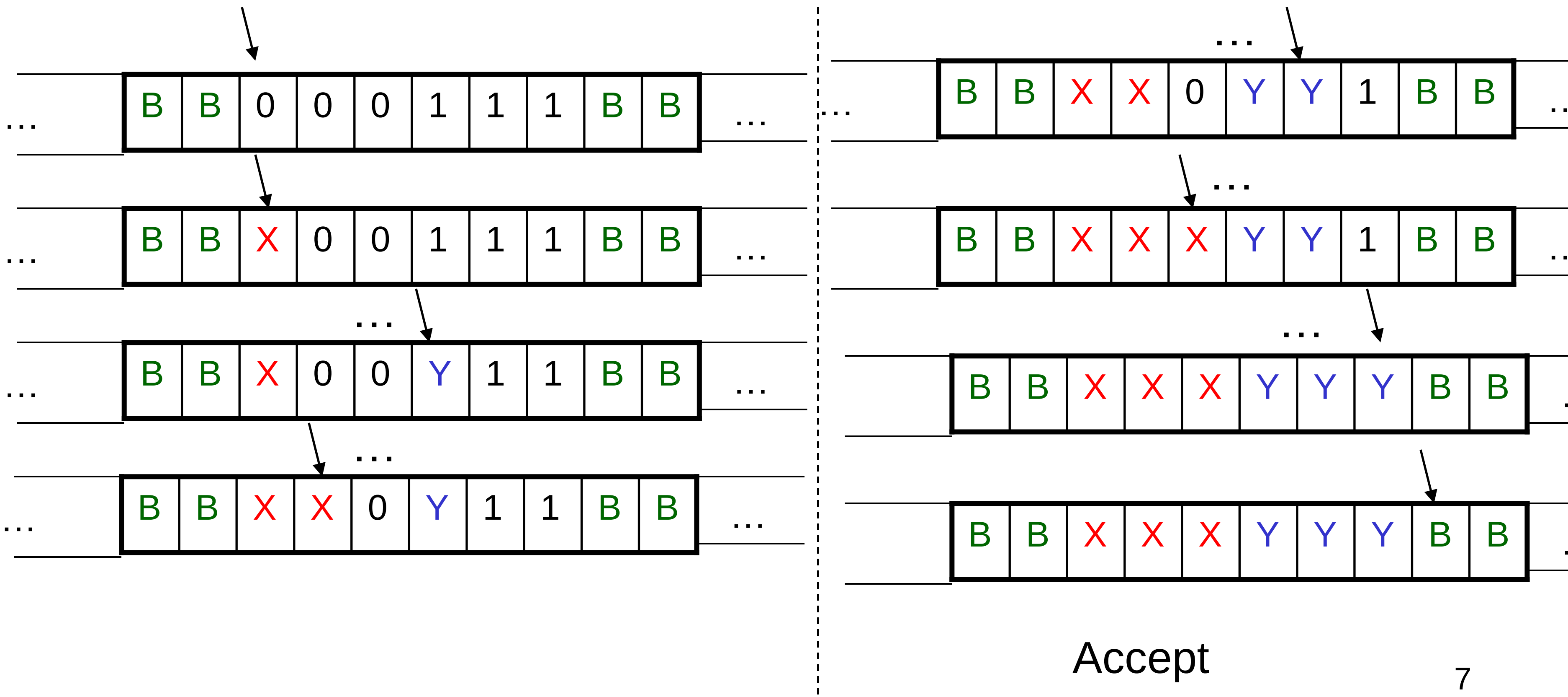
Way to check for Membership

- Is a string *w* accepted by a TM?

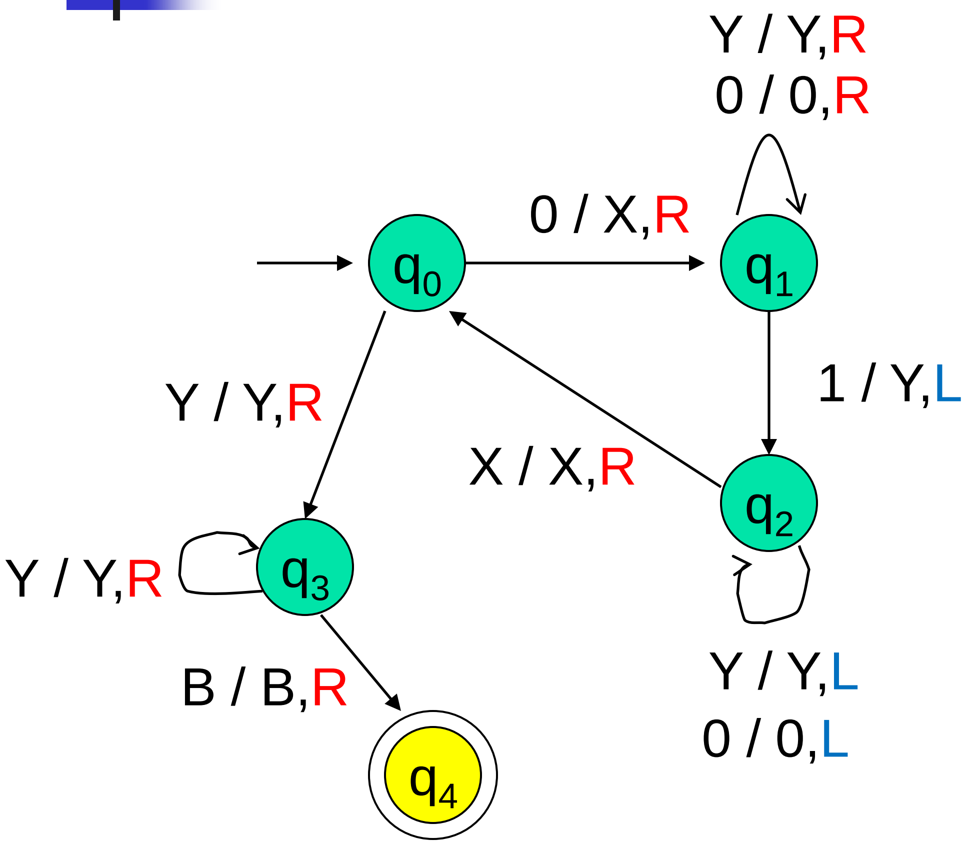
- <u>Initial condition:</u>
  - The (whole) input string *w* is present in TM, preceded and followed by infinite blank symbols
- <u>Final acceptance:</u>
  - Accept *w* if TM enters <u>final state</u> and halts
  - If TM halts and not final state, then reject

Example: L = $\{0^n1^n \mid n \geq 1\}$

- Strategy: w = 000111



TM for $\{0^n1^n \mid n \geq 1\}$



1. Mark next unread 0 with X and move right
2. Move to the right all the way to the first unread 1, and mark it with Y
3. Move back (to the left) all the way to the last marked X, and then move one position to the right
4. If the next position is 0, then goto step 1.
   Else move all the way to the right to ensure there are no excess 1s. If not move right to the next blank symbol and stop & accept.

*state diagram representation preferred

TM for $\{0^n1^n \mid n \geq 1\}$

	Next Tape Symbol				
Curr. State	0	1	X	Y	B
→ q_0	(q_1,X,R)	-	-	(q_3,Y,R)	-
q_1	$(q_1,0,R)$	(q_2,Y,L)	-	(q_1,Y,R)	-
q_2	$(q_2,0,L)$	-	(q_0,X,R)	(q_2,Y,L)	-
q_3	-	-	-	(q_3,Y,R)	(q_4,B,R)
$*q_4$	-	--	-	-	-

Table representation of the state diagram

TMs for calculations

- TMs can also be used for calculating values
  - Like arithmetic computations
  - Eg., addition, subtraction, multiplication, etc.

Example 2: monus subtraction

"m -- n" = max{m-n,0}

$0^m 1 0^n \rightarrow$ ...B 0^{m-n} B.. (*if m>n*)
...BB…B.. (*otherwise*)

1. For every 0 on the left (mark X), mark off a 0 on the right (mark Y)
2. Repeat process, until one of the following happens:
   1. // No more 0s remaining on the left of 1
      Answer is 0, so flip all excess 0s on the right of 1 to Bs (and the 1 itself) and halt
   2. //No more 0s remaining on the right of 1
      Answer is m-n, so simply halt after making 1 to B

Give state diagram

Example 3: Multiplication

- $0^m 1 0^n 1$ (input), $0^{mn}1$ (output)

- Pseudocode:
  1. Move tape head back & forth such that for every 0 seen in 0^m, write n 0s to the right of the last delimiting 1
  2. Once written, that zero is changed to B to get marked as finished
  3. After completing on all m 0s, make the remaining n 0s and 1s also as Bs

Give state diagram

Calculations vs. Languages

A "calculation" is one that takes an input and outputs a value (or values)

The "language" for a certain calculation is the set of strings of the form "<input, output>", where the output corresponds to a valid calculated value for the input

A "language" is a set of strings that meet certain criteria

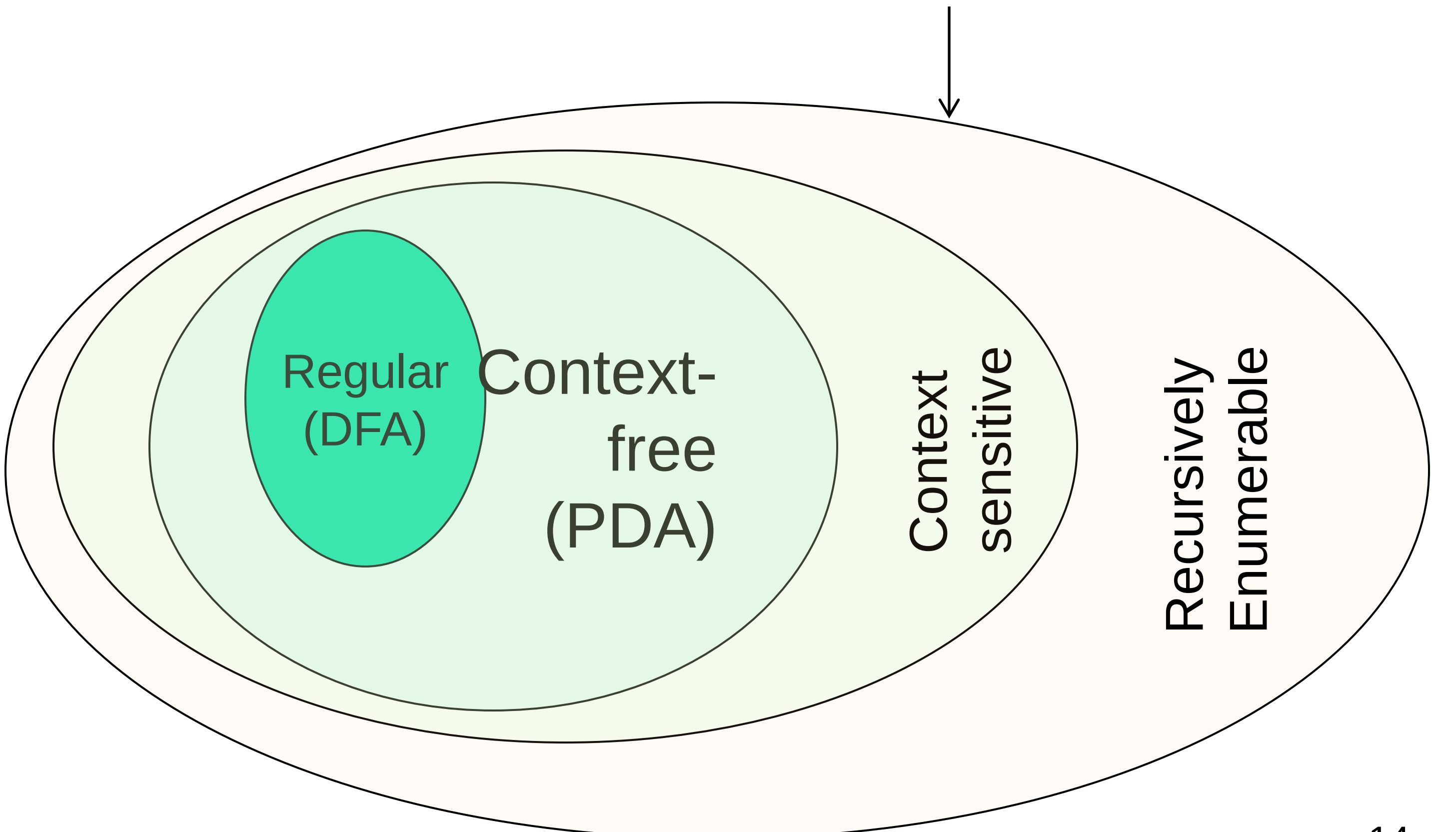
E.g., The language L_{add} for the addition operation

"<0#0,0>"
"<0#1,1>"
…
"<2#4,6>"
…

Membership question == verifying a solution
e.g., is "<15#12,27>" a member of L_{add} ?

Language of the Turing Machines

- *Recursive Enumerable (RE) language*

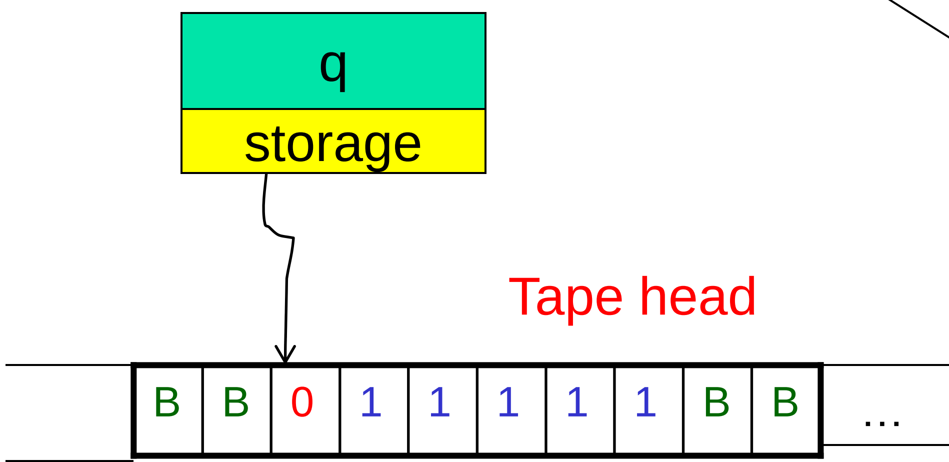


Variations of Turing Machines

TMs with *storage*

Generic description
Will work for both a=0 and a=1

- E.g., TM for 01* + 10*



Current state

Current Storage symbol

Tape symbol

Next state

New Storage symbol

Transition function δ:

- $\delta([q_0,B],a) = ([q_1,a], a, R)$
- $\delta([q_1,a],\bar{a}) = ([q_1,a], \bar{a}, R)$
- $\delta([q_1,a],B) = ([q_2,B], B, R)$

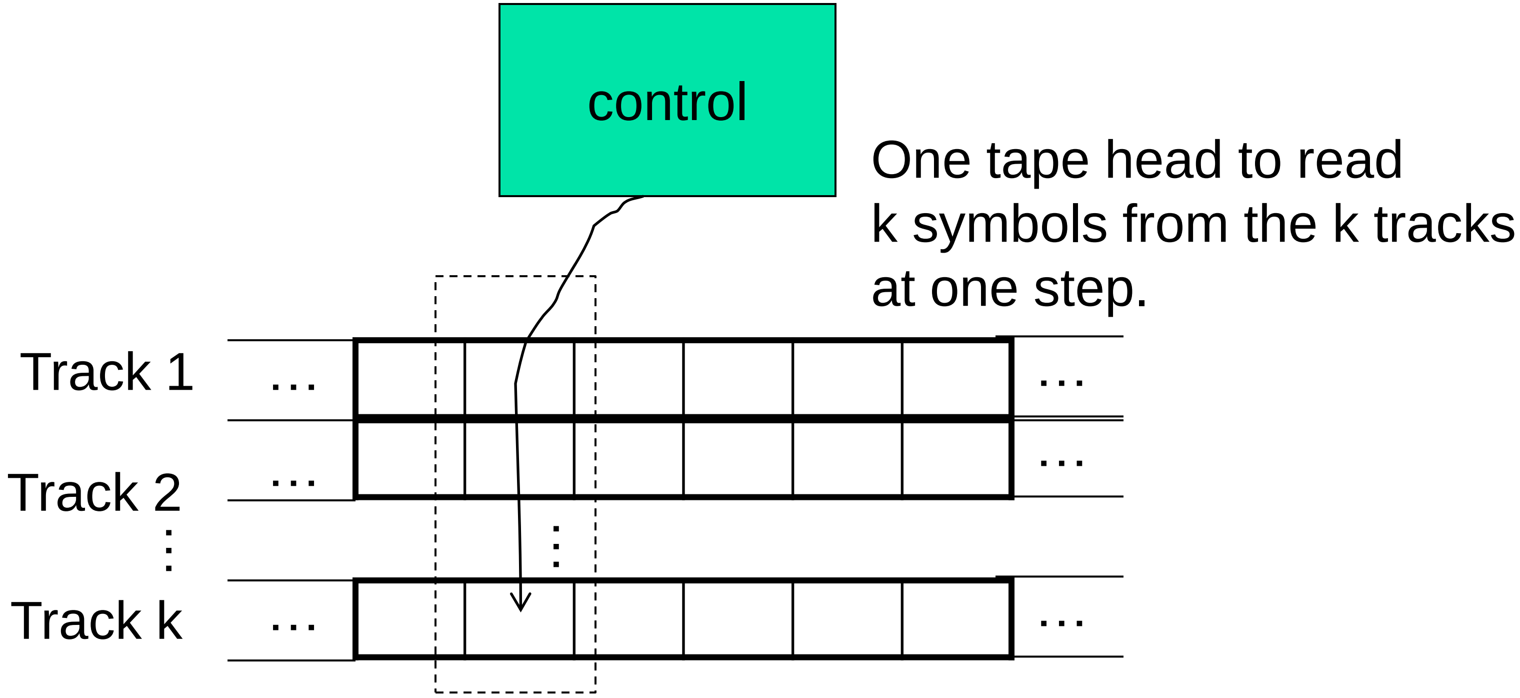
[q,a]: where q is current state,
a is the symbol in storage

Are the standard TMs equivalent to TMs with storage?

Yes

Multi-track Turing Machines

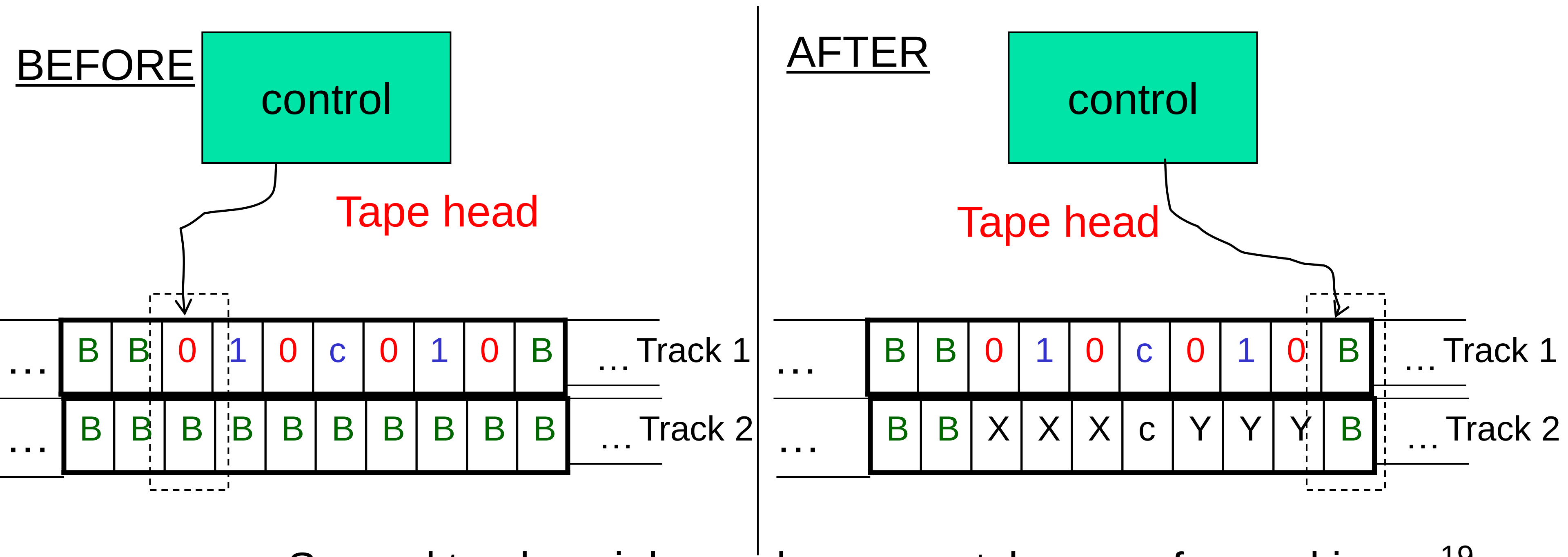
- TM with multiple tracks,
  but just one unified tape head



Multi-Track TMs

- TM with multiple "tracks" but just one head

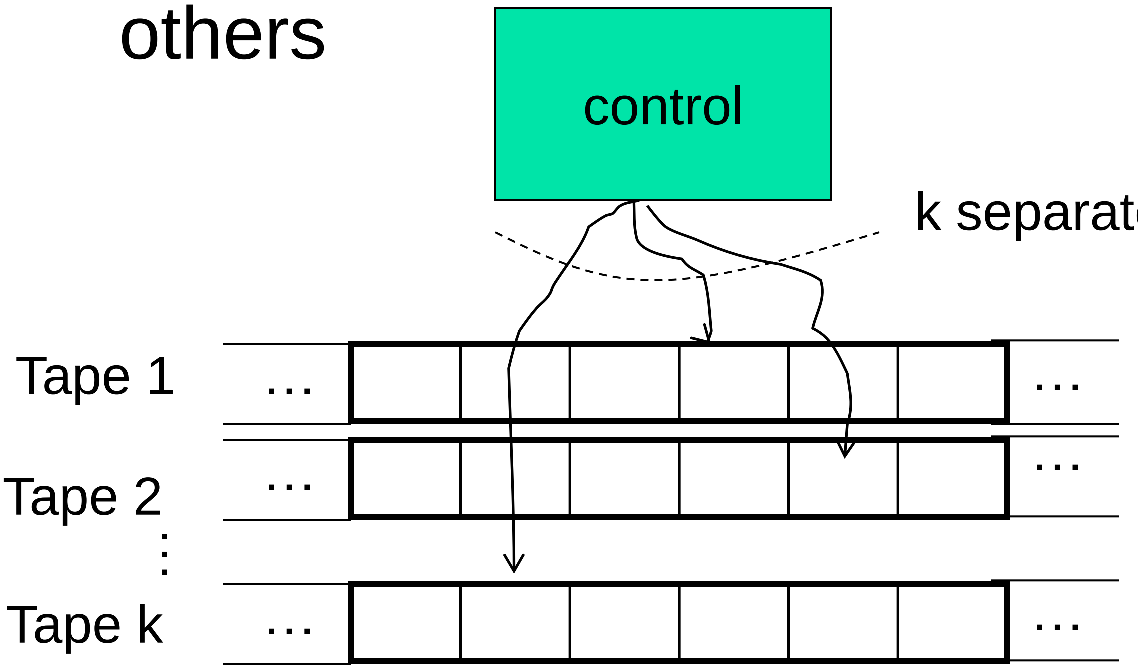
E.g., TM for {wcw | w∈ {0,1}* }
but w/o modifying original input string



Second track mainly used as a scratch space for marking

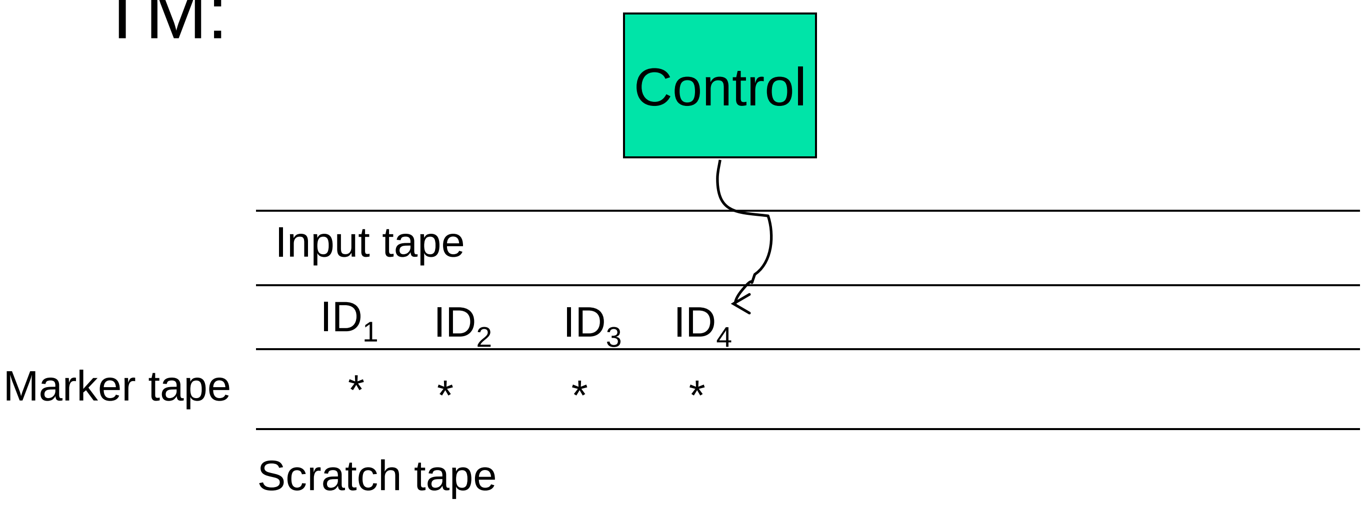
Multi-tape Turing Machines

- TM with multiple tapes, *each tape with a separate head*
  - Each head can move independently of the others



Non-deterministic TMs

- A TM can have non-deterministic moves:
  - $\delta(q,X) = \{ (q_1,Y_1,D_1), (q_2,Y_2,D_2), \ldots \}$
- Simulation using a multitape deterministic TM:



Unrestricted Grammar

- An unrestricted grammar or phrase structured grammar is 4 tuple G=(V, T, P, S), where V and T are disjoint set of variables and terminals, respectively. S is the element of V called the starting symbol; and P is the set of productions of the form

$$\alpha \rightarrow \beta$$

where $\alpha,\beta \in (V \cup T)^*$ and α contains at least one variable.

- Recall from CFG, the process of derivation from G

$$\alpha \Rightarrow^*_G \beta$$

means that β can be derived from α in zero or more steps.

- The language of G is formally defined as

$$L(G)=\{x \in T^* \mid S \Rightarrow^*_G x\}.$$

An Unrestricted Grammar for generating $\{a^n b^n c^n \mid n \geq 1\}$

- Let

  $L = \{a^n b^n c^n \mid n \geq 1\}$

- The grammar to generate L has the productions

$S \rightarrow FS_1$	$S1 \rightarrow ABCS_1$	$S1 \rightarrow ABC$
$BA \rightarrow AB$	$CA \rightarrow AC$	$CB \rightarrow BC$
$FA \rightarrow a$	$aA \rightarrow aa$	$aB \rightarrow ab$
$bB \rightarrow bb$	$bC \rightarrow bc$	$cC \rightarrow cc$

An Unrestricted Grammar for generating $\{a^n b^n c^n \mid n \geq 1\}$

- Let us consider the string *aabbcc*

- The derivation for this string is as fallows.

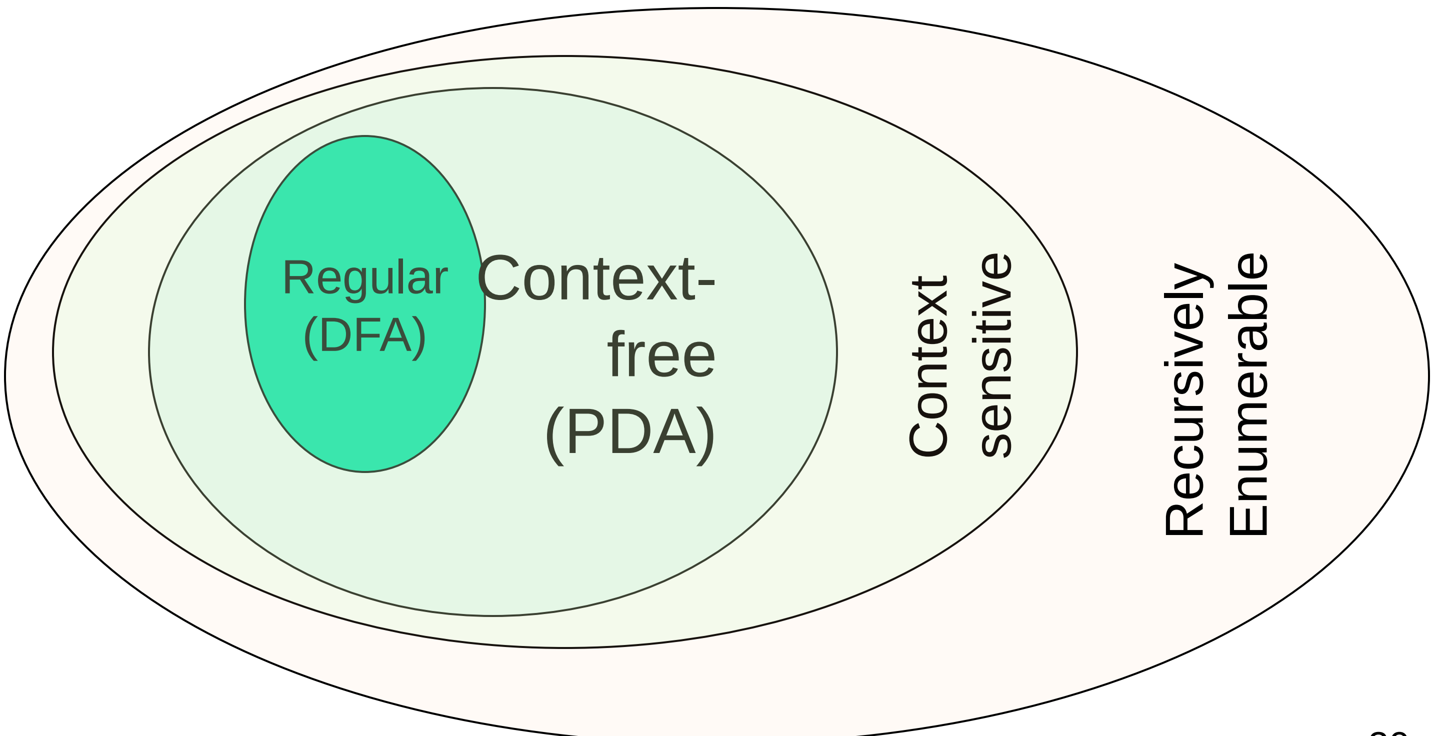
  $S \Rightarrow F\underline{S_1} \Rightarrow FABC\underline{S_1} \Rightarrow FAB\underline{CA}BC$

  $\Rightarrow FA\underline{BA}CBC \Rightarrow FAAB\underline{CB}C \Rightarrow \underline{FA}ABBCC$

  $\Rightarrow \underline{aA}BBCC \Rightarrow a\underline{aB}BCC \Rightarrow aa\underline{bB}CC$

  $\Rightarrow aab\underline{bC}C \Rightarrow aabb\underline{cC} \Rightarrow aabbcc$

Chomsky Hierarchy



Chomsky Hierarchy

Type	Languages	Forms of production in grammar	Accepting Device
3	Regular	$A \rightarrow aB$, $A \rightarrow a$ ($A,B \in V$, $a \in T$)	DFA
2	Context free	$A \rightarrow \alpha$ ($A \in V$, $\alpha \in (V \cup T)^*$)	PDA
1	Context Sensitive	$\alpha \rightarrow \beta$ ($\alpha, \beta \in (V \cup T)^*$, $\|\beta\| \geq \|\alpha\|$, α contains a variable)	LBA
0	Recursively Enumerable	$\alpha \rightarrow \beta$ ($\alpha, \beta \in (V \cup T)^*$, α contains a variable)	TM

Undecidability

Post's Correspondence Problem (PCP)

- An instance of PCP is called a correspondence system and consists of a set of pairs $(a_1, b_1), (a_2, b_2), \ldots\ldots (a_n, b_n)$, where a_i's and b_i's are non null strings over an alphabet Σ.

- The question we are interested in for an instance like this is whether there is a sequence of one or more integers $i_1, i_2, \ldots., i_k$, each i_k satisfying $1 \leq i_k \leq n$ and i_j's are not necessarily distinct, so that $a_{i1}, a_{i2}, \ldots\ldots, a_{ik} = b_{i1}, b_{i2}, \ldots\ldots., b_{ik}$.

Undecidability

- The instance is a yes-instance if there is a such a sequence, and we call the sequence a solution sequence for the instance.
- *15234434*

A	B
10	101
01	100
0	10
100	0
1	010

Summary

- TMs == Recursively Enumerable languages
- TMs can be used as both:
  - Language recognizers
  - Calculators/computers
- ***Basic TM is <u>equivalent</u> to all the below:***
  1. *TM + storage*
  2. *Multi-track TM*
  3. *Multi-tape TM*
  4. *Non-deterministic TM*
- TMs are like universal computing machines with unbounded storage