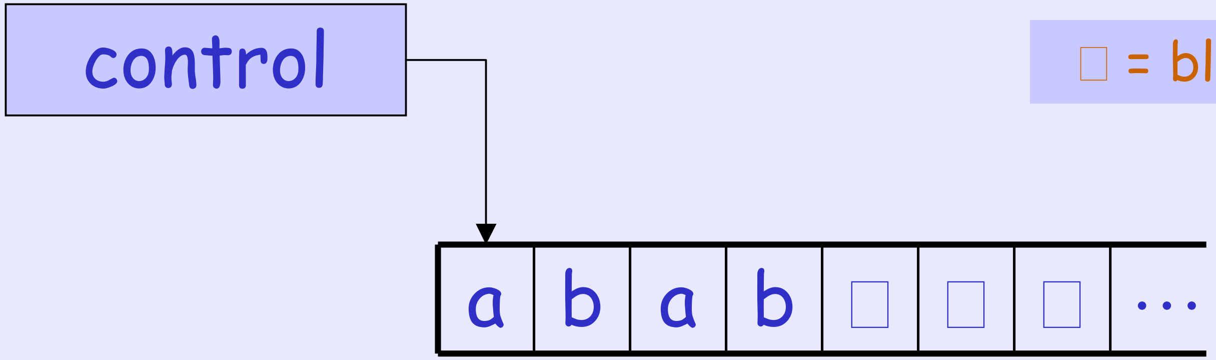
CS5371
Theory of Computation

Lecture 10: Computability Theory I
(Turing Machine)

Objectives

- Introduce the Turing Machine (TM)?
  - Proposed by Alan Turing in 1936
  - finite-state control + infinitely long tape
  - A stronger computing device than the DFA or PDA

What is a TM?



□ = blank symbols

- Control is similar to (but not the same as) DFA
- It has an infinite tape as memory
- A tape head can read and write symbols and move around the tape
- Initially, tape contains input string (on the leftmost end) and is blank everywhere

What is a TM? (2)

- Finite number of states, with two special states: accept, reject
- Based on the current state and the tape symbol under the tape head, TM then decides the tape symbol to write on the tape, goes to the next state, and moves the tape head left or right
- When TM enters accept state, it accepts the input immediately; when TM enters reject state, it rejects the input immediately
- If it does not enter the accept or reject states, TM will run forever, and never halt

TM versus DFA

- Similarities:
  - Finite set of states
- Differences:
  - TM has an infinite tape and
    - TM can both read and write on the tape
    - Tape head can move both left and right
  - Input string of TM is stored in tape
  - The special states in TM take effect immediately

TM in Action

- Let us introduce a TM that recognizes the language

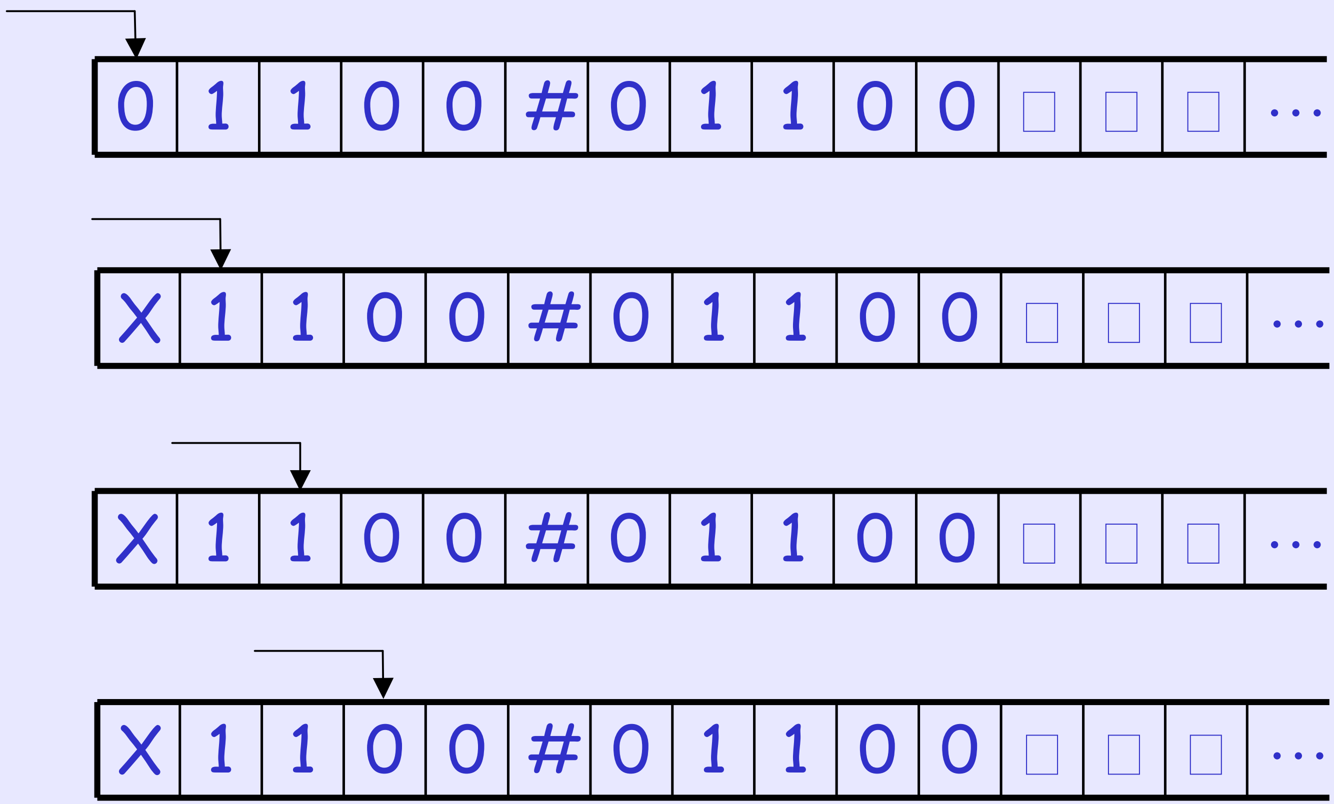
  B = { w#w | w is in {0,1}* }

- We want the TM to accept if the input is in B, and to reject otherwise
- What should the TM do?

TM in Action (2)

- Use multiple passes
- Starts matching corresponding chars, one on each side of #
- To keep track of which chars are checked already, TM crosses off each char as it is examined

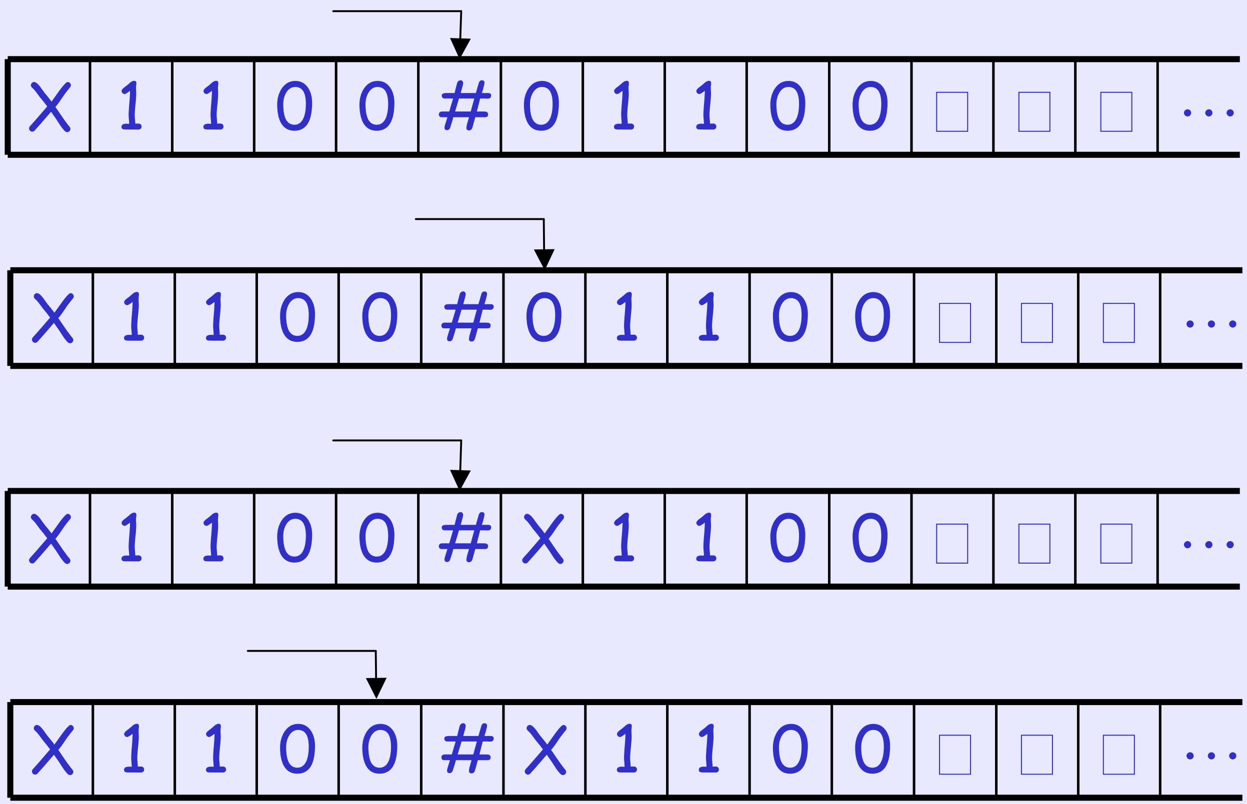
Snapshots of Execution (1)



⋮

Tape head moves to right

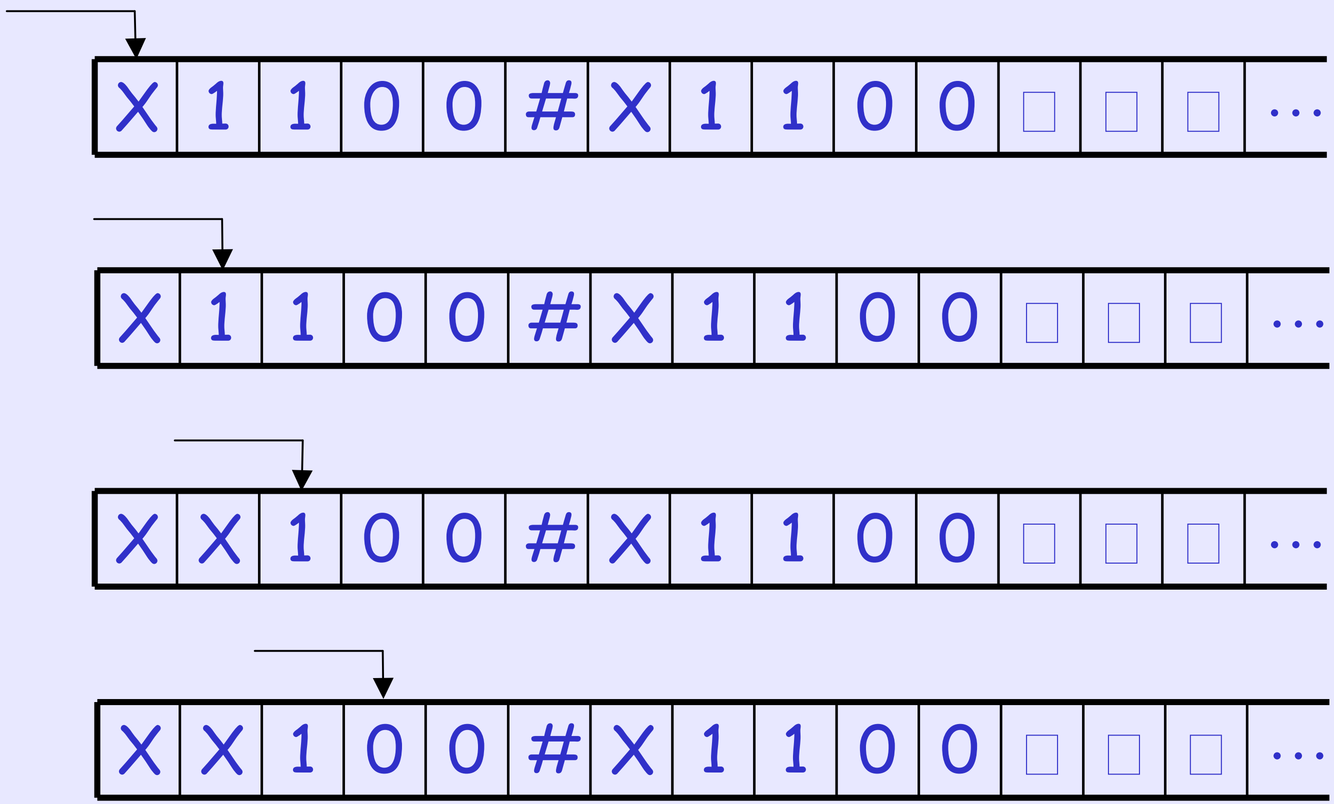
Snapshots of Execution (2)



⋮

Tape head moves to left

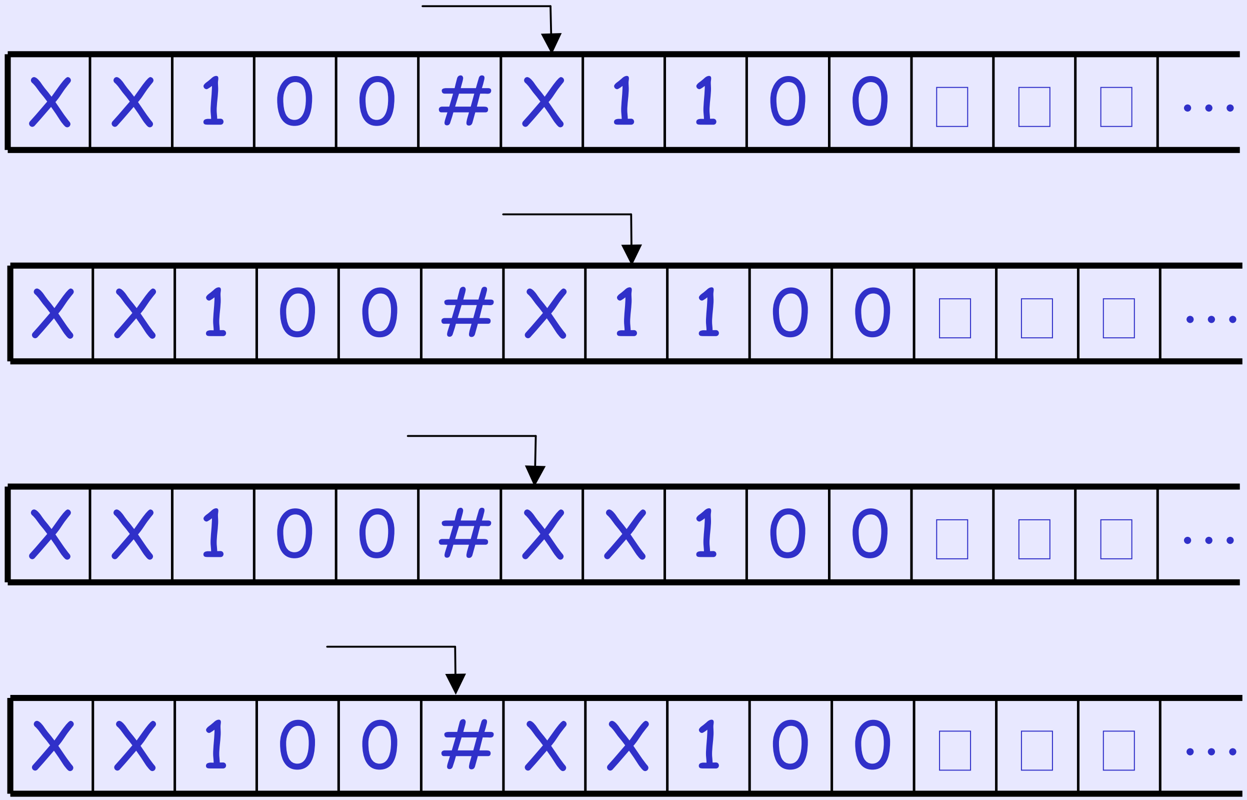
Snapshots of Execution (3)



⋮

Tape head moves to right

Snapshots of Execution (4)



⋮

Tape head moves to left

Snapshots of Execution (5)

X	X	1	0	0	#	X	X	1	0	0	□	□	□	…

X	X	1	0	0	#	X	X	1	0	0	□	□	□	…

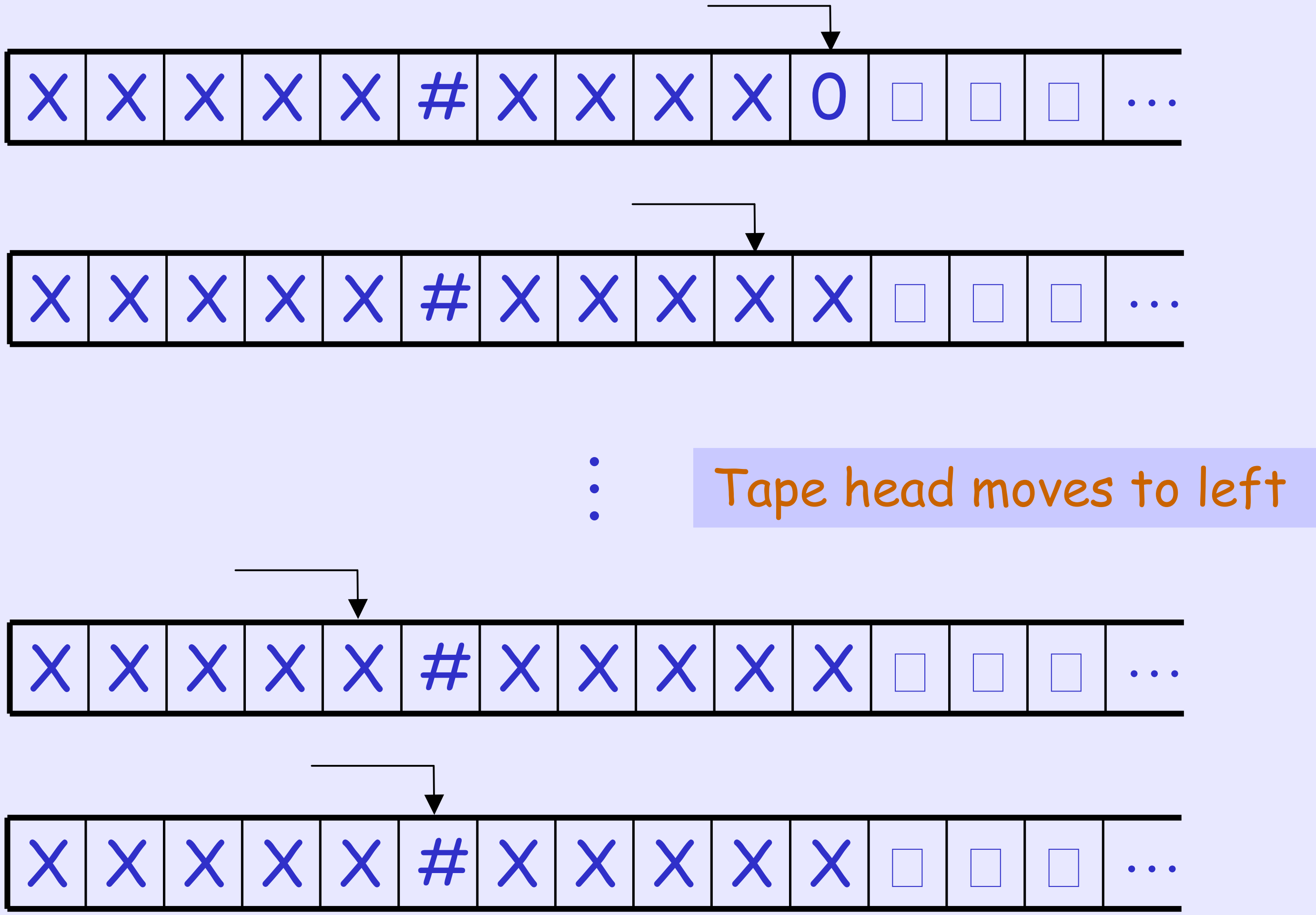
X	X	X	0	0	#	X	X	1	0	0	□	□	□	…

X	X	X	0	0	#	X	X	1	0	0	□	□	□	…

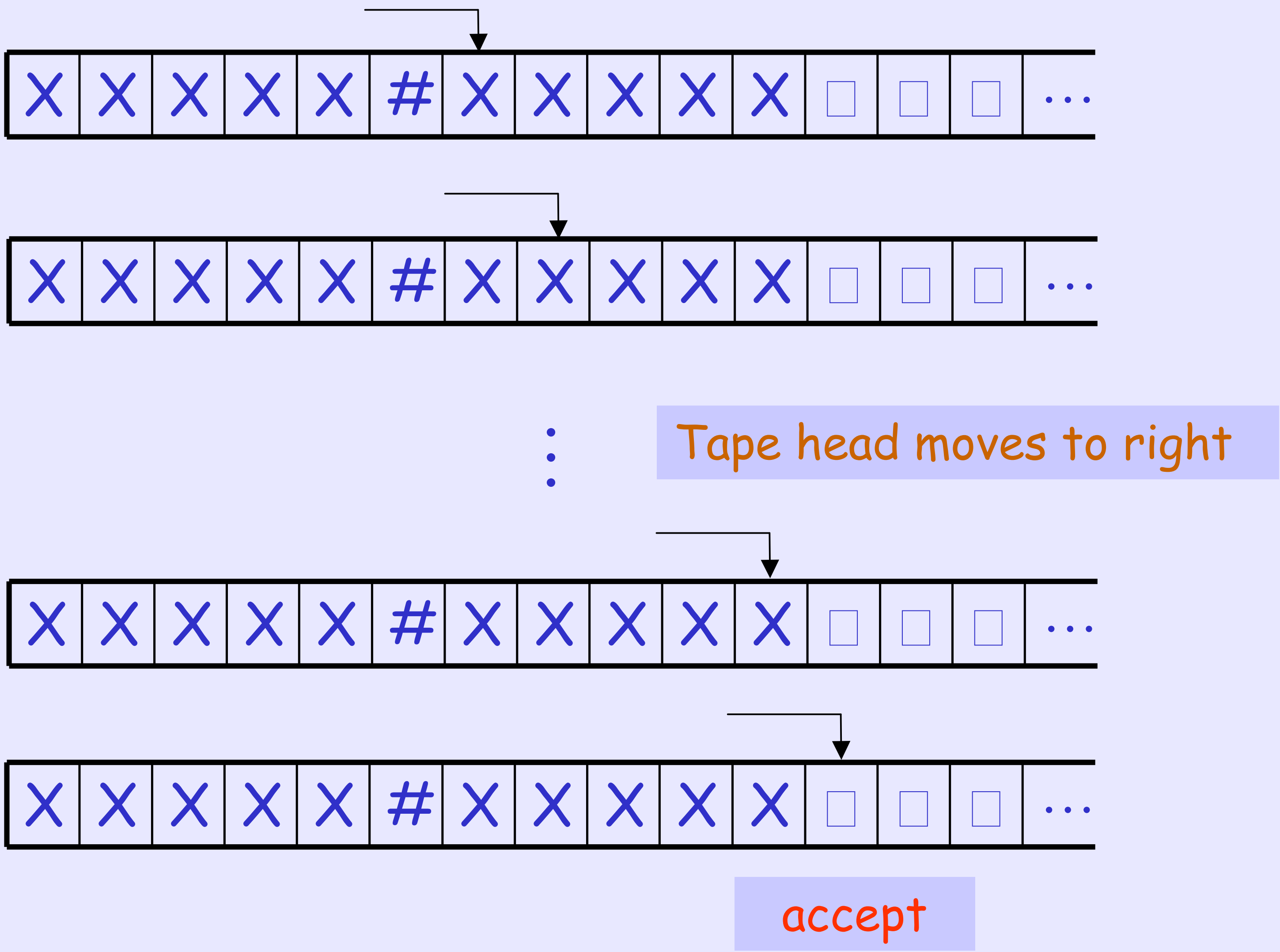
⋮

Tape head moves to right

Snapshots of Execution (6)



Snapshots of Execution (7)



TM (Formal Definition)

- A TM is a 7-tuple $(Q, \Sigma, \Gamma, \delta, q_0, q_{Acc}, q_{Rej})$
  - Q = finite set of states
  - Σ = finite input alphabet, where blank symbol $\square \notin \Sigma$
  - Γ = finite tape alphabet, where $\square \in \Gamma$ and $\Sigma \subset \Gamma$
  - δ is the transition function of the form:

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

    where L, R indicates whether the tape head moves left or right after the transition
  - q_0 is the start state
  - q_{Acc} = accept state, q_{Rej} = reject state

Computation of TM

- Let $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{Acc}, q_{Rej})$ be a TM
- First, M receives input $w = w_1 w_2 \ldots w_n \in \Sigma^*$ on the leftmost n squares of the tape
  - Rest of tape is blank (i.e., filled with □'s)
  - Note: as $\square \notin \Sigma$, first blank on the tape marks the end of input
- Once M has started, the computation proceeds according to the transition function

Computation of TM (2)

- (important) If M tries to move its head to the left of the leftmost end of tape, the head simply stays for that move
- The computation continues until M enters accept state or reject state. Otherwise, M goes on forever

Configuration of TM

- The configuration of a TM specifies the current state, and the current string in the tape, and the current location of the tape head
- When the configuration of a TM is:

  current state = q, current string $w = uv$ with tape head over the first symbol of v, we write:

$$u\ q\ v$$

  as a shorthand notation
- E.g., $1100\ q_7\ 01111$ represents the configuration of TM when tape is 11000111, current state is q_7, and the tape head is over the 3rd 0 in the tape

Configuration of TM (2)

- We say a configuration *C* yields another configuration *C'* if the machine can go from *C* to *C'* in a single transition step
- E.g., if $\delta(q, b) = (q', c, R)$

$$ua\ q\ bv \quad \text{yields} \quad uac\ q'\ v$$

  - special case when off the left end: E.g.,

    $q\ bv$ yields $q'\ cv$ if $\delta(q, b) = (q', c, L)$
- How to represent the start configuration?

Configuration of TM (3)

- More special cases:
  - In an accepting configuration, the current state is the accept state q_{Acc}
  - In a rejecting configuration, the current state is the accept state q_{Rej}
  - These two kinds of configuration are called halting configurations and will not yield further configurations

Acceptance of TM (Formal Definition)

- Turing Machine $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{Acc}, q_{Rej})$ **accepts** input w if a sequence of configurations $C_1, C_2, \ldots, C_k$ exists with
  - $C_1 = q_0\, w$

    i.e., this indicates C_1 is the start configuration
  - For i = 1 to k-1, C_i yields C_{i+1}

    i.e., M moves according to transition function
  - C_k is an accepting configuration

    i.e., M enters accept state in the end

Example of TM

- Let us try to describe formally a TM that recognizes { w#w | w in {0,1}* }

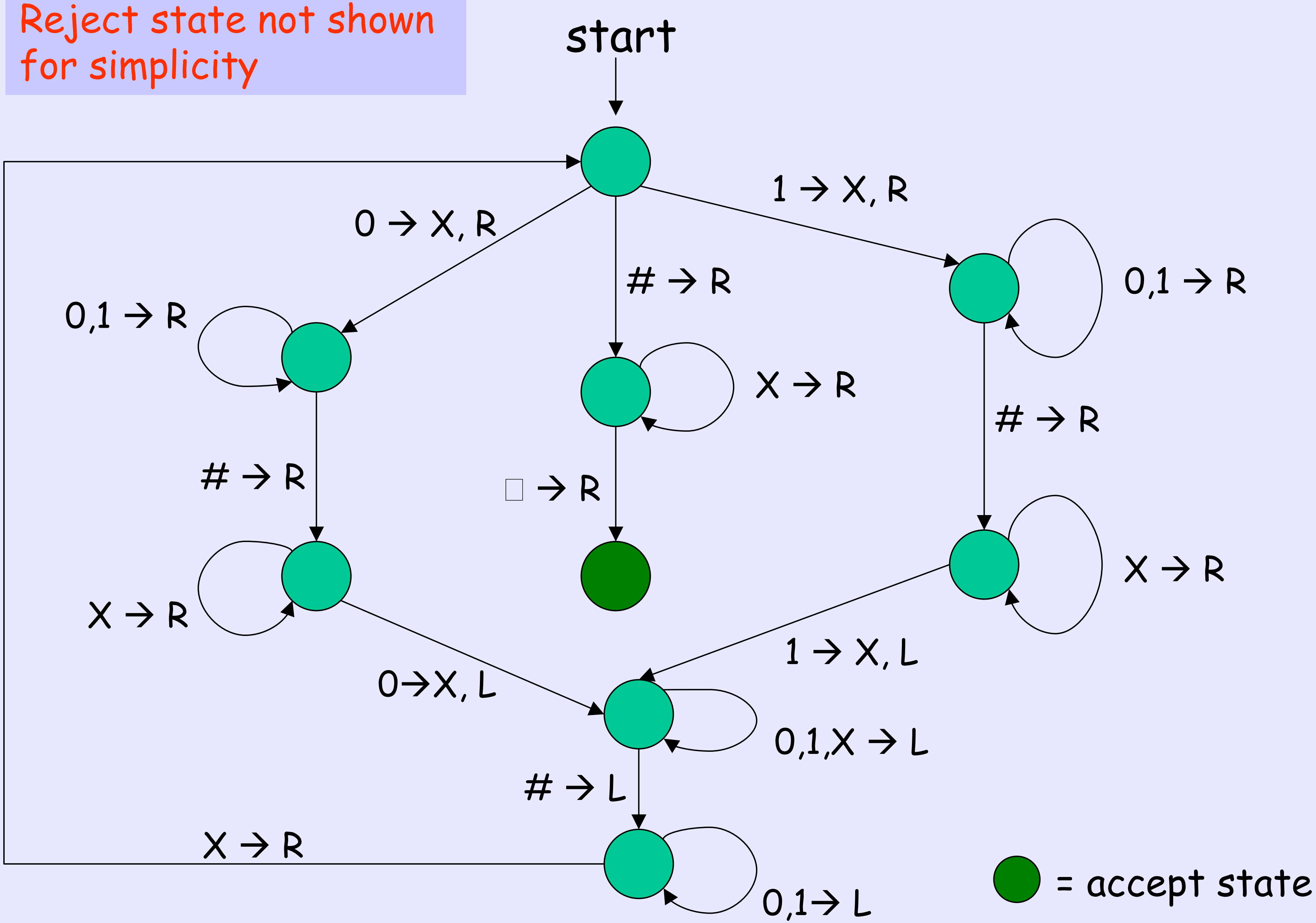
- Also, let us use the shorthand

  a → b, L to denote current tape symbol is changed from a to b after transition, and tape head moves to L

  and

  a → L to mean a → a, L

Reject state not shown for simplicity



Example of TM (2)

- Giving full details of TM are sometimes time-consuming
- Usually, people give only high-level details instead (but they must be precise enough for understanding)
- Let us try to describe the high-level details of a TM M_2 that recognizes the language

$$\{ a^i b^j c^k \mid i \times j = k \text{ and } i,j,k \geq 1 \}$$

High-Level Details of M_2

- On any input string w
  - Scan the input from left to right to check if the string is in the form $a^+b^+c^+$ (rejects if not) (how?)
  - Return the head to left end of tape (how?)
  - Cross off an ‘a’. Scan right to find the first ‘b’. Zig-zag the input string, so that we match each ‘b’ with each ‘c’ by crossing off a ‘b’ and a ‘c’ each time. If not enough ‘c’, rejects
  - Restore all crossed ‘b’. Repeat the above step if there are ‘a’ remaining (how?)
  - If all ‘a’ are gone, check if all ‘c’ are crossed. If yes, accepts. If no, rejects

Recursively Enumerable Language

- The set of strings that M accepts is called the language of M, or the language recognized by M, and is denoted by L(M)
- We call a language Turing-recognizable (or, recursively enumerable) if there is some Turing machine that recognizes it

Recursive Language

- On a given input to a TM, there are three possible outcomes: TM accepts, TM rejects, or TM loops forever
- A TM machine that halts (i.e., never loops) on all inputs is called a decider
- We say a TM M decides a language L if M accepts all strings in L and M rejects all strings not in L (so, M is a decider)
- A language is Turing-decidable (or, recursive) if there is some TM that decides it

Quick Quiz

Is the following true?

1. If L is Turing-decidable, L is Turing-recognizable
2. If L is Turing-recognizable, L is Turing-decidable
3. If L is Turing-decidable, so is $\overline{L}$
4. If L is Turing-recognizable, so is $\overline{L}$
5. If both L and $\overline{L}$ are Turing-recognizable, L is Turing-decidable

Next Time

- Multi-tape Turing Machine
- Non-deterministic Turing Machine (NTM)