CS3063 Theory of Computing

Semester 4 (21 Intake), Jan – May 2024

Lecture 10

Turing Machines: Session 1

Sanath Jayasena

Announcements

- Assignment 2 is out
  - **Due 22 April**

- Assignment 1:

Outline:

Lecture 10

Turing Machines - 1

- Turing Machine (TM) Model
- Definitions, Etc
  - Configuration of a TM
  - Acceptance
- Examples
- Computing a (partial) Function

PART 1

Outline:

Lecture 10

Turing Machines - 1

- **Turing Machine (TM) Model**
- **Definitions, Etc**
- **Examples**
- **Computing a (partial) Function**

Introduction

- Limits of 2 previous models of computation
  - FA: can remember only current state
  - PDA: can access only top of stack
  - Less powerful than real computers we use

- Turing machine (TM)
  - Can do computations a modern computer can
  - …But efficiency and the ways may differ

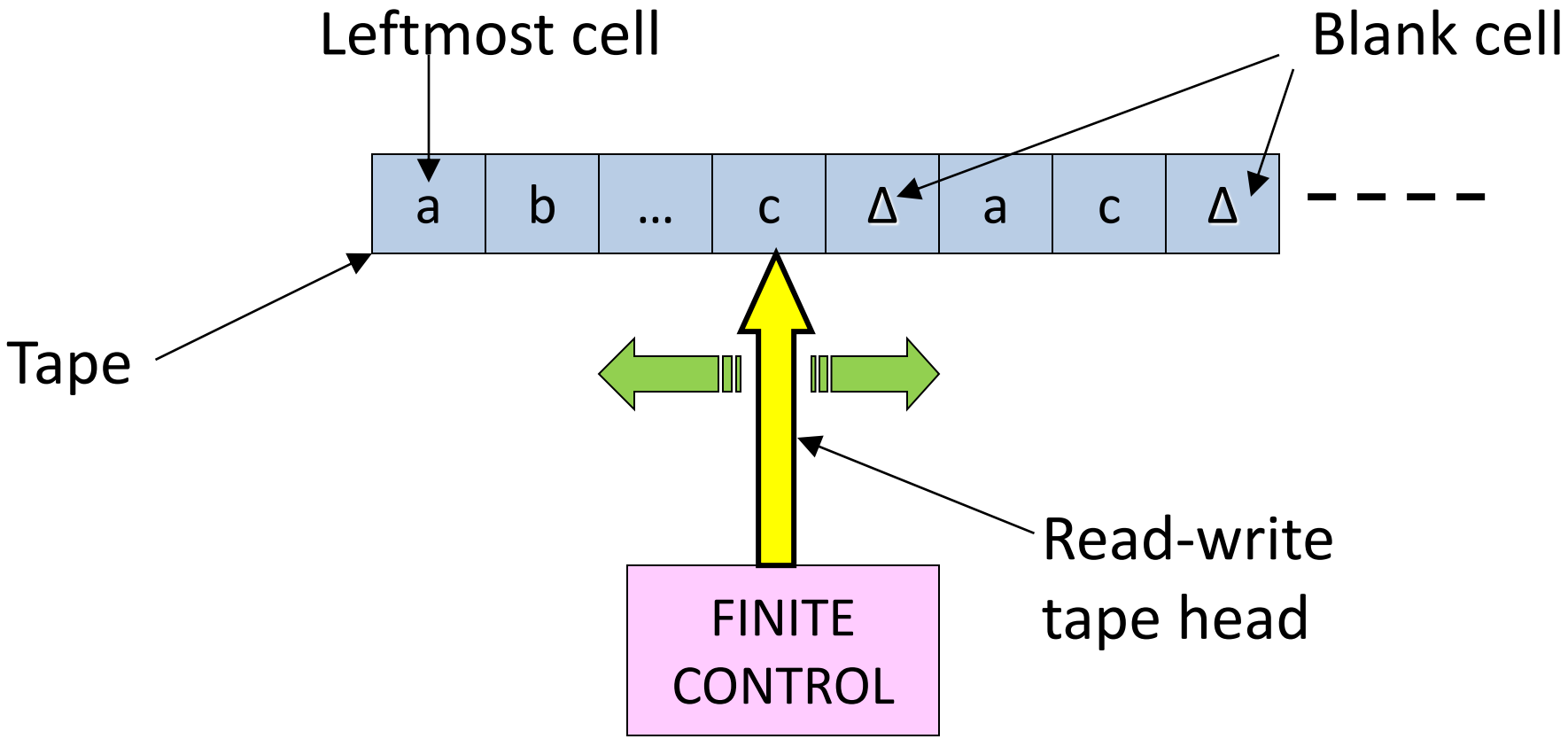
Introduction …contd

- TM: most general model for computation (by *Alan Turing* in 1936)
- Basis: a human computer with 3 rules
  1. Read/write on a paper (with finite alphabet)
  2. Each step taken depends only on current symbol being examined and “state of mind”
  3. State of mind may change but the number of states is finite

Turing Machine Model

- A TM has:
  - A finite control
  - An infinitely long input tape divided into cells
  - A cell is either blank or may hold a symbol from alphabet
  - A leftmost cell on tape
  - A tape head that can read and write a cell and can move to the left or right one cell on tape

Turing Machine Model …contd



Turing Machine Model …contd

- A single move depends on
  - Current state
  - Current tape symbol
- A single move consists of
  - Replacing symbol on current cell
  - Changing state
  - Moving the tape head to left or right one cell (if on leftmost cell, no move to left) or neither

Turing Machine Model …contd

- The tape serves as
  - Input device (input: nonblank-symbol-string)
  - Memory for use during computation
  - Output device (output: string left on tape at the end)

- Important difference from FA, PDA
  - Processing a string means not a simple left-to-right scan of it

Turing Machine Model …contd

- Two final (halting) states
  - Computation need not continue beyond them
  - $\mathbf{h_a}$: acceptance and $\mathbf{h_r}$: rejection
  - TM can move to these to mark completion
- Computation stops when a TM reaches either of the halt states
- But, computation may not stop (i.e., a TM may continue making moves forever)

PART 2

Outline:

Lecture 10

Turing Machines - 1

- Turing Machine (TM) Model
- **Definitions, Etc**
- Examples
- Computing a (partial) Function

Turing Machine: Definition

- A TM is a 5-tuple, T=(Q, Σ, Γ, q_0, δ) where
  - Q : a finite set of states not containing h_a or h_r
  - Σ : a finite input alphabet, subset of Γ
  - Γ : a finite tape alphabet (excluding blank Δ)
  - q_0 : the start state, an element of Q
  - δ is the transition function:

$$Q \times (\Gamma \cup \{\Delta\}) \rightarrow (Q \cup \{h_a, h_r\}) \times (\Gamma \cup \{\Delta\}) \times \{R, L, S\}$$

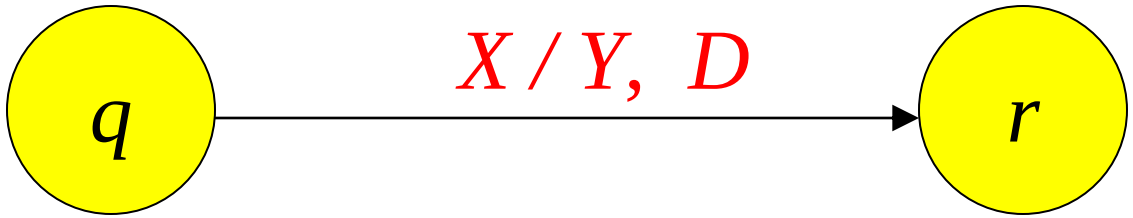
[δ may not be defined for some points]

Turing Machine Definition …contd

- Notation: $\delta(q, X) = (r, Y, D)$ means, when T is in state q and the symbol on current tape cell is X, the machine T
  - replaces X by Y on that cell
  - changes state to r
  - moves the head to right or left one cell or neither depending on whether D is R, L or S

Transition Diagram

- Notation $\delta(q, X) = (r, Y, D)$ indicates a transition from state q to r



Turing Machine Definition …contd

- When r is either h_a or h_r, we say T halts
  - δ Not defined for (h_a, X) or (h_r, X)

- [Note: some authors use terminology and definitions slightly different from these]

Configuration of a TM

- A TM begins with an input string x near the beginning of its tape (other cells blank)
  - There are some non-blank symbols on tape (at any stage during computation)

- The **status** of the TM is specified by
  - Current state
  - Contents of tape (until rightmost non-blank)
  - Current position of tape head

Configuration of a TM …contd

- **Configuration** (i.e., **status**) of a TM is denoted by a pair

  $(q, x\underline{a}y)$

  - q is in Q
  - x and y are strings over Γ U $\{\Delta\}$ (possibly null)
  - a is a symbol in Γ U $\{\Delta\}$
  - Underlined symbol $\underline{a}$ is tape head position
  - The string $x\underline{a}y$ appears on tape, beginning at cell 0, tape head at cell containing a and all cells to the right of y are blank

Configuration of a TM …contd

- If w is a non-null string, $(q, x\underline{w})$ or $(q, x\underline{w}y)$ means tape head is at the 1st symbol of w
- Given a configuration $(q, x\underline{a}y)$
  - y ends in blank(s)
  - $(q, x\underline{a}y\Delta)$ represents the same configuration
  - We usually consider y is either null or has the last non-blank symbol

Configuration Changes

- $(q, x\underline{a}y) \vdash_T (r, z\underline{b}w)$
  means TM, T, goes from the configuration on LHS to that on RHS in 1 move
- $(q, x\underline{a}y) \vdash^*_T (r, z\underline{b}w)$
  means T goes from the configuration on LHS to that on RHS in 0 or more moves
- E.g., if T is currently in $(q, aab\underline{a}\Delta a)$ and $\delta(q, a) = (r, \Delta, L)$, we write

  $(q, aab\underline{a}\Delta a) \vdash_T (r, aa\underline{b}\Delta\Delta a)$

Acceptance by a TM

- Input to a TM and initial configuration
  - Input string on tape initially beginning at cell 1
  - Tape head at cell 0 (which is blank)
  - Initial configuration for input x is $(q_0, \underline{\Delta} x)$

- A string x in Σ^* is **accepted** by a TM, T, if there are y and z in $(\Gamma \cup \{\Delta\})^*$ and a in $\Gamma \cup \{\Delta\}$ so that $(q_0, \underline{\Delta} x) \vdash^*_T (h_a, y\underline{a}z)$

Acceptance by a TM …contd

- In other words: starting in the initial configuration corresponding to input x, T eventually reaches an accepting configuration
- The **language accepted** by a TM, T, is the set L(T) of all input strings accepted by T

- Note: If x is not in L, this does not necessarily mean that T will reject by moving to h_r

What can result?

- When a TM processes a given string, the TM can:

1. Accept it by entering state h_a
2. Reject it by entering state h_r
3. Enter an infinite loop (i.e., the TM never halts but continues moving forever)

What can result? …contd

- Cases 1, 2: we see the outcome
  - Can say if the string is accepted or not

- Case 3
  - Even if the string is not accepted, we will never find out ☹
  - No outcome
  - In some cases, this will happen inevitably

PART 3

Outline:

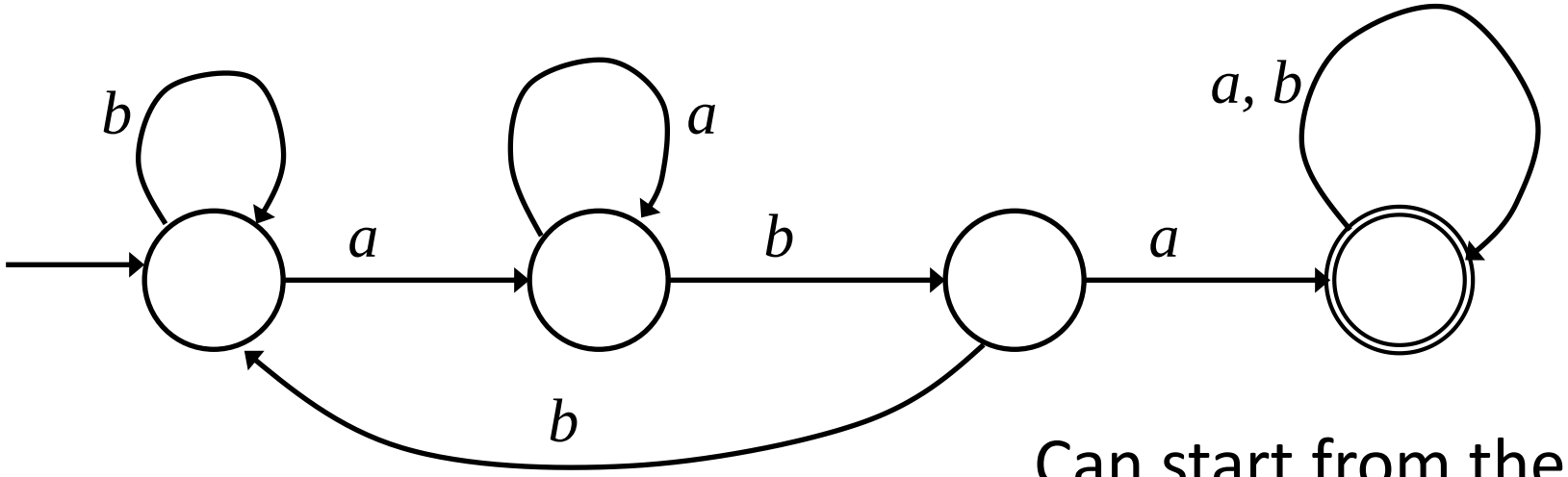
Lecture 10

Turing Machines - 1

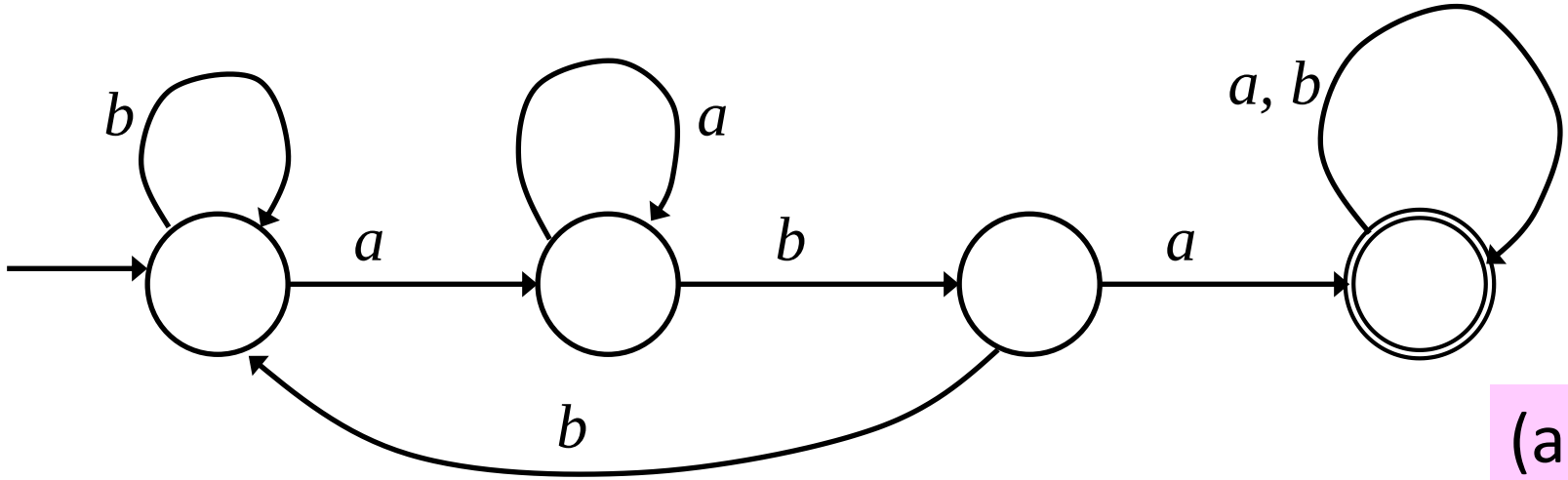
- Turing Machine (TM) Model
- Definitions, Etc
- **Examples**
- Computing a (partial) Function

Example 1

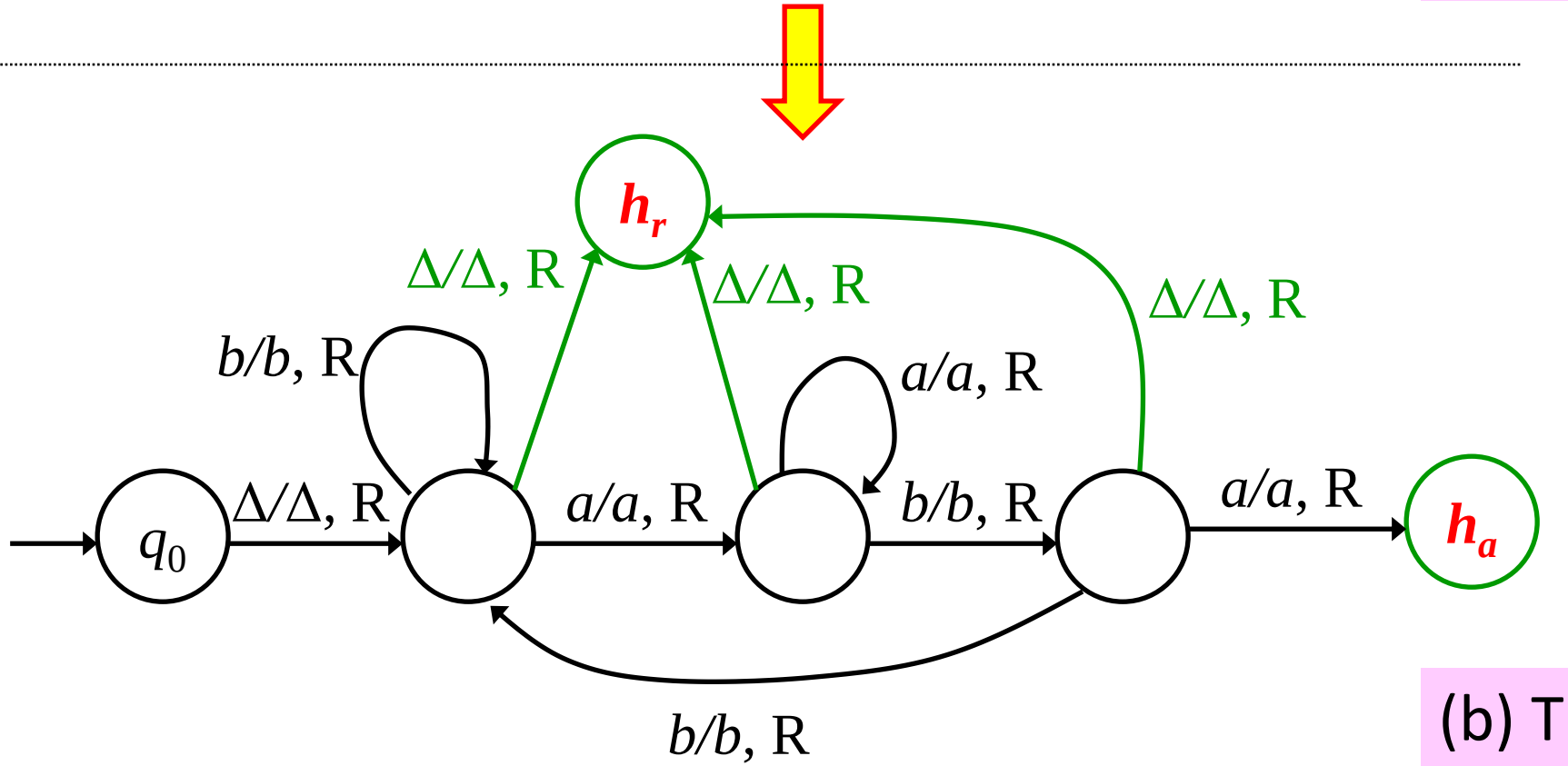
- Example 9.1, p. 323
  - Construct a TM to accept the **regular language** L= $(a|b)^*aba(a|b)^*$ for $\Sigma=\{a,b\}$



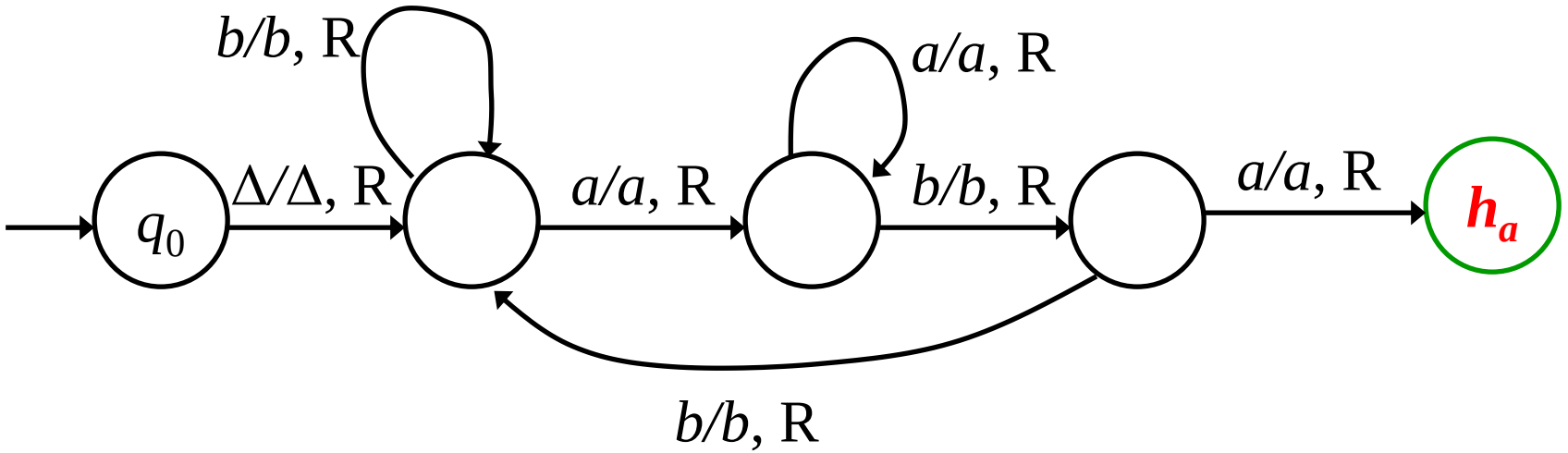
Can start from the FA for L



(a) FA for L



(b) TM for L



(c) Simplified TM for L (transitions to h_r omitted)

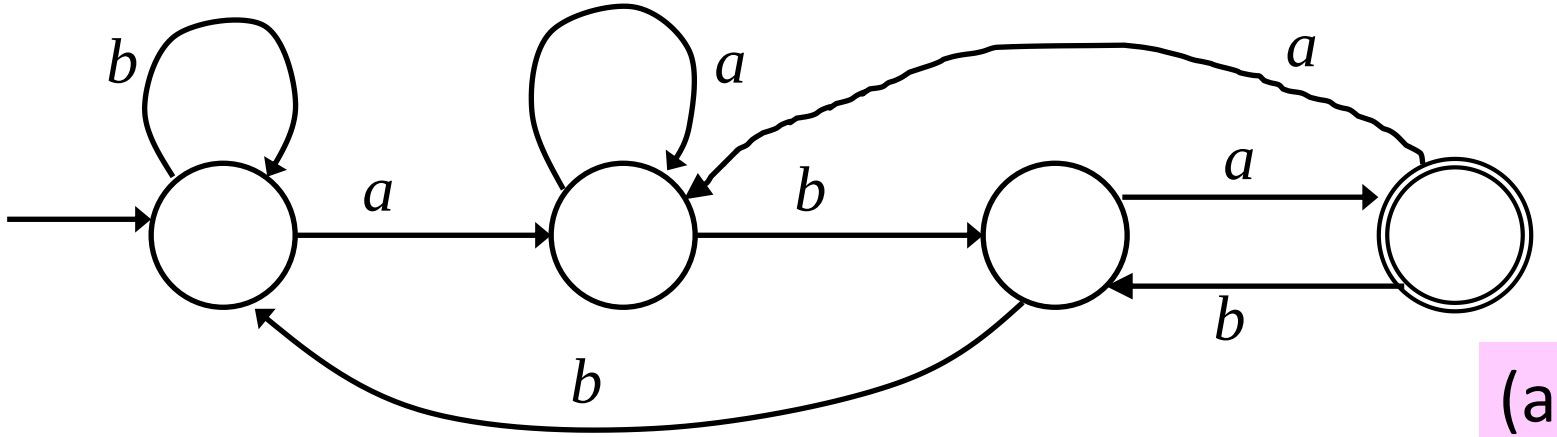
In the simplified TM, the diagram is to be interpreted as moving to the state h_r for each combination of state and tape symbol for which no move is shown explicitly

The TM mimics the FA (moves the tape head to the right at each step, never changing any tape symbol) – this is OK for a regular language.

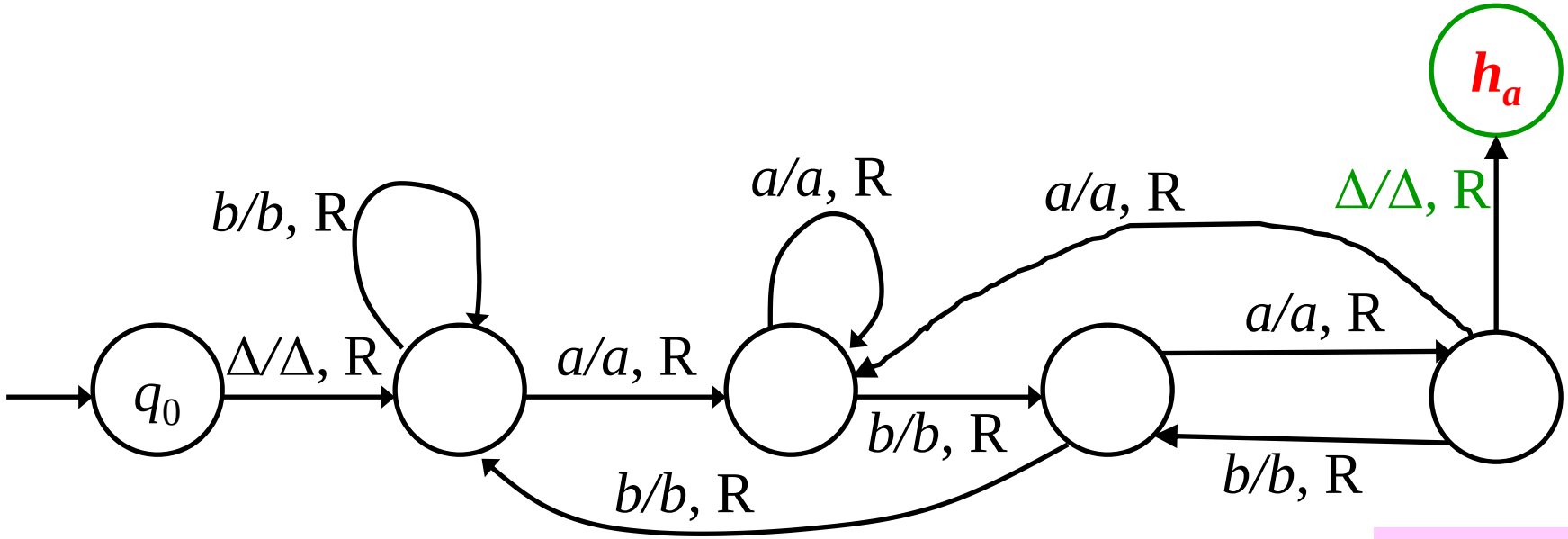
Example 2

- Construct a TM to accept the language L= {x in $(a|b)^*$ | x ends with aba}
  - Note: In the previous example, as soon as the TM sees aba on the tape, it enters the state h_a and accepts the entire input string, even though it may not have read all of it
  - But with some regular languages, all the input must be read to accept or reject (like this example)

– Can we use the ideas in previous example?
  - Solution on next slide



(a) FA for L



(b) TM for L

Example 3

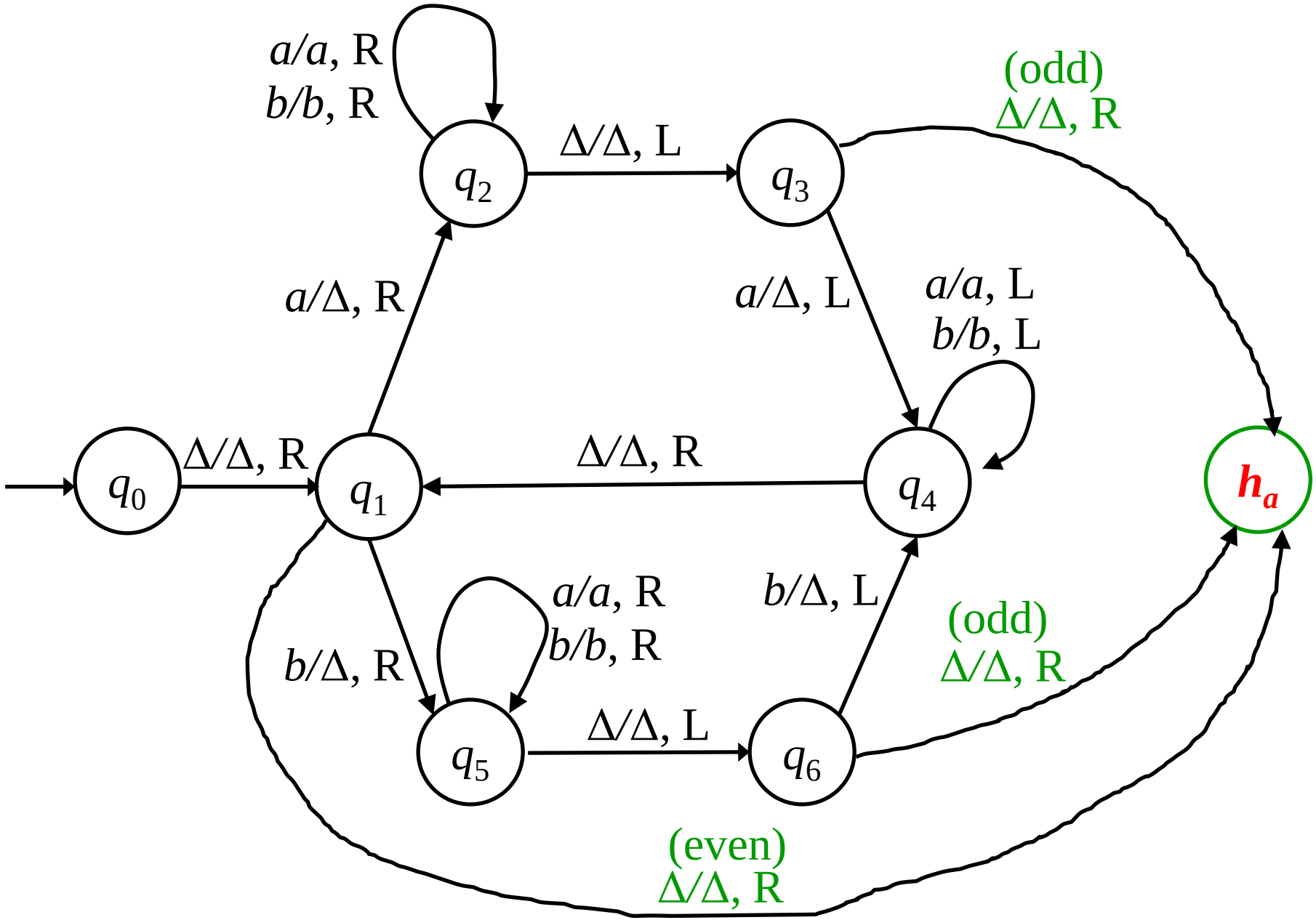
- Example 9.2, p. 324:
- Construct a TM to accept the language pal of palindromes over $\Sigma=\{a,b\}$
  - (This is a context-free language)

- Then, trace the moves of the TM for
  - $abaa$ (a non-palindrome)
  - aa (even-length palindrome)
  - aba (odd-length palindrome)

Example 3: Solution

- Recall: we constructed a non-deterministic PDA to accept the language pal
- But we can have a deterministic TM for this case…
- Think of how you might solve it by hand (given a very long string)?
- Final TM next slide…
  - Do the tracing of moves as homework…

TM to accept the language ***pal*** of palindromes over Σ={*a*,*b*}



Exercise

- Homework
  - Read Example 9.3 (p. 326)
  - Construct a TM accepting the non-context-free language, {ss | s is in {a,b}*)

PART 4

Outline:

Lecture 10

Turing Machines - 1

- Turing Machine (TM) Model
- Definitions, Etc
- Examples
- **Computing a (partial) Function**

Computing a Function

- A computer program producing a specific output string for every legal input string computes a function from one string set to another
- Similarly, a TM, T, can compute a function f whose domain is a subset of Σ^*
  - This means: for any string x in the domain of f, when T starts in initial configuration, it will halt eventually with output string $f(x)$ on the tape

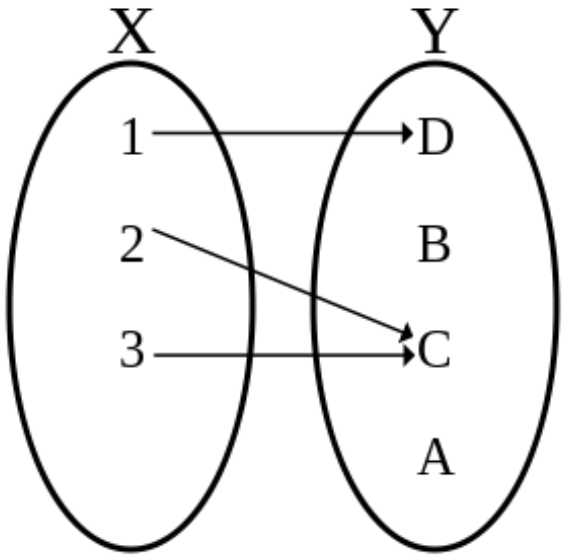
Computing a Function …contd

- So far, the tape contents at the end of computation were not important to us
  - Whether TM halts in accepting state was important
- Now, the emphasis is the output produced for an input string in the domain of f
  - String not in domain → output result irrelevant
  - Need precise computation of f for the domain
  - In the process accepts the domain (language)
  - For x not in domain, T will not accept x

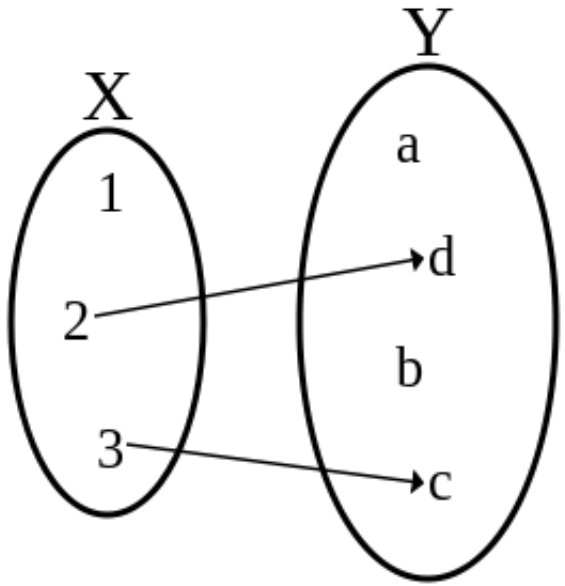
Computing a Partial Function

- For convenience, we will consider ***partial functions*** on Σ^*
  - Rather than *functions on subsets of* Σ^*
  - A partial function can be undefined at some points
  - [a *total* function is defined everywhere]
- A TM can handle a function of several variables
  - E.g., a k-tuple $(x_1, x_2, \ldots, x_k)$ in $(\Sigma^*)^k$

Functions: Review



a function
(or, total function)



a partial function

A TM Computing a Function

- **Definition** (from Def 9.3, p. 329 in text)
  - Let $T=(Q, \Sigma, \Gamma, q_0, \delta)$ be a TM, f be a partial function on Σ^* with values in Γ^*
  - We say that T computes f , if for every x in Σ^* at which f is defined

$$(q_0, \underline{\Delta} x) \vdash^*_T (h_a, \underline{\Delta} f(x))$$

and no other x in Σ^* is accepted by T

A TM Computing a Function

- Definition …contd
  - If f is a partial function on $(\Sigma^*)^k$ with values in Γ^*, T computes f if for every k-tuple $(x_1, x_2, \ldots, x_k)$ at which f is defined

    $(q_0, \underline{\Delta} x_1 \Delta x_2 \Delta \ldots \Delta x_k) \vdash^*_T (h_a, \underline{\Delta} f(x_1, x_2, \ldots, x_k))$

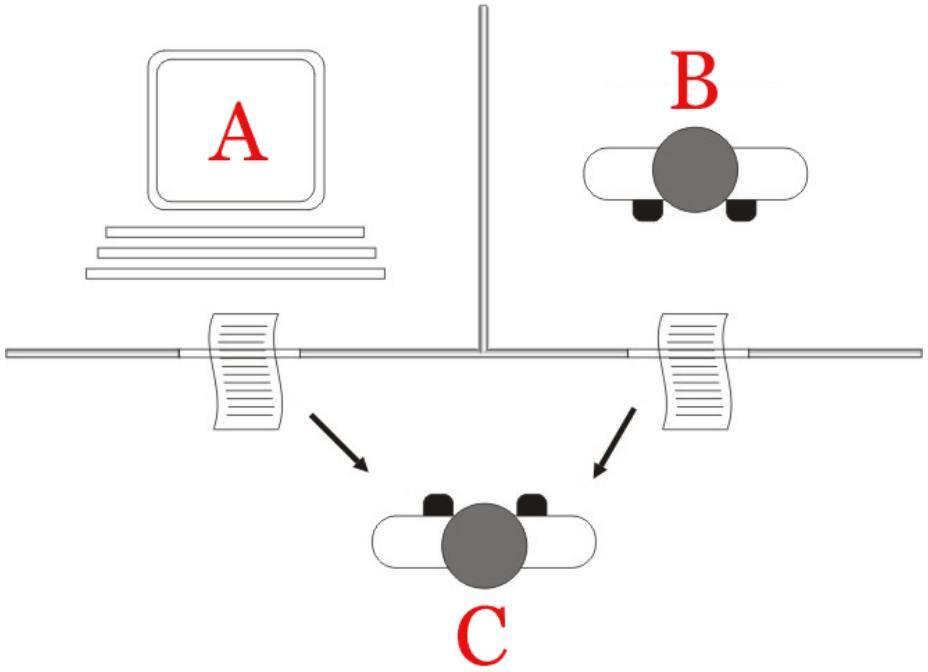
    and no other k-tuple input string is accepted
  - For two alphabets Σ_1 and Σ_2 and a +ve integer k, $f : (\Sigma_1^*)^k \rightarrow \Sigma_2^*$ is ***Turing computable*** (or ***computable***) if there is a TM computing f

Computing a Function …contd

- Further reading…
  - Similarly, numerical functions can also be computed by a TM (see Def: 9.4 on p. 329; see also Examples 9.4, 9.5, 9.6)
- ***Characteristic Function*** (E.g., 9.6)
  - For language L in Σ^* the ***characteristic function*** of L is the function $f : \Sigma^* \rightarrow \{0,1\}$ defined by:

$$f(x) = \begin{cases} 1 & \text{if } x \in L \\ 0 & \text{otherwise} \end{cases}$$

Turing Test



The "standard interpretation" of the Turing test:

- Player C, the interrogator, is given the task of trying to determine which player – A or B – is a computer and which is a human.
- The interrogator is limited to using the responses to written questions to make the determination.

Ref: Wikipedia

Related Concepts

- *Turing Test* (*Imitation Game*)

- *Turing Completeness/Equivalence*
  - What is the meaning of “X is *Turing complete*” or “X and Y are *Turing equivalent*”?

- Homework: read on these!

Conclusion

- We started the topic of Turing Machines
  - Turing machine (TM) model
  - Definitions
    - Configurations
    - Acceptance
  - Example TMs
  - Computing a Function with a TM