CSCI 361 Lecture 12: Turing Machines II

Shikha Singh

Announcements & Logistics

- **HW 4** was due last night
- No **reading assignment** until after midterm
- **Practice midterm** was posted on GLOW
  - Solutions will become available today noon
- CSCI 361 Midterm on **Oct 22 (Tuesday)**:
  - **Extended help hours** on Mon Oct 21: **1.30-4 pm**
  - Different TA hour schedule Sun - Mon: see course calendar
  - See this document with more details
- Reminder: tomorrow colloquium
  - ***What I did Last Summer (Research)***

Last Time

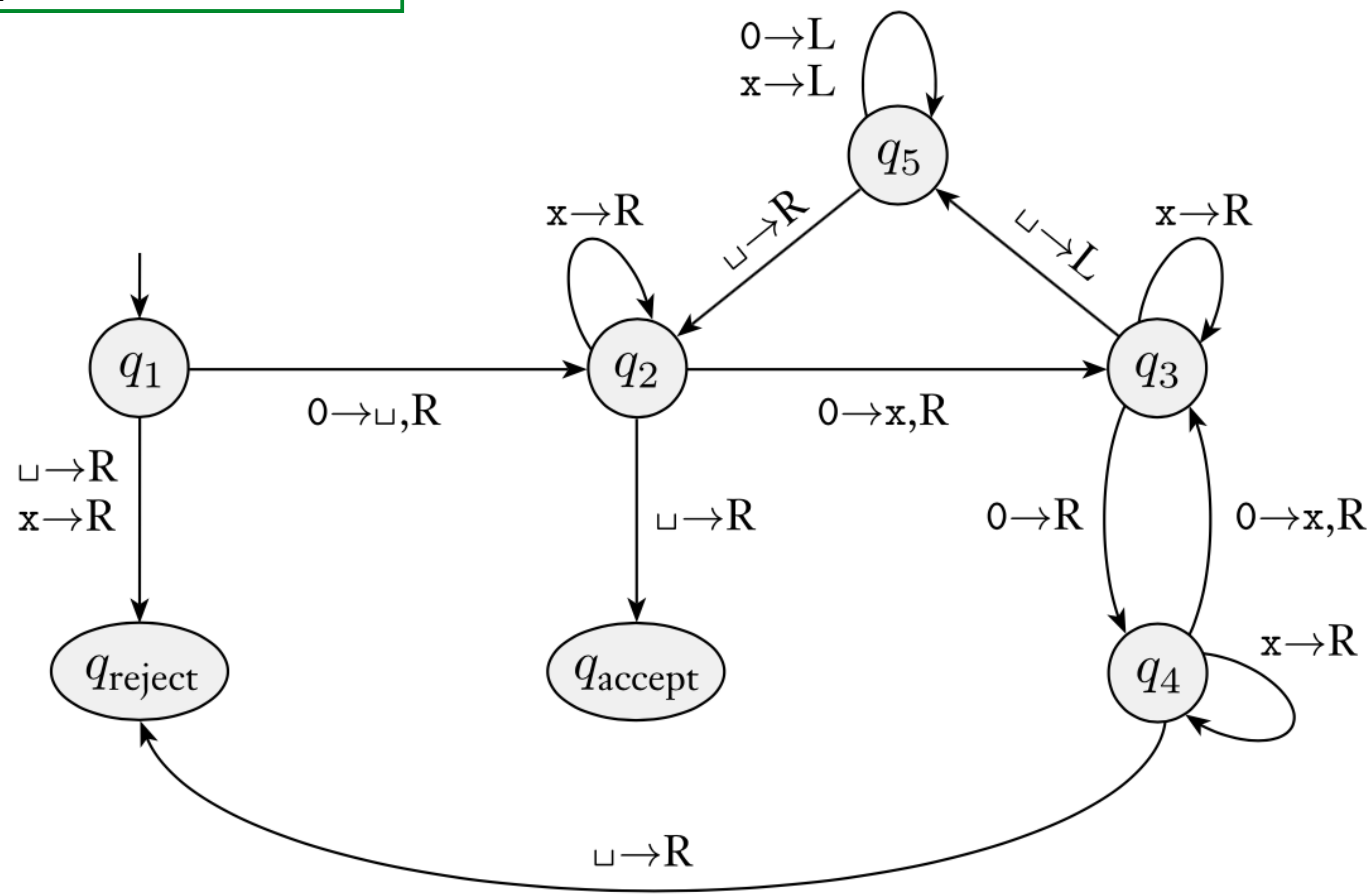
- Introduced Turing machines
  - Definition of TM and how it computes
  - TM-decidable vs TM-recognizable languages

Today

- More practice with Turing Machines and decidability
- Discuss practice midterm and review list of topics for exam

- **Example TM**: Consider a TM for the language $A = \{0^{2^n} \mid n \geq 0\}$

Each transition of the form x → y, D means "upon reading x, replace it with symbol y and move the tape head in direction D". If y is omitted x is left unchanged



Medium-Level Description

Consider a TM M for for the language $A = \{0^{2^n} \mid n \geq 0\}$:

$M =$ "On input string w,

1. Sweep left to right across the tape, crossing off every other zero.
2. If in Stage 1 and there is a single zero, ***accept***
3. If in Stage 1 and there are more than one odd zeros, ***reject***
4. Return to the lefthand end of tape and go to stage 1."

Call such description medium level: says how the TM works but not as explicit as a state-diagram.

Practice

- **Exercise.** Give a medium-level description of a TM that recognizes $L = \{a^n b^n c^n \mid n \geq 0\}$

Exercise. Give a medium-level description of a TM that recognizes $L = \{a^n b^n c^n \mid n \geq 0\}$

Solution.

In the following we assume that tape symbols $x, y, z \notin \Sigma$

$M =$ "On input string w:

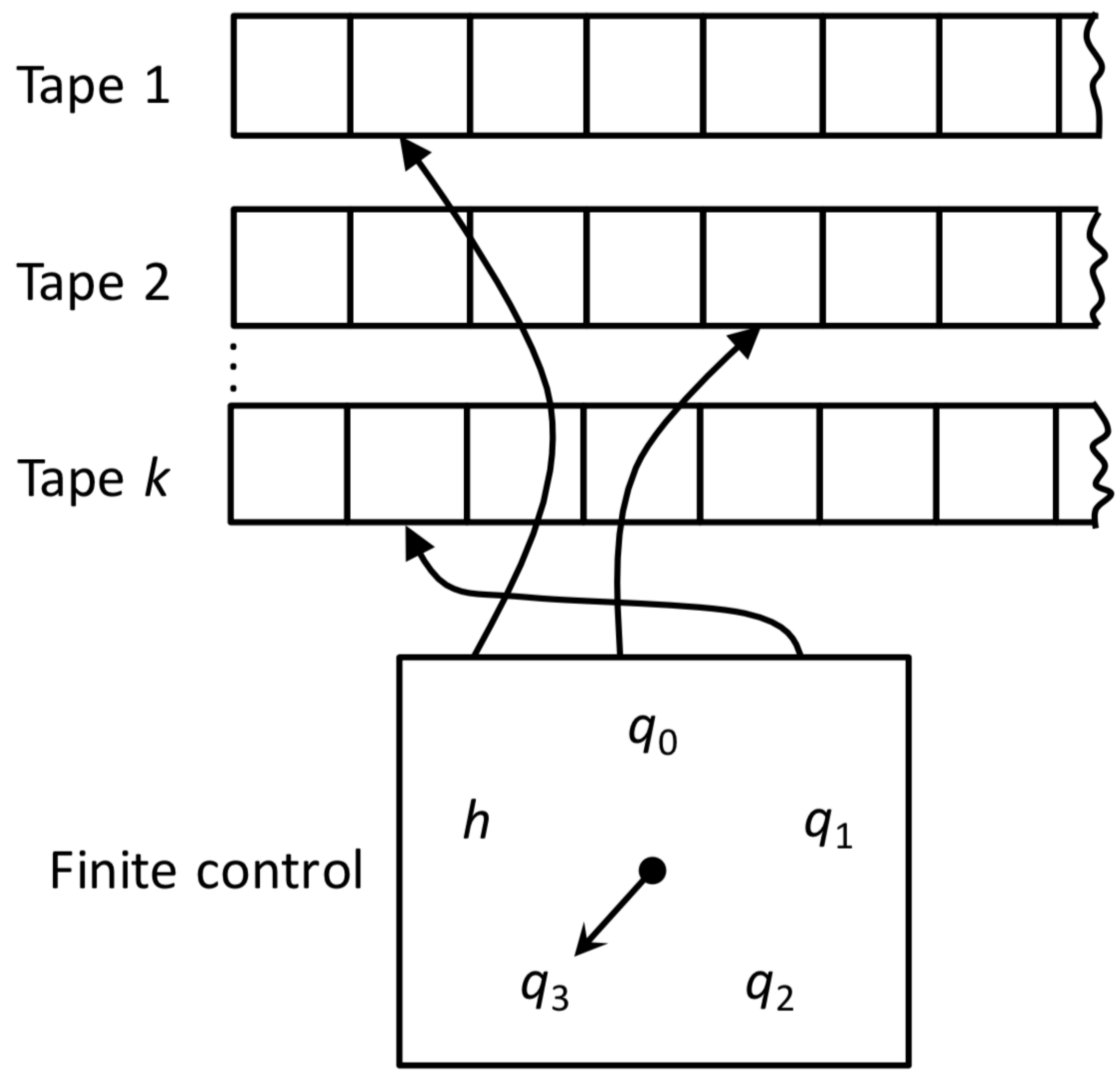
1. Scan the input from left to right to check if it is of the form $a^*b^*c^*$ and reject if it is not.
2. Return head to left hand end of tape.
3. While there are a's remaining on the tape, do:
    - Replace the first a with an x, scan right until a b occurs; replace it with y, and scan to the right until a c occurs; replace it with z. If the corresponding b and c for each a are not found, reject.
4. If there are no b's or c's remaining on the tape, accept. Otherwise, reject.

Many Models of TM

- Many equivalent to define a Turing machine
  - Two models are equivalent if they can simulate each other
- Invariance to certain changes makes a TM robust
- Will discuss two variants:
  - Multitape TM
  - Non-deterministic TM

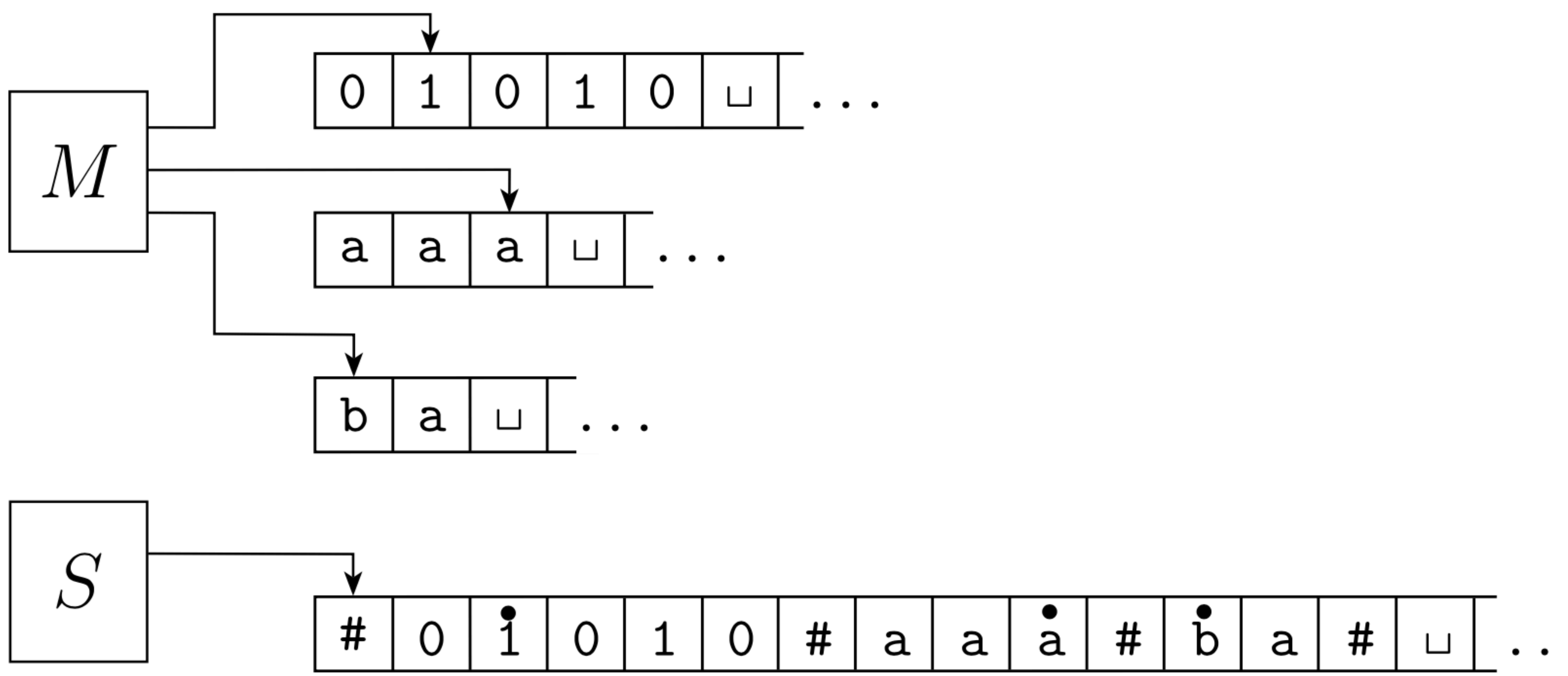
Multitape Turing Machines

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



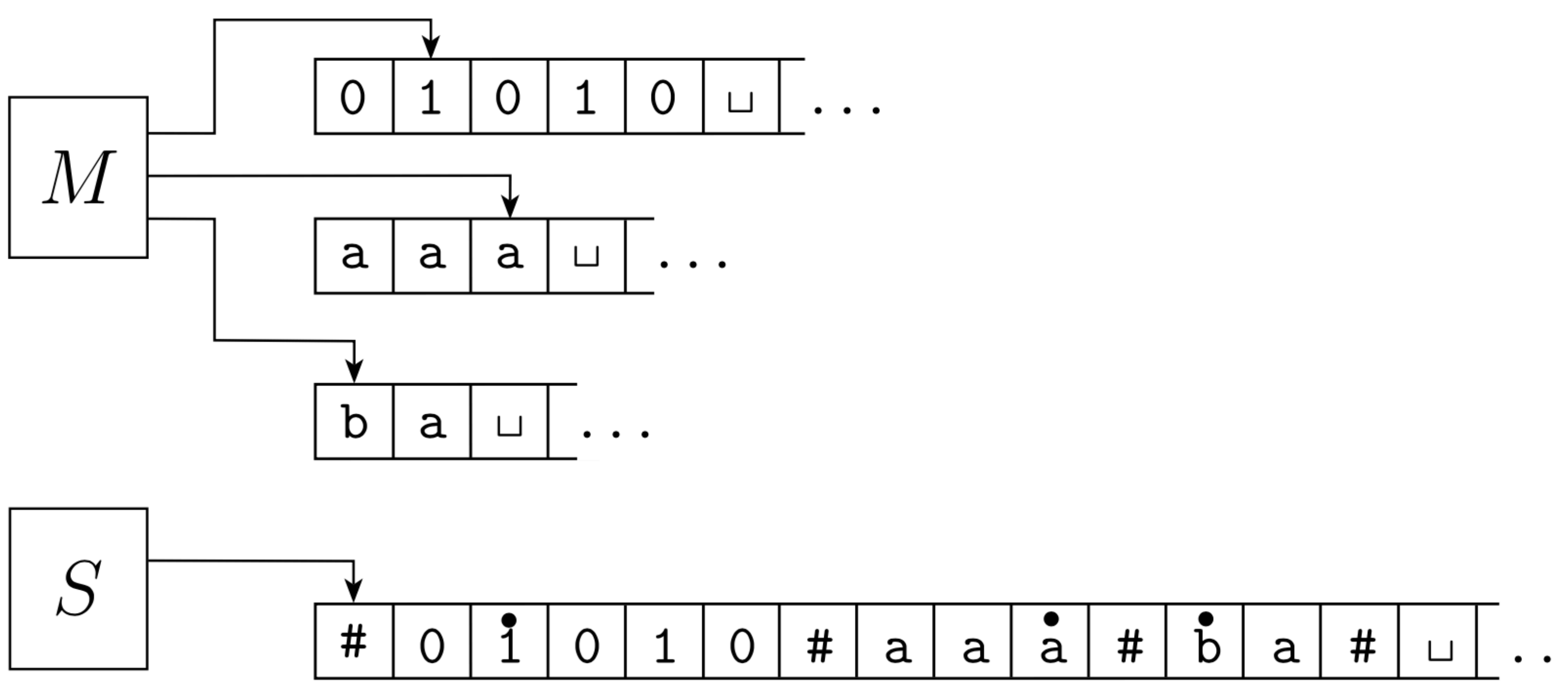
Multitape TM $\Longleftrightarrow$ Single Tape TM

- Can a single tape TM simulate a multi-tape TM?
- **Theorem.** Every multi-tape TM has an equivalent single-tape TM.



Multitape TM $\Longleftrightarrow$ Single Tape TM

- Can a single tape TM simulate a multi-tape TM?
- **Theorem.** Every multi-tape TM has an equivalent single-tape TM.
- Intuition: S uses # to demarcate single tape into k parts
- Uses special tape symbols $\mathring{a}$ to indicate head location on each tape



Nondeterministic TM Machine

- At any point in the computation, the TM can "guess" and be in one of several different possibilities
- Transition function:
  $\delta : Q \times \Gamma \rightarrow \mathscr{P}(Q \times \Gamma \times \{L, R\}^k)$
- The computation of a NTM is a tree whose branches correspond to different possibilities for the machine
- If some branch leads to an accepts state, the machine accepts
- Intuition for non-determinism: "perfect guessing"
  - When there are several options, magically guess the correct one

Designing NTM

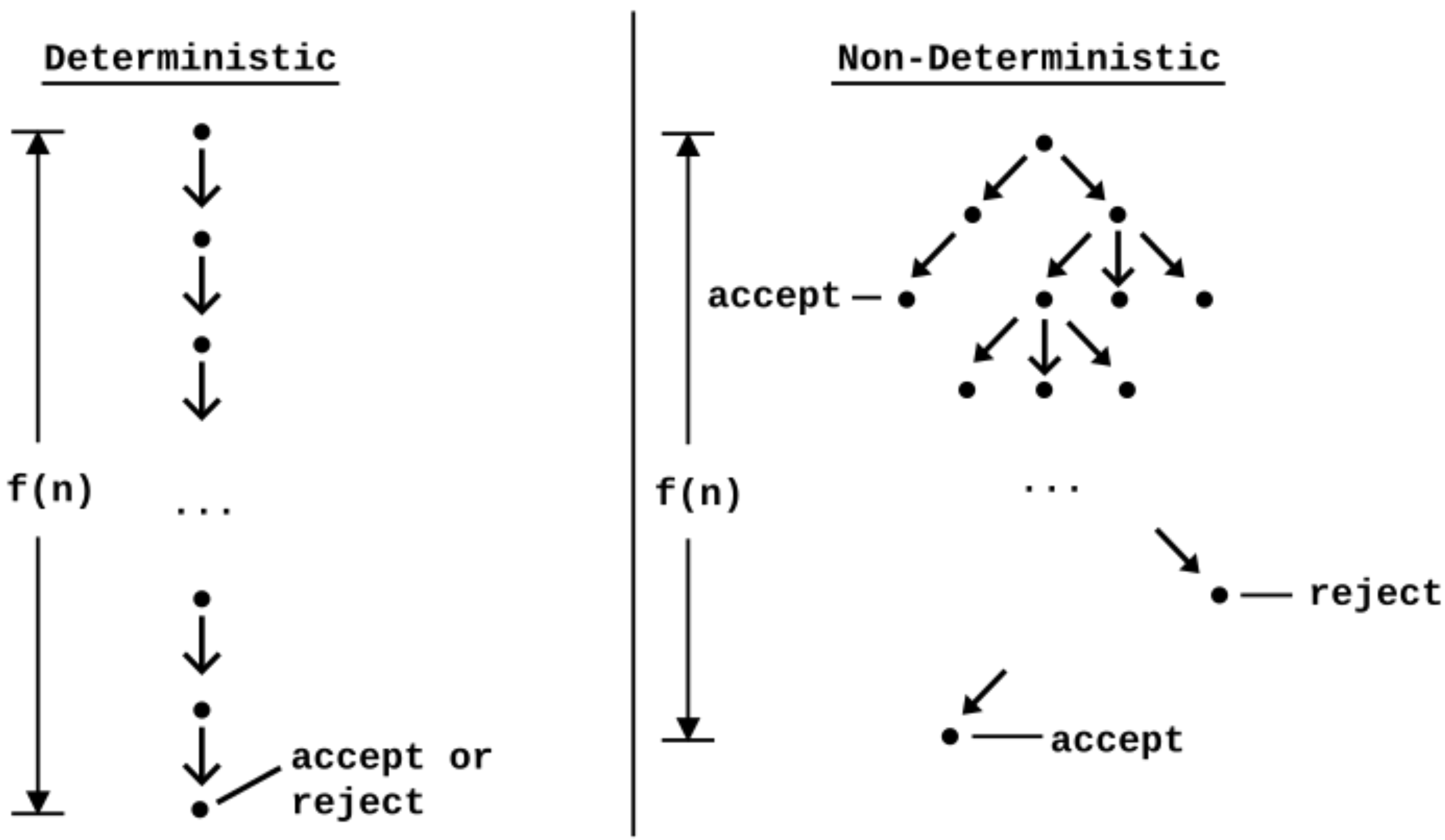
- When designing NTMs, it is often useful to use the **guess-and-check** approach
- Non-deterministically guess a string that can prove $w \in L$
- Deterministically verify that the guess is correct
  - If $w \in L$, there is some guess that works
  - If $w \notin L$, then no guess will work

Designing NTM

- **Problem.** Design a NTM for the language
  $L = \{1^n \mid n \text{ is composite}\}$
- A non-deterministic TM for it?
  - Guess a number $1 < m < n$
  - Divide n by m, if it is divisible then accept
  - Else, reject

NTM ⟺ DTM

- **Theorem.** Every nondeterministic TM has an equivalent deterministic TM.

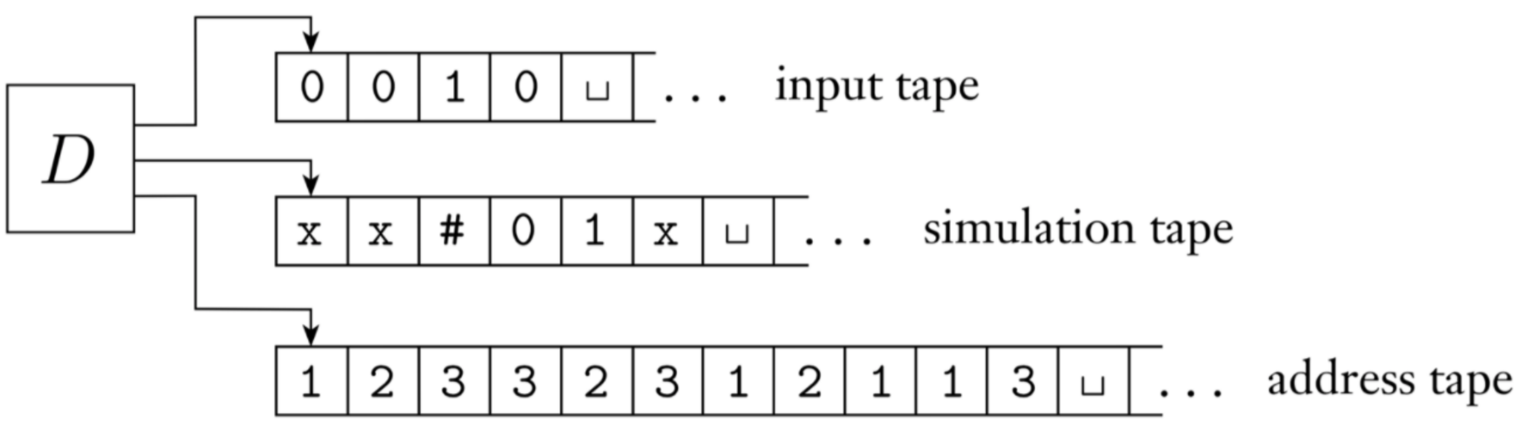


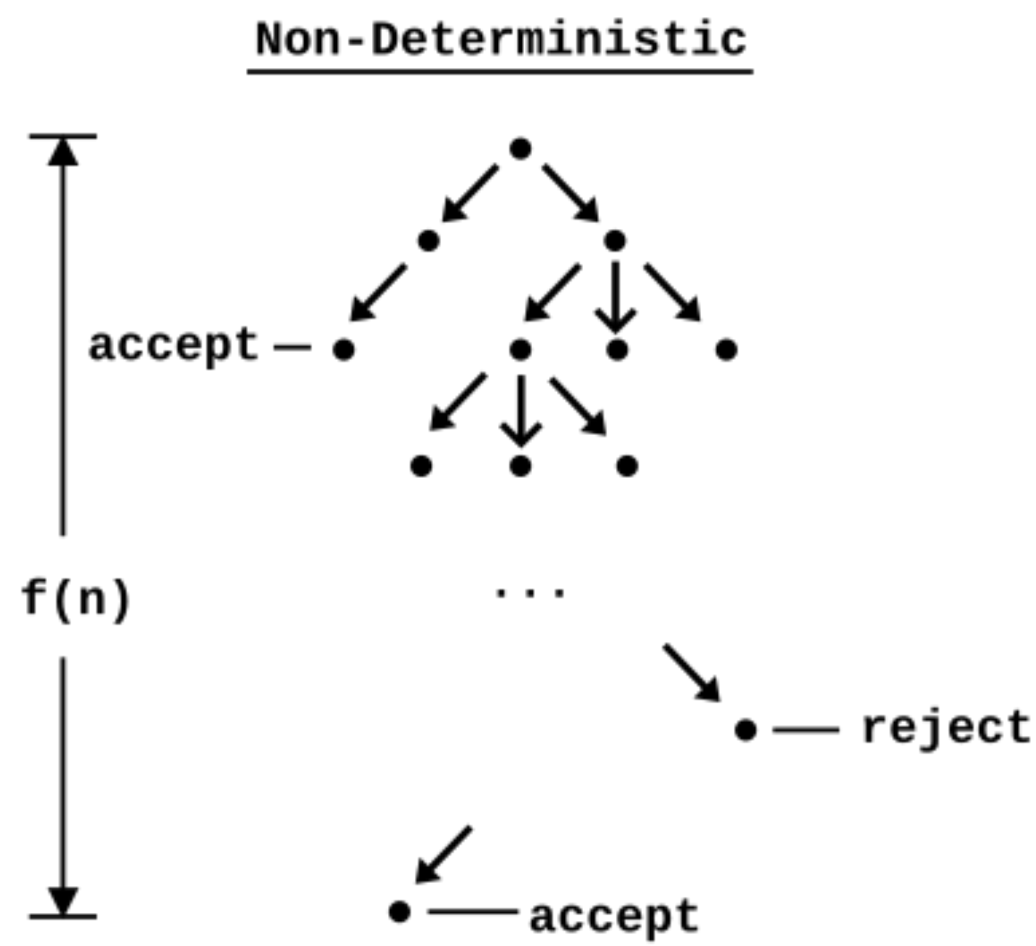
NTM $\Longleftrightarrow$ DTM

- **Theorem.** Every nondeterministic TM has an equivalent deterministic TM.
- Intuition. Use a multi-tape machine to simulate NTM
  - Computation of NTM is a tree
  - Each node in the tree is a TM configuration
  - Try all possible branches by traversing the tree
  - Which traversal to do: DFS or BFS?
  - If an accept is ever reached, accept

NTM $\Longleftrightarrow$ DTM

- **Proof Idea.** Three tapes: input tape is never altered, simulation tape stores the tape contents of the "current branch" being simulated, address tape keeps track of which node is being traversed in the tree





Main Takeaways

- Alternate characterizations
  - A language is TM recognizable iff some NTM recognizes it
  - A language is TM decidable iff some NTM decides it
- Will revisit NTMs vs DTMs when discussing time complexity
  - P versus NP problem
- Many different ways to define a TM: all are equivalent
  - Makes it a good universal model of computation

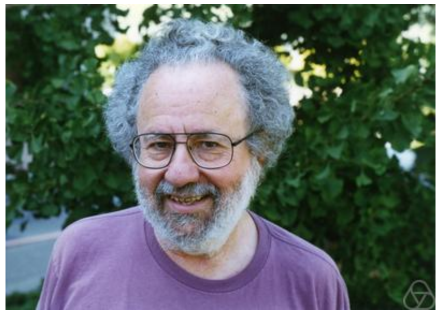
Why Study Turing Machines

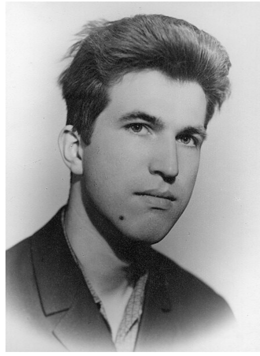
- Not a good model to think about ***fast*** computation
- Fast algorithms are a subject of CS 256
- In this class, we are interested in finding out if we can solve a problem at all
- To show a problem is not solvable, we need a model of what it means to solve a problem
    - Church-Turing Thesis:

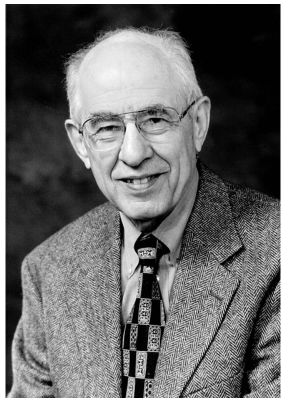
Intuitive notion of algorithms	equals	*Turing machine algorithms*

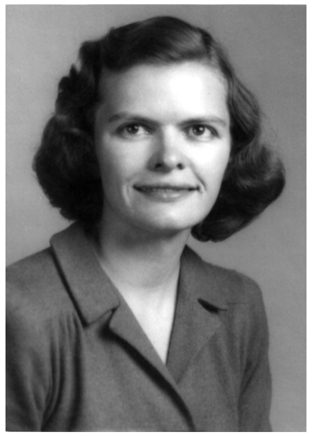
Recall: Hilbert's Challenges

- Hilbert's 10th problem [1900]:
  - Given a multivariate polynomial with integer coefficients, is there a **process that determines in a finite number of operations** whether the equation is solvable
  - **No:** Martin Davis, Yuri Matiyasevich, Hilary Putnam and Julia Robinson [1970]

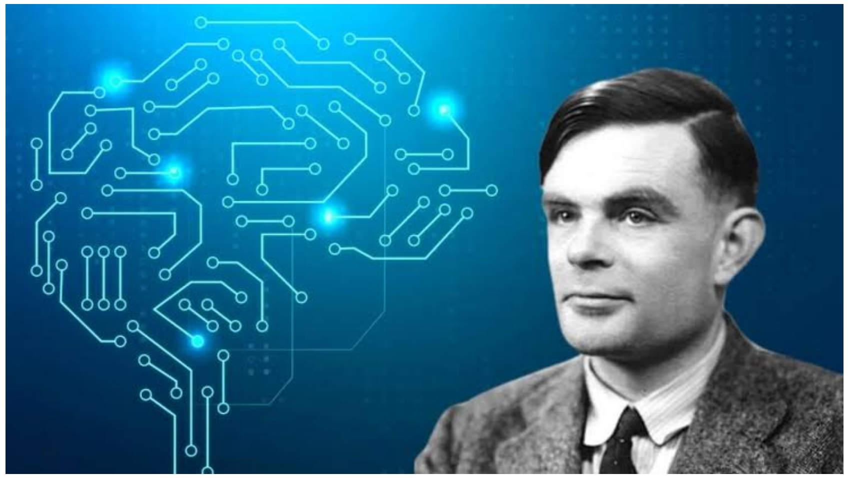








- Hilbert's Entscheidungsproblem (Decision problem) [1928]:
  - **Is there a finite procedure** that determines whether a given mathematical statement is true or false?
  - **No:** Alan Turing [1936]



Church-Turing Thesis Discussion

- A natural law of computation
  - Similar to laws of other physical sciences
- Scott Aaronson: ***"whatever it is, the Church-Turing thesis can only be regarded as extremely successful"***
- No candidate computing device (including quantum computers) have posed a serious challenge to it
    - New devices might make things more efficient but do not change the nature of what is fundamentally computable

Midterm Review/
Practice Midterm Questions