香港中文大學（深圳）
The Chinese University of Hong Kong, Shenzhen

SCHOOL OF DATA SCIENCE
數據科學學院

CSC 6011 - Theory of Computation - Lecture 6: TM Variants, the Church-Turing Thesis

Time: Mon & Wed, 4:00 - 5:20 PM

Location: Teaching A, 107

Instructor: Tao LIN (林涛)

TA: Yefan GAO (高叶繁)

Fall 2026

Lecture 6

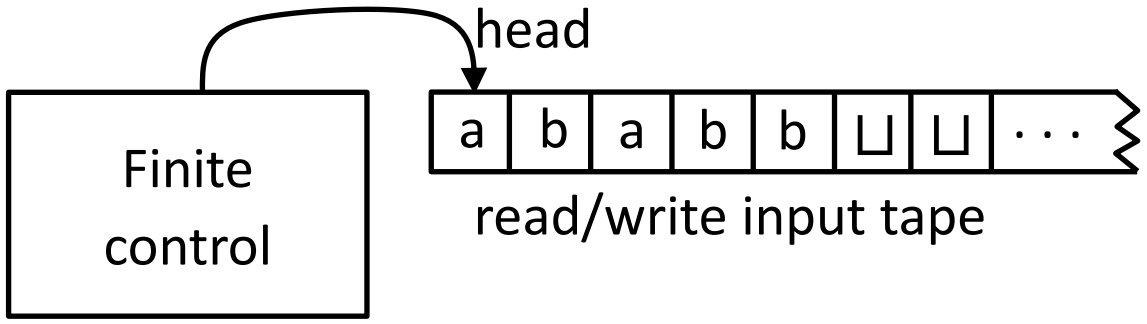
Last time:

- Proving languages are not context free
- Turing machines
- Turing-recognizable and Turing-decidable languages

Today:

- Equivalence of variants of the Turing machine model:
  - Multi-tape TMs, nondeterministic TMs, and enumerators
- Church-Turing Thesis
- Notation for encodings and TMs

Turing machine model – review



On input w, a TM M may halt or loop.

Three possible outcomes:

1. *Accept* w (enter q_{acc})
2. *Reject* w by halting (enter q_{rej})
3. *Reject* w by looping (running forever).

Recognizers and deciders

A is T-recognizable if $A = L(M)$ for some TM M.

A is T-decidable if $A = L(M)$ for some TM decider M (halts on every input).

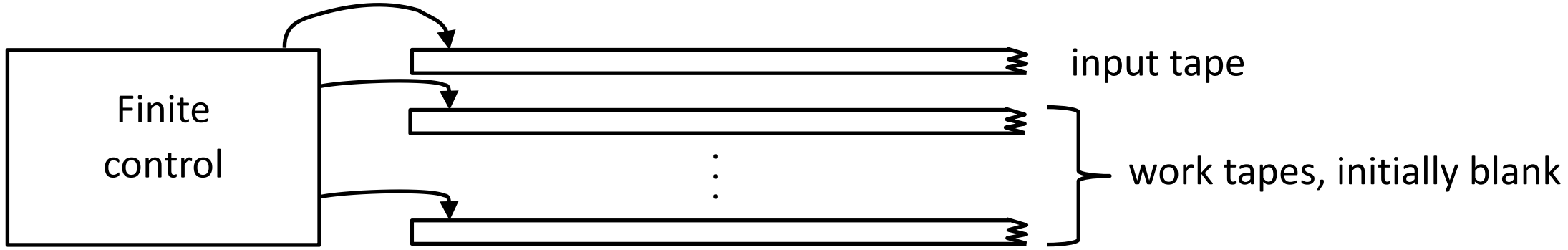
Why this model?

Turing machines model general-purpose computation.

All reasonable models are equivalent in power (we will show some examples).

Virtues of TMs: simplicity and familiarity.

Multi-tape Turing machines

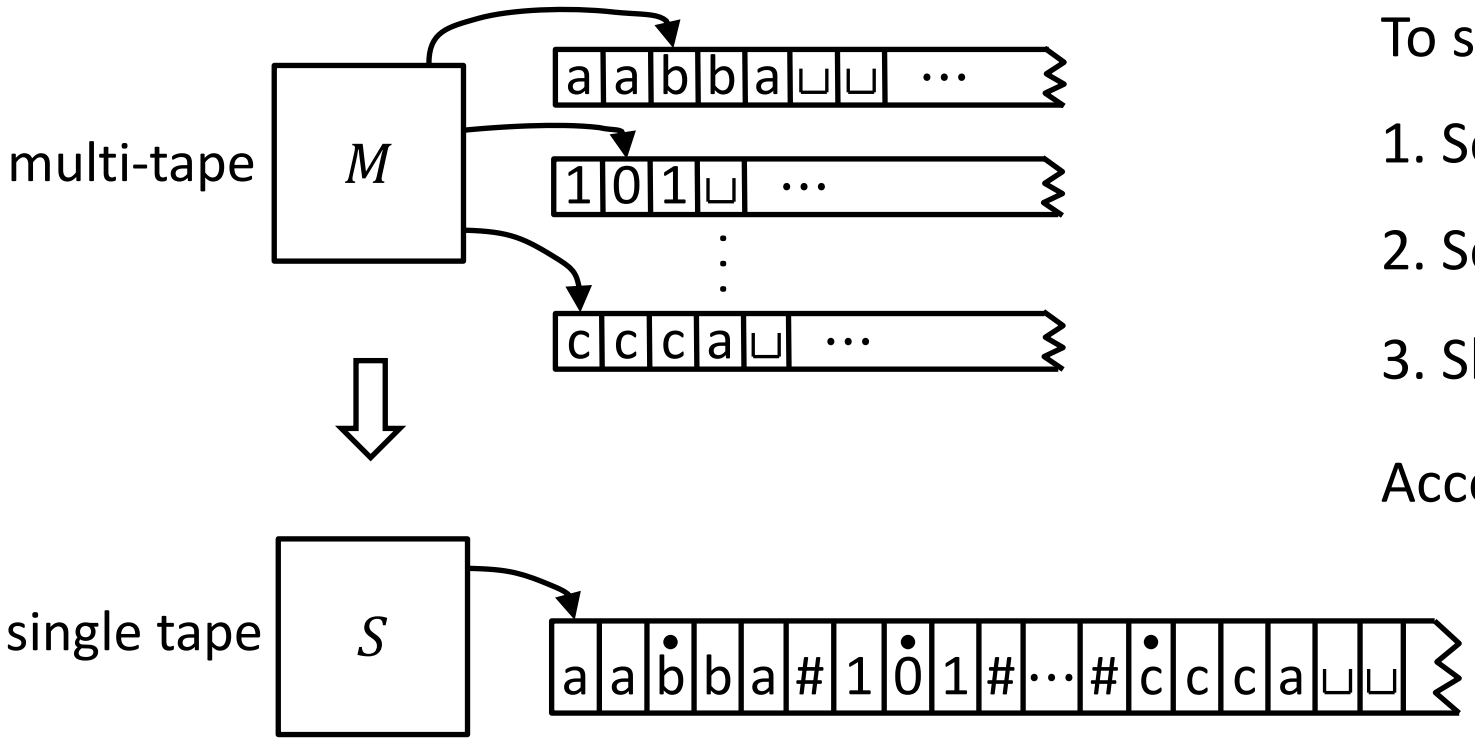


Each transition reads and writes all tapes.

Theorem: A is T-recognizable iff some multi-tape TM recognizes A

Proof: $(\rightarrow)$ immediate. $(\leftarrow)$ convert multi-tape to single tape:

Simulating multiple tapes with one tape



To simulate one step of M:

1. Scan to find all dotted symbols.
2. Scan again and update using M's δ.
3. Shift symbols to add room if needed.

Accept/reject when M does.

S stores M's tapes in multiple blocks on one tape.

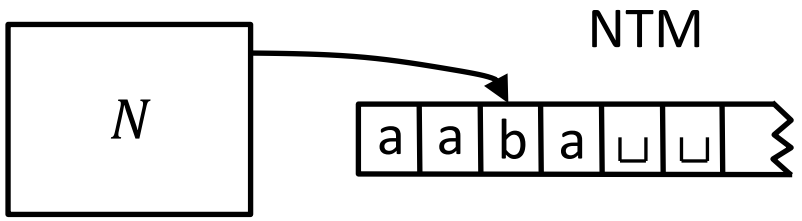
Record head positions using dotted symbols.

Nondeterministic Turing machines

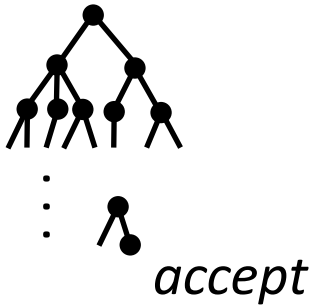
A Nondeterministic TM (NTM) is similar to a Deterministic TM except for its transition function δ: $Q \times \Gamma \to \mathcal{P}(\, Q \times \Gamma \times \{\text{L, R}\}\,)$.

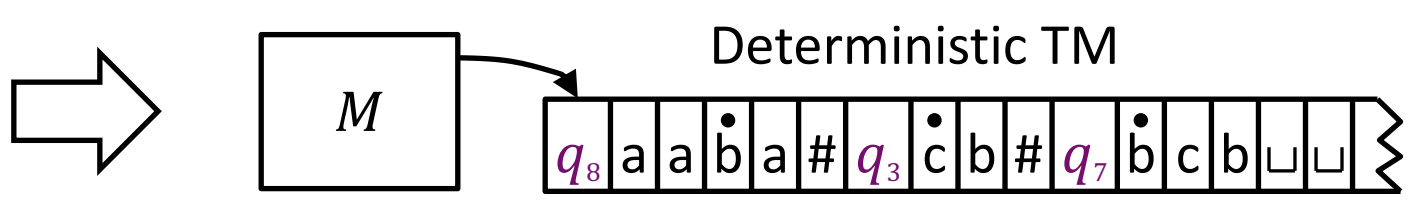
Theorem: A is T-recognizable iff some NTM recognizes A

Proof: $(\to)$ immediate. $(\leftarrow)$ convert NTM to Deterministic TM.



Nondeterministic computation tree for N on input w.





M stores the threads' tapes and head positions in blocks;

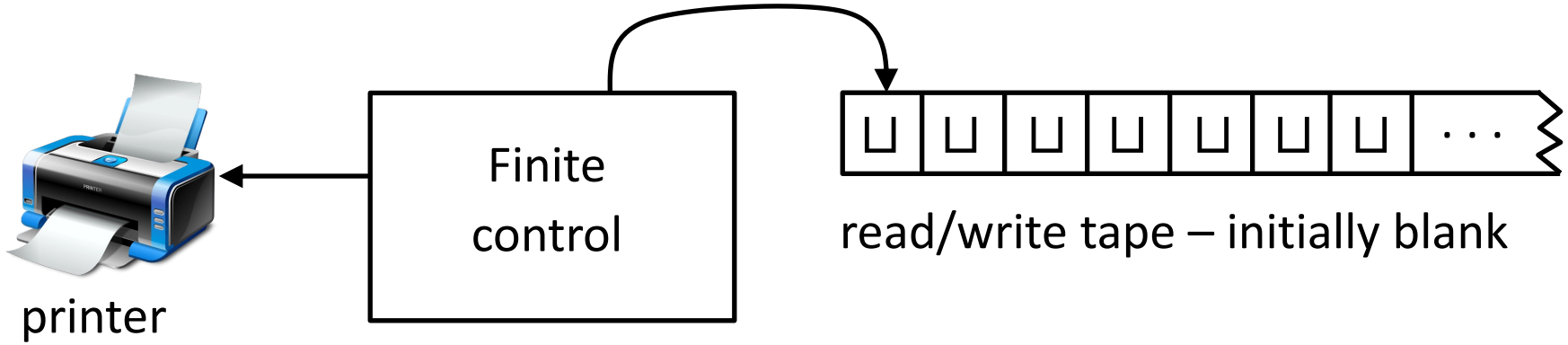
also store their states.

Advance all active threads one step per round.

If a thread forks, copy its block.

If any thread accepts, M accepts.

Turing enumerators



Defn: A Turing Enumerator is a deterministic TM with a printer.
It starts on a blank tape and it can print strings $w_1, w_2, w_3, \ldots$, possibly going forever.

The language of a T-enumerator E is the set of all strings it prints:

$$L(E) = \{w \mid E \text{ prints } w\}$$

E is a generator, not a recognizer.

Theorem: A is T-recognizable iff $A = L(E)$ for some T-enumerator E.

Enumerators and recognizers are equivalent

Theorem: A is T-recognizable iff $A = L(E)$ for some T-enumerator E.

Proof ($\leftarrow$): Convert E to equivalent TM M.

$M =$ for input w:

Simulate E (on blank input).

Whenever E prints x, test $x = w$:

- If $x = w$: Accept;

- Otherwise: continue.

Proof ($\rightarrow$): Convert TM M to enumerator E

Attempt:

E enumerates w_i in Σ^*= {ε, 0, 1, 00, 01, 10, …}

Simulate M on w_i , print w_i if M accepts.

Problem:

What if M on w_i loops?

Fix:

For i = 1, 2, …, run M on $w_1, w_2, \ldots, w_i$ for i steps.

Print every string accepted in these simulations.

Enumerators and recognizers are equivalent

Theorem: A is T-recognizable iff $A = L(E)$ for some T-enumerator E.

Proof ($\leftarrow$): Convert E to equivalent TM M.

$M =$ for input w:

Simulate E (on blank input).

Whenever E prints x, test $x = w$:

- If $x = w$: Accept;
- Otherwise: continue.

Proof ($\rightarrow$): Convert TM M to enumerator E

Attempt:

E enumerates w_i in Σ^*= {ε, 0, 1, 00, 01, 10, …}

Simulate M on w_i , print w_i if M accepts.

Problem:

What if M on w_i loops?

Fix:

For i = 1, 2, …, run M on $w_1, w_2, \ldots, w_i$ for i steps.

Print every string accepted in these simulations.

Bonus Question:

When converting TM M to enumerator E, does E always print strings in **string order**?

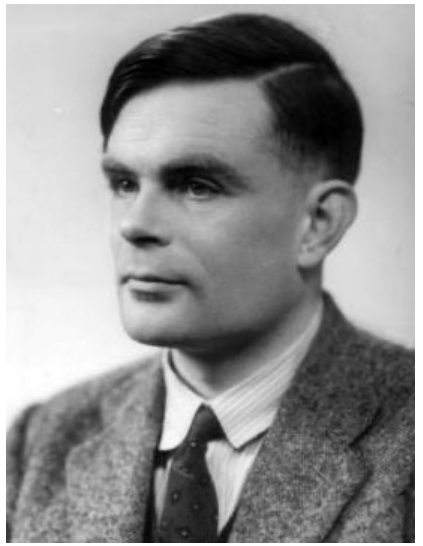
a) Yes. *Shorter string first.*

b) No. *Lexicographic order for same-length strings*

Church-Turing Thesis (~1936)

Alonzo Church
1903–1995

Algorithm	=	Turing machine
Intuitive		Formal



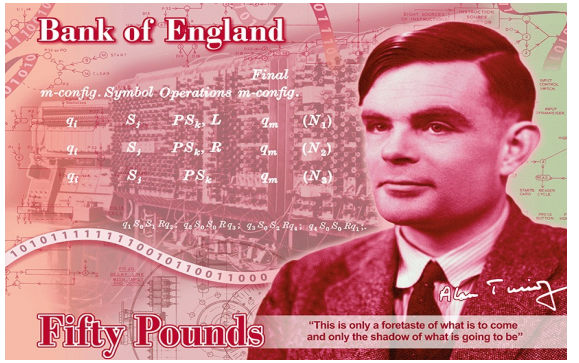
Alan Turing
1912–1954

Besides Turing machines,
many other models of computation
(with unrestricted memory):

λ-calculus, random access machine,
your favorite programming language,
…

Big impact on mathematics.

Issued in 2021



Hilbert's 10th Problem

In 1900 David Hilbert posed 23 problems

#1) Problem of the continuum: Does a set A exist with $|\mathbb{N}| < |A| < |\mathbb{R}|$?

#2) Prove that the axioms of mathematics are consistent.

#10) Give an algorithm for solving *Diophantine equations.*

David Hilbert
1862—1943

Diophantine equations:

Equations of polynomials where solutions must be integers.

Example: $3x^2 - 2xy - y^2z = 7$ solution: $x = 1,\ y = 2,\ z = -2$

Hilbert's 10th Problem as a language

#10) Give an algorithm for solving *Diophantine equations.*

Let $D = \{ p \mid$ polynomial $p(x_1, x_2, \ldots, x_k) = 0$ has a solution in integers $\}$

Rephrase Hilbert's 10th problem: *Give an algorithm (Turing machine) to decide* D.

Matiyasevich's theorem (1970): D is not decidable.

Note: D is T-recognizable.

To recognize D, enumerate integer tuples and test them until a solution is found.

Notation for encodings and TMs

Notation for encoding objects into strings

- If O is some object (e.g., polynomial, automaton, graph, etc.),
we write $\langle O \rangle$ to be an encoding of that object into a string.

- If $O_1, O_2, \ldots, O_k$ is a list of objects, then we write $\langle O_1, O_2, \ldots, O_k \rangle$
to be an encoding of them together into a single string.

Notation for writing Turing machines

We will use high-level English descriptions of algorithms when we describe TMs, knowing that we could (in principle) convert those descriptions into states, transition function, etc. Our notation for writing a TM M is

$M =$ "On input w

[English description of the algorithm]"

Notation for encodings and TMs

Notation for encoding objects into strings

- If O is some object (e.g., polynomial, automaton, graph, etc.), we write $\langle O \rangle$ to be an encoding of that object into a string.

- If $O_1, O_2, \ldots, O_k$ is a list of objects, then we write $\langle O_1, O_2, \ldots, O_k \rangle$ to be an encoding of them together into a single string.

Notation for writing Turing machines

We will use high-level English descriptions of a
knowing that we could (in principle) convert th
function, etc. Our notation for writing a TM M

$M =$ "On input w

[English description of the algorithm]"

Mini-Quiz

If x and y are strings, is xy a good choice for their encoding $\langle x, y \rangle$ into one string?

a) Yes.

b) No.

TM – example revisited

TM M recognizing $B = \{\mathrm{a}^k\mathrm{b}^k\mathrm{c}^k \mid k \geq 0\}$

M = "On input w

1. Check if $w \in \mathrm{a}^*\mathrm{b}^*\mathrm{c}^*$, *reject* if not.
2. Count the number of a's, b's, and c's in w.
3. *Accept* if all counts are equal; *reject* if not."

High-level description is ok, but make sure each step can be implemented.

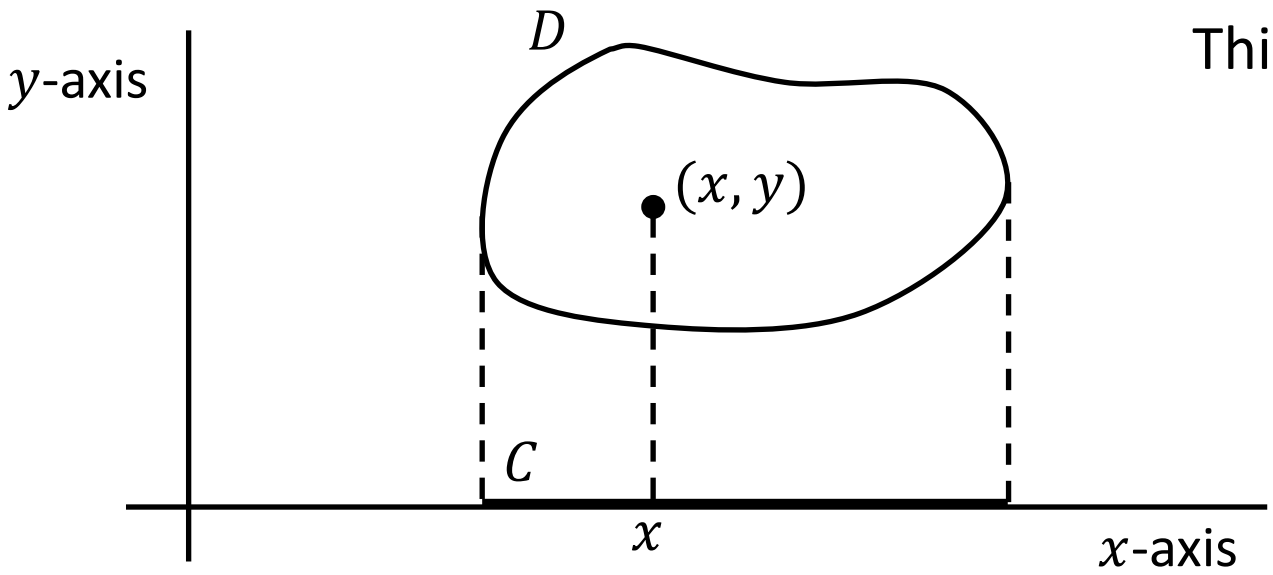
You do not need to manage tapes, states, etc…

Example use of encoding

Claim: C is T-recognizable iff there is a T-decidable D where

$$C = \{\, x \mid \exists y \;\; \langle x, y \rangle \in D \,\} \qquad x, y \in \Sigma^*$$

$\langle x, y \rangle$ is an encoding of the pair of strings x and y into a single string.



Think of D as a collection of pairs of strings.

C is a "projection" of D

Bonus question:
How to prove this claim?

Quick review of today

1. TM variants (multi-tape, nondeterministic, and enumerators) are all equivalent to the single-tape model.
2. All reasonable models with unrestricted memory are equivalent.
3. Church-Turing Thesis: Turing machines are equivalent to “algorithms”.
4. Encodings let us describe objects and TMs as strings.

These slides are adapted from Michael Sipser, *18.404J Theory of Computation*, Fall 2020. Massachusetts Institute of Technology:

- Source: MIT OpenCourseWare https://ocw.mit.edu