COMP30026 Models of Computation

Lecture 18: TM Variants and Decidable Languages

Mak Nazecic-Andrlon and William Umboh

Semester 2, 2024

Some material from Michael Sipser's slides

Pre-Lecture Polls

TM recognizing $B = \{\mathrm{a}^k\mathrm{b}^k\mathrm{c}^k \mid k \geq 0\}$ (how to program this?)

1) Scan right until $\sqcup$ while checking if input is in $\mathrm{a}^*\mathrm{b}^*\mathrm{c}^*$, *reject* if not.
2) Return head to left end.
3) Scan right, crossing off single a, b, and c.
4) If the last one of each symbol, *accept*.
5) If the last one of some symbol but not others, *reject*.
6) If all symbols remain, return to left end and repeat from (3).

Diophantine equations:

Equations of polynomials where solutions must be integers.
Example: $3x^2 - 2xy - y^2z = 7$ integer solution: $x = 1,\ y = 2,\ z = -2$

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

Show that D is Turing-recognizable

Where are we?

Last time:
Turing machines
- Recognizers and deciders
- Turing-recognizable and Turing-decidable languages
- Church-Turing Thesis

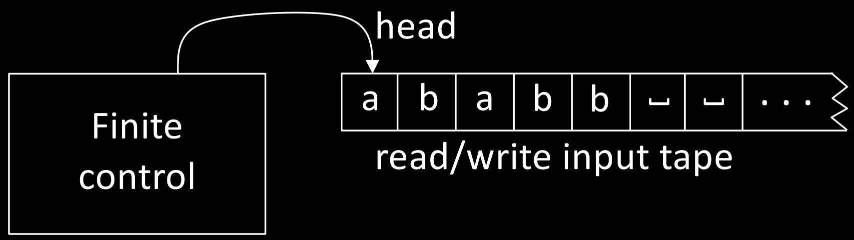
Equivalence of variants of the Turing machine model
- Enumerators

Today: (Sipser §3.2 – §4.1)
Equivalence of variants of the Turing machine model
- Multi-tape TMs
- Nondeterministic TMs

Notation for encodings and TMs
Decision procedures for automata and grammars
A_{DFA}, A_{NFA}, E_{DFA}, A_{CFG}, E_{CFG} are decidable

Technique: Simulation

Turing machine model – review



On input w a TM M may halt (enter q_{acc} or q_{rej}) or loop (run forever).

So M has 3 possible outcomes for each input w:

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

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 all inputs

Turing machines model general-purpose computation.

Q: Why pick this model?

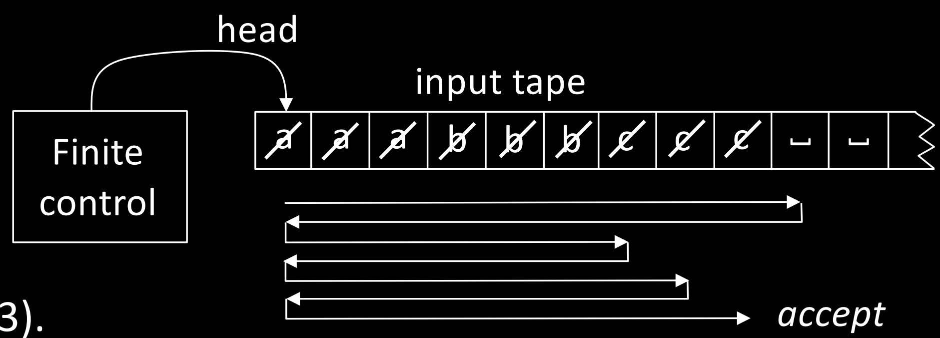
A: C-T Thesis: Choice of model doesn't matter.
All reasonable models are equivalent in power.

Virtues of TMs: simplicity, familiarity.

TM – Informal example

TM recognizing $B = \{a^k b^k c^k \mid k \geq 0\}$ (how to program this?)

1) Scan right until $\sqcup$ while checking if input is in $a^* b^* c^*$, *reject* if not.
2) Return head to left end.
3) Scan right, crossing off single a, b, and c.
4) If the last one of each symbol, *accept*.
5) If the last one of some symbol but not others, *reject*.
6) If all symbols remain, return to left end and repeat from (3).



Check-in 17.1

How do we get the effect of "crossing off" with a Turing machine?

a) We add that feature to the model.

b) We use a tape alphabet $\Gamma = \{a, b, c, \cancel{a}, \cancel{b}, \cancel{c}, \sqcup\}$.

c) All Turing machines come with an eraser.

Hilbert's 10th Problem

In 1900 David Hilbert posed 23 problems

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

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

Diophantine equations:

Equations of polynomials where solutions must be integers.

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

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

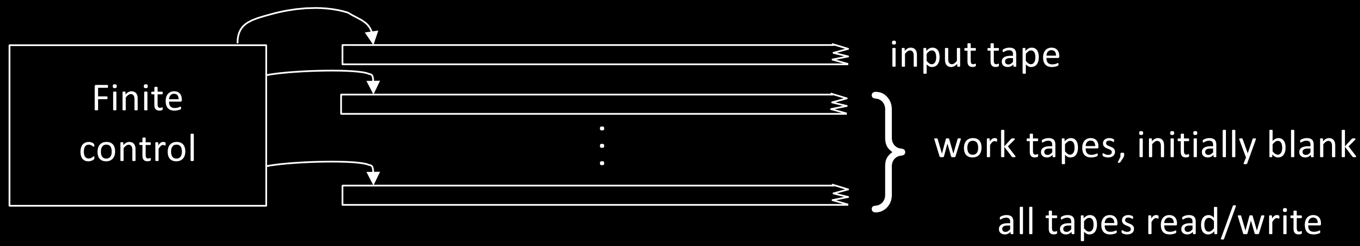
Hilbert's 10th problem: Give an algorithm to decide D.

Matiyasevich proved in 1970: D is not decidable.

Exercise: D is T-recognizable.

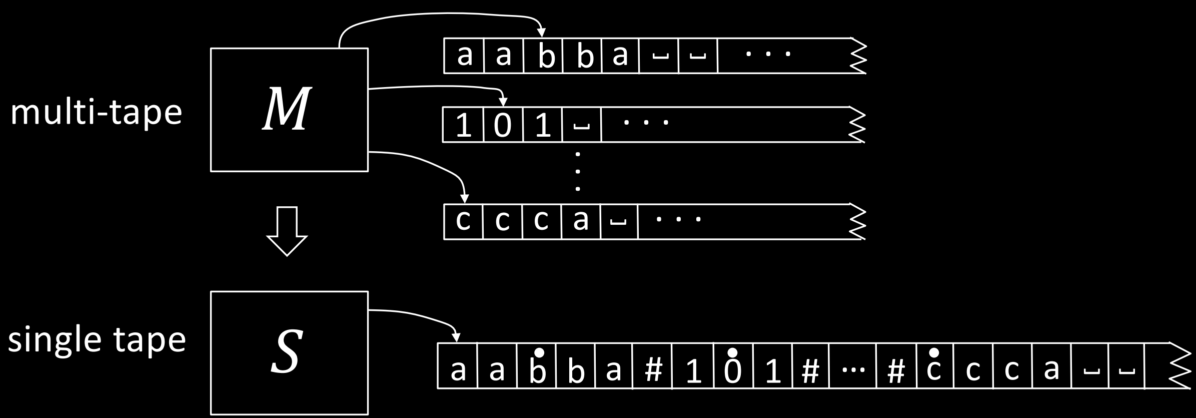
David Hilbert
1862—1943

Multi-tape Turing machines



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

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

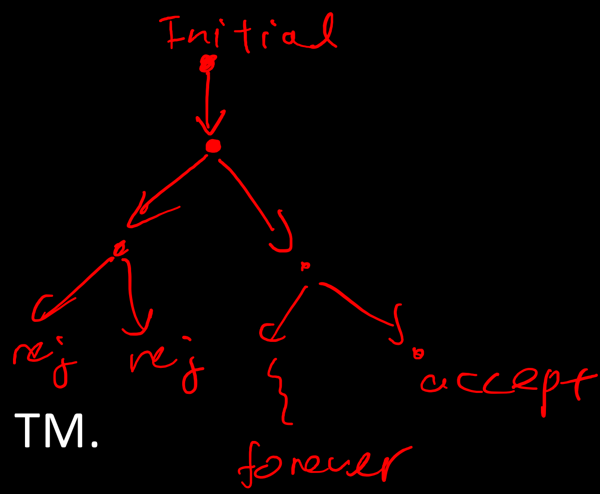


S simulates M by storing the contents of multiple tapes on a single tape in "blocks". Record head positions with dotted symbols.

Some details of S:

1) To simulate each of M's steps
   a. Scan entire tape to find dotted symbols.
   b. Scan again to update according to M's δ.
   c. Shift to add room as needed.
2) Accept/reject if M does.

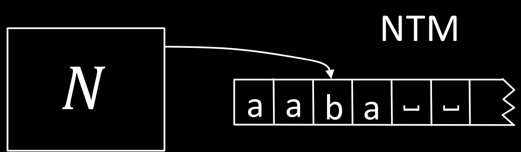
Nondeterministic Turing machines



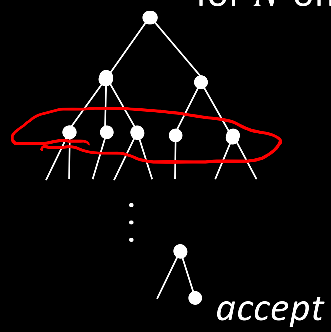
A <u>Nondeterministic TM</u> (NTM) is similar to a Deterministic TM except for its transition function $\delta\colon \mathrm{Q}\times\Gamma \to \mathcal{P}(\, Q\times\Gamma\times \{\mathrm{L, R}\}\,)$.

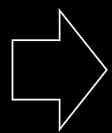
Theorem: A is T-recognizable iff some NTM recognizes A

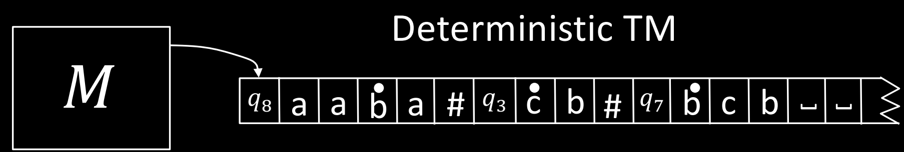
Proof: $(\rightarrow)$ immediate. $(\leftarrow)$ convert NTM to Deterministic TM.



Nondeterministic computation tree for N on input w.







M simulates N by storing each thread's tape in a separate "block" on its tape.
Also need to store the head location,
and the state for each thread, in the block.

If a thread forks, then M copies the block.

If a thread accepts then M accepts.

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. Every discrete object has a string representation!

- 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.

$O_1 \# O_2 \# \cdots \# O_k$

xy

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]"

Check-in 18.1

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

a) Yes.

b) No.

Acceptance Problem for DFAs

Let $A_{\mathrm{DFA}} = \{\langle B, w\rangle \mid B$ is a DFA and B accepts $w\}$

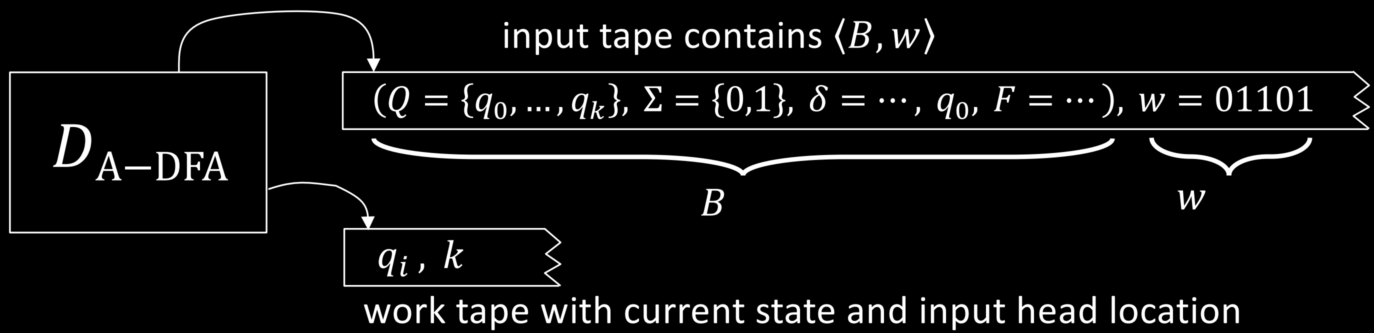
Theorem: A_{DFA} is decidable

Proof: Give TM $D_{\mathrm{A-DFA}}$ that decides A_{DFA}.

$D_{\mathrm{A-DFA}}$ = "On input s

1. Check that s has the form $\langle B, w\rangle$ where B is a DFA and w is a string; *reject* if not.
2. Simulate the computation of B on w.
3. If B ends in an accept state then *accept*. If not then *reject*."

Shorthand:
On input $\langle B, w\rangle$



Acceptance Problem for NFAs

Let $A_{\mathrm{NFA}} = \{\langle B, w\rangle |\ B$ is a NFA and B accepts $w\}$

Theorem: A_{NFA} is decidable

Proof: Give TM $D_{\mathrm{A-NFA}}$ that decides A_{NFA}.

$D_{\mathrm{A-NFA}} =$ "On input $\langle B, w\rangle$

1. Convert NFA B to equivalent DFA B'.
2. Run TM $D_{\mathrm{A-DFA}}$ on input $\langle B', w\rangle$. [Recall that $D_{\mathrm{A-DFA}}$ decides A_{DFA}]
3. *Accept* if $D_{\mathrm{A-DFA}}$ accepts.
   Reject if not."

New technique: Reduction

Use conversion construction and previously constructed TM as a subroutine.

We say that we reduce A_{NFA} to A_{DFA}

Emptiness Problem for DFAs

Let $E_{\text{DFA}} = \{\langle B \rangle \mid B$ is a DFA and $L(B) = \emptyset\}$

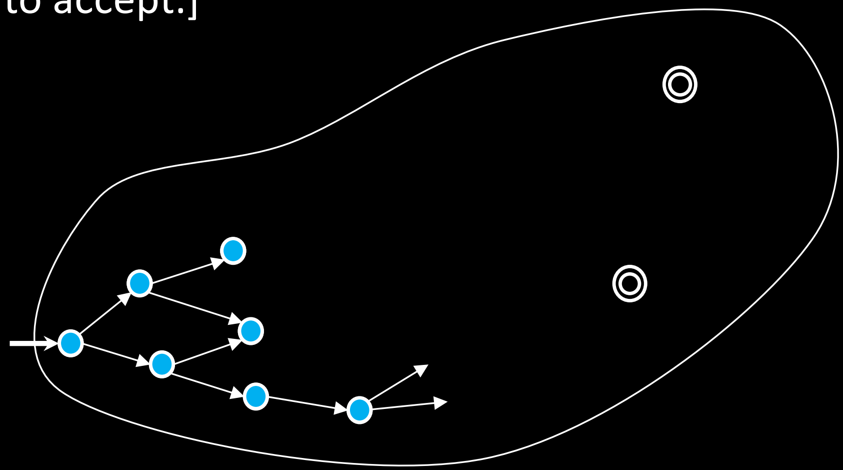
Theorem: E_{DFA} is decidable

Proof: Give TM $D_{\text{E-DFA}}$ that decides E_{DFA}.

$D_{\text{E-DFA}} =$ "On input $\langle B \rangle$ [IDEA: Check for a path from start to accept.]

1. Mark start state.
2. Repeat until no new state is marked:

   Mark every state that has an incoming arrow from a previously marked state.
3. *Accept* if no accept state is marked.
   Reject if some accept state is marked."



Emptiness Problem for DFAs

Let $E_{\mathrm{DFA}} = \{\langle B\rangle | \ B$ is a DFA and $L(B) = \emptyset\}$

Theorem: E_{DFA} is decidable

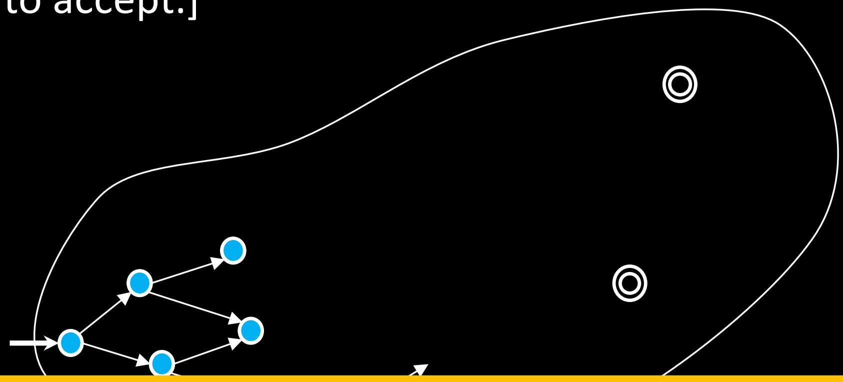
Proof: Give TM $D_{\mathrm{E-DFA}}$ that decides E_{DFA} .

$D_{\mathrm{E-DFA}} =$ "On input $\langle B\rangle$ [IDEA: Check for a path from start to accept.]

1. Mark start state.
2. Repeat until no new state is marked:

   Mark every state that has an incoming arrow from a previously marked state.
3. *Accept* if no accept state is marked.

   Reject if some accept state is marked."



Check-in 18.2

Can we use the path from start to accept to find a string accepted by *B?*

a) Yes.

b) No.

Quick review of today

1. Equivalence of variants of the Turing machine model
    1. Multi-tape TMs
    2. Nondeterministic TMs
2. Notation for encodings and TMs
3. We showed the decidability of various problems about automata and grammars:
   A_{DFA} , A_{NFA} , E_{DFA}