Lecture 11:

• TM Variants and Closure Properties
• Church-Turing Thesis

Reading:

Sipser Ch 3.2

Mark Bun
March 1, 2021
TM Variants
TMs are equivalent to...

- TMs with “stay put”
- TMs with 2-way infinite tapes
- Multi-tape TMs
- Nondeterministic TMs
- Random access TMs
- Enumerators
- Finite automata with access to an unbounded queue
- Primitive recursive functions
- Cellular automata

...
Multi-Tape TMs

Fixed number of tapes $k$ (can’t change during computation)
Transition function $\delta : Q \times \Gamma^k \rightarrow Q \times \Gamma^k \times \{L, R, S\}^k$
How to Simulate It

To show that a TM variant is no more powerful than the basic, single-tape TM:

Show that if $M$ is any variant machine, there exists a basic, single-tape TM $M'$ that can simulate $M$

(Usual) parts of the simulation:
• Describe how to initialize the tapes of $M'$ based on the input to $M$
• Describe how to simulate one step of $M$’s computation using (possibly many steps of) $M'$
Multi-Tape TMs are Equivalent to Single-Tape TMs

**Theorem:** Every $k$-tape TM $M$ with can be simulated by an equivalent single-tape TM $M'$

1) Initialize $m'$ tape
2) Simulate each step of $m$ on $m'$

$M$'s first tape
$M$'s second tape
3rd tape
Simulating Multiple Tapes

Implementation-Level Description of $M'$

On input $w = w_1 w_2 \ldots w_n$

1. Format tape into $\# w_1 \# w_2 \# \ldots \# w_n \# L \# L \# \# \ldots \#$

2. For each move of $M$:
   - Scan left-to-right, finding current symbols
   - Scan left-to-right, writing new symbols, $\circ$
   - Scan left-to-right, moving each tape head $\leftrightarrow$

   If a tape head goes off the right end, insert blank
   If a tape head goes off left end, move back right
Why are Multi-Tape TMs Helpful?

To show a language is Turing-recognizable or decidable, it’s enough to construct a multi-tape TM

Often easier to construct multi-tape TMs

Ex. Decider for \{a^i b^j | i > j\}

On input w:

1) Check w/ a left-right scan \( w \in L(a^* b^*) \)

2) Copy all b’s from w to tape 2

3) Starting from left ends of tapes 1 and 2, check that every b on tape 2 has an accompanying a on tape 1

4) Check first blank on tape 2 has an accompanying a on tape 1
Why are Multi-Tape TMs Helpful?

To show a language is Turing-recognizable or decidable, it’s enough to construct a multi-tape TM

Very helpful for proving closure properties

Ex. Closure of recognizable languages under union. Suppose $M_1$ is a single-tape TM recognizing $L_1$, $M_2$ is a single-tape TM recognizing $L_2$.

Design 2-tape TM recognizing $L_1 \cup L_2$:

First attempt:

- On input $w$:
  1) Copy $w$ to tape 2
  2) Run $M_1$ on tape 2. If accepts, accept. If rejects, go on.
  3) Copy $w$ back to tape 2
  4) Run $M_2$ on tape 2. If accepts, accept. If rejects, reject.

What’s wrong with this construction?
Why are Multi-Tape TMs Helpful?

To show a language is Turing-recognizable or decidable, it’s enough to construct a multi-tape TM.

Very helpful for proving **closure properties**

**Ex.** Closure of recognizable languages under union. Suppose $M_1$ is a single-tape TM recognizing $L_1$, $M_2$ is a single-tape TM recognizing $L_2$.

[Correct attempt: 3-tape TM]

On input $w$:

1) **Copy $w$ to tape 2 and to tape 3**

2) Repeat forever:
   - Run $M_1$ on tape 2 for 1 step
   - Run $M_2$ on tape 3 for 1 step
   - If either accepts, accept.
Closure Properties

The Turing-decidable languages are closed under:

- Union
- Concatenation
- Star
- Intersection
- Reverse
- Complement

The Turing-recognizable languages are closed under:

- Union
- Concatenation
- Star
- Intersection
- Reverse
Nondeterministic TMs

At any point in computation, may nondeterministically branch. Accepts iff there exists an accepting branch.

Transition function $\delta : Q \times \Gamma \rightarrow P(Q \times \Gamma \times \{L, R, S\})$
Nondeterministic TMs

At any point in computation, may nondeterministically branch. Accepts iff there exists an accepting branch.
Nondeterministic TMs

At any point in computation, may nondeterministically branch. Accepts iff there exists an accepting branch.

What is the language of this NTM?

On input $w \in \{a, b\}^*$

1) Scan $w$ left to right, at some point, nondeterministically go to step 2

2) Read next character, call this $s$
   - Cross it off
   - Move head left. If char matches $s$, cross it off, move head right until reaches a $\textit{non-x}$
   - Repeat

3) Once read $w$, check that string is all $x$'s and accept if so. Else reject
Nondeterministic TMs

Ex. Given TMs $M_1$ and $M_2$, construct an NTM recognizing $L(M_1) \cup L(M_2)$

On input $w$:

1) Nondeterministically either:
   a) Run $M_1$ on $w$, accept if accepts, reject if rejects
   b) Run $M_2$ on $w$, accept if accepts, reject if rejects
Nondeterministic TMs

$L:\$

Ex. NTM for \( \{ w \mid w \text{ is a binary number representing the product of two positive integers } a, b \} \)

On input \( w \):

1) Non deterministically guess \( a \in \{2, \ldots, w^3\}, b \in \{2, \ldots, w^3\} \)

2) Check \( a \times b = w \): accept if so, reject otherwise.

Analysis: If \( w \in L \), \( \exists a, b \in \{2, \ldots, w^3\} \text{ s.t. } a \times b = w \)

\( \Rightarrow \) branch of computation where guessed \( a = \hat{a}, b = \hat{b} \) leads to accept.

If \( w \notin L \), all choices of \( a, b \) will lead to \( a \times b \neq w \)

\( \Rightarrow \) all branches lead to reject.
Nondeterministic TMs

An NTM $N$ accepts input $w$ if when run on $w$ it accepts on at least one computational branch

$L(N) = \{w \mid N \text{ accepts input } w\}$

An NTM $N$ is a decider if on every input, it halts on every computational branch