Lecture 12:

- TM Variants
- Decidable Languages

Reading:
Sipser Ch 3.2, 4.1

Mark Bun
March 4, 2020
Recognizers vs. Deciders

\[ L(M) = \text{the set of all strings } w \text{ which } M \text{ accepts} \]

A is Turing-recognizable if \( A = L(M) \) for some TM \( M \):
- \( w \in A \implies M \) halts on \( w \) in state \( q_{\text{accept}} \)
- \( w \notin A \implies M \) halts on \( w \) in state \( q_{\text{reject}} \) OR \( M \) runs forever

A is (Turing-)decidable if \( A = L(M) \) for some TM \( M \) which halts on every input
- \( w \in A \implies M \) halts on \( w \) in state \( q_{\text{accept}} \)
- \( w \notin A \implies M \) halts on \( w \) in state \( q_{\text{reject}} \)
TM Variants
Extensions that do not increase the power of the TM model

• TMs with a 2-way infinite tape, unbounded left to right

Proof that TMs with 2-way infinite tapes are no more powerful:

Simulation: Convert any TM $M$ with 2-way infinite tape into a 1-way infinite TM $M'$ with a “two-track tape”
Formalizing the Simulation

\[ M' = (Q', \Sigma, \Gamma', \delta', q_0', q_{\text{accept}}', q_{\text{reject}}') \]

New tape alphabet: \( \Gamma' = (\Gamma \times \Gamma) \cup \{\$\} \)

New state set: \( Q' = Q \times \{+, -\} \)

\((q, -)\) means “\(q\), working on upper track”
\((q, +)\) means “\(q\), working on lower track”

New transitions:

If \( \delta(p, a_-) = (q, b, L) \), let \( \delta'( (p, -), (a_-, a_+) ) = ((q, -), (b, a_+), R) \)

Also need new transitions for moving right, lower track, hitting $, initializing input into 2-track format
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 = 2-stack PDAs
- Primitive recursive functions
- Cellular automata
- “Turing-complete” programming languages (C, Python, Java...)

...
Church-Turing Thesis

The equivalence of these models is a **mathematical theorem**

Church-Turing *Thesis*: Each of these models captures our intuitive notion of algorithms

The Church-Turing Thesis is **not** a mathematical statement!
Multi-Tape TMs

Fixed number of tapes $k$ (can’t change during computation)

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

Finite control

Finite control

The new symbol do more handshakes between tapes
Simulating Multiple Tapes

Implementation-Level Description

On input $w = w_1w_2 \ldots w_n$

1. Format tape into $\# w_1w_2 \ldots w_n \# \# \# \# \# \# \ldots \#$

2. For each move of $M$:
   - Scan left-to-right, storing current symbols in finite control
   - Scan left-to-right, writing new symbols,
   - Scan left-to-right, moving each tape head

   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, suffices 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 $L_2$

Proof via implementation description:
reconstruct 2-tape TM recognizing $L_1 \cup L_2$

On input $w$:
1. Scan Tape 1 left-to-right, copying characters onto Tape 2
2. Move both heads to left end of tapes
3. Repeatedly:
   - Simulate 1 step of $M_1$ on Tape 1
   - Simulate 1 step of $M_2$ on Tape 2
   - If either machine accept, accept
Nondeterministic TMs

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

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

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

```
1. Non-deterministically "guess" $a, b \leq w$
2. Compute $a \times b$, check if equal to $w$, accept if so.
3. Reject otherwise
```
Nondeterministic TMs

**Theorem:** Every nondeterministic TM has an equivalent deterministic TM

**Proof idea:**
Systematically try all 1-step computations, all 2-step computations, ... and see if one of them accepts
Nondeterministic TMs

**Theorem:** Every nondeterministic TM has an equivalent deterministic TM

**Proof idea:** Simulate an NTM $N$ using a 3-tape TM

```
Finite control
```

```
\begin{array}{c}
  w_1 \mid w_2 \mid w_3 \mid w_4 \\
  w_1 \mid \# \mid w_3 \mid w_4 \\
  1 \mid 3 \mid 3 \mid 7 \\
\end{array}
```

1. **Input $w$ to $N$ (read-only)**
2. **Simulation tape (run $N$ on $w$ using nondeterministic choices from tape 3)**
3. **Address in computation tree**

3/4/2020  CS332 - Theory of Computation
Deterministic

\[ q_0 \]

\[ q_1 \]

\[ q_2 \]

\[ q_n \]

Addresses

First step: 1, 2, ..., B
Second step: 11, 12, 13, 14, ..., 1B, 21, 22, ...

Branching factor B

Non-deterministic

\[ q_0 \]
Enumerators

- Starts with two blank tapes
- Prints strings to printer

\[ L(E) = \{ \text{strings eventually printed by } E \} \]
- May never terminate (even if language is finite)
- May print the same string many times
Enumerable = Turing-Recognizable

**Theorem:** A language is Turing-recognizable \iff some enumerator enumerates it
\iff Start with an enumerator $E$ for $A$ and give a TM
Enumerable = Turing-Recognizable

**Theorem:** A language is Turing-recognizable if and only if some enumerator enumerates it.

⇒ Start with a TM $M$ for $A$ and give an enumerator.
Mid-Semester Feedback Form

https://forms.gle/LTBEY1BoSZh8nupV6