Lecture 10:

- Turing Machines
- TM Variants and Closure Properties

Reading:
Sipser Ch 3.1-3.3

Mark Bun
October 13, 2022

https://forms.gle/44vcjAzahbobkuAQ8
The Basic Turing Machine (TM)

- Input is written on an infinitely long tape
- Head can both read and write, and move in both directions
- Computation halts as soon as control reaches “accept” or “reject” state
Three Levels of Abstraction

High-Level Description
An algorithm (like CS 330)

Implementation-Level Description
Describe (in English) the instructions for a TM
• How to move the head
• What to write on the tape

Low-Level Description
State diagram or formal specification
Example

Determine if a string $w \in \{0\}^*$ is in the language

$$A = \{0^{2^n} \mid n \geq 0\}$$

Not a regular language

High-Level Description

Repeat the following forever:

- If there is exactly one 0 in $w$, accept
- If there is an odd (> 1) number of 0s in $w$, reject
- Delete half of the 0s in $w$
Example

Determine if a string $w \in \{0\}^*$ is in the language $A = \{0^{2^n} \mid n \geq 0\}$

Implementation-Level Description

1. While moving the tape head left-to-right:
   a) Cross off every other 0 \textit{(replace every other 0 with X)}
   b) If there is exactly one 0 when we reach the right end of the tape, \textbf{accept}
   c) If there is an odd (> 1) number of 0s when we reach the right end of the tape, \textbf{reject}

2. Return the head to the left end of the tape
3. Go back to step 1
Example

Determine if a string \( w \in A = \{0^{2^n} \mid n \geq 0\} \)

Low-Level Description
Differences between TMs and Finite Automata

TMs can use their head in both directions

TMs can write

TMs have unlimited memory in the form of the tape

TMs make a decision when they explicitly reach accept or reject vs. FAs make a decision when they reach the end of the input

TMs can loop forever w/o halting

TMs have one accept (and one reject) state
Formal Definition of a TM

A TM is a 7-tuple $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}})$

- $Q$ is a finite set of states
- $\Sigma$ is the input alphabet *(does not include $\Lambda$)*
- $\Gamma$ is the tape alphabet *(contains $\Lambda$ and $\Sigma$)*
- $\delta$ is the transition function
  
  \[ \delta : (Q \times \Sigma \rightarrow Q \times \Sigma \times \Gamma) \]
  
  e.g. $\delta(q, \sigma) = (q', \tau, \rho)$

  e.g. $\delta(q, \Lambda) = (q', \Lambda, \Lambda)$

- $q_0 \in Q$ is the start state
- $q_{\text{accept}} \in Q$ is the accept state
- $q_{\text{reject}} \in Q$ is the reject state ($q_{\text{reject}} \neq q_{\text{accept}}$)
**TM Transition Function**

\[ \delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\} \]

- \(L\) means “move left” and \(R\) means “move right”

\[ \delta(p, a) = (q, b, R) \]

- Replace \(a\) with \(b\) in current cell
- Transition from state \(p\) to state \(q\)
- Move tape head right

\[ \delta(p, a) = (q, b, L) \]

- Replace \(a\) with \(b\) in current cell
- Transition from state \(p\) to state \(q\)
- Move tape head left UNLESS we are at left end of tape, in which case don’t move
Configuration of a TM

A string that captures the **state** of a TM together with the **contents of the tape**

\[ 101q_50111 \]

\[ \begin{array}{ccccccc}
1 & 0 & 1 & 0 & 1 & 1 & 1 \\
\end{array} \]

\[ q_5 \]
Configuration of a TM: Formally

A configuration is a string $uqv$ where $q \in Q$ and $u, v \in \Gamma^*$

- Tape contents = $uv$ (followed by infinitely many blanks $\sqcup$)
- Current state = $q$
- Tape head on first symbol of $v$ \( \left( q \text{ written @ left of tape head location}\right) \)

Example: $101q_50111$

\[
\begin{array}{cccccccc}
1 & 0 & 1 & 0 & 1 & 1 & 1 & \sqcup \\
\end{array}
\]

\[q_5\]
How a TM Computes

Start configuration: $q_0 w$

One step of computation:

• If $\delta(q, b) = (q', c, R)$, then $u a q b v$ yields $u a c q' v$
• If $\delta(q, b) = (q', c, L)$, then $u a q b v$ yields $u q' a c v$
• If we are at the left end of the tape in configuration $q b v$, what configuration do we reach if $\delta(q, b) = (q', c, L)$?
  
  a) $c q' v$
  b) $q' c v$
  c) $q' \sqcup c v$
  d) $q' c b v$
How a TM Computes

Start configuration: $q_0 w$

One step of computation:
• If $\delta(q, b) = (q', c, R)$, then $ua q b v$ yields $u ac q' v$
• If $\delta(q, b) = (q', c, L)$, then $ua q b v$ yields $u q' ac v$
• If $\delta(q, b) = (q', c, L)$, then $q b v$ yields $q' cv$

Accepting configuration: $q = q_{\text{accept}}$

Rejecting configuration: $q = q_{\text{reject}}$
How a TM Computes

$M$ accepts input $w$ if there exists a sequence of configurations $C_1, \ldots, C_k$ such that:

• $C_1 = q_0 w$
• $C_i$ yields $C_{i+1}$ for every $i$
• $C_k$ is an accepting configuration

$L(M) = \text{the language recognized by } M$

$L(M)$ = the set of all strings $w$ which $M$ 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 on $w$
Recognizers vs. Deciders

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

\( A \) is \textbf{Turing-recognizable} if \( A = L(M) \) for some TM \( M \):

- \( w \in A \implies M \text{ halts on } w \text{ in state } q_{\text{accept}} \)
- \( w \notin A \implies M \text{ halts on } w \text{ in state } q_{\text{reject}} \) \text{ OR } \( M \text{ runs forever on } w \)

\( A \) is \textbf{(Turing-)decidable} if \( A = L(M) \) for some TM \( M \)

which halts on every input

- \( w \in A \implies M \text{ halts on } w \text{ in state } q_{\text{accept}} \)
- \( w \notin A \implies M \text{ halts on } w \text{ in state } q_{\text{reject}} \)
Recognizers vs. Deciders

Which of the following is true about the relationship between decidable and recognizable languages?

a) The decidable languages are a subset of the recognizable languages

b) The recognizable languages are a subset of the decidable languages

c) They are incomparable: There might be decidable languages which are not recognizable and vice versa
Example: Arithmetic on a TM

The following TM decides \( \text{MULT} = \{a^i b^j c^k \mid i \times j = k\} \):

On input string \( w \):

1. Check \( w \) is formatted correctly
   
2. For each \( a \) appearing in \( w \):
   
3. For each \( b \) appearing in \( w \):
   
4. Attempt to cross off a \( c \). If none exist, reject.

5. If all \( c \)'s are crossed off, accept. Else, reject.
Example: Arithmetic on a TM

The following TM decides \( \text{MULT} = \{a^i b^j c^k \mid i \times j = k\} \):

On input string \( w \):

1. Scan the input from left to right to determine whether it is a member of \( L(a^* b^* c^*) \).

2. Return head to left end of tape.

3. Cross off an \( a \) if one exists. Scan right until a \( b \) occurs. Shuttle between \( b \)'s and \( c \)'s crossing off one of each until all \( b \)'s are gone. Reject if all \( c \)'s are gone but some \( b \)'s remain.

4. Restore crossed off \( b \)'s. If any \( a \)'s remain, repeat step 3.

5. If all \( c \)'s are crossed off, accept. Else, reject.
Back to Hilbert’s Tenth Problem

**Computational Problem:** Given a Diophantine equation, does it have a solution over the integers?

$L = \{ p(x_1, \ldots, x_n) \mid n \geq 0, \exists (x_1, \ldots, x_n) \in \mathbb{Z}^n \text{ s.t. } p(x_1, \ldots, x_n) = 0 \}$

- $L$ is Turing-recognizable
- $L$ is not decidable (1949-70)

**Special case:** $L_2 = \{ p(x, y) \mid \ldots \}$

**Algorithm:** For each cell $(x, y)$ in grid, test if $p(x, y) = 0$; if so, accept

**Analysis:** If $p \in L_2$

Then $\exists (x, y) \in \text{grid s.t. } p(x, y) = 0 \Rightarrow \text{alg. accept}$

If $p \not\in L_2$, alg. loops forever

- $L$ is **not** decidable (1949-70)