Outline

– Language Hierarchy
– Definition of Turing Machine
– TM Variants and Equivalence
– Decidability
– Reducibility

Language Hierarchy

• Regular: finite memory
• CFG/PDA: infinite memory but in stack space
• TM: infinite and unrestricted memory
  – TM Decidable/Recursive
  – TM Recognizable/Recursively Enumerable

Semantics of TM

• Not a real machine, but a model of computation
• Components:
  – 1-way infinite tape: unlimited memory
    • Store input, output, and intermediate results
    • Infinite cells
    • Each cell has a symbol from a finite alphabet
  – Tape head:
    • Point to one cell
    • Read or write a symbol to that cell
    • move left or right
States of a TM

• Initial state:
  – Head on leftmost cell
  – input on the tape
  – Blank everywhere else

• Accept state

• Reject state

• Loop

• Accept or reject immediately

**Formal Definition**

A **Turing machine** is a 7-tuple \((Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}})\), where

1. \(Q\) is the set of states,
2. \(\Sigma\) is the input alphabet, where the blank symbol \(\_ \notin \Sigma\),
3. \(\Gamma\) is the tape alphabet, where \(\_ \in \Gamma\) and \(\Sigma \subseteq \Gamma\),
4. \(\delta : Q \times \Gamma \longrightarrow (Q \times \Gamma \times \{L, R\})\) is the transition function,
5. \(q_0 \in Q\) is the start state,
6. \(q_{\text{accept}} \in Q\) is the accept state, and
7. \(q_{\text{reject}} \in Q\) is the reject state.

**Example of transition function:**

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

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

**An Example**

\[B = \{w \# w \mid w \in \{0, 1\}^*\}, \text{ and } B = L(M_1)\]

• The tape changing:

<table>
<thead>
<tr>
<th>01100000110000...</th>
<th>01100000110000...</th>
</tr>
</thead>
<tbody>
<tr>
<td>x110000111000...</td>
<td>x110000111000...</td>
</tr>
<tr>
<td>x110000111000...</td>
<td>x110000111000...</td>
</tr>
<tr>
<td>x110000111000...</td>
<td>x110000111000...</td>
</tr>
<tr>
<td>x110000111000...</td>
<td>x110000111000...</td>
</tr>
<tr>
<td>x110000111000...</td>
<td>x110000111000...</td>
</tr>
<tr>
<td>x110000111000...</td>
<td>x110000111000...</td>
</tr>
</tbody>
</table>

**Configuration**

• A configuration of TM:
  – Current state
  – Symbols on tape
  – Head of location

• A formal specification of a configuration:
  – where
  – \(u, v\) are strings on \(\Gamma\), and \(w\) is the current content on tape
  – \(q\) is current state
  – head is in the first symbol of \(v\).
  – ex: 1011 \(q_1 \ 01111\)
Configuration

- For two configurations:
  \( uaq_{ibv} \) and \( uq_{jvc} \), where
  \( a, b, c \in \Gamma \), and \( a, c \in \Gamma^* \)
  \( uaq_{ibv} \) yields \( uq_{jvc} \) if \( \delta(q_i, b) = (q_j, c, L) \)
  \( uaq_{ibv} \) yields \( uaq_{jvc} \) if \( \delta(q_i, b) = (q_j, c, R) \)

- Two special cases:
  - the leftmost cell
    - \( q_{ibv} \) yields \( q_{jvc} \) if \( \delta(q_i, b) = (q_j, c, L) \)
    - \( q_{ibv} \) yields \( cq_{jvc} \) if \( \delta(q_i, b) = (q_j, c, R) \)
  - on the cell with blank symbol
    - \( uaq_{i} \) is equivalent to \( uaq_{i,j} \)

Languages

- Turing-recognizable Languages:
  - For a \( L \subset \Gamma^* \), exists a \( M \) such that \( M \) recognizes \( L \)
  - “Recognize” means accept, reject, or loop
- Turing-decidable languages:
  - For a \( L \subset \Gamma^* \), exists a \( M \) such that \( M \) decides \( L \)
  - “Decide” means halting: either accept or reject
- Turing-decidable \( \subseteq \) Turing-recognizable
  - Halting Problem is Turing-recognizable, but not decidable.
- Not all languages are Turing-recognizable
  - There are some languages cannot be recognized by a TM.
    - Complement of Halting problem is Turing-unrecognizable

An example

\[ A = L(M_2), \text{ where } A = \{0^n \mid n \geq 0 \} \]

- Semantical description:
  For an input string \( w \):
  1. \( \text{sweep left to the right along the tape, crossing off every other 0} \)
     if tape contains single 0
     (return accepted)
  2. \( \text{if tape contains odd number and more than one of 0s} \)
     (return rejected)
  3. \( \text{else go back to leftmost cell} \)

- Formal description:
  \( M_2 = (Q, \Sigma, \Gamma, q_0, \phi, \phi_{accept}, \phi_{reject}), \) where
  \( Q = \{q_0, q_1, q_2, q_3, q_4, q_5, q_6\} \)
  \( \Sigma = \{0\} \)
  \( \Gamma = \{0, 1, x\} \)
  \( \delta \) - state transition diagram
Outline

– Language Hierarchy
– Definition of Turing Machine
– TM Variants and Equivalence
– Decidability
– Reducibility

Simple variant

• $\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R, S\}$
• $\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R, RR, LL\}$
• They are equivalent in recognizing language:
  – They can be simulated by original the TM
  – The difference is not significant

TM Variants

• Multitape TM
• Nondeterministic TM
• Enumerators
• Equivalence: All have same power
  – Recognize the same class of languages
  – Can be simulated by an ordinary TM

Multitape TM

• A multitape TM is identical to ordinary TM except:
  – $k$ tapes, where $k \geq 1$
  – Each tap has its own head
  – $\delta : Q \times \Gamma^k \rightarrow Q \times \Gamma^k \times \{L, R, S\}^k$
  – $\delta(q_i, \alpha_1, \alpha_2, \ldots, \alpha_k) = (q_j, b_1, b_2, \ldots, b_k, L, R, \ldots, R)$
Multitape TM

- Theorem: each multitape TM has an equivalent single tape TM
  - Put # in a single tape for demarcation of original $k$ tapes.
  - Each movement of $M$ is simulated by a series movement of $S$ on each segment.
  - For a right-move on the rightmost cell of $i$th tape in $M$, $S$ write blank symbol in $(i+1)$th #, and right-shifts all symbols after that one cell.

| $M$ | 0|1|0|1|0|u| ... |
|-----|---|---|---|---|---|---|
|     | b|a|b|c| ... |
| $S$ | 1|0|1|0|0|a|a|a|a|b|b|b|u| ... |

Nondeterministic TM

- A nondeterministic TM is identical to an ordinary TM except:
  - $\delta : Q \times \Gamma \to \mathcal{P}(Q \times \Gamma \times \{L,R\})$
  - At any point the head has several possibilities to read/write/move.
- In deterministic TM, a computation is a single path with sequence of configurations.
- In nondeterministic TM, a computation is a tree or a directed acyclic graph.
  - A NTM accepts an input string if there exists a path leading to an accept state.
  - If all paths lead to reject state, then this input is rejected.

NTM

- A computation single path and multi-path in a tree:

<table>
<thead>
<tr>
<th>Deterministic computation</th>
<th>Non-deterministic computation</th>
</tr>
</thead>
<tbody>
<tr>
<td>start</td>
<td>reject</td>
</tr>
<tr>
<td></td>
<td>accept or reject</td>
</tr>
<tr>
<td></td>
<td>accept</td>
</tr>
</tbody>
</table>

Nondeterminism

- Is nondeterministic model always equivalent to a deterministic model?
  - Yes, for FA
  - No, for PDA
  - Some CFL cannot be recognized by any DPDA.
  - Yes, for TM!
**NTM**

- Theorem: *Every NTM has an equivalent DTM.*

- For a computing tree of a NTM $N$ with an input $w$, simulated with a 3-tape DTM $M$:
  - 1st tape: input $w$
  - 2nd tape: tape of a computing path with $N$
  - 3rd tape: node address (finite)

---

**Enumerator**

- Theorem: *A language is Turing-recognizable iff some enumerator enumerates it.*
  - For a language, if $E$ enumerates it, then construct a TM $M$ works as:
    - Run $E$. Every time that $E$ outputs a string, compare it with input $w$.
    - If $w$ appears in the output of $E$, accept.
  - For a language recognized by a TM $M$, construct $E$ such that:
    - Run $M$ for $i$ steps on each input, $s_1, s_2, \ldots, s_i$.
    - If any computations accept, print out the corresponding $i$.
    - Repeat the above two steps with all possible inputs.
  - An enumerator can be regarded as a 2-tape TM.
    - Write accepted list on the 2nd tape.

---

**Other Variants**

- Write-twice TM
  - Each cell on tape can only be written twice
- Write-once TM
  - Each cell on tape can only be written once
- TM with doubly infinite tape
  - Two-way infinite tape
- Universal TM
  - A TM that takes input of description of another TM.
Thesis

• Church-Turing Thesis:
  – *Any algorithm can be expressed as a TM*
  – Formally defines an algorithm:

<table>
<thead>
<tr>
<th>Instance</th>
<th>equals</th>
<th>Turing machine</th>
</tr>
</thead>
<tbody>
<tr>
<td>algorithm</td>
<td></td>
<td>algorithm</td>
</tr>
</tbody>
</table>

• Extended Church-Turing Thesis:
  – *Any polynomial-time algorithm can be expressed as a TM that operates in polynomial time.*
  – A polynomial-time algorithm: number of element operations is a polynomial function of input length.
  – A polynomial-time TM: number of state transition is a polynomial function of input length.

Describing TM

• Formal description
  – specifying Turing machine’s states, transition function, and so on.

• Implementation description
  – using natural language to describe the way that the Turing machine moves its head and the way that it stores data on its tape.

• High-level description
  – using natural language describe an algorithm, ignoring the implementation model.

Outline

– Language Hierarchy
– Definition of Turing Machine
– TM Variants and Equivalence
– Decidability
– Reducibility

Solvability

• Solvable:
  – an algorithm to solve it,
  – a TM decides it.

• Unsolvable:
  – not algorithm to solve it
  – no TM can decide it.
Decidable Language

$A_{\text{DFA}} = \{(B, w) \mid B \text{ is a DFA that accepts } w\}$

- Acceptance problem:
  - Whether a particular DFA $B$ accepts a given input string $w$.
- Membership problem:
  - Another way to say: whether $\langle B, w \rangle$ is a member of $A_{\text{DFA}}$.
- Theorem: $A_{\text{DFA}}$ is a decidable language.

$M = "\text{On input } \langle B, w \rangle, \text{ where } B \text{ is a DFA and } w \text{ is a string:}\n1. \text{Simulate } B \text{ on input } w;\n2. \text{If the simulation ends in an accept state, accept; otherwise, reject.}"$

Decidable Language

$A_{\text{NFA}} = \{(B, w) \mid B \text{ is an NFA that accepts } w\}$

- Theorem: $A_{\text{NFA}}$ is a decidable language.

$N = "\text{On input } \langle B, w \rangle, \text{ where } B \text{ is an NFA and } w \text{ is a string:}\n1. \text{Convert NFA } B \text{ to an equivalent DFA } C;\n2. \text{Run TM } M \text{ for deciding } A_{\text{DFA}} \text{ (as a "procedure") on input } \langle C, w \rangle;\n3. \text{If } M \text{ accepts, accept; otherwise, reject.}"$

Decidable Language

$A_{\text{REX}} = \{(R, w) \mid R \text{ is a regular expression that generates } w\}$

- Theorem: $A_{\text{REX}}$ is a decidable language.

$P = "\text{On input } \langle R, w \rangle, \text{ where } R \text{ is a regular expression and } w \text{ is a string:}\n1. \text{Convert regular expression } R \text{ to an equivalent DFA } A;\n2. \text{Run TM } M \text{ for deciding } A_{\text{DFA}} \text{ on input } \langle A, w \rangle;\n3. \text{If } M \text{ accepts, accept; otherwise, reject.}"$

Decidable Language

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

- Emptiness test problem:
  - Whether the language of a particular DFA is empty.
- Theorem: $E_{\text{DFA}}$ is a decidable language.

$T = "\text{On input } \langle A \rangle, \text{ where } A \text{ is a DFA:}\n1. \text{Mark the start state of } A;\n2. \text{Repeat Step 3 until no new states get marked.}\n3. \text{Mark any state that has a transition coming into it from any state that is already marked.}\n4. \text{If no accept state is marked, accept; otherwise, reject.}"$
Decidable Language

\[ EQ_{DFA} = \{ (A, B) | A \text{ and } B \text{ are DFAs and } L(A) = L(B) \} \]

- Equivalence problem:
  - Test whether two DFAs recognize the same language.
- Theorem: \( EQ_{DFA} \) is a decidable language.

Other Problems

- \( A_{CFG} \) is decidable.
- \( E_{CFG} \) is decidable.
- \( EQ_{CFG} \) is undecidable.
  - CFG is not closed in intersection and complementation.
- \( A_{TM} \) is undecidable.
  - Halting problem
- \( E_{TM} \) is undecidable.
- \( EQ_{TM} \) is undecidable.

Halting Problem

\[ A_{TM} = \{ (M, w) | M \text{ is a TM and } M \text{ accepts } w \} \]

Theorem: \( A_{TM} \) is Turing-recognizable.

\[ U = \text{"On input } (M, w), \text{ where } M \text{ is a TM and } w \text{ is a string:} \]
1. Simulate \( M \) on input \( w \).
2. If \( M \) ever enters its accept state, accept; if \( M \) ever enters its reject state, reject.
   - \( U \) is an example of universal TM.
   - \( U \) keeps looping if \( M \) neither accepts or rejects.

Halting Problem

- Theorem: \( A_{TM} \) is undecidable.
  - Can be proved by recursive theorem.

\[ D((M)) = \begin{cases} 
  \text{accept} & \text{if } M \text{ does not accept } (M) \\
  \text{reject} & \text{if } M \text{ accepts } (M) 
\end{cases} \]
**Unrecognizable**

- **Theorem:** There are languages that cannot be recognized by any TM.
  - The set of TMs are countable
    - $Q$, $\Sigma$, and $\Gamma$ are all finite sets
    - Number of transition functions is countable.
  - The set of languages is uncountable.
    - $w \in \Gamma^*$
    - $L \subseteq \Gamma^*$
    - $L \in \mathcal{P}(\Gamma^*)$. $\mathcal{P}(\Gamma^*)$ is uncountable
      - Diagonalization method to prove this

---

**Countable**

- Set of position rational numbers is countable: $\{m/n, m, n \in \mathcal{N}\}$

---

**Countable and Uncountable**

- Two infinite sets $A$ and $B$ are the same size if there is a correspondence from $A$ to $B$.
  - A correspondence is a one-to-one and onto function:
    - $f : A \rightarrow B$
    - one-to-one: $f(a) \neq f(b)$ whenever $a \neq b$
    - Onto: $\forall b \in B, \exists a \in A, f(a) = b$

- A set is countable if either it is finite or it has the same size as $\mathcal{N} = \{1,2,3…\}$; otherwise it is uncountable.

---

**Uncountable**

- Set of real numbers $\mathcal{R}$ is uncountable:

  Assume that a correspondence $f$ existed between $\mathcal{N}$ and $\mathcal{R}$.

<table>
<thead>
<tr>
<th>$n$</th>
<th>$f(n)$</th>
</tr>
</thead>
</table>
  | 1   | 3.14159...
  | 2   | 55.55555...
  | 3   | 0.12345...
  | 4   | 0.50000...

  We can find an $x$, $0 < x < 1$, so that the $i$-th digit following the decimal point of $x$ is different from that of $f(i)$: for example, $x = 0.4641\ldots$ is a possible choice.
Uncountable

- The set of all languages over an alphabet is uncountable.
  - Think that a real number is a string over alphabet of \{ . , 0,1,2,3,4,5,6,7,8,9\}
  - Similar diagonalization way to prove with general alphabet

\[ \overline{A_{TM}} = \{ \langle M, w \rangle | M \text{ is a TM and } M \text{ does not accept } w \} \]

- Theorem: \( \overline{A_{TM}} \text{ is not Turing-recognizable} \)
  - If \( \overline{A_{TM}} \) is Turing-recognizable, and \( A_{TM} \) is Turing-recognizable, then \( A_{TM} \) must be decidable. — contradiction!

---

Theorem: A language is decidable iff both it and its complement language are Turing-recognizable.

- If \( A \) is decided by \( M_j \), then :
  - \( M_j \) on input \( w \):
    1. Run \( M_j \) on \( w \).
    2. If \( M_j \) rejects, accept; if \( M_j \) accepts, reject.
  - \( M_j \) decides \( A \)

- If \( A \) and \( \overline{A} \) are Turing-recognizable:
  Let \( M_2 \) be a recognizer for \( A \) and \( M_2 \) be a recognizer for \( \overline{A} \).
  \( M = \) “On input \( w \):
  1. Run both \( M_1 \) and \( M_2 \) on input \( w \) in parallel. (\( M \) takes turns simulating one step of each machine until one of them halts.)
  2. If \( M_1 \) accepts and \( M_2 \) accepts, reject.”

Outline

- Language Hierarchy
- Definition of Turing Machine
- TM Variants and Equivalence
- Decidability
- Reducibility
Reducibility

- Semantics
- Reduce $A_{TM}$ to $HALT_{TM}$
- PCP Problem
- Mapping Reducibility