Chapters 11 and 12

Decision Problems and Undecidability
11.1 Decision Problems

- A decision problem
  - consists of a set of questions whose answers are either yes or no
  - is undecidable if no algorithm that can solve the problem; otherwise, it is decidable

- The Church-Turing thesis asserts that
  - a decision problem $P$ has a solution if, and only if, there exists a TM that determines the answer for every $p \in P$
  - if no such TM exists, the problem is said to be undecidable

- An unsolvable problem
  - is a problem such that there does not exist any TM that can solve the problem
Decision Problems

- Algorithm \( L \) that solves a decision problem should be **effective**, i.e.,
  - **Complete**: \( L \) produces the correct **answer** (yes/or) to each question (of the problem)
  - **Mechanistic**: \( L \) consists of a **finite** sequence of instructions
  - **Deterministic**: \( L \) produces the same **result** for the same input

- **The Church-Turing Thesis for Computable Functions:**
  - A function \( f \) is effective, i.e., effectively computable, if and only if there is a TM that computes \( f \).
11.2 Recursive Languages

- **Defn.** A recursive language $L$ is a formal language for which there exists a TM that will *halt* and *accept* an input string in $L$, and *halt* and *reject*, otherwise.

- **Example 11.2.1** The decision problem of determining whether a natural number is a *perfect square* (represented by using the string $a^n$) is decidable.

- **Example 11.2.2** The decision problem of determining whether there is a *path* $P$ from node $v_i$ to a node $v_j$ in a directed graph $G$ (with nodes $v_1, \ldots, v_n$) using a NTM $M$ with 2-tape is decidable. $G$ is represented over $\{0, 1\}$ as:
  - Encode $v_k (1 \leq k \leq n)$ as $1^{k+1}$, and arc $(v_s, v_t)$ as $1^{s+1} 0 1^{t+1}$
  - Separate each arc by 00; three 0’s separate $G$ and $v_i$ and $v_j$
  - Write $v_i$ (as $v_s$) on tape 2 and consider each arc $(v_s, v_t)$ in $G$
  - $M$ *accepts*, if $v_t = v_j$, or *rejects* if $v_t$ has been visited/no edge
12.1 The Halting Problem for TMs

- The halting problem
  Given an arbitrary TM $M$ with input alphabet $\Sigma$ and a string $w \in \Sigma^*$, will the computation of $M$ with $w$ halt?

- There is no algorithm that solves the halting problem, i.e., the halting problem is undecidable.

- A solution to the halting problem requires a general algorithm that answers the halting question for each combination of TM and input string.

  - Proposed solution: encode the TM $M$ and the string $w$ as an input over the alphabet $\{0, 1\}$ and tape alphabet $\{0,1,B\}$ with $\{q_0, q_1, \ldots, q_n\}$ being the states of $M$ and $q_0$ is the start state.
Consider the following encoding scheme (as shown in Section 11.5):

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Encoding</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>11</td>
</tr>
<tr>
<td>B</td>
<td>111</td>
</tr>
<tr>
<td>$q_0$</td>
<td>1</td>
</tr>
<tr>
<td>$q_1$</td>
<td>11</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>$q_n$</td>
<td>$1^{n+1}$</td>
</tr>
<tr>
<td>L</td>
<td>1</td>
</tr>
<tr>
<td>R</td>
<td>11</td>
</tr>
</tbody>
</table>

Let $en(Z)$ denotes the encoding of a symbol $Z$. A transition $\delta(q_i, X) = [q_j, Y, d]$ is encoded by the string

$$en(q_i) \ 0 \ en(X) \ 0 \ en(q_j) \ 0 \ en(Y) \ 0 \ en(d)$$

where 0’s separate the components of the transition, 2 0s separate transitions, and 3 0s designate the beginning & end of the encoding.
12.1 The Halting Problem for TMs

- We can construct a TM to determine whether an arbitrary string \( u \in \{0, 1\}^* \) is the encoding of a DTM \( M \).

- **Theorem 12.1.1.** The halting problem of TMs is *undecidable*.

  **Proof.** The proof is by *contradiction*. Assume that there is a TM \( H \) that solves the *halting* problem. A string \( z \) is accepted by \( H \) if

  (i) \( z \) consists of the representation of a TM \( M \) following by a string \( W \), and

  (ii) the computation of \( M \) with input \( W \) halts.

  If either of these conditions is not satisfied, then \( H \) rejects the input. …
11.2 Recursive vs. Recursively Enumerable Languages

- **Defn.** A recursively enumerable language $L$ is a formal language for which there exists a TM that will *halt* and *accept* an input string in $L$, and may either (i) *halt* and *reject*, or (ii) *loop forever*, otherwise.

- **Defn.** A recursive language $L$ is a formal language for which there exists a TM that will *halt* and *accept* an input string in $L$, and *halt* and *reject*, otherwise.

- **Corollary 12.1.3** The recursive languages are a *proper subset* of recursively enumerable languages.

**Proof.** Let a language $L$ be $L_H = \{ R(M)w \mid R(M) \text{ is the representation of a TM } M \text{ and } M \text{ halts with input } w \}$ over $\{0, 1\}^*$ is recursively enumerable according to Theorem 11.5.1 (i.e., $L_H$ is recursively enumerable). $L_H$ is not recursive according to Corollary 12.1.2, which states that the language $L_H$ is *not* recursive.
11.2 Recursive vs. Recursively Enumerable Languages

Given that $L$ and $P$ are two recursively enumerable languages, then the following languages are recursively enumerable:

- The union, $L \cup P$
- The intersection, $L \cap P$
- The concatenation $LP$ of $L$ and $P$
- The Kleene star $L^*$ of $L$

Recursively enumerable languages are not closed under set difference or complementation, i.e., given two recursively enumerable languages $L$ and $P$

- If $\bar{L}$ is also recursively enumerable, then $L$ is recursive
- $L – P$ may or may not be recursively enumerable, since

$$L – P = L \cap \overline{L} \cap P$$
11.2 Recursive vs. Recursively Enumerable Languages

- The union of two recursively enumerable languages is recursively enumerable

Proof. Let $L_1$ and $L_2$ be two recursively enumerable languages accepted by TMs $M_1$ and $M_2$, respectively. We show that $L_1 \cup L_2$ is accepted by a 2-tape TM $M$.

Let $x = w_1 \lor w_2$. To determine if $M_1$ or $M_2$ accepts $x$, i.e., $w_1 \in L_1$ or $w_2 \in L_2$, run both $M_1$ & $M_2$ on $x$ simultaneously using the 2-tape TM $M$.

$M$ simulates $M_1$ on the first tape & $M_2$ on the second tape. If either one of the TM enters the final state and halts, then the input $x$ is accepted by $M$, i.e.,
11.2 Recursive vs. Recursively Enumerable Languages

- If the languages \( L \) and \( \overline{L} \) are recursively enumerable, then \( L \) is recursive.

**Proof.** Let \( M_1 \) & \( M_2 \) be two TMs, such that \( L = L(M_1) \) and \( \overline{L} = L(M_2) \). Construct a 2-tape TM that simulates \( M_1 \) & \( M_2 \) in parallel, with \( M_1 \) on tape 1 & \( M_2 \) on tape 2.

If an input \( x \) to \( M \) is in \( L \), then \( M_1 \) halts & accepts \( x \), and hence \( M \) accepts \( x \) and halts.

If input \( x \) to \( M \) is not in \( L \), hence it is in \( \overline{L} \), then \( M_2 \) accepts and halts for \( x \) and \( M \) halts w/o accepting. Hence, \( M \) halts with every input and \( L = L(M) \), and \( L \) is recursive.

![Diagram](image.png)
11.2 Recursive vs. Recursively Enumerable Languages

- Given that $L$ and $P$ are two recursive languages, then the following languages are recursive as well:
  - The union, $L \cup P$
  - The intersection, $L \cap P$
  - The difference, $L - P$
  - The complement of $L$, $\overline{L}$
  - The concatenation $LP$ of $L$ and $P$
  - The Kleene star $L^*$ of $L$

- Recursively languages $L$ and $P$ are closed under set difference, since

$$L - P = L \cap \overline{L} \cap P$$
11.2 Recursive vs. Recursively Enumerable Languages

- Given that \( L \) is a recursive language and \( P \) is a recursively enumerable language, then the following languages are recursively enumerable as well:
  - The union, \( L \cup P \)
  - The intersection, \( L \cap P \)
  - The concatenation \( LP \) of \( L \) and \( P \)
  - The difference, \( P - L \) (but not \( L - P \))
11.2 Recursive vs. Recursively Enumerable Languages

- The **intersection** of a recursive language \(L_1\) & a recursively enumerable language \(L_2\) is recursively enumerable.

**Proof.** Let \(L_1\) and \(L_2\) be the languages accepted by TMs \(M_1\) and \(M_2\), respectively. We show that \(L_1 \cap L_2\) is accepted by a TM \(M\) which halts and accepts an input string \(x\) if \(x \in L_1 \cap L_2\).

\(M\) simply simulates \(M_1\) & \(M_2\) one after the other on the same input \(x\). If \(M_1\) halts and accepts \(x\), \(M\) clears the tape, copies \(x\) on the tape & starts simulating \(M_2\). If \(M_2\) also halts & accepts \(x\), then \(M\) accepts \(x\).

Clearly, \(M\) accepts \(L_1 \cap L_2\), and if \(M_1\) & \(M_2\) halts on all inputs, then \(M\) also halts on all inputs.
11.4 The Church-Turing Thesis

- The **Church-Turing thesis** asserts that every solvable decision problem can be transformed into an equivalent Turing machine problem.

- The **Church-Turing thesis for decision problems:**
  
  There is an *effective* procedure to solve a decision problem if, and only if, there is a TM that *halts* for all input strings and solves the problem.

  - A solution to a decision problem is equivalent to the question of membership in a *recursive language*.

- The **Church-Turing thesis for Recognition Problems:**

  A decision problem $P$ is *partially* solvable if, and only if, there is a TM that *accepts* precisely the instances of $P$ whose answer is “yes”.

  - A partial solution to a decision problem is equivalent to the question of membership in a *recursively enumerable language*.
11.5 A Universal Machine

- Universal Turning machine ($U$)
  - designed to simulate the computations of any TM $M$
  - Accepts the input $R(M)w$, whenever $M$ (accepts by halting) halts with $w$
  - Loop whenever $M$ does not halt with $w$
  - $U$ accepts the set of strings in $L(U)$, which consists of all strings $R(M)w$ for which $M$ halts with input $w$
  - $U$ represents the entire family of TMs, since the outcome of the computation of any TM $M$ with input $w$ can be obtained by the computation of $U$ with input $R(M)w$
11.5 A Universal Machine

Theorem 11.5.1 The language $L_H = \{ R(M)w \mid M \text{ halts with input } w \}$ is recursively enumerable.

Proof. Use a deterministic 3-tape TM $U$ to accept $L_H$ by halting, where

1) an input string $S$ is placed on tape 1 and $U$ moves indefinitely to the right if $S$ does not have the form $R(M)w$

2) the computation of $M$ with $w$, which is copied to tape 3, is simulated on tape 3

3) the current state, i.e., start state $q_0$, is encoded as ‘1’ on tape 2

4) let $x$ be the symbol on tape 3 and $q_i$ the state encoded on tape 2:
   a) Scan tape 1 for $en(q_i)$ and $en(x)$. If such transition does not exist, $U$ halts and accepts $S$. (* Recall that $U$ accepts by halting *)
   b) Otherwise, $en(q_i) \ 0 \ en(x) \ 0 \ en(q_j) \ 0 \ en(y) \ 0 \ en(d)$ exists. Then
      i. Replace $en(q_i)$ by $en(q_j)$ on tape 2.
      ii. Write $y$ to tape 3.
      iii. Move the tape head of tape 3 in the direction of $d$.

5) Repeat Step 4 to simulate the next transition of $M$. 