**A_{TM} is Turing Undecidable**

In our study of Turing decidability, we consider Turing machines with states \( q_{\text{accept}} \) and \( q_{\text{reject}} \). We start with the following definition: A language \( L \) is **Turing decidable** if and only if there exists a Turing machine \( M_L \) such that:

- \( M_L \) halts on every input string \( w \) with either “accept” or “reject”.
- \( M_L \) accepts \( w \) if and only if \( w \in L \).

When these two conditions hold, we say \( M_L \) **decides** \( L \). However, there are languages for which no Turing machine exists to decide them. Such languages are called **Turing undecidable**.

Another important concept is **encoding**. We use the notation \( \langle M \rangle \) to denote the encoding of a Turing machine as a string. We now define (and prove) a Turing undecidable language, \( A_{TM} \)

\[
A_{TM} = \{ \langle M, w \rangle \mid M \text{ is a Turing machine and } M \text{ accepts } w \}
\]

**Theorem:** \( A_{TM} \) is Turing undecidable.

**Proof:** (by contradiction) Let \( H \) be a Turing machine which decides \( A_{TM} \). We now construct Turing machine \( D \) that uses \( H \) as a subroutine.

\[
D = \text{“On input } \langle M \rangle \text{”} \\
1. \text{Construct the string } \langle M, \langle M \rangle \rangle \\
2. \text{Run } H \text{ on input } \langle M, \langle M \rangle \rangle \\
   a. \text{If } H \text{ accepts, then reject} \\
   b. \text{If } H \text{ rejects, then accept } ^{\dagger}
\]

If \( M \) accepts its own description\(^{\dagger} \) \( \langle M \rangle \) then \( D \) rejects \( \langle M \rangle \). Conversely, if \( M \) does not accept its own description, then \( D \) accepts \( \langle M \rangle \).

We now consider the case when \( D \) is run on its own description \( \langle D \rangle \). By definition of \( D \) we have:

- If \( H \) decides that \( D \) accepts \( \langle D \rangle \) (on line 2), then \( D \) rejects \( \langle D \rangle \) (on line 2a).
- If \( H \) decides that \( D \) rejects \( \langle D \rangle \) (on line 2), then \( D \) accepts \( \langle D \rangle \) (on line 2b).

These two statements above are a contradiction and the theorem is proven.

The language \( A_{TM} \) is known as the “Acceptance Problem”.

\(^{\dagger}\)The C compiler “gcc” is written in C and it is capable of compiling its own source code. The concept of a program that accepts its own description is a little unusual, but certainly possible.
When a Turing machine $M$ runs on input $w$ there are three possible outcomes:

- $M$ halts with “accept”
- $M$ halts with “reject”
- $M$ runs forever

The possibility that a Turing machine (i.e., a computer program) never stops has important implications to computer science theory. Given a machine $M$ and input $w$, the problem of deciding whether $M$ eventually halts on input $w$ is known as the **Halting Problem**. The Halting Problem is also Turing undecidable.

**Turing Recognizability** A language $L$ is Turing recognizable if and only if there exists a Turing machine $M$ such that if $M$ is run on an input string $w \in L$, $M$ will accept $w$ in finitely many steps. However, given a string $x \notin L$, $M$ may reject $x$, but it may also run forever.

**Theorem:** $A_{TM}$ is Turing recognizable.

**Proof:** The following Turing machine is a recognizer for $A_{TM}$.

$$R = \text{"On input } \langle M, w \rangle \text{ 
1. Run (simulate) } M \text{ on input } w \text{ 
   a. If } M \text{ accept, then accept } 
   b. \text{ If } M \text{ rejects, then reject "}$$

If $M$ accepts $w$, then $\langle M, w \rangle \in A_{TM}$ and $R$ will accept on line 1a. Therefore, $R$ recognizes $A_{TM}$

**Theorem:** If a language $L$ and its complement $\overline{L}$ are both recognizable, then they are both decidable.

**Proof:** Let $R_L$ and $R_{\overline{L}}$ be recognizers for $L$ and $\overline{L}$ respectively. We construct the following Turing machine:

$$S = \text{"On input } \langle w \rangle \text{ 
1. For } k = 1, 2, 3, \ldots \text{ do } 
   a. \text{ Run } R_L \text{ for } k \text{ steps. If } R_L \text{ accepts, then accept. }
   b. \text{ Run } R_{\overline{L}} \text{ for } k \text{ steps. If } R_{\overline{L}} \text{ accepts, then reject."
}$$

Observe that $S$ decides $L$. The input string $w$ must belong to either $L$ or $\overline{L}$, but not both. Eventually either $R_L$ or $R_{\overline{L}}$ must recognize (accept) $w$. At that point we know whether $w \in L$ or $w \in \overline{L}$.

The Turing machine $S$ decides $L$ but a similar machine can be used to decide $\overline{L}$. Therefore, both $L$ and $\overline{L}$ are decidable.

**Corollary:** $A_{TM}$ is not Turing recognizable.