1 Halting Problem

Can we tell if a given TM is a decider? Or, even simpler, can we tell if a given TM halts on a given input string? No, the latter is the famous Halting Problem that was shown undecidable by Alan Turing in 1936.

Consider the language

\[ A_{TM} = \{ \langle M, w \rangle \mid \text{TM } M \text{ accepts input } w \} \].

Theorem 1. \( A_{TM} \) is semi-decidable.

Proof. Recall our universal TM \( U \): “On input \( \langle M, w \rangle \), simulate TM \( M \) on input \( w \), accepting if \( M \) accepts \( w \).”

We argued that \( U \) exists. Obviously, \( U \) accepts \( A_{TM} \).

Theorem 2. \( A_{TM} \) is undecidable.

Proof. We’ll prove it in two stages.

- **Stage 1**: construct a language \( D \) such that \( D \) is not semi-decidable.

- **Stage 2**: show that if \( A_{TM} \) were decidable, then \( D \) would be decidable. A contradiction.

Now we provide more details.

**Stage 1**: Define \( D \) using the diagonalization method. Define

\[ D = \{ \langle M_i \rangle \mid \text{TM } M_i \text{ does not accept input } \langle M_i \rangle \} \],

where \( M_i \) is the \( i \)th TM in the complete enumeration of all TMs.

Claim 1. \( D \) is not semi-decidable.

Proof of Claim: Suppose \( D \) is recognizable by some TM \( M_n \). Then consider the question "Is \( \langle M_n \rangle \) in \( L(M_n) \)?"

If the answer is Yes, then we get:

\( \langle M_n \rangle \in L(M_n) \Rightarrow (\text{by the definition of } D) \)

\( \langle M_n \rangle \notin D \Rightarrow (\text{since } D = L(M_n)) \)

\( \langle M_n \rangle \notin L(M_n) \). A contradiction.

If the answer is No, then we get:
\[ \langle M_n \rangle \not\in L(M_n) \Rightarrow \text{(by the definition of } D) \]
\[ \langle M_n \rangle \in D \Rightarrow \text{(since } D = L(M_n)) \]
\[ \langle M_n \rangle \in L(M_n). \text{ A contradiction.} \]

Thus, in both cases we derive a contradiction. Hence, our assumption that \( D \) is semi-decidable must be false.

\[ \square \]

Stage 2: Suppose \( A_{TM} \) is decidable by a TM \( H \). Then we obtain the decider TM for the language \( D \) as follows:

“On input \( \langle M_i \rangle \), simulate \( H \) on input \( \langle M_i, \langle M_i \rangle \rangle \). If \( H \) accepts its input, then Reject.
Otherwise, Accept.”

Since we just proved in Stage 1 that \( D \) is not even semi-decidable (let alone decidable), it must be the case that our assumption that \( A_{TM} \) is decidable is false. So, \( A_{TM} \) is undecidable.

Thus, we know that there are problems that are semi-decidable but not decidable (e.g., \( A_{TM} \) is such a problem). There are also problems that are not even semi-decidable (e.g., language \( D \) from the proof above).

2 Semi-decidable vs. decidable

We know that a language may be semi-decidable but not decidable. However, if both \( L \) and its complement \( \bar{L} \) are semi-decidable then \( L \) must in fact be decidable.

Claim 2. If a language \( L \) and its complement \( \bar{L} \) are both semi-decidable, then \( L \) is decidable.

Proof. Let \( M_L \) be a TM accepting \( L \), and let \( M_{\bar{L}} \) be a TM accepting \( \bar{L} \). On input \( x \), run both TMs “in parallel”, until one of them accepts. (At some finite point in time, one of the machines must accept as every input \( x \) is either in \( L \) or in \( \bar{L} \).) If \( M_L \) accepted, then halt and accept. If \( M_{\bar{L}} \) accepted, then halt and reject.

As a corollary, we get that the complement of \( A_{TM} \) is not semi-decidable! Do you see why?

We also have the following.

Theorem 3. The class of decidable languages is closed under complementation. On the other hand, the class of semi-decidable languages is not closed under complementation.

Proof. Given a DTM \( M \) deciding a language \( L = L(M) \), construct a new DTM \( M' \) by taking \( M \) and swapping \( q_{accept} \) and \( q_{reject} \) states. It’s easy to see that the new DTM \( M' \) accepts exactly those strings that are rejected by \( M \), and rejects exactly those strings that are accepted by \( M \). So, we have \( L(M') = \bar{L} \), as required.

On the other hand, \( A_{TM} \) is semi-decidable, but, as observed above, its complement is not semi-decidable.

3 Types of proofs of undecidability

There are two types of proof for undecidability:

1. diagonalization (e.g., language \( D \))
2. reduction (e.g., \( A_{TM} \))

Most of our proofs will be proofs by reduction. We give some examples next.
4 Examples of undecidable languages

**Theorem 4.** The language 

\[ \text{ETM} = \{ \langle M \rangle \mid L(M) \text{ is empty} \} \]

is undecidable.

**Proof.** Proof by reduction from \( A_{TM} \). Given input \( \langle M, w \rangle \), design a TM \( M' \) as follows:

\( M' \): “On input \( x \), simulate \( M \) on input \( w \). If \( M \) accepts, then Accept.”

Observe that

1. if \( M \) accepts \( w \), then \( L(M') = \Sigma^* \) (i.e., \( M' \) accepts every input \( x \)),
2. if \( M \) does not accept \( w \), then \( L(M') = \emptyset \).

Now, if we have a decider TM \( R \) for the language \( E_{TM} \), we can decide \( A_{TM} \) as follows:

“On input \( \langle M, w \rangle \),

1. Construct the TM \( M' \) for this pair \( \langle M, w \rangle \), as explained above.
2. Run \( R \) on input \( \langle M' \rangle \).
3. If \( R \) accepts \( \langle M' \rangle \), then Reject. If \( R \) rejects \( \langle M' \rangle \), then Accept.”

\[ \square \]

**Theorem 5.** The language 

\[ \text{ALL}_{TM} = \{ \langle M \rangle \mid L(M) = \Sigma^* \} \]

is undecidable.

**Proof.** Suppose that \( \text{ALL}_{TM} \) is decidable by \( R \). Show how to decide \( A_{TM} \).

On input \( \langle M, w \rangle \), construct TM \( M' \) as follows:

\( M' \): “On input \( x \), simulate \( M \) on \( w \), accepting if \( M \) accepts \( w \)”.

Now, if \( M \) accepts \( w \), then \( L(M') = \Sigma^* \); and if \( M \) does not accept \( w \), then \( L(M') = \emptyset \).

So to decide \( A_{TM} \), do the following:

“On input \( \langle M, w \rangle \), construct TM \( M' \) defined above. Run \( R \) on input \( \langle M' \rangle \). If \( R \) accepts \( \langle M' \rangle \), then Accept; otherwise, Reject.”

Since \( A_{TM} \) is undecidable, we conclude that \( R \) cannot exist. \[ \square \]