Undecidable languages

Define:

\[ L_{TM} = \{ \langle M, w \rangle \mid M \text{ is a TM and } M \text{ accepts the input } w \} \]

**Theorem.** \( L_{TM} \) is undecidable.

**Note.** We present a proof based on a technique called “diagonalisation”.

**Proof.** We prove by contradiction. Assume, that there exists a Turing machine \( H \), where

\[
H(\langle M, w \rangle) = \begin{cases} 
\text{accepts,} & \text{if } M \text{ accepts } w, \\
\text{rejects,} & \text{if } M \text{ does not accept } w \text{ (either rejects or loops).}
\end{cases}
\]

Now we construct a new machine \( D \), which uses \( H \) as a subroutine. On input \( \langle M \rangle \), \( D \) does the following:

1. Runs \( H \) on input \( \langle M, \langle M \rangle \rangle \).

2. Outputs the opposite of what \( H \) outputs. That is, if \( H \) accepts – \( D \) rejects; if \( H \) rejects – \( D \) accepts.
In summary,

\[ D(\langle M \rangle) = \begin{cases} 
\text{accepts,} & \text{if } M \text{ does not accept } \langle M \rangle, \\
\text{rejects,} & \text{if } M \text{ accepts } \langle M \rangle. 
\end{cases} \]

Question: what happens when we run \( D \) with its own encoding \( \langle D \rangle \) as an input? In this case

\[ D(\langle D \rangle) = \begin{cases} 
\text{accepts,} & \text{if } D \text{ does not accept } \langle D \rangle, \\
\text{rejects,} & \text{if } D \text{ accepts } \langle D \rangle. 
\end{cases} \]

No matter what \( D \) is supposed to do, it does the opposite. Contradiction. Therefore such \( H \) does not exist.

We will show that there exist languages, which are not even Turing-recognisable.

**Definition.** A language \( L \) is called co-Turing recognisable if it is the complement of a Turing-recognisable language.

**Theorem.** A language \( L \) is decidable if and only if it is Turing-recognisable and co-Turing recognisable.

**Proof.** (1) If \( L \) is decidable, the it is clearly also recognisable. Moreover, its complement is also Turing-recognisable (construct Turing-machine \( M \) that simulates the machine \( M_L \) that decides \( L \), \( M \) rejects if and only if \( M_L \) accepts).

(2) Assume that \( L \) and \( \bar{L} \) (complement of \( L \)) are Turing recognisable. Let \( M_L \) be a machine that recognises \( L \) and \( M_{\bar{L}} \) be a machine that recognises \( \bar{L} \). The following machine \( M \) decides \( L \) then.

Machine \( M \):

1. Runs both \( M_L \) and \( M_{\bar{L}} \) on input \( w \) in parallel.

2. If \( M_L \) accepts – accepts, if \( M_{\bar{L}} \) accepts – rejects.

(Running in parallel means that \( M \) simulates one step of \( M_L \) after one step of \( M_{\bar{L}} \), etc.)

Now we show that \( M \) indeed decides \( L \). Any string \( w \) is either in \( L \) or in \( \bar{L} \). Therefore, either \( M_L \) or \( M_{\bar{L}} \) accepts \( w \). \( M \) always halts since at least one of the machines halts. If \( w \in L \) then \( M_L \) accepts and so \( M \) accepts. If \( w \in \bar{L} \) then \( M_{\bar{L}} \) accepts and so \( M \) rejects.

**Corollary.** Language \( L_{TM} \) is not Turing-recognisable.
Proof. If $L_{TM}$ were Turing-recognisable, then (since $L_{TM}$ is Turing-recognisable) $L_{TM}$ would be Turing-decidable. Contradiction.

Define the language:

$HALT = \{⟨M, w⟩ | M \text{ is a Turing machine and } M \text{ halts on input } w\}$.

**Theorem.** $HALT$ is undecidable.

**Proof.** For the sake of contradiction, assume that $HALT$ is decidable. We will show that from this assumption it follows that $L_{TM}$ is decidable.

Assume that $M_H$ is a Turing machine that decides $HALT$. We use $M_H$ to construct $M_L$, which will decide $L_{TM}$. On input $⟨M, w⟩$ machine $M_L$ does the following:

1. Runs $M_H$ on input $⟨M, w⟩$. Since we assumed $HALT$ to be decidable, $M_H$ always halts.
2. If $M_H$ rejects $- M_L$ rejects.
3. If $M_H$ accepts $- M_L$ simulates $M$ on $w$ until it halts.
4. If $M$ accepted $w$ $- M_L$ accepts, if $M$ rejected $w$ $- M_L$ rejects.

If $M$ accepts $w$ then $M_L$ will accept $⟨M, w⟩$. If $M$ rejects $w$ or if $M$ runs infinitely long on $w$, $M_L$ will reject $⟨M, w⟩$. Therefore, $M_L$ decides $L_{TM}$. Contradiction! 

This method of proof is called “reduction from $L_{TM}$”:

$$L_{TM} \leq_M HALT$$

can decide $\iff$ can decide

$HALT$ is at least as hard as $L_{TM}$.

**Practise session**

1. Define the language

$L_{k,STR} = \{⟨A, k⟩ | A \text{ is a DFA and } L(A) \text{ consists of exactly } k \text{ strings, } k \in \mathbb{N}\}$.

Prove that $L_{k,STR}$ is decidable.
Proof. We construct a TM $M$, which decides $L_{k\text{-STR}}$. On the input $\langle A, k \rangle$, $M$ does the following.

1. Checks the number of states of $A$. Denote this number by $p$.
2. Constructs a DFA $B$, that accepts all strings of length $p$ or longer. Also constructs a DFA $C$, such that $L(C) = L(A) \cap L(B)$.
3. Generates all strings of length $\leq p - 1$ and tests whether each string is accepted by $A$. Counts the number of such strings, denote this number by $c_A$.
4. Tests whether $L(C) = \emptyset$.
5. If $L(C) = \emptyset$ and $c_A = k$ – accepts, otherwise – rejects.

Let us show that $M$ does what we want.

- First, note that due to the pumping lemma, if $A$ accepts any string of length $\geq p$, then it accepts infinitely many strings. This condition is tested by testing if $L(C) = \emptyset$.
- Provided $A$ does not accept any strings of length $\geq p$, $c_A$ is exactly the cardinality of $L(A)$. Thus $M$ accepts if and only if $|L(A)| = k$. \hfill \Box

2. Define

$$L_\emptyset = \{ \langle M \rangle \mid M \text{ is a Turing machine and } L(M) = \emptyset \}.$$ 

Show that $L_\emptyset$ is undecidable.

Solution. We show reduction from $L_{TM}$ to $L_\emptyset$ where

$$L_{TM} = \{ \langle M, w \rangle \mid M \text{ is a Turing machine and } M \text{ accepts } w \},$$

which is known to be undecidable. Reduction:

$$L_{TM} \leq_M L_\emptyset$$

decidable \iff decidable

Let $M_{\emptyset}$ be a Turing machine that decides language $L_\emptyset$. We use it to construct Turing machine $M_L$ that decides $L_{TM}$.

Given Turing machine $M$, construct Turing machine $M_w$ that rejects any input except $w$, but on input $w$ it works as before (i.e. simulates $M$ on $w$).
If $M$ accepts $w$ then $M_w$ accepts $w$. If $M$ does not accept $w$, then $M_w$ does not accept $w$:

$$L(M_w) = \begin{cases} \{w\}, & \text{if } M \text{ accepts } w \\ \emptyset, & \text{if } M \text{ does not accept } w. \end{cases}$$

Machine $M_w$ is formally defined as follows:

1. If input is not $w$, then $M_w$ rejects.
2. If input is $w$, then $M_w$ simulates $M$ on $w$ and answers accordingly.

Now, we construct Turing machine $M_L$ as follows. On the input $\langle M, w \rangle$, $M_L$ does the following:

1. Constructs a Turing machine $M_w$ as described above.
2. Runs $M_\emptyset$ on $\langle M_w \rangle$ (i.e. on description of $M_w$).
3. If $M_\emptyset$ accepts – reject, if $M_\emptyset$ rejects – accept.

Let us show that $M_L$ is correct.

- If $M$ accepts $w$ then $M_w$ accepts $w$. Therefore $L(M_w) \neq \emptyset$ and therefore in Step 2, $M_\emptyset$ rejects $\langle M_w \rangle$. Therefore, $M_L$ accepts $\langle M, w \rangle$.

- If $M$ does not accept $w$, then $M_w$ does not accept $w$. $M_w$ also does not accept any other input. Therefore, $L(M_w) = \emptyset$. Therefore, in Step 2, $M_\emptyset$ accepts $\langle M_w \rangle$. And, hence, $M_L$ rejects $\langle M, w \rangle$.

Conclusion. We constructed $M_L$, the Turing machine that decides $L_{TM}$. This is impossible. Contradiction!

Note. The machine $M_L$ should be able to construct $M_w$ from $M$. However, $M_w$ works exactly as $M$, except that in the beginning it checks that the input is exactly $w$. This can be easily done algorithmically.

3. Define

$$L_{EQ} = \{ \langle M_1, M_2 \rangle \mid M_1 \text{ and } M_2 \text{ are Turing machines and } L(M_1) = L(M_2) \}.$$ 

Show that $L_{EQ}$ is undecidable.

Solution. We perform reduction:

$$L_{\emptyset} \leq_M L_{EQ}.$$ 

For the sake of contradiction, assume that $M_{EQ}$ is a Turing machine that decides $L_{EQ}$. We construct a machine $M_\emptyset$ that decides $L_{\emptyset}$.

The machine $M_\emptyset$ does the following on input $\langle M \rangle$:
1. Runs $M_{\text{EQ}}$ on input $(M, M_1)$, where $M_1$ is the machine that rejects all inputs.

2. If $M_{\text{EQ}}$ accepts – accept, if $M_{\text{EQ}}$ rejects – reject.

Let us show that $M_\varnothing$ is correct.

- If $L(M) = \emptyset$, then $L(M) = L(M_1)$ and hence $M_{\text{EQ}}$ accepts and $M_\varnothing$ accepts.

- If $L(M) \neq \emptyset$ then $L(M) \neq L(M_1)$, thus $M_{\text{EQ}}$ rejects and $M_\varnothing$ rejects.

We constructed machine $M_\varnothing$ that decides $L_\varnothing$. Contradiction! Therefore the assumption that $L_{\text{EQ}}$ is decidable was wrong.