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. $\square$
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. \qed

Define the language:

$$\text{HALT} = \{\langle M, w \rangle \mid M \text{ is a Turing machine and } M \text{ halts on input } w\}.$$

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 $\langle M, w \rangle$ machine $M_L$ does the following:

1. Runs $M_H$ on input $\langle M, w \rangle$. 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 $\langle M, w \rangle$. If $M$ rejects $w$ or if $M$ runs infinitely long on $w$, $M_L$ will reject $\langle M, w \rangle$. Therefore, $M_L$ decides $L_{TM}$. Contradiction! \qed

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

$$L_{TM} \leq_M \text{HALT} \quad \text{can decide} \leftrightarrow \text{can decide}$$

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

Definition. Function $f : \Sigma^* \rightarrow \Sigma^*$ is a computable function if some Turing machine $M$ on every input $w$ halts with just $f(w)$ on its tape.

Definition. Language $A$ is mapping reducible to language $B$, written $A \leq_M B$, if there is a computable function $f : \Sigma^* \rightarrow \Sigma^*$, where for every $w$

$$w \in A \iff f(w) \in B.$$

The function $f$ is called the reduction from $A$ to $B$. 


**Theorem.** If $L_A \leq_M L_B$ and $L_B$ is decidable, then $L_A$ is decidable too.

**Proof.** Let $M_B$ be a Turing machine that decides $L_B$ and $f$ be a reduction from $L_A$ to $L_B$. We describe a machine $M_A$ that decides $L_A$:

1. On input $w$ compute $f(w)$;
2. Run $M_B$ on $f(w)$ and output what $M_B$ outputs.

$M_A$ decides language $L_A$ indeed:

- If $w \in A$, then $f(w) \in B$, since $f$ is reduction. $M_B$ accepts $f(w)$ – therefore $M_A$ accepts $w$.
- If $w \notin A$, then $f(w) \notin B$. $M_B$ rejects $f(w)$ and therefore $M_A$ rejects $w$.

**Practise session**

1. 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 know to be undecidable. Reduction:

\[
\begin{array}{ccc}
L_{TM} & \leq_M & L_\emptyset \\
\text{decidable} & \iff & \text{decidable}
\end{array}
\]

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.

2. 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:

\[ L_\varnothing \leq_M L_{\text{EQ}}. \]

For the sake of contradiction, assume that \( M_{\text{EQ}} \) is a Turing machine that decides \( L_{\text{EQ}} \). We construct a machine \( M_\varnothing \) that decides \( L_\varnothing \).

The machine \( M_\varnothing \) does the following on input \( \langle M \rangle \):

1. Runs \( M_{\text{EQ}} \) on input \( \langle M, M_1 \rangle \), 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 \( L(M) = \varnothing \), then \( L(M) = L(M_1) \) and hence \( M_{\text{EQ}} \) accepts and \( M_\varnothing \) accepts.
- If \( L(M) \neq \varnothing \) 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.

3. Let

\[ \text{REGULAR} = \{ \langle M \rangle \mid M \text{ is a Turing machine and } L(M) \text{ is a regular language} \} \]

Prove that \( \text{REGULAR} \) is undecidable.

Solution. We show reduction:

\[ L_{\text{TM}} \leq_M \text{REGULAR}. \]

Assume that \( \text{REGULAR} \) is decidable, and let \( M_R \) be a Turing machine that decides \( \text{REGULAR} \). We construct Turing machine \( M_L \) that decides \( L_{\text{TM}} \). On the input \( \langle M, w \rangle \), \( M_L \) does the following:

1. Constructs machine \( M_0 \), which on input \( x \) does the following:
   
   (a) if \( x \) is of the form \( 0^n1^n \) – accepts;
   (b) if \( x \) is not of the form \( 0^n1^n \), run \( M \) on input \( w \) and accept if and only if \( M \) accepts.
2. Runs \( M_R \) on input \( \langle M_0 \rangle \).
3. If \( M_R \) accepts – accept, if \( M_R \) rejects – reject.
What is the language of $M_0$?

- If $M$ accepts $w$, then $L(M_0) = \Sigma^*$. This is regular language.
- If $M$ does not accept $w$, then $L(M_0) = \{0^n1^n \mid n \geq 0\}$. This is non-regular language.

Therefore:

- if $M$ accepts $w$ then $L(M_0) = \Sigma^*$ and $M_R$ accepts $\langle M_0 \rangle$ in Step 2. Therefore $M_L$ accepts.
- If $M$ does not accept $w$ then $L(M_0)$ is irregular and $M_R$ rejects $\langle M_0 \rangle$ in Step 2. Therefore, $M_L$ rejects.

So $M_L$ accepts $\langle M, w \rangle$ if and only if $M$ accepts $w$.

**Note.** All steps are computable by the Turing machines. In particular, constructing $M_0$ is possible: first $M_0$ checks for certain type of input and then simulates $M$ on $w$.

**Conclusion.** We found that if there exists $M_R$ (the machine that decides REGULAR), then there exists $M_L$ (the machine that decides $L_{TM}$). Not possible. Contradiction!