Prove that the following languages are undecidable.

See outline of how to solve such problems in the original problem set.

1 \textbf{AcceptIllini} := \{ \langle M \rangle \mid M \text{ accepts the string } ILLINI \}

\textbf{Solution:}

For the sake of argument, suppose there is an algorithm \textbf{DecideAcceptIllini} that correctly decides the language \textbf{AcceptIllini}. Then we can solve the halting problem as follows:

\begin{center}
\textbf{DecideHalt}(\langle M, w \rangle):
\begin{align*}
\text{Encode the following Turing machine } M' : \\
M'(x): &= \begin{cases}
\text{run } M \text{ on input } w \\
\text{return True}
\end{cases} \\
\text{if } \textbf{DecideAcceptIllini}(\langle M' \rangle) \\
\text{return True} \\
\text{else} \\
\text{return False}
\end{align*}
\end{center}

We prove this reduction correct as follows:

\[\Rightarrow\text{ Suppose } M \text{ halts on input } w.\]

Then \( M' \) accepts \textit{every} input string \( x \).

In particular, \( M' \) accepts the string \textit{ILLINI}.

So \textbf{DecideAcceptIllini} accepts the encoding \( \langle M' \rangle \).

So \textbf{DecideHalt} correctly accepts the encoding \( \langle M, w \rangle \).

\[\Leftarrow\text{ Suppose } M \text{ does not halt on input } w.\]

Then \( M' \) diverges on \textit{every} input string \( x \).

In particular, \( M' \) does not accept the string \textit{ILLINI}.

So \textbf{DecideAcceptIllini} rejects the encoding \( \langle M' \rangle \).

So \textbf{DecideHalt} correctly rejects the encoding \( \langle M, w \rangle \).

In both cases, \textbf{DecideHalt} is correct. But that’s impossible, because \textbf{Halt} is undecidable. We conclude that the algorithm \textbf{DecideAcceptIllini} does not exist.

As usual for undecidability proofs, this proof invokes four distinct Turing machines:

- The hypothetical algorithm \textbf{DecideAcceptIllini}.
- The new algorithm \textbf{DecideHalt} that we construct in the solution.
- The arbitrary machine \( M \) whose encoding is part of the input to \textbf{DecideHalt}.
- The special machine \( M' \) whose encoding \textbf{DecideHalt} constructs (from the encoding of \( M \) and \( w \)) and then passes to \textbf{DecideAcceptIllini}.

2 \textbf{AcceptThree} := \{ \langle M \rangle \mid M \text{ accepts exactly three strings} \}
Solution:

For the sake of argument, suppose there is an algorithm \textbf{DecideAcceptThree} that correctly decides the language \textbf{AcceptThree}. Then we can solve the halting problem as follows:

\[
\text{\textbf{DecideHalt}(⟨M, w⟩):}
\]
\[
\text{Encode the following Turing machine } M':
\]
\[
M'(x):
\]
\[
\begin{align*}
\text{run } M \text{ on input } w \\
\text{if } x = ε \text{ or } x = 0 \text{ or } x = 1 \\
\text{return True} \\
\text{else} \\
\text{return False}
\end{align*}
\]

if \textbf{DecideAcceptThree}(⟨M'⟩) then return True
else return False

We prove this reduction correct as follows:

\[\implies\]
Suppose \( M \) halts on input \( w \).
Then \( M' \) accepts exactly three strings: \( ε, 0, \) and \( 1 \).
So \textbf{DecideAcceptThree} accepts the encoding \( ⟨M'⟩ \).
So \textbf{DecideHalt} correctly accepts the encoding \( ⟨M, w⟩ \).

\[\iff\]
Suppose \( M \) does not halt on input \( w \).
Then \( M' \) diverges on every input string \( x \).
In particular, \( M' \) does not accept exactly three strings (because \( 0 \neq 3 \)).
So \textbf{DecideAcceptThree} rejects the encoding \( ⟨M'⟩ \).
So \textbf{DecideHalt} correctly rejects the encoding \( ⟨M, w⟩ \).

In both cases, \textbf{DecideHalt} is correct. But that’s impossible, because \textbf{HALT} is undecidable. We conclude that the algorithm \textbf{DecideAcceptThree} does not exist.

\section{AcceptPalindrome := \{⟨M⟩ | M accepts at least one palindrome\}}

Solution:

For the sake of argument, suppose there is an algorithm \textbf{DecideAcceptPalindrome} that correctly decides the language \textbf{AcceptPalindrome}. Then we can solve the halting problem as follows:

\[
\text{\textbf{DecideHalt}(⟨M, w⟩):}
\]
\[
\text{Encode the following Turing machine } M':
\]
\[
M'(x):
\]
\[
\begin{align*}
\text{run } M \text{ on input } w \\
\text{return True}
\end{align*}
\]

if \textbf{DecideAcceptPalindrome}(⟨M'⟩) then return True
else return False
We prove this reduction correct as follows:

\[\rightarrow\] Suppose \( M \) halts on input \( w \).
  Then \( M' \) accepts every input string \( x \).
  In particular, \( M' \) accepts the palindrome \( RACECAR \).
  So \textcolor{red}{\text{DecideAcceptPalindrome}} accepts the encoding \( \langle M' \rangle \).
  So \textcolor{red}{\text{DecideHalt}} correctly accepts the encoding \( \langle M, w \rangle \).

\[\leftarrow\] Suppose \( M \) does not halt on input \( w \).
  Then \( M' \) diverges on every input string \( x \).
  In particular, \( M' \) does not accept any palindromes.
  So \textcolor{red}{\text{DecideAcceptPalindrome}} rejects the encoding \( \langle M' \rangle \).
  So \textcolor{red}{\text{DecideHalt}} correctly rejects the encoding \( \langle M, w \rangle \).

In both cases, \textcolor{red}{\text{DecideHalt}} is correct. But that’s impossible, because \textcolor{red}{\text{Halt}} is undecidable. We conclude that the algorithm \textcolor{red}{\text{DecideAcceptPalindrome}} does not exist.

Yes, this is \textit{exactly} the same proof as for problem 1.