Prove that the following languages are undecidable.

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

1. $\text{AcceptIllini} := \{\langle M \rangle | M \text{ accepts the string } ILLINI\}$

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

$\text{DecideHalt}(\langle M, w \rangle)$:

Encode the following Turing machine $M'$:

$M'(x)$:

- run $M$ on input $w$
- return True

if $\text{DecideAcceptIllini}(\langle M' \rangle)$
- return True
else
- return False

We prove this reduction correct as follows:

$\implies$ Suppose $M$ halts on input $w$.
Then $M'$ accepts every input string $x$.
In particular, $M'$ accepts the string $ILLINI$.
So $\text{DecideAcceptIllini}$ accepts the encoding $\langle M' \rangle$.
So $\text{DecideHalt}$ correctly accepts the encoding $\langle M, w \rangle$.

$\impliedby$ Suppose $M$ does not halt on input $w$.
Then $M'$ diverges on every input string $x$.
In particular, $M'$ does not accept the string $ILLINI$.
So $\text{DecideAcceptIllini}$ rejects the encoding $\langle M' \rangle$.
So $\text{DecideHalt}$ correctly rejects the encoding $\langle M, w \rangle$.

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

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

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

2. $\text{AcceptThree} := \{\langle M \rangle | M \text{ accepts exactly three strings}\}$
Solution:

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

\[
\text{DecideHalt}(\langle M, w \rangle):
\]

Encode the following Turing machine \( M' \):
\[
M'(x):
\begin{align*}
&\text{run } M \text{ on input } w \\
&\text{if } x = \varepsilon \text{ or } x = 0 \text{ or } x = 1 \\
&\quad \text{return True} \\
&\text{else} \\
&\quad \text{return False}
\end{align*}
\]

if DecideAcceptThree(\( \langle M' \rangle \))
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: \( \varepsilon, 0, \) and \( 1 \).
So DecideAcceptThree accepts the encoding \( \langle M' \rangle \).
So DecideHalt correctly accepts the encoding \( \langle M, w \rangle \).

\[\iff\] 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 DecideAcceptThree rejects the encoding \( \langle M' \rangle \).
So DecideHalt correctly rejects the encoding \( \langle M, w \rangle \).

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

3 \hspace{1cm} \text{AcceptPalindrome} := \{ \langle M \rangle \mid M \text{ accepts at least one palindrome} \}

Solution:

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

\[
\text{DecideHalt}(\langle M, w \rangle):
\]

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

if DecideAcceptPalindrome(\( \langle M' \rangle \))
return True
else
return False
We prove this reduction correct as follows:

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

\[\iff\] 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 DecideAcceptPalindrome rejects the encoding \( \langle M' \rangle \).
So DecideHalt correctly rejects the encoding \( \langle M, w \rangle \).

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

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