Prove that the following languages are undecidable.

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

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

\textbf{Solution}:

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

\[
\textsc{DecideHalt}(\langle M, w \rangle):
\]
\[\quad\text{Encode the following Turing machine } M':\]
\[\quad\quad M'(x):\]
\[\quad\quad\quad\text{run } M \text{ on input } w\]
\[\quad\quad\quad\text{return } \text{TRUE}\]
\[\quad\text{if } \textsc{DecideAcceptIllini}(\langle M' \rangle)\]
\[\quad\quad\text{return } \text{TRUE}\]
\[\quad\text{else}\]
\[\quad\quad\text{return } \text{FALSE}\]

We prove this reduction correct as follows:

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

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

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

As usual for undecidability proofs, this proof invokes \textit{four} distinct Turing machines:

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

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

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

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

Encode the following Turing machine \(M'\):

\[
M'(x):
\]

- run \(M\) on input \(w\)
- if \(x = \varepsilon\) or \(x = 0\) or \(x = 1\) return \texttt{TRUE}
- else return \texttt{FALSE}

if \texttt{DecideAcceptThree}(\(\langle M' \rangle \))
  return \texttt{TRUE}
else
  return \texttt{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 \texttt{DecideAcceptThree} accepts the encoding \(\langle M' \rangle\).
- So \texttt{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 \texttt{DecideAcceptThree} rejects the encoding \(\langle M' \rangle\).
- So \texttt{DecideHalt} correctly rejects the encoding \(\langle M, w \rangle\).

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

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

Solution:

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

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

Encode the following Turing machine \(M'\):

\[
M'(x):
\]

- run \(M\) on input \(w\)
- return \texttt{TRUE}

if \texttt{DecideAcceptPalindrome}(\(\langle M' \rangle \))
  return \texttt{TRUE}
else
  return \texttt{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 \texttt{DecideAcceptPalindrome} accepts the encoding \( \langle M' \rangle \).
    So \texttt{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 \texttt{DecideAcceptPalindrome} rejects the encoding \( \langle M' \rangle \).
    So \texttt{DecideHalt} correctly rejects the encoding \( \langle M, w \rangle \).

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

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