The Halting Problem is Undecidable

We define the language HALT to be the set of all strings of the form $\langle M \rangle w$ such that $M$ halts with input $w$. HALT is the language which is equivalent to the halting problem.

**Theorem 1** HALT is not decidable.

**Proof:** By contradiction. Suppose HALT is decidable. Let $D$ be a machine which implements the following program:

read a machine description $\langle M \rangle$.
if $M$ halts with input $\langle M \rangle$
   run forever.
else
   halt.

We now run $D$ with input $\langle D \rangle$. One of the following two cases must hold.

Case 1. $D$ halts with input $\langle D \rangle$. That means that, when $D$ reads $\langle D \rangle$, it runs forever, hence $D$ does not halt with input $\langle D \rangle$, contradiction.

Case 2. $D$ does not halt with input $\langle D \rangle$. That means that, when $D$ reads $\langle D \rangle$, it halts, hence $D$ halts with input $\langle D \rangle$, contradiction.

In either case, we obtain a contradiction, hence HALT is undecidable.

**Theorem 2** HALT is recognizable.

**Proof:** The following program $P$ recognizes HALT.

read $\langle M \rangle w$
run $M$ with input $w$.
if the $M$ halts with input $w$
   accept $\langle M \rangle w$.

Thus, $P$ accepts every member of HALT, but no other string.

Note that the program will run forever if $\langle M \rangle w \notin$ HALT.