The Halting Problem is Undecidable

Deciding and Accepting

Definition 1 A deterministic machine \( M \) accepts a language \( L \) if for \( w \in L \), the computation of \( M \) with input \( w \) halts in an accepting state, and for \( w \notin L \), the computation of \( M \) with input \( w \) does not halt in an accepting state.

Definition 2 A deterministic machine \( M \) decides a language \( L \) if for \( w \in L \), the computation of \( M \) with input \( w \) halts in an accepting state, and for \( w \notin L \), the computation of \( M \) with input \( w \) halts in a rejecting state.

Definition 3 A non-deterministic machine \( M \) accepts a language \( L \) if for \( w \in L \), there is a computation of \( M \) with input \( w \) which halts in an accepting state, and for \( w \notin L \), there is no computation of \( M \) with input \( w \) which halts in an accepting state.

We will sometimes use the following slightly different, but equivalent, definition:

Definition 4 A non-deterministic machine \( M \) accepts a language \( L \) if for \( w \in L \), there is a computation of \( M \) with input \( w \) which halts, and for \( w \notin L \), there is no computation of \( M \) with input \( w \) which halts.

An accepting computation may require that \( M \) “guesses” correctly at each step.

We say that a language \( L \) is acceptable if there is some machine that accepts \( L \), and that \( L \) is decidable if there is some machine that decides \( L \). Clearly, any decidable language is acceptable.

Let \( T \) be an increasing function on integers. We assume \( T(n) \geq n \).

Definition 5 A deterministic machine \( M \) accepts a language \( L \) in time \( T \) if for \( w \in L \) the computation of \( M \) with input \( w \) halts in an accepting state within \( T(n) \) steps, where \( n = |w| \), and for \( w \notin L \), the computation of \( M \) with input \( w \) does not halt in an accepting state.

Definition 6 A deterministic machine \( M \) decides a language \( L \) in time \( T \) if if for \( w \in L \) the computation of \( M \) with input \( w \) halts in an accepting state within \( T(n) \) steps, where \( n = |w| \), and for \( w \notin L \), the computation of \( M \) with input \( w \) halts in a rejecting state within \( T(n) \) steps.

Theorem 1 If \( T(n) \) is a recursive function (that means computable) and for any \( n \), \( T(n) \) can be computed within \( O(T(n)) \) steps, and if a language \( L \) is accepted by some deterministic machine in time \( T \), Then \( L \) is decided by some deterministic machine in time \( O(T) \).

We define \( \mathcal{P} \text{-time} \) to be the class of all languages which are decided by some deterministic machine in \( \mathcal{P} \text{-time} \), that is in time \( T \) for some polynomially bounded function \( T \).

Definition 7 A non-deterministic machine \( M \) accepts a language \( L \) in time \( T \) if for \( w \in L \), there is a computation of \( M \) with input \( w \) which halts in an accepting state within \( T(|w|) \) steps, and for \( w \notin L \), there is no computation of \( M \) with input \( w \) which halts in an accepting state.

We define \( \mathcal{NP} \text{-time} \) to be the class of all languages which are accepted by some non-deterministic machine in \( \mathcal{P} \text{-time} \), that is in time \( T \) for some polynomial function \( T \).
The Halting Problem and the Diagonal Language

For consistency, we assume all machines are Turing machines. However, we could substitute any sufficiently powerful class of machines, such as all C++ programs.¹

If $M$ is any machine, let $\langle M \rangle$ be its description, which is a string. We assume there is an emulator, a machine which emulates machines. Such an emulator is called a universal machine. If $M$ is a machine and $w$ is a string, and if the input of the emulator is $\langle M \rangle w$, the output of the emulator is the same as the output of $M$ with input $w$.

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. The universal machine accepts HALT, and thus HALT is acceptable.

We define the diagonal language $\text{DIAG} = \{ \langle M \rangle : \langle M \rangle \notin \text{HALT} \}$, that is, the set of all descriptions of machines which do not accept their own descriptions.

**Theorem 2** $\text{DIAG}$ is not acceptable.

**Proof:** By contradiction. Assume that $\text{DIAG}$ is acceptable. Let $M_{\text{DIAG}}$ be a machine that accepts $\text{DIAG}$.

Claim 1: For any machine description $\langle M \rangle$, $M$ halts with input $\langle M \rangle$ if and only if $\langle M \rangle \notin \text{DIAG}$.

Claim 2: For any machine description $\langle M \rangle$, $M_{\text{DIAG}}$ halts with input $\langle M \rangle$ if and only if $\langle M \rangle \in \text{DIAG}$.

The first claim follows from the definition of $\text{DIAG}$, while the second follows from the definition of $M_{\text{DIAG}}$. We now replace $M$ by $M_{\text{DIAG}}$ in each claim. We obtain

Claim 3: $M_{\text{DIAG}}$ halts with input $\langle M_{\text{DIAG}} \rangle$ if and only if $\langle M_{\text{DIAG}} \rangle \notin \text{DIAG}$.

Claim 4: $M_{\text{DIAG}}$ halts with input $\langle M_{\text{DIAG}} \rangle$ if and only if $\langle M_{\text{DIAG}} \rangle \in \text{DIAG}$.

Claim 3 follows from Claim 1 by universal instantiation, while Claim 4 follows from Claim 2 by universal instantiation. These two claims contradict each other. We conclude that no machine accepts $\text{DIAG}$.²

**Theorem 3** HALT is not decidable.

**Proof:** By contradiction. Suppose HALT is decidable. Then, for any machine description $\langle M \rangle$, we can decide whether $M$ halts with input $\langle M \rangle$, which implies that $\text{DIAG}$ is decidable, hence acceptable, contradicting Theorem 2.