Halting Problem for Register Machines

Definition. A register machine \( H \) decides the halting problem if for all \( e, a_1, \ldots, a_n \in \mathbb{N} \), starting \( H \) with

\[
R_0 = 0 \quad R_1 = e \quad R_2 = \lceil [a_1, \ldots, a_n] \rceil
\]

and all other registers zeroed, the computation of \( H \) always halts with \( R_0 \) containing 0 or 1; moreover when the computation halts, \( R_0 = 1 \) if and only if

the register machine program with index \( e \) eventually halts when started with \( R_0 = 0, R_1 = a_1, \ldots, R_n = a_n \) and all other registers zeroed.
Halting Problem for Register Machines

**Definition.** A register machine $H$ **decides the halting problem** if for all $e, a_1, \ldots, a_n \in \mathbb{N}$, starting $H$ with

\[
R_0 = 0 \quad R_1 = e \quad R_2 = \lceil [a_1, \ldots, a_n] \rceil
\]

and all other registers zeroed, the computation of $H$ always halts with $R_0$ containing 0 or 1; moreover when the computation halts, $R_0 = 1$ if and only if

the register machine program with index $e$ eventually halts when started with $R_0 = 0, R_1 = a_1, \ldots, R_n = a_n$ and all other registers zeroed.

**Theorem** No such register machine $H$ can exist.
Proof of the theorem

Assume we have a RM $H$ that decides the halting problem and derive a contradiction.

- Let $H'$ be obtained from $H$ by replacing $\text{START} \rightarrow$ by $START \rightarrow$ \begin{align*}
\text{copy } R_1 \\
\text{to } Z
\end{align*} \rightarrow \begin{align*}
\text{push } Z \\
\text{to } R_2
\end{align*} 
(where $Z$ is a register not mentioned in $H$’s program).

- Let $C$ be obtained from $H'$ by replacing each $\text{HALT}$ (& each erroneous halt) by $\rightarrow R_0^-$ $\leftrightarrow R_0^+$.$

- Let $c \in \mathbb{N}$ be the index of $C$’s program.
Proof of the theorem

Assume we have a RM $H$ that decides the halting problem and derive a contradiction.

$C$ started with $(R_0, R_1, R_2) = (0, c, 0)$ eventually halts if and only if

$H'$ started with $(R_0, R_1, R_2) = (0, c, 0)$ halts with $R_0 = 0$
Proof of the theorem

Assume we have a RM $H$ that decides the halting problem and derive a contradiction.

$C$ started with $(R_0, R_1, R_2) = (0, c, 0)$ eventually halts if and only if

$H'$ started with $(R_0, R_1, R_2) = (0, c, 0)$ halts with $R_0 = 0$ if and only if

$H$ started with $(R_0, R_1, R_2) = (0, c, \neg [c])$ halts with $R_0 = 0$
Proof of the theorem

Assume we have a RM $H$ that decides the halting problem and derive a contradiction.

$C$ started with $(R_0, R_1, R_2) = (0, c, 0)$ eventually halts
if and only if

$H'$ started with $(R_0, R_1, R_2) = (0, c, 0)$ halts with $R_0 = 0$
if and only if

$H$ started with $(R_0, R_1, R_2) = (0, c, \neg[c]\neg)$ halts with $R_0 = 0$
if and only if

$\text{prog}(c)$ started with $(R_0, R_1, R_2) = (0, c, 0)$ does not halt

$\text{prog}(c)$ means the program given by the number $c$. 
Proof of the theorem

Assume we have a RM $H$ that decides the halting problem and derive a contradiction.

$C$ started with $(R_0, R_1, R_2) = (0, c, 0)$ eventually halts
if and only if

$H'$ started with $(R_0, R_1, R_2) = (0, c, 0)$ halts with $R_0 = 0$
if and only if

$H$ started with $(R_0, R_1, R_2) = (0, c, \lceil c \rceil)$ halts with $R_0 = 0$
if and only if

$prog(c)$ started with $(R_0, R_1, R_2) = (0, c, 0)$ does not halt
if and only if

$C$ started with $(R_0, R_1, R_2) = (0, c, 0)$ does not halt
Proof of the theorem

Assume we have a RM $H$ that decides the halting problem and derive a contradiction.

- $C$ started with $(R_0, R_1, R_2) = (0, c, 0)$ eventually halts if and only if
- $H'$ started with $(R_0, R_1, R_2) = (0, c, 0)$ halts with $R_0 = 0$ if and only if
- $H$ started with $(R_0, R_1, R_2) = (0, c, \lceil [c] \rceil)$ halts with $R_0 = 0$ if and only if
- $\text{prog}(c)$ started with $(R_0, R_1, R_2) = (0, c, 0)$ does not halt if and only if
- $C$ started with $(R_0, R_1, R_2) = (0, c, 0)$ does not halt

Contradiction!
For each $e \in \mathbb{N}$, let $\varphi_e \in \mathbb{N} \rightarrow \mathbb{N}$ be the unary partial function computed by the RM with program $prog(e)$. So for all $x, y \in \mathbb{N}$:

$\varphi_e(x) = y$ holds iff the computation of $prog(e)$ started with $R_0 = 0, R_1 = x$ and all other registers zeroed eventually halts with $R_0 = y$.

Thus

$$e \mapsto \varphi_e$$

defines an onto function from $\mathbb{N}$ to the collection of all computable partial functions from $\mathbb{N}$ to $\mathbb{N}$. 
An uncomputable function

Let $f \in \mathbb{N} \rightarrow \mathbb{N}$ be the partial function $\{(x, 0) \mid \varphi_x(x) \uparrow\}$.
Thus $f(x) = \begin{cases} 0 & \text{if } \varphi_x(x) \uparrow \\ undefined & \text{if } \varphi_x(x) \downarrow \end{cases}$
An uncomputable function

Let $f \in \mathbb{N} \rightarrow \mathbb{N}$ be the partial function $\{(x, 0) \mid \varphi_x(x) \uparrow\}$.

Thus $f(x) = \begin{cases} 0 & \text{if } \varphi_x(x) \uparrow \\ \text{undefined} & \text{if } \varphi_x(x) \downarrow \end{cases}$

$f$ is not computable, because if it were, then $f = \varphi_e$ for some $e \in \mathbb{N}$ and hence

- if $\varphi_e(e) \uparrow$, then $f(e) = 0$ (by def. of $f$); so $\varphi_e(e) = 0$ (by def. of $e$), i.e. $\varphi_e(e) \downarrow$

- if $\varphi_e(e) \downarrow$, then $f(e) \uparrow$ (by def. of $f$); so $\varphi_e(e) \uparrow$ (by def. of $e$)

Contradiction! So $f$ cannot be computable.
Given a subset $S \subseteq \mathbb{N}$, its **characteristic function** $\chi_S \in \mathbb{N} \rightarrow \mathbb{N}$ is given by:

$$\chi_S(x) \triangleq \begin{cases} 
1 & \text{if } x \in S \\
0 & \text{if } x \notin S.
\end{cases}$$
Definition. $S \subseteq \mathbb{N}$ is called (register machine) \textbf{decidable} if its characteristic function $\chi_S \in \mathbb{N} \rightarrow \mathbb{N}$ is a register machine computable function. Otherwise it is called \textbf{undecidable}.

So $S$ is decidable iff there is a RM $M$ with the property: for all $x \in \mathbb{N}$, $M$ started with $R_0 = 0$, $R_1 = x$ and all other registers zeroed eventually halts with $R_0$ containing $1$ or $0$; and $R_0 = 1$ on halting iff $x \in S$. 
Claim: \( S_0 \triangleq \{ e \mid \varphi_e(0) \downarrow \} \) is undecidable.
Claim: \( S_0 \triangleq \{ e \mid \varphi_e(0) \downarrow \} \) is undecidable.

Proof (sketch): Suppose \( M_0 \) is a RM computing \( \chi_{S_0} \). From \( M_0 \), using similar techniques to those used for constructing a universal RM, we can construct a RM \( H \) to carry out:

\[
\begin{align*}
&\text{let } R_0 = 0, R_1 = e, R_2 = \lceil [a_1, \ldots, a_n] \rceil \text{ in} \\
&R_1 ::= \lceil (R_1 ::= a_1) ; \cdots ; (R_n ::= a_n) ; \text{prog}(e) \rceil ; \\
&R_2 ::= 0 ; \\
\text{run } M_0
\end{align*}
\]

Then by assumption on \( M_0 \), \( H \) decides the halting problem. **Contradiction.**

So no such \( M_0 \) exists, i.e. \( \chi_{S_0} \) is uncomputable, i.e. \( S_0 \) is undecidable.

[The program instruction \( R_1 ::= a_1 \) means copy \( a_1 \) into the register \( R_1 \).]
Claim: $S_1 \triangleq \{e \mid \varphi_e \text{ total function}\}$ is undecidable.
Claim: \( S_1 \triangleq \{ e \mid \varphi_e \text{ total function} \} \) is undecidable.

Proof (sketch): Suppose \( M_1 \) is a RM computing \( \chi_{S_1} \). From \( M_1 \)'s program we can construct a RM \( M_0 \) to carry out:

\[
\begin{align*}
\text{let } R_0 &= 0, R_1 = e \text{ in } R_1 ::= \neg R_1 ::= 0 ; \ prog(e) \neg ; \\
\text{run } M_1
\end{align*}
\]

Then by assumption on \( M_1 \), \( M_0 \) decides membership of \( S_0 \) from previous example (i.e. computes \( \chi_{S_0} \)). **Contradiction.** So no such \( M_1 \) exists, i.e. \( \chi_{S_1} \) is uncomputable, i.e. \( S_1 \) is undecidable.