The halting problem
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.
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, as follows:

- Let $H'$ be obtained from $H$ by replacing $START \rightarrow$ by
  $START \rightarrow [Z := R_1] \rightarrow\text{push } Z \text{ to } R_2$
  (where $Z$ is a register not mentioned in $H$’s program).

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

- 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, as follows:

$C$ started with $R_1 = c$ eventually halts
if & only if

$H'$ started with $R_1 = c$ halts with $R_0 = 0$
Proof of the theorem

Assume we have a RM $H$ that decides the Halting Problem and derive a contradiction, as follows:

- $C$ started with $R_1 = c$ eventually halts if & only if $H'$ started with $R_1 = c$ halts with $R_0 = 0$
- $H$ started with $R_1 = c, R_2 = \lceil [c] \rceil$ halts with $R_0 = 0$
Proof of the theorem

Assume we have a RM $H$ that decides the Halting Problem and derive a contradiction, as follows:

- $C$ started with $R_1 = c$ eventually halts if & only if
- $H'$ started with $R_1 = c$ halts with $R_0 = 0$ if & only if
- $H$ started with $R_1 = c, R_2 = \overline{[c]}$ halts with $R_0 = 0$ if & only if
- $\text{prog}(c)$ started with $R_1 = c$ does not halt
Proof of the theorem

Assume we have a RM \( H \) that decides the Halting Problem and derive a contradiction, as follows:

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

Assume we have a RM $H$ that decides the Halting Problem and derive a contradiction, as follows:

- $C$ started with $R_1 = c$ eventually halts if & only if
- $H'$ started with $R_1 = c$ halts with $R_0 = 0$ if & only if
- $H$ started with $R_1 = c, R_2 = \lceil c \rceil$ halts with $R_0 = 0$ if & only if
- $\text{prog}(c)$ started with $R_1 = c$ does not halt if & only if
- $C$ started with $R_1 = c$ does not halt — contradiction!
Computable functions

Recall:

**Definition.** \( f \in \mathbb{N}^n \rightarrow \mathbb{N} \) is (register machine) computable if there is a register machine \( M \) with at least \( n + 1 \) registers \( R_0, R_1, \ldots, R_n \) (and maybe more) such that for all \( (x_1, \ldots, x_n) \in \mathbb{N}^n \) and all \( y \in \mathbb{N} \), the computation of \( M \) starting with \( R_0 = 0 \), \( R_1 = x_1, \ldots, R_n = x_n \) and all other registers set to 0, halts with \( R_0 = y \) if and only if \( f(x_1, \ldots, x_n) = y \).

Note that the same RM \( M \) could be used to compute a unary function \( (n = 1) \), or a binary function \( (n = 2) \), etc. From now on we will concentrate on the unary case...
Enumerating computable functions

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 \( \text{prog}(e) \). So for all \( x, y \in \mathbb{N} \):

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

Thus

\[
e \mapsto \varphi_e
\]

defines an onto function from \( \mathbb{N} \) to the collection of all computable partial functions from \( \mathbb{N} \) to \( \mathbb{N} \).
Enumerating computable functions

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

$\varphi_e(x) = y$ holds iff the computation of $\text{prog}(e)$ 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 $\mathbb{N}$.

So this is countable.
An uncomputable function

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

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

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

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 \( \varphi_e(e) = 0 \) (since \( f = \varphi_e \)), hence \( \varphi_e(e) \downarrow \)

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

—contradiction! So \( f \) cannot be computable.
Decision problems

*Entscheidungsproblem* means “decision problem”. Given

- a set $S$ whose elements are finite data structures of some kind
  (e.g. formulas of first-order arithmetic)
- a property $P$ of elements of $S$
  (e.g. property of a formula that it has a proof)

the associated decision problem is:
find an algorithm which
terminates with result 0 or 1 when fed an element $s \in S$
and
yields result 1 when fed $s$ if and only if $s$ has property $P$. 

(Un)decidable sets of numbers

Given a subset $S \subseteq \mathbb{N}$, its characteristic function $\chi_S \in \mathbb{N} \rightarrow \mathbb{N}$ is given by:

$$\chi_S(x) \triangleq \begin{cases} 
1 & \text{if } x \in S \\
0 & \text{if } x \notin S.
\end{cases}$$
(Un)decidable sets of numbers

Definition. $S \subseteq \mathbb{N}$ is called (register machine) \textit{decidable} if its characteristic function $\chi_S \in \mathbb{N} \rightarrow \mathbb{N}$ is a register machine computable function. Otherwise it is called \textit{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 $x \in S$. 
(Un)decidable sets of numbers

Definition. $S \subseteq \mathbb{N}$ is called (register machine) **decidable** if its characteristic function $\chi_S \in \mathbb{N} \rightarrow \mathbb{N}$ is a register machine computable function. Otherwise it is called **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 $x \in S$.

Basic strategy: to prove $S \subseteq \mathbb{N}$ undecidable, try to show that decidability of $S$ would imply decidability of the Halting Problem.

For example...
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 \( M_0 \)’s program (using the same techniques as for constructing a universal RM) we can construct a RM \( H \) to carry out:

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

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.
Claim: $S_1 \triangleq \{ e \mid \varphi_e \text{ a total function} \}$ is undecidable.
Claim: $S_1 \triangleq \{ e \mid \varphi_e \text{ a total function} \}$ 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:

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

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.
Exercise 5  If $f: \mathbb{N} \to \mathbb{N}$ is a RM computable function, $S_0 \subseteq \mathbb{N} \& S_1 \subseteq \mathbb{N}$ satisfy

$$\forall e \in \mathbb{N}. \ e \in S_0 \iff f(e) \in S_1$$

then if $S_1$ is decidable, then so is $S_0$. 