**The Halting Problem**

**Definition:** A register machine $H$ decides the Halting Problem if, loading $R_1$ with $e$ and $R_2$ with $[a_1, \ldots, a_n]$ (and all other registers with 0), the computation of $H$ halts with $R_0$ containing either 0 or 1; moreover, $R_0$ contains 1 when $H$ halts if and only if the computation of the register machine program $Prog_e$ started with registers $R_1, \ldots, R_n$ set to $a_1, \ldots, a_n$ (and all other registers set to 0) does halt.

**Theorem:** No such register machine $H$ can exist.

**Proof:** Suppose such an $H$ exists and derive a contradiction ...
Let $H'$ be obtained from $H$ by replacing

\[ \text{START} \rightarrow \text{START} \rightarrow \quad \text{copy R1 to Z} \quad \text{push Z to R2} \rightarrow \]

(where $Z$ is a register not mentioned in $H$'s program)

Let $C$ be obtained from $H'$ by replacing each HALT (\& each jump to a label with no instruction)

\[ \quad \text{RO}^{-} \rightarrow \text{RO}^{+} \]

\[ \downarrow \]

\[ \text{HALT} \]

Let $c \in \mathbb{N}$ be the index of $C$'s program.

$C$ started with $R1 = c$ eventually halts

iff

$H'$ started with $R1 = c$ halts with $RO = 0$

iff

$H$ started with $R1 = c$ \& $R2 = [C]$ halts with $RO = 0$

iff

$\text{Prog}_c$ started with $R1 = c$ does not halt

iff

$C$ started with $R1 = c$ does not halt

CONTRADICTION!

(to the assumption that such an $H$ exists)
Recall:

**Definition:**

A function \( f \in \text{Pfn}(\mathbb{N}^n, \mathbb{N}) \) is *(register machine) computable* if and only if there is a register machine \( M \) with at least \( n+1 \) registers, \( R_0, R_1, R_2, \ldots, R_n \) say, (and maybe some other registers as well) with the property that for all \( (x_1, \ldots, x_n) \in \mathbb{N}^n \) and all \( y \in \mathbb{N} \)

\[ f(x_1, \ldots, x_n) = y \] if and only if the computation of \( M \) starting with \( R_1 = x_1, \ldots, R_n = x_n \), and all other registers \( = 0 \), halts with \( R_0 = y \).

---

**Enumerating computable functions**

For each \( e \in \mathbb{N} \) let \( \varphi_e \in \text{Pfn}(\mathbb{N}, \mathbb{N}) \) be the partial function computed by \( \text{Pro} \varphi_e \), i.e.

\[ \varphi_e(x) = y \leftrightarrow \text{the computation of } \text{Pro} \varphi_e \text{ started with } R_1 = x \text{ (and all other registers zeroed) halts with } R_0 = y \]

Thus:

The function \( e \mapsto \varphi_e \) maps \( \mathbb{N} \) onto the collection of all computable partial functions from \( \mathbb{N} \) to \( \mathbb{N} \).
Not all partial functions are computable.

Define \( f \in \text{Pfn}(\mathbb{N}, \mathbb{N}) \) by:

\[
f(e) = \begin{cases} 
0 & \text{if } \varphi_e(e) \uparrow \\
\text{undefined} & \text{if } \varphi_e(e) \downarrow
\end{cases}
\]

CLAIM: \( f \) is not computable.

PROOF: If \( f \) computable, then \( f = \varphi_e \) for some \( e \).

Then

- \( \varphi_e(e) \uparrow \iff f(e) = 0 \iff \varphi_e(e) = 0 \Rightarrow \varphi_e(e) \downarrow \)
- \( \varphi_e(e) \downarrow \Rightarrow f(e) \uparrow \Rightarrow \varphi_e(e) \uparrow \) \( \text{contradiction!} \)

(\text{Un}) decidable sets of numbers

A subset \( S \subseteq \mathbb{N} \) is \textbf{(register machine) decidable} if and only if there is a register machine \( M \) with the property: for all \( x \in \mathbb{N} \), \( M \) started with \( R1 = x \) (and other registers zeroed) always halts with \( RO \) containing either 0 or 1; moreover \( RO = 1 \) when \( M \) halts if and only if \( x \in S \).

Equivalently: \( S \) is decidable if and only if there is some \( e \) such that for all \( x \in \mathbb{N} \)

either \(( \varphi_e(x) = 0 \land x \notin S ) \) or \(( \varphi_e(x) = 1 \land x \in S ) \)

\( S \) is called \textbf{undecidable} if no such \( M \) (or \( e \)) exists.
Some examples of undecidable sets of numbers

\[ S_1 \triangleq \{ (e, a) \mid \varphi_e(a) \downarrow \} \]
i.e. one-argument version of Halting Problem

\[ S_2 \triangleq \{ e \mid \varphi_e(0) \downarrow \} \]
i.e. \# register machine to decide whether any program halts when supplied with input 0

\[ S_3 \triangleq \{ e \mid \varphi_e \text{ is a total function} \} \]
i.e. \# register machine to decide whether any program halts for all input data

Ex. 1. The proof that \( S_1 \) is undecidable is like the proof of the undecidability of the \( n \)-argument Halting Problem given above, except that now the modification of \( H \) to \( H' \) is:

replace \( \text{START} \rightarrow \text{copy R1 to Z} \rightarrow \text{push Z to R1} \rightarrow \text{R1} \rightarrow \) by

\[ \{ \text{R1} = e, \text{Z} = 0 \} \rightarrow \{ \text{R1} = e, \text{Z} = e \} \rightarrow \{ \text{R1} = \langle e, e \rangle, \text{Z} = 0 \} \rightarrow \{ \text{R1} = \langle e, e \rangle, \text{Z} = 0 \} \]

(the rest of the argument is the same).
Ex. 2. Undecidability of $S_2$ can be reduced to the undecidability of $S_1$:

If $M$ were a register machine for deciding membership of $S_2$, then the register machine specified by

- **START**
  - decode $R_1$ as a pair $<e, a>$
  - and put $e$ in $R_1$ and $a$ in $R_2$

would decide membership of $S_1$. So no such $M$ exists.

**Remark.** We can restate the proof of Ex. 2 in terms of functions: it suffices to show that there is a function $f \in \text{Fun}(\mathbb{N}, \mathbb{N})$ satisfying

- $f$ is computable
- for all $e, a \in \mathbb{N}$ $\varphi_f(<e, a>) (0) \equiv \varphi_e (a)$

and hence $<e, a> \in S_1 \iff f(<e, a>) \in S_2$.

For in general we have for subsets $S_1, S_2 \subseteq \mathbb{N}$

$S_2$ decidable, $f$ computable $\Rightarrow \forall x \in \mathbb{N}. x \in S_1 \iff f(x) \in S_2$

$\Rightarrow S_1$ decidable

(Why?)
Ex.3. Undecidability of $S_3$ can be reduced to that of $S_2$:

If $M$ were a register machine for deciding membership of $S_3$, then the register machine specified by

\[
\text{START} \xrightarrow{\text{decode } R_1 \text{ as a program } \text{START} \rightarrow \text{Prog}.} \]

and put in $R_1$ a code for the program

\[
\text{START} \rightarrow R_1 \rightarrow \text{Prog.}
\]

would decide membership of $S_2$. So no such $M$ exists.