The Halting Problem

The halting problem is historically important because it was one of the first problems to be proved *undecidable*: that is, not computable by, for example, a register machine. (Turing's proof using Turing machines went to press in May 1936, whereas Alonzo Church's proof using the lambda calculus had already been published in April 1936.) Subsequently, many other undecidable problems have been described. The typical method of proving a problem to be undecidable is to reduce it to a problem that is already known to be undecidable. To do this, it is sufficient to show that if a solution to the new problem were found, it could be used to decide an undecidable problem by transforming instances of the undecidable problem into instances of the new problem. Since we already know that no method can decide the old problem, no method can decide the new problem either. Often the new problem is reduced to solving the halting problem.

Slide 1

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

$$R_0 = 0 \qquad R_1 = e \qquad R_2 = \ulcorner [a_1, \ldots, a_n] \urcorner$$

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.

Slide 2

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

$$R_0 = 0 \qquad R_1 = e \qquad R_2 = \ulcorner [a_1, \ldots, a_n] \urcorner$$

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.

Slide 3

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

  $$START \rightarrow \boxed{\begin{array}{c} copy\ R_1 \\ to\ Z \end{array}} \rightarrow \boxed{\begin{array}{c} push\ Z \\ to\ R_2 \end{array}} \rightarrow$$

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

- Let C be obtained from H' by replacing each $HALT$ (& each erroneous halt) by

  $$\begin{tikzcd} {} \arrow[r] & R_0^- \arrow[r, bend left] \arrow[d] & R_0^+ \arrow[l, bend left] \\ & HALT & \end{tikzcd} .$$

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

Slide 4

Proof of the theorem

Assume we have a RM H that decides the halting problem and derive a contradiction, as follows:

$$C \text{ started with } (R_0, R_1, R_2) = (0, c, 0) \text{ eventually halts}$$

if and only if

$$H' \text{ started with } (R_0, R_1, R_2) = (0, c, 0) \text{ halts with } R_0 = 0$$

Slide 5

Proof of the theorem

Assume we have a RM H that decides the halting problem and derive a contradiction, as follows:

$$C \text{ started with } (R_0, R_1, R_2) = (0, c, 0) \text{ eventually halts}$$

if and only if

$$H' \text{ started with } (R_0, R_1, R_2) = (0, c, 0) \text{ halts with } R_0 = 0$$

if and only if

$$H \text{ started with } (R_0, R_1, R_2) = (0, c, \ulcorner [c] \urcorner) \text{ halts with } R_0 = 0$$

Slide 6

Proof of the theorem

Assume we have a RM H that decides the halting problem and derive a contradiction, as follows:

C started with $(R_0, R_1, R_2) = (0, c, 0)$ 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 $(R_0, R_1, R_2) = (0, c, \ulcorner [c] \urcorner)$ 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

$prog(c)$ means the program given by the number c.

Slide 7

Proof of the theorem

Assume we have a RM H that decides the halting problem and derive a contradiction, as follows:

C started with $(R_0, R_1, R_2) = (0, c, 0)$ 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 $(R_0, R_1, R_2) = (0, c, \ulcorner [c] \urcorner)$ 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

Slide 8

Proof of the theorem

Assume we have a RM H that decides the halting problem and derive a contradiction, as follows:

C started with $(R_0, R_1, R_2) = (0, c, 0)$ 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 $(R_0, R_1, R_2) = (0, c, \ulcorner[c]\urcorner)$ 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

Contradiction!

Slide 9

Enumerating computable functions

For each $e \in \mathbb{N}$, let $\varphi_e \in \mathbb{N} \rightharpoonup \mathbb{N}$ 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 $\mathbb{N}$.

Notice that the collection of all computable partial functions from $\mathbb{N}$ to $\mathbb{N}$ is countable. So $\mathbb{N} \rightharpoonup \mathbb{N}$ (uncountable, by Cantor) contains uncomputable functions.

Slide 10

An uncomputable function

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

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

Slide 11

An uncomputable function

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

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

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 e); so $\varphi_e(e)\uparrow$ (by def. of f)

Contradiction! So f cannot be computable.

Slide 12

(Un)decidable sets of numbers

Given a subset $S \subseteq \mathbb{N}$, its **characteristic function** $\chi_S \in \mathbb{N} \to \mathbb{N}$ is given by: $\chi_S(x) \triangleq \begin{cases} 1 & \text{if } x \in S \\ 0 & \text{if } x \notin S. \end{cases}$

Slide 13

(Un)decidable sets of numbers

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

In order to prove that a set $S \subseteq \mathbb{N}$ is undecidable, we show that the decidability of S would imply the decidability of the halting problem.

Slide 14

Claim: $S_0 \triangleq \{e \mid \varphi_e(0){\downarrow}\}$ is undecidable.

Slide 15

Claim: $S_0 \triangleq \{e \mid \varphi_e(0){\downarrow}\}$ is undecidable.

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

$$
\begin{aligned}
&\mathit{let}\ R_0 = 0, R_1 = e, R_2 = \ulcorner[a_1, \ldots, a_n]\urcorner\ \mathit{in} \\
&\quad R_1 ::= \ulcorner(R_1 ::= a_1)\ ; \cdots ; (R_n ::= a_n)\ ; \mathit{prog}(e)\urcorner\ ; \\
&R_2 ::= 0\ ; \\
&\mathit{run}\ M_0
\end{aligned}
$$

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

So no such M_0 exists, i.e. χ_{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.]

Slide 16

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

Slide 17

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

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

$$let \; R_0 = 0, R_1 = e \; in \; R_1 ::= \ulcorner R_1 ::= 0 \,;\, prog(e) \urcorner \,;$$
$$run \; M_1$$

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