## The Halting Problem

$A_{TM} = \{ \langle M, w \rangle \mid M \text{ is a TM and } M \text{ accepts } w \}$

U = "On input ⟨M, w⟩, where M is a TM and w is a string:
1. Simulate M on input w.
2. If M ever enters its accept state, accept; if M ever enters its reject state, reject.

[ U recognizes $A_{TM}$, but does not decide it, because if M loops forever, so does U ]

[ $A_{TM}$ is not decidable, but how do we prove it? ]

## Diagonalization

### Definition

A set A is ***countable*** if it is finite or there is a one-to-one correspondence between all the elements of A and **N**

[ If there is a one-to-one correspondence between all the elements of any two sets, we say they have the same cardinality (or size) ]

[ show that the even numbers are countable ]

[ show that the rational numbers are countable ]

### Theorem

**R** is uncountable

### Proof

It's sufficient to show that [0, 1] is uncountable.
Let f: N → [0, 1] be one-to-one and onto.

[ one-to-one: f(47) and f(635) can't map to the same real number ]
[ onto: every real is included in the mapping ]

$f(1) = 0.b_{1,1}b_{1,2}b_{1,3}b_{1,4}b_{1,5} \ldots$
$f(2) = 0.b_{2,1}b_{2,2}b_{2,3}b_{2,4}b_{2,5} \ldots$
$f(3) = 0.b_{3,1}b_{3,2}b_{3,3}b_{3,4}b_{3,5} \ldots$
$f(4) = 0.b_{4,1}b_{4,2}b_{4,3}b_{4,4}b_{4,5} \ldots$
$f(5) = 0.b_{5,1}b_{5,2}b_{5,3}b_{5,4}b_{5,5} \ldots$
…

where each $b_{i,j}$ is a binary digit (0 or 1)

We construct a real number $a = 0.a_1a_2a_3\ldots$ that is not included in this mapping.

$a_1 \neq b_{1,1}$ ( if $b_{1,1}$ is 0, $a_1$ is 1; if $b_{1,1}$ is 1, $a_1$ is 0 )
$a_2 \neq b_{2,2}$
$a_3 \neq b_{3,3}$
$a_4 \neq b_{4,4}$
$a_5 \neq b_{5,5}$
…

Suppose a is in the mapping.
Then f(n) = a for some n.
The n-th digit in f(n) is $b_{n,n}$
The n-th digit in a is $a_n$
But by construction $a_n \neq b_{n,n}$

[ why can't we have 1 = .100000…, 2 = .010000…, 3 = .1100000…, … ]

---

**Theorem**

$A_{TM}$ is undecidable ( recall that $A_{TM} = \{ \langle M, w\rangle \mid M \text{ is a TM and } M \text{ accepts } w \}$ )

**Proof**

Suppose $A_{TM}$ is decidable
Let H be a decider for $A_{TM}$

$$\text{Then } H = \begin{cases} \textit{accept} & \text{if M accepts w} \\ \textit{reject} & \text{if M rejects or loops on w} \end{cases}$$

Construct D = " On input ⟨M⟩ : [ M is a TM ]
1. Run H on input ⟨M, ⟨M⟩⟩ [ ex: Pascal compiler written in Pascal ]
2. Output the opposite of what H outputs
(if H accepts, reject; if H rejects, accept) "

Running H on input ⟨D, ⟨D⟩⟩ yields a contradiction:
Case A: H accepts ⟨D, ⟨D⟩⟩ ( meaning that D accepts ⟨D⟩ )
Therefore we reject ( meaning D rejects ⟨D⟩ )
$\Rightarrow\Leftarrow$
Case B: H rejects ⟨D, ⟨D⟩⟩ ( meaning that D rejects ⟨D⟩ )
Therefore we accept ( meaning D accepts ⟨D⟩ )
$\Rightarrow\Leftarrow$

In both case, we get a contradiction, therefore $A_{TM}$ is not decidable.

[ the book shows how this proof can be viewed as a diagonalization proof ]

---

**Definition** A language is ***co-Turing-recognizable*** if its complement in Turing-recognizable.

**Theorem**

A language is decidable iff it is Turing-recognizable and co-Turing-recognizable.

**Proof**

( $\Rightarrow$ )
Assume A is decidable
Then L is Turing-recognizable
And $L'$ is decidable
So $L'$ is Turing-recognizable
Therefore, A is decidable $\Rightarrow$ A and $A'$ are both Turing-recognizable

( $\Leftarrow$ )
Assume both A and $A'$ are Turing-recognizable
Let $M_1$ be a TM that recognizes A
Let $M_2$ be a TM that recognizes $A'$
Construct M = " On input w:

1. Run both $M_1$ and $M_2$ on input w in parallel
2. If $M_1$ accepts, accept; if $M_2$ accepts, reject "

$w \in A \Rightarrow M_1$ halts & accepts $\Rightarrow$ M halts & accepts
$w \notin A \Rightarrow M_2$ halts & rejects $\Rightarrow$ M halts & rejects
Therefore, M decides A
Therefore, A & $A'$ are Turing-recognizable $\Rightarrow$ A is decidable

**Corollary**

$A'_{TM}$ is not Turing-recognizable

**Proof**

If it were, ATM would be decidable (which is isn't)

**Reducibility**

**Theorem** $HALT_{TM} = \{ \langle M, w\rangle \mid \text{TM } M \text{ halts on input } w \}$ is undecidable

**Proof**

Suppose $HALT_{TM}$ is decidable
    Let R be a decider for $HALT_{TM}$
    (*) Construct TM S that uses R to decide $A_{TM}$
    $A_{TM}$ is undecidable $\Rightarrow\Leftarrow$
$HALT_{TM}$ is undecidable

S = " On input $\langle M, w\rangle$:
1. Run R on $\langle M, w\rangle$
2. If R rejects (M does not halt on w), *reject*
3. If R accepts (M halts on w), run M on w
    4. If M accepts, *accept*
    5. If M rejects, *reject*

**Theorem** $E_{TM} = \{ \langle M\rangle \mid M \text{ is a TM and } L(M) = \varnothing \}$ is undecidable

**Proof**

Suppose $E_{TM}$ is decidable
    Let R be a decider for $E_{TM}$
    (*) Construct TM S that uses R to decide $A_{TM}$
    $A_{TM}$ is undecidable $\Rightarrow\Leftarrow$
$E_{TM}$ is undecidable

S = " On input $\langle M, w\rangle$:
1. Construct $M_1$ that rejects all strings that are not w, and accepts w only if M accepts w.
   ( M1 = On x: **if** $x \neq w$, *reject* **else** Run M on w; if M accepts, *accept* )
   [ $M_1$ is not a decider]
   [ we are not running it, we are merely constructing it ]
2. Run R on $M_1$
3. R rejects $M_1 \Rightarrow L(M_1) \neq \varnothing \Rightarrow M_1$ accepts w $\Rightarrow$ M accept w; accept $\langle M, w\rangle$
4. R accepts M1 $\Rightarrow L(M_1) = \varnothing \Rightarrow M_1$ does not accepts w $\Rightarrow$ M does not accept w (it reject or loops on w); reject $\langle M, w\rangle$

**Theorem** $REGULAR_{TM} = \{ \langle M\rangle \mid M \text{ is a TM and } L(M) \text{ is regular} \}$ is undecidable

**Proof**

Suppose $REGULAR_{TM}$ is decidable
    Let R be a decider for $REGULAR_{TM}$
    (*) Construct TM S that uses R to decide $A_{TM}$
    $A_{TM}$ is undecidable $\Rightarrow\Leftarrow$
$REGULAR_{TM}$ is undecidable

S = " On input ⟨M, w⟩:

5. Construct $M_2$ that accepts all string in the non-regular language $0^n1^n$, and accepts all other string only if M accepts w.
   [ therefore if M accepts w, $M_2$ recognizes $\Sigma^*$, which is regular ]
   ( $M_2$ = On x: if x has form $0^n1^n$, *accept* **else** Run M on w; if M accepts, *accept* )
   [ $M_2$ is not a decider]
   [ we are not running it, we are merely constructing it ]
6. Run R on $M_2$
7. R rejects $M_2 \Rightarrow L(M_2)$ is regular $\Rightarrow M_2$ accepts all strings $\Rightarrow$ M accepts w; accept ⟨M, w⟩
8. R accepts M1 $\Rightarrow L(M_1)$ is not regular $\Rightarrow M_2$ only accepts string of form $0^n1^n \Rightarrow$ M does not accept w (it reject or loops on w); reject ⟨M, w⟩ "

**Theorem** $EQ_{TM} = \{ \langle M_1, M_2 \rangle \mid L(M_1) = L(M_2) \}$ is undecidable

**Proof**
( show that if $EQ_{TM}$ is decidable, so is $E_{TM}$ ) [ fairly easy ]

**Theorem** $ALL_{CFG} = \{ \langle G \rangle \mid G \text{ is a CFG and } L(G) = \Sigma^* \}$

[ proof is in book; non-trivial ]

**The Domino Problem (PCP)**

[ describe the domino problem, state that its undecidable ]

A single domino: $\left[\frac{a}{ab}\right]$

A set of dominos: $\left\{ \left[\frac{b}{ca}\right], \left[\frac{a}{ab}\right], \left[\frac{ca}{a}\right], \left[\frac{abc}{c}\right] \right\}$

Problem: write a program that list the dominos (repeats OK) so that:

top string of symbols = bottom string of symbols (if such a listing exists)

For example: $\left[\frac{a}{ab}\right]\left[\frac{b}{ca}\right]\left[\frac{ca}{a}\right]\left[\frac{a}{ab}\right]\left[\frac{abc}{c}\right]$ is a solution to the set above.

Impossible! [ Not that it "takes to long" you can't do it on a computer ]

---

[ next week : mapping reducibility ]