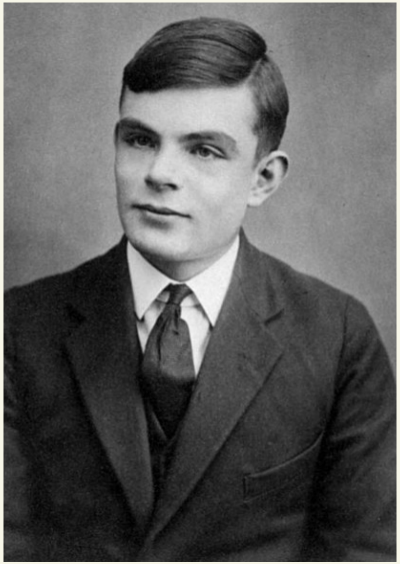
CSC 438/2404 Week 7

- HW3 Due Nov 11
- TODAY: computability
  (see notes from last Friday on Turing Machines)

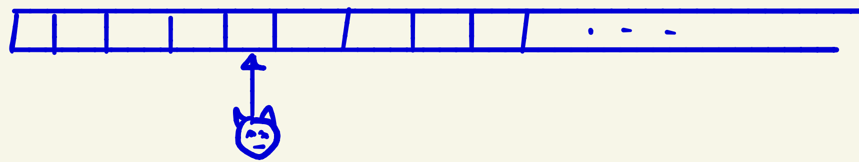
Turing Machines

"On computable Numbers, with an application to the Entscheidungsproblem"
1936



1912 - 1954

Turing Machines:

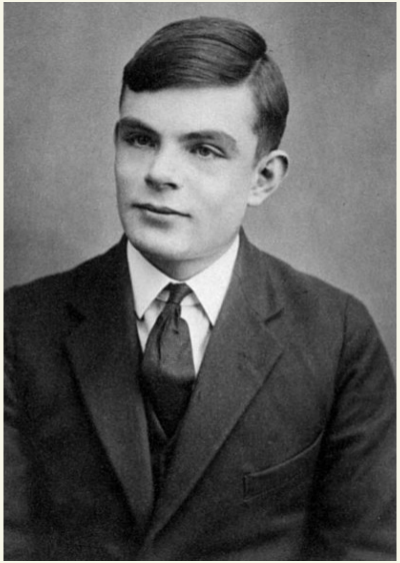


q_3

$$M = (Q, \Sigma, \Gamma, \delta, q_1, B, \{q_2\})$$

Turing Machines

"On computable Numbers, with an application to the Entscheidungsproblem"
1936



1912 - 1954

Church-Turing Thesis

A function/predicate is computable/realizable in physical world $\Longrightarrow$ it is computable by a TM

Notation

$\{x\}$ = Turing machine M such that $\#M = x$

$\{x\}_1$ = the unary function computed by x

$\{x\}_n$ = the n-ary function computed by x

(can generalize earlier so M takes n inputs instead of 1)

Today

What is computable and what isn't?

We will mostly focus on unary relations
or languages — $L \subseteq \{0,1\}^*$

all finite length strings over $\{0,1\}$

Definition Let M be a TM, $\Sigma = \{0,1\}$

$\mathcal{L}(M) \subseteq \{0,1\}^*$ is the set of all (finite-length) strings $x \in \{0,1\}^*$ such that $M(x)$ halts and outputs 1

the language accepted by M

Recursive / RE Sets

A language $L \subseteq \{0,1\}^*$ is <u>recursively</u> <u>enumerable</u> if there exists a TM M such that $\mathcal{L}(M) = L$

So $\forall x \in \{0,1\}^*$

$x \in L \Rightarrow M$ on x halts and outputs "1"

$x \notin L \Rightarrow M$ on x halts and does not output 1

or M does not halt on x

Recursive / RE Sets

A language $L \subseteq \{0,1\}^*$ is recursively enumerable if there exists a TM M such that $\mathcal{L}(M) = L$

So $\forall x \in \{0,1\}^*$

$x \in L \Rightarrow M$ on x halts and outputs "1"

$x \notin L \Rightarrow M$ on x halts and does not output 1

or M does not halt on x

recursively enumerable (r.e.) also called semidecidable, partial computable

Recursive / RE Sets

A language $L \subseteq \{0,1\}^*$ is <u>recursive</u> if there exists a TM M such that $\mathcal{L}(M) = L$ and M always halts

So $\forall x \in \{0,1\}^*$

$x \in L \Rightarrow$ M on x halts and outputs "1"

$x \notin L \Rightarrow$ M on x halts and does not output 1

(without loss of generality, $x \notin L \Rightarrow M(x)$ halts & outputs "0")

Recursive / RE Sets

A language $L \subseteq \{0,1\}^*$ is <u>recursive</u> if there exists a TM M such that $\mathcal{L}(M) = L$ and M always halts

So $\forall x \in \{0,1\}^*$

$x \in L \Rightarrow M$ on x halts and outputs "1"

$x \notin L \Rightarrow M$ on x halts and does not output 1

(without loss of generality, $x \notin L \Rightarrow M(x)$ halts + outputs "0")

recursive also called decidable, computable

Recursive / RE Sets

A function $f : \{0,1\}^* \to \{0,1\}^*$ (or $f : \mathbb{N}^n \to \mathbb{N}$) is **total computable** if there exists a TM M such that $\forall x \in \{0,1\}^*$ $M(x)$ halts and outputs $f(x)$.

Easy Properties

① L recursive $\Rightarrow$ L r.e.

② Class of recursive languages is closed under

$\cap, \cup, \neg$:

L_1, L_2 recursive $\Rightarrow$ $L_1 \cup L_2$, $L_1 \cap L_2$, $\neg L_1$, $\neg L_2$
are recursive

③ Total computable functions closed under composition: f, g computable

$\Rightarrow f \circ g \stackrel{d}{=} f(g(x))$ is computable

Another Property (not as easy)

④ L r.e., and $\bar{L}$ r.e. $\Rightarrow$ L is recursive

$\{x \mid x \notin L\}$

* **Note**: often $L \subseteq \{0,1\}^*$ is a set of encodings. Example $L = \{x \mid \{x\}_1 \text{ accepts input "111"}\}$ then we usually think of $\bar{L}$ as $\{x \mid \{x\}_1 \text{ does not accept } 111\}$ although technically

$\bar{L} = \{x \mid x \text{ is not a legal encoding or } \{x\}_1 \text{ does not accept } 111\}$

Another Property (not as easy)

④ L r.e., and $\bar{L}$ r.e. $\Rightarrow$ L is recursive

Proof: (Dovetailing)

Let M_1 be a TM st $\mathcal{L}(M) = L$,
M_2 be a TM st $\mathcal{L}(M) = \bar{L}$

New TM M on x:

For $i = 1, 2, 3, \ldots$

Run M_1 on x for i steps
- if M_1 accepts x halt + accept

Run M_2 on x for i steps
- if M_2 accepts x, halt + reject

Another Property (not as easy)

④ L r.e., and $\overline{L}$ r.e. $\Rightarrow$ L is recursive

Proof: (Dovetailing)

Let M_1 be a TM st $\mathcal{L}(M) = L$,
M_2 be a TM st $\mathcal{L}(M) = \overline{L}$

New TM M on x:

For $i = 1, 2, 3, \ldots$

Run M_1 on x for i steps
if M_1 accepts x halt + accept

Run M_2 on x for i steps
if M_2 accepts x, halt + reject

- M on x eventually halts since x accepted by exactly one of M_1, M_2
- $x \in L \Rightarrow M_1$ accepts $x \Rightarrow M$ accepts x
- $x \notin L \Rightarrow M_2$ accepts $x \Rightarrow M$ halts and rejects x

Many Languages are Not r.e.

Proof : Diagonalization

Main idea : There are many more Languages (subsets of $\{0,1\}^*$) than there are TMs.

Proof very similar to Cantor's argument showing that there is NO 1-1 mapping from the Real numbers to the Natural numbers

Many Languages are Not r.e.

Proof : Diagonalization

- Fix an enumeration of all TMs with $\Sigma = \{0,1\}$
  $M_1, M_2, M_3, \ldots\ldots$
  (use encoding of TMs to give an enumeration)
- Make a 2-way infinite (but countable) table
  rows correspond to $M_1, M_2, \ldots$
  columns correspond to enumeration of
  encodings of Turing machines $x_1, x_2, \ldots$
- Entry (i,j): 0 if M_i accepts x_j
  1 otherwise

Many Languages are Not r.e.

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	$\cdots$
M_1	0	1	1	0	1	0	0	$\cdots$
M_2	0	0	1	1	0	1	1	
M_3	1	1	1	1	1	0	1	
M_4	1	1	0	0	0	0	1	
M_5	0	0	0	0	1	1	1	
$\vdots$	0	1	0	1	0	1	0	
	$\vdots$							

Many Languages are Not r.e.

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	$\cdots$
M_1	0	1	1	0	1	0	0	$\cdots$
M_2	0	0	1	1	0	1	1	
M_3	1	1	1	1	1	0	1	
M_4	1	1	0	0	0	0	1	
M_5	0	0	0	0	1	1	1	
$\vdots$	0	1	0	1	0	1	0	

Diagonal Language D

$$D(x) = \{ x_j \mid M_j \text{ does not accept } x_j \}$$

Theorem D is not r.e.

Proof By construction: For all TMs M_i,

$M_i(x_i) \neq D(x_i)$ so $\mathcal{L}(M_i) \neq D$

$\therefore$ D not r.e.

Using Reductions to show other (more natural) languages/functions are not computable/recursive/r.e.

High Level:

1. Say we know L_1 not recursive
   To show L_2 not recursive, design a TM M_1 always halts & $\mathcal{L}(M_1) = L_1$, assuming a TM M_2 that always halts & $\mathcal{L}(M_2) = L_2$

2. Suppose L_1 not r.e.
   To show L_2 not r.e., construct M_1 st $\mathcal{L}(M_1) = L_1$ assuming a TM M_2 st $\mathcal{L}(M_2) = L_2$

The Halting Problem is Not Recursive

$$K \stackrel{d}{=} \{ x \mid \text{TM } \{x\} \text{ halts on input } x \}$$

$$HALT \stackrel{d}{=} \{ \langle x, y \rangle \mid \text{TM } \{x\} \text{ halts on input } y \}$$

Claim HALT, K are both r.e.

Pf: simply run $\{x\}$ on y. Accept if simulation halts.

The Halting Problem is Not Recursive

$K \overset{d}{=} \{ x \mid \text{TM } \{x\} \text{ halts on input } x \}$

Theorem K is not recursive

Proof Let $L_1 = D$. We know L_1 is not r.e.
Assume $L_2 = K$ is recursive, & let M_2 always halt & $\mathcal{L}(M_2) = L_2$

Construction of TM M_1 for D on input x:

Run M_2 on x
- If M_2 accepts x then
  Run $\{x\}$ on x and output 1 iff $\{x\}(x) \neq 1$
- If M_2 halts & does not accept x then output 1

The Halting Problem is Not Recursive

$K \overset{d}{=} \{ x \mid TM \{x\} \text{ halts on input } x \}$

Theorem K is not recursive

Proof Let $L_1 = D$. We know L_1 is Not r.e.
Assume $L_2 = K$ is recursive, + Let M_2 always halt + $\mathcal{L}(M_2) = L_2$
Construction of TM M_1 for D on input x:

Run M_2 on x
- If M_2 accepts x then
  Run $\{x\}$ on x and output 1 iff $\{x\}(x) \neq 1$
- If M_2 halts + does not accept x then output 1

- M_1 halts on all x
- $x \in D \Rightarrow \{x\}(x) \neq 1 \Rightarrow M_1(x) = 1$
- $x \notin D \Rightarrow \{x\}(x) = 1 \Rightarrow M_1(x) \neq 1$

The Halting Problem is Not Recursive

$$K \stackrel{d}{=} \{ x \mid \text{TM } \{x\} \text{ halts on input } x \}$$

✓ <u>Theorem</u> K is not recursive

<u>Theorem</u> $\overline{K}$ is not r.e.

K is r.e.

K r.e. and $\overline{K}$ r.e. $\underset{\text{property (4)}}{\Longrightarrow}$ K recursive

$\therefore$ $\overline{K}$ not r.e.

The Halting Problem is Not Recursive

$K \stackrel{d}{=} \{ x \mid \text{TM } \{x\} \text{ halts on input } x \}$

✓ **Theorem** K is not recursive

✓ **Theorem** $\overline{K}$ is not r.e.

Theorem HALT is not recursive

* K is a special case of HALT & K not recursive

→ $L_1 = K$, $L_2 =$ HALT. Assume M_2 always halts and accepts L_2. Construct M_1 for L_1

→ <u>M_1 on x:</u>

Run M_2 on $\langle x, x \rangle$. Accept iff M_2 accepts

The Halting Problem is Not Recursive

$K \stackrel{d}{=} \{ x \mid \text{TM } \{x\} \text{ halts on input } x \}$

✓ **Theorem** K is not recursive

✓ **Theorem** $\overline{K}$ is not r.e.

✓ **Theorem** HALT is not recursive

* K is a special case of HALT & K not recursive

→ $L_1 = K$, $L_2 = $ HALT. Assume M_2 always halts and accepts L_2. Construct M_1 for L_1

→ <u>M_1 on x:</u>

Run M_2 on $\langle x, x \rangle$. Accept iff M_2 accepts

Tips

(1.) Try obvious algorithms to see if you think language is recursive, re, or neither

(2.) To show L not r.e., sometimes it helps to work with $\bar{L}$

(ie. if $\bar{L}$ r.e., + $\bar{L}$ not recursive then L not r.e.)

(3) get reduction in correct direction.

many times constructed TM M_1 will ignore its own input