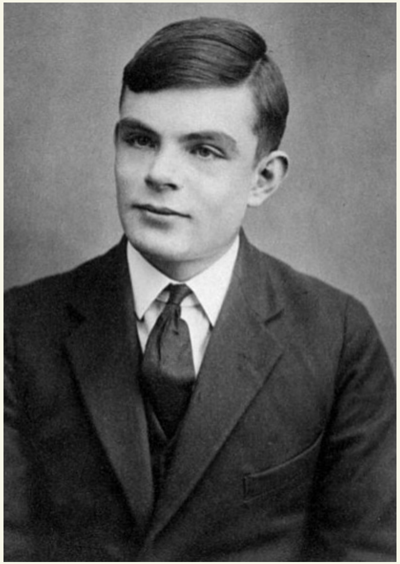
COMPUTABILITY, II

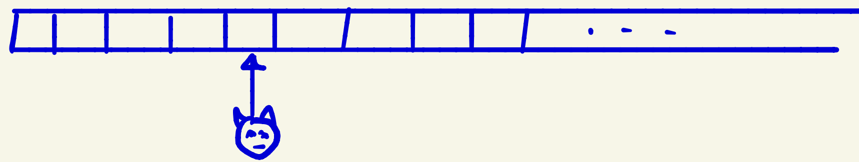
Turing Machines

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



1912 - 1954

Turing Machines:

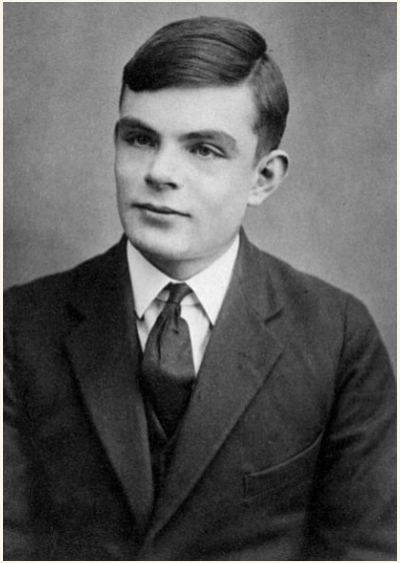


q_3

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

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 $\{0,1\}$

$\{0,1\}^*$ = all binary strings of finite length

$= \{\varepsilon, 0, 1, 00, 01, 10, 11, \ldots \}$

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

recognizable / semi-computable

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

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

computable / decidable

A language $L \subseteq \{0,1\}^*$ is recursive 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 recursive 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")

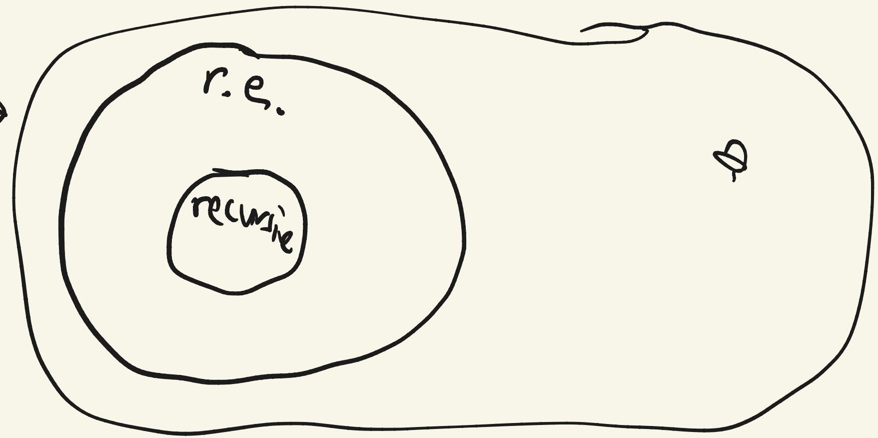
recursive also called decidable, computable

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 $f(x)$.

---

all $L \subseteq \{0,1\}^*$
all subsets of $\{0,1\}^*$



CLOSURE PROPERTIES

1. L recursive $\Rightarrow$ L r.e.

2. Total computable functions closed under composition:

$$f, g \text{ computable} \Rightarrow f \circ g = f(g(x)) \text{ is computable}$$

CLOSURE PROPERTIES

1. L recursive $\Rightarrow$ L r.e.

2. Total computable functions closed under composition:

   f, g computable $\Rightarrow$ $f \circ g = f(g(x))$ is computable

3. Closure of recursive languages 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

CLOSURE PROPERTIES

① L recursive $\Rightarrow$ L r.e.

② Total computable functions closed under composition:

f, g computable $\Rightarrow$ $f \circ g = f(g(x))$ is computable

③ Closure of recursive languages under $\cap, \cup, \neg$:

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

③' Closure of r.e. languages under $\cap, \cup$

L_1, L_2 r.e. $\Rightarrow$ $L_1 \cup L_2$, $L_1 \cap L_2$ recursive

use dovetailing re for $i = 1, \ldots$
run M_1 for i steps, M_2 for i steps

CLOSURE PROPERTIES

① L recursive $\Rightarrow$ L r.e.

② Total computable functions closed under composition:

f, g computable $\Rightarrow$ $f \circ g = f(g(x))$ is computable

③ Closure of recursive languages 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' Closure of r.e. languages under $\cap, \cup$

L_1, L_2 r.e. $\Rightarrow$ $L_1 \cup L_2$, $L_1 \cap L_2$ recursive

use dovetailing re for $i=1, \ldots$
run M_1 for i steps, M_2 for i steps

What about closure of r.e. under $\neg$?

CLOSURE PROPERTIES, cont'd

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

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

* <u>Note</u>: 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 $\bar{L} = \{x \mid x \text{ is not a legal encoding or } \{x\}_1 \text{ does not accept } 111\}$

$L \subseteq \{0,1\}^*$

$\{0,1\}^*$ = set of all strings over 0/1 of finite length

$\{\varepsilon, 0, 1, 00, 01, 10, 11, \ldots \ldots\}$

$\overline{L} = \{ y \in \{0,1\}^* \mid y \notin L \}$

CLOSURE PROPERTIES, cont'd

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

CLOSURE PROPERTIES, cont'd

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 aren't Recursively Enumerable!

Intuition:

Every TM M maps uniquely to a string in $\{0,1\}^*$

$$\{0,1\}^* = \{\varepsilon, 0, 1, 00, 01, 10, 11, 000, 001, \ldots\}$$

so the # of TMs is countable, and therefore so are the r.e. languages $L \subseteq \{0,1\}^*$

On the other hand, how large is the set of all languages?
i.e. the set of all subsets of $\{0,1\}^*$

This set is uncountable! (So Many more languages than r.e. languages)

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 $\Sigma = \{0,1\}$

  $\{x_1\}, \{x_2\}, \{x_3\}, \dots$

- Make a 2-way infinite (but countable) table

  rows correspond to $\{x_1\}, \{x_2\}, \dots$

  columns correspond to enumeration of

  encodings of Turing machines $x_1, x_2, \dots$

- Entry (i,j): 0 if $\{x_i\}$ accepts x_j

  1 otherwise

Many Languages are Not r.e.

all binary strings in $\{0,1\}^*$

	ε	0	1	00	01	10	11	
	x_1	x_2	x_3	x_4	x_5	x_6	x_7	...
M_1 $\{x_1\}$	0	1	1	0	1	0	0	...
M_2 $\{x_2\}$	0	0	1	1	0	1	1	
$\{x_3\}$	1	1	1	1	1	0	1	
⋮	1	1	0	0	0	0	1	
	0	0	0	0	1	1	1	
⋮	0	1	0	1	0	1	0	
	⋮							

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	1	0	1	0	0	$\cdots$
M_2	0	~~0~~ 1	1	1	0	1	1	
M_3	1	1	~~1~~ 0	1	1	0	1	
M_4	1	1	0	~~0~~ 1	0	0	1	
M_5	0	0	0	0	0	1	1	
$\vdots$	0	1	0	1	0	0	0	

Diagonal Language D

$$D = \{x_j \mid \{x_j\}_1 \text{ does not accept } x_j\}$$

Theorem D is not r.e.

Proof By construction: For all TMs M_i,

$\{x_i\}(x_i) \neq D(x_i)$ 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

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

Yellow language $= \{ x \mid \text{TM } \{x\}$ ~~acc~~ accepts input $x \}$

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

<u>Claim</u> HALT, K are both r.e.

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

The Halting Problem is Not Recursive

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

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

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

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

- 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

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

Theorem K is not recursive

Theorem $\overline{K}$ 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

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

→ M_1 on x:

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

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

→ M_1 on x:

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

SUMMARY SO FAR

1. We saw $D = \{x \mid \{x\}_1(x) \text{ does not accept}\}$

   is not r.e. by diagonalization

2. Using reductions we proved

   K, Halt are not recursive

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

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

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

Theorem. HALT, K are both r.e., Neither are recursive

The Halting Problem is Not Recursive

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

Theorem K is not recursive

If K recursive then D also recursive

Theorem Halt Not recursive

If Halt recursive then K recursive

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

$L = \{ x \mid \{x\} \text{ accepts at least one input} \}$

Enumerate all strings in $\{0,1\}^*$

$\{ \varepsilon, 0, 1, 00, 01, 10, 11, 000, \ldots \}$

$w_1 \quad w_2 \quad w_3$

Dovetail Procedure for L on input x:

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

For $j = 1, \ldots i$

Simulate $\{x\}$, on w_j for i steps

If any of the simulations accepts, HALT & accept

$L = \{x \mid \{x\} \text{ accepts at least one input}\}$

- L is r.e. (Dovetailing)
- L is Not recursive

$L_1 = K = \{y \mid \{y\}(y) \text{ halts}\}$

Assume $L_2 = L$ is recursive & Let M_2 be TM $\mathcal{L}(M_2) = L$
and M_2 always halts

M_1 on input y:

Construct encoding z of TM $\{z\}$ where
$\{z\}$ on input x: Ignores x & runs $\{y\}$ on y
and accepts x if $\{y\}(y)$ halts

Run M_2 on z and accept y iff $M_2(z)$ accepts

<u>Claim</u> $\mathcal{L}(M_1) = K$ and M_1 always halts

$y \in K \Rightarrow \{y\}(y)$ halts $\Rightarrow \{z\}$ accepts all inputs $\Rightarrow M_2(z) = 1 \Rightarrow M_1(y) = 1$

$y \notin K \Rightarrow \{y\}(y)$ doesn't halt $\Rightarrow \{z\}$ accepts No input $\Rightarrow M_2(z) \neq 1 \Rightarrow M_1(y) \neq 1$

Completeness

A language $A \subseteq \{0,1\}^*$ is r.e.-complete if

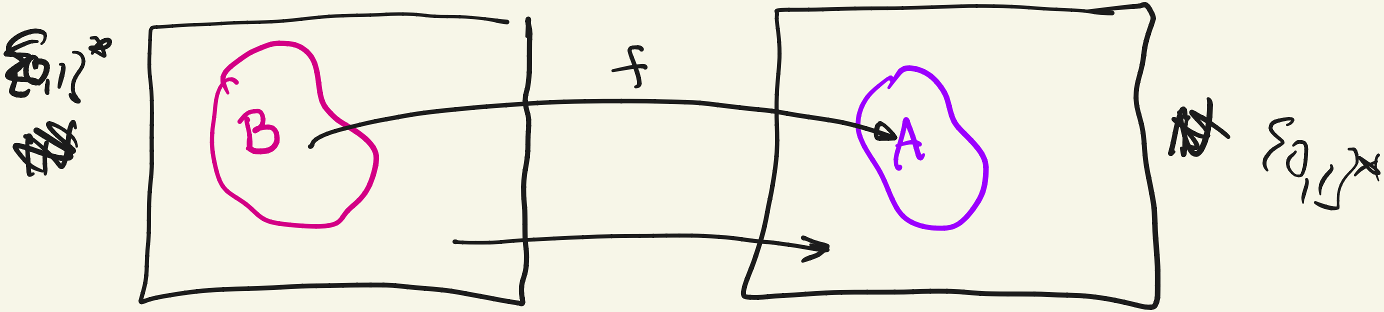
(1) A is r.e.

(2) $\forall B \subseteq \{0,1\}^*$, if B is r.e. then $B \leq_m A$

f is computable

B reduces to A

so if A is recursive then B recursive



Completeness

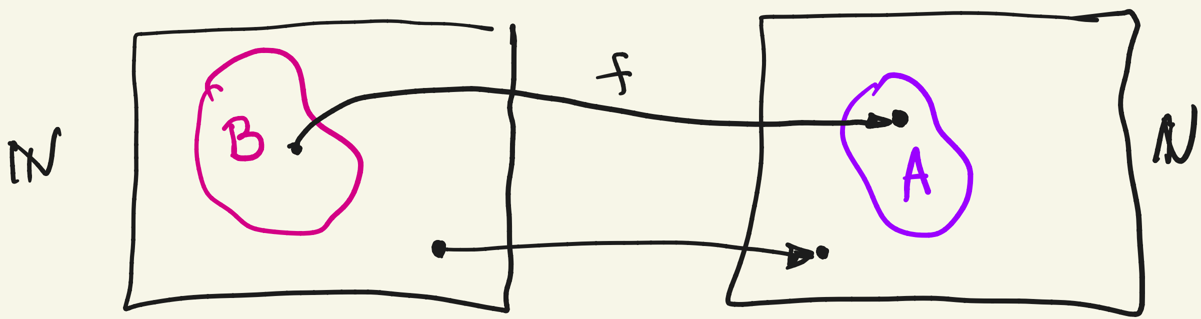
A set $A \subseteq \mathbb{N}$ is r.e.-complete if

(1) A is r.e.

(2) $\forall B \subseteq \mathbb{N}$, if B is r.e. then $B \leq_m A$

$\exists$ computable function $f : \mathbb{N} \Rightarrow \mathbb{N}$ such that

$\forall x \quad f(x) \in A \Leftrightarrow x \in B$



Hilbert's 10^{th} Problem (1900)

A diophantine equation is of the form $p(\vec{x}) = 0$ where p is a polynomial over variables $x_1, \ldots, x_n$ with integer coefficients

Ex $3x_1^5 x_2^3 + (x_1 + 1)^8 - x_7^{10} = 0$

$$\mathcal{L}_{DIOPH} = \{ \langle p \rangle \mid p \text{ has a solution over } \mathbb{N} \}$$

Theorem

$\mathcal{L}_{DIOPH}$ is r.e.-complete

An Equivalent Characterization of RE Sets/Languages

Let $f : \mathbb{N} \to \mathbb{N}$

Then $R_f \subseteq \mathbb{N} \times \mathbb{N}$

is the set of all pairs (x, y) such that $f(x) = y$

* **Theorem** f computable if and only if R_f is r.e.

An Equivalent Characterization of RE Sets

Let $f : \mathbb{N} \to \mathbb{N}$

Then $R_f \subseteq \mathbb{N} \times \mathbb{N}$

is the set of all pairs (x, y) such that $f(x) = y$

* **Theorem** f computable if and only if R_f is r.e.

Proof $\Rightarrow$: Suppose f computable.

TM for R_f on input (x, y):

Run TM computing f on x.

If it halts and outputs y then accept (x, y)

Otherwise reject (x, y)

An Equivalent Characterization of RE Sets

Let $f: \mathbb{N} \to \mathbb{N}$

Then $R_f \subseteq \mathbb{N} \times \mathbb{N}$

is the set of all pairs (x, y) such that $f(x) = y$

* <u>Theorem</u> f computable if and only if R_f is r.e.

<u>Proof</u> $\Leftarrow$: Let R_f be r.e. with TM M

<u>On x</u>: Enumerate all $\mathbb{N}$: $y_1, y_2, \ldots$

For $i = 1, 2, \ldots$

For all $j \leq i$: simulate M on (x, y_j) for i steps

If simulation accepts (x, y_j),

halt + output y_j

A second Characterization of RE Sets

A language $\overbrace{L \subseteq \{0,1\}^*}$

***Theorem** A relation $A \subseteq \mathbb{N}^k$ is r.e.
if and only if there is a recursive relation
recursive relation $R \subseteq \{0,1\}^* \times \{0,1\}^*$
$R \subseteq \mathbb{N}^{k+1}$ such that

$$\vec{x} \in A \iff \exists y\, R(\vec{x}, y) \quad \forall \vec{x} \in \mathbb{N}^n$$

Note We defined A to be r.e. iff there is a TM M such that $\forall \vec{x} \in \mathbb{N}^n \left(M(\langle x \rangle) \text{ accepts} \iff \vec{x} \in A \right)$

A language $L \subseteq \{0,1\}^*$ is r.e.

iff there exists a ~~[illegible]~~ relation $R \subseteq \{0,1\}^* \times \{0,1\}^*$

s.t. $\forall x \in \{0,1\}^*$

$$x \in L \text{ iff } \exists z \in \{0,1\}^* \ R(x,z)$$

where R is recursive

Ex. Let $L = \text{Halt} = \{\langle x,y\rangle \mid$ TM encoded by x halts on input $y\}$

Let $R(\langle x,y\rangle, \underset{\uparrow \text{ \# of steps}}{z}) = \begin{cases} 1/\text{accept} & \text{if } x \text{ halts on } y \text{ in exactly } z \text{ steps} \\ 0 & \text{o.w.} \end{cases}$

A Second Characterization of RE Sets

*** Theorem** A relation $A \subseteq \mathbb{N}^k$ is r.e. if and only if there is a recursive relation $R \subseteq \mathbb{N}^{k+1}$ such that

$$\vec{x} \in A \iff \exists y\, R(\vec{x}, y) \quad \forall \vec{x} \in \mathbb{N}^n$$

Proof sketch

$\Rightarrow$: Let A be r.e., $\mathcal{L}(M) = A$

$R(\vec{x}, y)$: view y as encoding of an $m \times m$ tableaux for some $m \in \mathbb{N}$

$(\vec{x}, y) \in R \iff M(\vec{x})$ halts in m steps and accepts and y is the $m \times m$ tableaux of $M(\vec{x})$

A second Characterization of RE Sets

*** Theorem** A relation $A \subseteq \mathbb{N}^k$ is r.e. if and only if there is a recursive relation $R \subseteq \mathbb{N}^{k+1}$ such that

$$\vec{x} \in A \iff \exists y \, R(\vec{x}, y) \quad \forall \vec{x} \in \mathbb{N}^n$$

Proof sketch

$\Leftarrow$ Let $R \subseteq \mathbb{N}^{k+1}$ be recursive relation such that $\vec{x} \in A \iff \exists y \, R(\vec{x}, y)$, + Let $\mathcal{L}(M) = R$

on input $\vec{x}$:

For $i = 1, 2, \ldots$

For $j = 1$ to i

Run M on $(\vec{x}, y_j)$

halt + accept if $M(\vec{x}, y_j)$ accepts

1. $L_1 = \{ x \mid$ TM encoded by x never moves head left on any input $\}$

2. $L_2 = \{ \langle x, y \rangle \mid$ TM x on input y never moves head left $\}$

$\overline{L_2} = \{ \langle x, y \rangle \mid x$ on input y moves head left at some point $\}$

r.e.

~~Not recursive~~ is recursive

L_2:

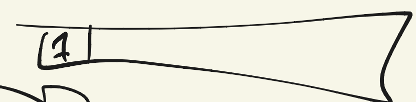
If state transitions all have head going R then we can ~~ass~~ halt + accept

~~y = 1 moves Rt~~

$(q_0, 1) \to (\quad R)$

$(q_0, 0 \quad R)$

$(q_0, \Delta) \to R$



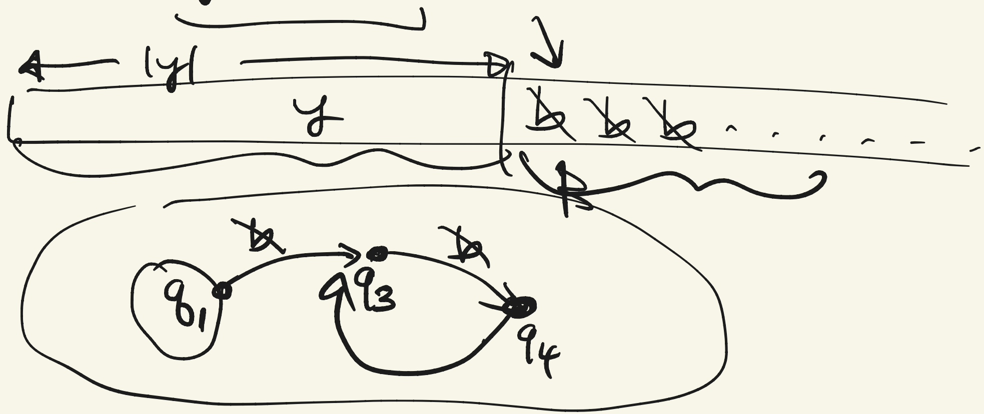
Harder case (of deciding L_2)

State transition table does have some transitions that move head to left

Machine x. Assume states of x are $q_0\, q_1\, q_2\, q_3\, q_4$

assume input/tape alphabet $= \{0, 1, b\}$

Let $y \in \{0,1\}^*$



$L = \{ x \mid \text{TM encoded by } x \text{ halts on } \geq 2 \text{ inputs in } \{0,1\}^* \}$

r.e.

$L' = \{ x \mid \text{TM encoded by } x \text{ halts on exactly } 2 \text{ inputs in } \{0,1\}^* \}$

?

$\overline{L} = \{ x \mid \text{TM } x \text{ halts on } \leq 1 \text{ inputs} \}$

Not re?

If $L, \overline{L}$ are both re. then both are recursive

So if L not recursive then $\overline{L}$ not r.e.