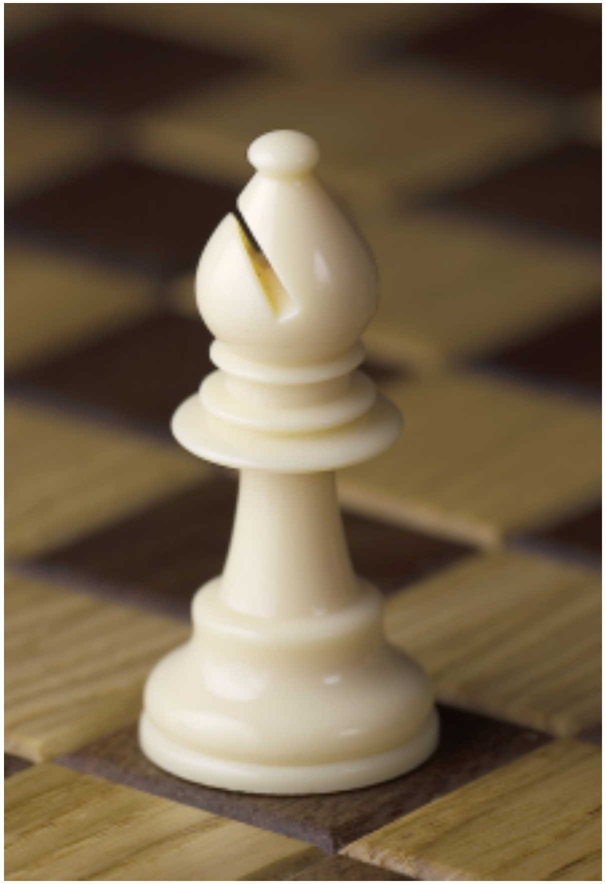
Diagonalization & Undecidability



Recap: Cantor's diagonalization

Theorem: no surjective function f : ℕ→ P(ℕ)

- This time: suppose such f exists ⇒ derive contradiction

$f(0) = \varnothing$

$f(1) = \mathbb{N}$

$f(2) =$

	0	1	2	3	4	...
0	0	0	0	0	0	--
1	1	1	1	1	1	--
2	0	1	0	1	0	---
3	1	1	0	0	1	
4	0	0	1	1	0	
⋮						⋱
D	1	0	1	1	1	...

translate D as a set $\subseteq \mathbb{N}$

$D = \{ i \mid i^{th} \text{ entry is } 1 \}$

$= \{ i \mid i^{th} \text{ diagonal of table is } 0 \}$

$= \boxed{\{ i \mid i \notin f(i) \}}$

"If i^{th} diagonal entry of table is 1, then i^{th} entry of D is 0"

What you need to know about Turing Machines (for today)

- On specific input, TM can either $\underbrace{\text{accept, reject}}_{\text{"halt"}}$, or run forever

- OK to write "simulate TM M on input x" (simulation may run forever)

  even if M, x determined at runtime

- Any string can be interpreted as encoding of a TM

What you need to know about Turing Machines (for today)

- A <u>language</u> is any set of strings
  - Yes-instances of a "decision problem"
- **Def:** S is <u>Turing-recognizable</u> ("recursively enumerable / r.e.") $\exists$ TM M:

  if $x \in S$ then M accepts x

  if $x \notin S$ then M $\underbrace{\text{doesn't accept}}_{\text{either rejects or runs forever}}$ x

- **Def:** S is <u>Turing-decidable</u> ("recursive")

  $\exists$ TM M:

  $$\left.\begin{array}{l}\text{if } x \in S \text{ then } M \text{ accepts } x \\ \text{if } x \notin S \text{ then } M \text{ rejects } x\end{array}\right\} M \text{ always halts}$$

Theorem (Turing 1936)

"encoding of a pair consisting of M, x"

- L_{acc} = { <M,x> | M is a TM that accepts x } is Turing-**recognizable** but not Turing-**decidable**

easy: on input ⟨M, x⟩, run M on x, see if it accepts

Theorem (Turing 1936)

- L_{acc} = { <M,x> | M is a TM that accepts x } is Turing-**recognizable** but not Turing-**decidable**

Idea: DIAGONALIZE: make a table that captures acceptance behavior of all TMs

all strings, in lexicographic order

$$\begin{array}{c|ccccc} & x_1 & x_2 & x_3 & x_4 & \cdots \\ \hline M_1 & & & & & \\ M_2 & & & & & \\ M_3 & & & & & \\ M_4 & & & & \boxed{} & \\ \vdots & & & & & \end{array}$$

entry i,j is

1 if M_i accepts x_j

0 otherwise

Theorem (Turing 1936)

	x_1	x_2	x_3	x_4	...
M_1	1	0	1	0	
M_2	0	0	0	0	
M_3	1	1	1	1	
M_4	0	0	1	1	...
⋮					

D = 0 1 0 0

D disagrees w/ every row of table

No TM has behavior that is described by D !

Goal: Assume L_{acc} is decidable, get contradiction by constructing a TM whose behavior is D

D(S):

interpret as TM

if S accepts S then reject
else accept

this step always halts if L_{acc} is decidable

Discussion

- L_{acc} is recognizable but not decidable

No algorithm ALWAYS halts
and ALWAYS gets
right answer to

"Does this M accept this x?"

An algo CAN get answer right <u>some</u>
<u>of</u> <u>the</u> <u>time</u>, though

Another Example:

- L_{halt} = { <M,x> | M is a TM that halts on x } is undecidable

Other ways to show undecidability?

- Formalize other logical paradoxes
  - Diagonalization works like **Russell's paradox**
    "Define D = { x | x $\notin$ x}; does D $\in$ D?"
  - Another approach works like **Berry's paradox**
    "x = smallest positive integer not definable in 8 words"

- Another approach: reductions [next time]
  - "if L were decidable, then I could use its algorithm to solve the halting problem"

Undecidability from Berry's paradox

- **Def:** $K(x)$ = length of shortest C program that outputs string x
- **Theorem:** $\{ \langle x,n \rangle \mid K(x) \leq n \}$ is undecidable

Undecidability from Berry's paradox