

# Computability Theory

Seen problems that are hard (e.g. EXPTIME)  
but some are impossible.

Key example: the Halting Problem

$\text{HALT}(f, x) :=$  will the function  $f$   
halt on input  $x$   
(i.e., no infinite loop)

First:  $\text{HALT}$  is NP-hard.

want  $3\text{SAT} \leq_p \text{HALT}$ .

Given instance  $x$  of  $3\text{SAT}$

Let  $f$  be a (slow) algorithm for  $3\text{SAT}$ .

def  $f'(x)$ :

if  $f(x) == \text{False}$ :

while 1: pass

else:

return

Then:  $\text{HALT}(f', x) = \text{True} \Leftrightarrow f(x) = \text{True} \Leftrightarrow$   
 $x$  a true instance.

$f'$  takes  $O(1)$  bits to describe  $\Rightarrow 3\text{SAT} \leq_p \text{HALT}$   
 $\Rightarrow \text{HALT}$  is NP-hard.

But we didn't use anything about  $\exists$ SAT other than that it is computable.

For any computable problem  $g$ ,  
 $g \leq_p \text{HALT}$  by same argument

[ $\Rightarrow$  Halt is  $\underbrace{\text{EXPTIME}}_{2^{2^n} \text{ time}} - \text{hard}$ ]

So HALT is maximally hard; but actually,  
it is *impossible*:

```
def g(x):  
    if HALT(x, x):  
        while 1: true  
    else: return
```

If HALT can be implemented, the program  $g()$  has some fixed size.

Q: does  $g(g)$  ever halt?

Paradox: YES  $\Rightarrow$  NO; NO  $\Rightarrow$  YES.

$\Rightarrow$  cannot compute HALT in finite time.

# Busy Beaver Numbers

What about the following way to solve HALT:

- Run program for  $L = 10^{10}$  steps
- if not terminated, output False

Correct if  $L$  is large enough: longer than time to compute  $f(x)$ .

If  $(f, x)$  takes  $n$  bits to describe, there are  $2^n$  inputs of length  $n$ .  
time until termination:

$T_1, \dots, T_{2^n}$  ← Some  $\infty$   
rest finite

Let  $B(n) := \max_{T_i < \infty} T_i$

def Halt  $(f, x)$ :  
 $L = B(|(f, x)|)$   
Run  $f(x)$  for  $L$  steps  
output True if  $f(x)$  terminated.

This solves the Halting problem  
The halting problem is not computable

$\Rightarrow B(n)$  is not computable.  
[Nor any upper bound on it.]

## UN provability

Of course, the halting problem is solvable for any finite input size:

Hard code

$$B(1000000) \leq L = 999,000,000$$

ridiculously long

to solve an inputs  $\leq 1000000$  bits long

But... how can you know if you've written enough 9's?

A: provably impossible to tell.

Gödel's 1<sup>st</sup> incompleteness Theorem:

No consistent set of axioms can prove their own consistency

[ "consistent" means cannot prove contradiction ]

Modern math is built on ZFC axioms.

$\Rightarrow$  impossible to prove ZFC consistent.

[ Even hope! if it is possible, modern math is broken. ]

def Find Contradiction():

For all  $s \in \{0, 1\}^*$ :

check if  $s$  is valid ZFC proof of contradiction

if so, return.

Halt(Find Contradiction) = False  $\Leftrightarrow$  ZFC consistent

$\Rightarrow$  impossible to prove Find Contradiction doesn't halt

$\Rightarrow$   $B(|\text{Find Contradiction}|)$  cannot be proven.

Q: how big is  $|\text{Find Contradiction}|$ ?

$S(n) = \max \#$  steps any  $n$ -state Turing machine takes on blank binary tape before halting

$S(n)$  behaves like  $B(n)$ , but more formally specified  
[so we can evaluate specific  $n$ ]

$S(2) = 6, \quad S(3) = 21, \quad S(4) = 107$

$S(5) = 47$  million, probably.  
[assuming  $\sim 15$  Turing machines run forever]

$S(6) \gg 10^{36000}$   
 $S(7) \gg 10^{10^{10^{10^6}}}$  ← Surely vast underestimates  
|  
|

$S(n)$  uncomputable in general

$S(\text{Find Contradiction})$  computable (it's a fixed constant)  
but not provably correct.

Scott Aaronson & Udema:  $|\text{Find Contradiction}| < 8000$   
more recent:  $< 2000$

$S(2000)$ : no upper bound can be proven in principle.

$S(741)$ : upper bound proves (hard part of) Riemann hypothesis.

# Models of Computation

Turing machines

Church Numerals / lambda calculus

Theorem:

Functions computable by Turing  
= Functions computable by Church

Church-Turing Thesis: <sup>(conjecture)</sup>

= functions computable by any "physical process"  
or "human following algorithm"

**Extended** Church-Turing Thesis:

P is the same in all processes

[Probably false: Quantum computers (BQP)  $\neq$  P ?]