PS7 due today
PS8 due next Monday, Apr 7
Coming soon: PS9, PRR10

Class 19: Uncomputability

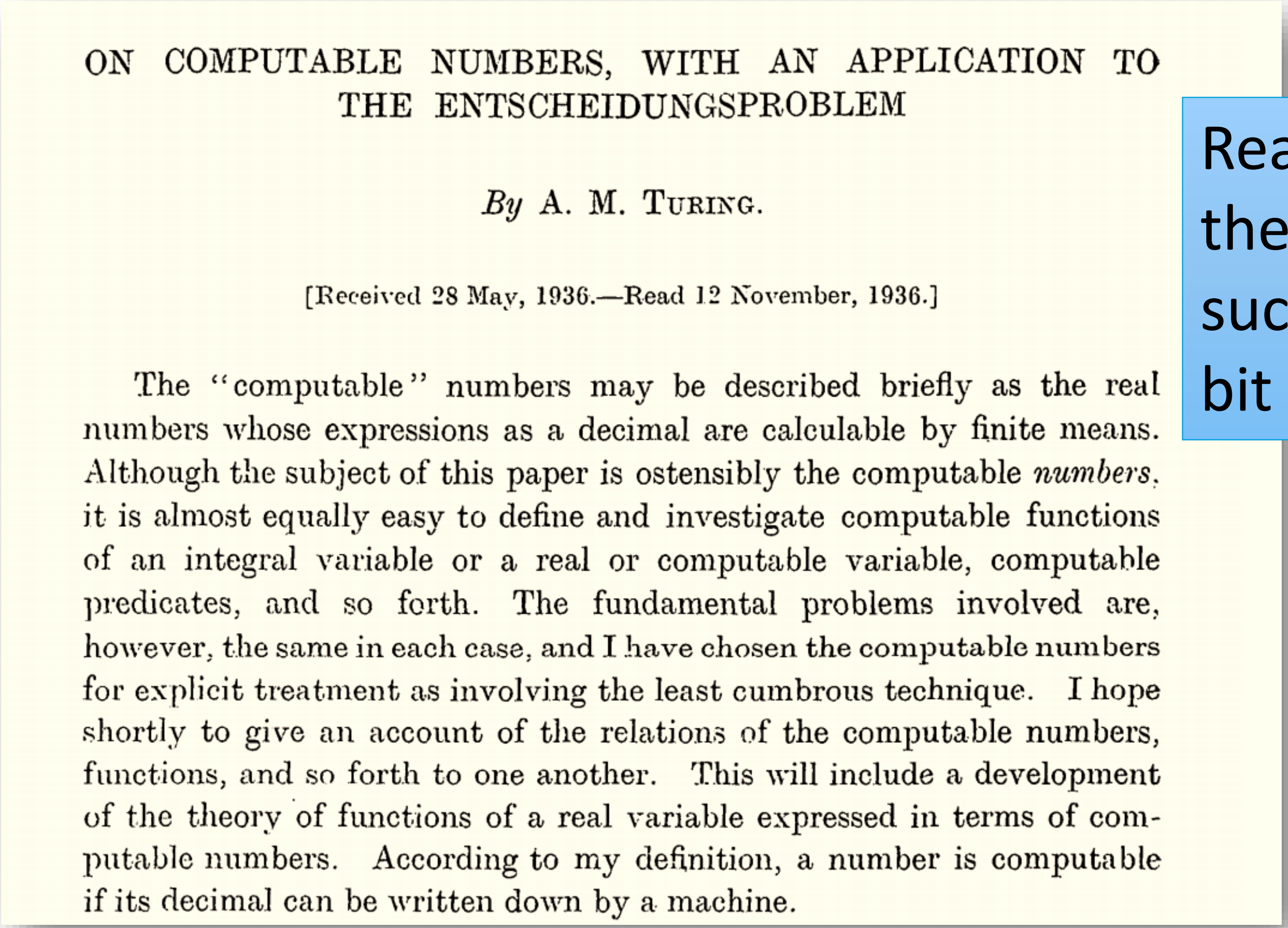
University of Virginia
cs3120: DMT2
Wei-Kai Lin

Photo credit

More Turing Machines

https://morphett.info/turing/turing.html

Recap: Computable numbers



ON COMPUTABLE NUMBERS, WITH AN APPLICATION TO THE ENTSCHEIDUNGSPROBLEM

By A. M. Turing.

[Received 28 May, 1936.—Read 12 November, 1936.]

The "computable" numbers may be described briefly as the real numbers whose expressions as a decimal are calculable by finite means. Although the subject of this paper is ostensibly the computable *numbers*, it is almost equally easy to define and investigate computable functions of an integral variable or a real or computable variable, computable predicates, and so forth. The fundamental problems involved are, however, the same in each case, and I have chosen the computable numbers for explicit treatment as involving the least cumbrous technique. I hope shortly to give an account of the relations of the computable numbers, functions, and so forth to one another. This will include a development of the theory of functions of a real variable expressed in terms of computable numbers. According to my definition, a number is computable if its decimal can be written down by a machine.

Real number r is **computable** if there exists a Turing machine M such that $M(n)$ outputs r to the nth bit precision for all natural number n.

Are there any ***un****computable* numbers?

How many *computable* numbers are there?

The "computable" numbers may be described briefly as the real numbers whose expressions as a decimal are calculable by finite means.

$\exists$ TM M

st $M(n) = r|n$ bit

How many Turing Machines are there?

A *Turing Machine,* is defined by (Σ, k, δ):
$\boldsymbol{k} \in \mathbb{N}$: a finite number of states
Σ: finite set of symbols, $\Sigma \supseteq \{0, 1, \rhd, \emptyset\}$
$\boldsymbol{\delta}$: $[k] \times \Sigma \rightarrow [k] \times \Sigma \times \{\mathbf{L}, \mathbf{R}, \mathbf{S}, \mathbf{H}\}$

$\rightarrow w \in \{0,1\}^*$

$|\text{CompNum}| \leq |\{TM\}| \leq |\{0,1\}^*|$

$||$

$|\mathbb{R}| > |\mathbb{N}|$

$\exists\, r \in \mathbb{R}$ not computable

Representing Turing Machines

How to represent a Turing machine as a binary string?

binary

A *Turing Machine,* is defined by (Σ, k, δ):

$\boldsymbol{k} \in \mathbb{N}$: a finite number of states

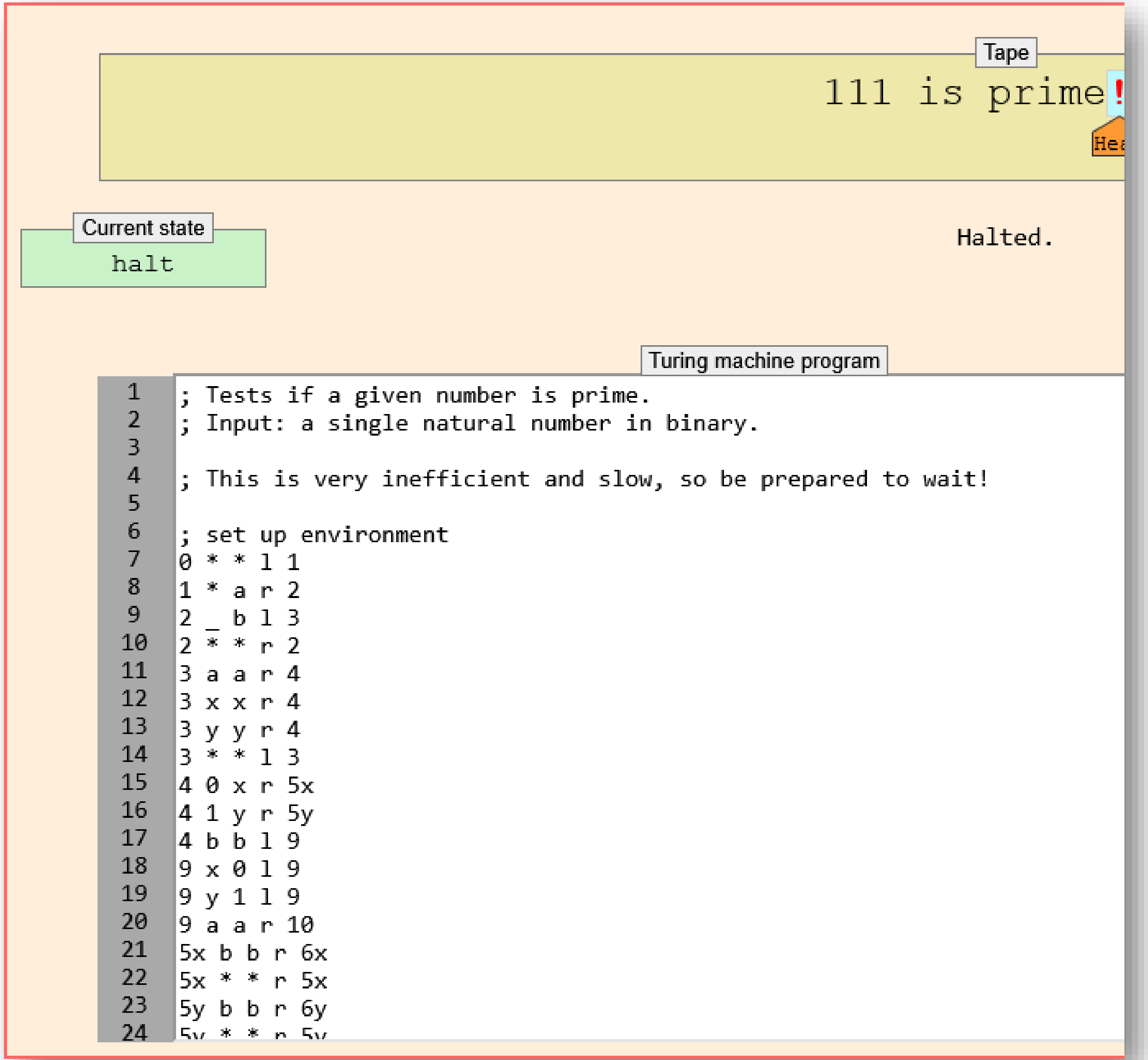
Σ : alphabet – finite set of symbols

$$\Sigma \supseteq \{0, 1, \triangleright, \emptyset\}$$

$\boldsymbol{\delta}$: transition function

$$\boldsymbol{\delta}: [k] \times \Sigma \rightarrow [k] \times \Sigma \times \{\mathbf{L}, \mathbf{R}, \mathbf{S}, \mathbf{H}\}$$

- k
- Σ
- δ



Some Numbers are Uncomputable!

$$|TMs| = |\text{finite binary strings}| = |\mathbb{N}|$$

$$|\mathbb{R}| = |pow(\mathbb{N})| > |\mathbb{N}|$$

Is there an *interesting* number that cannot be computed?

Boolean $F : \{0,1\}^* \to \{0,1\}$

$r \in [0,1)$, $r = 0.b_0 b_1 b_2 \dots$

$F(0)$ $F(1)$

Are there *functions* that cannot be computed by any Turing Machine?

Computable Functions

Definition:

A Boolean function $F: \{0,1\}^* \rightarrow \{0,1\}$ is **computable** if and only if there exists a Turing machine M such that for all $x \in \{0,1\}^*$, $M(x) = F(x)$.

Are there “interesting” uncomputable functions?

Turing Machines to/from bit strings

For any $w \in \{0,1\}^*$,
let TM_w be the Turing machine M such that
w represents M; $= (k, \Sigma, \delta) \leftarrow TM_w$
if no such Turing machine,
define TM_w=NIL.

A *Turing Machine,* is defined by (Σ, k, δ):

$\boldsymbol{k} \in \mathbb{N}$: a finite number of states
Σ : alphabet – finite set of symbols
$$\Sigma \supseteq \{0, 1, \rhd, \emptyset\}$$
$\boldsymbol{\delta}$: transition function
$$\boldsymbol{\delta}: [k] \times \Sigma \rightarrow [k] \times \Sigma \times \{\mathbf{L}, \mathbf{R}, \mathbf{S}, \mathbf{H}\}$$

Proving Interesting Functions are not Computable

1. Show that we can build a TM that simulates any other TM (a "Universal Turing Machine")
2. Construct a TM using the Universal Turing Machine as a component that leads to a contradiction.

Like many proof strategies we have seen (e.g., proving uncountability), once we have *one* (uncomputable function), we can use it to more easily prove new functions are also uncomputable.

Universal Machines

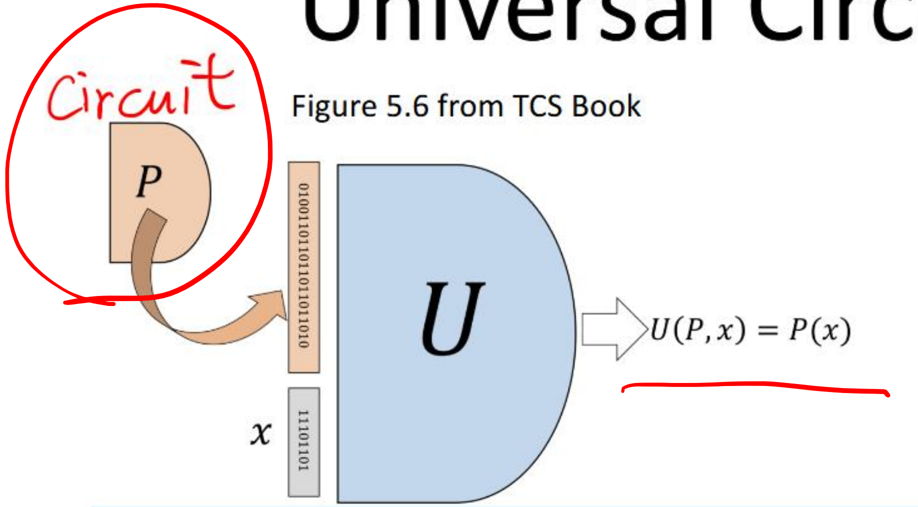
https://dori-mic.org/

From Classes 11

Recall from Boolean Circuits

Universal Circuit/Program

Figure 5.6 from TCS Book



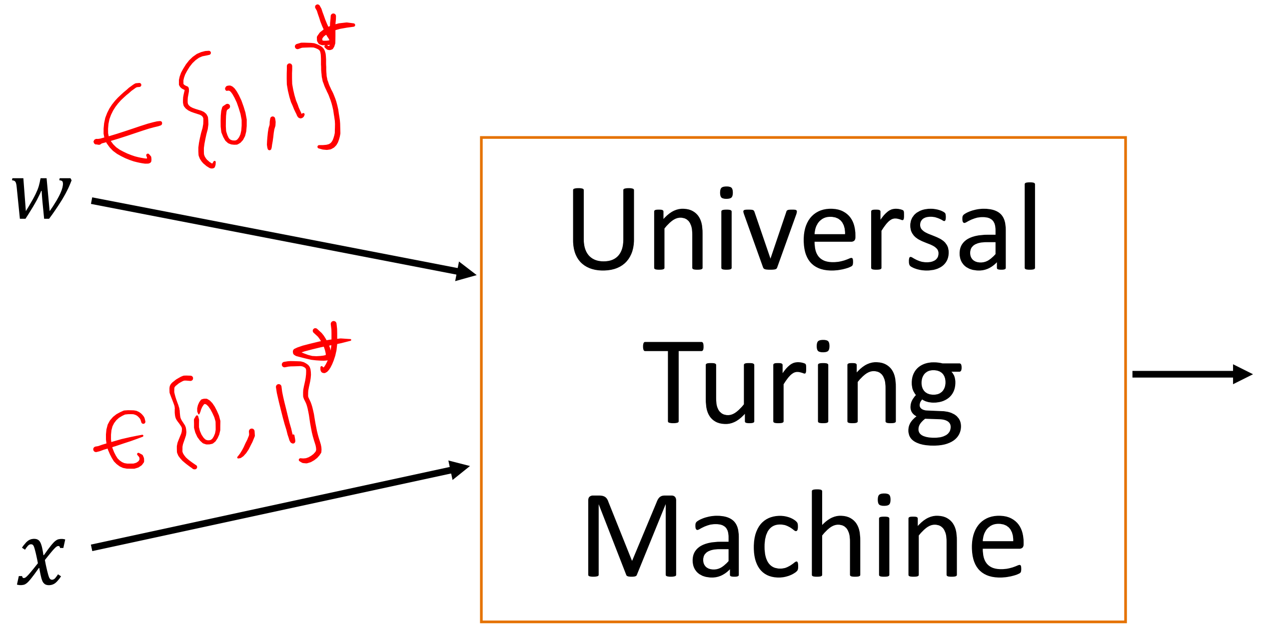
program U takes a program description P and input x as its input, and "simulates" running P on x:

$$U(P,x) = P(x)$$

Theorem 5.9 (Bounded Universality of NAND-CIRC programs)

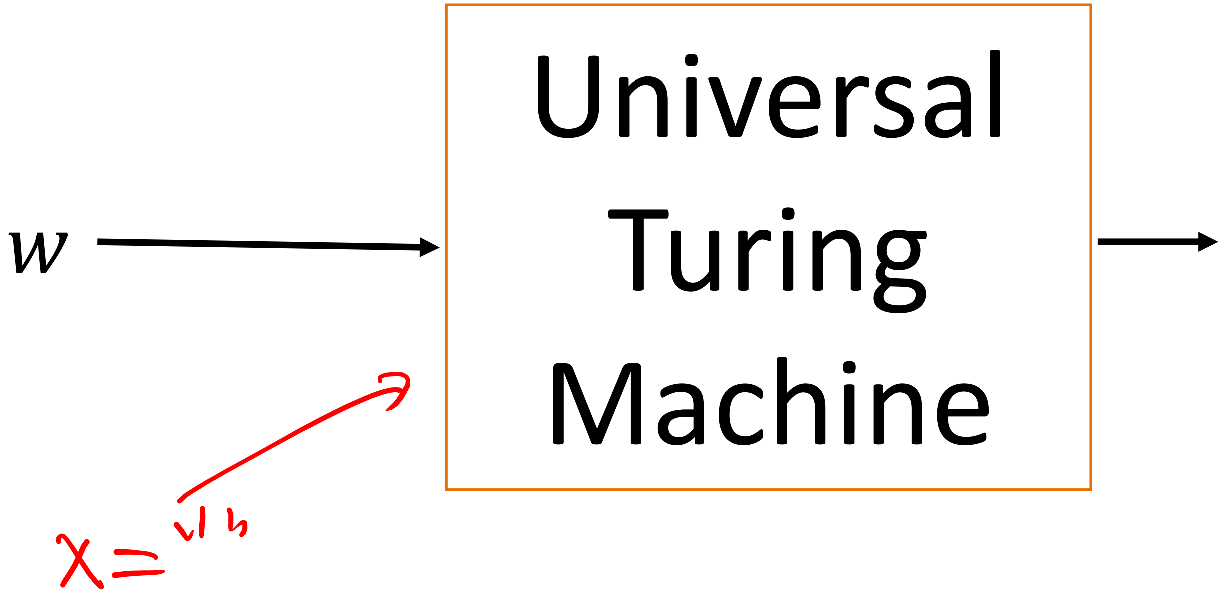
For every $s, n, m \in \mathbb{N}$ with $s \geq m$ there is a NAND-CIRC program $U_{s,n,m}$ that computes the function $EVAL_{s,n,m}$.

Is there a “Universal Turing Machine”?



$$\to \begin{cases} TM_w(x) & \text{if } w \text{ represents TM} \\ 0 & \text{otherwise} \end{cases}$$

Is there a "Universal (no input) Turing Machine"?



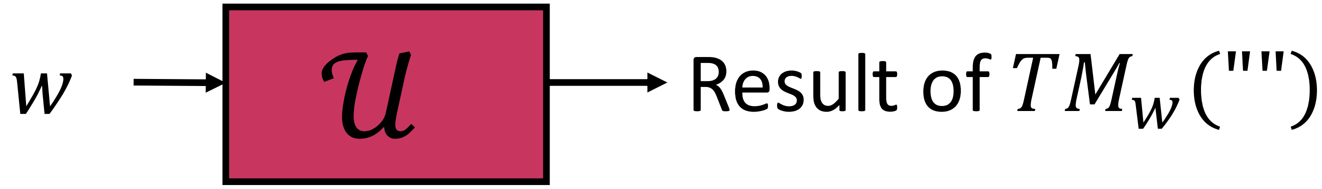
$$\begin{cases} TM_w("") & \text{if } w \text{ represents TM} \\ 0 & \text{otherwise} \end{cases}$$

6. *The universal computing machine.*

It is possible to invent a single machine which can be used to compute any computable sequence. If this machine $\mathcal{U}$ is supplied with a tape on the beginning of which is written the S.D of some computing machine $\mathcal{M}$,

then $\mathcal{U}$ will compute the same sequence as $\mathcal{M}$. In this section I explain in outline the behaviour of the machine. The next section is devoted to giving the complete table for $\mathcal{U}$.

What do we need to build $\mathcal{U}$?



$$\mathcal{U}(w) = TM_w("").$$

7. *Detailed description of the universal machine.*

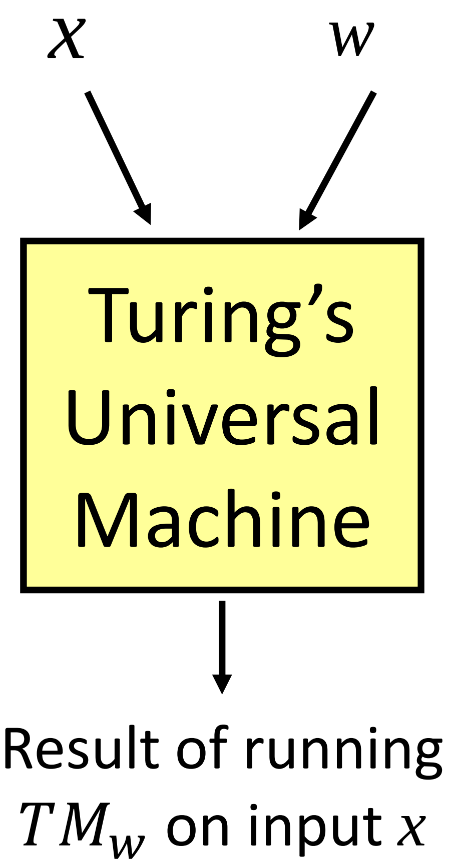
A table is given below of the behaviour of this universal machine. The m-configurations of which the machine is capable are all those occurring in the first and last columns of the table, together with all those which occur when we write out the unabbreviated tables of those which appear in the table in the form of m-functions. *E.g.*, $\mathfrak{e}(\mathfrak{anf})$ appears in the table and is an m-function. Its unabbreviated table is (see p. 239)

$\mathfrak{e}(\mathfrak{anf})$	ə	R	$\mathfrak{e}_1(\mathfrak{anf})$
	not ə	L	$\mathfrak{e}(\mathfrak{anf})$
$\mathfrak{e}_1(\mathfrak{anf})$	Any	R, E, R	$\mathfrak{e}_1(\mathfrak{anf})$
	None		$\mathfrak{anf}$

Consequently $\mathfrak{e}_1(\mathfrak{anf})$ is an m-configuration of $\mathfrak{U}$.

When $\mathfrak{U}$ is ready to start work the tape running through it bears on it the symbol ə on an F-square and again ə on the next E-square; after this, on F-squares only, comes the S.D of the machine followed by a double colon "::" (a single symbol, on an F-square). The S.D consists of a number of instructions, separated by semi-colons.

Each instruction consists of five consecutive parts



[Turing 1936]

The table for $\mathfrak{U}$.

$\mathfrak{b}$			$\mathfrak{f}(\mathfrak{b}_1, \mathfrak{b}_1, ::)$	$\mathfrak{b}$. The machine prints $:DA$ on the F-squares after $:: \rightarrow \mathfrak{anf}$.
$\mathfrak{b}_1$		$R, R, P:, R, R, PD, R, R, PA$	$\mathfrak{anf}$	
$\mathfrak{anf}$			$\mathfrak{g}(\mathfrak{anf}_1, :)$	$\mathfrak{anf}$. The machine marks the configuration in the last complete configuration with y. $\rightarrow \mathfrak{fom}$.
$\mathfrak{anf}_1$			$\mathfrak{con}(\mathfrak{fom}, y)$	
$\mathfrak{fom}$	;	R, Pz, L	$\mathfrak{con}(\mathfrak{fmp}, x)$	$\mathfrak{fom}$. The machine finds the last semi-colon not marked with z. It marks this semi-colon with z and the configuration following it with x.
	z	L, L	$\mathfrak{fom}$	
	not z nor ;	L	$\mathfrak{fom}$	
$\mathfrak{fmp}$			$\mathfrak{cpe}\big(\mathfrak{e}(\mathfrak{fom}, x, y), \mathfrak{sim}, x, y\big)$	$\mathfrak{fmp}$. The machine compares the sequences marked x and y. It erases all letters x and y. $\rightarrow \mathfrak{sim}$ if they are alike. Otherwise $\rightarrow \mathfrak{fom}$.

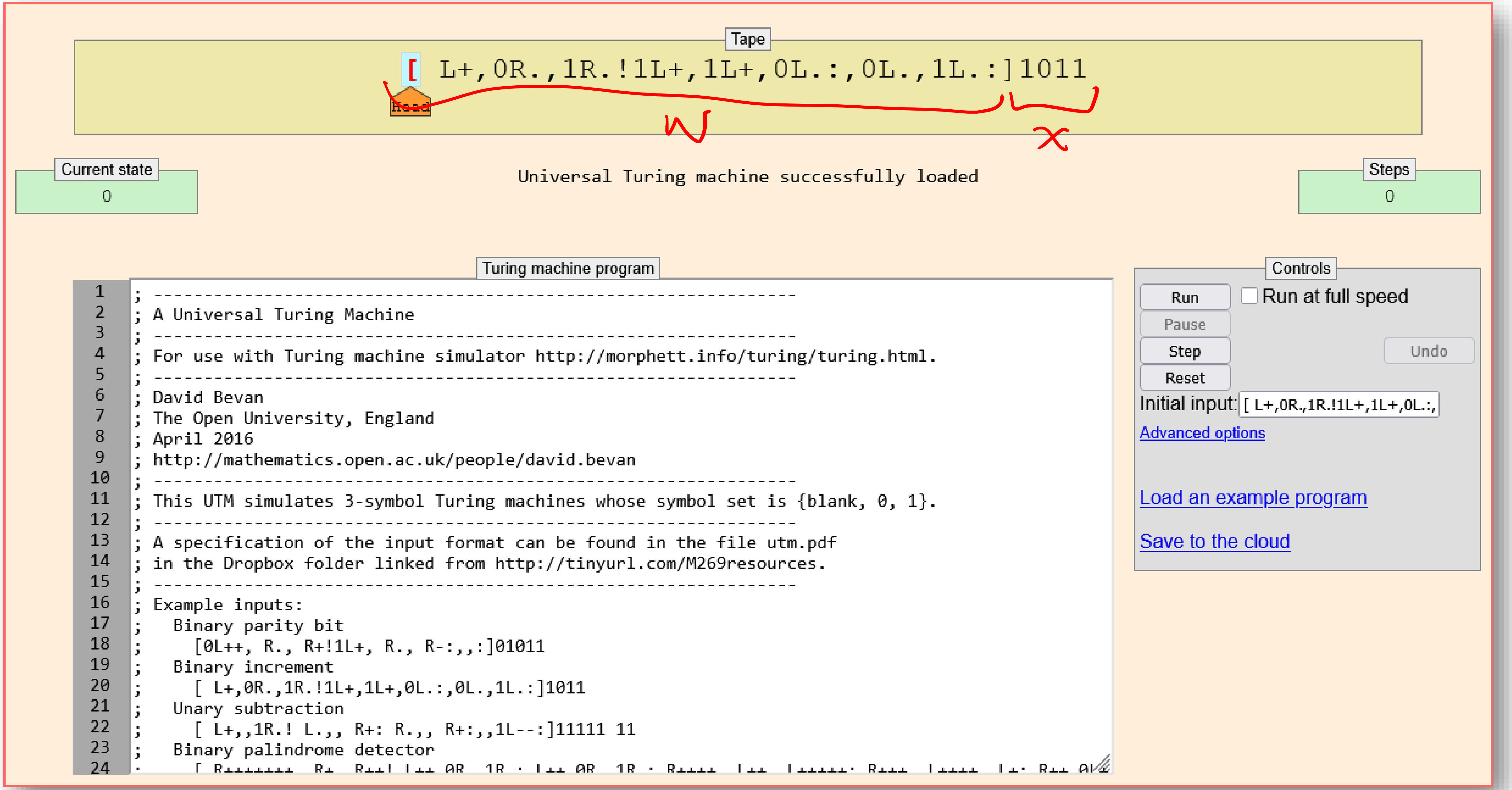
PRR9: Universal Turing Machine

Q1.5

2 Points

Load and read the example "Universal Turing machine." Find and run the "Binary increment" example as input. How many steps are used? Choose the smallest that is correct.

- 100
- 1000
- 10000
- It does not stop.


Tape
[L+,0R.,1R.!1L+,1L+,0L.:,0L.,1L.:]1011
Head
w
x
Current state
0
Universal Turing machine successfully loaded
Steps
0
Turing machine program
; --
; A Universal Turing Machine
; --
; For use with Turing machine simulator http://morphett.info/turing/turing.html.
; --
; David Bevan
; The Open University, England
; April 2016
; http://mathematics.open.ac.uk/people/david.bevan
; --
; This UTM simulates 3-symbol Turing machines whose symbol set is {blank, 0, 1}.
; --
; A specification of the input format can be found in the file utm.pdf
; in the Dropbox folder linked from http://tinyurl.com/M269resources.
; --
; Example inputs:
;   Binary parity bit
;   [0L++, R., R+!1L+, R., R-:,,:]01011
;   Binary increment
;   [L+,0R.,1R.!1L+,1L+,0L.:,0L.,1L.:]1011
;   Unary subtraction
;   [L+,,1R.! L.,, R+: R.,, R+:,,1L--:]11111 11
;   Binary palindrome detector
Controls
Run
Run at full speed
Pause
Step
Undo
Reset
Initial input: [L+,0R.,1R.!1L+,1L+,0L.:,
Advanced options
Load an example program
Save to the cloud

(Interesting) Uncomputable Functions

ACCEPTS Function

A Turing Machine, $M = (\Sigma, k, \delta)$, **accepts** a string, x, if $M(x) = 1$.

$$ACCEPTS(w, x) = \begin{cases} 1, & \text{if } TM_w \text{ accepts } x \\ 0, & \text{otherwise} \end{cases}$$

w ~~is~~ NOT TM

$TM_w(x) = 0$

$TM_w(x) = 011$

$TM_w(x) = \bot$

Is $ACCEPTS$ computable?

Computing ACCEPTS?

A Turing Machine, $M = (\Sigma, k, \delta)$, **accepts** a string, x, if $M(x) = 1$.

$$ACCEPTS(w, x) = \begin{cases} 1, & \text{if } TM_w \text{ accepts } x \\ 0, & \text{otherwise} \end{cases}$$

w, x  Result of running TM_w on x

$TM_w(x) = 1$

$M_{ACCEPTS}(w, x) =$ **if** $(\mathcal{U}(w, x) = 1)$ output 1
else output 0

Computing ACCEPTS?

A Turing Machine, $M = (\Sigma, k, \delta)$, **accepts** a string, x, if $M(x) = 1$.

$$ACCEPTS(w, x) = \begin{cases} 1, & \text{if } TM_w \text{ accepts } x \\ 0, & \text{otherwise} \end{cases}$$

$M_{ACCEPTS}$ **does not compute** *ACCEPTS*, since it may not finish: if $\mathcal{U}(w, x)$ does not terminate execution, $M_{ACCEPTS}$ does not output correctly!

$$M_{ACCEPTS}(w, x) = \textbf{if } (\mathcal{U}(w, x) = 1) \text{ output } 1 \\ \textbf{else} \text{ output } 0$$

Note: this does not **prove** that ACCEPTS is uncomputable, since it doesn't show there isn't some other way to compute it (eg, without using $\mathcal{U}(w, x)$)

ACCEPTS is Uncomputable

How to reach a contradiction?

Proof by contradiction:

Assume some TM, $M_A(w, x)$, computes *ACCEPTS*.

Table of Machines

Input, x

TM, w	ε	0	1	00	01	10	11	000	001	010	…
ε (w not TM)											…
0											…
1	✓	✓	N	✓				✓			
00											
01	✓		✓				✓				
10				✓							
11											
000	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
…											

✓ in (w, x) means $TM_w(x) = 1$

Not the actual table (of course!)

What's this table?

Table of Machines

Input, x

TM, w	ε	0	1	00	01	10	11	000	001	010	…
ε											…
0											…
1	✓	✓		✓				✓			
00											
01	✓		✓				✓				
10				✓							
11											
000	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
…											

✓ in (w, x) means $TM_w(x) = 1$

$ACCEPTS(w, x)$

What's the TM that computes "negation of the diagonal"?

ACCEPTS is Uncomputable

Definition:

A Boolean function $F: \{0,1\}^* \to \{0,1\}$ is **computable** if and only if there exists a Turing machine M such that for all $x \in \{0,1\}^*$, $M(x) = F(x)$.

Proof by contradiction:

Assume some TM, $M_A(w, x)$, computes *ACCEPTS*.

There must be some string, w_A such that, $M_A = TM_{w_A}$.

Let's define a new TM,

$$M_D(x) = \text{NOT}\left(\mathcal{U}(w_A, (x, x))\right).$$

$= M_A(x, x)$

$= ACCEPTS(x, x)$

$\to \bot$?

What's $\mathcal{U}(w_A, (x, x))$?

What's NOT(…)?

Does M_D finish?

Yes

ACCEPTS is Uncomputable

$$ACCEPTS(w, x) = \begin{cases} 1, & \text{if } TM_w \text{ accepts } x \\ 0, & \text{otherwise} \end{cases}$$

Proof by contradiction:

Assume some TM, $M_A(w, x)$, computes *ACCEPTS*.

There must be some string, w_A such that, $M_A = TM_{w_A}$.

Let's define a new TM, $M_D(x) = \text{NOT}\left(\mathcal{U}(w_A, (x, x))\right)$

We have $= NOT\left(M_A(x, x)\right) = NOT(ACCEPTS(x, x))$

There exists w_D such that $M_D = TM_{w_D}$. Consider $M_D(w_D)$.

What's $M_D(w_D)$?

Option 1: $M_D(w_D) = 0$.

$0 = M_D(w_D) = TM_{w_D}(w_D)$

$ACCEPTS(w_D, w_D) = 0$

$NOT(ACCEPTS(w_D, w_D)) = 1 = M_D(w_D)$

ACCEPTS is Uncomputable

$$ACCEPTS(w,x) = \begin{cases} 1, & \text{if } TM_w \text{ accepts } x \\ 0, & \text{otherwise} \end{cases}$$

Proof by contradiction:

Assume some TM, $M_A(w,x)$, computes *ACCEPTS*.

There must be some string, w_A such that, M_A = TM_{w_A}.

Let's define a new TM, $M_D(x) = \text{NOT}\left(\mathcal{U}\left(w_A, (x,x)\right)\right)$

We have = $NOT\left(M_A(x,x)\right) = NOT(ACCEPTS(x,x))$

There exists w_D such that $M_D = TM_{w_D}$. Consider $M_D(w_D)$.

Option 1: $M_D(w_D) = 0$.

$$0 = M_D(w_D) = TM_{w_D}(w_D)$$
$$ACCEPTS(w_D, w_D) = 0$$
$$M_D(w_D) = NOT\left(ACCEPTS(w_D, w_D)\right) = 1$$

Option 2: $M_D(w_D) = 1$.

$$1 = M_D(w_D) = TM_{w_D}(w_D)$$
$$ACCEPTS(w_D, w_D) = 1$$
$$M_D(w_D) = NOT\left(ACCEPTS(w_D, w_D)\right) = 0$$

ACCEPTS is Uncomputable

$$ACCEPTS(w, x) = \begin{cases} 1, & \text{if } TM_w \text{ accepts } x \\ 0, & \text{otherwise} \end{cases}$$

Proof by contradiction:

Assume some TM, $M_A(w, x)$, computes *ACCEPTS*.

There must be some string, w_A such that, $M_A = TM_{w_A}$.

Let's define a new TM, $M_D(x) = \text{NOT}\left(\mathcal{U}(w_A, (x, x))\right)$

$= NOT\left(M_A(x, x)\right) = NOT(ACCEPTS(x, x))$

There exists x such that $M_D = TM_x$. Consider $M_D(x)$.

Option 1: $M_D(w_D) = 0.$

$0 = M_D(w_D) = TM_{w_D}(w_D)$

$ACCEPTS(w_D, w_D) = 0$

$M_D(w_D) = NOT\left(ACCEPTS(w_D, w_D)\right) = 1$

Option 2: $M_D(w_D) = 1.$

$1 = M_D(w_D) = TM_{w_D}(w_D)$

$ACCEPTS(w_D, w_D) = 1$

$M_D(w_D) = NOT\left(ACCEPTS(w_D, w_D)\right) = 0$

Contradiction: D must not exist!
But, if we have M_A, we can construct D.
So, M_A must not exist. Therefore, *ACCEPTS* is uncomputable.

A Most Learned Video about Computability!

https://www.youtube.com/watch?v=xx5t5ps-bwc

Ali G Function

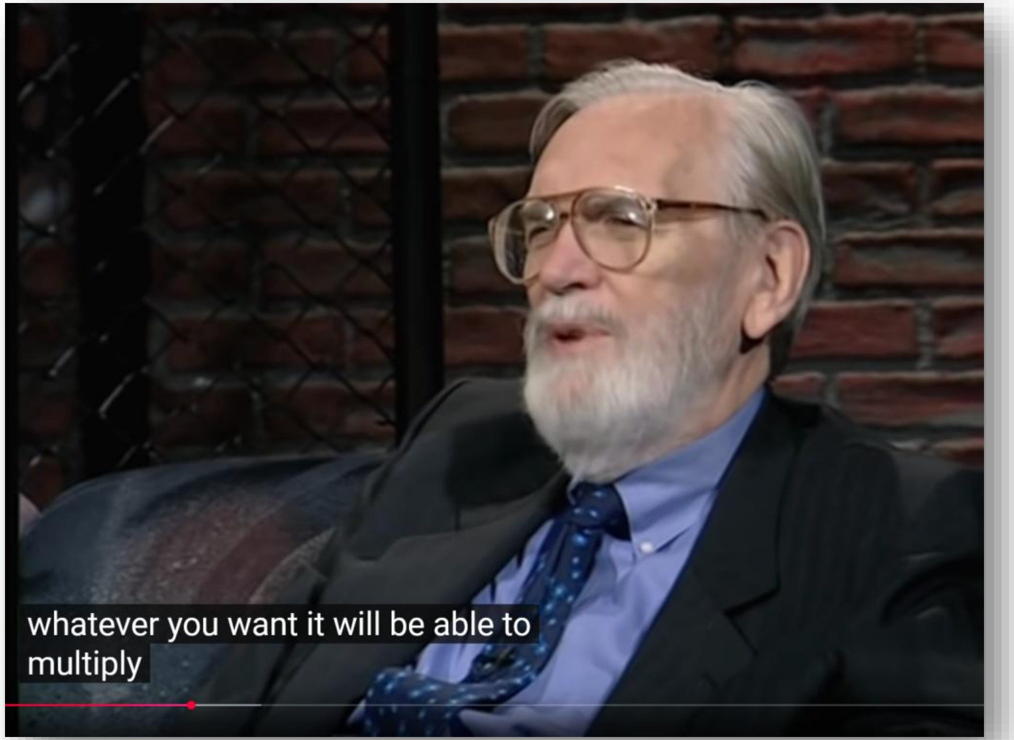
"Will there computers ever be able to work out what 99999999…9 multiplied by 99999999…9"

Ali G Function

$$\forall x, y \in \{9\}^*, AliG(x, y) := x \times y$$

Is there a Turing Machine M_{AliG} that computes $AliG$?

YES



Computability ≠ Practical Solvability

$$x, y \in \{9\}^*, AliG(x, y) := x \times y$$

What does ***computability*** *of Ali G function mean?*

Exist Algo multiplies

2^{200} > # atom Earth

Practical: $9^{(2^{200})}$

Computability ≠ Practical Solvability

$$ACCEPTS(w, x) = \begin{cases} 1, & \text{if } TM_w \text{ accepts } x \\ 0, & \text{otherwise} \end{cases}$$

*What does **uncomputability** of ACCEPTS mean?*

Computability ≠ Practical Solvability

$$ACCEPTS(w, x) = \begin{cases} 1, & \text{if } TM_w \text{ accepts } x \\ 0, & \text{otherwise} \end{cases}$$

*What does **uncomputability** of ACCEPTS mean?*

Can prove Accepts (w, x) for some (w, x).

There is no Turing Machine that, for ***all*** inputs $w, x \in \{0, 1\}^*$ can output the value of $ACCEPTS(w, x)$. Any TM must, for at least one input $w, x \in \{0, 1\}^*$, either output the **wrong value** or **run forever**.

Exercise: prove that any TM must, for at least **two** inputs, either output the wrong value or run forever

More Uncomputable Functions

$$ACCEPTS(w, x) = \begin{cases} 1, & \text{if } TM_w \text{ accepts } x \\ 0, & \text{otherwise} \end{cases}$$

$$HALTS(w, x) = \begin{cases} 1, & \text{if } TM_w \text{ terminates on } x \\ 0, & \text{otherwise} \end{cases}$$

Is *HALTS* computable?

Strategy: look for an infinite loop in w for x

More Uncomputable Functions

$$ACCEPTS(w,x) = \begin{cases} 1, & \text{if } TM_w \text{ accepts } x \\ 0, & \text{otherwise} \end{cases}$$

$$HALTS(w,x) = \begin{cases} 1, & \text{if } TM_w \text{ terminates on } x \\ 0, & \text{otherwise} \end{cases}$$

Can we show *HALTS* is computable?

~~**Strategy:** look for an infinite loop in w~~

Strategy: run $TM_w(x)$ and look for repeated states

repeat a lot

? how many

Exercise: write a TM that increments a counter infinitely

Halting Problem

$$ACCEPTS(w, x) = \begin{cases} 1, & \text{if } TM_w \text{ accepts } x \\ 0, & \text{otherwise} \end{cases}$$

$$HALTS(w, x) = \begin{cases} 1, & \text{if } TM_w \text{ terminates on } x \\ 0, & \text{otherwise} \end{cases}$$

How can we show *HALTS* is uncomputable?

Strategy 1: show we can use a machine that computes it to produce a contradiction (like we did to show ACCEPTS is uncomputable)

Halting Problem

$$ACCEPTS(w, x) = \begin{cases} 1, & \text{if } TM_w \text{ accepts } x \\ 0, & \text{otherwise} \end{cases}$$

$$HALTS(w, x) = \begin{cases} 1, & \text{if } TM_w \text{ terminates on } x \\ 0, & \text{otherwise} \end{cases}$$

How can we show *HALTS* is uncomputable?

Strategy 1: show we can use a machine that computes it to produce a contradiction (like we did to show ACCEPTS is uncomputable)

Strategy 2: show we can use a machine that computes it to produce a machine that computes ACCEPTS.