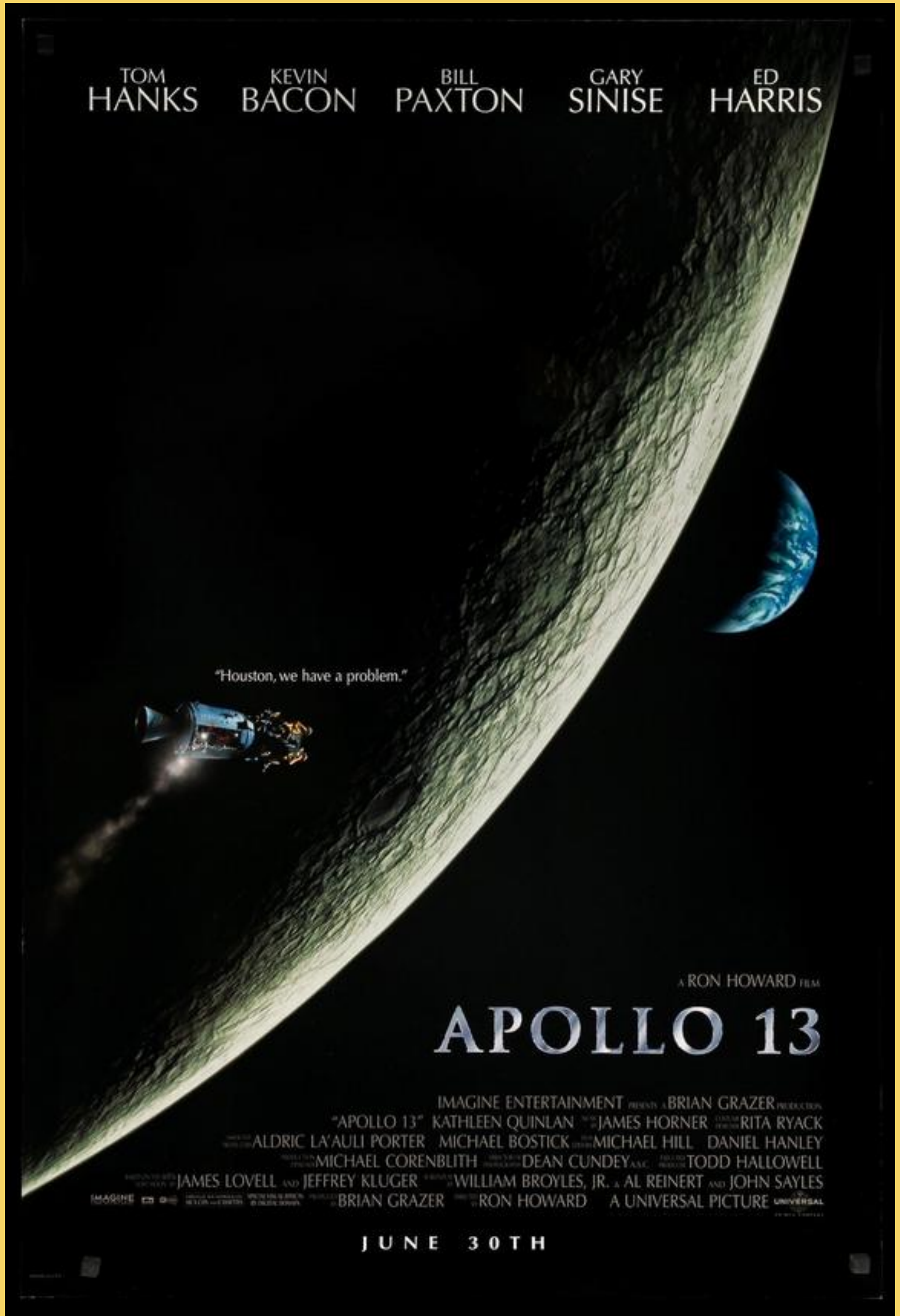


PS8 due next Monday, Apr 7
Coming today: PS9, PRR10

Class 20:
Uncomputability and Reductions

University of Virginia
cs3120: DMT2
Wei-Kai Lin

[Photo credit](#)

Recap: Computable Functions

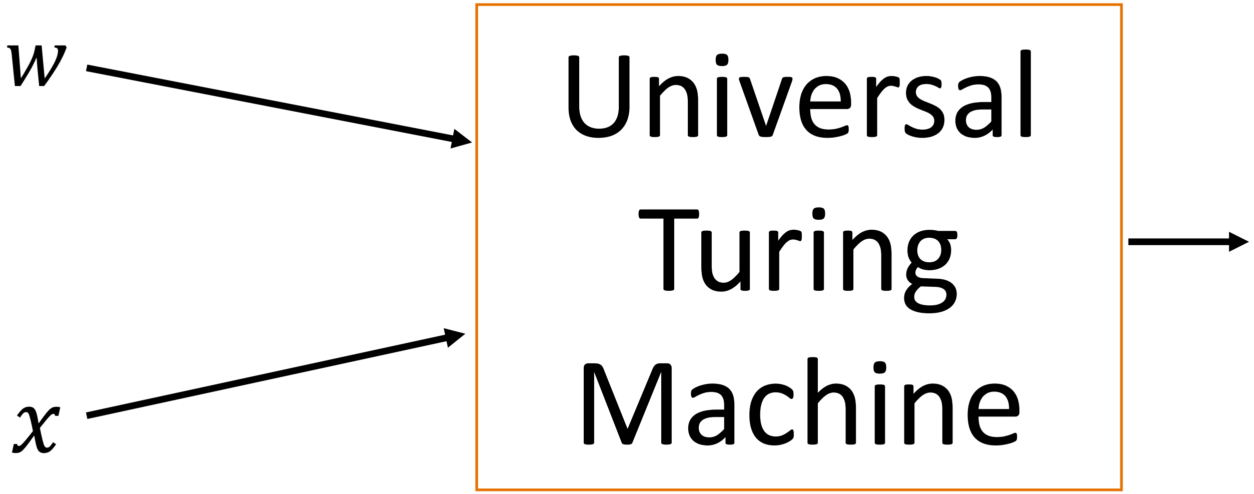
Definition:
A Boolean function $F: \{0,1\}^* \rightarrow \{0,1\}$ is computable if and only if there exists a Turing machine M such that for all $x \in \{0,1\}^*$, $M(x) = F(x)$.

Definition:

A language $L \subseteq \{0,1\}^*$ is **computable** if and only if there exists a Turing machine M such that for all $x \in \{0,1\}^*$,

$$M(x) = \begin{cases} 0 & if\ x \notin L \\ 1 & if\ x \in L. \end{cases}$$

Recap: Universal Turing Machine



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

Recap: $ACCEPTS$ is uncomputable

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

Boolean function:

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

(A statement)
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**.

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"?

Efficiency of Turing Machines

Halting Problem

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

— w not TM

— $TM_w(x) = \bot$

Can we show *HALTS* is computable?

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

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

Exercise: write a TM that increments a counter infinitely

Prove 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)

Prove by Reduction

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

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

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.

Prove by Reduction

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

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

If there exists M_B that solves B, then we have M_A that solves A

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

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

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

Proving *HALTS* is Uncomputable

Assume $HALTS$ is computable. for contradiction

By the assumption (and definition of computable), there exists some TM M_{HALTS} that computes $HALTS$.

We can use M_{HALTS} to build a machine that decides computes $ACCEPTS(w,x)$:

$M_A(w, x)$

1. Call $M_{HALTS}(w,x) = HALTS(w,x)$
   Let b be the output
2. Output $U(w,x)$ if $b=1$
   Output 0 o.w. $b=0$

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

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

Proving *HALTS* is Uncomputable

Assume $HALTS$ is computable.

By the assumption (and definition of computable), there exists some TM M_{HALTS} that computes $HALTS$.

We can use M_{HALTS} to build a machine that decides $ACCEPTS(w,x)$:

$M_{ACCEPTS}\,(w,x)$: $h = \mathcal{U}(M_{HALTS}, (w,x))$

if h: return $\mathcal{U}(w,x)$

else: return **False**

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

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

Proving *HALTS* is Uncomputable

Assume $HALTS$ is computable. By the assumption (and definition of computable), there exists some TM M_{HALTS} that computes $HALTS$. We can use M_{HALTS} to build a machine that decides $ACCEPTS(w,x)$:

$M_{ACCEPTS}(w,x)$: $h = \mathcal{U}(M_{HALTS}, (w,x))$
if h: return $\mathcal{U}(w,x)$
else: return **False**

Thus, since we know $M_{ACCEPTS}$ does not exist, but if we had M_{HALTS} we could build it, we have a contradiction! This proves that M_{HALTS} must not exist which means $HALTS$ is not computable.

Running Time and Busy Beavers

PRR9:

Q1.3

2 Points

Load and look at the example "Binary addition." How many step are used to perform any 8-bit addition? Choose the smallest that is correct.

- ( ) 150
- ( ) 200
- (•) 250
- ( ) 300

Running time:

The number of steps.

Tape

110001000

Head

Current state

halt

Halted.

Steps

215

Turing machine program

```
; Binary addition - adds two binary numbers
; Input: two binary numbers, separated by a single space, eg '100 1110'

0 _ _ r 1
0 * * r 0
1 _ _ l 2
1 * * r 1
2 0 _ l 3x
2 1 _ l 3y
2 _ _ l 7
3x _ _ l 4x
3x * * l 3x
3y _ _ l 4y
3y * * l 3y
4x 0 x r 0
4x 1 y r 0
4x _ x r 0
4x * * l 4x    ; skip the x/y's
4y 0 1 * 5
4y 1 0 l 4y
4y _ 1 * 5
4y * * l 4y    ; skip the x/y's
5 x x l 6
5 y y l 6
```

Controls

Run | Run at full speed

Pause

Step | Undo

Reset

Initial input: 11011001 10101111

Advanced options

Load an example program

Save to the cloud

Busy Beaver

$BB(n)$: the TM such that among all n-state Turing machines that halt on no input, $BB(n)$ runs longest

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

For any n, a Turing Machine (Σ, n, δ) consists of n state.

How many TMs consists of n states?

$\# \delta : (n \times 4 \times 4)^{4n}$

$\#$ TM n states

$|\Sigma| = 4$

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

For any n, a Turing Machine (Σ, n, δ) consists of n state.

Busy Beaver

$BB(n)$: the TM such that among all n-state Turing machines that halt on no input, $BB(n)$ runs longest

Huh?

Finitely many $O(n)^{O(n)}$

Can we compute BB(n)?

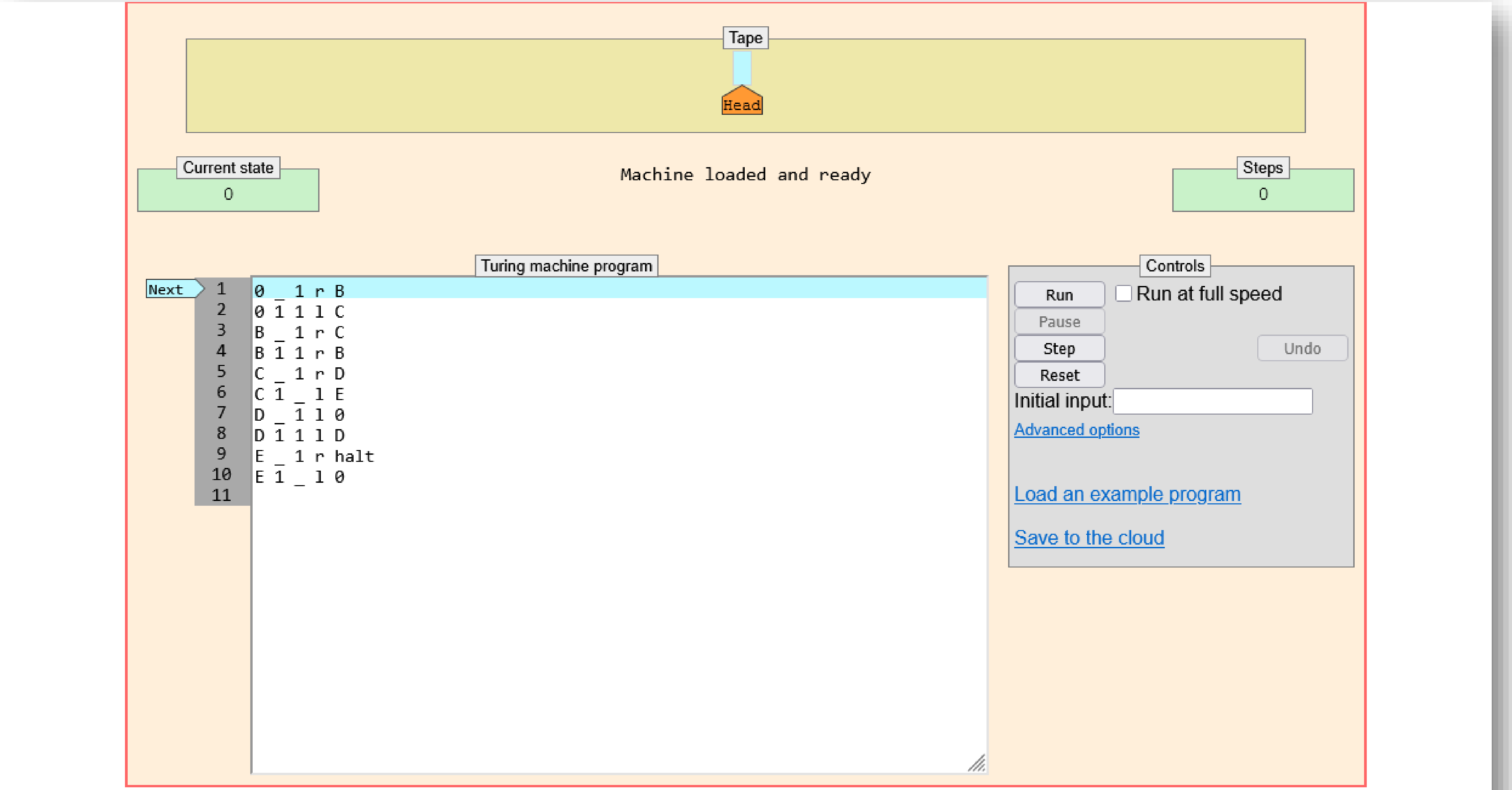
$BB(n): \mathbb{N} \to \mathbb{N}$

$\{0,1\}^* \to \mathbb{N}$

~~$\times \{0,1\}$~~

BB(n)	HALTS
NO	

Nemo, photo by Sankalpa



http://morphett.info/turing/turing.html?94a288fb9c40906d7e6face4bc422ece
https://bbchallenge.org/1RB1LC_1RC1RB_1RD0LE_1LA1LD_1RZ0LA

Quanta magazine

https://www.quantamagazine.org/amateur-mathematicians-find-fifth-busy-beaver-turing-machine-20240702/

Today, the team declared victory. They've finally verified the true value of a number called BB(5), which quantifies just how busy that fifth beaver is. They obtained the result — 47,176,870 — using a piece of software called the Coq proof assistant, which certifies that mathematical proofs are free of errors.

Church-Turing Thesis

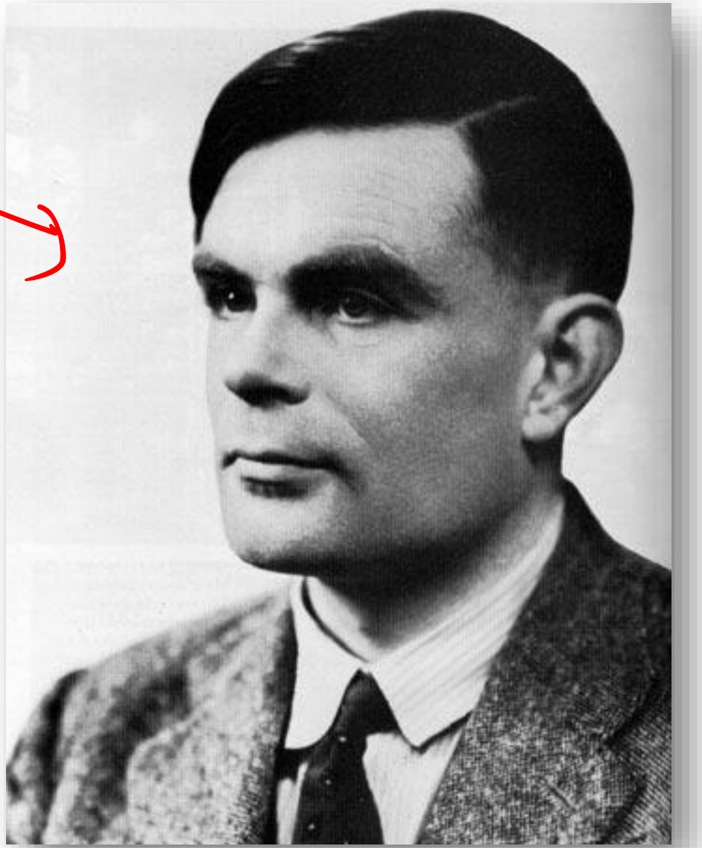
Church-Turing Thesis

Alonzo Church, 1903-1995

A Turing Machine (or Lambda Calculus) can simulate ***any*** "mechanical computer".

$\lambda x.(x+x)$

$\lambda y.y$



Alan Turing, 1912-1954

Is this a statement that can be proven true or false?

Church-Turing Thesis

9. *The extent of the computable numbers.*

No attempt has yet been made to show that the "computable" numbers include all numbers which would naturally be regarded as computable. All arguments which can be given are bound to be, fundamentally, appeals to intuition, and for this reason rather unsatisfactory mathematically. The real question at issue is "What are the possible processes which can be carried out in computing a number?"

The arguments which I shall use are of three kinds.

(*a*) A direct appeal to intuition.

(*b*) A proof of the equivalence of two definitions (in case the new definition has a greater intuitive appeal).

(*c*) Giving examples of large classes of numbers which are computable.

Alan Turing, 1912-1954

III. This may be regarded as a modification of I or as a corollary of II.

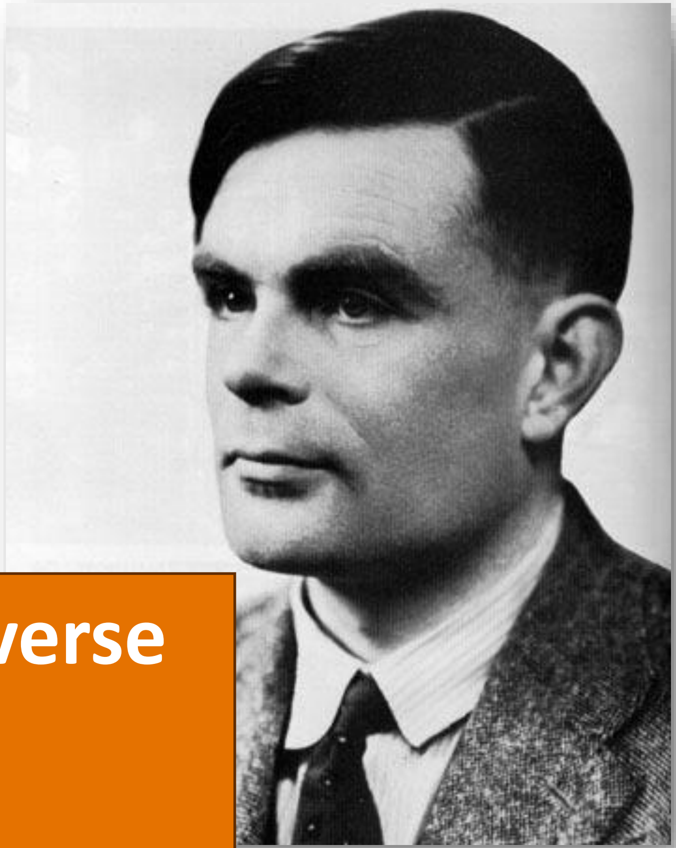
We suppose, as in I, that the computation is carried out on a tape; but we avoid introducing the "state of mind" by considering a more physical and definite counterpart of it. It is always possible for the computer to break off from his work, to go away and forget all about it, and later to come back and go on with it. If he does this he must leave a note of instructions (written in some standard form) explaining how the work is to be continued. This note is the counterpart of the "state of mind". We will suppose that the computer works in such a desultory manner that he never does more than one step at a sitting. The note of instructions must enable him to carry out one step and write the next note. Thus the state of progress of the computation at any stage is completely determined by the note of

Church-Turing Thesis

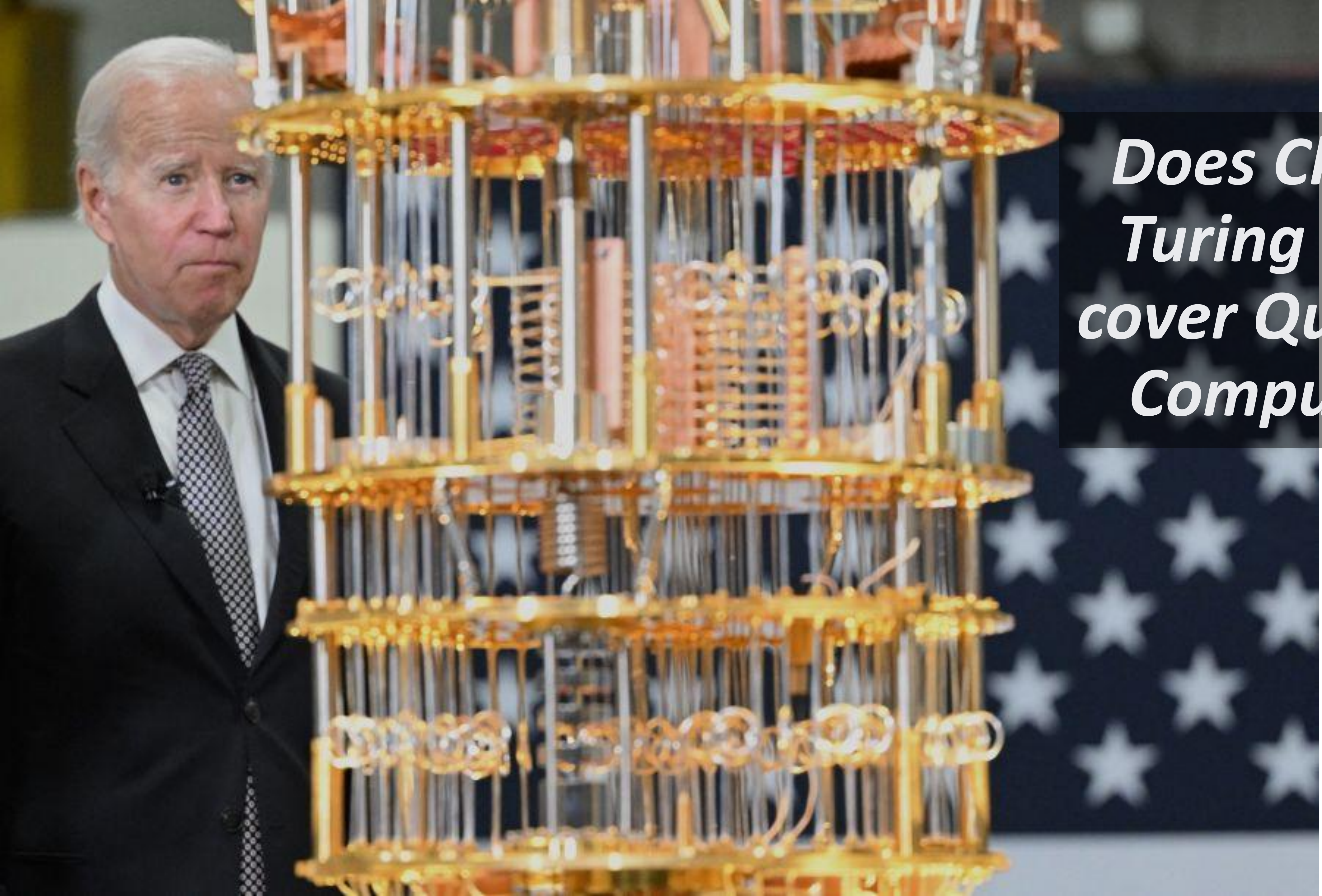
Alonzo Church, 1903-1995

A Turing Machine (or Lambda Calculus) can simulate ***any*** “mechanical computer”.

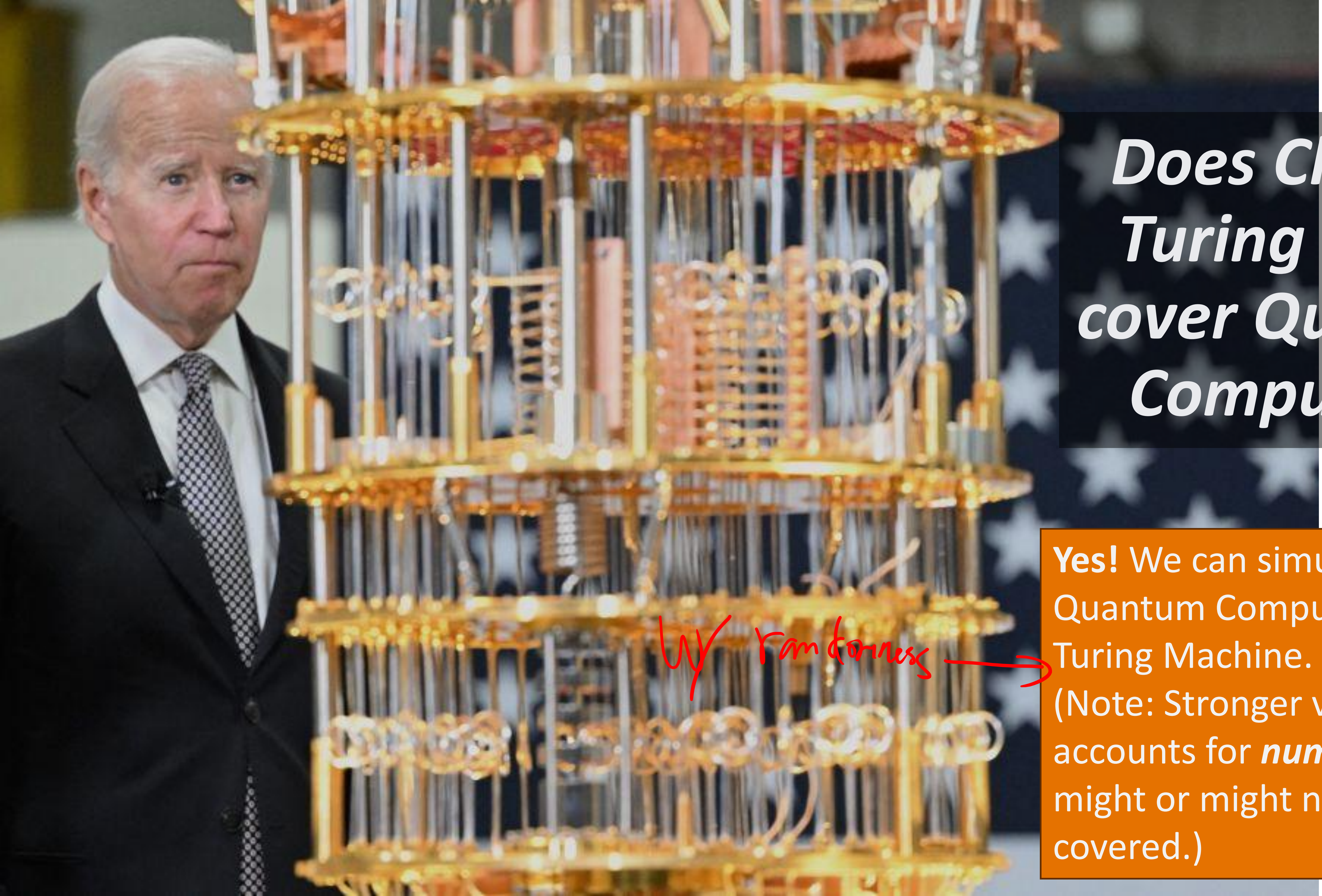
All conceivable computers in this universe (or any imaginable one) are no more powerful than a Turing Machine.



Alan Turing, 1912-1954



Does Church-Turing thesis cover Quantum Computers?



Does Church-Turing thesis cover Quantum Computers?

Yes! We can simulate a Quantum Computer with a Turing Machine.
(Note: Stronger version that accounts for ***number of steps*** might or might not be covered.)

Reductions

How we proved uncomputability

FACT

ACCEPT = $\{(w,x) : TM_w \text{ accepts } x\}$ not computable

Proof: …

HALT = $\{(w,x) : TM_w \text{ halts on } x\}$

Proof: if HALT computable → ACCEPT computable

Three step process:

1. **Assume** M_H decides HALT
2. **Construct** $M_A(w,x) \coloneqq$ If $M_H(w,x)$ return $U(w,x)$ else return 0
3. **Prove** that M_A would decide ACCEPT (assuming M_H decides HALT)

Reductions at abstract level

Reducing "task" A to "task" B, denoted by $A \leq_R B$
Showing that solving A is easier than or equal to B

Corollary:
1. If B is easy → A is easy too
2. If A is hard → B is hard too.

How the proof looks like:
1. Assume that algorithm M_B solves B
2. Design algorithm M_A (that uses M_B as subroutine)
3. Prove that M_A solves A if M_B solves B

How we proved uncomputability

ACCEPT = $\{(w, x) : M_w \text{ accepts } x\}$

Proof: …

HALT = $\{(w, x) : M_w \text{ halts on } x\}$

Proof: if HALT computable → ACCEPT computable

This was a proof by reduction!

It proved that HALT $\leq_R$ ACCEPT (in their hardness level)

https://ericllong.wordpress.com/2022/09/26/square-peg-round-hole/

APOLLO 13 LITHIUM HYDROXIDE CANISTER

How it's made

Equipment needed: 1 lithium hydroxide canister, 1 roll special gray tape (duct tape), bags from 2 Liquid Cooling Garments (LCG), 1 LM cue card, 1 piece of a towel, red hose from EVA suit

Step 1 – Cut off outer bag on the Liquid Cooling Garment (LCG).

Step 2 – Remove inner bag from the Liquid Cooling Garment (LCG).

Step 3 – Make "belts" with tape, two on side, one near top and one near bottom. Sticky side out.

Step 4 – Anchor tape with more tape, two 2-foot strips wrapped around canister at right angles to other tape strips. Forms a square grid.

Step 5 – Create arch over top of canister with EVA cue card.

Step 6 – Stop up bypass hole with part of a towel.

Step 7 – Put inner bag over the top of the canister, with "ears" or corners of the bag oriented along the open ends of the arch.

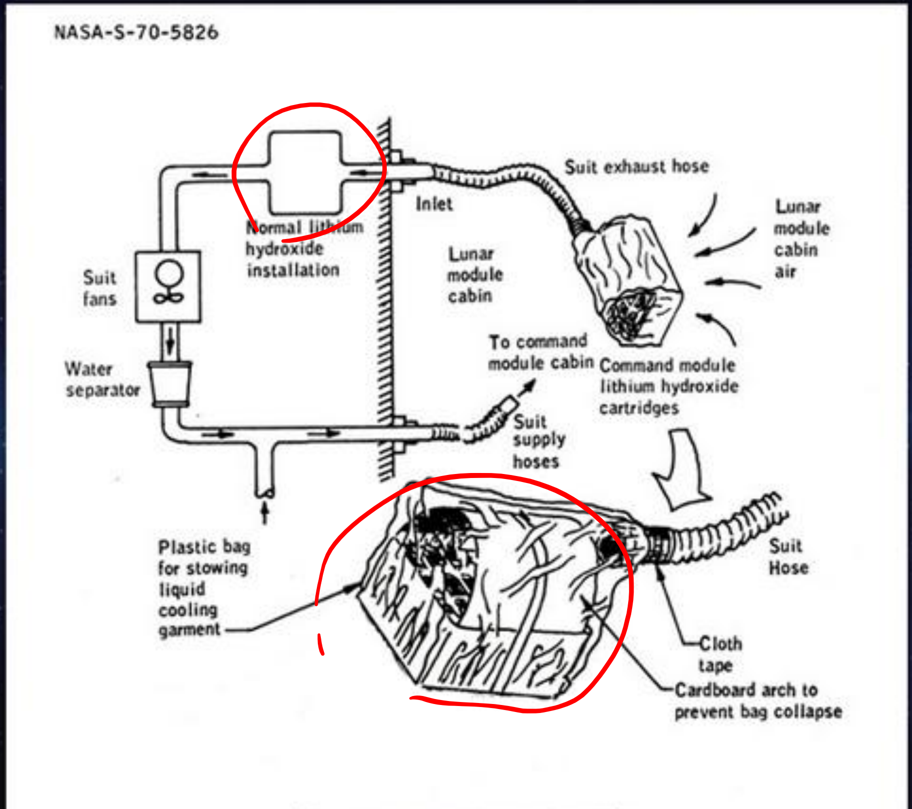
Step 8 – Press bag against sticky parts of tape on sides of canister. Use 3-foot strip of tape and wrap it around outside of bag over the bottom sticky belt, sealing things up.

Step 9 – Trim excess bag material to the bottom of the container.

Step 10 – Tape four 12-inch strips along outside of bag across the ribs for stability.

Step 11 – Place red suit hose in top of bag by cutting a diagonal hole in one "ear" of the plastic bag near the arch. Slip hose to the center of the canister. Tape bag to hose.

Step 12 – Secure towel in bypass hole with two pieces of tape.



(a) Configuration schematic.

Figure 6.7-1.- Supplemental carbon dioxide removal system.

SPACE CENTER HOUSTON | APOLLO 50 NEXT GIANT LEAP

https://spacecenter.org/apollo-13-infographic-how-did-they-make-that-co2-scrubber/

Charge

Uncomputability

Halting Problem

Church-Turing Thesis

Reductions

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

Photo credit