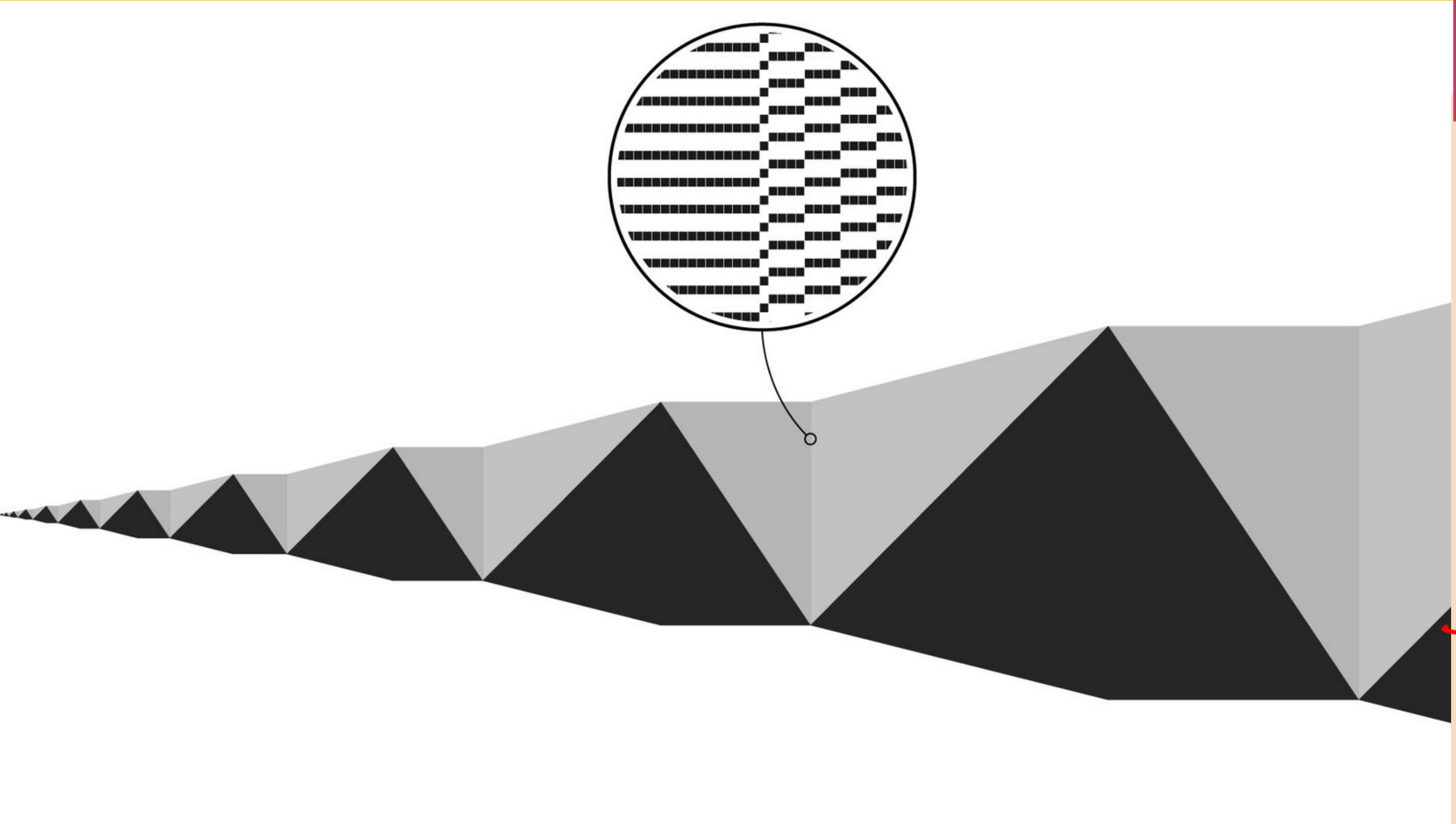
PS9 due next Monday, Apr 14
PS10 will be released next Tue, Apr 15.



Class 22:
Reductions
Rice's Theorem

University of Virginia
cs3120: DMT2
Wei-Kai Lin

[Photo credit](#)

Remaining Assignments

Weekly PRR: there are 3 more to go, total 14

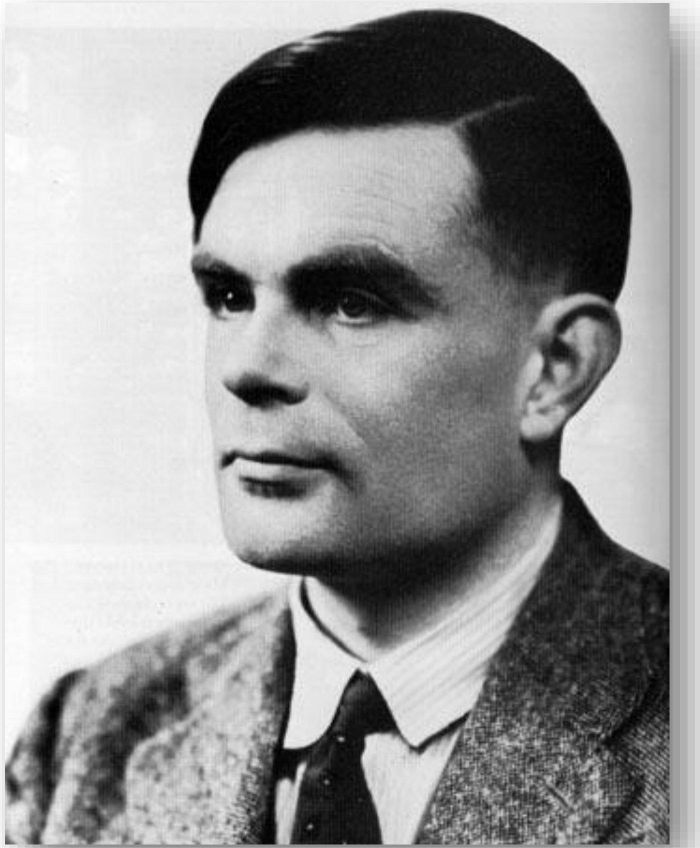
Finish 11 and get all points

PS10: Release tentatively April 15, due April 25

Recap: Church-Turing Thesis

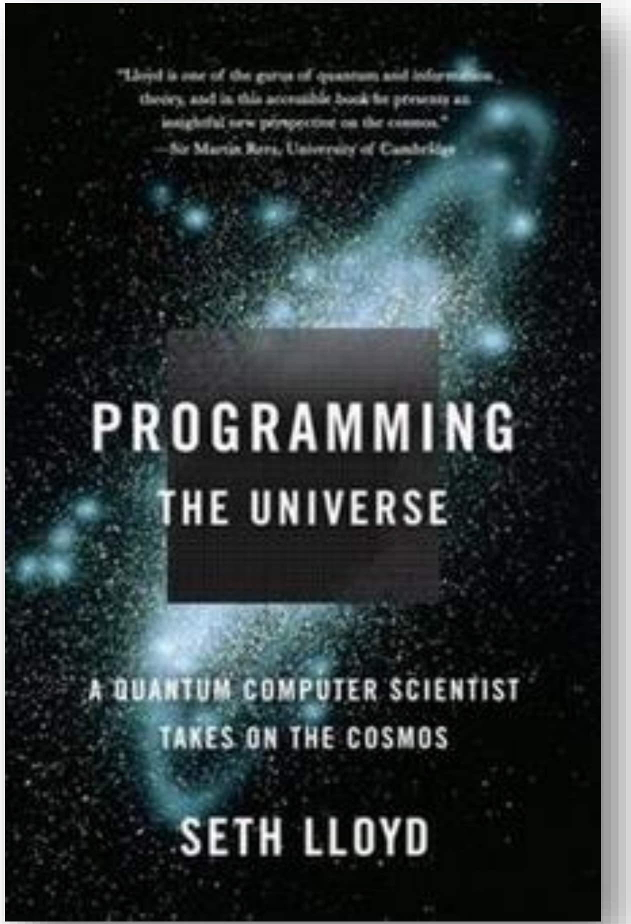
Alonzo Church, 1903-1995

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



Alan Turing, 1912-1954

Seth Lloyd: *Programming the Universe*



https://en.wikipedia.org/wiki/Seth_Lloyd
https://en.wikipedia.org/wiki/Programming_the_Universe

Recap: Halting is uncomputable

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

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

Question: Does this program halt?

```python
# This program prints Hello, world!

print('Hello, world!')
```

```python
while True:
    pass
```

Reductions

Recap: How we proved uncomputability

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

Proof: …

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

Proof: if HALT computable → ACCEPT computable

Three step process:

1. **<u>Assume</u>** M_H decides HALT
2. **<u>Construct</u>** $M_A(w, x) \coloneqq$ If $M_H(w, x)$ return $U(w, x)$ else return 0
3. **<u>Prove</u>** 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

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.

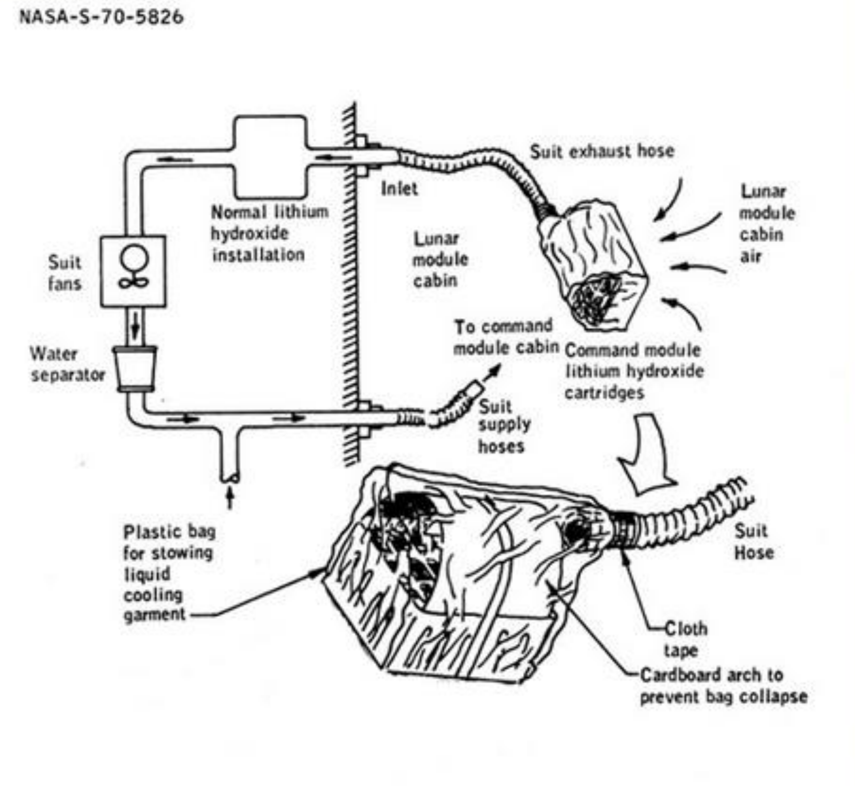
Step 8 – Press bag against sticky parts of tape on sides of canister. Use 3-foot strip of tape and wrap it around outside of bag over the bottom sticky belt, sealing things up.

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.

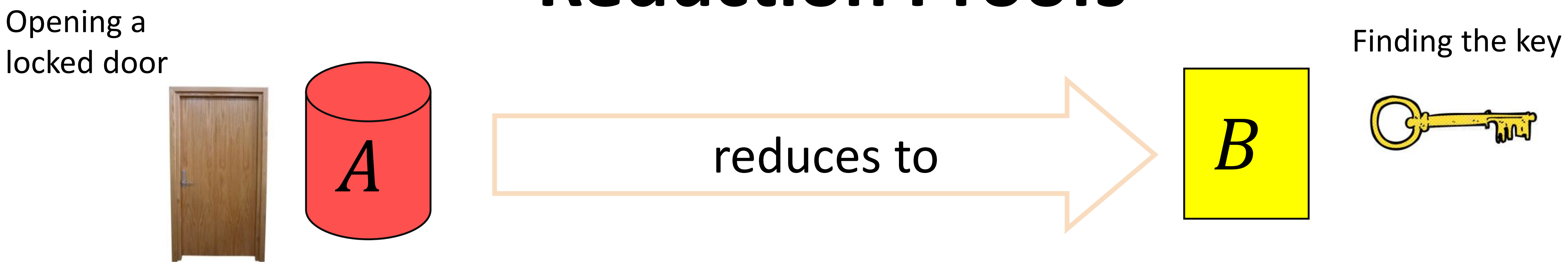
Figure 6.7-1.- Supplemental carbon dioxide removal system.

SPACE CENTER HOUSTON

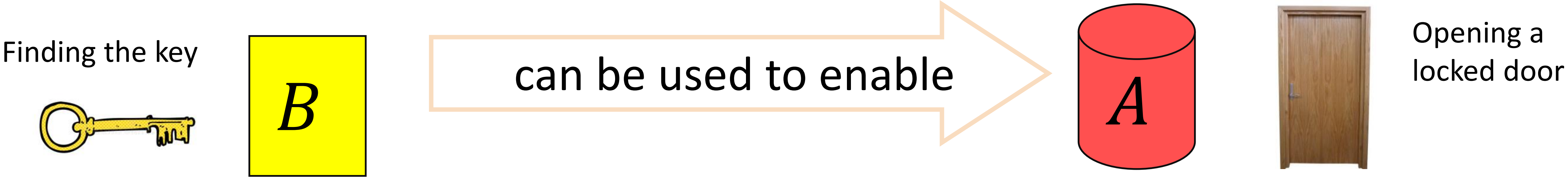
APOLLO 50 NEXT GIANT LEAP

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

Reduction Proofs



Reduction is in the *opposite* direction of the doing things



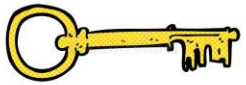
A is not a harder problem than B

$$A \leq_R B$$

Reduction Proofs

Can we open the door? DK
Can we find the key? DK
If we can find the key, can we open the door? Y
If we can NOT find the key, can we open the door? DK
If we can open the door, can we find the key? DK
If we can NOT open the door, can we find the key? N

Finding the key



B can be used to enable A

Opening a locked door

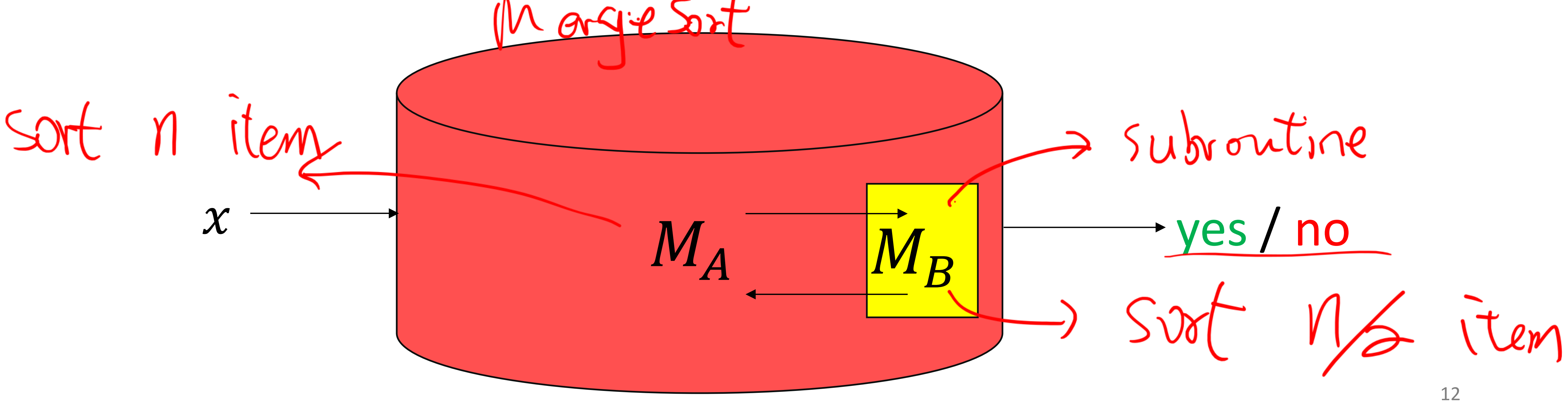
A is not a harder problem than B

$$A \leq_R B$$

Another way to imagine reduction

Reducing "task" A to "task" B, denoted by $A \leq_R B$

1. Assume that algorithm M_B solves B
2. Design algorithm M_A (that uses M_B as subroutine)



Terminology

To solve A by solving B

A reduction from A to B

$A \leq_R B$

Which is easier, A or B?

A is easier or eq to B

Terminology

To solve A by solving B

A reduction from A to B

$A \leq_R B$

Which is easier, A or B?

B may “look” easier!

Example: Halting on the zero problem

$HoF(w) = \begin{cases} 1 & w \text{ halts } \cancel{\text{on}} \text{ on } 0 \\ 0 & \text{o.w.} \end{cases}$

HALT-ON-ZERO $= \{w \mid M_w$ halts on input $0\}$

Maybe HALT-ON-ZERO is easier, and so decideable?

No. It is undecidable.

$HoZ(w) = HALTS(w, 0)$

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

Example: Halting on the zero problem

HALT-ON-ZERO $= \{w \mid M_w$ halts on input $0\}$

Maybe HALT-ON-ZERO is easier, and so decideable?

No. It is undecidable.

Proof by reduction:

want HALT HoZ

What should be the direction?

from ~~HoZ~~ HALT **to** ~~HALT~~ HoZ trivial HoZ(w)=H(w,0)

Assume we can decide HALT-ON-ZERO → decide HALT

HALT-ON-ZERO is uncomputable

Assume $M_Z(w)$ computes HALT-ON-ZERO(w).

Want: $M_H(w, x)$ that computes HALTS(w,x).

1. Construct TM $w'(y)$: a. if $y=0$, $U(w,x)$
   b. o.w. $y \neq 0$, halt.
2. Output $\underline{M_Z(w')}$
   $= HoZ(w')$

Claim: For all $\underline{w, x}$, $M_H(w,x) = HALT(w,x)$

Uncomputability:
How often does it happen?

What can we ask about TMs?

Does the machine **behave** in a certain way?

- Does it write "3" during its execution?
- Does it run for 15 steps on input x?
- (questions that could have different answers for different machines which compute the same function/language)

Does the machine **compute** a certain kind of function/language?

- Does the machine accept only finitely many inputs?
- Could the function of this machine also be computed by a finite state automaton?
- (questions that will always have the same answer for machines which compute the same function/language)
- Call these **Semantic Properties**

Definition: Semantic Property

Two machines M_1, M_2 are **functionally equivalent** (denoted $M_1 \equiv M_2$) if $\forall x \in \{0,1\}^*$, $M_1(x) = M_2(x) = \cancel{g(x)}$

$\perp$ $\quad$ $\perp$

TM

$F(W) \mapsto \{0,1\}$

A function $F: \{0,1\}^* \to \{0,1\}$, defined on Turin machines, is **semantic** if for every pair of functionally equivalent TMs (M_1, M_2), $F(M_1) = F(M_2)$.

if $W_1 \equiv W_2$ then $\underbrace{H}_{\text{Halt}}(W_1) = H(W_2)$

Namely, this is a property of the **function**/language of the machine, not of the **behavior** of the machines

Examples of Semantic properties

What's $F(M)$?
Is F a semantic property?

$\hat{F}(M)$

- Does the machine **behave** in a certain way?
  - Does it write "3" during its execution?
  - Does it run for 15 steps on input x?
  - (questions that could have different answers for different machines which compute the same function/language)
- Does the machine **compute** a certain kind of function/language?
  - Does the machine accept only finitely many inputs?

    Def $F(M) = 1$ iff

    $\forall M_1 \equiv M_2 \overset{?}{\Rightarrow} F(M_1) = F(M_2)$
  - Could the function of this machine also be computed by a finite state automaton? $F(M) = 1$ iff $\exists D$ s.t. $D(x) = M(x)$ all x

    $M_1 \equiv M_2 \Rightarrow F(M_1) = F(M_2)$
  - (questions that will always have the same answer for machines which compute the same **function**/language)

Rice's Theorem

If F is a sematic property, then either F is **not computable**, or F is trivial

Trivial: for all Turing machine M,

$$F(M) = 0$$

$$F(M) = 1$$

?

$F_{eq}(M_1, M_2) = 1$ iff $M_1 = M_2$

What's $F(M)$? Is F a semantic property?

~~def $F_1(M)$: run $M(n)$ $n=1,2,3\ldots$~~

Does Rice's theorem apply?

1. Machine M decides prime numbers $F_1(M) = 1$ iff M decides

   $\forall\ M_1 \equiv M_2$ is $F_1(M_1) = F_1(M_2)$? YES

   1 iff 1

2. M on input 0 uses more than 10 memory cells on its tape.

   $F_2(M) = 1$ iff M on 0 ...... IS F_i Sematic prop?

   $\exists\ M_1 \equiv M_2$ $F_2(M_1) \neq F_2(M_2)$ No

3. M always outputs a TM N that has property 2.

   $F_3(M)$

   semantic uncomp

4. M accepts TMs that have property 2

   $F_4(M)$

Proof of Rice's Theorem (Sketch)

Assume for contradict, ∃ F non trivial but computable

Suppose F is a non-trivial property of TMs:

$$M \equiv N \ \rightarrow F(M) = F(N) \in \{0,1\}$$

Exists M, N **such that** $F(M) = 0$ **and** $F(N) = 1$ by non-trivial

Proof of uncomputability of F:

Use a reduction from Halt-on-Zero to computing F

Assume there is M_F (W) that decides F. compute Want: a solver M_H that decides Halt-on-Zero

M_H

Carefully pick TMs P, Q such that $F(P) = 0 \neq F(Q) = 1$

M_H (w) gets TM description w and modifies it to $\widetilde{w}$ so that:

- **If** TM_w **halts on zero** → $\widetilde{w} \equiv Q$
- **If** TM_w **does not halt on zero** → $\widetilde{w} \equiv P$

So, all M_H does is to return $M_F(\widetilde{w})$

Analysis: If M_w (W) halts on 0 → $\widetilde{w} \equiv Q$, F(Q)=0
and if it (w) does not halt → $\widetilde{w} \equiv P$. F(Q)=1
So, $M_F(\widetilde{w})$ will tell us whether M_w halts on 0 or not.

Proof of Rice's Theorem (Sketch)

Suppose F is a non-trivial property of TMs:

$$M_1 \equiv M_2 \;\rightarrow F(M_1) = F(M_2) \in \{0,1\}$$

Exists P, Q **such that** $F(P) = 0$ **and** $F(Q) = 1$

Proof of uncomputability of F:

Use a reduction from Halt-on-Zero to computing F

Assume there is M_F that decides F. Want: a solver M_H that decides Halt-on-Zero

Carefully pick TMs P, Q such that $F(P) = 0 \neq F(Q) = 1$

M_H gets TM description w and modifies it to $\widetilde{w}$ so that:

If TM_w **halts on zero** → $\widetilde{w} \equiv Q$

If TM_w **does not halt on zero** → $\widetilde{w} \equiv P$

So, all M_H does is to return $M_F(\widetilde{w})$

How to modify w into $\widetilde{w}$?

Full Proof of Rice's Theorem

Suppose F is a non-trivial property of TMs:

$$M_1 \equiv M_2 \ \rightarrow F(M_1) = F(M_2) \in \{0,1\}$$

Exists P, Q **such that** $F(P) = 0$ **and** $F(Q) = 1$

Proof of uncomputability of F:

Use a reduction from Halt-on-Zero to computing F

Assume there is M_F that decides F. Want: a solver M_H that decides Halt-on-Zero

Pick TMs P, Q such that P never halt and $F(Q) \neq F(P)$. By non-triviality, Q exists.

Assume w.l.o.g. $F(P) = 0$ so $F(Q) = 1$ (if not, swap P and Q)

M_H gets TM description w and modifies it to $\tilde{w}$ so that:

If TM_w **halts on zero** → $\tilde{w} \equiv Q$

If TM_w **does not halt on zero** → $\tilde{w} \equiv P$

So, all M_H does is to return $M_F(\tilde{w})$

Full Proof of Rice's Theorem

Assume for contra $\exists F$

Suppose F is a non-trivial property of TMs:

$$M_1 \equiv M_2 \; \rightarrow F(M_1) = F(M_2) \in \{0,1\}$$

Exists P, Q **such that** $F(P) = 0$ **and** $F(Q) = 1$

Proof of uncomputability of F:

Use a reduction from Halt-on-Zero to computing F

Assume there is M_F that decides F. Want: a solver M_H that decides Halt-on-Zero

Pick TMs P, Q such that P never halt and $F(Q) \neq F(P)$. By non-triviality, Q exists.

Assume w.l.o.g. $F(P) = 0$ so $F(Q) = 1$ (if not, swap P and Q)

M_H gets TM description w and modifies it to $\widetilde{w}$ so that:

If TM_w **halts on zero** → $\widetilde{w} \equiv Q$

If TM_w **does not halt on zero** → $\widetilde{w} \equiv P$

So, all M_H does is to return $M_F(\widetilde{w})$

Q

P

$M_F(\widetilde{w}) = HZ(w)$

Given input TM description w, modify w into $\widetilde{w}$ as follows:

$\widetilde{w}(x)$:

"Given input ~~a~~ x, first run $M_w(0)$.

Then return $Q(x)$."