Catalytic Computing

(in 10 Minutes or Less)

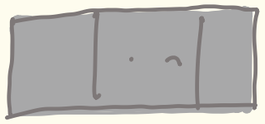
Ian Mertz
Charles University

Theory of Computation

A "trivial" fact:

THEORY OF COMPUTATION

A "trivial" fact:

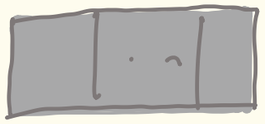
1. COMPUTE f  SPACE(f)

2. REMEMBER z 

SPACE(z)

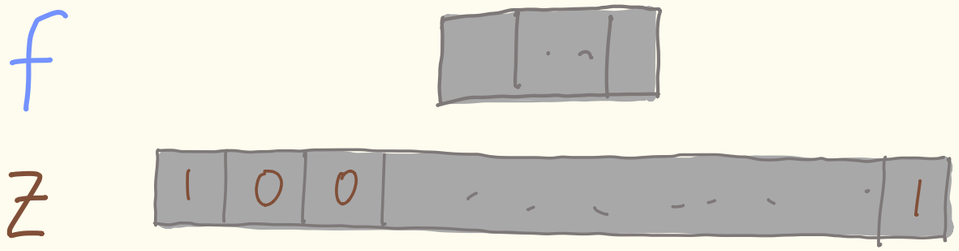
THEORY OF COMPUTATION

A "trivial" fact:

1. COMPUTE f  SPACE(f)

2. REMEMBER z 

SPACE(z)

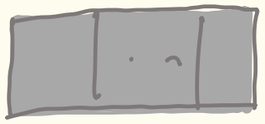
3. BOTH 

SPACE(f, z)

THEORY OF COMPUTATION

A "trivial" fact:

1. COMPUTE f



2. REMEMBER z

3. BOTH

f

z

Q:

$$\mathrm{SPACE}(f) + \mathrm{SPACE}(z)$$

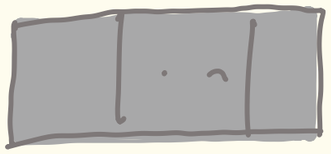
equals?

$$\mathrm{SPACE}(f, z)$$

Theory of Computation

A "trivial" fact:

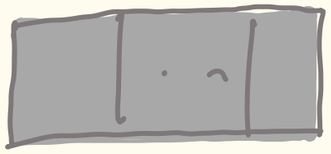
Option #1:



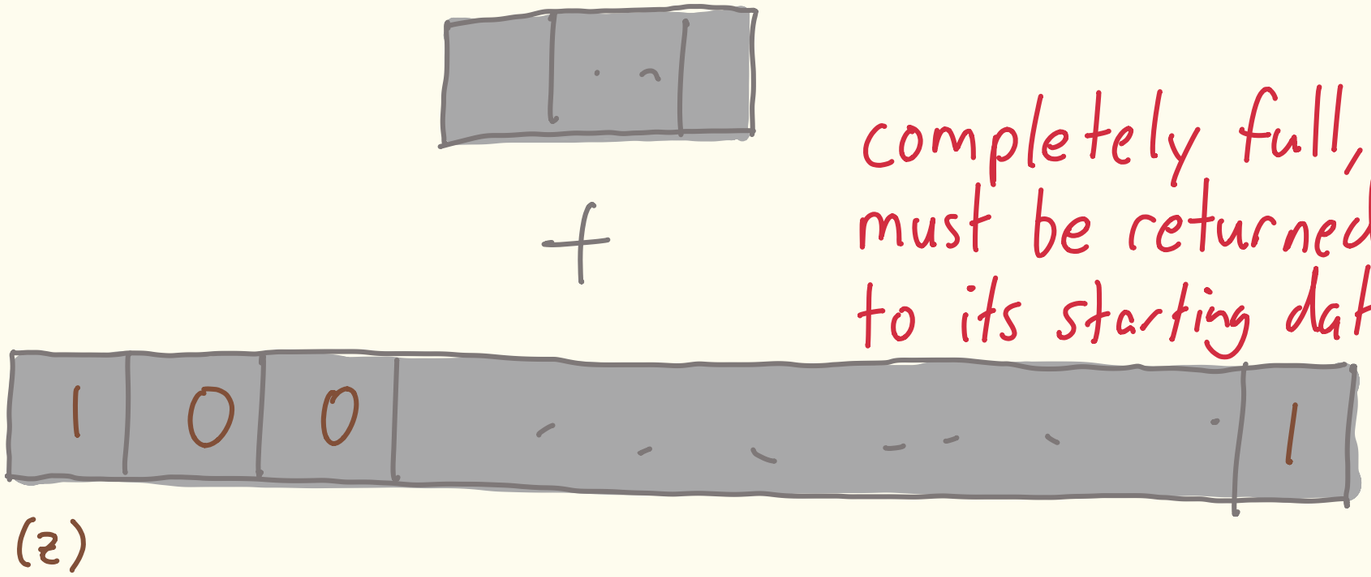
Theory of Computation

A "trivial" fact:

Option #1:



Option #2:



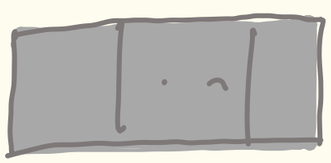
completely full, must be returned to its starting data

Theory of Computation

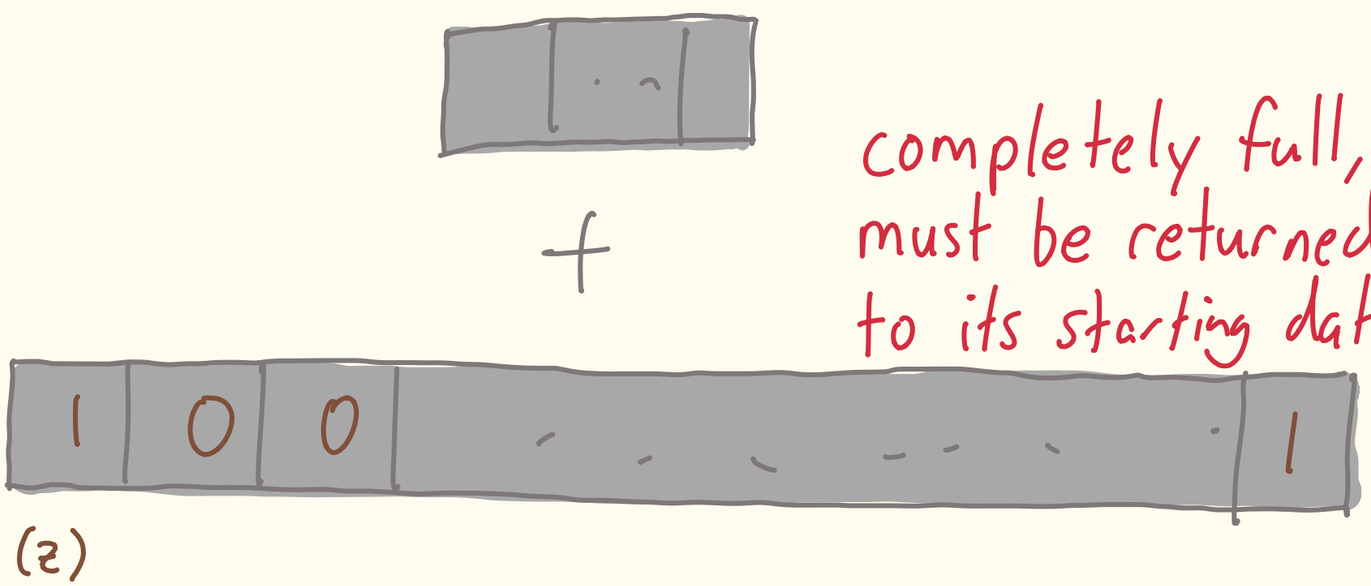
Q: is there an f solvable by #2 but <u>not</u> #1?

A "trivial" fact:

Option #1:



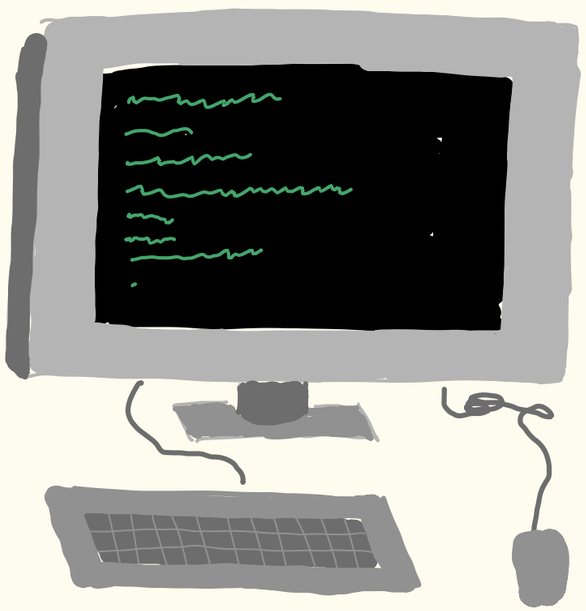
Option #2:



completely full, must be returned to its starting data

Theory of Computation

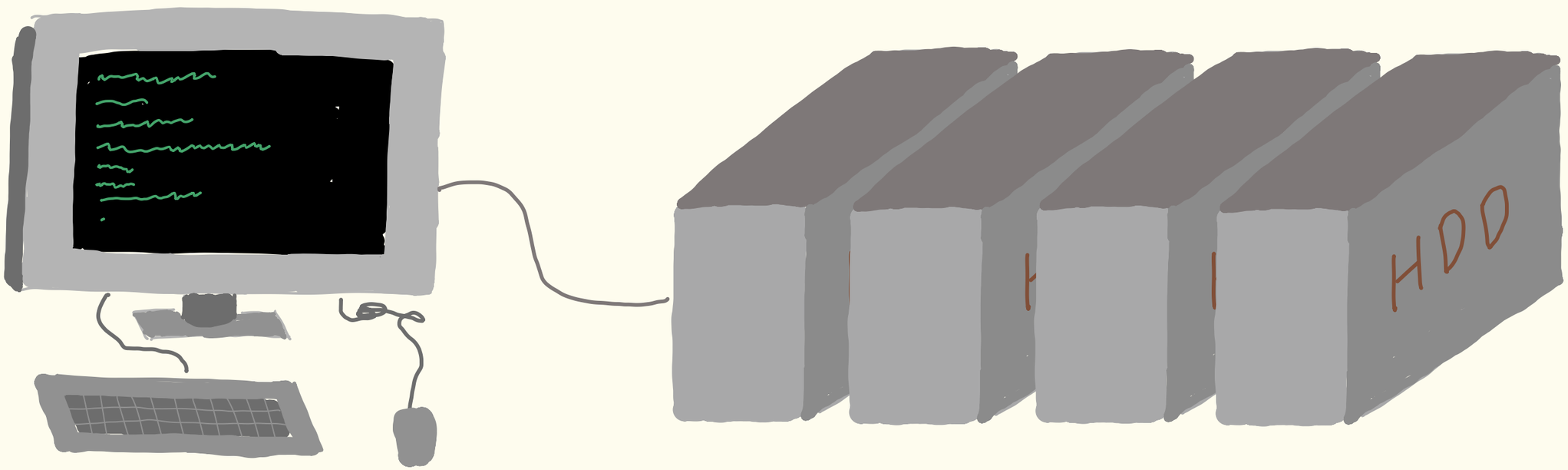
A "trivial" fact:



ERR: OUT_OF_MEM

Theory of Computation

A "trivial" fact:



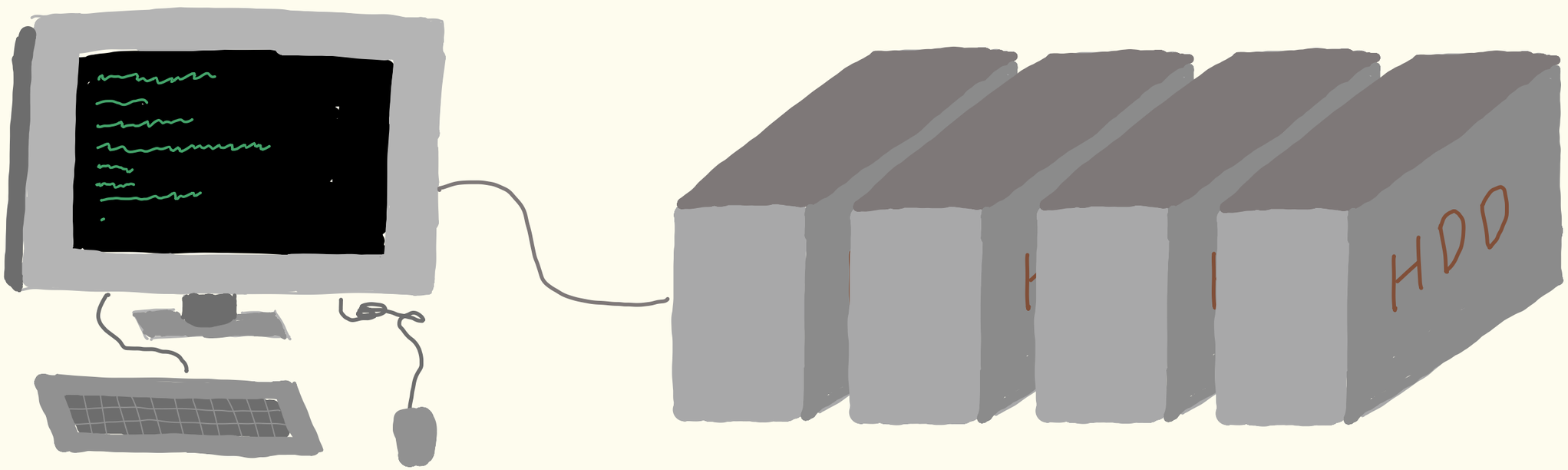
SUCCESS

full

Theory of Computation

A "trivial" fact:

NO WAY



SUCCESS

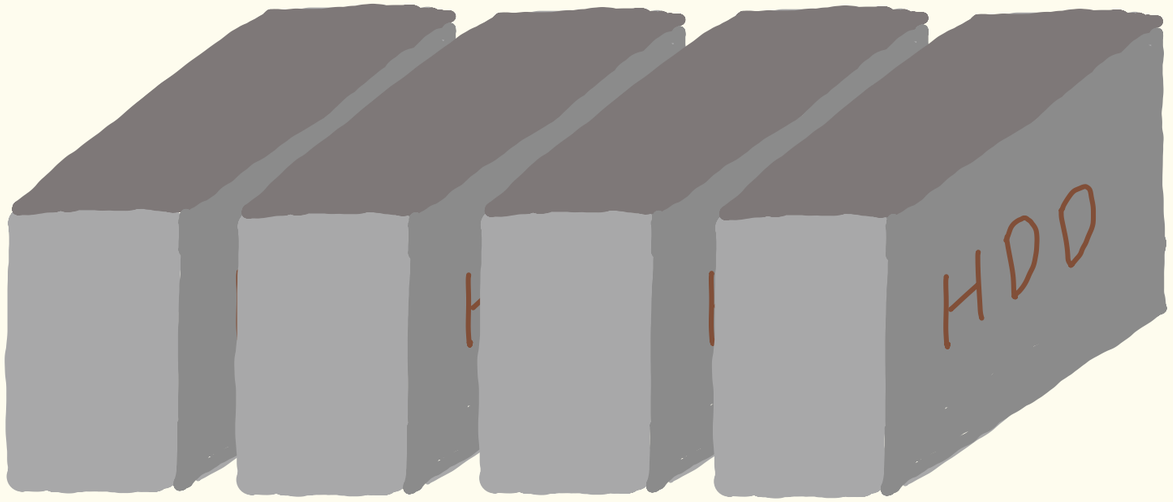
full

Theory of Computation

A "~~trivial~~" fact:
false

[BCKLS'14]:

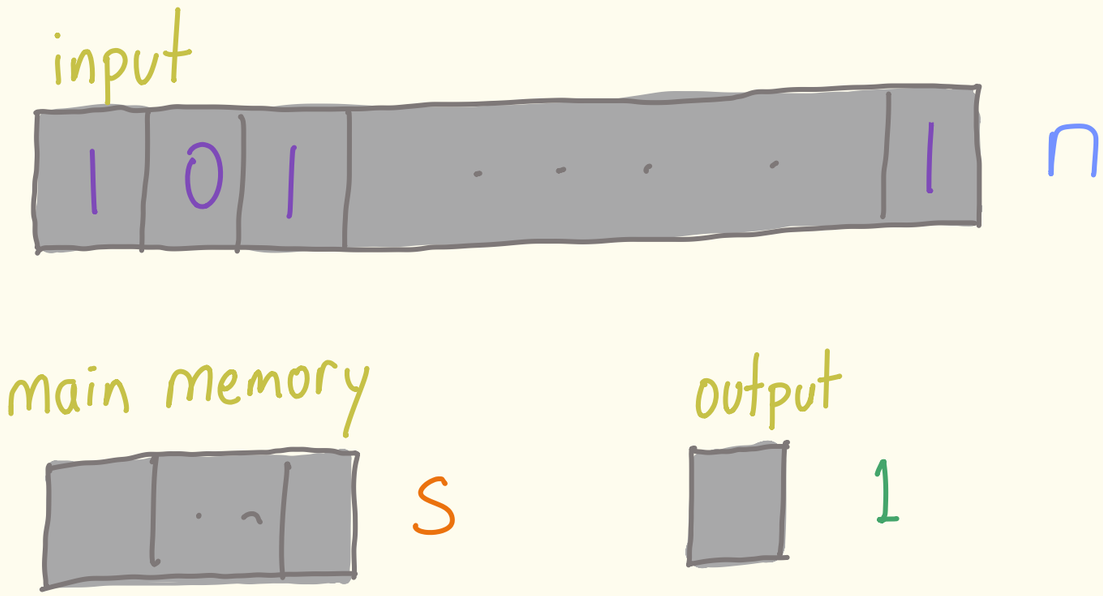
for many f
and every z



full memory provably adds a lot of power!

The Study of Reuse

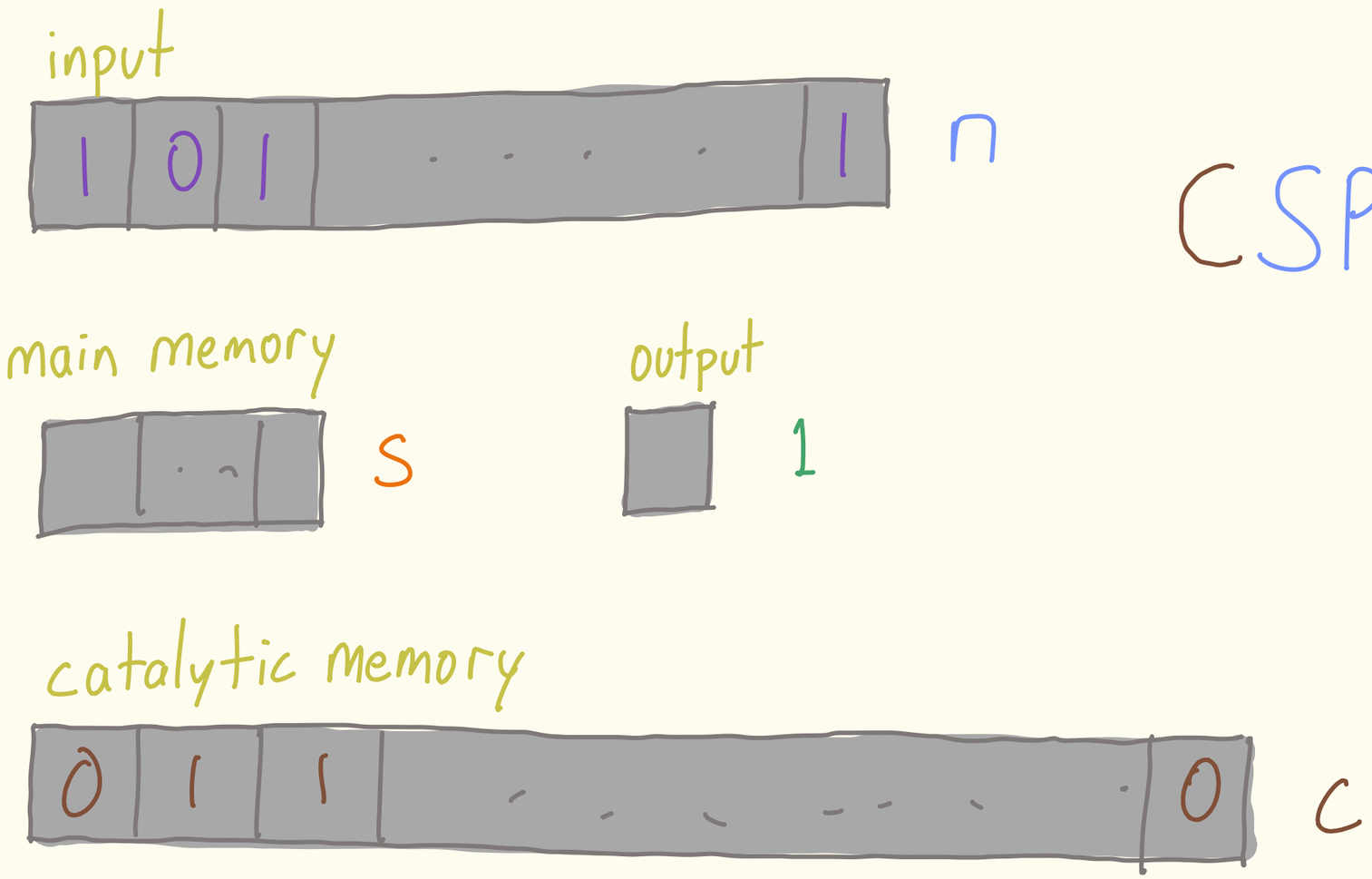
catalytic computing [BCKLS'14]



SPACE[s]

The Study of Reuse

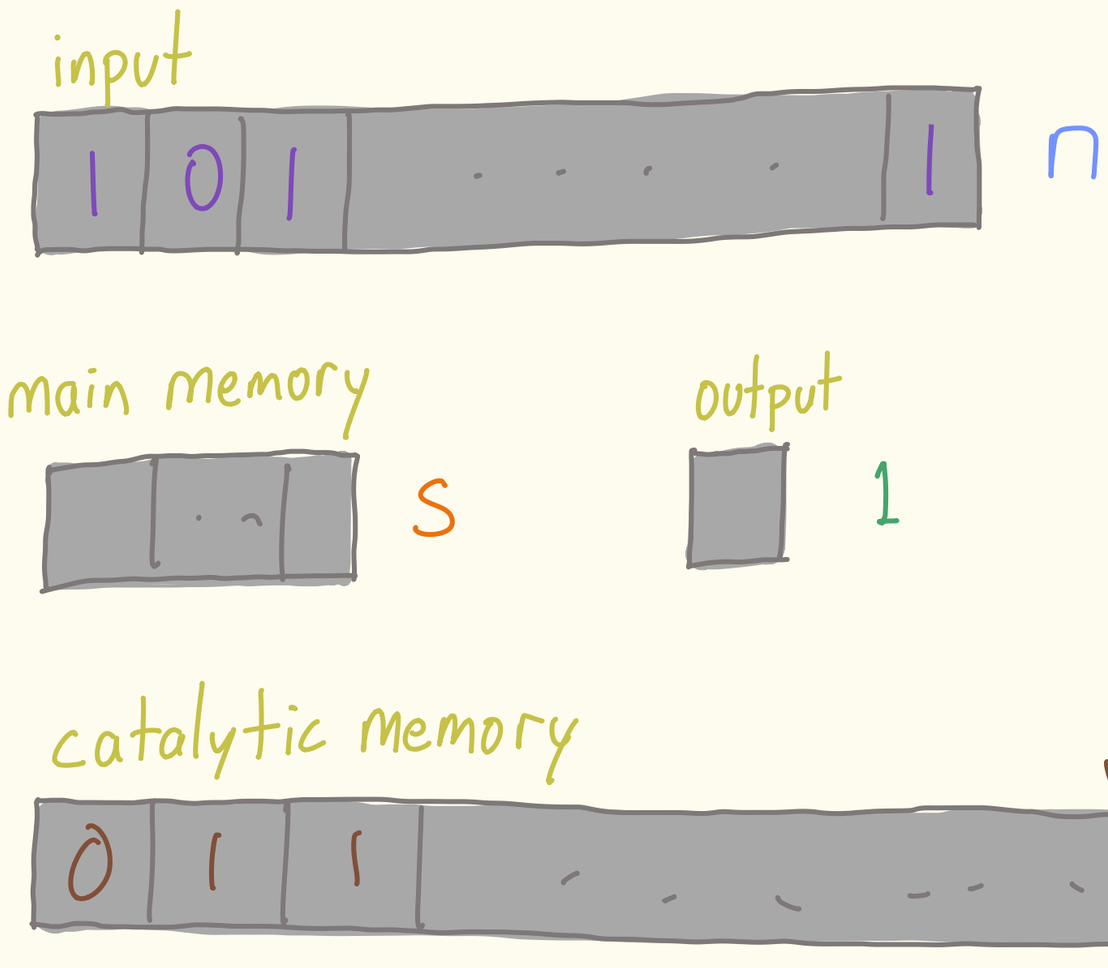
catalytic computing [BCKLS'14]



CSPACE[s, c]

The Study of Reuse

catalytic computing [BCKLS'14]

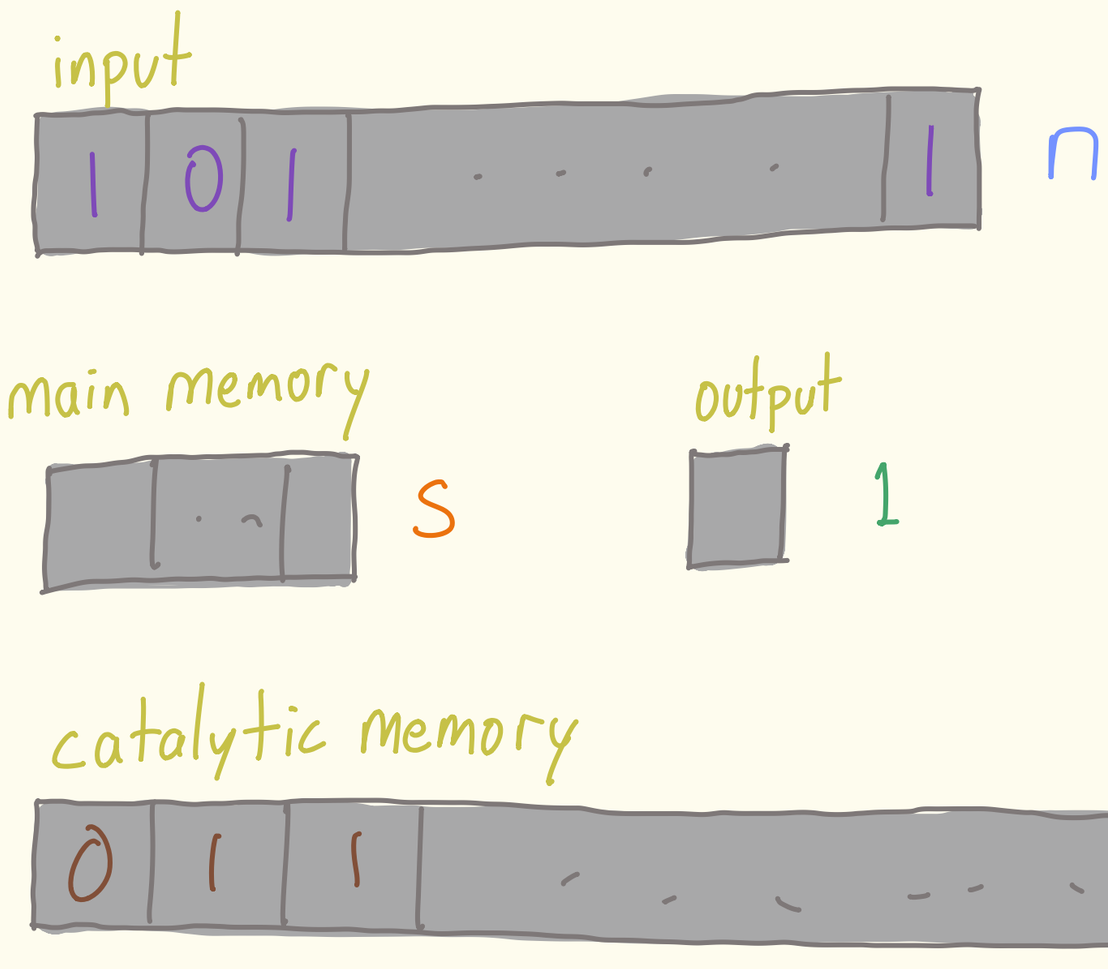


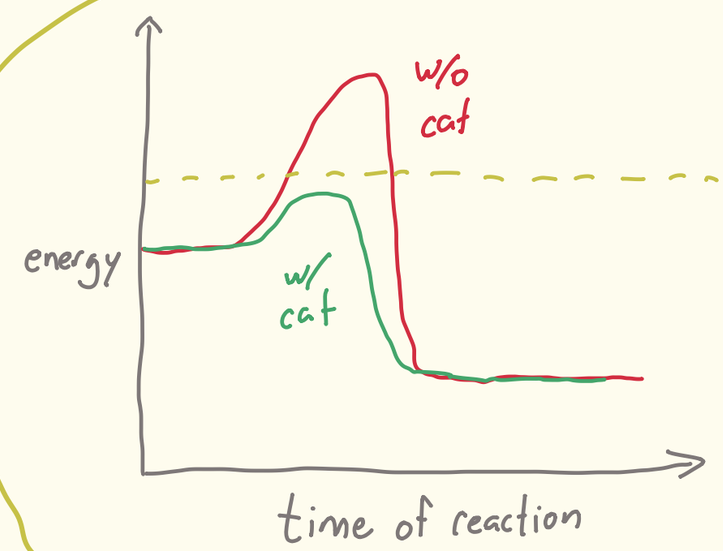
$\mathrm{CSPACE}[s, c]$

$\forall \tau$: can freely use
must reset to τ

The Study of Reuse

catalytic computing [BCKLS'14]





$2NO + O_2 \rightarrow 2NO_2$

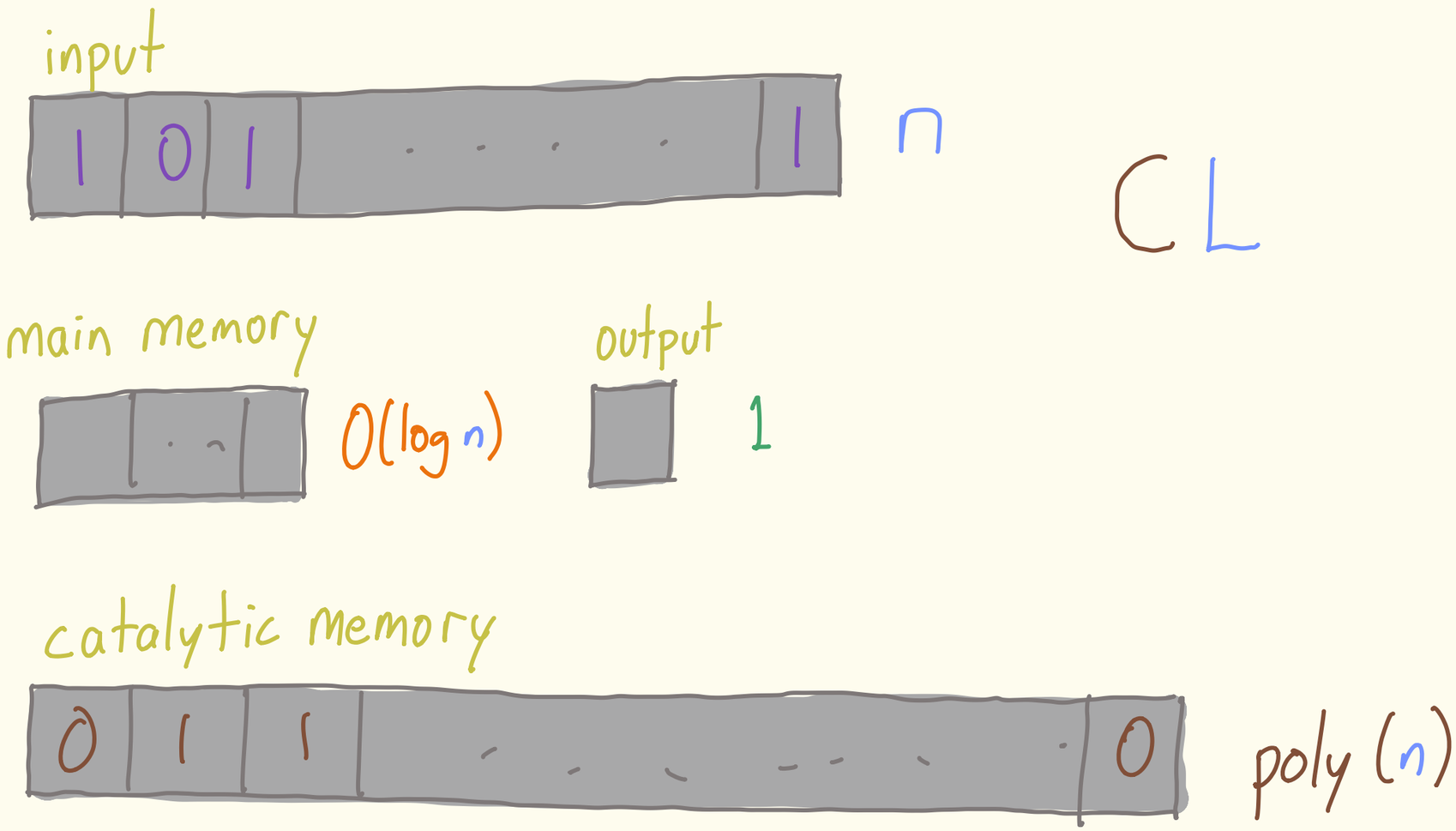
$NO_2 + SO_2 \rightarrow NO + SO_3$

Net: $2SO_2 + O_2 \rightarrow 2SO_3$

$+2NO$ $\quad$ $+2NO$

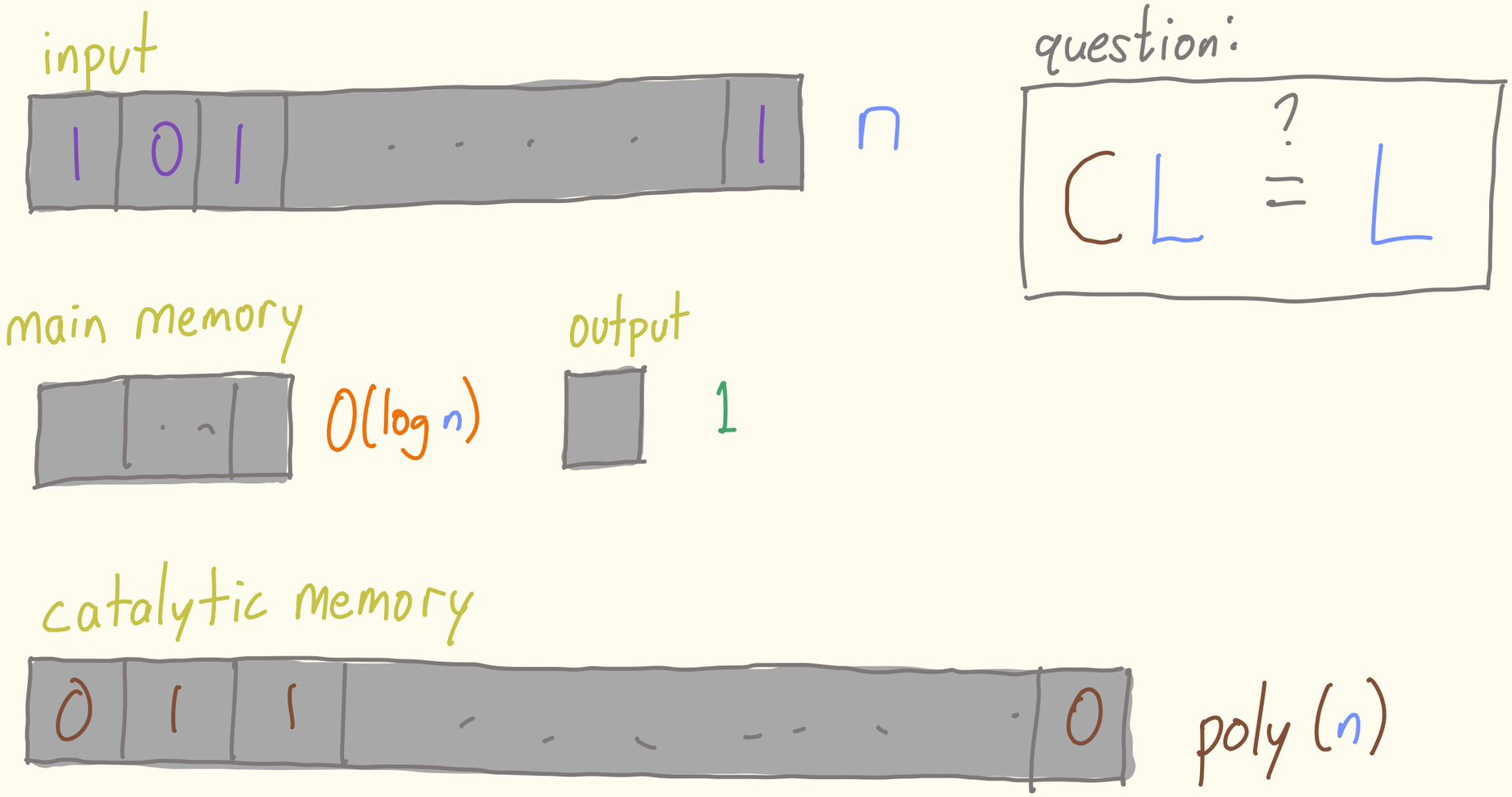
The Study of Reuse

catalytic computing [BCKLS'14]



The Study of Reuse

catalytic computing [BCKLS'14]



The Power of Reuse

trivial:

$$L \subseteq CL \subseteq PSPACE$$

The Power of Reuse

trivial:

$$L \subseteq CL \subseteq PSPACE$$

[BCKLS'14]:

$$CL \subseteq ZPP$$ (randomized polytime)

The Power of Reuse

trivial:

$$L \subseteq CL \subseteq PSPACE$$

[BCKLS'14]:

random walks
connectivity
determinant

$$\in CL \subseteq ZPP$$ (randomized polytime)

The Power of Reuse

trivial:

$$L \subseteq CL \subseteq PSPACE$$

[BCKLS'14]:

random walks
connectivity
determinant
bip. matching [AM'25]
l. matroid inters. [AAV'25]

$\in CL \subseteq ZPP$ (randomized polytime)

[CLMP'25] partial derand.

The Power of Reuse

most likely:

$$L \subsetneq CL \subsetneq PSPACE$$

[BCKLS'14]:

random walks
connectivity $\in CL \subseteq ZPP$ (randomized polytime)
determinant
bip. matching [AM'25]
l. matroid inters. [AAV'25]

[CLMP'25] partial derand.

The Power of Reuse

CL:

The Power of Reuse

CL:

- equals its non-deterministic and randomized counterparts (plus all hierarchies therein)

[CLMP'25]

[KMPS'25]

The Power of Reuse

CL:

- equals its non-deterministic and randomized counterparts (plus all hierarchies therein)
- contains EXP in the right oracle world

[BCKLS'14]

[CGMPS'25]

The Power of Reuse

CL:

- equals its non-deterministic and randomized counterparts (plus all hierarchies therein)
- contains EXP in the right oracle world
- can trade errors in resetting the catalytic tape for free space (exact tradeoff known)

[GJST'24]

[FMST'25]

The Power of Reuse

CL:

- equals its non-deterministic and randomized counterparts (plus all hierarchies therein)
- contains EXP in the right oracle world
- can trade errors in resetting the catalytic tape for free space (exact tradeoff known)
- is currently orthogonal to all major subclasses of P (NC, TIMESPACE$[n^{O(1)}, \log^{O(1)} n]$, CC, etc.)

The Power of Reuse

Implications for SPACE:

The Power of Reuse

Implications for SPACE:

- new approaches to space-bounded derandomization

[DT'22]

[LPT'24]

[P'25]

The Power of Reuse

Implications for SPACE:

- new approaches to space-bounded derandomization

- new relationship between TIME and SPACE:

$$\mathrm{TIME}[t] \subseteq \mathrm{SPACE}[\sqrt{t \log t}]$$

[CM'24]

[W'25]

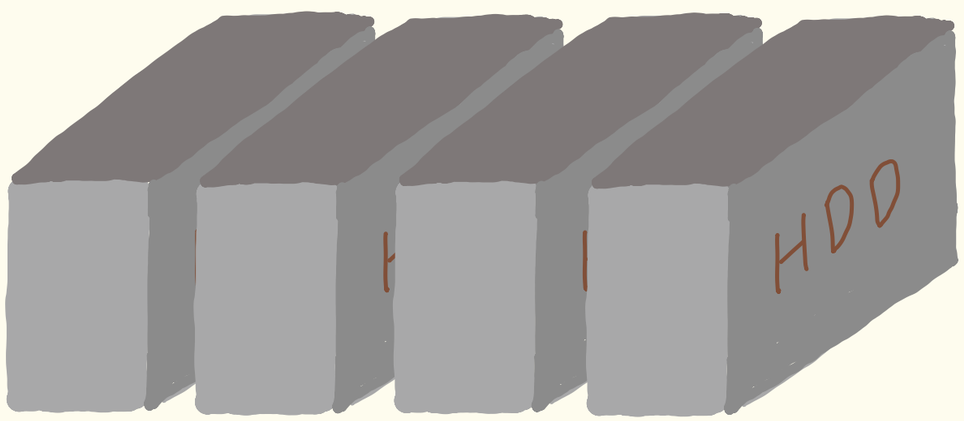
The Future of Reuse

CL vs P?

robustness

quantum

subroutines



hierarchies

time-space

in-place comp.

The Future of Reuse

1. Give a simple, direct proof of $\mathsf{uSTConn} \in \mathsf{L}$.
2. Give a simple, direct proof of $\mathsf{uSTConn} \in \mathsf{CL}$.
3. Give a simple, direct proof of $\mathsf{STConn} \in \mathsf{CL}$.
4. Try to improve Savich's Theorem: prove $\mathsf{NSPACE}(s) \subseteq \mathsf{SPACE}(o(s^2))$.
5. Improve the deterministic space complexity of $\mathsf{BPSPACE}(s)$.
6. Decide the space complexity of $\mathsf{TreeEval}$.
7. Give a register program for computing any polynomial $p(x_1 \dots x_n)$ using $O(n)$ registers over a constant size ring $\mathcal{R}$ and $O(1)$ recursive calls to the input x.
8. Show that for any branching program B of sufficiently large width $w = \Omega(1)$ and length ℓ, there exists a branching program B' of width $w/2$ and length $O(\ell)$ computing the same function.
9. Show that for any branching program B of sufficiently large width w and length ℓ, there exists a branching program B' of width $w - 1$ and length $\mathsf{poly}(\ell)$ computing the same function.
10. Find any function whose optimal space algorithm can be made almost entirely catalytic, i.e. a function requiring—or even that we only know how to do in—$\mathsf{SPACE}(s)$ but which is computable in $\mathsf{CSPACE}(\ll s, \approx s)$.
11. Prove $\mathsf{CL} \subseteq \mathsf{P}$.
12. Show that $\mathsf{P} \not\subseteq \mathsf{L/poly}$ implies $\mathsf{CL} \subseteq \mathsf{P}$.
13. Show that $\mathsf{CL} \subseteq \mathsf{P}$ would give strong evidence $\mathsf{ZPP} \subseteq \mathsf{P}$.
14. Show that NC^2, or even any circuit of $\omega(\log n)$ depth, can be computed in CL.
15. Give a register program for computing x^k in the non-commutative setting using linear space and a constant number of recursive calls to x.
16. Show that $\mathsf{BPNC}^1 \subseteq \mathsf{CL}$.
17. Design a catalytic branching program with $2^{O(n)}$ start nodes and total size $2^{O(n)} \cdot O(n)$ for any function f.
18. What is the power of $\mathsf{CL}/poly$, and does it have a natural syntactic characterization?
19. Show the existence of an oracle D such that $\mathsf{CL}^D = \mathsf{EXP}^D$.
20. Extend the $\mathsf{BPL} \subseteq \mathsf{CL}$ simulation to show $\mathsf{CBPL} \subseteq \mathsf{CL}$.
21. Show that CL is equivalent even if we allow $\omega(1)$ many errors on the catalytic tape at the end, or alternatively if we allow $O(1)$ such errors in expectation over all inputs x and catalytic tapes τ.
22. Utilize non-determinism in conjunction with catalytic computing in a non-trivial way.
23. Prove $\mathsf{CNSPACE}(s, c) \subseteq \mathsf{CSPACE}(s^2, c^2)$.
24. Implement a catalytic algorithm such that it is actually useful.
25. What does quantum catalytic space look like?
26. Devise a register program using basic instructions inspired by unitary computation, and use it to show non-trivial results for e.g. BQP.
27. Devise a circuit that uses known results from space reuse and catalytic computing to efficiently solve some problem in a way that we do not know how to do directly.
28. Show $\mathsf{TC}^1 \subseteq \mathsf{VP}$.
29. Is the network coding conjecture true or false?
30. Prove or disprove the network coding conjecture when all nodes are restricted to sending linear transformations of their incoming messages.
31. Is there a meaningful notion of a catalytic data structure, or is there anything to be gained from a data structure stored in catalytic memory?
32. Show CL is contained in some subclass of P, perhaps NC, given a believable cryptographic assumption.
33. Show evidence against objects in cryptography based on techniques in reusing space.
34. Show the existence, conditional or otherwise, of a natural class of cryptographic objects by using clean computation.
35. Prove that the existence of one-way functions in CL, or even any one-way function computable by a poly-size poly-length register program, implies the existence of one-way functions in NC^0.

source: [Mer'23]

THANKS!

Reversibility, Arithmetic, & Magic

Ian Mertz
Charles University

Reversibility

Q: how do we use a catalytic tape while still resetting it at the end of our computation?

Reversibility

Q: how do we use a catalytic tape while still resetting it at the end of our computation?

Idea: resetting $\longrightarrow$ reversing

Reversibility

Q: how do we use a catalytic tape while still resetting it at the end of our computation?

Idea: resetting $\rightarrow$ reversing

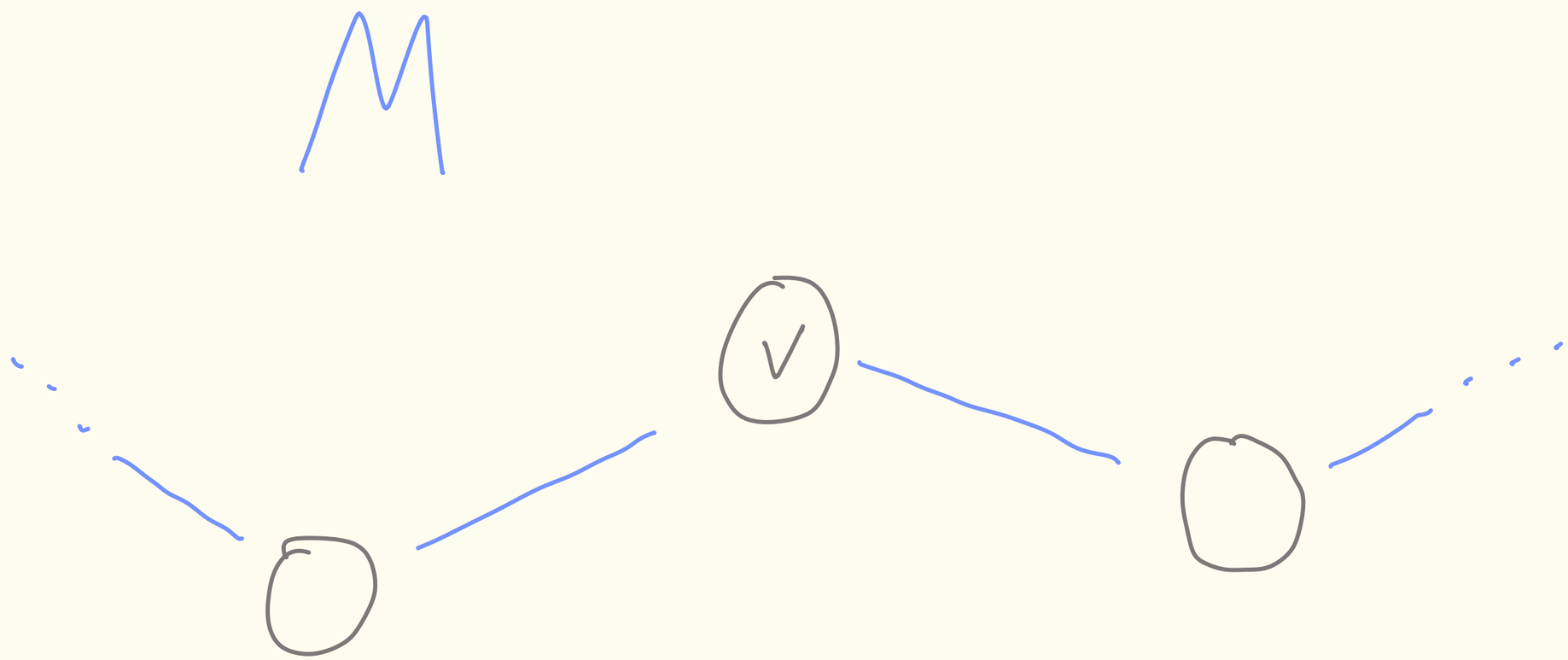
Holds a priori!

Reversibility

[LMT'00]: space is reversible

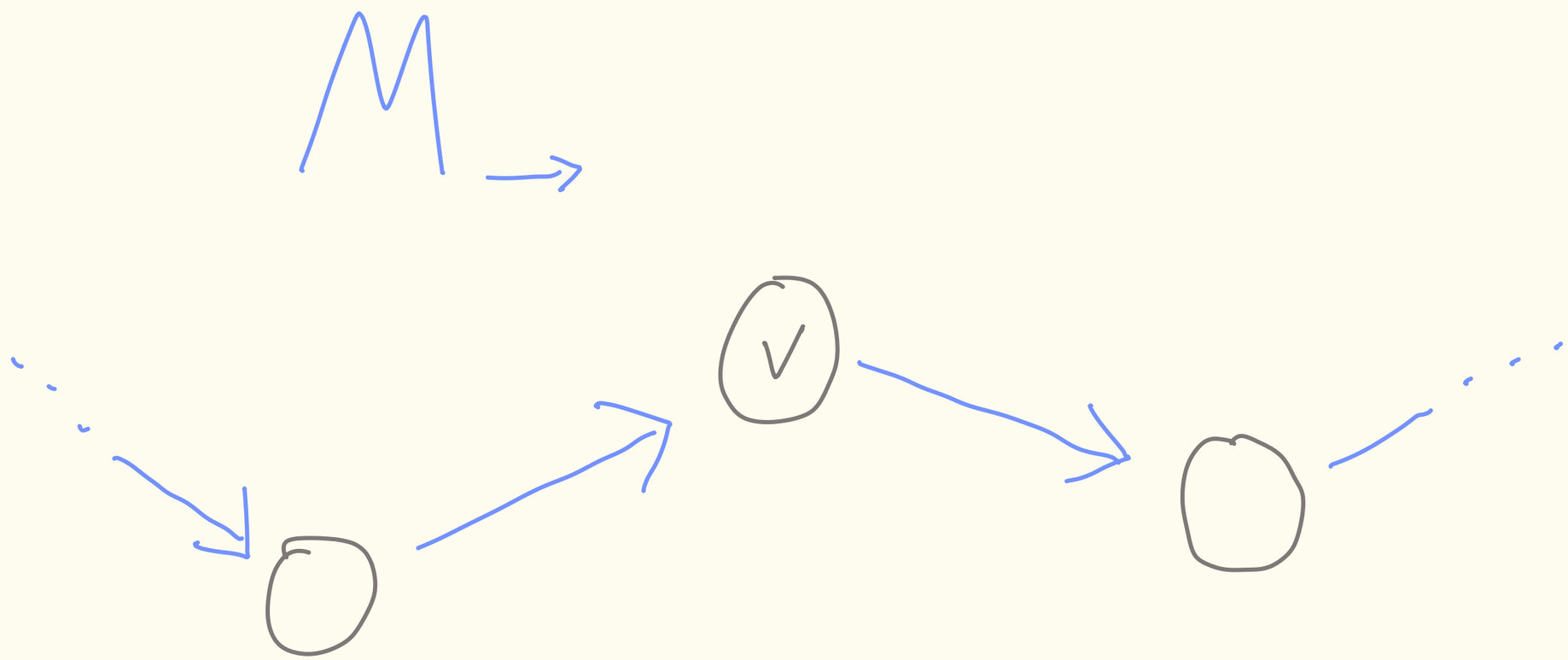
Reversibility

[LMT'00]: space is reversible



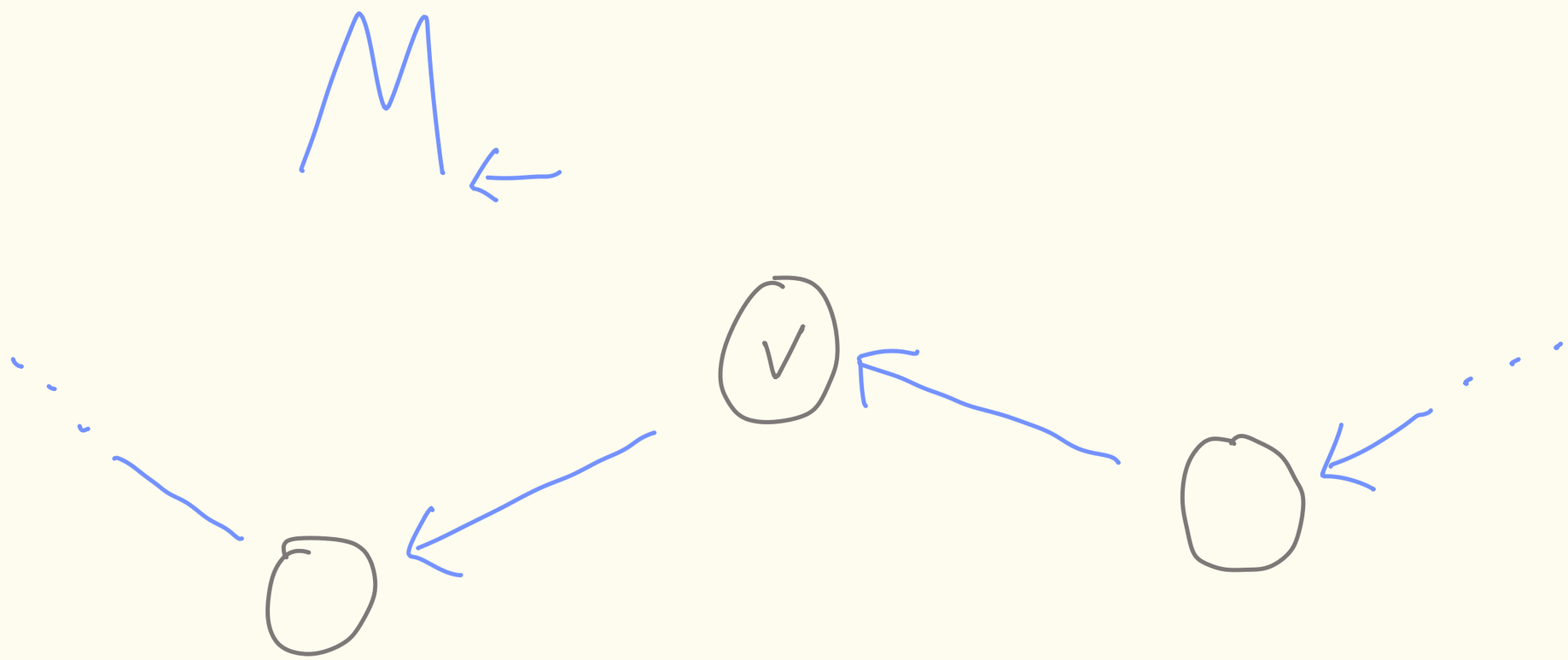
Reversibility

[LMT'00]: space is reversible



Reversibility

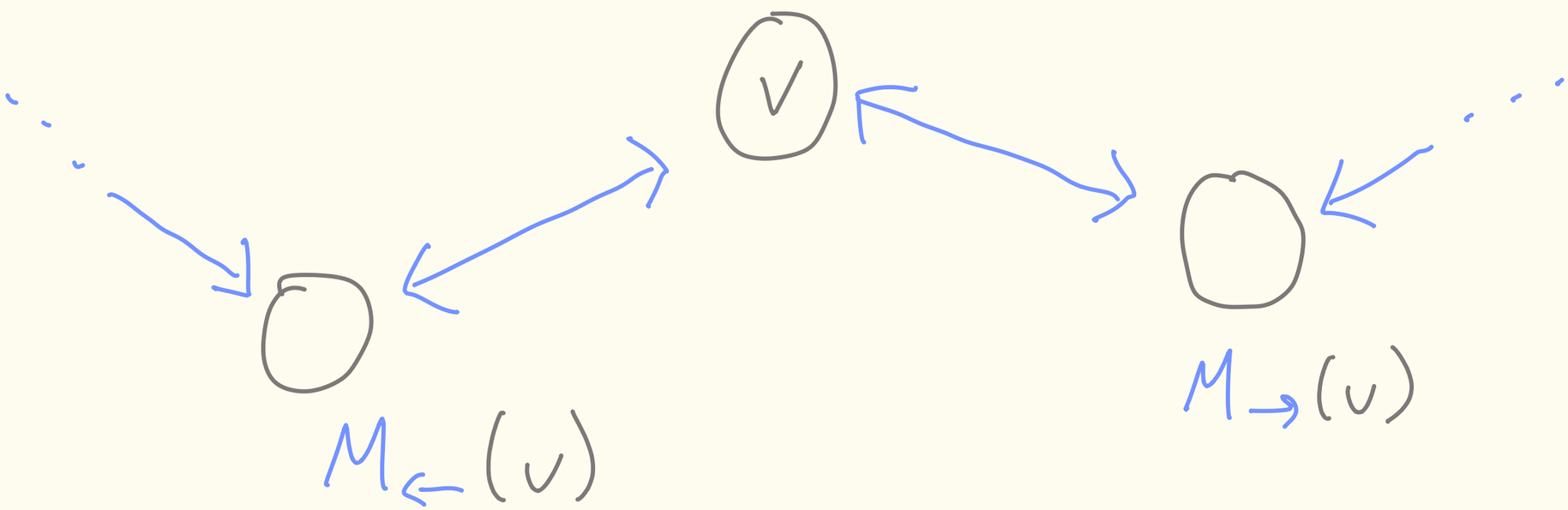
[LMT'00]: space is reversible



Reversibility

[LMT'00]: space is reversible

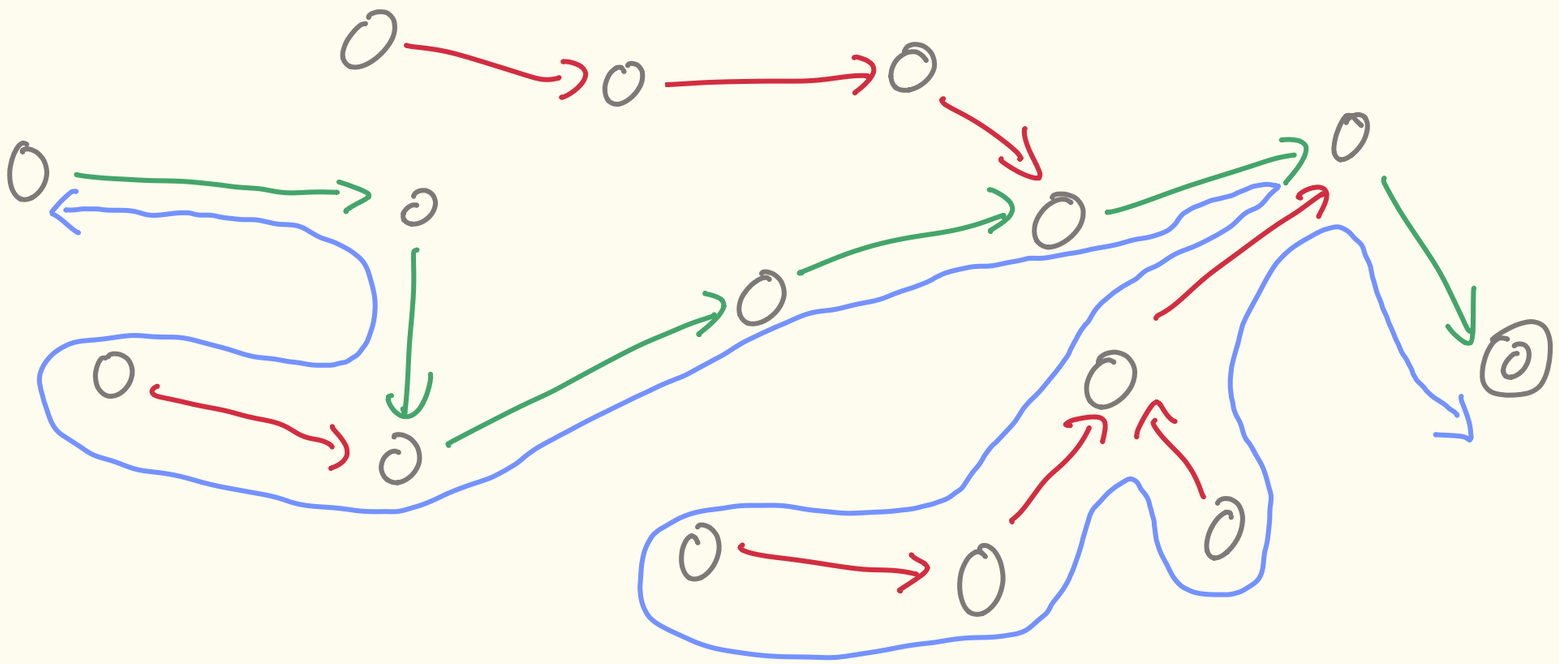
$$M_{\rightarrow}(M_{\leftarrow}(v)) = v = M_{\leftarrow}(M_{\rightarrow}(v))$$



Reversibility

[LMT'00]: space is reversible

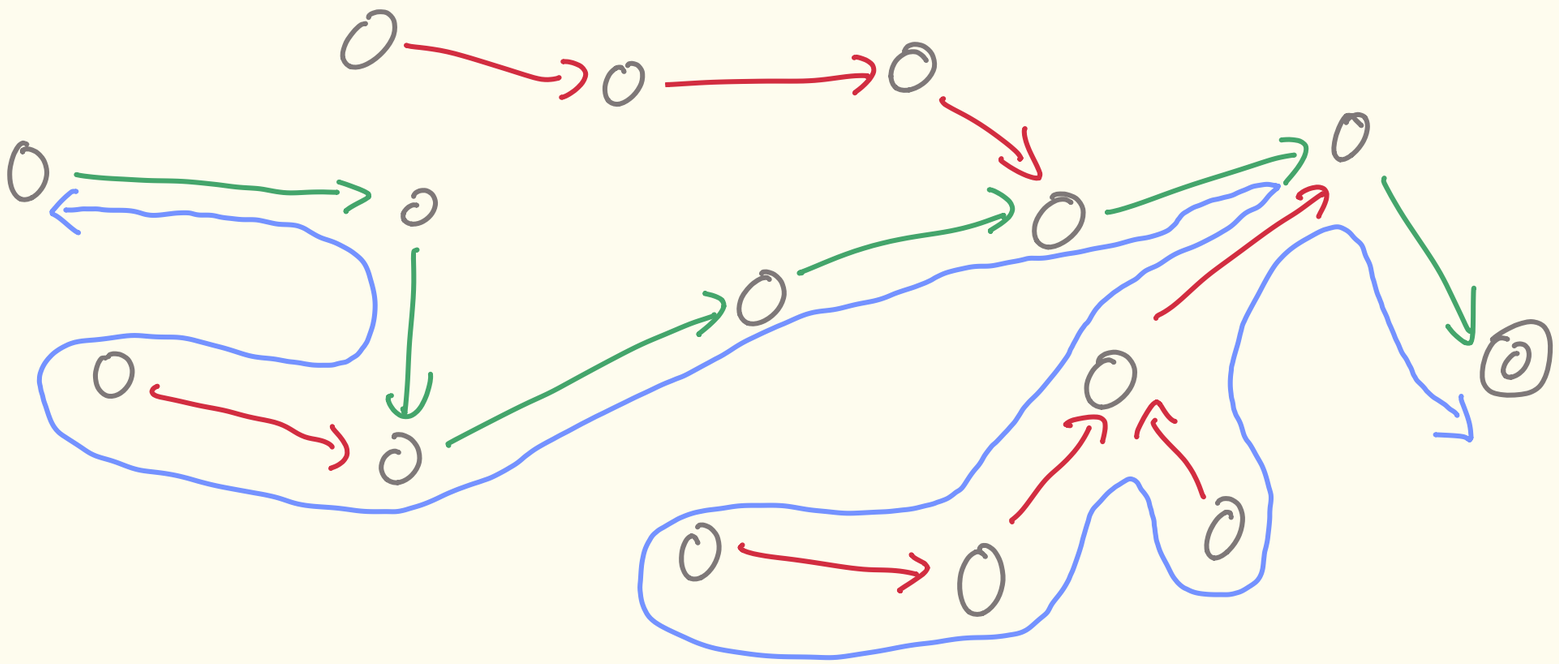
idea: Eulerian tour



Reversibility

[D'15]: catalytic space is reversible

idea: Eulerian tour



Reversibility

[D'15]: catalytic space is reversible

Q: can we use reversibility more directly when designing catalytic algorithms?

REVERSIBILITY

[D'15]: catalytic space is reversible

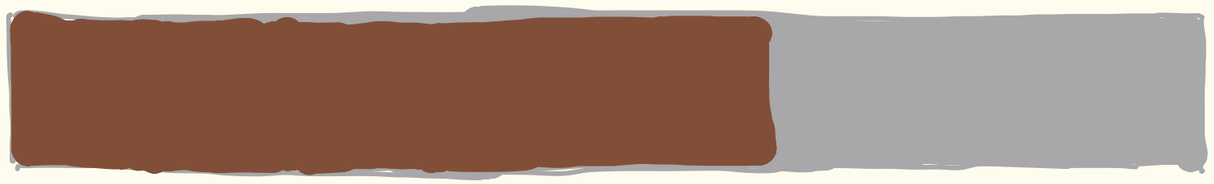
Q: can we use reversibility more directly when designing catalytic algorithms?

A: YES, via arithmetic

Reusing Space

Given: Want:

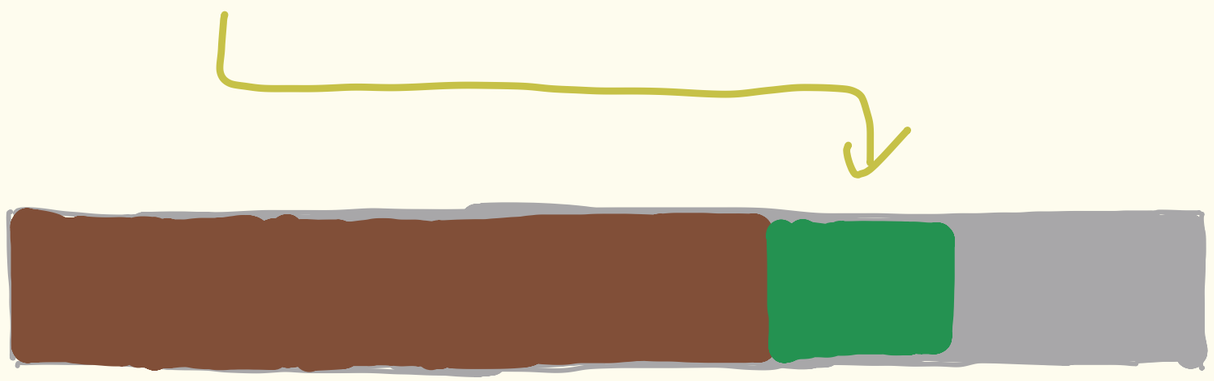
x $f(x)$



Reusing Space

Given: Want:

x $f(x)$



Input

Reusing Space

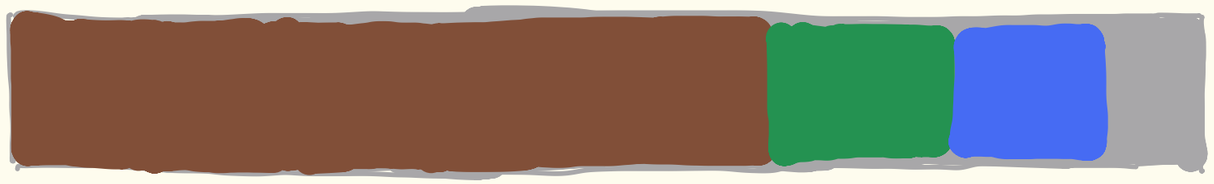
Given: Want:

Computation

Reusing Space

Given: Want:

x $f(x)$

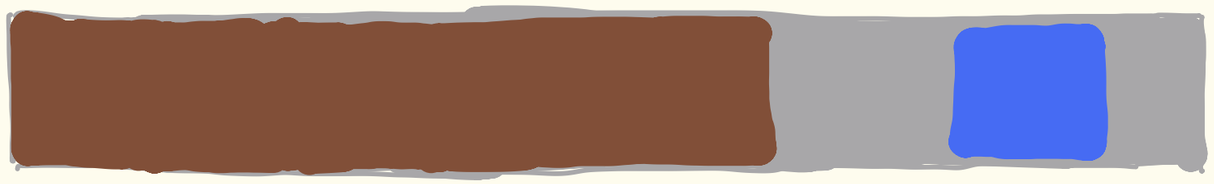


Output

Reusing Space

Given: Want:

x $f(x)$

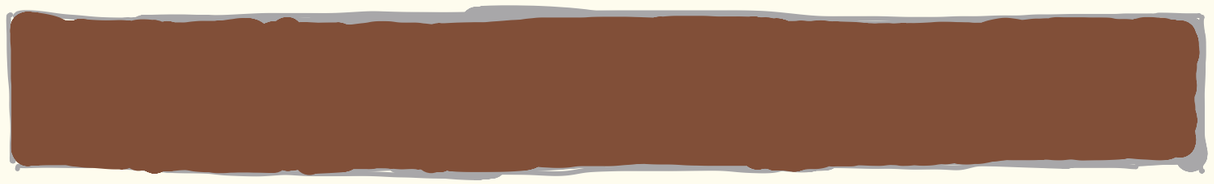


Reset

Reusing Space

Given: Want:

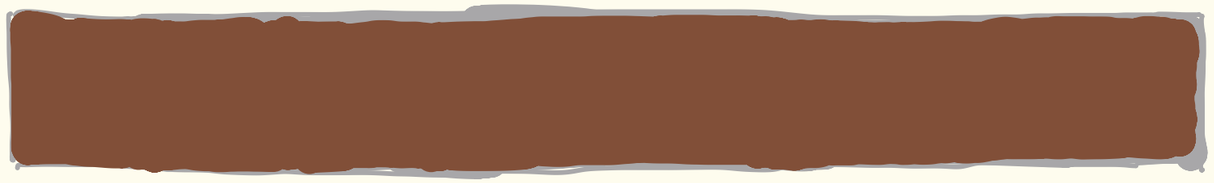
x $f(x)$



Reusing Space

Given: Want:

x $f(x)$

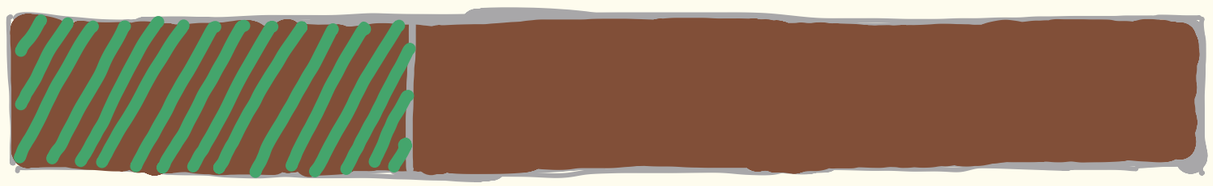


Idea 1: arithmetic is reversible

Reusing Space

Given: Want:

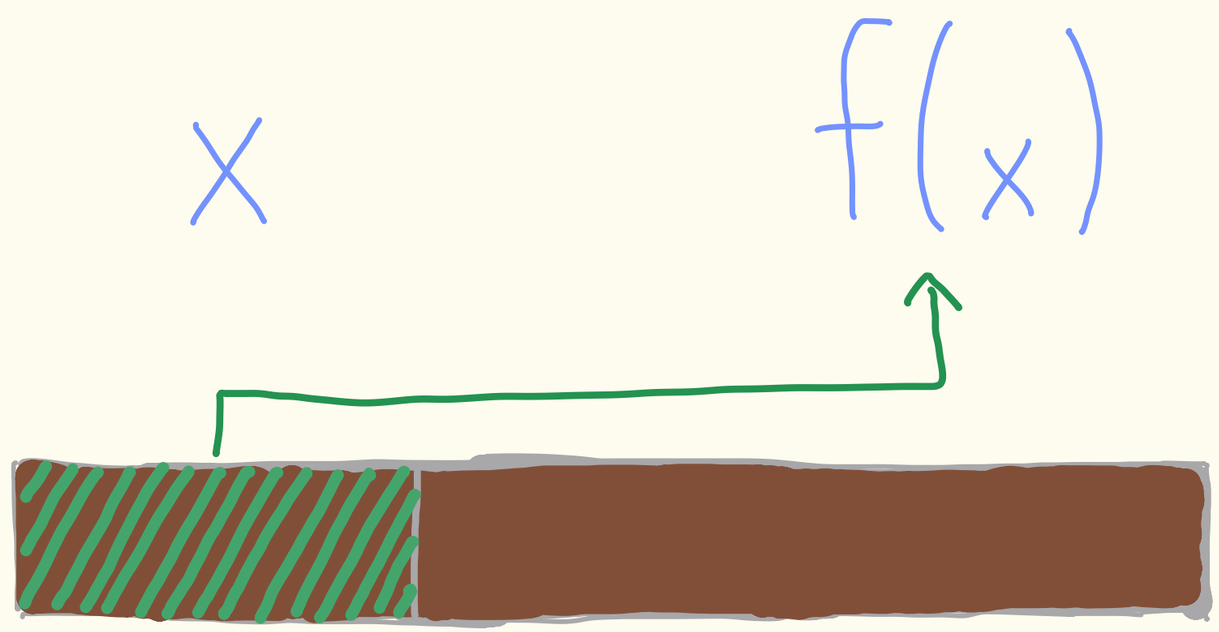
$+x$ $f(x)$



Input

Reusing Space

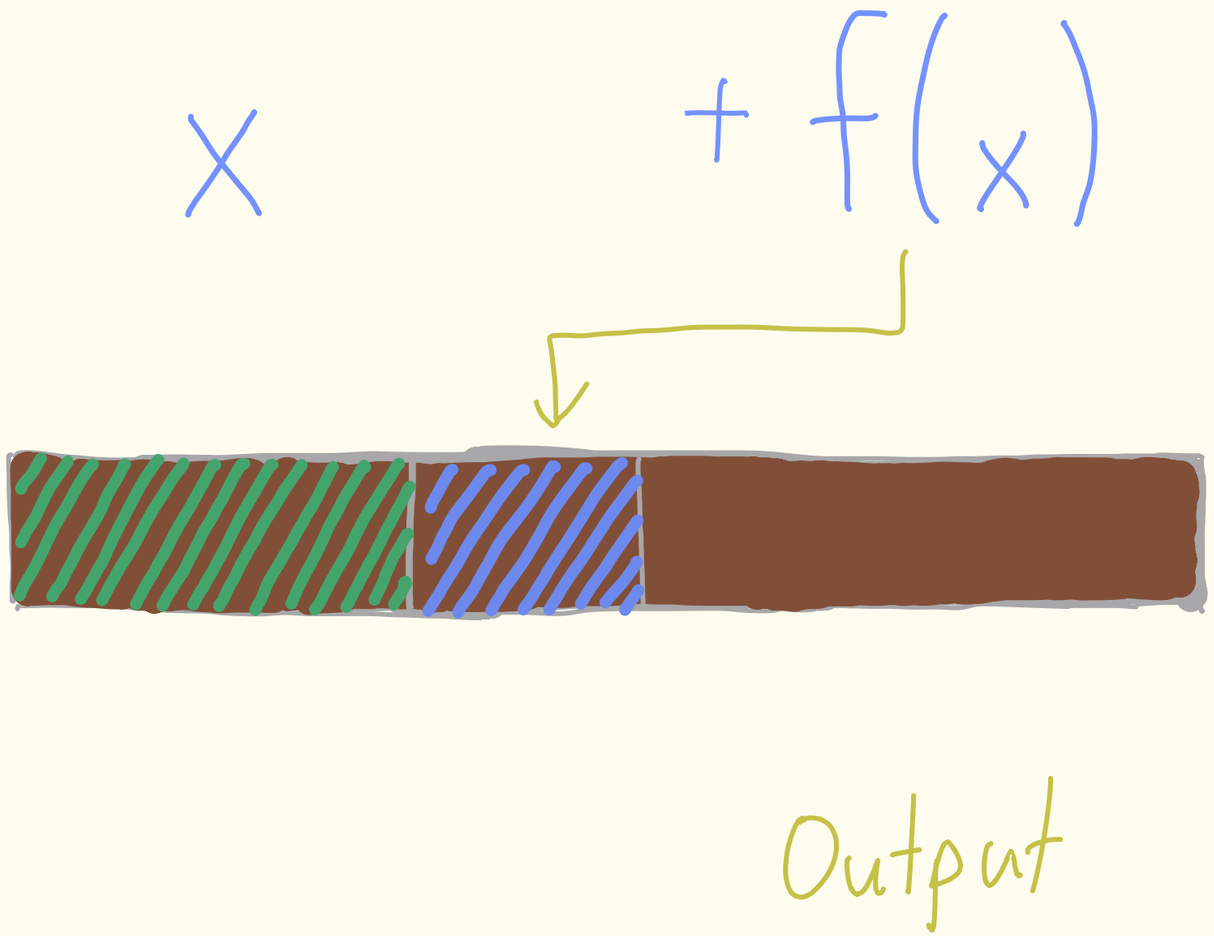
Given: Want:



Computation

Reusing Space

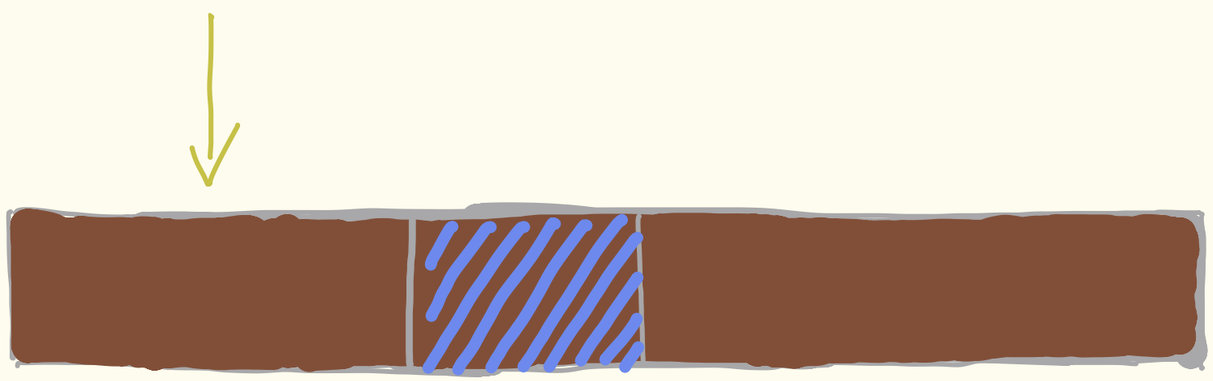
Given: Want:



Reusing Space

Given: Want:

$-x$ $f(x)$

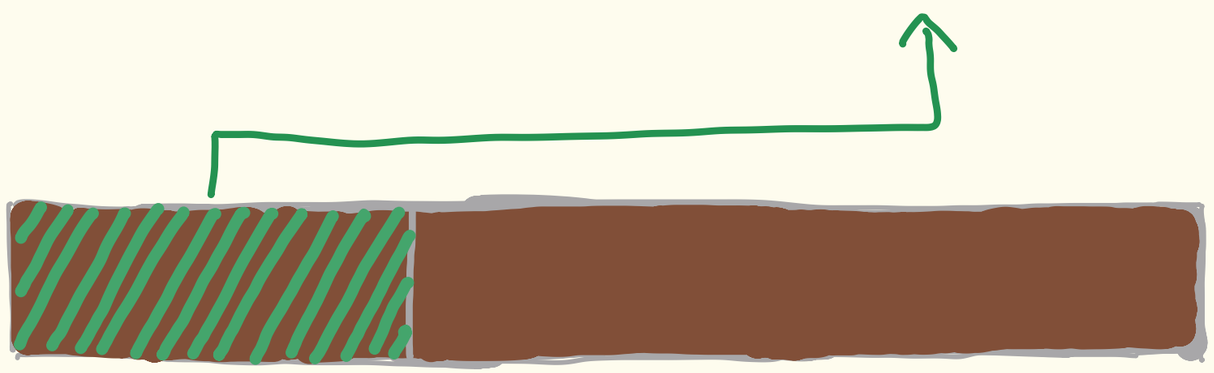


Reset

Reusing Space

Given: Want:

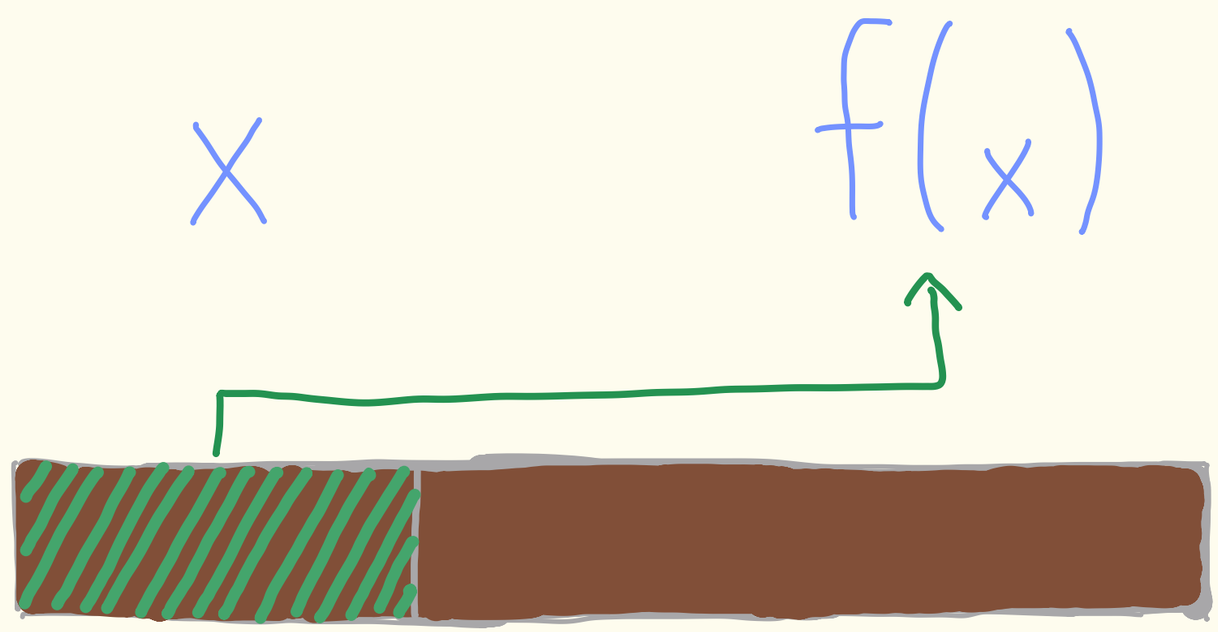
x $f(x)$



Computation ("the magic step")

Reusing Space

Given: Want:

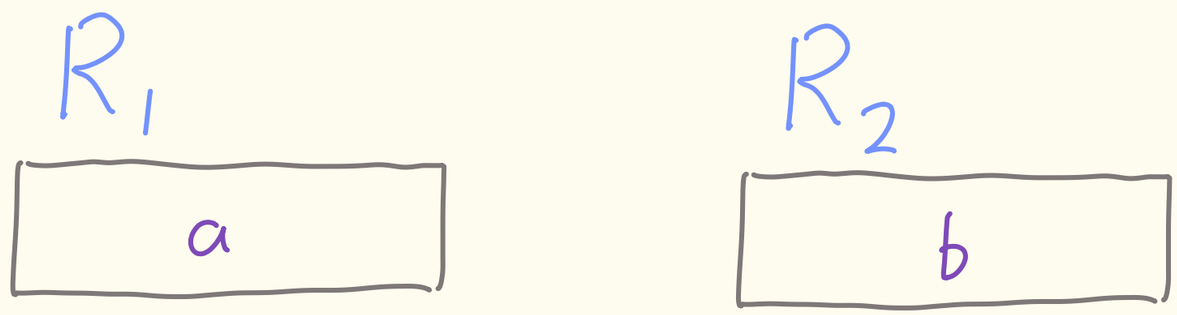


Idea 2: arithmetic allows cancellations

Arithmetic Magic

Example 1: swapping memory

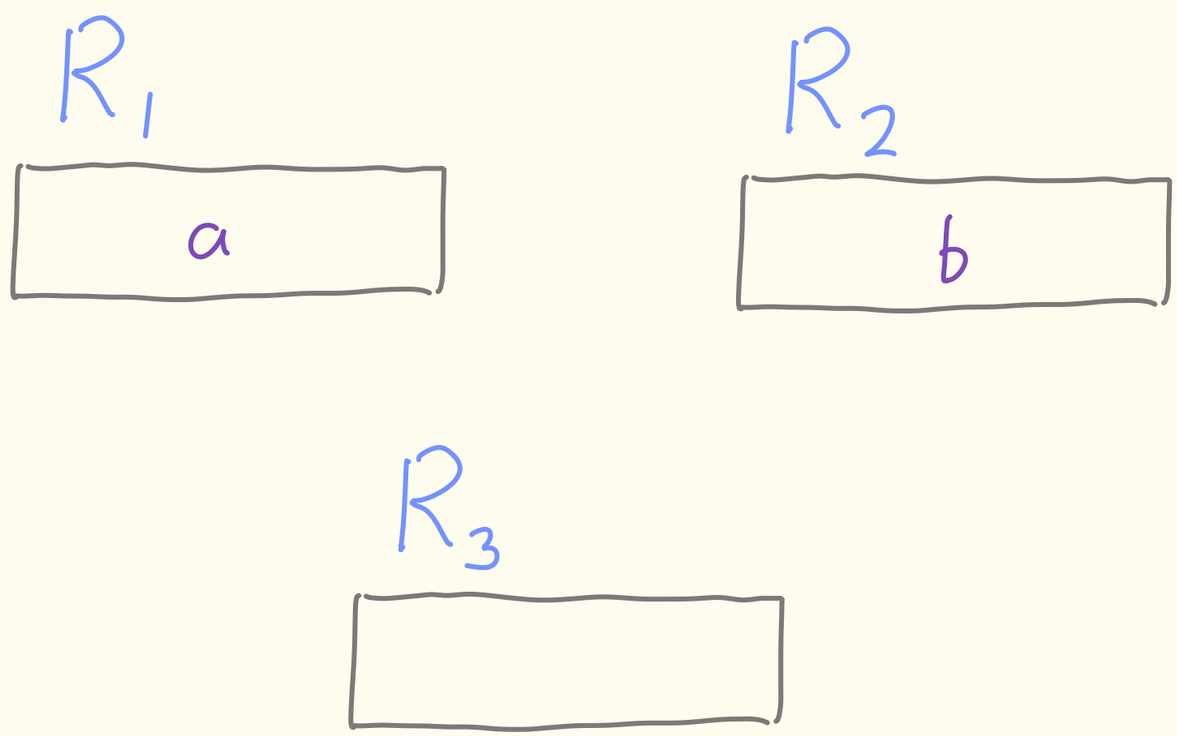
Given:



Want:

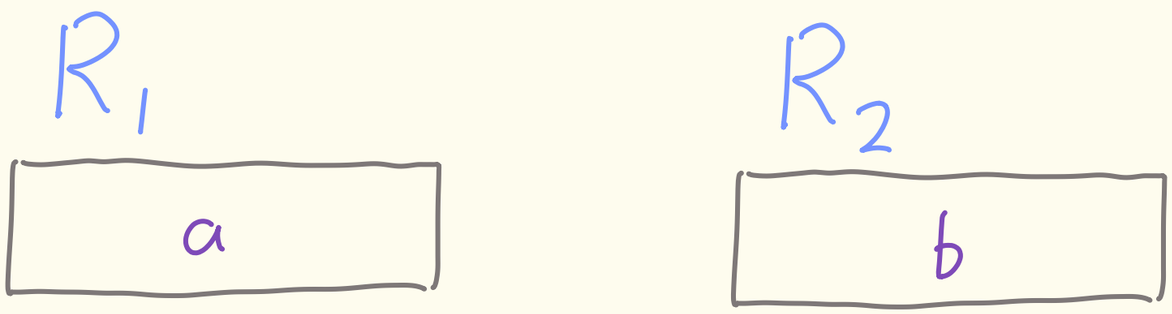
Arithmetic Magic

Example 1: swapping memory

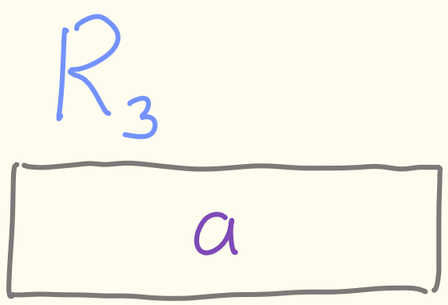


Arithmetic Magic

Example 1: swapping memory

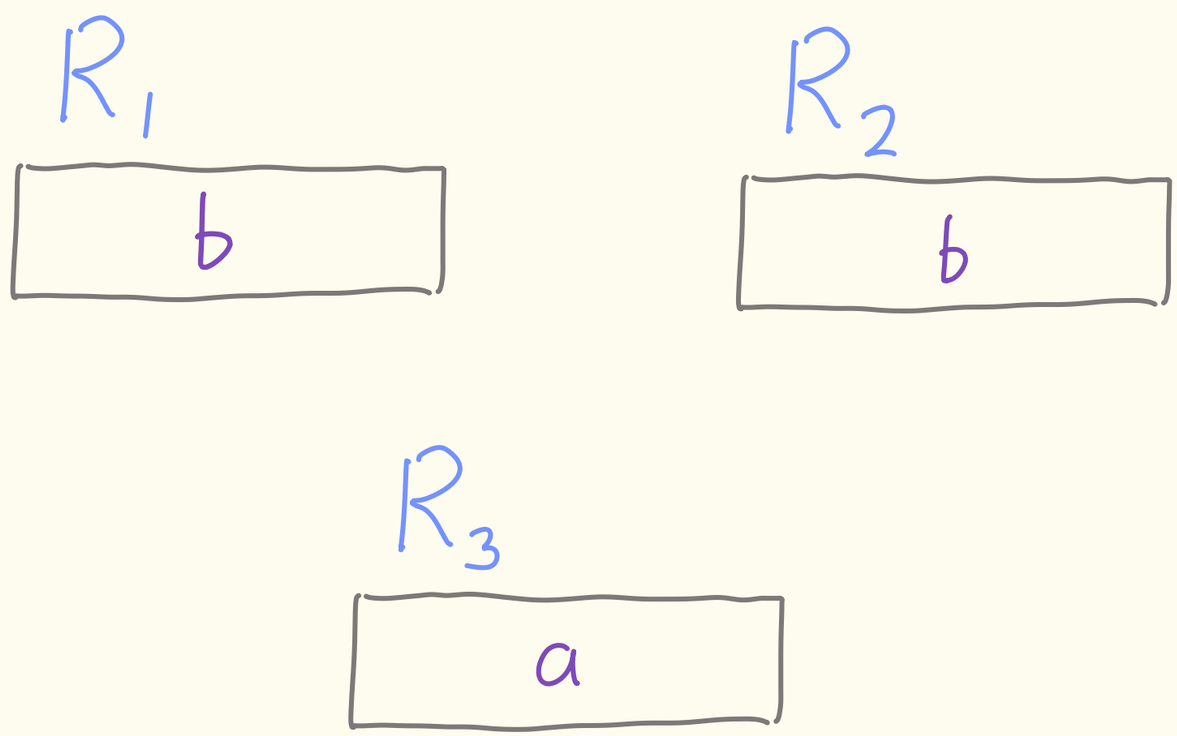


1. $R_3 = R_1$



Arithmetic Magic

Example 1: swapping memory

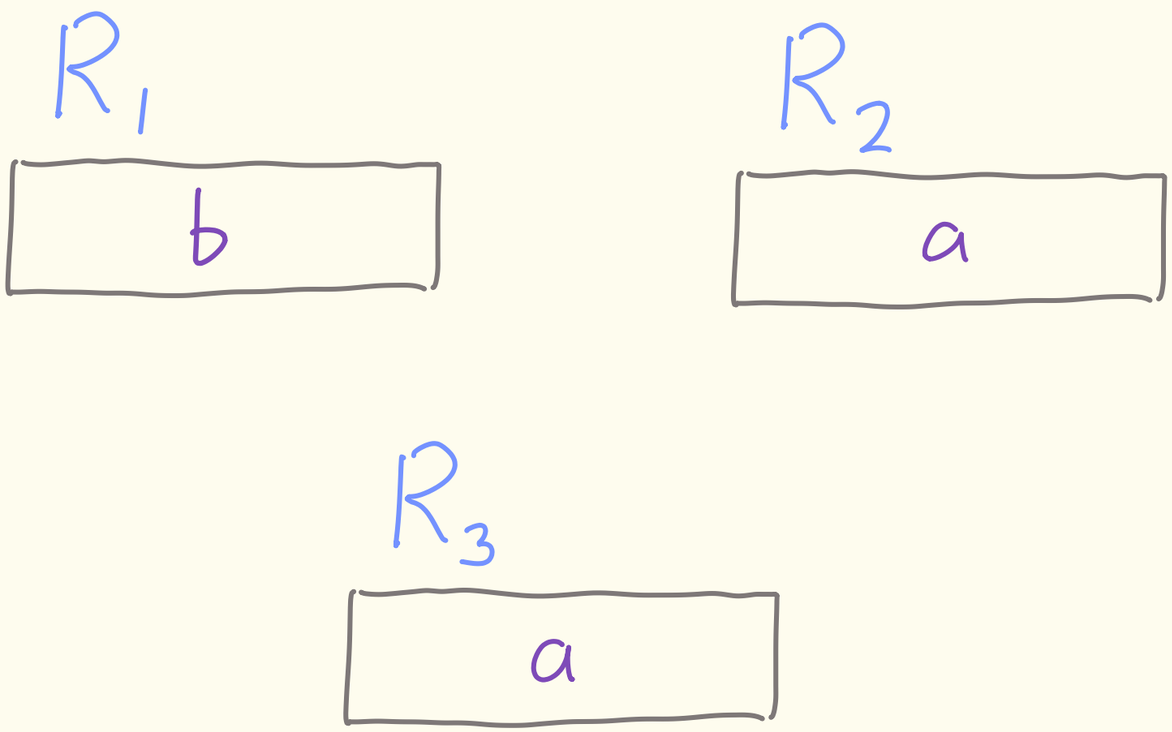


1. $R_3 = R_1$

2. $R_1 = R_2$

Arithmetic Magic

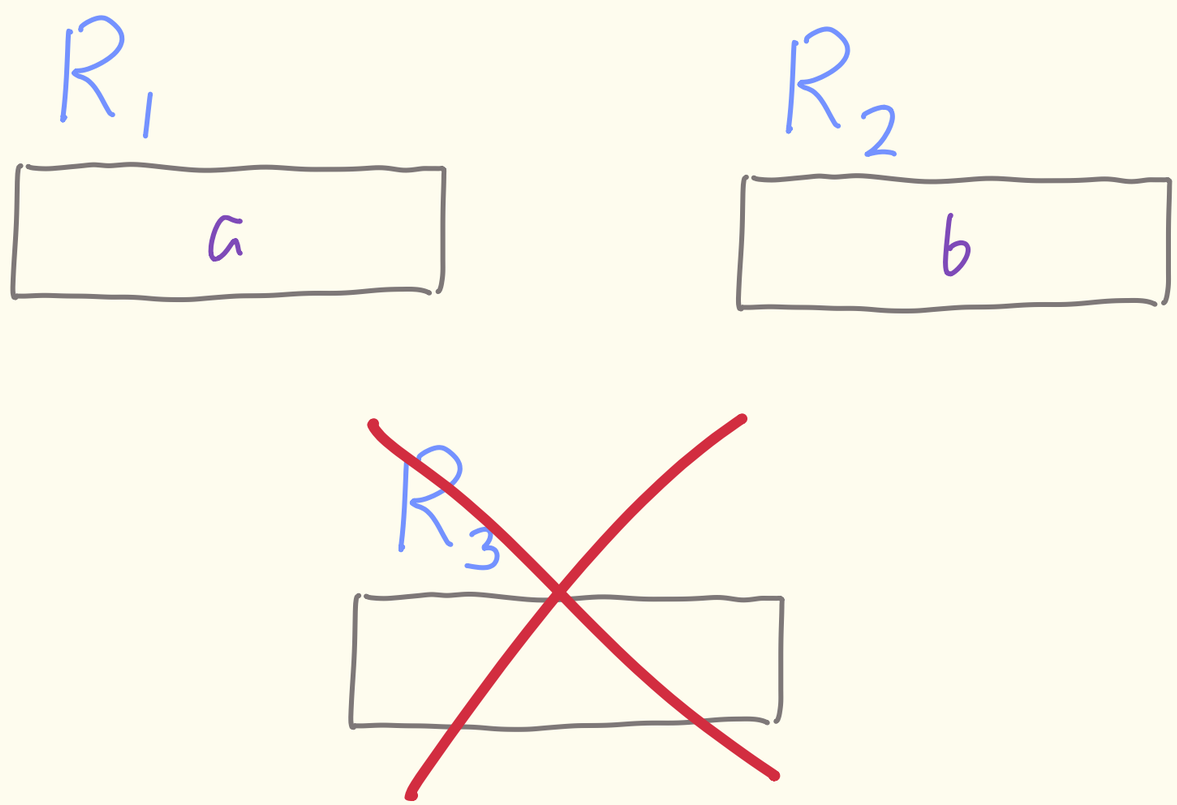
Example 1: swapping memory



1. $R_3 = R_1$
2. $R_1 = R_2$
3. $R_2 = R_3$

Arithmetic Magic

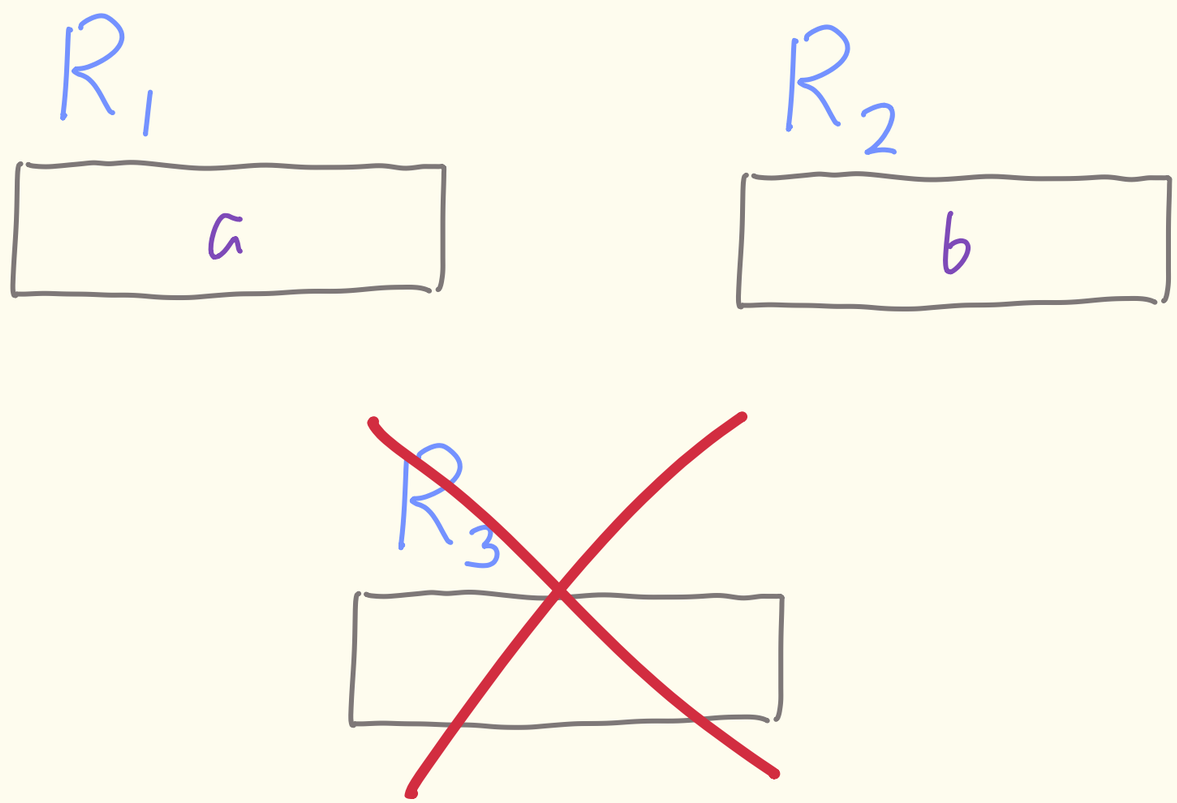
Example 1: swapping memory



Arithmetic Magic

for simplicity:
a, b numbers

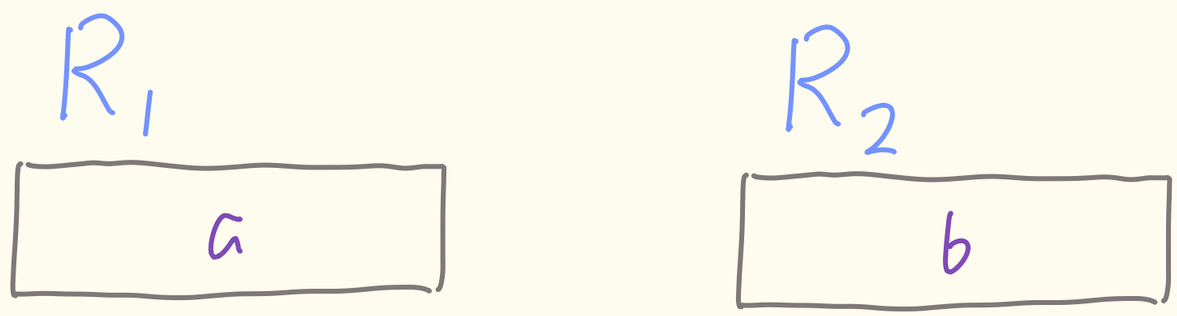
Example 1: swapping memory



Arithmetic Magic

for simplicity:
a, b numbers

Example 1: swapping memory



1. $R_1 = R_1 - R_2$

Arithmetic Magic

for simplicity:
a, b numbers

Example 1: swapping memory

R_1	R_2
$a-b$	b

1. $R_1 = R_1 - R_2$

Arithmetic Magic

for simplicity:
a, b numbers

Example 1: swapping memory

R_1	R_2
$a-b$	b

1. $R_1 = R_1 - R_2$
2. $R_2 = R_1 + R_2$

Arithmetic Magic

for simplicity:
a, b numbers

Example 1: swapping memory

R_1	R_2
$a-b$	$(a-b)+b$

1. $R_1 = R_1 - R_2$
2. $R_2 = R_1 + R_2$

Arithmetic Magic

for simplicity:
a, b numbers

Example 1: swapping memory

R_1	R_2
$a-b$	a

1. $R_1 = R_1 - R_2$
2. $R_2 = R_1 + R_2$

Arithmetic Magic

for simplicity:
a, b numbers

Example 1: swapping memory

R_1	R_2
$a-b$	a

1. $R_1 = R_1 - R_2$
2. $R_2 = R_1 + R_2$
3. $R_1 = R_2 - R_1$

Arithmetic Magic

for simplicity:
a, b numbers

Example 1: swapping memory

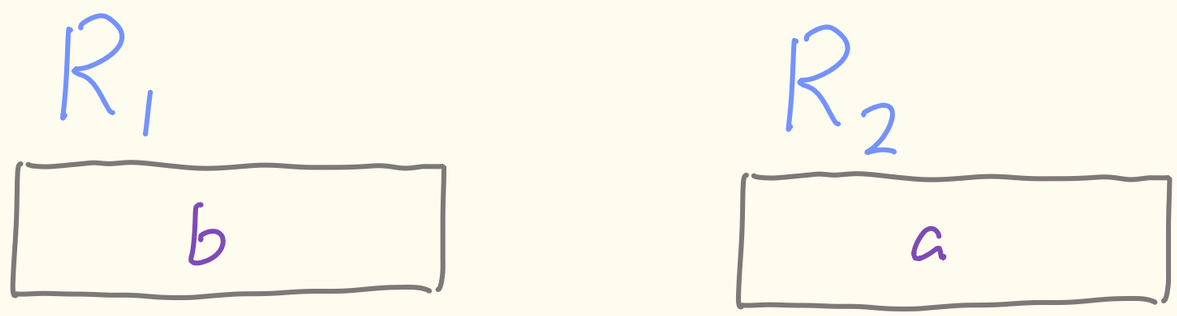
R_1	R_2
$a-(a-b)$	a

1. $R_1 = R_1 - R_2$
2. $R_2 = R_1 + R_2$
3. $R_1 = R_2 - R_1$

Arithmetic Magic

for simplicity:
a, b numbers

Example 1: swapping memory



1. $R_1 = R_1 - R_2$
2. $R_2 = R_1 + R_2$
3. $R_1 = R_2 - R_1$

Arithmetic Magic

Example 2: Inner Product

Given $x, y \in \{0,1\}^n$,
compute $\langle x, y \rangle \bmod 2$
with only two passes.

Arithmetic Magic

Example 2: Inner Product

Given $x, y \in \{0,1\}^n$,
compute $\langle x, y \rangle \bmod 2$
with only two passes.

$s = 1$

$x_1 \quad y_1 \quad x_2 \quad y_2 \quad \ldots$

$\boxed{\sum x_i y_i}$

Arithmetic Magic

Example 2: Inner Product

Given $x, y \in \{0,1\}^n$,
compute $\langle x, y \rangle \bmod 2$
with only two passes.

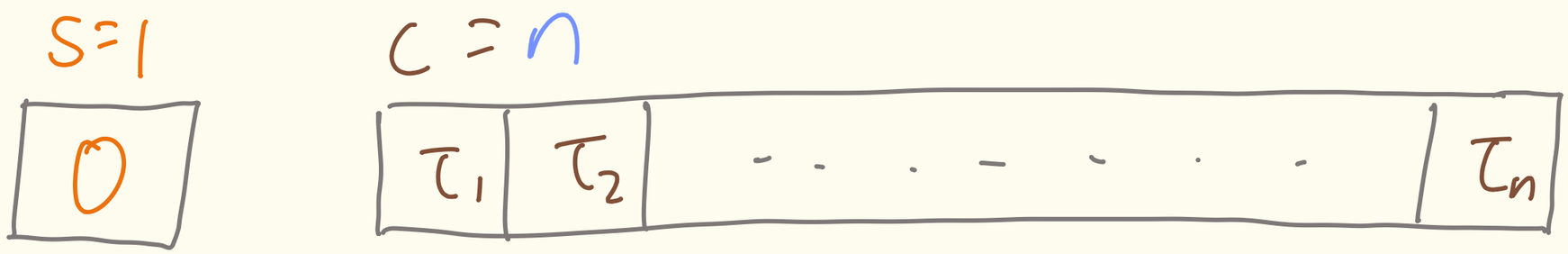
$s = n/3 + 1$

$x \quad y \quad x \quad y$

$x_1 \; \cdots \; x_{n/3}$	$\sum x_i y_i$

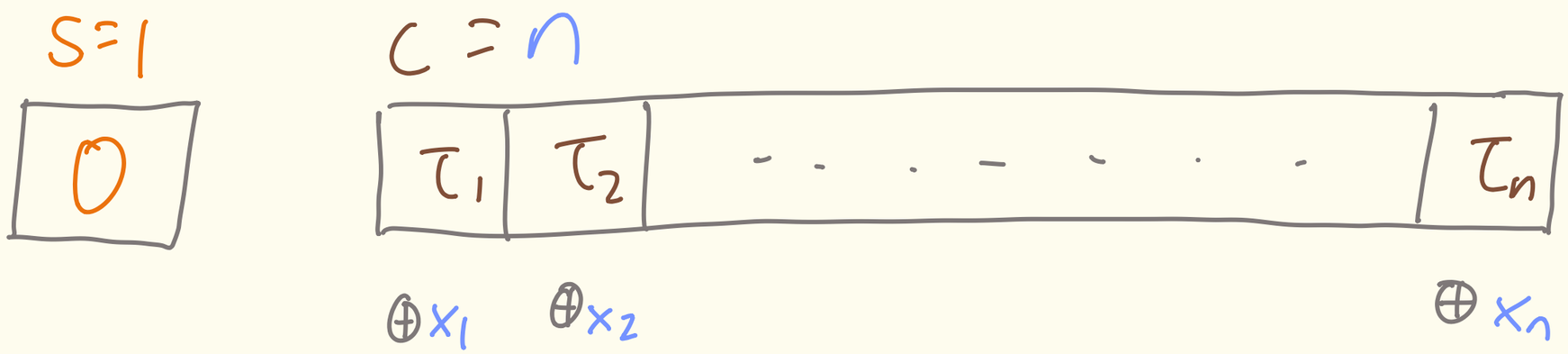
Arithmetic Magic

Example 2: Inner Product



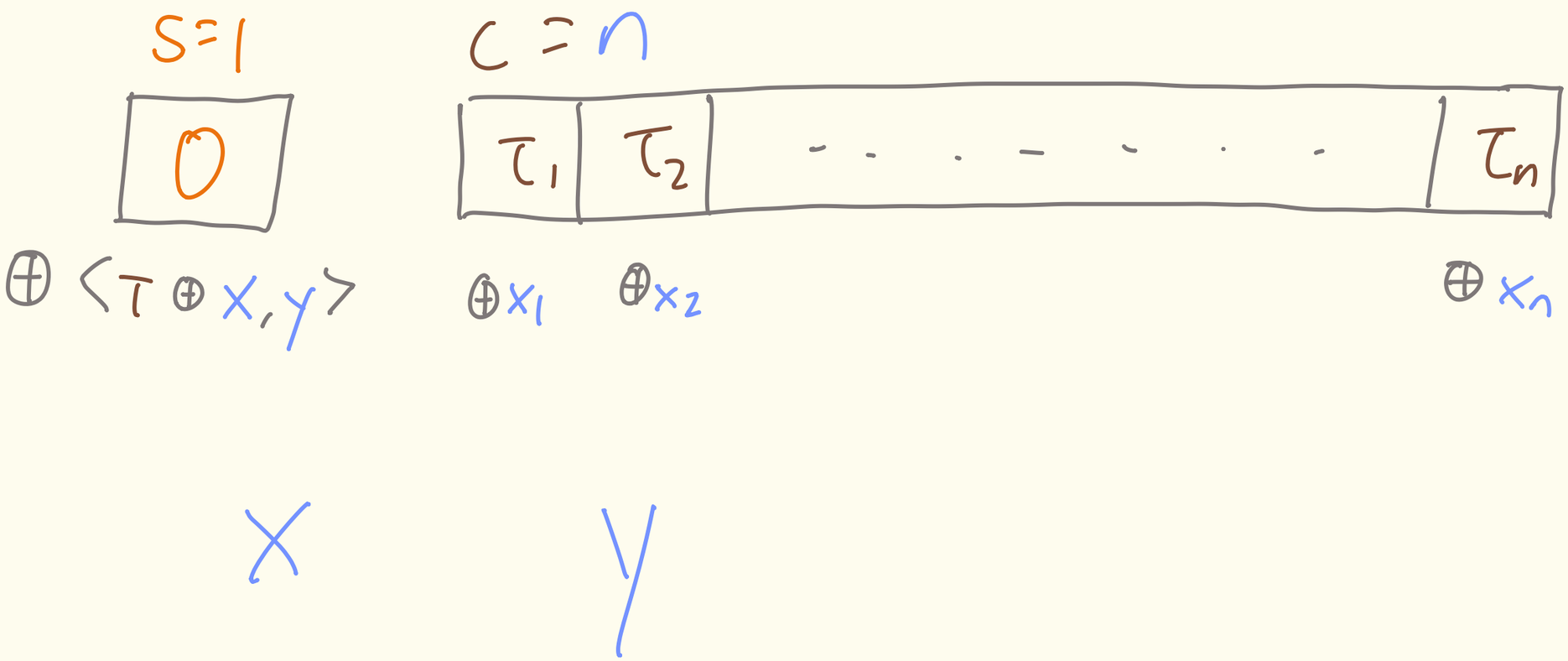
Arithmetic Magic

Example 2: Inner Product



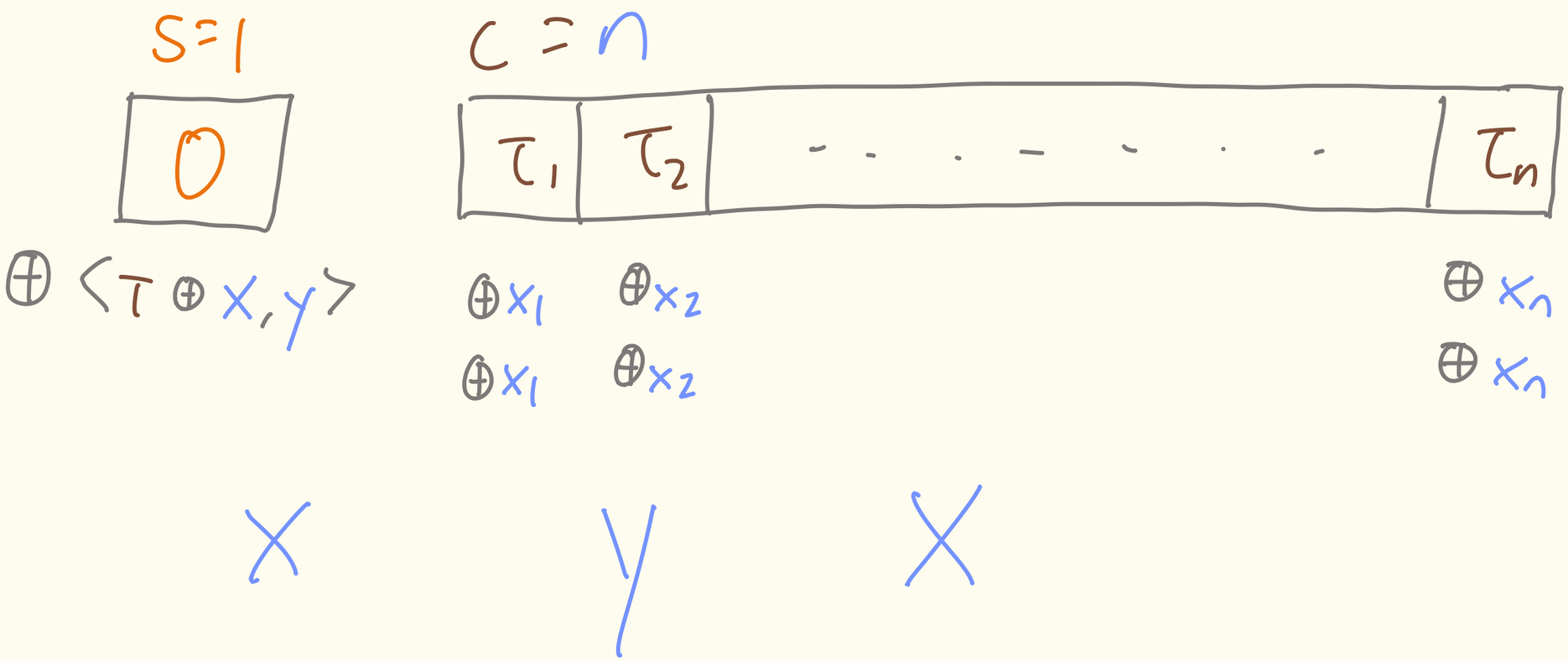
Arithmetic Magic

Example 2: Inner Product



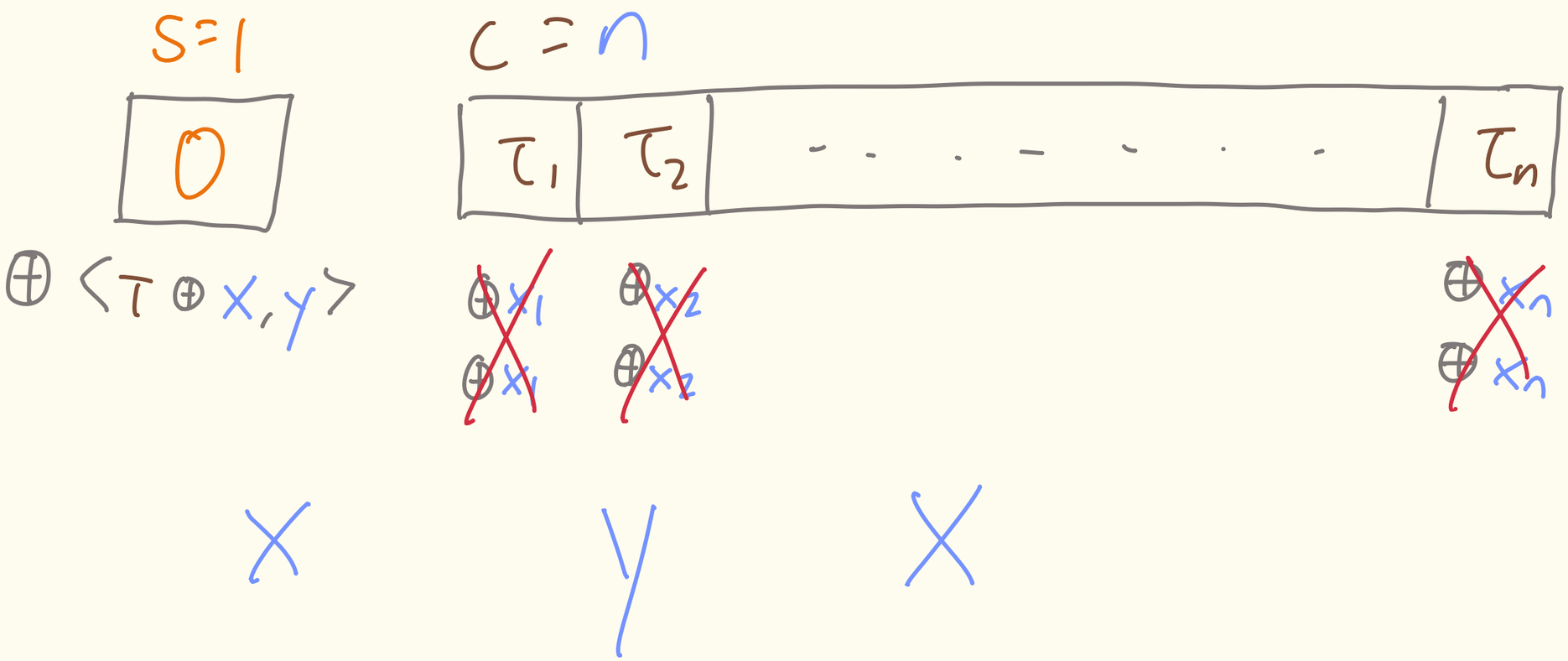
Arithmetic Magic

Example 2: Inner Product



Arithmetic Magic

Example 2: Inner Product



Arithmetic Magic

Example 2: Inner Product

$s = 1$ $\qquad$ $c = n$

0	τ_1	τ_2	$\cdots$	τ_n

$\oplus \langle \tau \oplus x, y \rangle$

$\oplus \langle \tau, y \rangle$

$x \qquad y \qquad x \qquad y$

Arithmetic Magic

Example 2: Inner Product

$s = 1$ $\qquad$ $c = n$

0	τ_1	τ_2	. . . . .	τ_n

$$\oplus \langle \tau \oplus x, y \rangle \oplus \langle \tau, y \rangle = \langle \tau, y \rangle \oplus \langle x, y \rangle \oplus \langle \tau, y \rangle$$

x $\qquad$ y $\qquad$ x $\qquad$ y

Arithmetic Magic

Example 2: Inner Product

$s = 1$ $\qquad$ $c = n$

$$\boxed{\langle x, y \rangle} \qquad \boxed{\tau_1 \mid \tau_2 \mid \cdots \cdots \mid \tau_n}$$

$$= \cancel{\langle \tau, y \rangle} \oplus \langle x, y \rangle \oplus \cancel{\langle \tau, y \rangle}$$

x $\qquad$ y $\qquad$ x $\qquad$ y

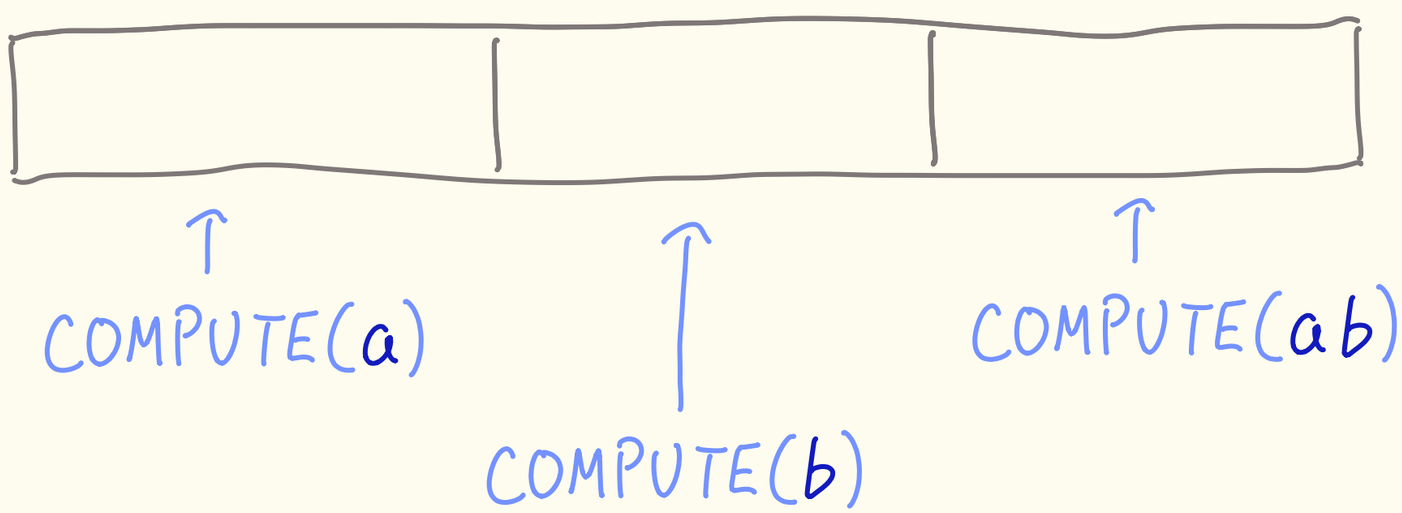
$\blacksquare$

Arithmetic Magic

Example 3: Multiplication

Given programs COMPUTE(a), COMPUTE(b)

design program COMPUTE($a \cdot b$)



Arithmetic Magic

Example 3: Multiplication

COMPUTE(a), COMPUTE(b)

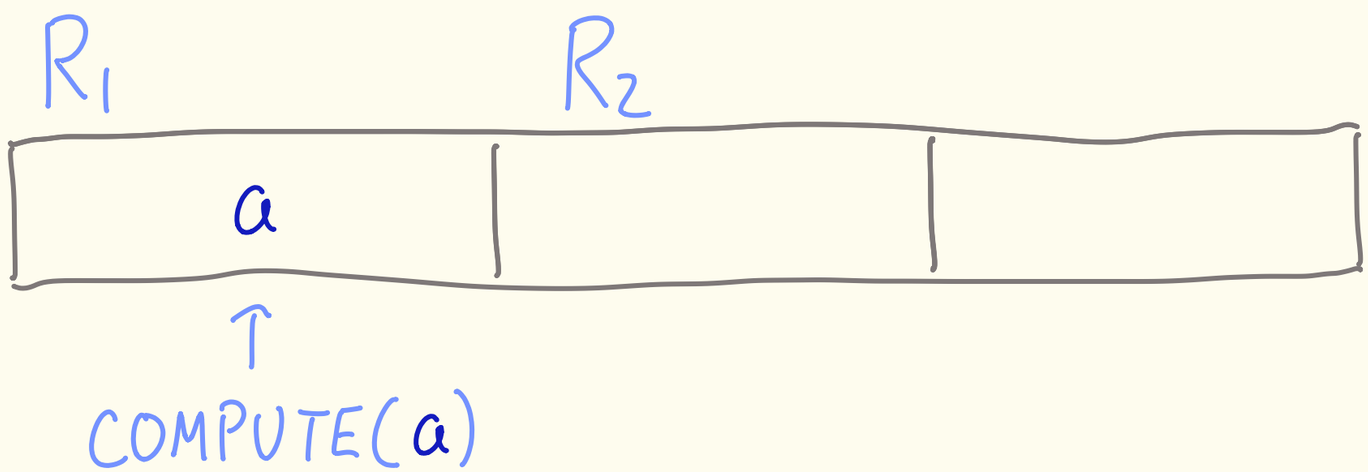
ordinary computation: → COMPUTE($a \cdot b$)

Arithmetic Magic

Example 3: Multiplication

ordinary computation:

COMPUTE(a), COMPUTE(b) $\rightarrow$ COMPUTE($a \cdot b$)

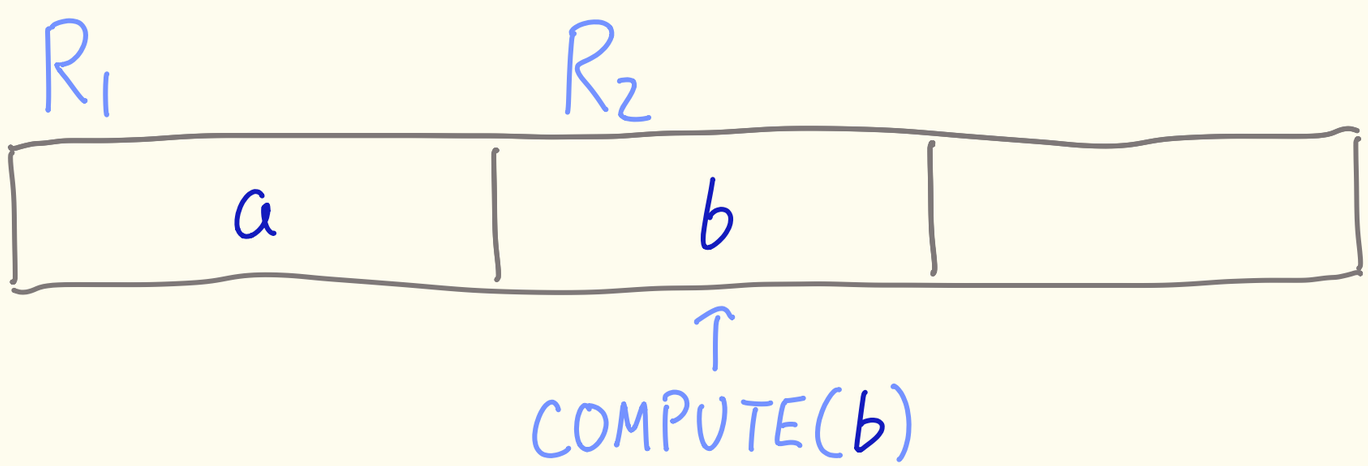


Arithmetic Magic

Example 3: Multiplication

ordinary computation:

COMPUTE(a), COMPUTE(b) → COMPUTE($a \cdot b$)

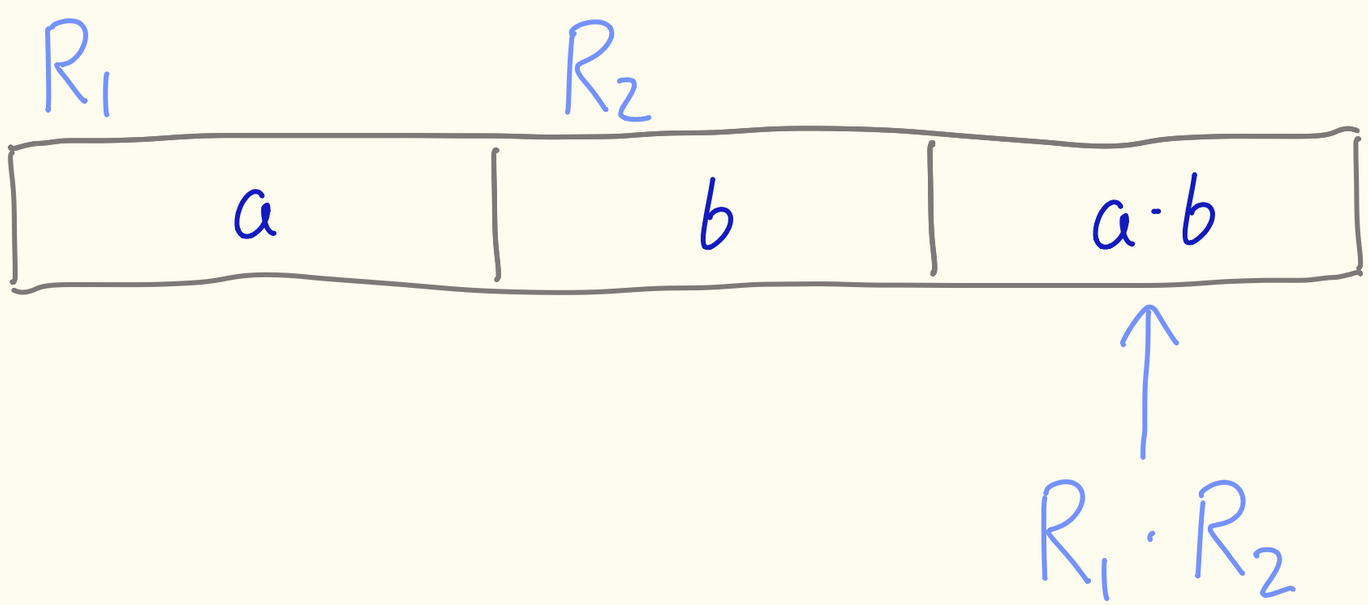


Arithmetic Magic

Example 3: Multiplication

ordinary computation:

COMPUTE(a), COMPUTE(b) → COMPUTE($a \cdot b$)

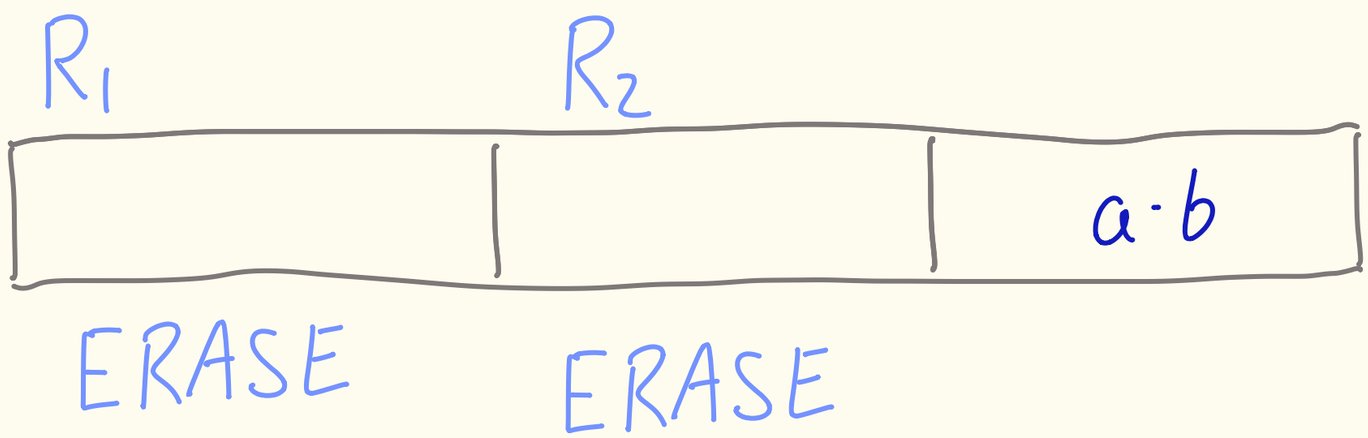


Arithmetic Magic

Example 3: Multiplication

ordinary computation:

COMPUTE(a), COMPUTE(b) → COMPUTE($a \cdot b$)



Arithmetic Magic

Example 3: Multiplication

COMPUTE(a), COMPUTE(b)

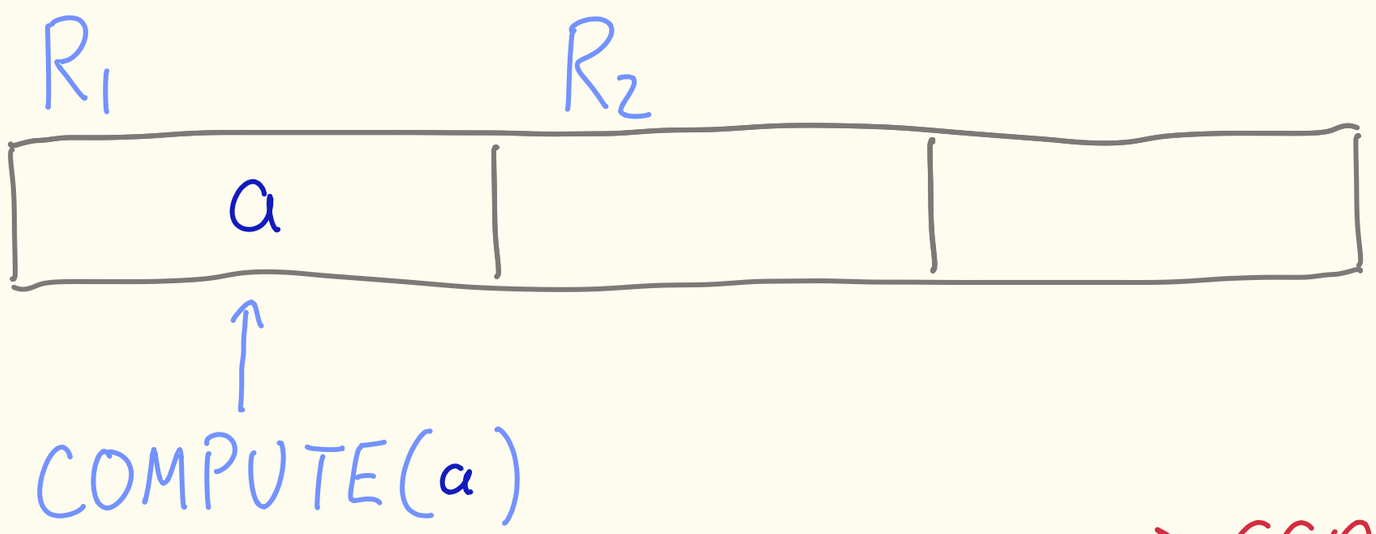
catalytic computation: → COMPUTE($a \cdot b$)

Arithmetic Magic

Example 3: Multiplication

COMPUTE(a), COMPUTE(b)

catalytic computation: → COMPUTE($a \cdot b$)



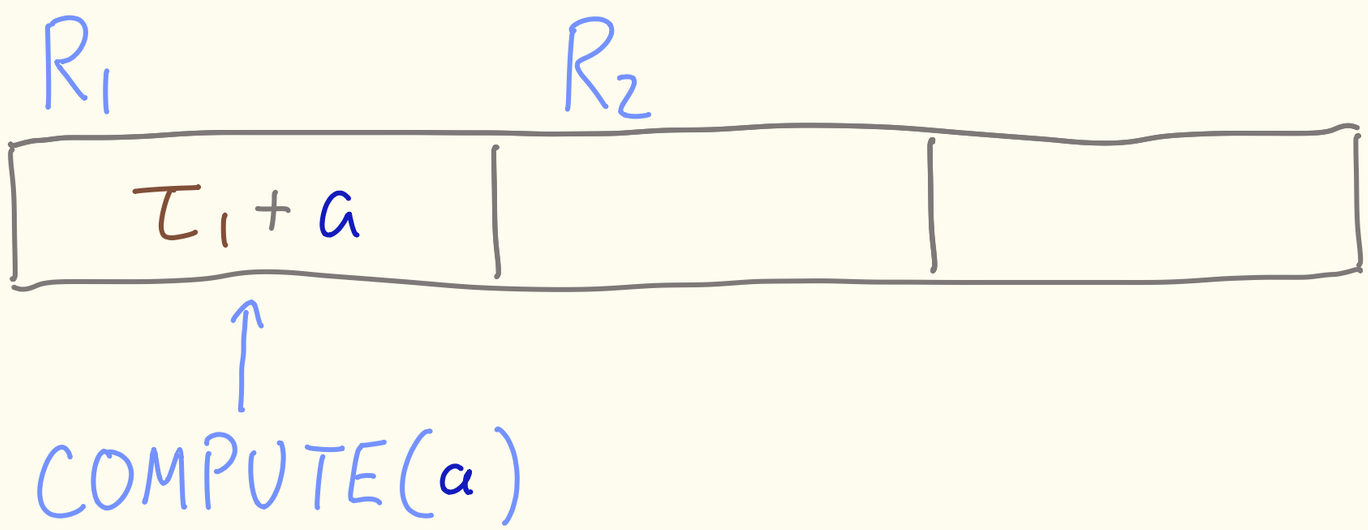
→ cannot reset

Arithmetic Magic

Example 3: Multiplication

COMPUTE(a), COMPUTE(b)

catalytic computation: → COMPUTE($a \cdot b$)

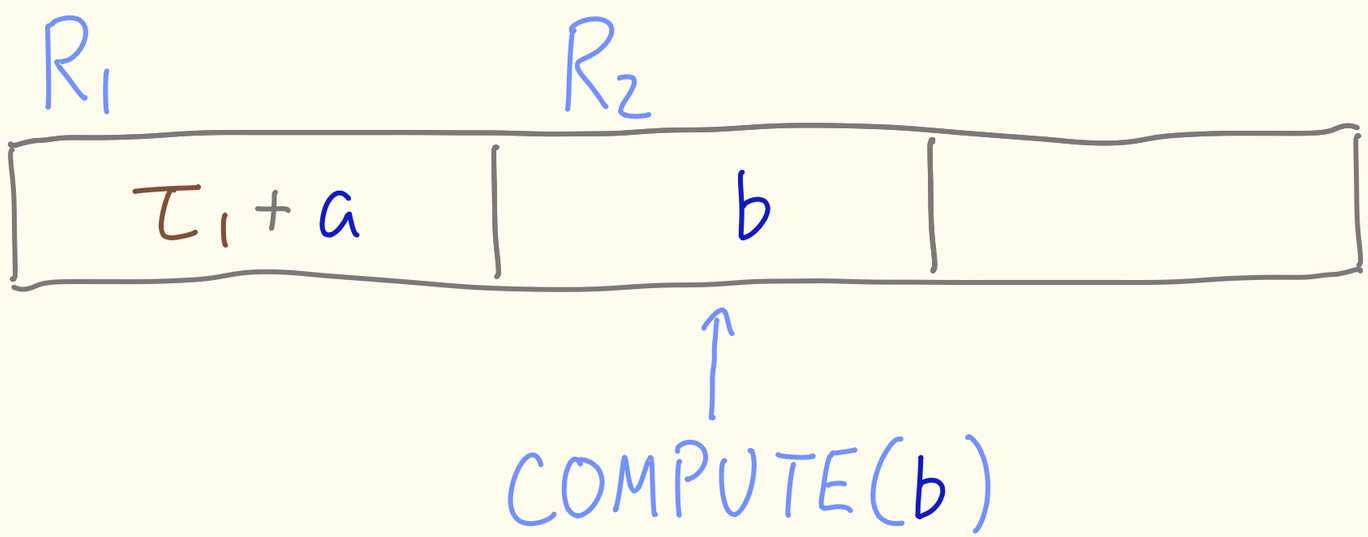


Arithmetic Magic

Example 3: Multiplication

COMPUTE(a), COMPUTE(b)

catalytic computation: → COMPUTE($a \cdot b$)

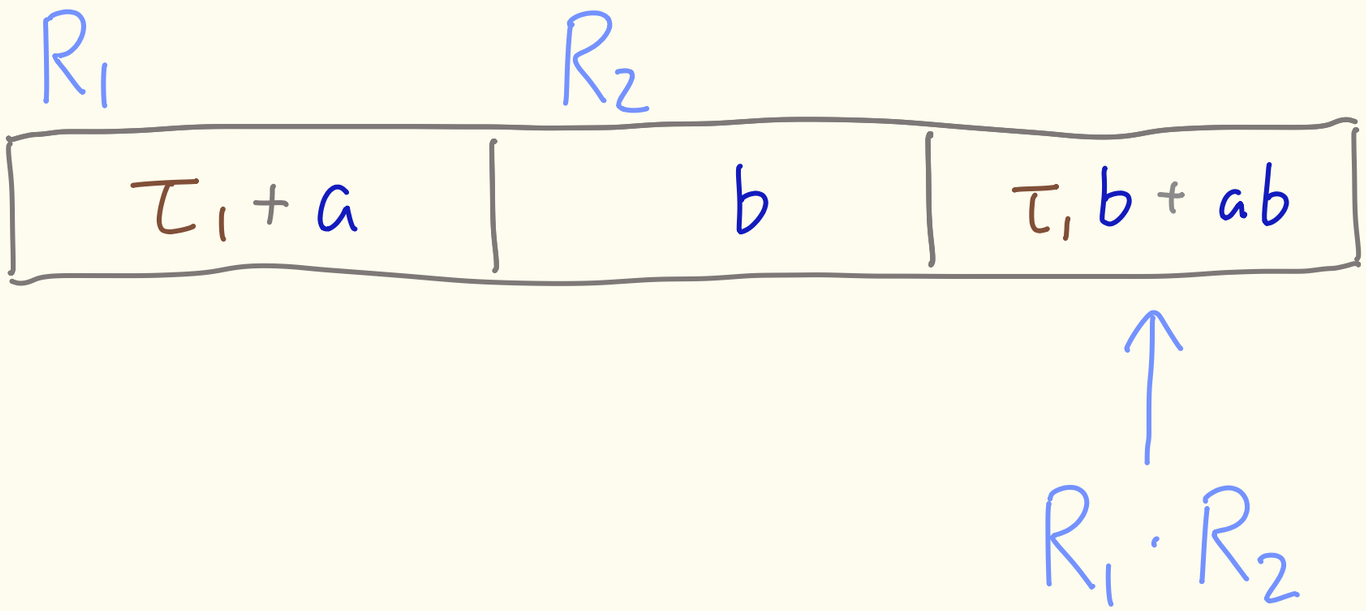


Arithmetic Magic

Example 3: Multiplication

COMPUTE(a), COMPUTE(b)

catalytic computation: → COMPUTE($a \cdot b$)



Arithmetic Magic

Example 3: Multiplication

COMPUTE(a), COMPUTE(b)

catalytic computation: → COMPUTE($a \cdot b$)

R_1	R_2	
$\tau_1 + a$	b	$\tau_1 b + ab$

Arithmetic Magic

Example 3: Multiplication

COMPUTE(a), COMPUTE(b)

catalytic computation: → COMPUTE($a \cdot b$)

R_1	R_2	
$\tau_1 + \cancel{a}$	b	$\tau_1 b + ab$

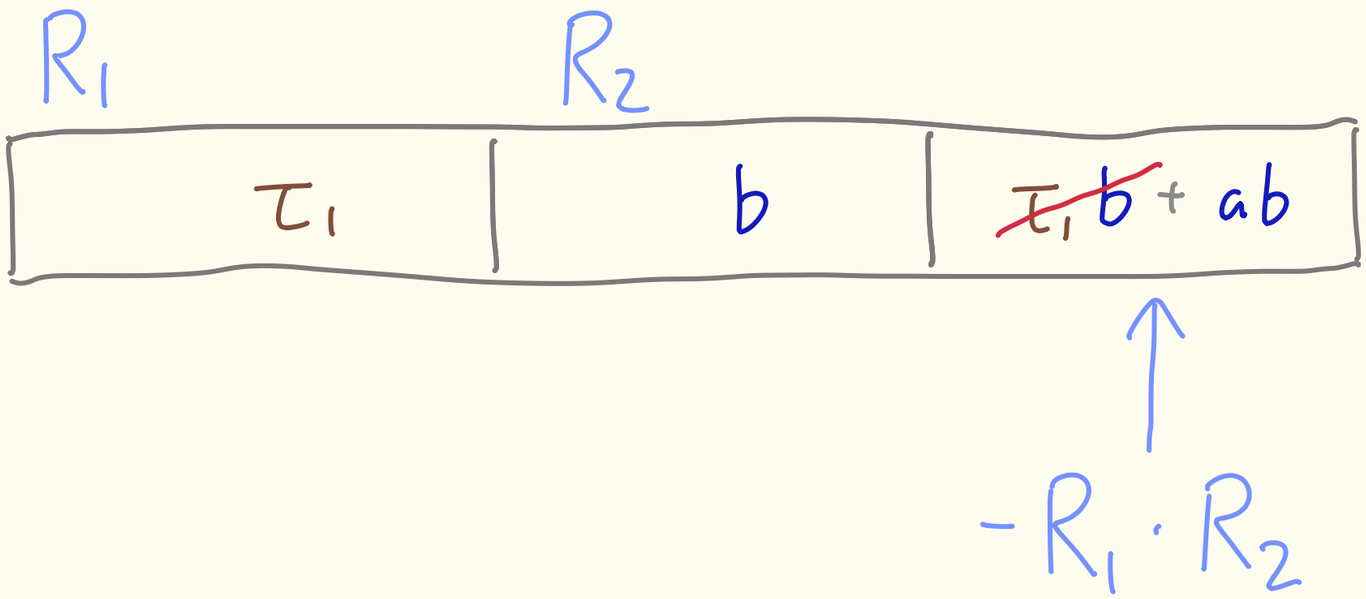
↑ COMPUTE($-a$)

Arithmetic Magic

Example 3: Multiplication

COMPUTE(a), COMPUTE(b)

catalytic computation: → COMPUTE($a \cdot b$)

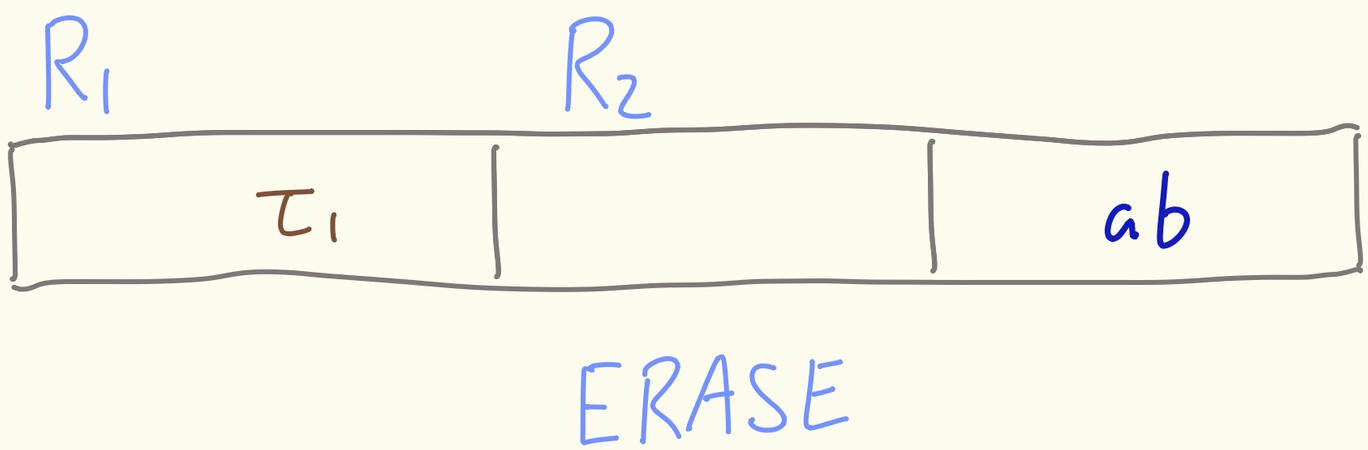


Arithmetic Magic

Example 3: Multiplication

COMPUTE(a), COMPUTE(b)

catalytic computation: → COMPUTE($a \cdot b$)

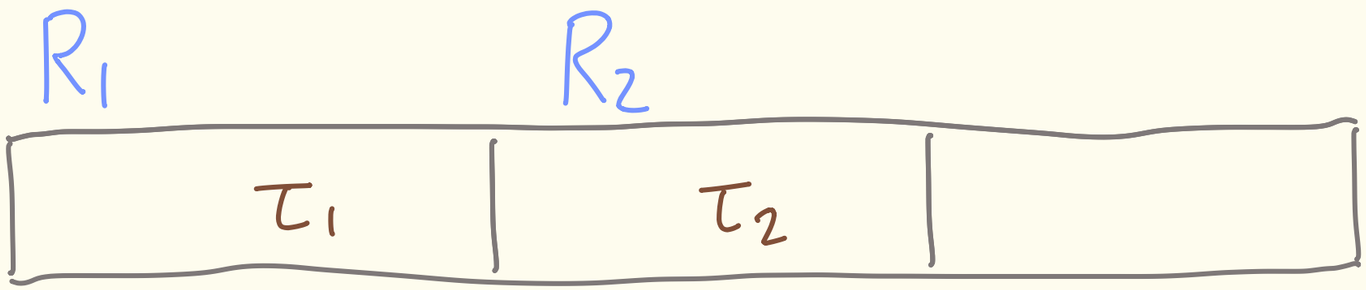


Arithmetic Magic

Example 3: Multiplication

COMPUTE(a), COMPUTE(b)

catalytic computation: → COMPUTE($a \cdot b$)

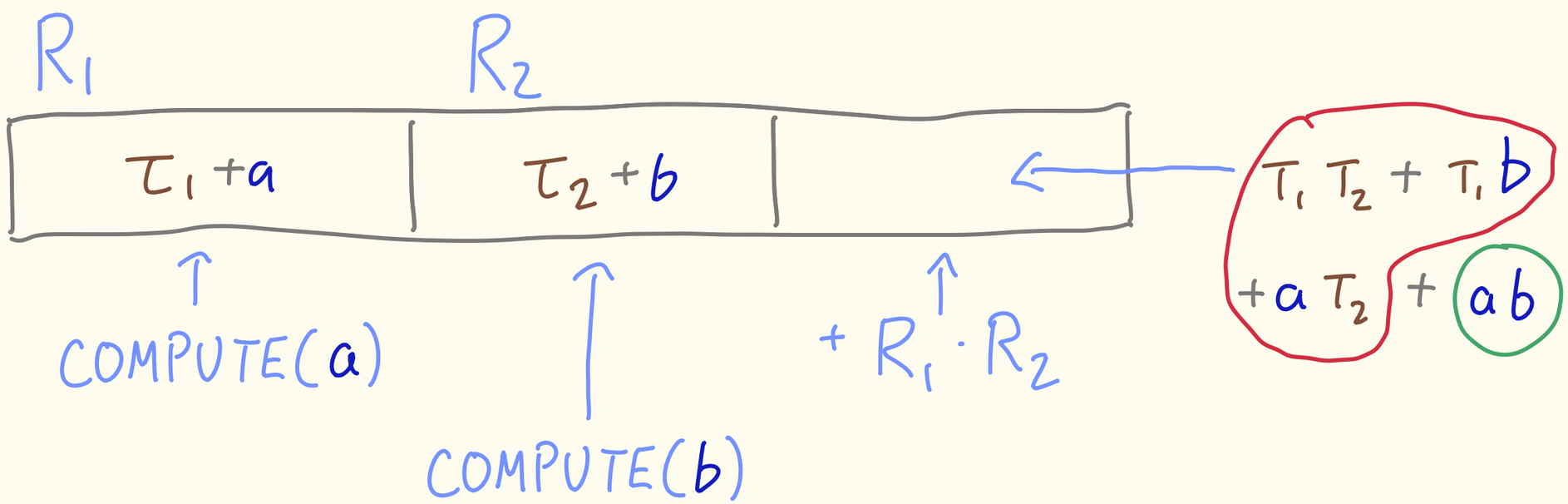


Arithmetic Magic

Example 3: Multiplication

COMPUTE(a), COMPUTE(b)

catalytic computation: → COMPUTE($a \cdot b$)

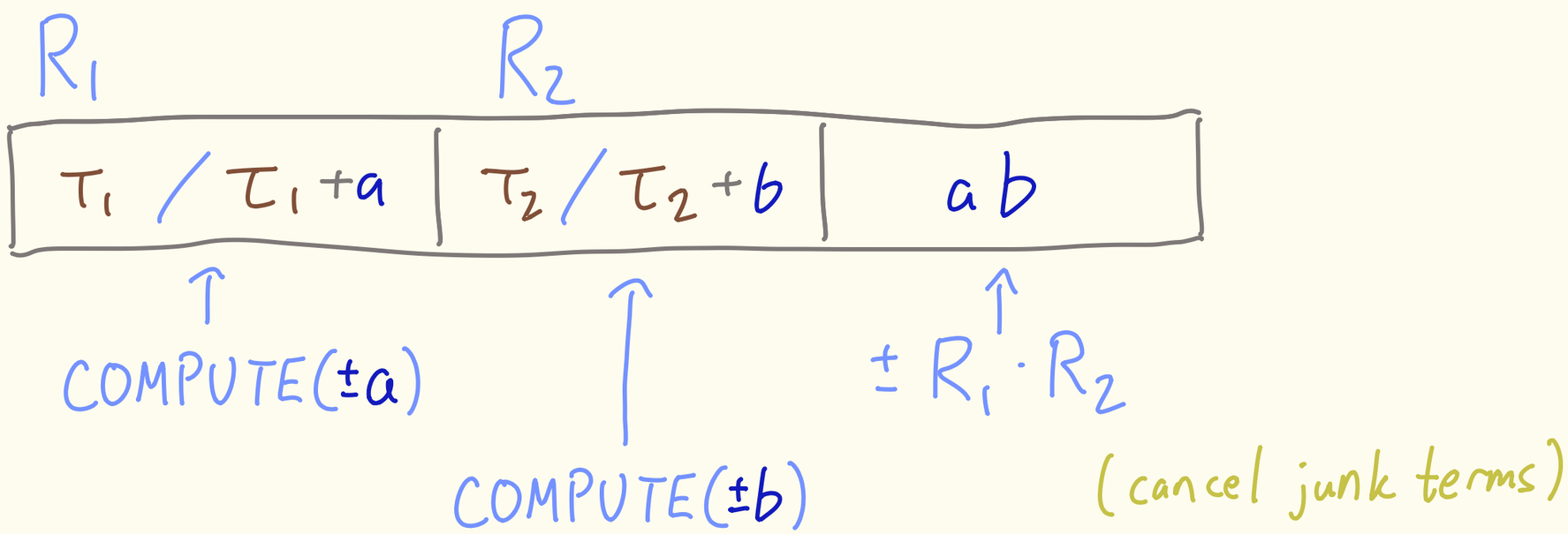


Arithmetic Magic

Example 3: Multiplication

COMPUTE(a), COMPUTE(b)

catalytic computation: $\longrightarrow$ COMPUTE($a \cdot b$)



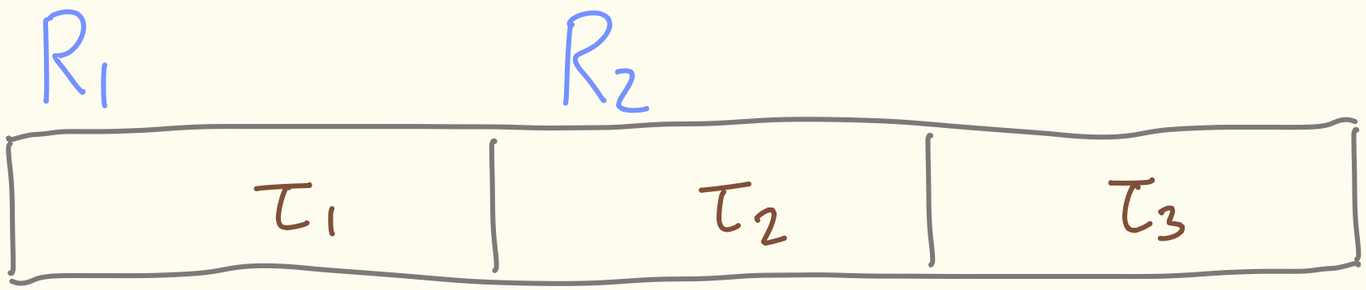
(cancel junk terms)

Arithmetic Magic

Example 3: Multiplication

COMPUTE(a), COMPUTE(b)

catalytic computation: → COMPUTE($a \cdot b$)

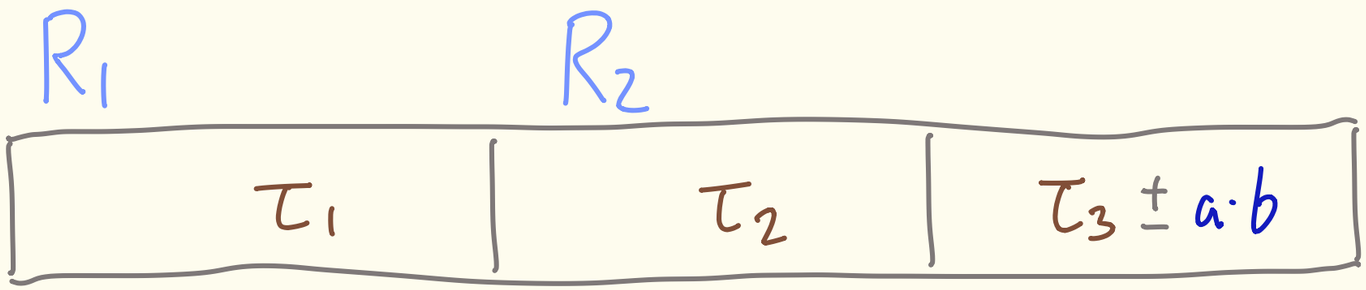


Arithmetic Magic

Example 3: Multiplication

COMPUTE(a), COMPUTE(b)

catalytic computation: $\longrightarrow$ COMPUTE($a \cdot b$)



$\longrightarrow$ COMPUTE(v): $R \pm v$

Reusing Space

$\tau_i \in \mathbb{F}$

R_1	R_2	R_3		R_s
τ_1	τ_2	τ_3	- - - -	τ_s

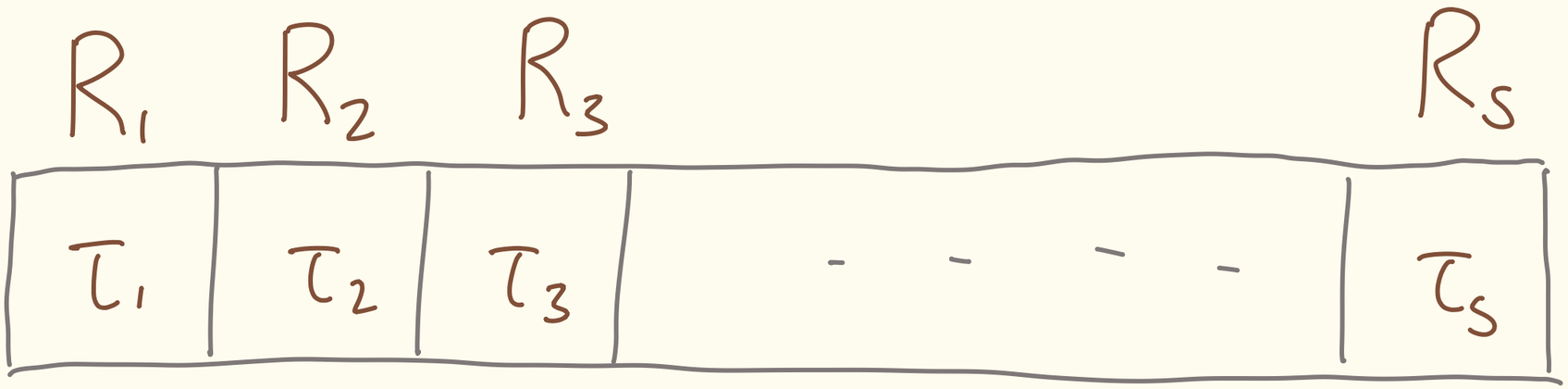
Reusing Space

R_1	R_2	R_3		R_s
τ_1	τ_2	τ_3	- - - -	τ_s

1. $R_1 \mathrel{+}= R_7 \cdot R_5 + R_{s-2}$
2. $R_5 \mathrel{-}= R_{s-2}$
3. $R_2 \mathrel{+}= \sum_{i=3}^{s-1} R_i R_{i+1}^2$ over $\mathbb{F}$

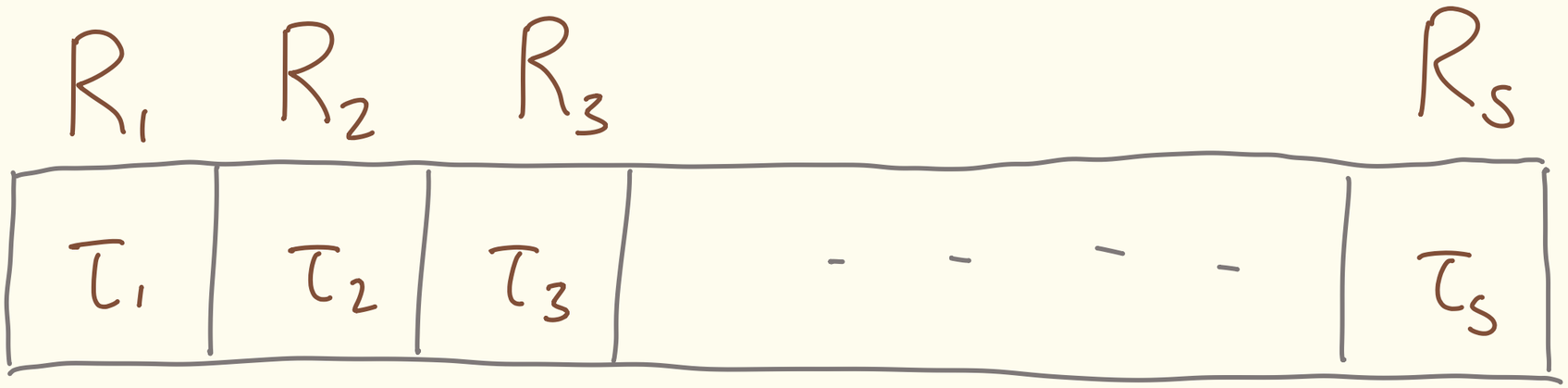
. . . . .

Reusing Space



$$P^{(\cdot 1)}_{\vec{x}} : \vec{R} \underset{(-)}{+} = \vec{x}$$

Reusing Space

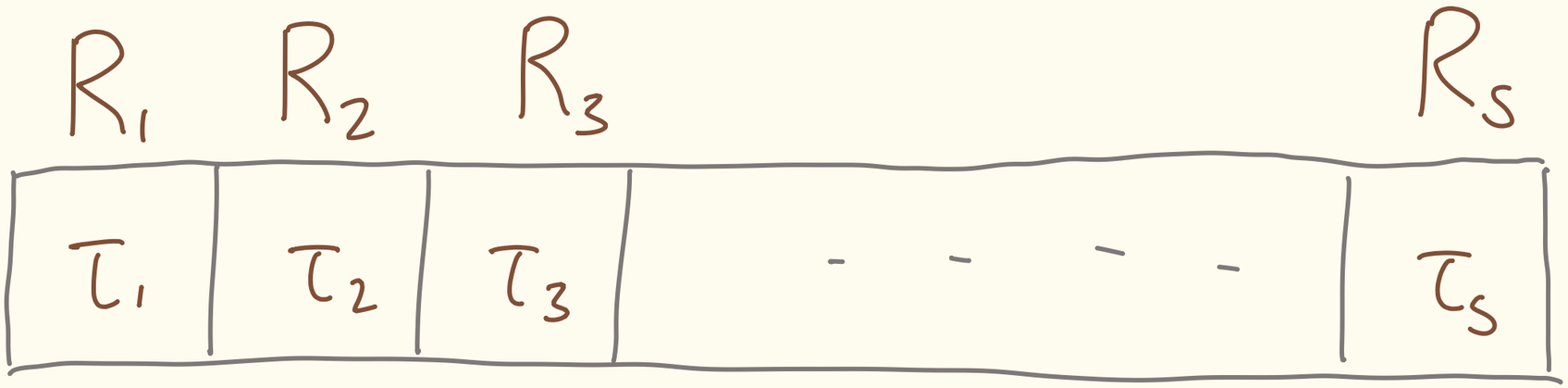


$P^{(\cdot 1)}_{\vec{x}}$: $\vec{R} \underset{(-)}{+}= \vec{x}$

$P_{q(x)}$:

$R' \mathrel{+}= q(x)$

Reusing Space



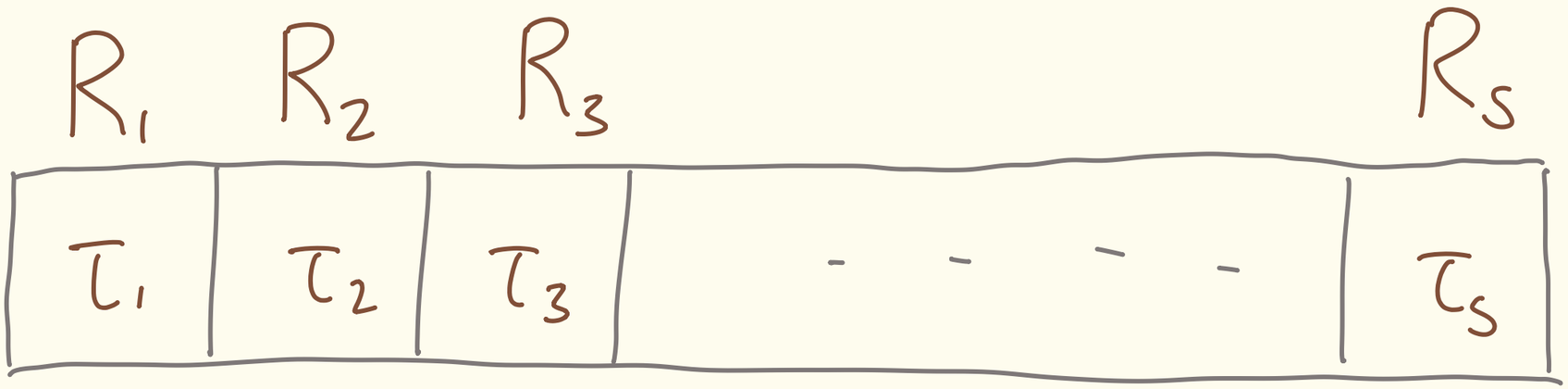
$$P_{\vec{x}}^{(\cdot 1)} : \vec{R} \underset{(-)}{+} = \vec{x}$$

$P_{q(x)}$

1. ——
2. $P_{\vec{x}}$
3. ——
4. $P_{\vec{x}}^{-1}$

. . . .

Reusing Space



$P_{\vec{x}}^{(\cdot|)} : \vec{R} \underset{(-)}{+}= \vec{x}$

efficiency:

- # registers
- # $P_{\vec{x}}$ calls

Reusing Space

[B'89]: $x \wedge y$

Reusing Space

[B'89]: $x \wedge y$

[BC'92]: $x \times y$

Reusing Space

[B'89]: $x \wedge y$

[BC'92]: $x \times y$

[BCKLS'14]: $\mathrm{MAJ}(\vec{x})$

Reusing Space

[B'89]: $x \wedge y$

[BC'92]: $x \times y$

[BCKLS'14]: $\mathrm{MAJ}(\vec{x})$

[AFMSV'25]: M^k

Reusing Space

[B'89]: $NC^1 \subseteq PBP[5]$

[BC'92]: $\#NC^1 \subseteq L$

[BCKLS'14]: $TC^1 \subseteq CL$

[AFMSV'25]: $SAC^2 \subseteq CSPACE[o(\log^2 n)]$

Reusing Space

(worse efficiency)

[CM'20, 21, 24]: <u>any</u> $f(x_1 \ldots x_n)$

Reusing Space

(worse efficiency)

[CM'20, 21, 24]: <u>any</u> $f(x_1 \ldots x_n)$

[CM'24]: $\mathrm{TEP} \subseteq \mathrm{SPACE}[\log n \cdot \log\log n]$

Reusing Space

(worse efficiency)

[CM'20, 21, 24]: <u>any</u> $f(x_1 \ldots x_n)$

[CM'24]: $\mathrm{TEP} \subseteq \mathrm{SPACE}[\log n \cdot \log\log n]$

[Wil'25]: $\mathrm{TTME}[t] \subseteq \mathrm{SPACE}[\sqrt{t \log t}]$

TAKEAWAYS

TAKEAWAYS

- power of arithmetic: reversibility + cancellation

TAKEAWAYS

- power of arithmetic: reversibility + cancellation
  - seems to bypass information complexity

TAKEAWAYS

- power of arithmetic: reversibility + cancellation
  - seems to bypass information complexity
  - also useful in e.g. non-monotone computation

TAKEAWAYS

- power of arithmetic: reversibility + cancellation
  - seems to bypass information complexity
  - also useful in e.g. non-monotone computation
- completely ignores content of catalytic tape

TAKEAWAYS

- power of arithmetic: reversibility + cancellation
  - seems to bypass information complexity
  - also useful in e.g. non-monotone computation
- completely ignores content of catalytic tape
  - always runs in (same) polynomial time

TAKEAWAYS

- power of arithmetic: reversibility + cancellation
  - seems to bypass information complexity
  - also useful in e.g. non-monotone computation
- completely ignores content of catalytic tape
  - always runs in (same) polynomial time
  - works in e.g. quantum setting

Open Problems

Open Problems

1. better programs for powering

Open Problems

1. better programs for powering

2. more functions converted into arithmetic form (ex. GCD)

Open Problems

1. better programs for powering

2. more functions converted into arithmetic form (ex. GCD)

3. structured catalysts (via CoR) as register program preprocessing

That's all Folks!