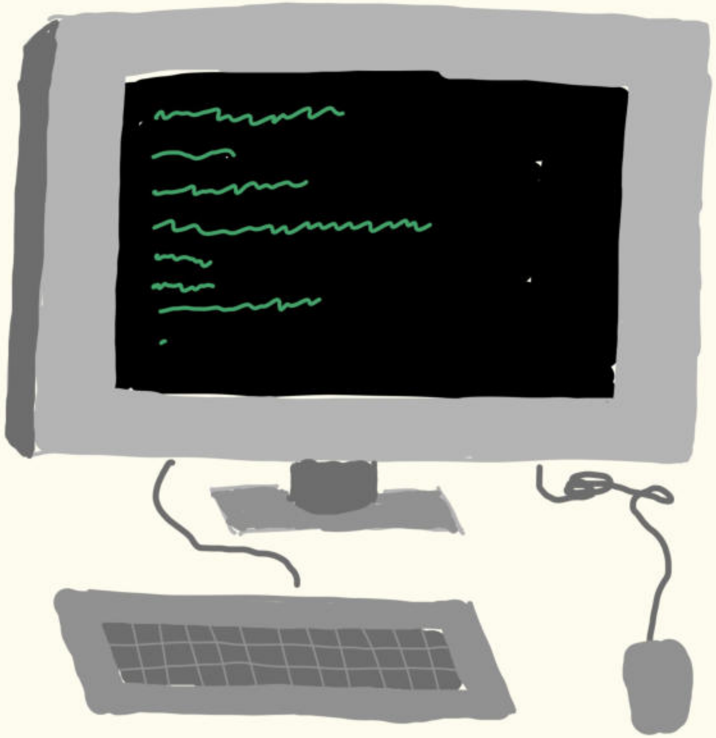
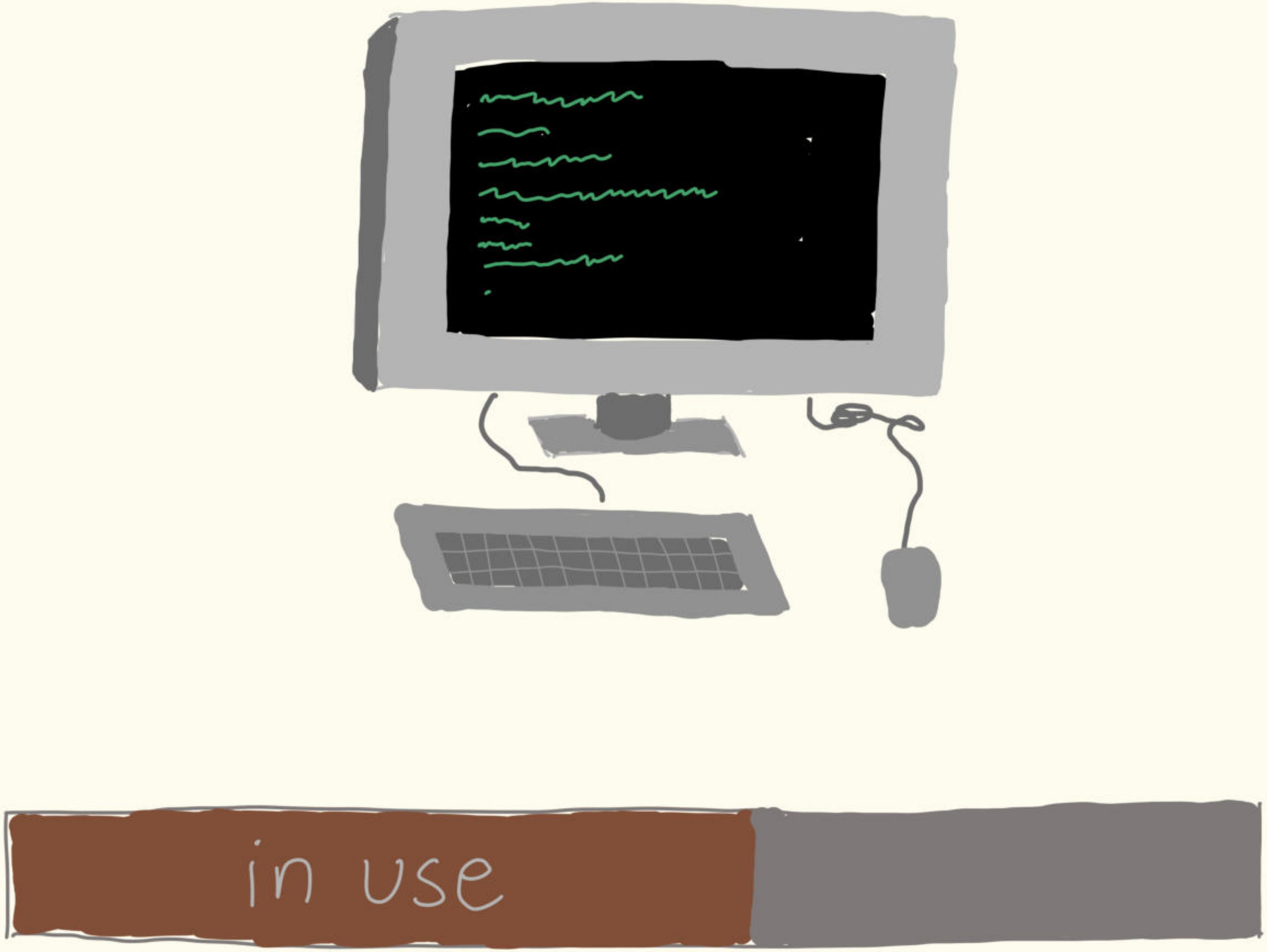
How to Reuse Space

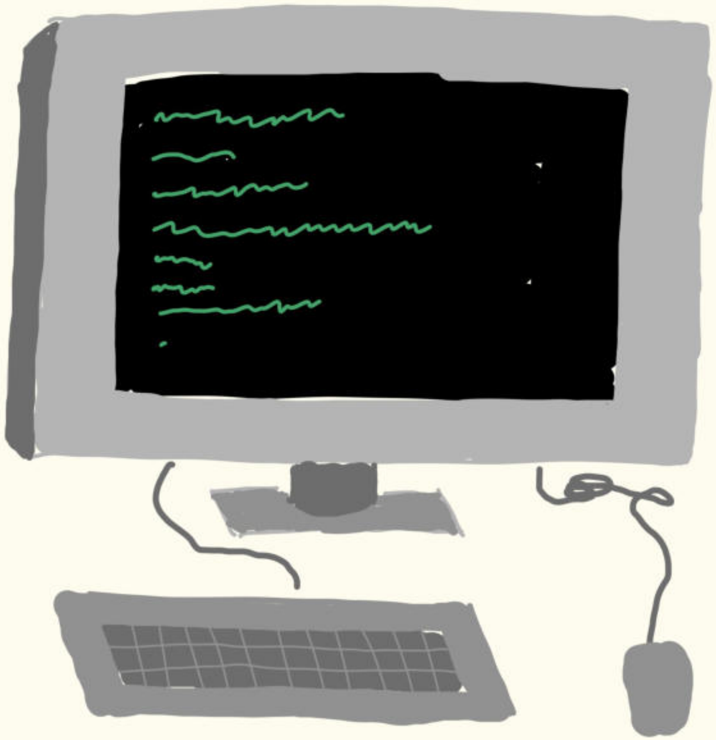
Ian Mertz
Charles University


Ian Mertz
Tech support
Fields Institute




in use

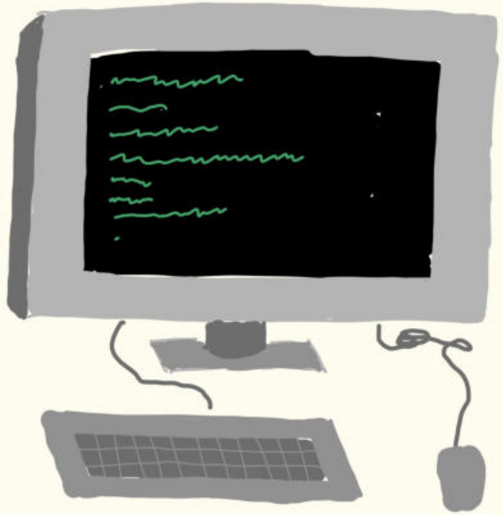

HEY! GET BACK TO WORK!
COMPILING!
OH. CARRY ON.




in use

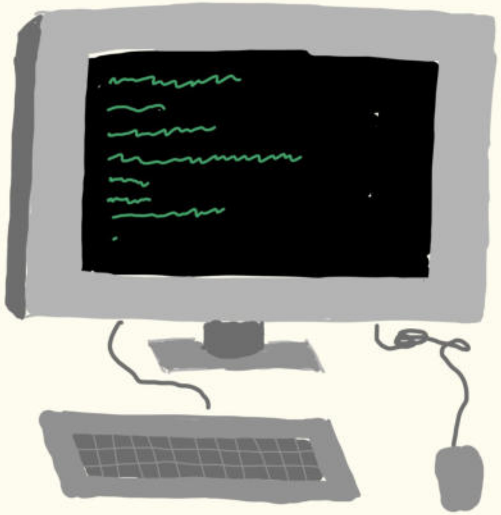
the next day...

Options:



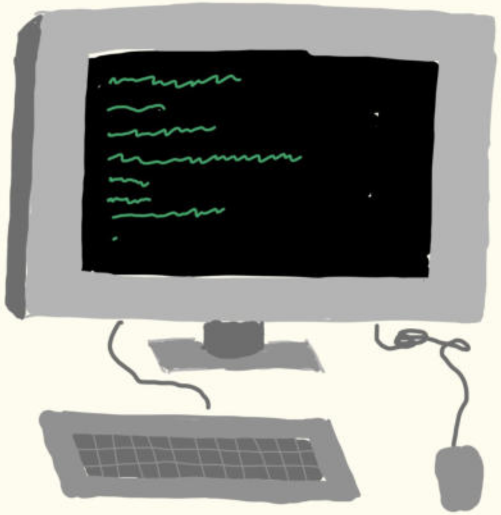
Options:

1. Delete



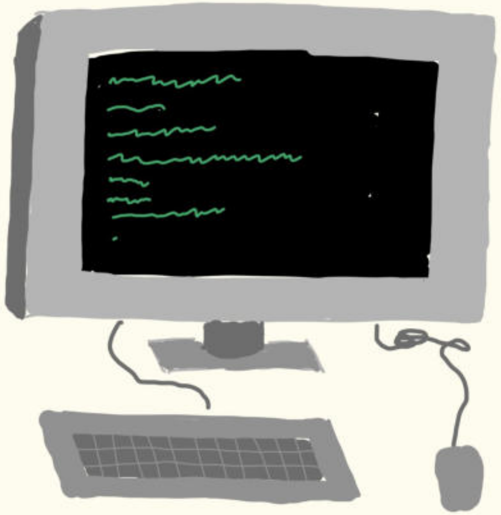
OPTIONS:

1. ~~DELETE~~



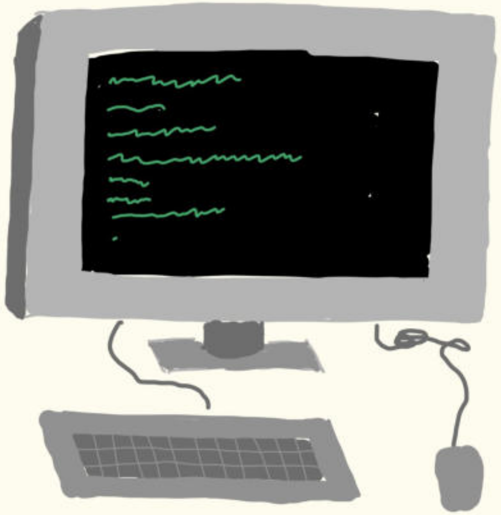
Options:

1. ~~Delete~~
2. Compress



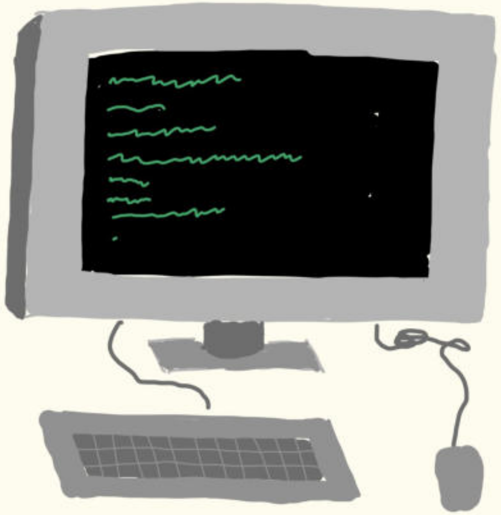
OPTIONS:

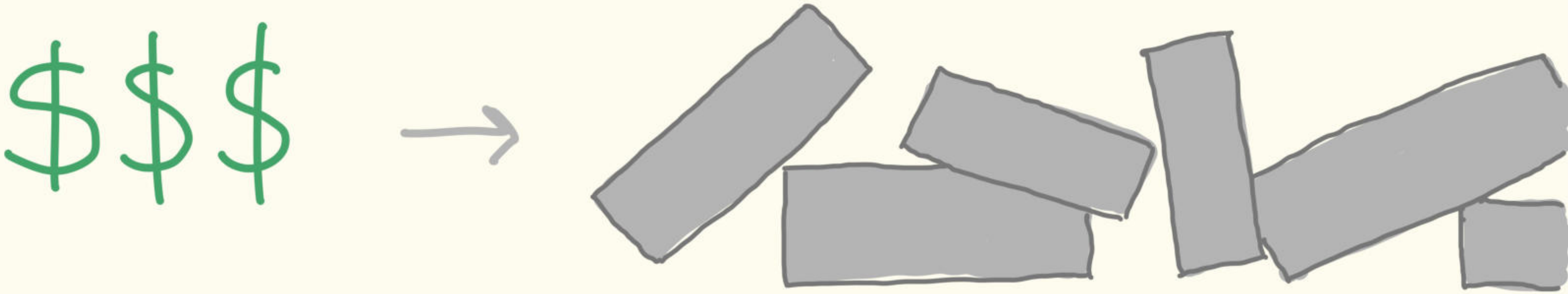
1. ~~DELETE~~
2. ~~COMPRESS~~



Options:

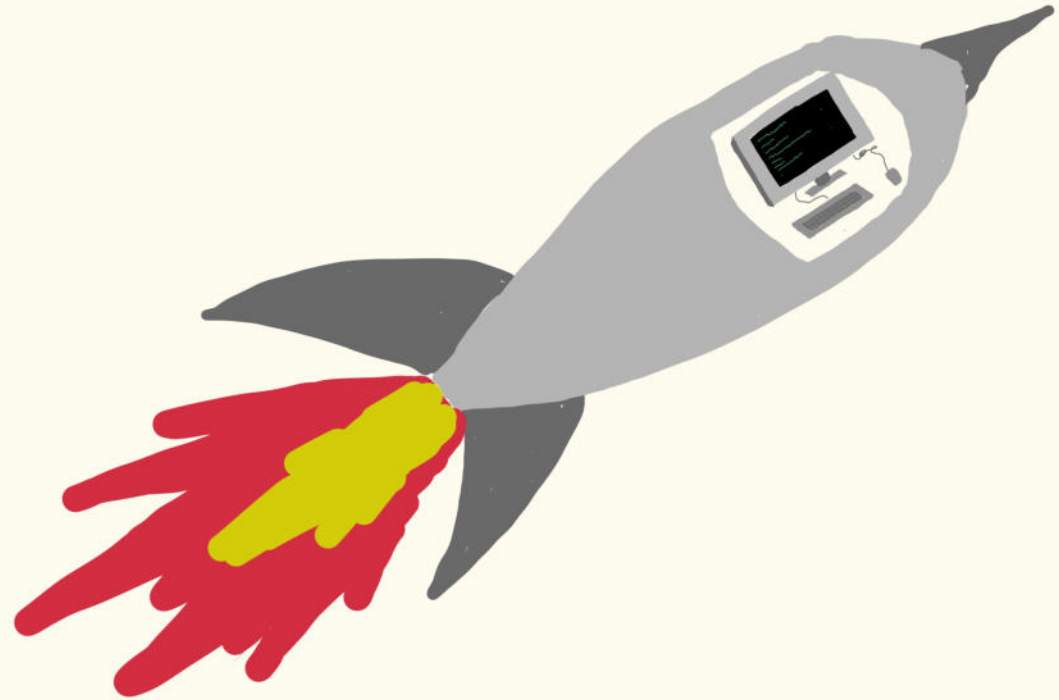
1. ~~Delete~~
2. ~~Compress~~
3. Capitalism

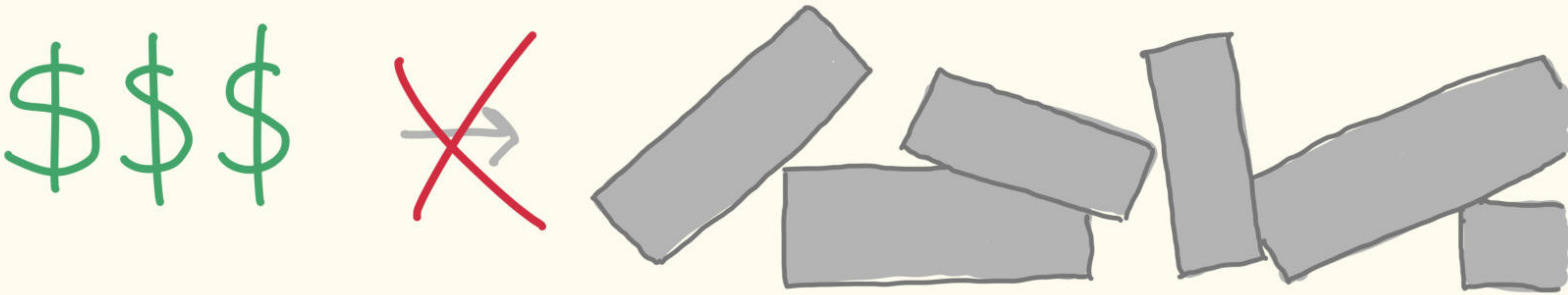




OPTIONS:

1. ~~DELETE~~
2. ~~COMPRESS~~
3. ~~CAPITALISM~~

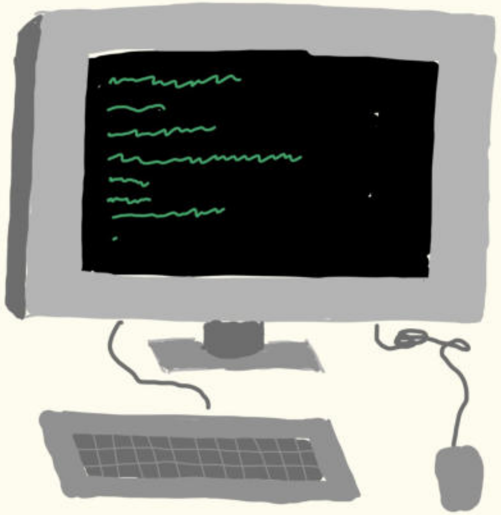




OPTIONS:

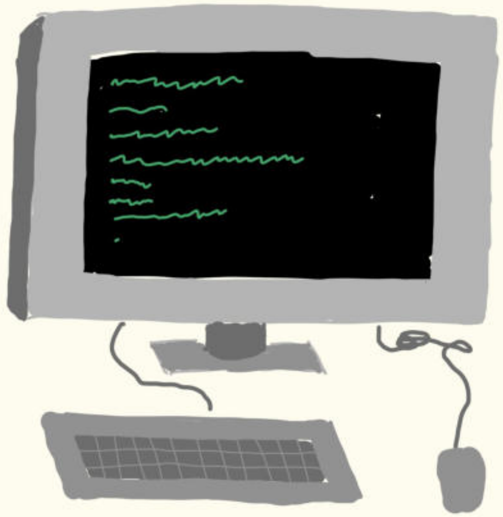
1. ~~DELETE~~
2. ~~COMPRESS~~
3. ~~CAPITALISM~~

4. ???



OPTIONS:

1. ~~DELETE~~
2. ~~COMPRESS~~
3. ~~CAPITALISM~~



4. REUSE

storage + computation

Today

Part I: Results

Prologue: Space & Time

Catalytic Computation

What We Know

Part II: Techniques

Configuration Graphs

Compress-Or-Random

Register Programs

Next Steps

Today

Part I: Results

Prologue: Space & Time

Catalytic Computation

What We Know

Part II: Techniques

Configuration Graphs

Compress-Or-Random

Register Programs

Next Steps

How to Show $L \neq P$

[..., KRW'95, CMWBS'12]:

Composition

How to Show $L \neq P$

[..., KRW'95, CMWBS'12]:

Composition

Idea: solving <u>many</u> functions is hard

How to Show $L \neq P$

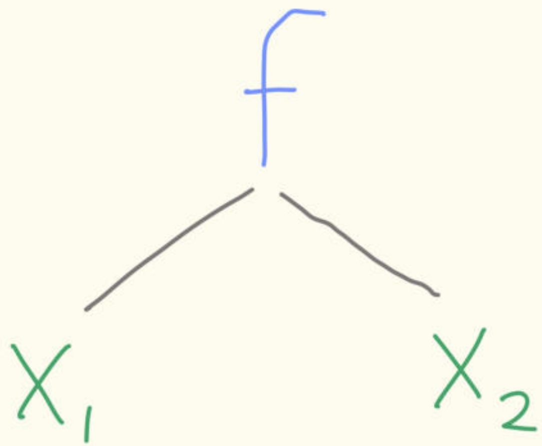
[..., KRW'95, CMWBS'12]:

Composition

Idea: solving <u>many</u> functions is hard even if each <u>individual</u> function is easy

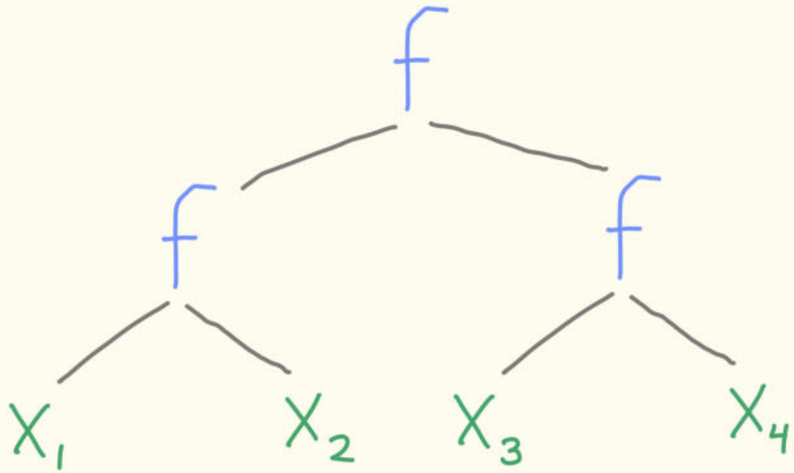
How to Show $L \neq P$

Composition



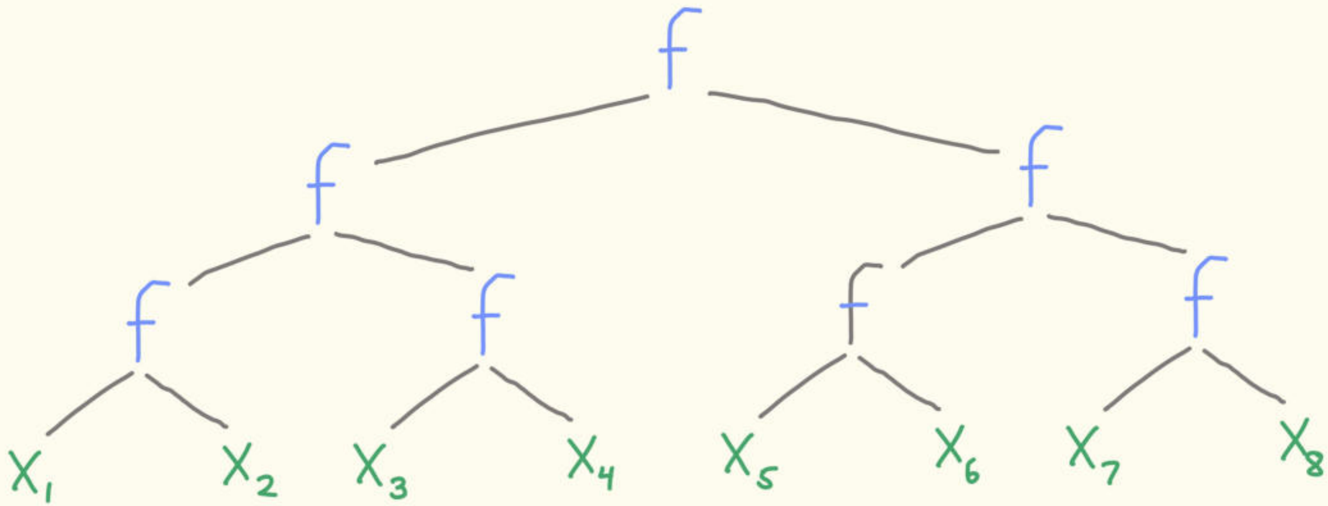
How to Show $L \neq P$

Composition



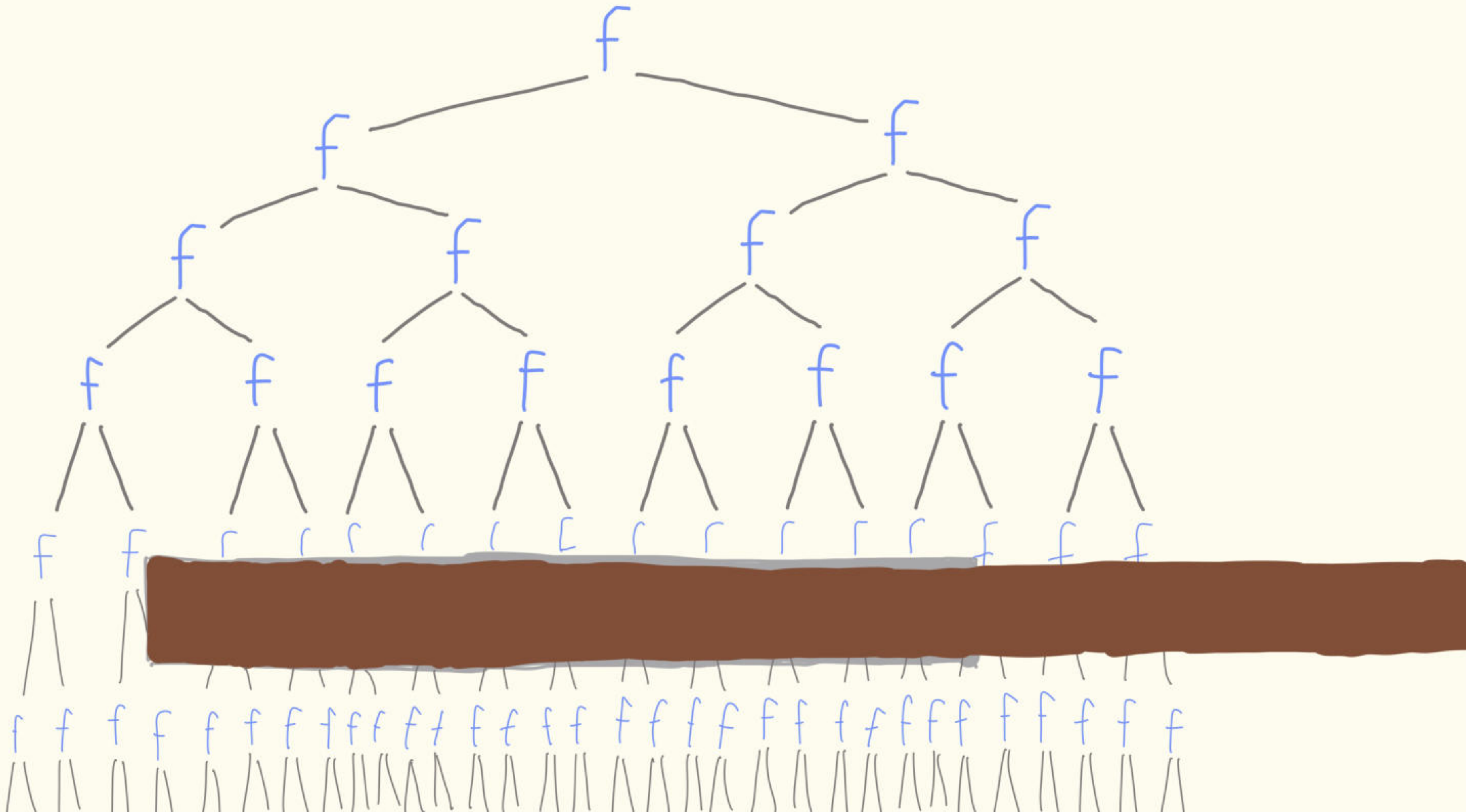
How to Show $L \neq P$

Composition



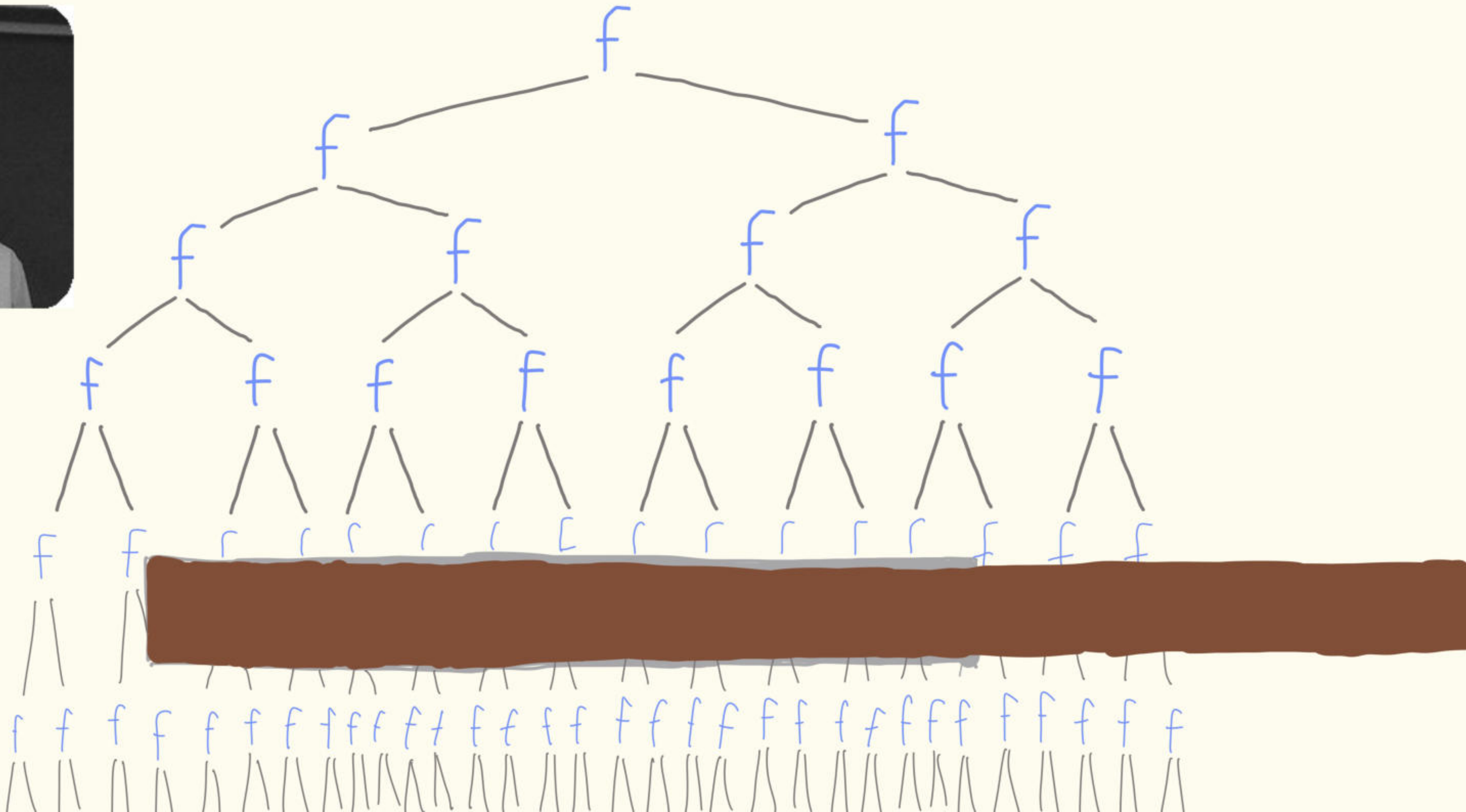
How to Show $L \neq P$

Composition



How to Show $L \neq P$

Tree Evaluation Problem



How to Show $L \neq P$

Tree Evaluation Problem

Conjecture [CMWBS'12]:

$$\mathrm{SPACE}(\text{Tree Evaluation}) = \Omega(\log^2 n)$$

How to Show $L \neq P$

Tree Evaluation Problem

f

Conjecture [CMWBS'12]:

$$\text{SPACE}(\text{Tree Evaluation}) = \Omega(\log^2 n)$$

[CMWBS'12, Liu'13, EMP'18, IN'19]:

full lower bound for restricted settings

How to Show $L \neq P$

Tree Evaluation Problem

f

Conjecture [CMWBS'12]:

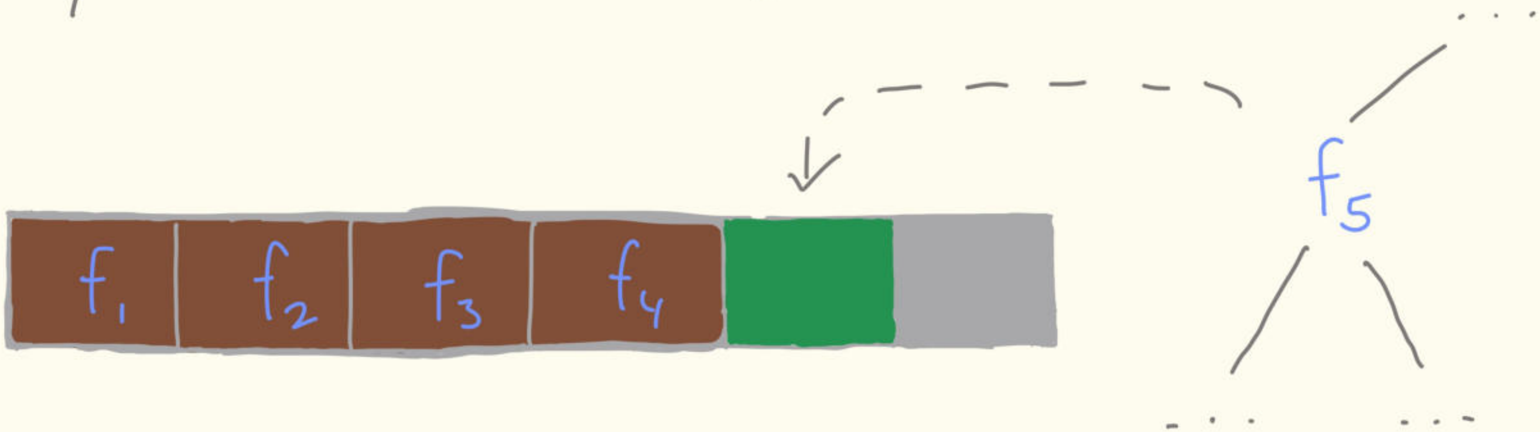
$$\text{SPACE}(\text{Tree Evaluation}) = \Omega(\log^2 n)$$

Cook: \$100 (canadian) for <u>any</u> improvement

How to Show $L \neq P$

Tree Evaluation Problem

Crucial Assumption: each part of the tree uses different memory, and cannot make any use of other (full) sections



How to Show $L \neq P$

Tree Evaluation Problem

Crucial Assumption: each part of the tree uses different memory, and cannot make any use of other (full) sections

How to Show $L \neq P$

Tree Evaluation Problem

Crucial Assumption: ~~each part of the tree uses different memory, and cannot make any use of other (full) sections~~

FALSE!

How to Show $L \neq P$

Tree Evaluation $\in$ SPACE

How to Show $L \neq P$

Tree Evaluation $\in$ SPACE

[CM'20]: $O(\log^2 n / \log\log n)$ sometimes

How to Show $L \neq P$

Tree Evaluation $\in$ SPACE

[CM'20]: $O(\log^2 n / \log\log n)$ sometimes

[CM'21]: $O(\log^2 n / \log\log n)$

↳ \$100

How to Show $L \neq P$

Tree Evaluation $\in$ SPACE

[CM'20]: $O(\log^2 n / \log\log n)$ sometimes

[CM'21]: $O(\log^2 n / \log\log n)$

↳ \$100

[CM'24]: $O(\log n \cdot \log\log n)$ and $O(\log n)$ sometimes

How to Show $L = P$?

[Wil'25]:

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

How to Show $L = P$?

[Wil'25]:

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

[HPV'77]:

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

How to Show $L = P$?

[Wil'25]:

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

[HPV'77]:

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

$$+$$

$$\mathrm{TEP} \subseteq \mathrm{SPACE}[\log n \cdot \log\log n]$$

Today

Part I: Results

Prologue: Space & Time

Catalytic Computation

What We Know

Part II: Techniques

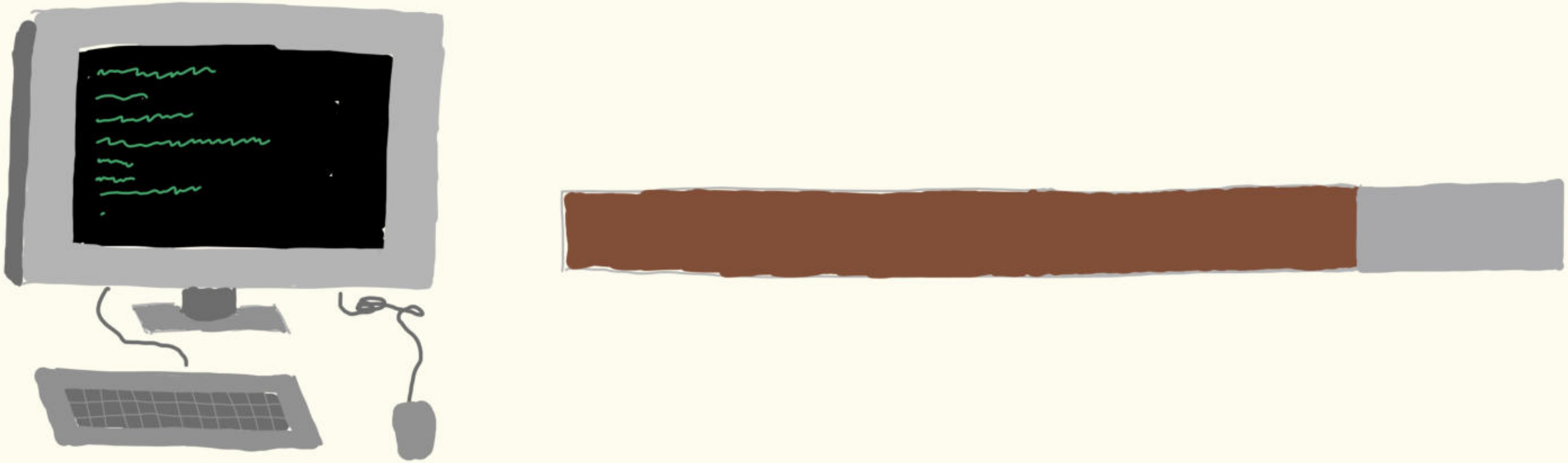
Configuration Graphs

Register Programs

Compress-Or-Random

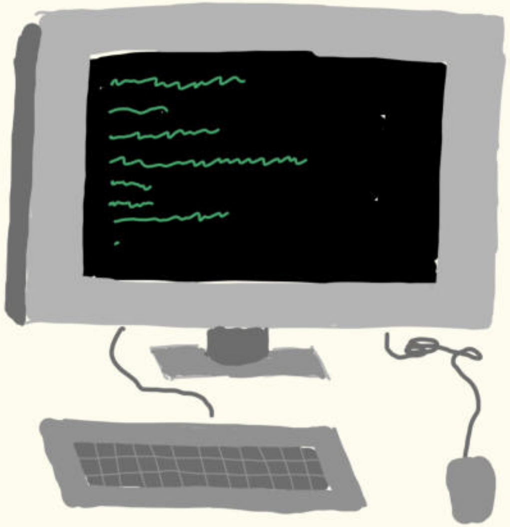
Next Steps

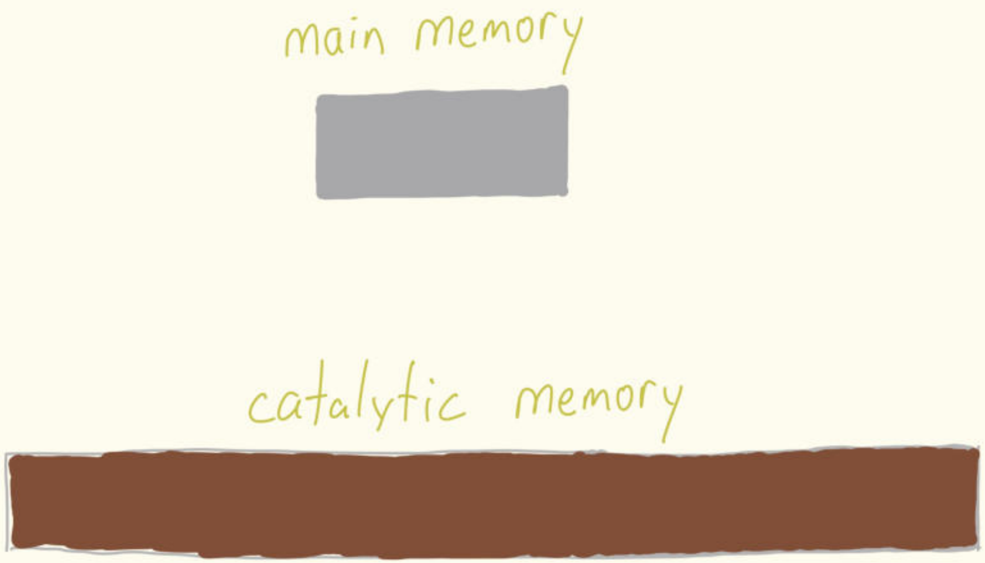
The Study of Reuse



The Study of Reuse

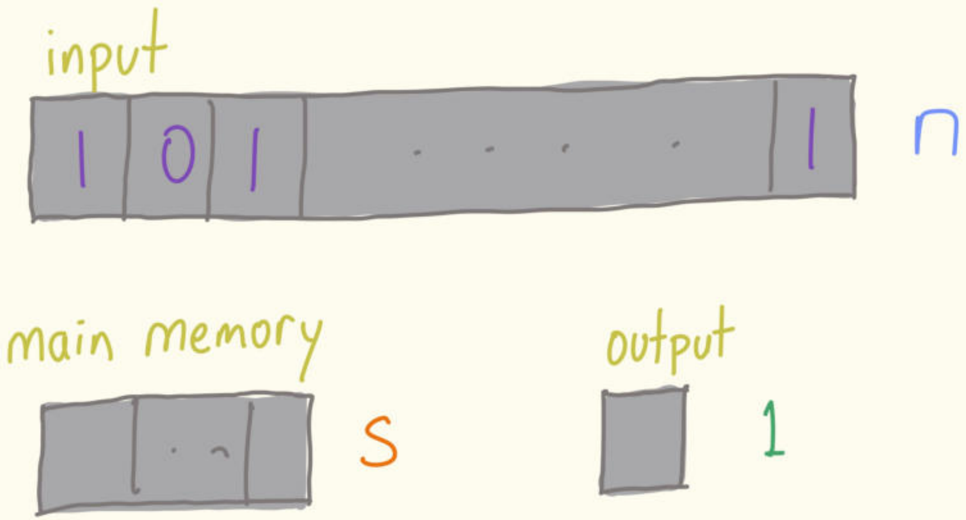
catalytic computing [BCKLS'14]





The Study of Reuse

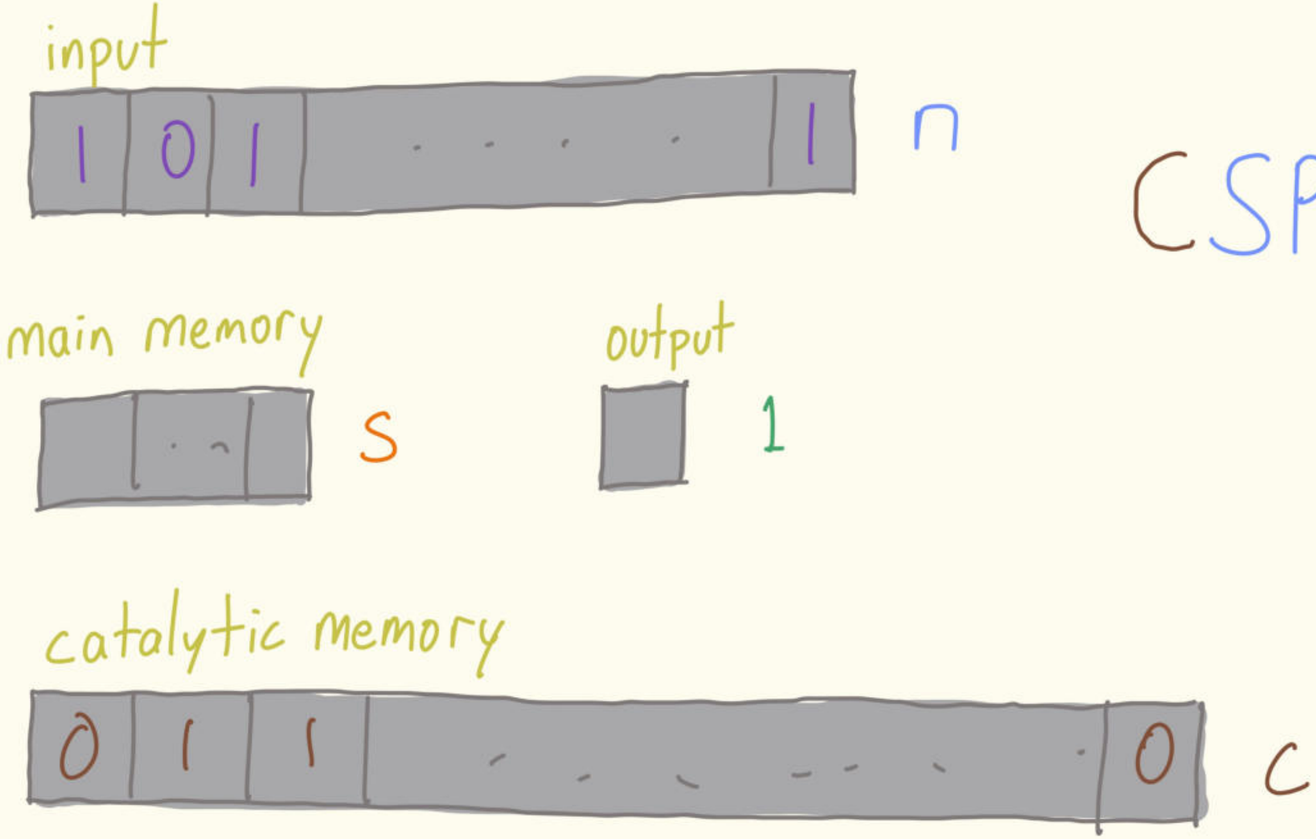
catalytic computing [BCKLS'14]



SPACE[s]

The Study of Reuse

catalytic computing [BCKLS'14]

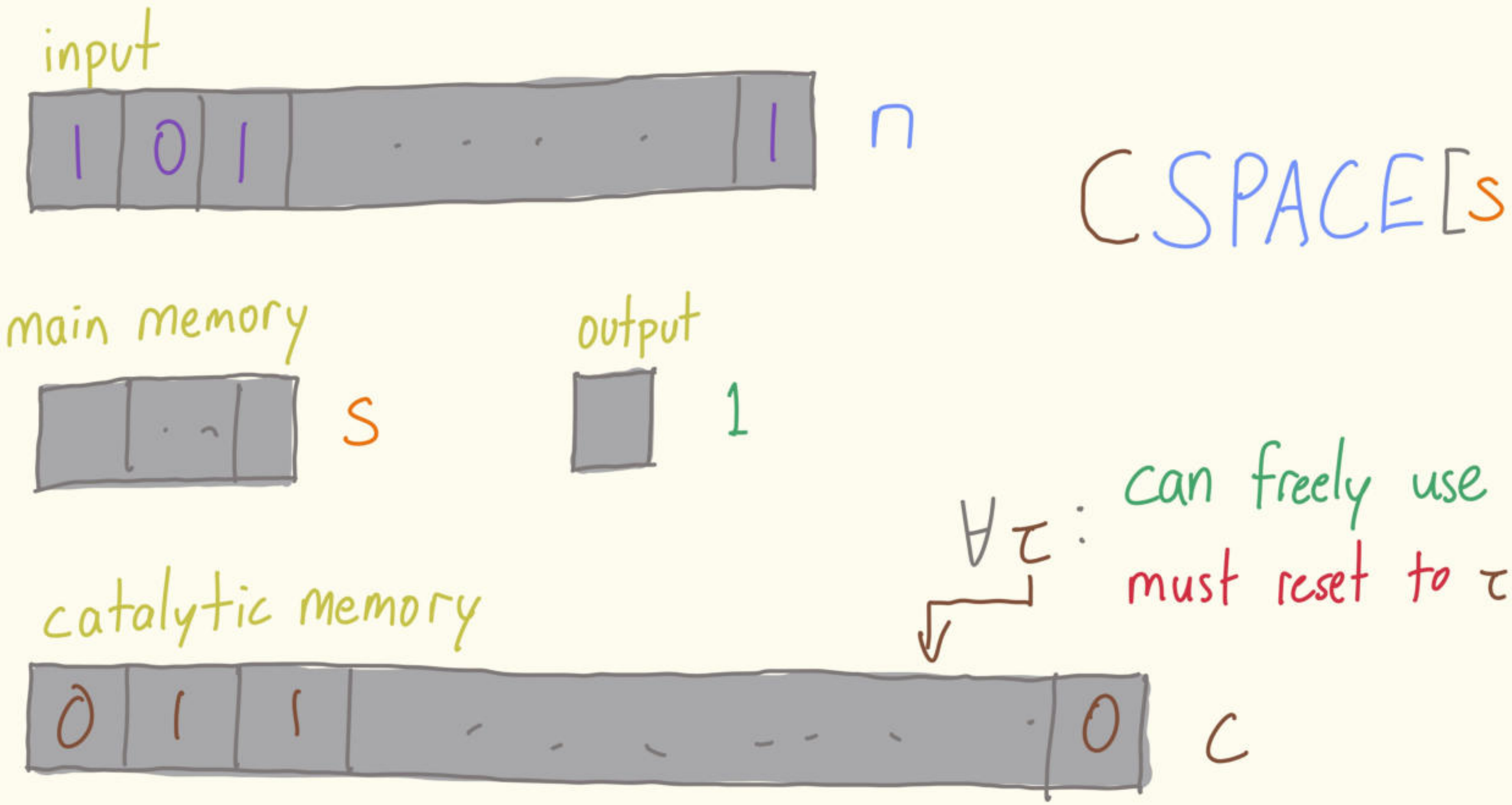


CSPACE[s,c]

The Study of Reuse

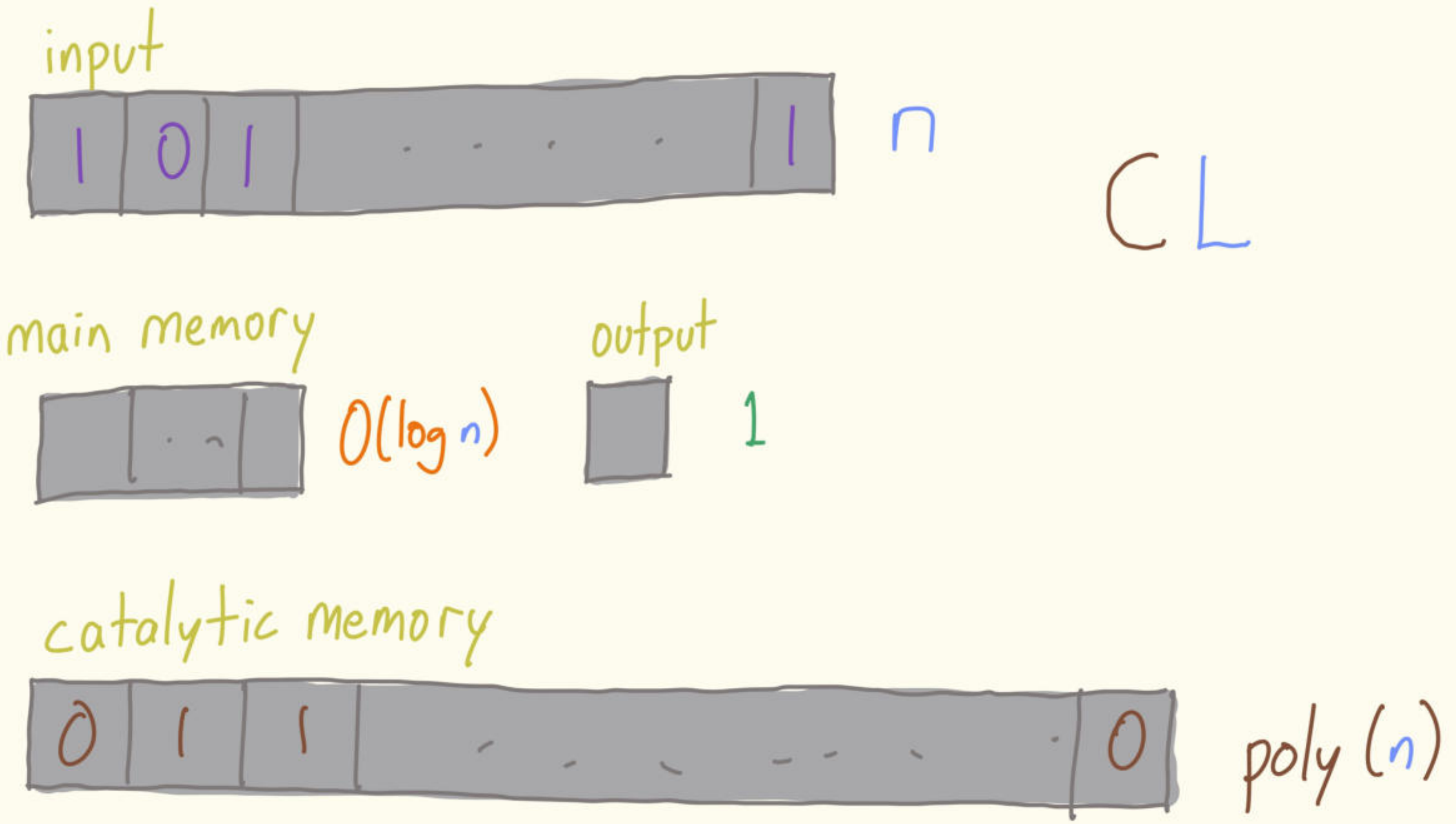
catalytic computing [BCKLS'14]

CSPACE[s, c]



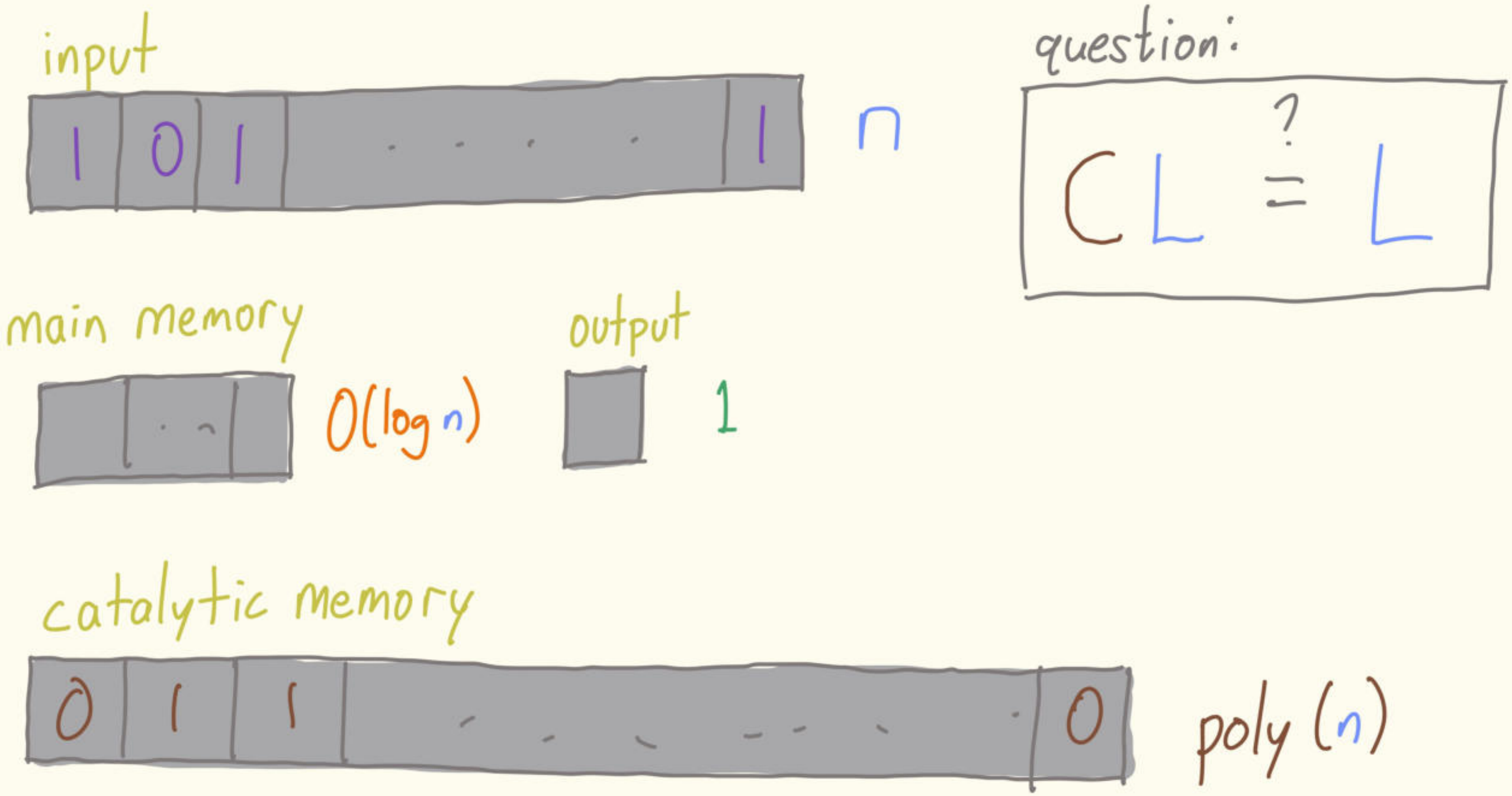
The Study of Reuse

catalytic computing [BCKLS'14]



The Study of Reuse

catalytic computing [BCKLS'14]



Today

Part I: Results

Prologue: Space & Time

Catalytic Computation

What We Know

Part II: Techniques

Configuration Graphs

Register Programs

Compress-Or-Random

Next Steps

The Power of Reuse

trivial:

$$\mathsf{L} \subseteq \mathsf{CL} \subseteq \mathsf{PSPACE}$$

The Power of Reuse

trivial:

$$L \subseteq CL \subseteq PSPACE$$

[BCKLS'14]:

$$CL$$

The Power of Reuse

trivial:

$$L \subseteq CL \subseteq PSPACE$$

[BCKLS'14]:

$$CL \subseteq ZPP$$

probably $\subseteq P$

The Power of Reuse

trivial:

$$L \subseteq CL \subseteq PSPACE$$

[BCKLS'14]:

$$TC^1 \subseteq CL \subseteq ZPP$$

contains NL, BPL, DET, etc.

probably $\subseteq P$

The Power of Reuse

most likely:

$$L \subsetneq CL \subsetneq PSPACE$$

[BCKLS'14]:

$$TC^1 \subseteq CL \subseteq ZPP$$

contains NL, BPL, DET, etc.

probably $\subseteq P$

The Power of Reuse

most likely:

$$L \subsetneq CL \subsetneq PSPACE$$

[BCKLS'14]:

$$TC^1 \subseteq CL \subseteq ZPP$$

$$\text{Match} \underset{\text{[AM'25]}}{\in} \qquad \underset{\text{[CLMP'25]}}{\leq} \text{Lossy-Code}$$

The Power of Reuse

* S := SPACE
CS := CSPACE

lower / upper	Classical	Catalytic
Classical		
Catalytic		

The Power of Reuse

* S := SPACE
CS := CSPACE

	Classical	Catalytic
Classical	$\mathrm{TEP} \in S[\log n \cdot \log\log n]$ $\mathrm{TIME}[t] \subseteq S[\sqrt{t \log t}]$	
Catalytic		

The Power of Reuse

* S := SPACE
CS := CSPACE

	Classical	Catalytic
Classical	$\text{TEP} \in S[\log n \cdot \log\log n]$ $\text{TIME}[t] \subseteq S[\sqrt{t \log t}]$	$\text{CL} \subseteq \text{ZPP}$ $\text{CL} \leq \text{Lossy-Code}$
Catalytic	$\text{TC}^1 \subseteq \text{CL}$ $\text{Match} \in \text{CL}$	

The Power of Reuse

randomness

$$\mathsf{BPL} \subseteq \mathsf{CL}$$

The Power of Reuse

randomness

$$\mathsf{BPL} \subseteq \mathsf{CL}$$

[Pyn'24]:

$$\mathsf{BPL} \subseteq \mathsf{CSPACE}[\log n, \log^2 n]$$

The Power of Reuse

randomness

$$\mathsf{BPL} \subseteq \mathsf{CL}$$

[Pyn'24]:

$$\mathsf{BPL} \subseteq \mathsf{CSPACE}[\log n, \log^2 n]$$

[CLMP'25]:

$$\mathsf{CBPL} \subseteq \mathsf{CL}$$

The Power of Reuse

randomness

$$\mathsf{BPL} \subseteq \mathsf{CL}$$

[Pyn'24]:

$$\mathsf{BPL} \subseteq \mathsf{CSPACE}[\log n, \log^2 n]$$

[CLMP'25]:

$$\mathsf{CBPL} \subseteq \mathsf{CL}$$

[KMPS'25]: very tight params!

The Power of Reuse

* S := SPACE
CS := CSPACE

	Classical	Catalytic
Classical	$\mathsf{TEP} \in \mathsf{S}[\log n \cdot \log\log n]$ $\mathsf{TIME}[t] \subseteq \mathsf{S}[\sqrt{t \log t}]$	$\mathsf{CL} \subseteq \mathsf{ZPP}$ $\mathsf{CL} \leq \textsc{Lossy-Code}$
Catalytic	$\mathsf{TC}^1 \subseteq \mathsf{CL}$ $\textsc{Match} \in \mathsf{CL}$	

The Power of Reuse

* S := SPACE
CS := CSPACE

	Classical	Catalytic
Classical	$\mathsf{TEP} \in \mathsf{S}[\log n \cdot \log\log n]$ $\mathsf{TIME}[t] \subseteq \mathsf{S}[\sqrt{t \log t}]$	$\mathsf{CL} \subseteq \mathsf{ZPP}$ $\mathsf{CL} \leq \text{Lossy-Code}$
Catalytic	$\mathsf{TC}^1 \subseteq \mathsf{CL}$ $\text{Match} \in \mathsf{CL}$ $\mathsf{BPL} \subseteq \mathsf{CS}[\log n, \log^2 n]$	$\mathsf{CBPL} \subseteq \mathsf{CL}$

The Power of Reuse

randomness + non-determinism

The Power of Reuse

randomness + non-determinism

[DPTW'25]:

"$\mathrm{BPL} \subseteq \mathrm{NL}$ or $\mathrm{NL} \subseteq \mathrm{SC}^2$"

THE POWER OF REUSE

randomness + non-determinism

[DPTW'25]:

"$BPL \subseteq NL$ or $NL \subseteq SC^2$"

[KMPS'25]:

$CNL \subseteq CL$ (+ AMCL hierarchy)

The Power of Reuse

randomness + non-determinism

[DPTW'25]:

"$\mathsf{BPL} \subseteq \mathsf{NL}$ or $\mathsf{NL} \subseteq \mathsf{SC}^2$"

[KMPS'25]:

$\mathsf{CNL} \subseteq \mathsf{CL}$ (+ AMCL hierarchy)

$\mathsf{CPrL} \subseteq \mathsf{CL}$

The Power of Reuse

* S := SPACE
CS := CSPACE

	Classical	Catalytic
Classical	$\mathsf{TEP} \in S[\log n \cdot \log\log n]$ $\mathsf{TIME}[t] \subseteq S[\sqrt{t \log t}]$	$\mathsf{CL} \subseteq \mathsf{ZPP}$ $\mathsf{CL} \leq \text{Lossy-Code}$
Catalytic	$\mathsf{TC}^1 \subseteq \mathsf{CL}$ $\text{Match} \in \mathsf{CL}$ $\mathsf{BPL} \subseteq CS[\log n, \log^2 n]$	$\mathsf{CBPL} \subseteq \mathsf{CL}$

The Power of Reuse

* S := SPACE
CS := CSPACE

	Classical	Catalytic
Classical	$TEP \in S[\log n \cdot \log\log n]$ $TIME[t] \subseteq S[\sqrt{t \log t}]$ "$BPL \subseteq NL$ or $NL \subseteq SC^2$"	$CL \subseteq ZPP$ $CL \leq$ Lossy-Code
Catalytic	$TC^1 \subseteq CL$ Match $\in CL$ $BPL \subseteq CS[\log n, \log^2 n]$	$CBPL \subseteq CL$ $CNL \subseteq CL$

The Power of Reuse

robustness

The Power of Reuse

robustness

$LCS[s,c,e]$: $CS[s,c]$ w/e errors

The Power of Reuse

robustness

$LCS[s,c,e]$: $CS[s,c]$ w/ e errors

[GJST'24]:

$$LCS[s,c,O(1)] = CS[s,c]$$

The Power of Reuse

robustness

$LCS[s,c,e]$: $CS[s,c]$ w/ e errors

[GJST'24]:

$$LCS[s,c,O(1)] = CS[s,c]$$

[FMST'25]:

$$LCS[s,c,e] = CS[s + e\log c, c]$$

The Power of Reuse

* $S := \mathsf{SPACE}$
$CS := \mathsf{CSPACE}$

	Classical	Catalytic
Classical	$\mathsf{TEP} \in S[\log n \cdot \log\log n]$ $\mathsf{TIME}[t] \subseteq S[\sqrt{t \log t}]$ "$\mathsf{BPL} \subseteq \mathsf{NL}$ or $\mathsf{NL} \subseteq \mathsf{SC}^2$"	$\mathsf{CL} \subseteq \mathsf{ZPP}$ $\mathsf{CL} \leq$ Lossy-Code
Catalytic	$\mathsf{TC}^1 \subseteq \mathsf{CL}$ Match $\in \mathsf{CL}$ $\mathsf{BPL} \subseteq CS[\log n, \log^2 n]$	$\mathsf{CBPL} \subseteq \mathsf{CL}$ $\mathsf{CNL} \subseteq \mathsf{CL}$ $\mathsf{LCS}[s,c,e] = CS[s + e \log c, c]$

The Power of Reuse

* S := SPACE
CS := CSPACE

	Classical	Catalytic
Classical	$\mathsf{TEP} \in S[\log n \cdot \log\log n]$ $\mathsf{TIME}[t] \subseteq S[\sqrt{t \log t}]$ "$\mathsf{BPL} \subseteq \mathsf{NL}$ or $\mathsf{NL} \subseteq \mathsf{SC}^2$" AND MORE!	$\mathsf{CL} \subseteq \mathsf{ZPP}$ $\mathsf{CL} \leq$ Lossy-Code ???
Catalytic	$\mathsf{TC}^1 \subseteq \mathsf{CL}$ Match $\in \mathsf{CL}$ $\mathsf{BPL} \subseteq CS[\log n, \log^2 n]$ AND MORE!	$\mathsf{CBPL} \subseteq \mathsf{CL}$ $\mathsf{CNL} \subseteq \mathsf{CL}$ $LCS[s, c, e] = CS[s + e \log c, c]$ AND MORE!

Today

Part I: Results

Prologue: Space & Time

Catalytic Computation

What We Know

Part II: Techniques

Configuration Graphs

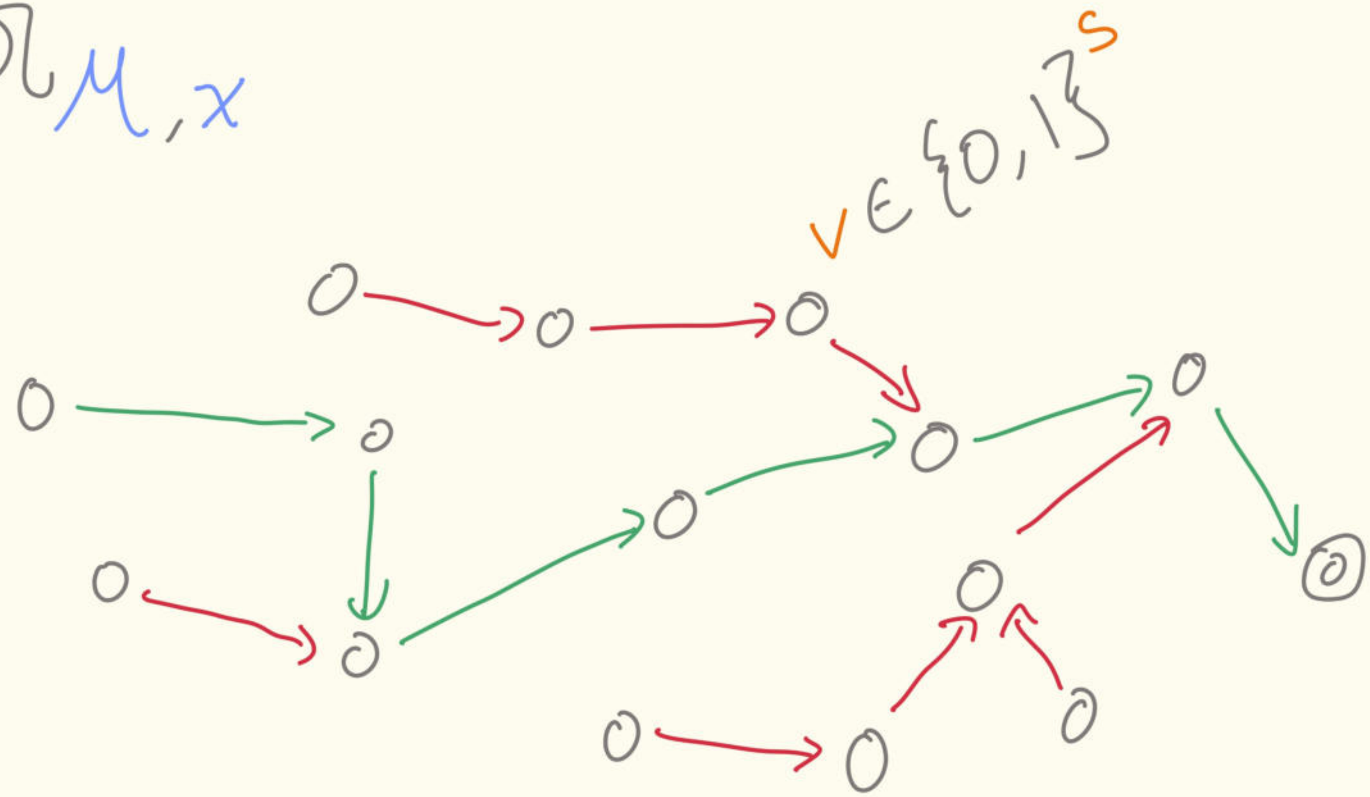
Compress-Or-Random

Register Programs

Next Steps

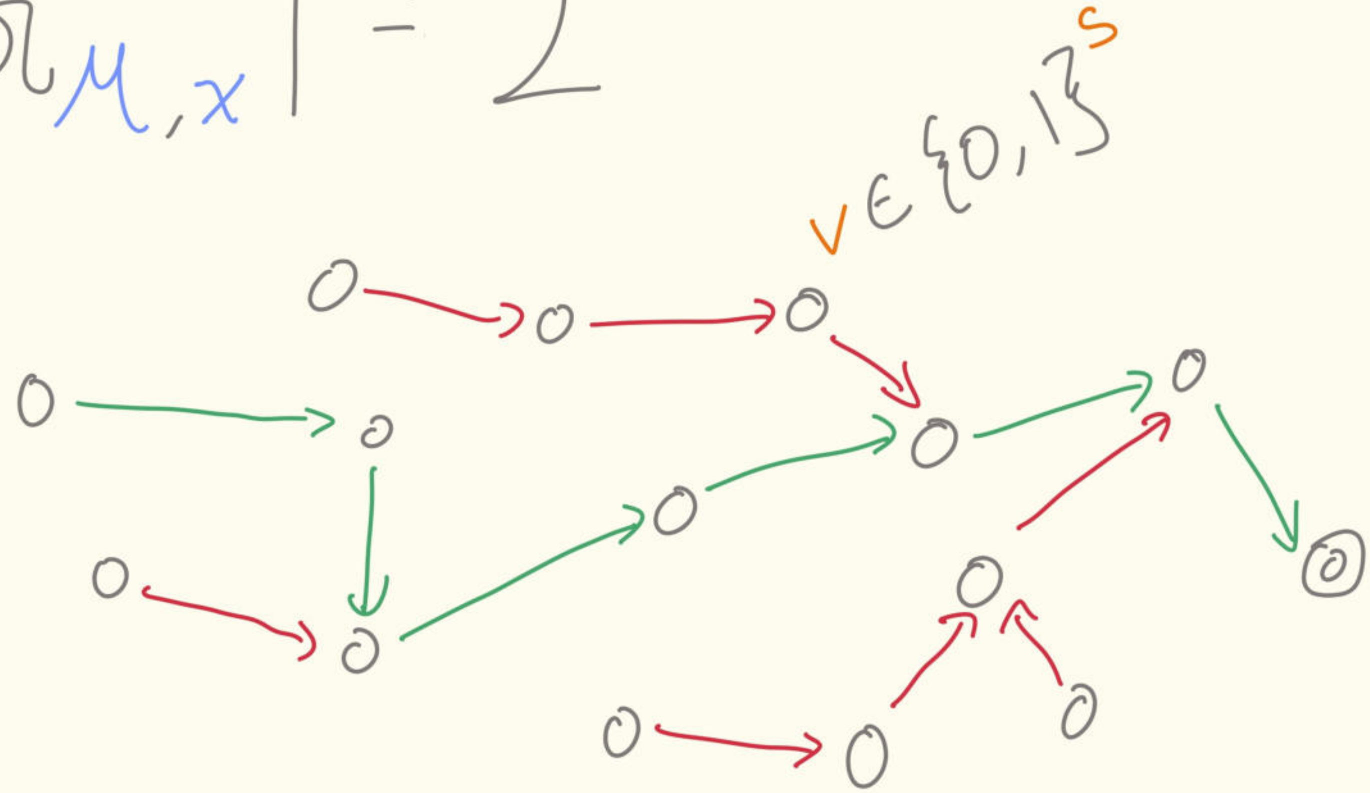
Configuration Graph

$G_{M,x}$



Configuration Graph

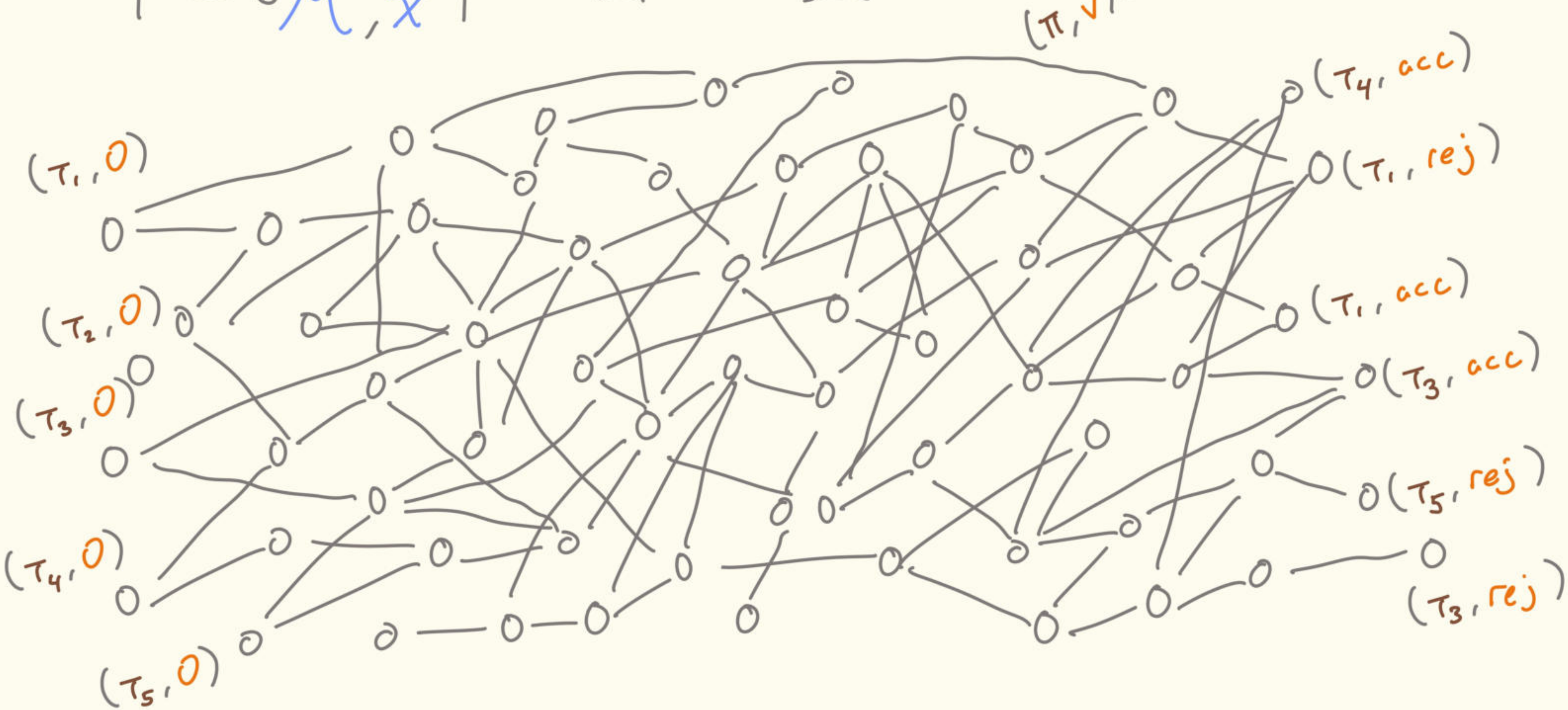
$$|G_{M,x}| = 2^{s}$$



$\rightarrow L \subseteq P$

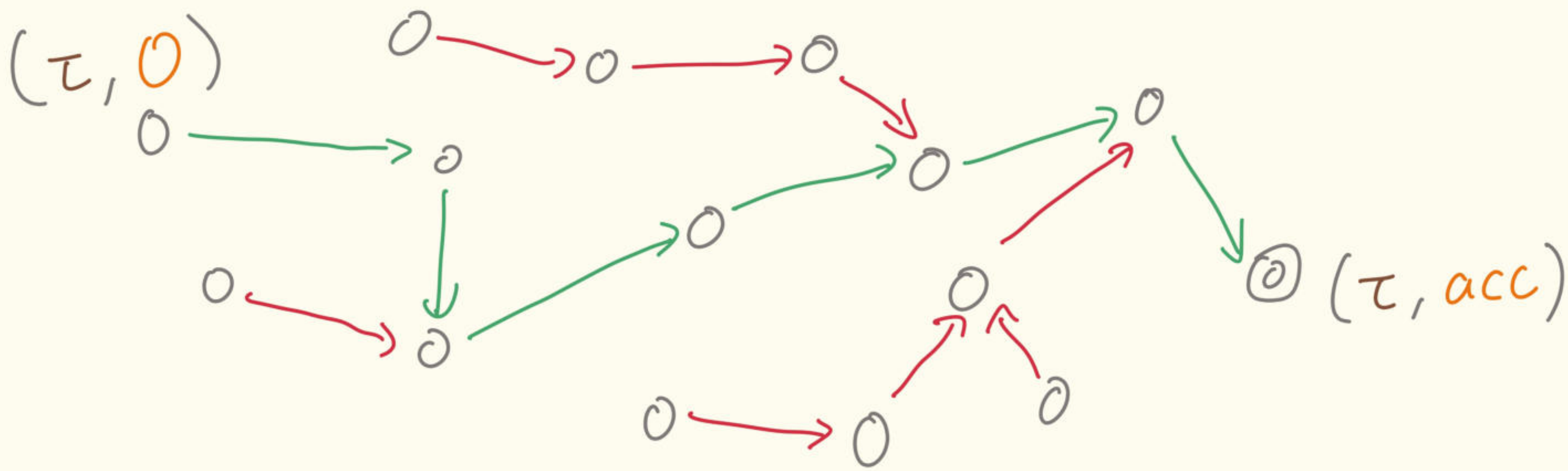
Configuration Graph

$$|G_{M,x}| = 2^c \cdot 2^s$$



Configuration Graph

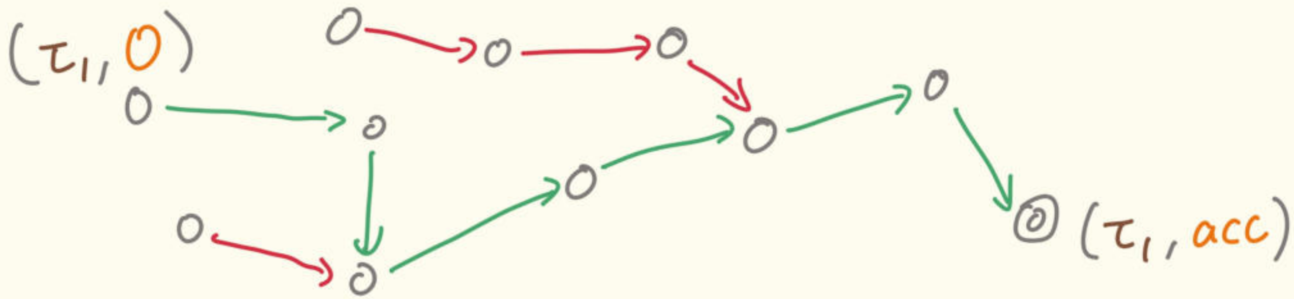
$G_{M,x,\tau}$



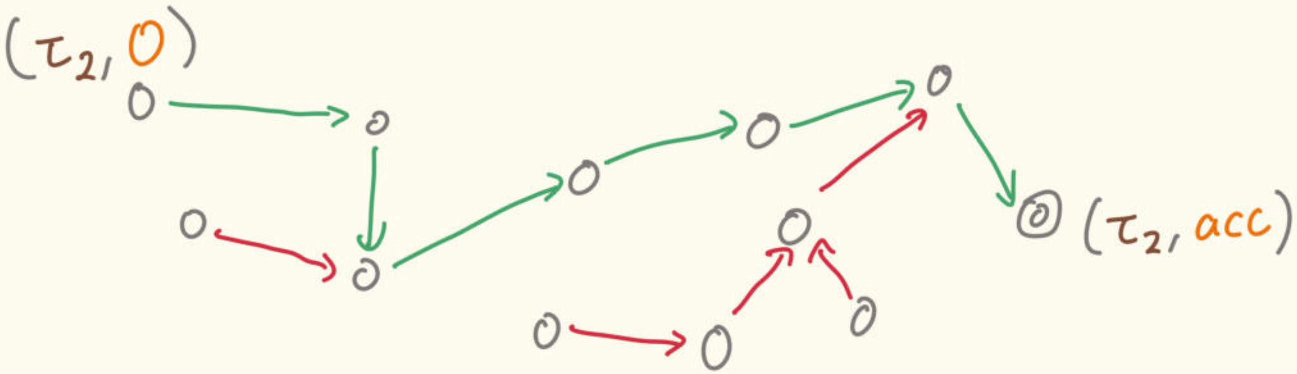
Configuration Graph

$$|G_{M,x}| = 2^{c} \cdot 2^{s}$$

G_{M,x,τ_1}

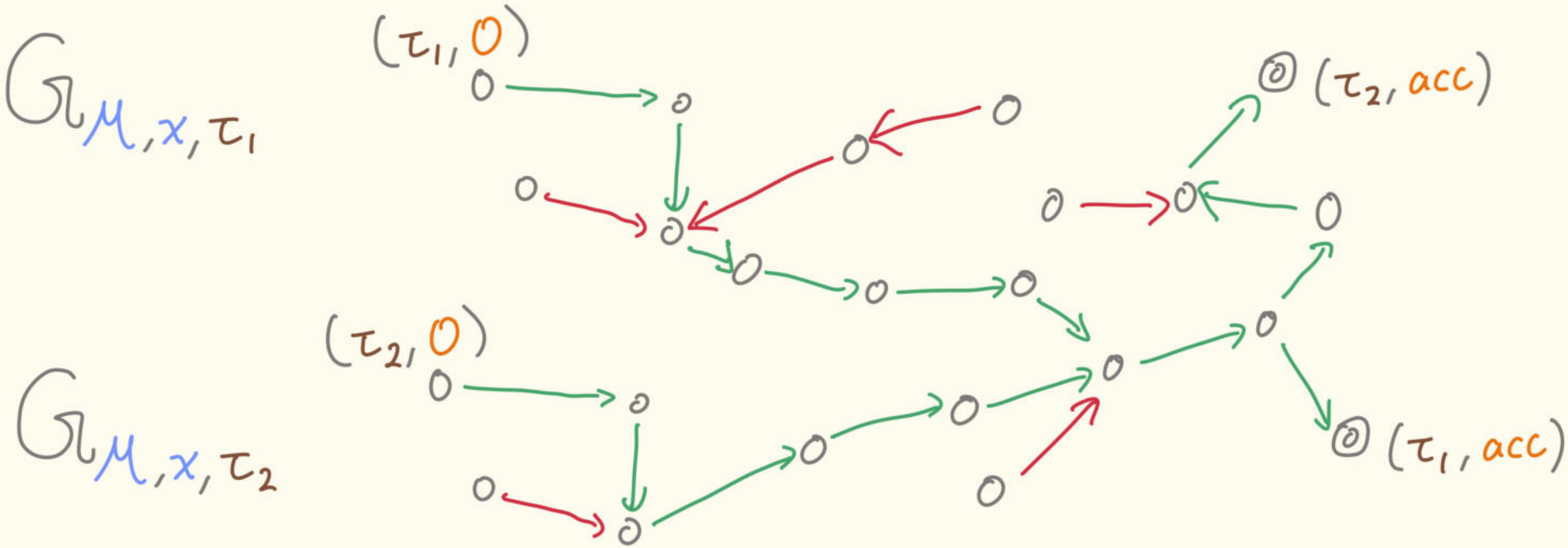


G_{M,x,τ_2}



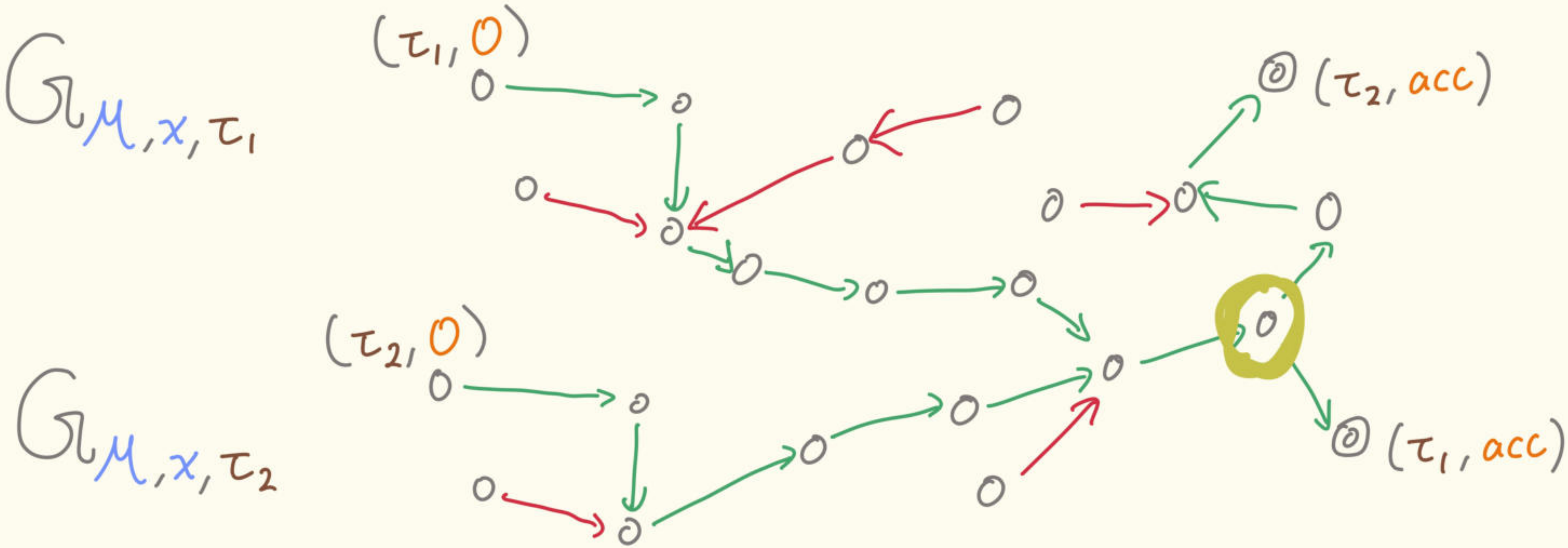
Configuration Graph

$$|G_{M,x}| = 2^{c} \cdot 2^{s}$$



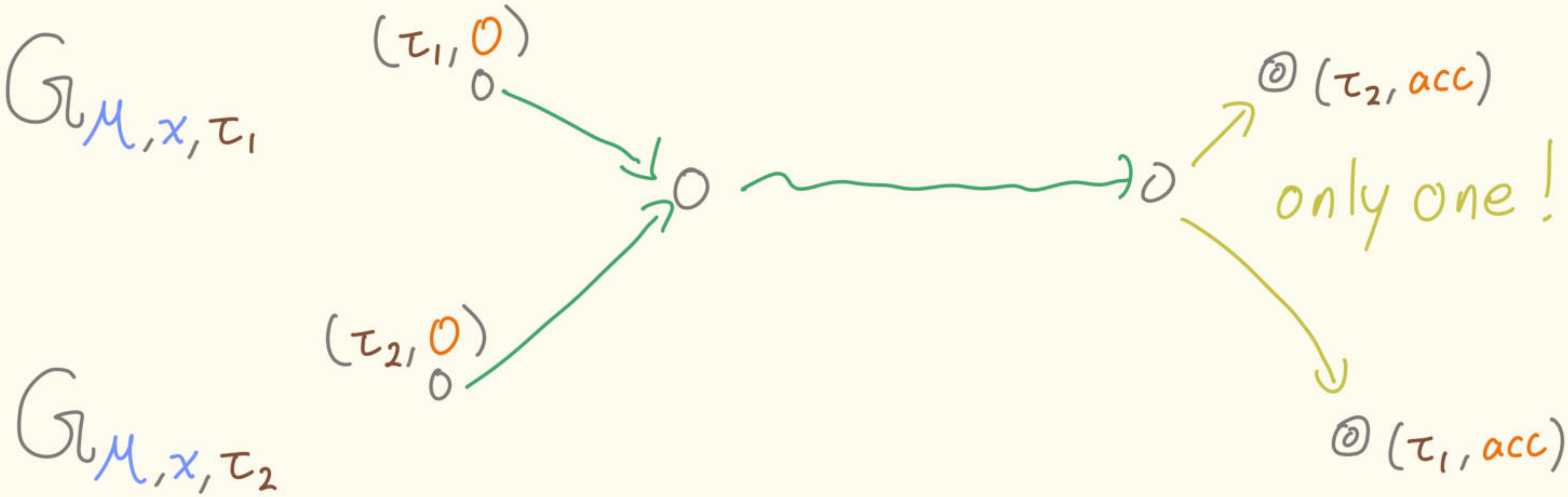
Configuration Graph

$$|G_{M,x}| = 2^{c} \cdot 2^{s}$$



Configuration Graph

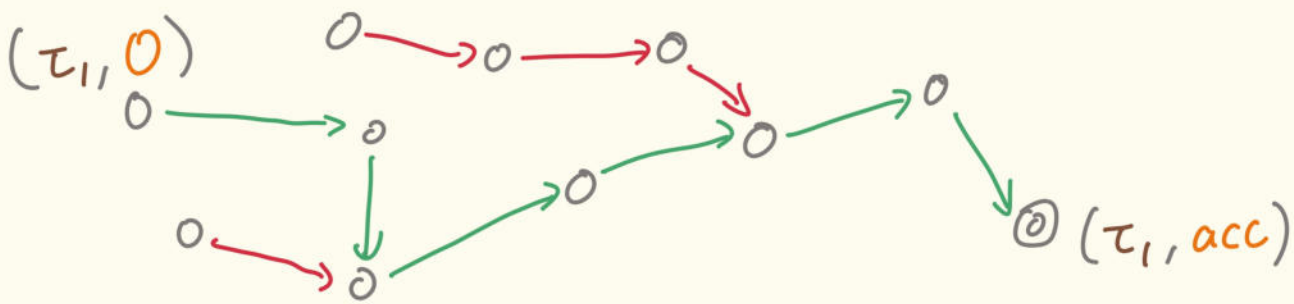
$$|G_{M,x}| = 2^{c} \cdot 2^{s}$$



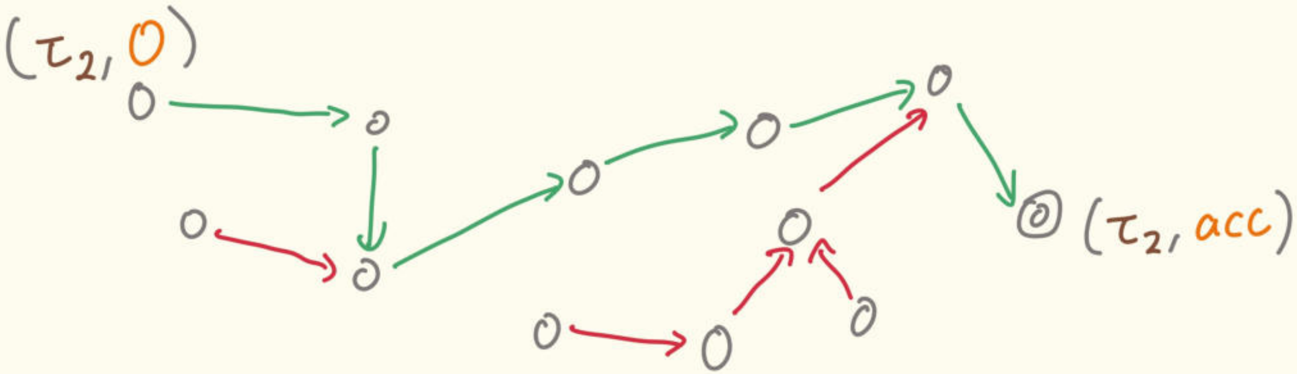
CONFIGURATION GRAPH

$$\sum_{\tau} |G_{M,x,\tau}| \leq 2^{c} \cdot 2^{s}$$

G_{M,x,τ_1}



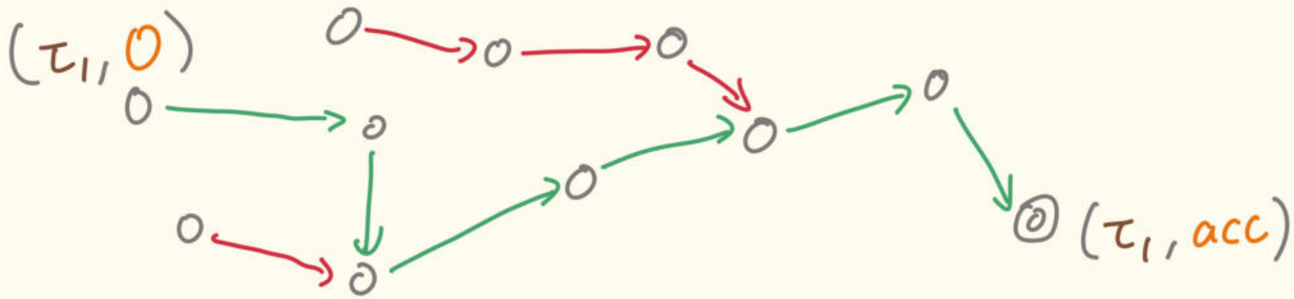
G_{M,x,τ_2}



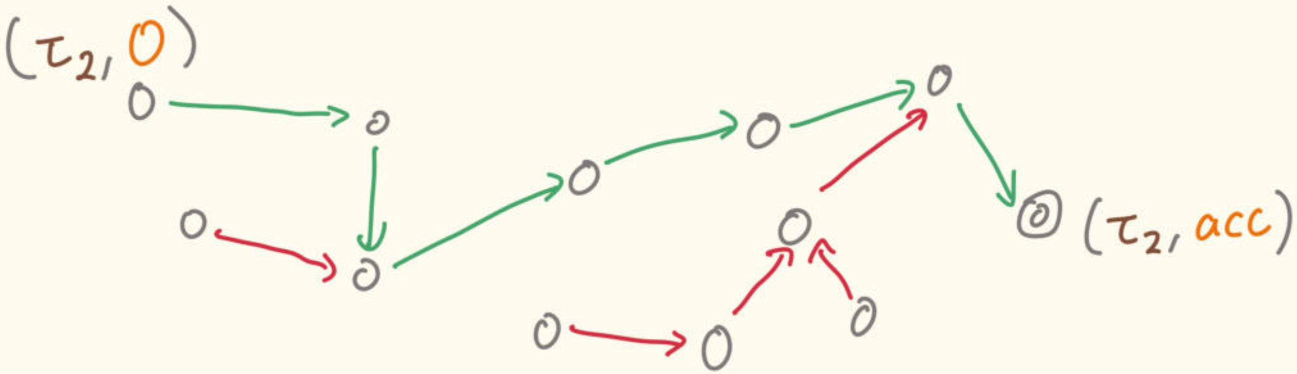
Configuration Graph

$$\mathbb{E}_{\tau} |G_{M,x,\tau}| \leq 2^{S}$$

G_{M,x,τ_1}



G_{M,x,τ_2}



Configuration Graph

$$\mathop{\mathbb{E}}_{\tau} |G_{\mu,x,\tau}| \leq 2^{s}$$

[BCKLS'14]: $CL \subseteq ZPP$

pick random τ, run for 2^{10s} steps, return $\perp$ if no answer $\blacksquare$

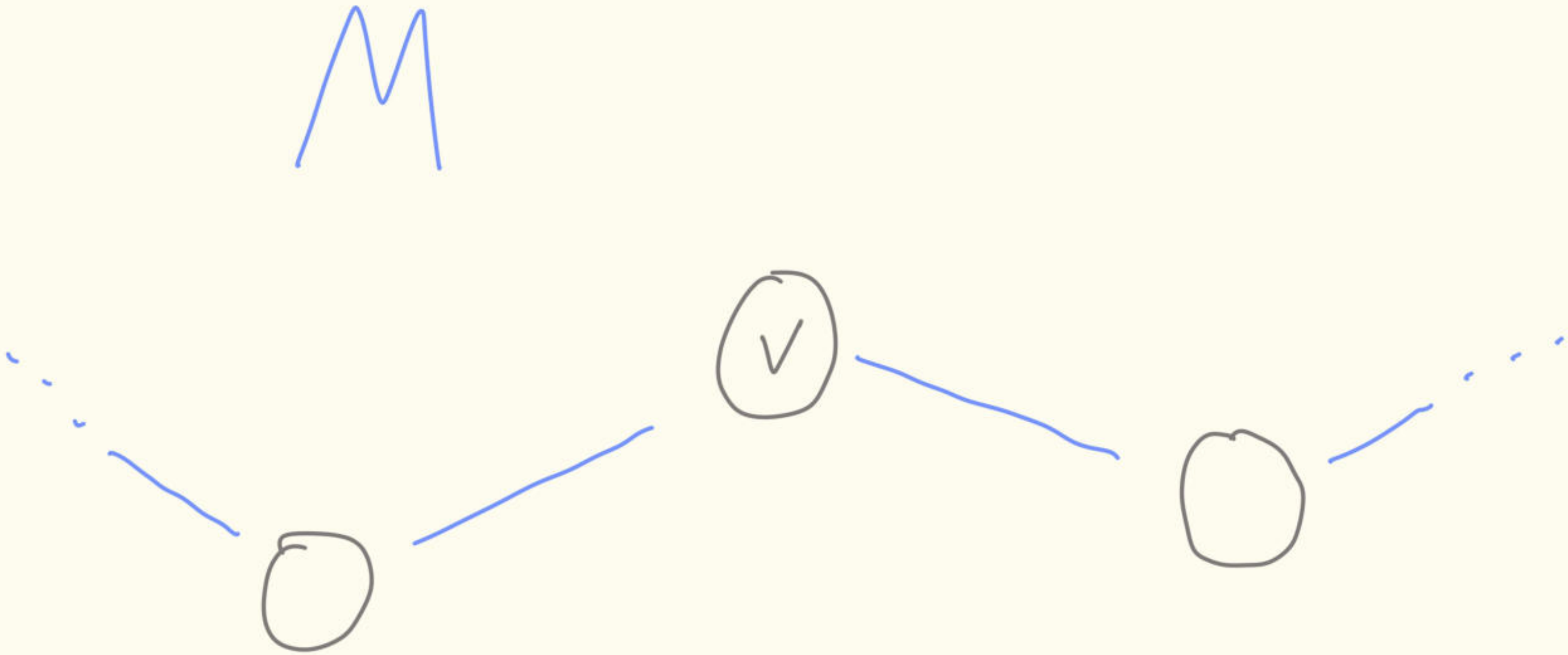
REVERSIBILITY

Reversibility

[LMT'00]: space is reversible

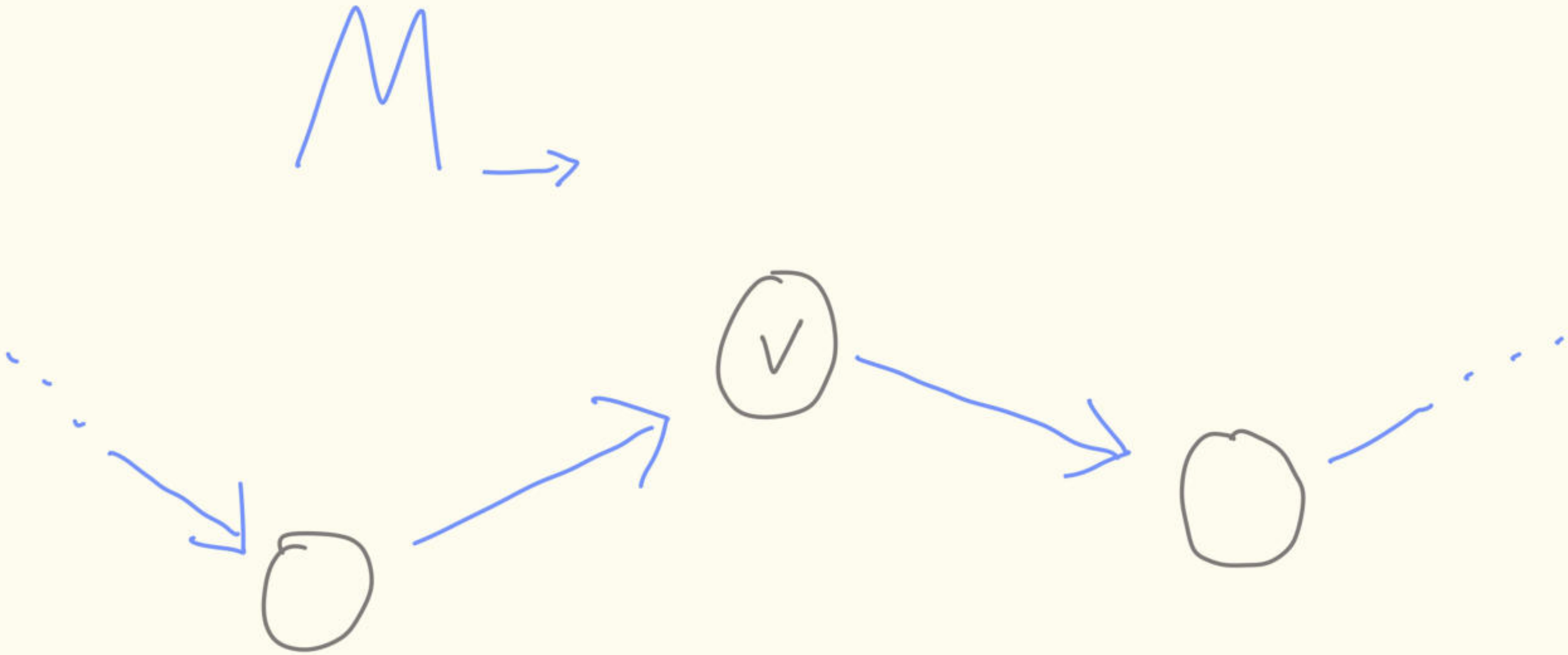
REVERSIBILITY

[LMT'00]: space is reversible



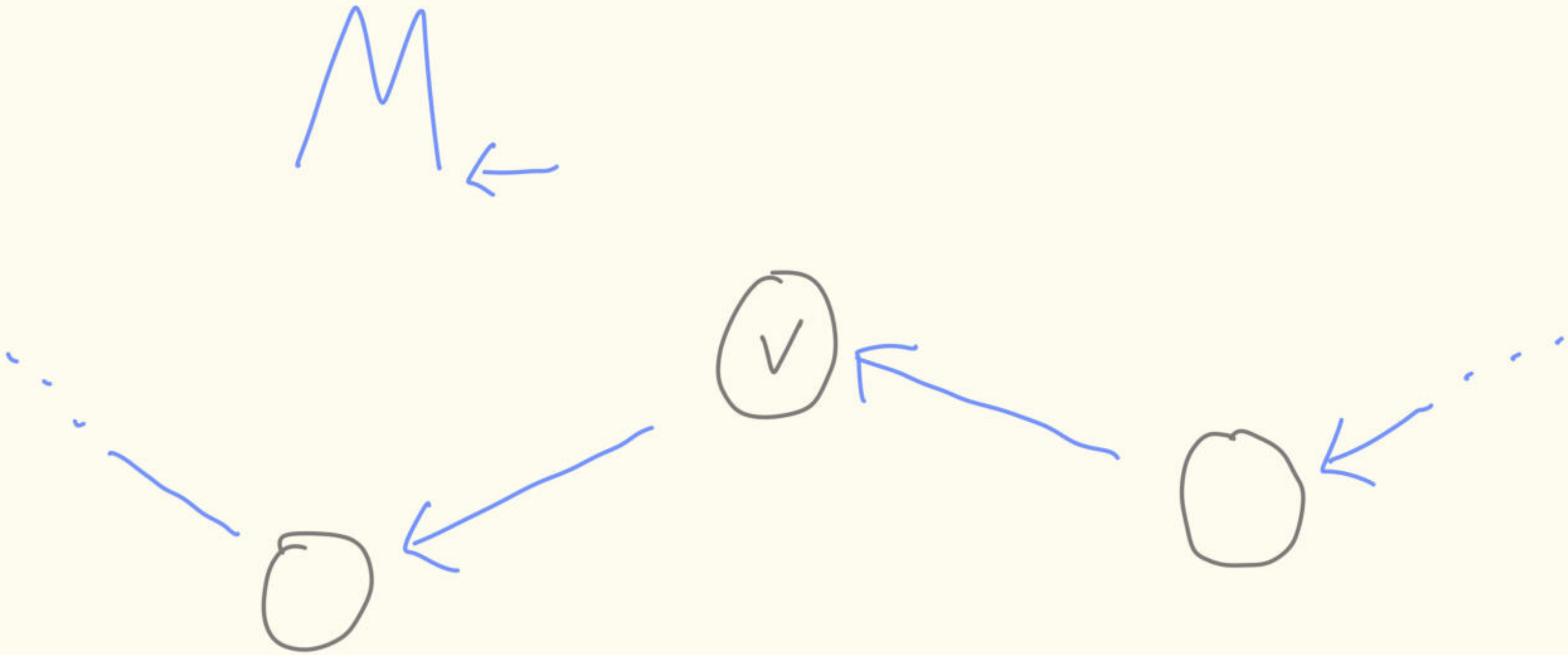
REVERSIBILITY

[LMT'00]: space is reversible



REVERSIBILITY

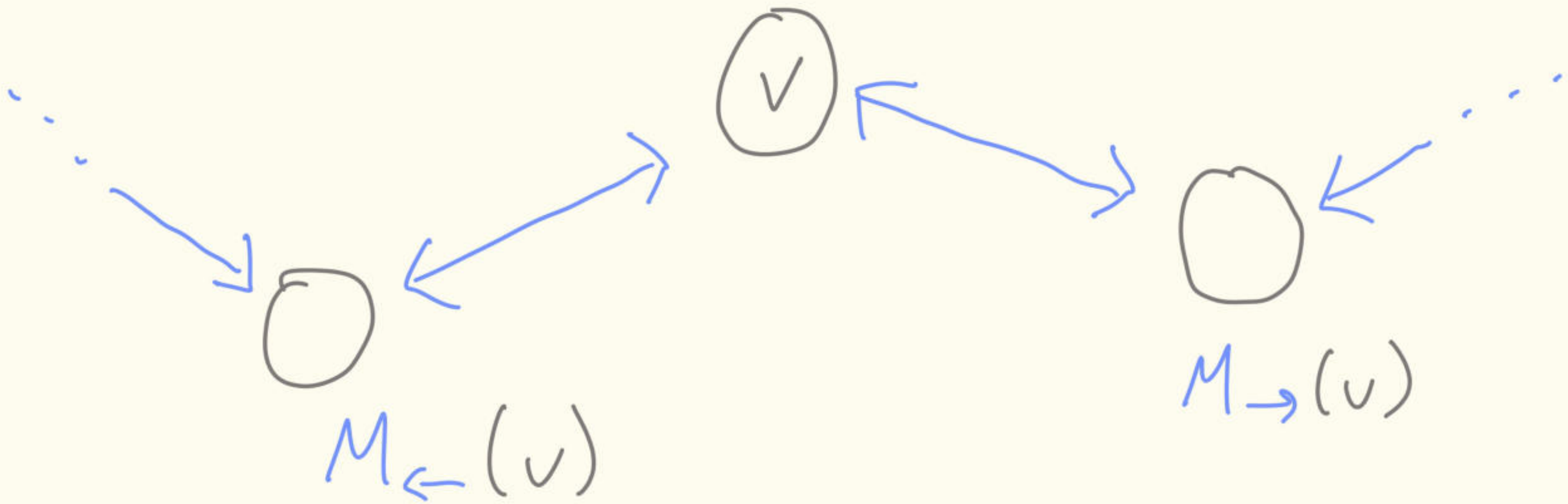
[LMT'00]: space is reversible



REVERSIBILITY

[LMT'00]: space is reversible

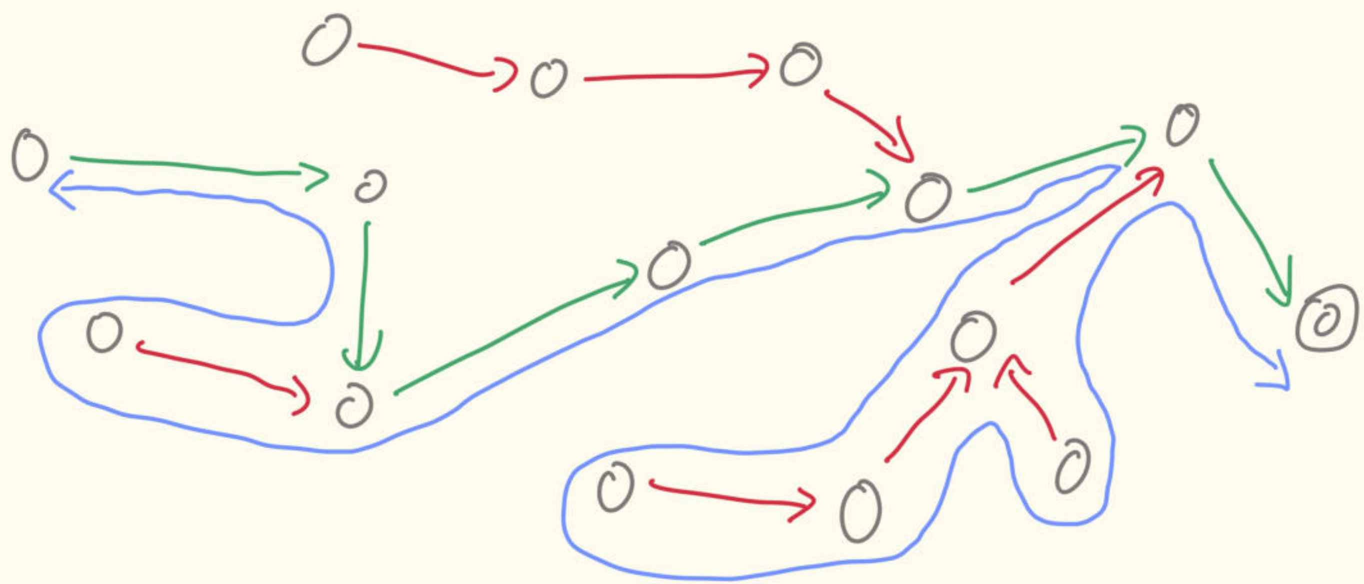
$$M_{\rightarrow}(M_{\leftarrow}(v)) = v = M_{\leftarrow}(M_{\rightarrow}(v))$$



Reversibility

[LMT'00]: space is reversible

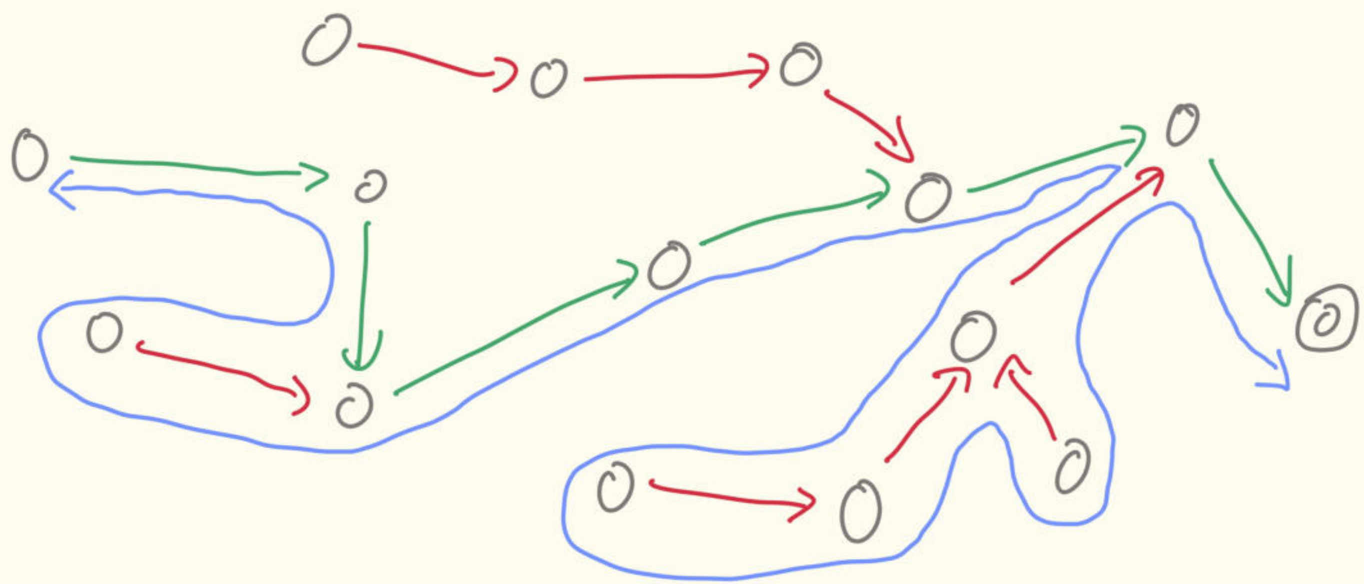
idea: Eulerian tour



REVERSIBILITY

[D'15]: catalytic space is reversible

idea: Eulerian tour



Today

Part I: Results

Prologue: Space & Time

Catalytic Computation

What We Know

Part II: Techniques

Configuration Graphs

Compress-Or-Random

Register Programs

Next Steps

Today

Part I: Results

Prologue: Space & Time

Catalytic Computation

What We Know

Part II: Techniques

Configuration Graphs

Compress-Or-Random

Register Programs

Next Steps

Compress-or-Random

previously...

Options:

1. ~~Delete~~
2. ~~Compress~~

Compress-or-Random

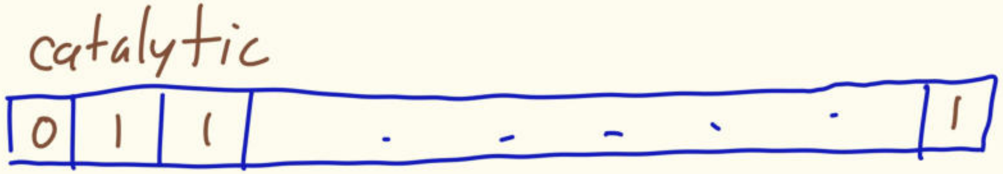
previously...

Options:

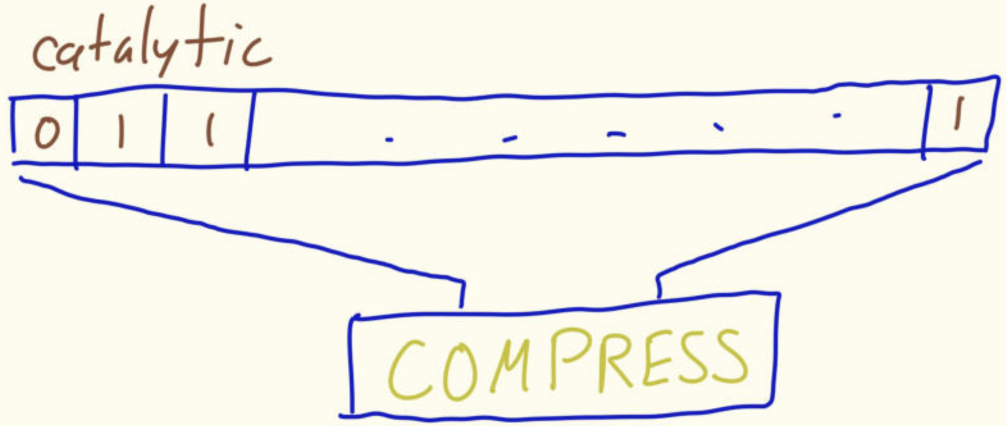
1. ~~Delete~~
2. ~~Compress~~

↳ Incompressible!

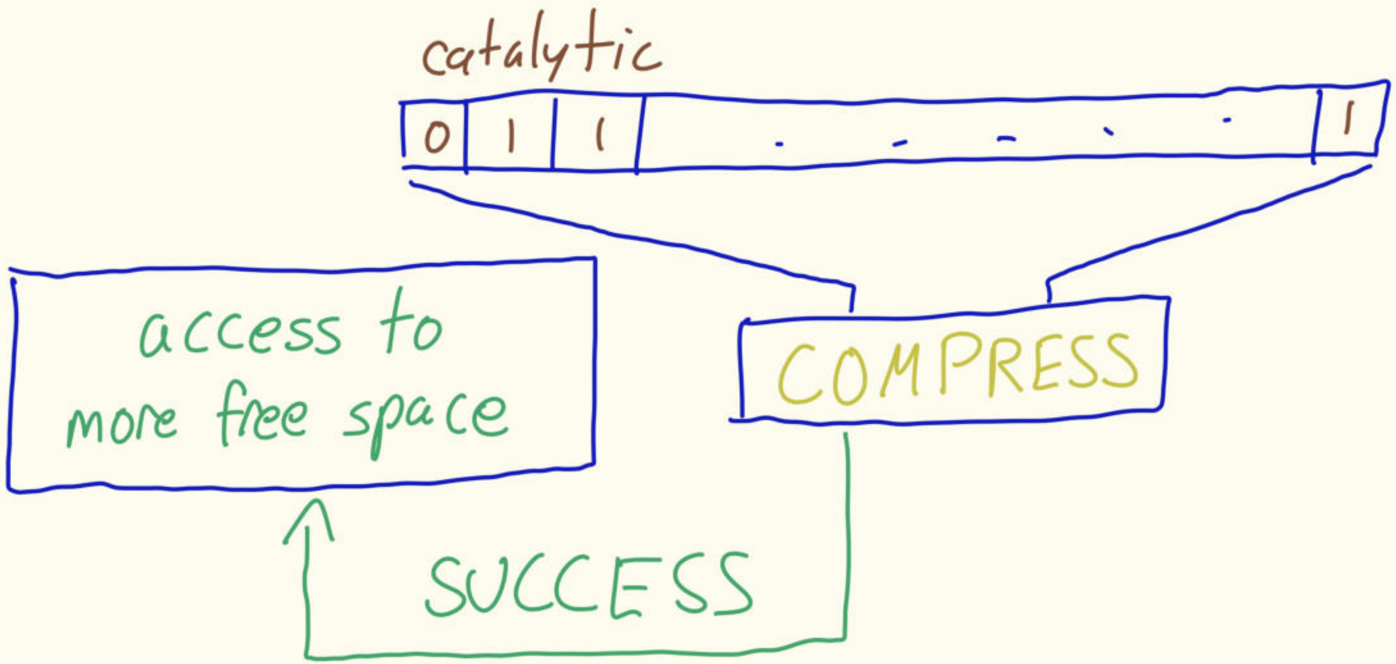
COMPRESS-OR-RANDOM



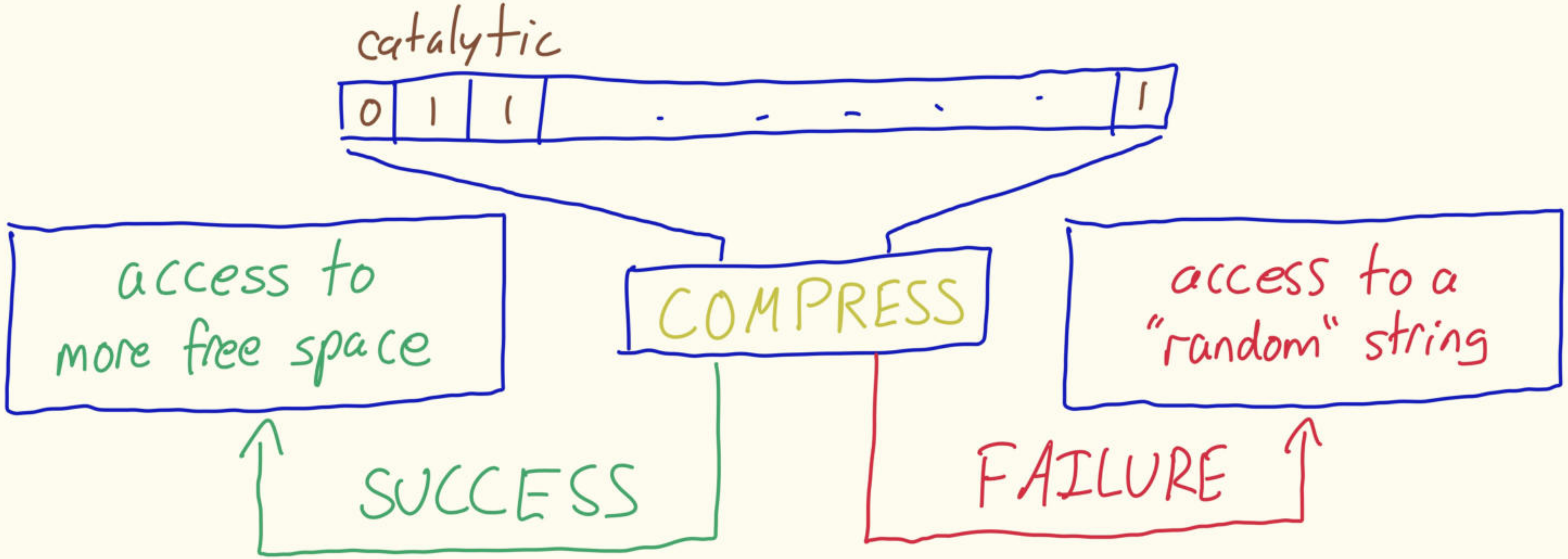
Compress-or-Random



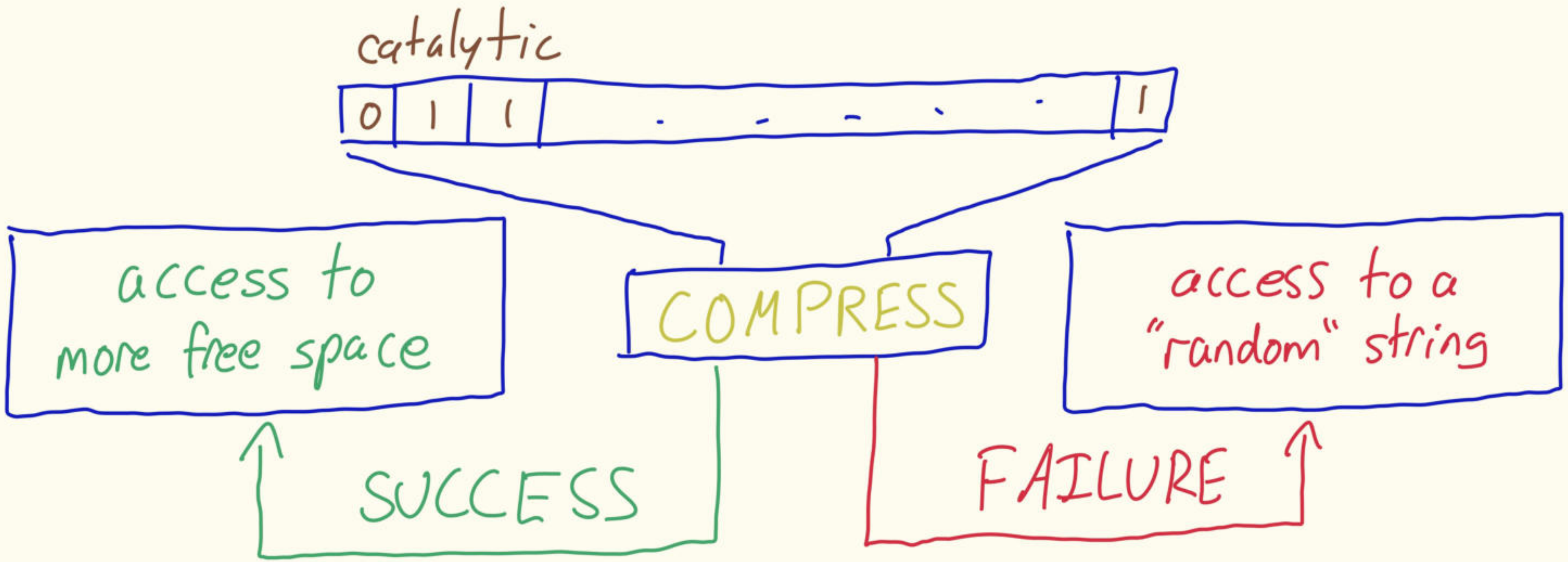
COMPRESS-OR-RANDOM



Compress-or-Random



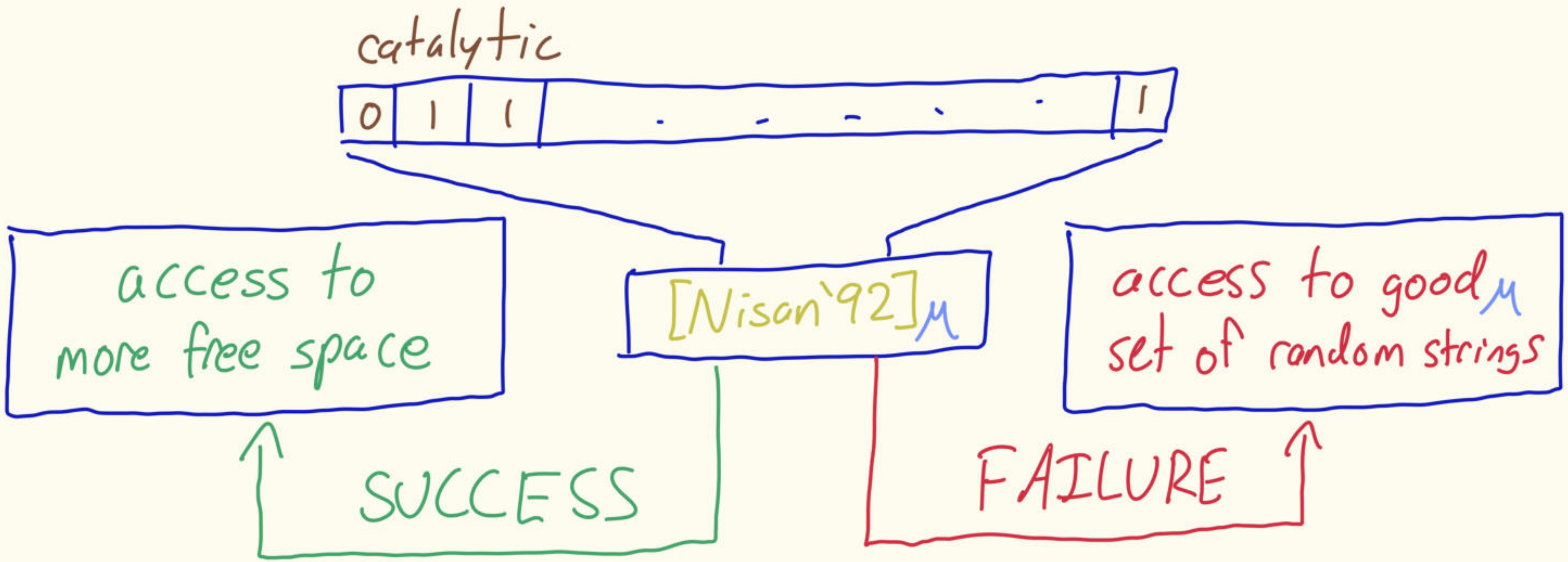
Compress-or-Random



Goal: 1) COMPRESS, DECOMP $\in$ CL;

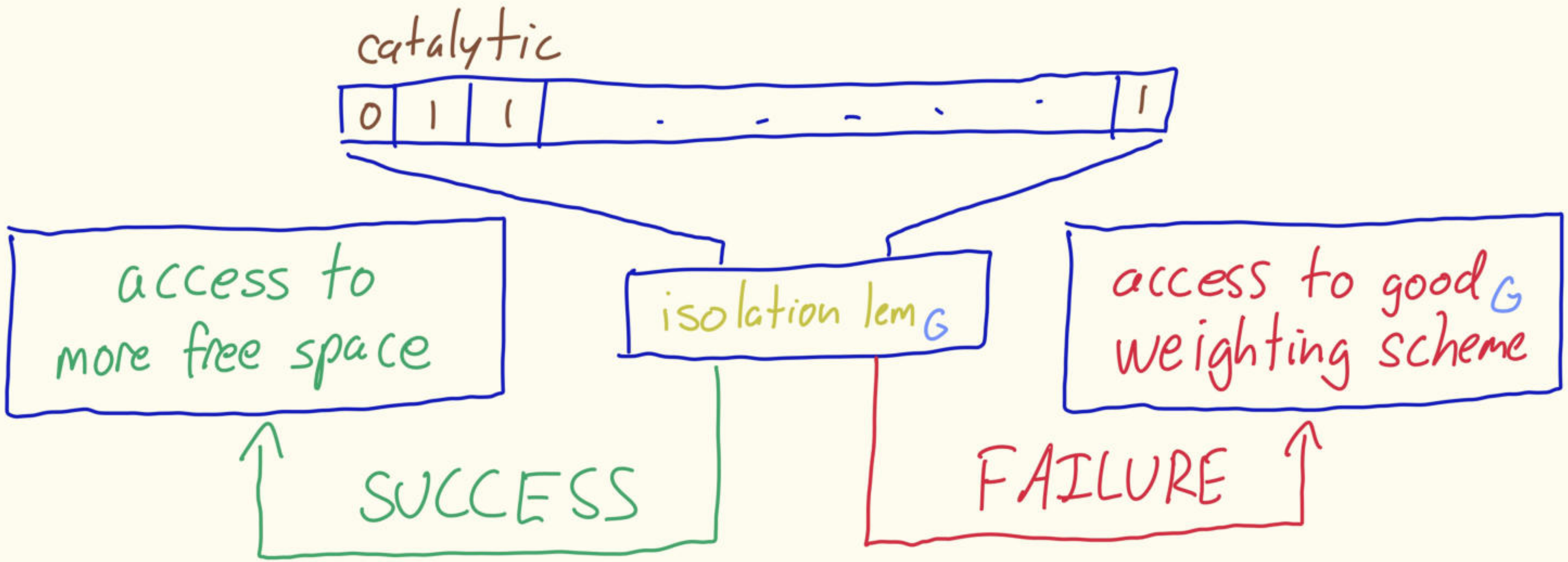
2) FAIL $\rightarrow$ M can solve f w/ access to τ

Compress-or-Random



[Dul'15]: $\mathsf{BPL} \subseteq \mathsf{CL}$

Compress-or-Random



[AM'25]: $\textsc{Match} \in \mathsf{CL}$

Compress-or-Random

previously...

$$\mathbb{E}_{\tau} |G_{M,x,\tau}| \leq 2^{s}$$

[BCKLS'14]: $CL \subseteq ZPP$

pick random τ, run for 2^{10s} steps, return $\perp$ if no answer

COMPRESS-OR-RANDOM

previously...

explicit COMPRESS algorithm?

$$\mathop{\mathbb{E}}_{\tau} |G_{M,x,\tau}| \leq 2^{s}$$

[BCKLS'14]: $CL \subseteq ZPP$

pick random τ, run for 2^{10s} steps, return $\bot$ if no answer

Compress-or-Random

[CLMP'25]: $\mathsf{CL} \cap \mathsf{P} = \text{p-time } \mathsf{CL}$

Compress-or-Random

[CLMP'25]: $CL \cap P =$ p-time CL

$$M_{CL} + M_P \longrightarrow M_{CLP}$$

Compress-or-Random

[CLMP'25]: $\mathsf{CL} \cap \mathsf{P} = \text{p-time } \mathsf{CL}$

$$M_{CL} + M_P \longrightarrow M_{CLP}$$

catalytic tape for M_{CLP}:

$$1\ \boxed{\qquad \tau \qquad}\ c$$

COMPRESS-OR-RANDOM

[CLMP'25]: $CL \cap P = \text{p-time } CL$

$$M_{CL} + M_P \longrightarrow M_{CLP}$$

catalytic tape for M_{CLP}:

$$\boxed{\qquad \tau \qquad}\ c$$

$$+$$

$$\boxed{\quad i \quad}\ s+1$$

Compress-or-Random

[CLMP'25]: $CL \cap P = $ p-time CL

$$M_{CL} + M_P \longrightarrow M_{CLP}$$

catalytic tape for M_{CLP}:

$\boxed{\quad \tau \quad}$ c

$+$

$\boxed{\quad i \quad}$ $s+1$

$\hookrightarrow i \in [2^{s+1}]$

$$\mathbb{E}[i] = 2^s \geq \mathbb{E}_{\tau} |G_{M,x,\tau}|$$

Compress-or-Random

[CLMP'25]: $CL \cap P = $ p-time CL

$$M_{CL} + M_P \longrightarrow M_{CLP}$$

catalytic tape for M_{CLP}:

$\boxed{\quad \tau \quad}$ c

$+$

$\boxed{\quad i \quad}$ $s+1$

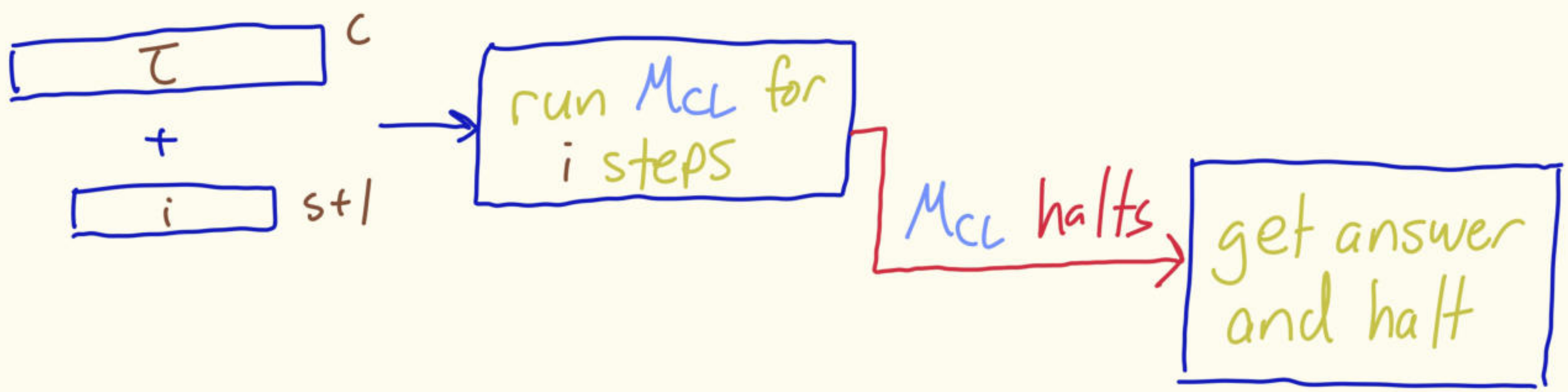
$\longrightarrow$ $\boxed{\text{run } M_{CL} \text{ for } i \text{ steps}}$

Compress-or-Random

[CLMP'25]: $\mathsf{CL} \cap \mathsf{P} = \text{p-time } \mathsf{CL}$

$$M_{CL} + M_P \longrightarrow M_{CLP}$$

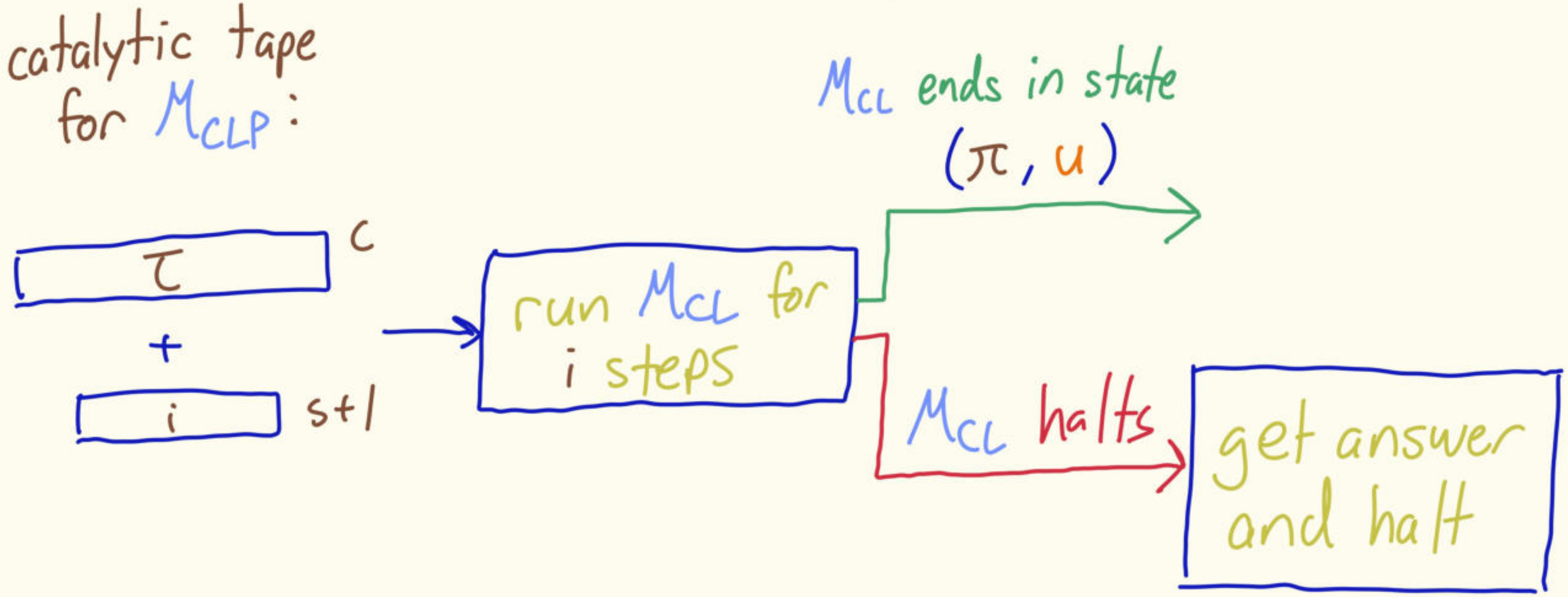
catalytic tape for M_{CLP}:



COMPRESS-OR-RANDOM

[CLMP'25]: $\mathsf{CL} \cap \mathsf{P} = \text{p-time } \mathsf{CL}$

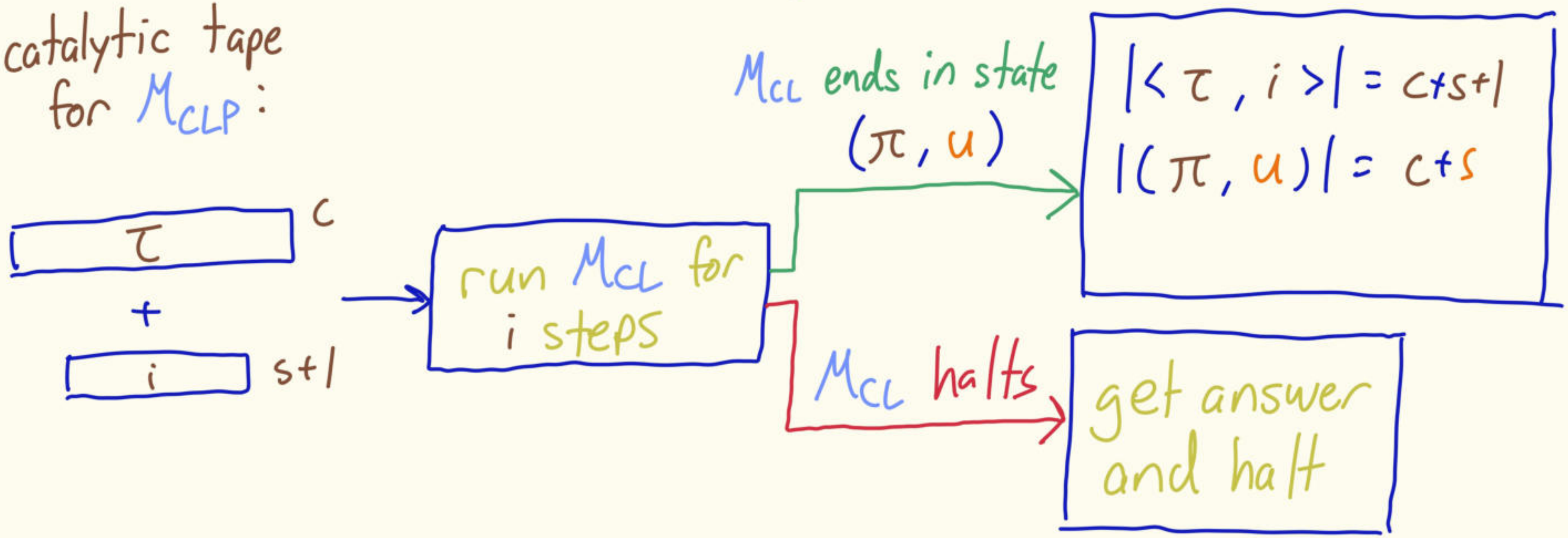
$$M_{CL} + M_P \longrightarrow M_{CLP}$$



Compress-or-Random

[CLMP'25]: $\mathsf{CL} \cap \mathsf{P} = \text{p-time } \mathsf{CL}$

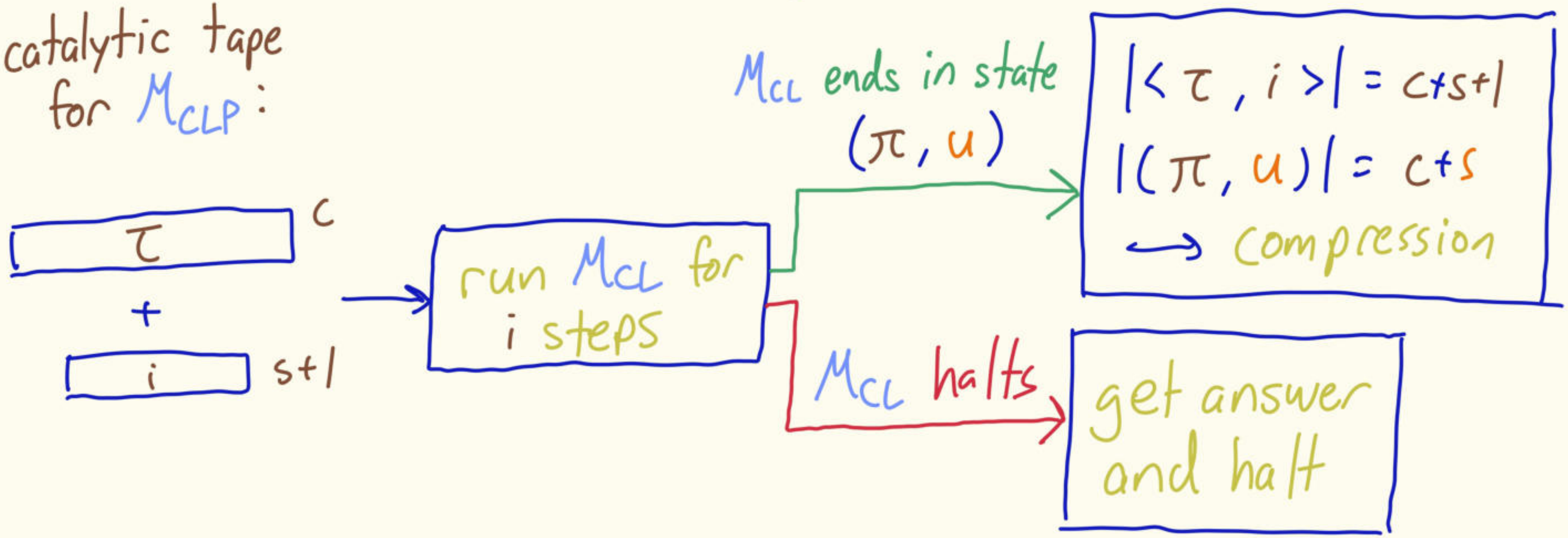
$$M_{CL} + M_P \longrightarrow M_{CLP}$$



Compress-or-Random

[CLMP'25]: $CL \cap P =$ p-time CL

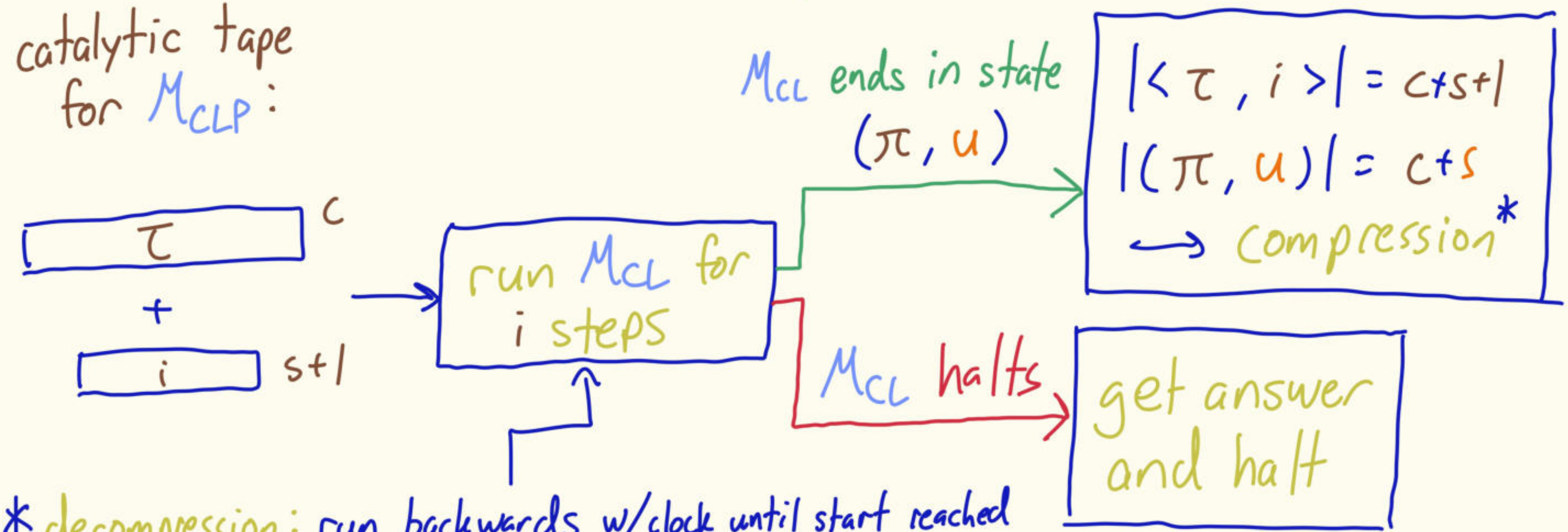
$M_{CL} + M_{P} \longrightarrow M_{CLP}$



Compress-or-Random

[CLMP'25]: $CL \cap P = $ p-time CL

$$M_{CL} + M_P \longrightarrow M_{CLP}$$

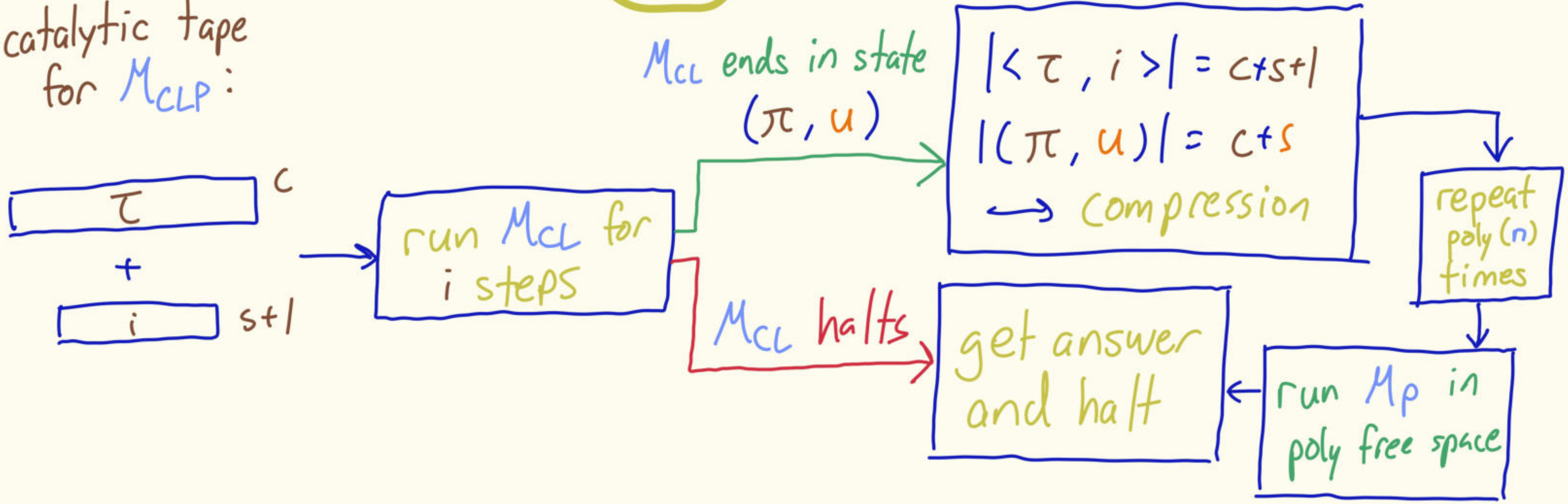


* decompression: run backwards w/clock until start reached

Compress-or-Random

[CLMP'25]: $\mathsf{CL} \cap \mathsf{P} = \text{p-time } \mathsf{CL}$

$$M_{CL} + M_P \longrightarrow M_{CLP}$$



COMPRESS-OR-RANDOM

[CLMP'25]: "$CL \leq_{CL} L$"

COMPRESS-OR-RANDOM

[CLMP'25]: "$\mathsf{CL} \leq_{\mathsf{CL}} \mathsf{L}$"

[KLMP'25]: for many resources $\mathcal{B}$,

"$\mathcal{B}\mathsf{CSPACE}[s, c] \leq_{\mathsf{CSPACE}[s,c]} \mathcal{B}\mathsf{SPACE}[s]$"

Compress-or-Random

[CLMP'25]: "$\mathsf{CL} \leq_{\mathsf{CL}} \mathsf{L}$"

[KLMP'25]: for many resources $\mathcal{B}$,

"$\mathcal{B}\mathsf{CSPACE}[s, c] \leq_{\mathsf{CSPACE}[s,c]} \mathcal{B}\mathsf{SPACE}[s]$"

$\rightarrow \mathsf{NCL} \leq_{\mathsf{CL}} \mathsf{NL} \subseteq \mathsf{CL}$

$\mathsf{BPCL} \leq_{\mathsf{CL}} \mathsf{BPL} \subseteq \mathsf{CL}$

COMPRESS-OR-RANDOM

[CLMP'25]: "$\mathsf{CL} \leq_{\mathsf{CL}} \mathsf{L}$"

[KLMP'25]: for many resources $\mathcal{B}$,

"$\mathcal{B}\mathsf{CSPACE}[s, c] \leq_{\mathsf{CSPACE}[s,c]} \mathcal{B}\mathsf{SPACE}[s]$"

$\to \mathsf{NCL}, \mathsf{BPCL}, \mathsf{AMCL}, \mathsf{PrCL} = \mathsf{CL}$

Today

Part I: Results

Prologue: Space & Time

Catalytic Computation

What We Know

Part II: Techniques

Configuration Graphs

Compress-Or-Random

Register Programs

Next Steps

Register Programs

storage + computation

Cancellation

example : Inner Product

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

CANCELLATION

example : Inner Product

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

$s = 1$

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

$\boxed{\sum x_i y_i}$

CANCELLATION

example: Inner Product

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

$s = n/3 + 1$

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

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

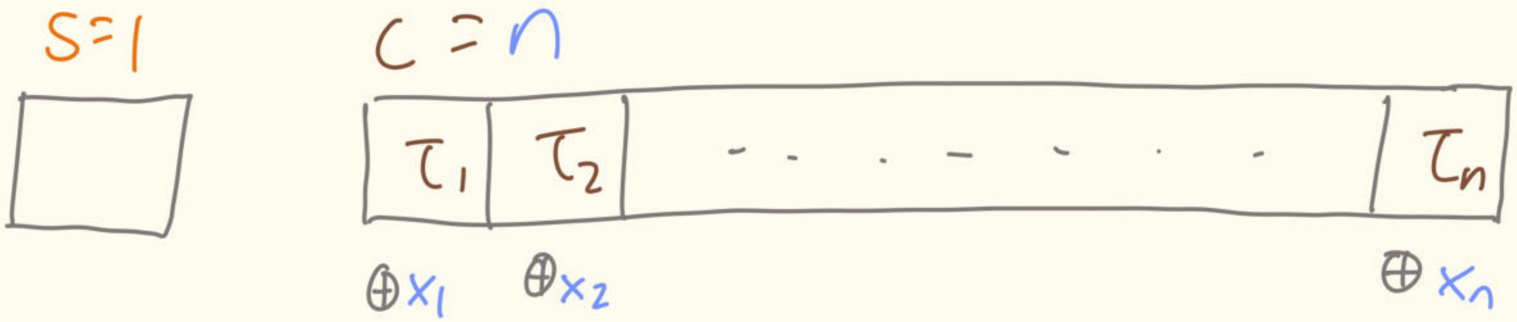
CANCELLATION

example : Inner Product

proof due to [PSW'25]

CANCELLATION

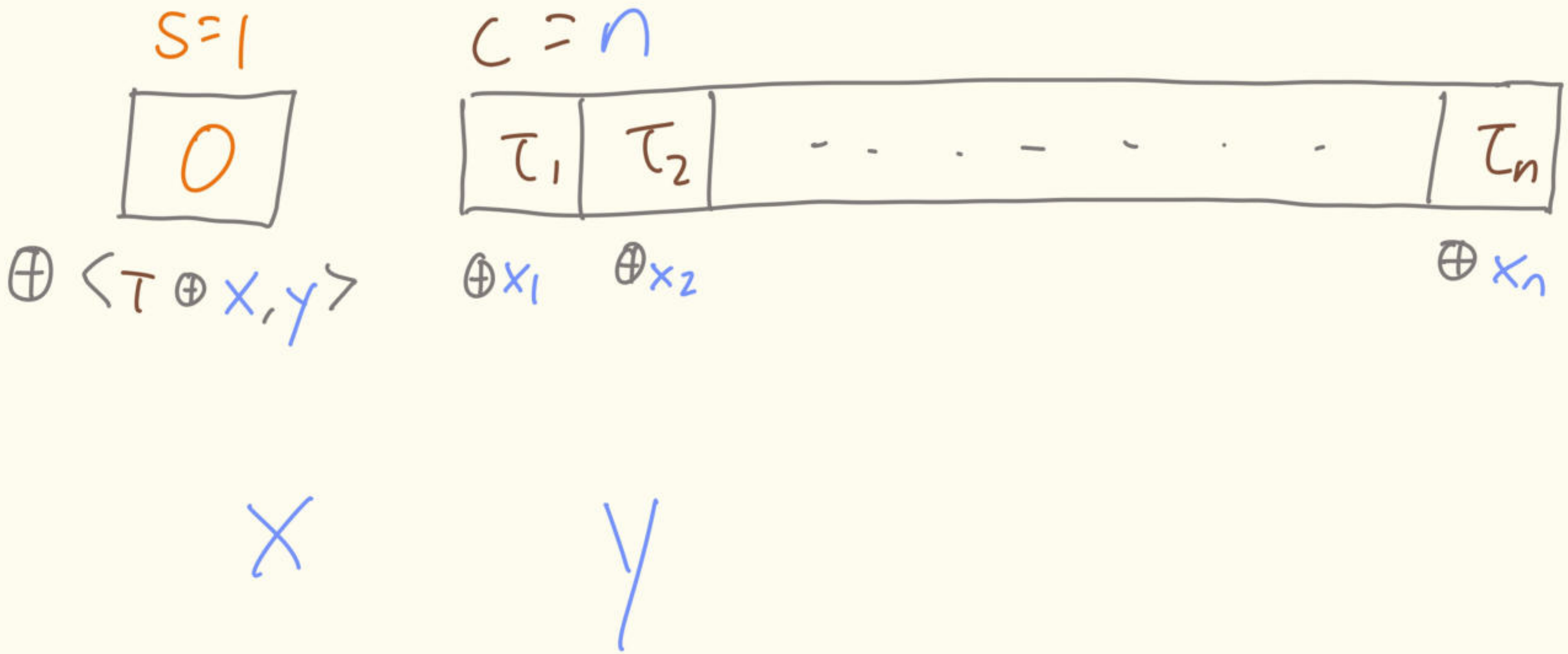
example : Inner Product



X

Cancellation

example : Inner Product



CANCELLATION

example : Inner Product

$s = 1$ $\quad\quad$ $c = n$

$\boxed{0}$ $\quad\quad$ $\boxed{\tau_1 \mid \tau_2 \mid \cdots \mid \tau_n}$

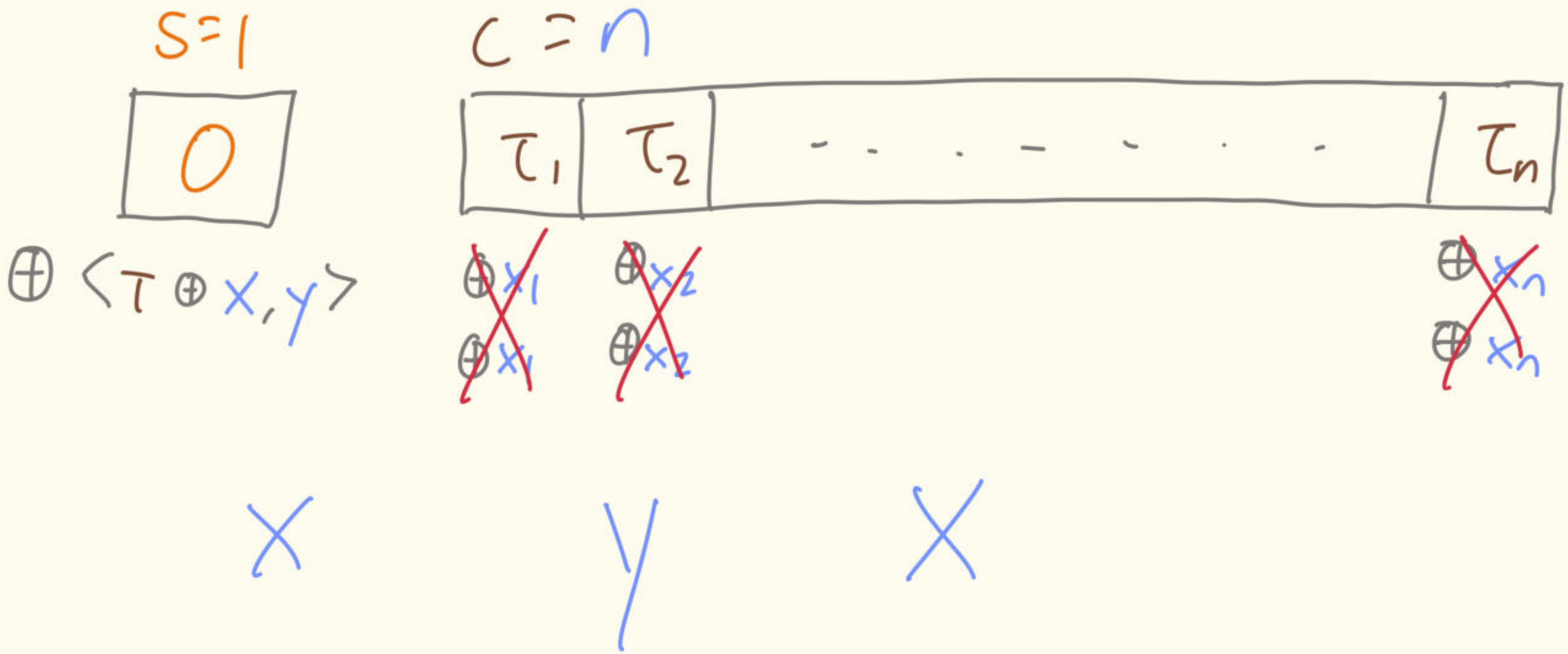
$\oplus \langle \tau \oplus x, y \rangle$ $\quad\quad$ $\oplus x_1 \quad \oplus x_2 \quad \cdots \quad \oplus x_n$

$\oplus x_1 \quad \oplus x_2 \quad \cdots \quad \oplus x_n$

$x \quad y \quad x$

CANCELLATION

example : Inner Product



CANCELLATION

example: Inner Product

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

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

x y x y

CANCELLATION

example : Inner Product

$s = 1$ $\quad$ $\boxed{0}$

$c = n$ $\quad$ $\boxed{\tau_1 \mid \tau_2 \mid \cdots \mid \tau_n}$

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

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

CANCELLATION

example: Inner Product

$s = 1$ $\quad$ $c = n$

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

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

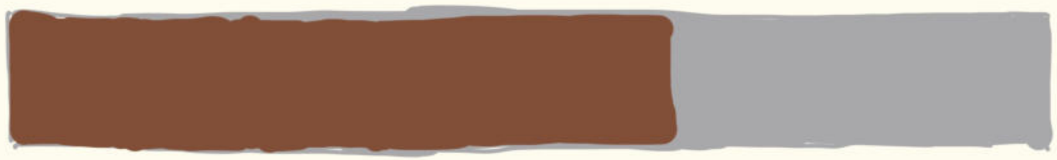
$x \quad y \quad x \quad y$

∎

Reusing Space

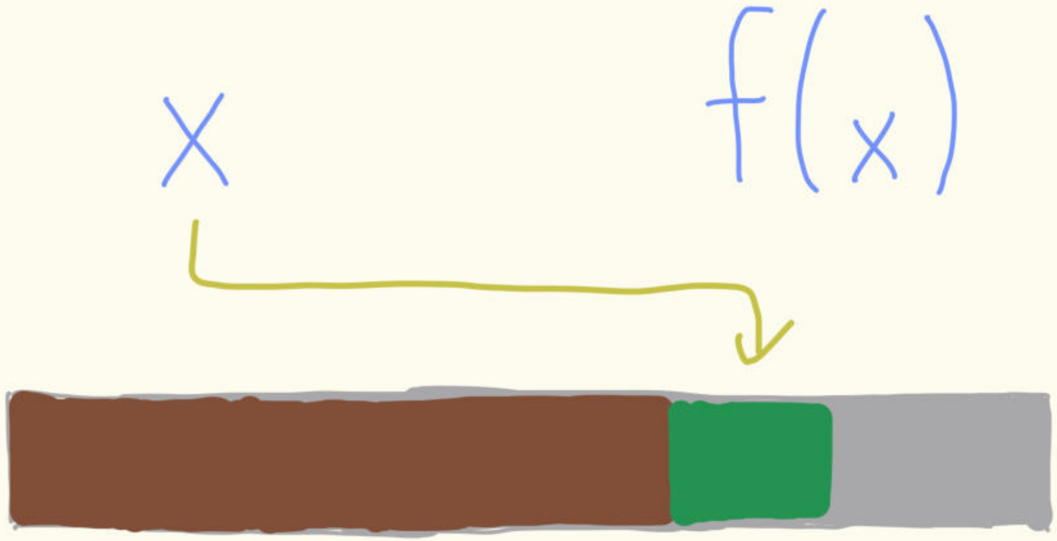
Given: x

Want: $f(x)$



REUSING SPACE

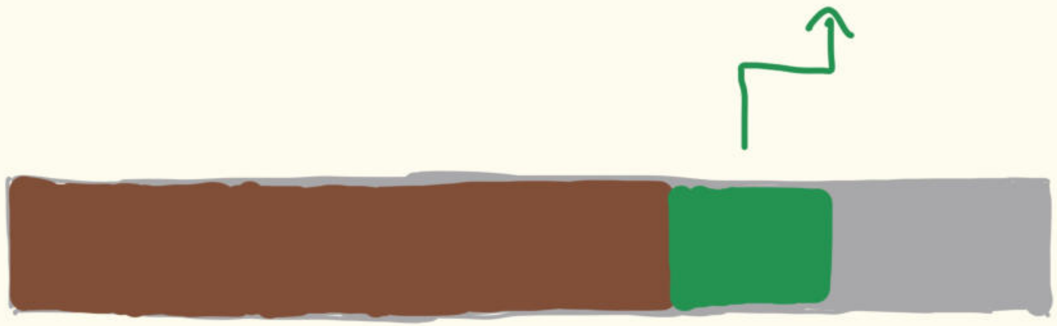
GIVEN: WANT:



Reusing Space

Given: Want:

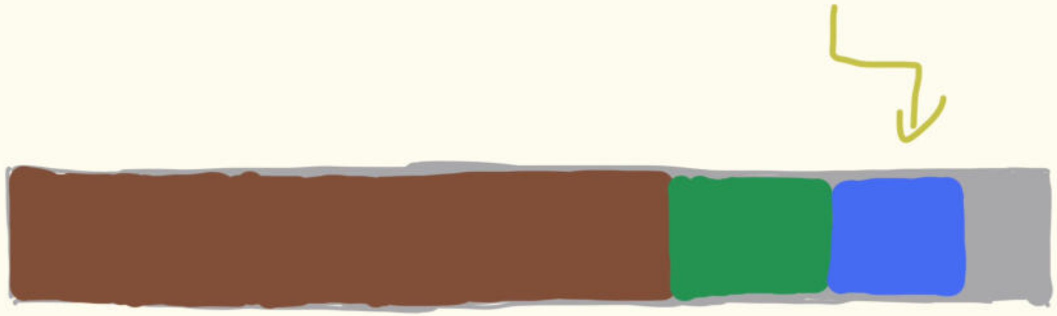
x $f(x)$



Reusing Space

Given: Want:

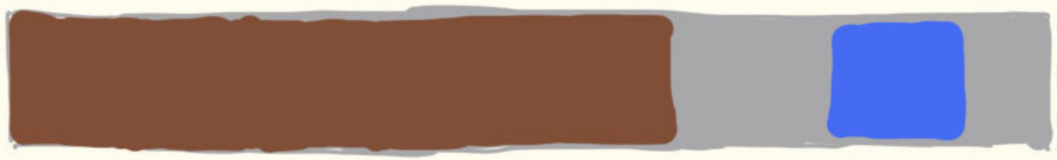
x $f(x)$



Reusing Space

Given: x

Want: $f(x)$



Reusing Space

Given: Want:

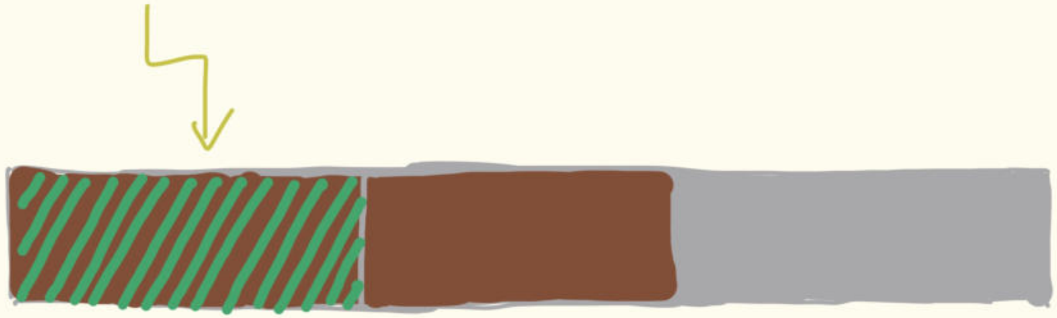
x $f(x)$

storage + computation

Reusing Space

Given: + x

Want: $f(x)$

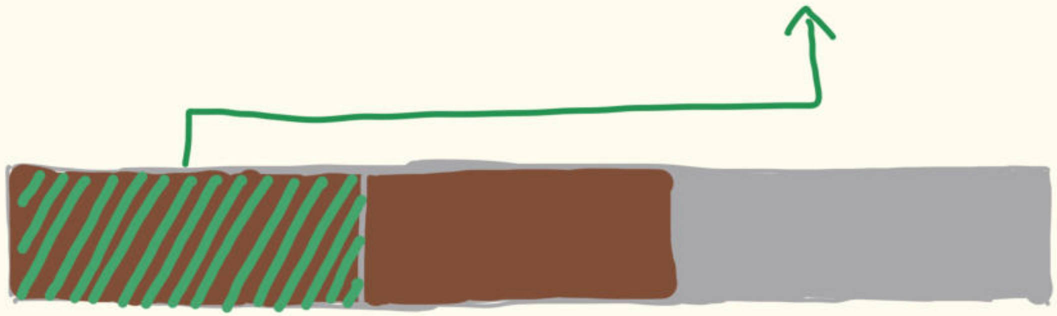


storage + computation

Reusing Space

Given: Want:

+ x $f(x)$

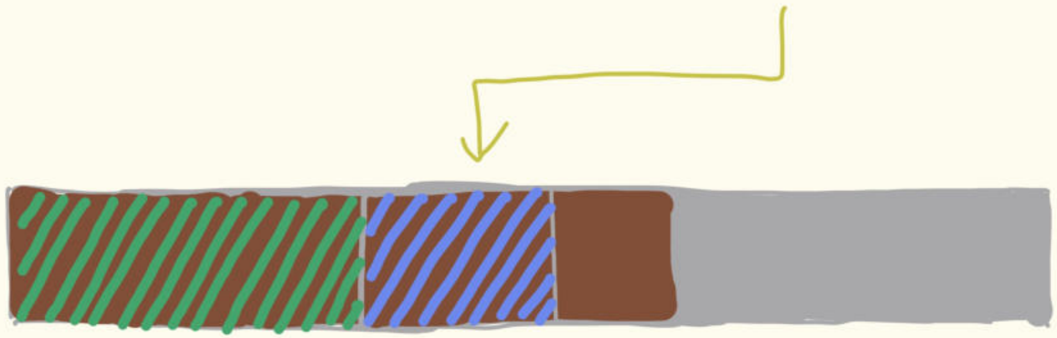


storage + computation

Reusing Space

Given: Want:

+ x + $f(x)$

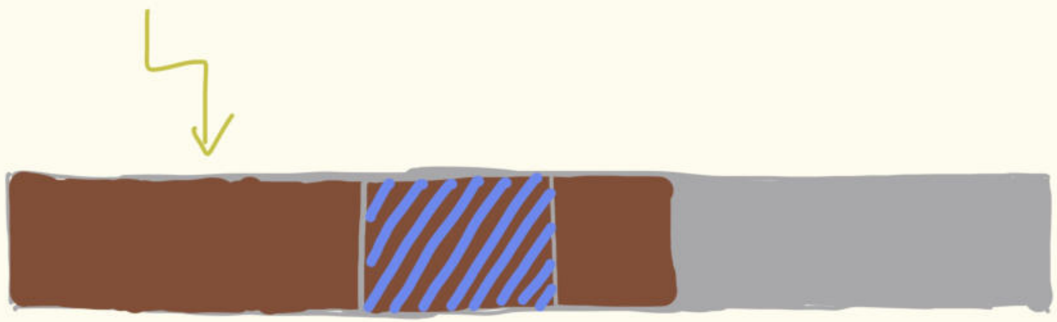


storage + computation

Reusing Space

Given: $\pm x$

Want: $\pm f(x)$



storage + computation

Register Programs

$\tau_i \in \mathbb{F}$

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

Register Programs

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 $\mathbb{F}$

. - . . .

Register Programs

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

$$P_{\vec{x}} : \vec{R} \mathrel{+}= \vec{x}$$

Register Programs

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

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

Register Programs

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

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

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

Register Programs

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

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

$P_{q(x)}$

1. ______
2. $P_{\vec{x}}$
3. ______
4. $P^{-1}_{\vec{x}}$

. . . .

Register Programs

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

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

efficiency:

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

REGISTER PROGRAMS

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

Register Programs

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

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

REGISTER PROGRAMS

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

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

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

Register Programs

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

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

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

[AFMSV'25]: M^k

REGISTER PROGRAMS

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

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

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

[AFMSV'25]: M^k

[CM'20, 21, 24]:

<u>any</u> $f(x_1 \ldots x_n)$

(worse efficiency)

Today

Part I: Results

Prologue: Space & Time

Catalytic Computation

What We Know

Part II: Techniques

Configuration Graphs

Compress-Or-Random

Register Programs

Next Steps

The Future of Reuse

1. Give a simple, direct proof of uSTConn $\in$ L.
2. Give a simple, direct proof of uSTConn $\in$ CL.
3. Give a simple, direct proof of STConn $\in$ CL.
4. Try to improve Savich's Theorem: prove NSPACE$(s) \subseteq$ SPACE$(o(s^2))$.
5. Improve the deterministic space complexity of BPSPACE(s).
6. Decide the space complexity of TreeEval.
7. Give a register program for computing any polynomial $p(x_1 \ldots x_n)$ 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 poly (ℓ) 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—SPACE(s) but which is computable in CSPACE$(\ll s, \approx s)$.
11. Prove CL $\subseteq$ P.
12. Show that P $\not\subseteq$ L/poly implies CL $\subseteq$ P.
13. Show that CL $\subseteq$ P would give strong evidence ZPP $\subseteq$ 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 BPNC$^1 \subseteq$ 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 CL$/poly$, and does it have a natural syntactic characterization?
19. Show the existence of an oracle D such that CL^D = EXP^D.
20. Extend the BPL $\subseteq$ CL simulation to show CBPL $\subseteq$ 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 CNSPACE$(s, c) \subseteq$ 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 TC$^1 \subseteq$ 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]

The Future of Reuse

1. Give a simple, direct proof of uSTConn $\in$ L.
2. Give a simple, direct proof of uSTConn $\in$ CL.
3. Give a simple, direct proof of STConn $\in$ CL.
4. Try to improve Savich's Theorem: prove NSPACE$(s) \subseteq$ SPACE$(o(s^2))$.
5. Improve the deterministic space complexity of BPSPACE(s).
6. Decide the space complexity of TreeEval.
7. Give a register program for computing any polynomial $p(x_1 \ldots x_n)$ 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 poly (ℓ) 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—SPACE(s) but which is computable in CSPACE$(\ll s, \approx s)$.
11. Prove CL $\subseteq$ P.
12. Show that P $\not\subseteq$ L/poly implies CL $\subseteq$ P.
13. Show that CL $\subseteq$ P would give strong evidence ZPP $\subseteq$ 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 BPNC^1 $\subseteq$ 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 CL/*poly*, and does it have a natural syntactic characterization?
19. Show the existence of an oracle D such that CL^D = EXP^D.
20. Extend the BPL $\subseteq$ CL simulation to show CBPL $\subseteq$ 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 CNSPACE$(s, c) \subseteq$ 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 TC^1 $\subseteq$ 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]

The Future of Reuse

catalytic + quantum : reversible computation

The Future of Reuse

catalytic + quantum : reversible computation

~~25~~. What does quantum catalytic space look like? see [BFMSSST'25]

26. Devise a register program using basic instructions inspired by unitary computation, and use it to show non-trivial results for e.g. BQP.

The Future of Reuse

catalytic + cryptography: homomorphic computation?

The Future of Reuse

catalytic + cryptography: homomorphic computation?

derandomization?

other?

The Future of Reuse

catalytic + cryptography: TBD

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.

The Future of Reuse

catalytic + network coding : arithmetization

The Future of Reuse

catalytic + network coding : arithmetization

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.

The Future of Reuse

24. Implement a catalytic algorithm such that it is actually useful.

(see James Cook's blog!)

Concluding Remarks

Summing Up

Summing Up

- memory can be used for both storage and computation at the same time

SUMMING UP

- memory can be used for both storage and computation at the same time

- space reuse has led to breakthroughs on the study of time and space

Summing Up

- memory can be used for both storage and computation at the same time
- space reuse has led to breakthroughs on the study of time and space
- catalytic computing is a rich area for study with many open directions & applications

RESOURCES

REUSING SPACE: TECHNIQUES AND OPEN PROBLEMS

Ian Mertz[*]

Abstract

In the world of space-bounded complexity, there is a strain of results showing that space can, somewhat paradoxically, be used for multiple purposes at once. Touchstone results include Barrington's Theorem and the recent line of work on catalytic computing. We refer to such techniques, in contrast to the usual notion of reclaiming space, as *reusing space*.

In this survey we will dip our toes into the world of reusing space. We do so in part by studying techniques, viewed through the lens of a few highlight results, but our main focus will be the wide variety of open problems in the field.

In addition to the broader and more challenging questions, we aim to provide a number of questions that are fairly simple to state, have clear practical and theoretical implications, and, most importantly, that a newcomer with little background experience can still sit down and play with for a while.

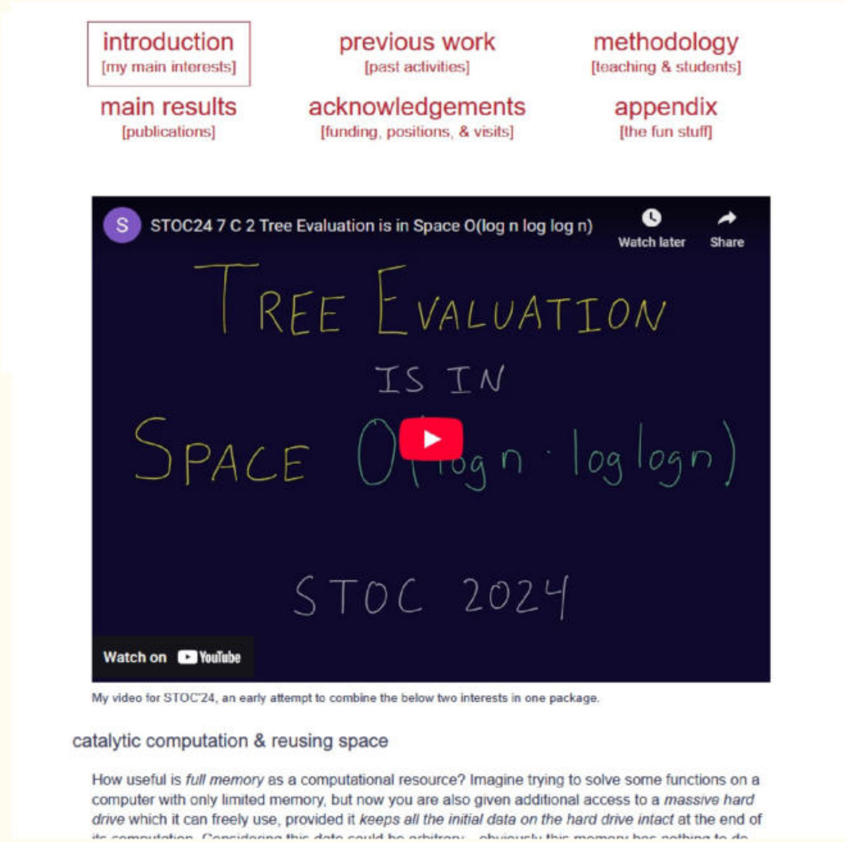
CATALYTIC COMPUTATION

Michal Koucký[*]
Computer Science Institute
Charles University, Prague
koucky@iuuk.mff.cuni.cz

Abstract

Catalytic computation was defined by Buhrman et al. (STOC, 2014). It addresses the question whether memory, that already stores some unknown data that should be preserved for later use, can be meaningfully used for computation. Buhrman et al. provide an intriguing answer to this question by giving examples where the occupied memory can be used to perform computation. In this expository article we survey what is known about this problem and how it relates to other problems.

Resources



introduction
[my main interests]

previous work
[past activities]

methodology
[teaching & students]

main results
[publications]

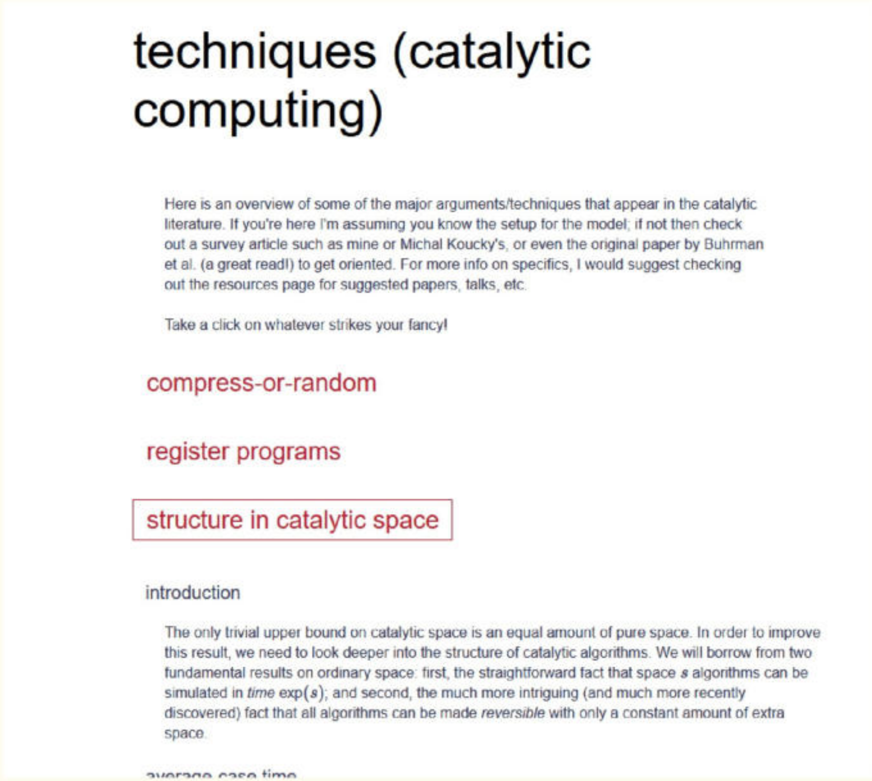
acknowledgements
[funding, positions, & visits]

appendix
[the fun stuff]

My video for STOC'24, an early attempt to combine the below two interests in one package.

catalytic computation & reusing space

How useful is *full memory* as a computational resource? Imagine trying to solve some functions on a computer with only limited memory, but now you are also given additional access to a *massive hard drive* which it can freely use, provided it *keeps all the initial data on the hard drive intact* at the end of



techniques (catalytic computing)

Here is an overview of some of the major arguments/techniques that appear in the catalytic literature. If you're here I'm assuming you know the setup for the model; if not then check out a survey article such as mine or Michal Koucky's, or even the original paper by Buhrman et al. (a great read!) to get oriented. For more info on specifics, I would suggest checking out the resources page for suggested papers, talks, etc.

Take a click on whatever strikes your fancy!

compress-or-random

register programs

structure in catalytic space

introduction

The only trivial upper bound on catalytic space is an equal amount of pure space. In order to improve this result, we need to look deeper into the structure of catalytic algorithms. We will borrow from two fundamental results on ordinary space: first, the straightforward fact that space s algorithms can be simulated in *time* $\exp(s)$; and second, the much more intriguing (and much more recently discovered) fact that all algorithms can be made *reversible* with only a constant amount of extra space.

open problems database to come...

ACKNOWLEDGEMENTS

- the CCC organizers for the invitation

ACKNOWLEDGEMENTS

- the CCC organizers for the invitation
- many people for feedback on the talk
  - Hongxun Wu, Ryan Williams for IP example

ACKNOWLEDGEMENTS

- the CCC organizers for the invitation
- many people for feedback on the talk
  - Hongxun Wu, Ryan Williams for IP example
- my coauthors (esp. James, Ted, Michal)

ACKNOWLEDGEMENTS

- the CCC organizers for the invitation
- many people for feedback on the talk
  - Hongxun Wu, Ryan Williams for IP example
- my coauthors (esp. James, Ted, Michal)
- the catalytic community for the exciting work

Acknowledgements

- the CCC organizers for the invitation
- many people for feedback on the talk
  - Hongxun Wu, Ryan Williams for IP example
- my coauthors (esp. James, Ted, Michal)
- the catalytic community for the exciting work

Pierre McKenzie Yaroslav Alekseev Sagar Bisoyi Richard Cleve Krishnamoorthy Dinesh Rahul Jain Vimal Raj Sharma Sathya Subramanian Aaron Potechin
Vincent Girard Aryan Agarwala Harry Buhrman James Cook Samir Datta Dean Doron Marten Folkertsma Chetan Gupta Ragunath Tewari Robert Robere
Nicollas Sdroievski Bhabya Deep Rai Jayalal Sarma Michal Koucký Bruno Loff Florian Speelman Ted Pyne Jiatu Li Roei Tell Sasha Sami Yfke Dulek
Jeroem Zuiddam Sergii Strelchuk Quinten Tupker Yuval Filmus Alexander Smal Antoine Vinciguerra William Wang Nathan Sheffield & you?

THANKS!