The Structure of Catalytic Computation

Capturing Randomness & Time via Compression

James Cook

Jiatu Li
MIT

Ian Mertz
U. of Warwick

Ted Pyne
MIT

Catalytic Computing

$$\text{SPACE}\begin{pmatrix}\text{COMPUTE } f \\ + \\ \text{REMEMBER } z\end{pmatrix}$$

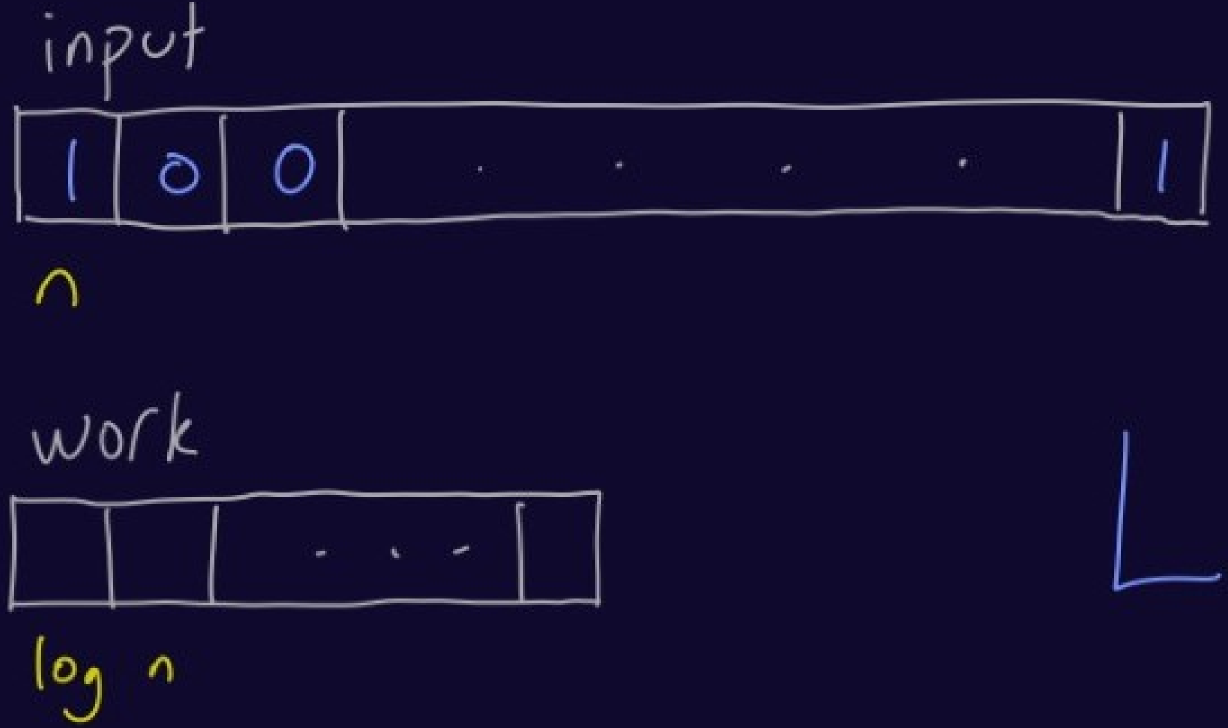
Catalytic Computing

$$\text{SPACE}\begin{pmatrix}\text{COMPUTE } f \\ + \\ \text{REMEMBER } z\end{pmatrix}$$

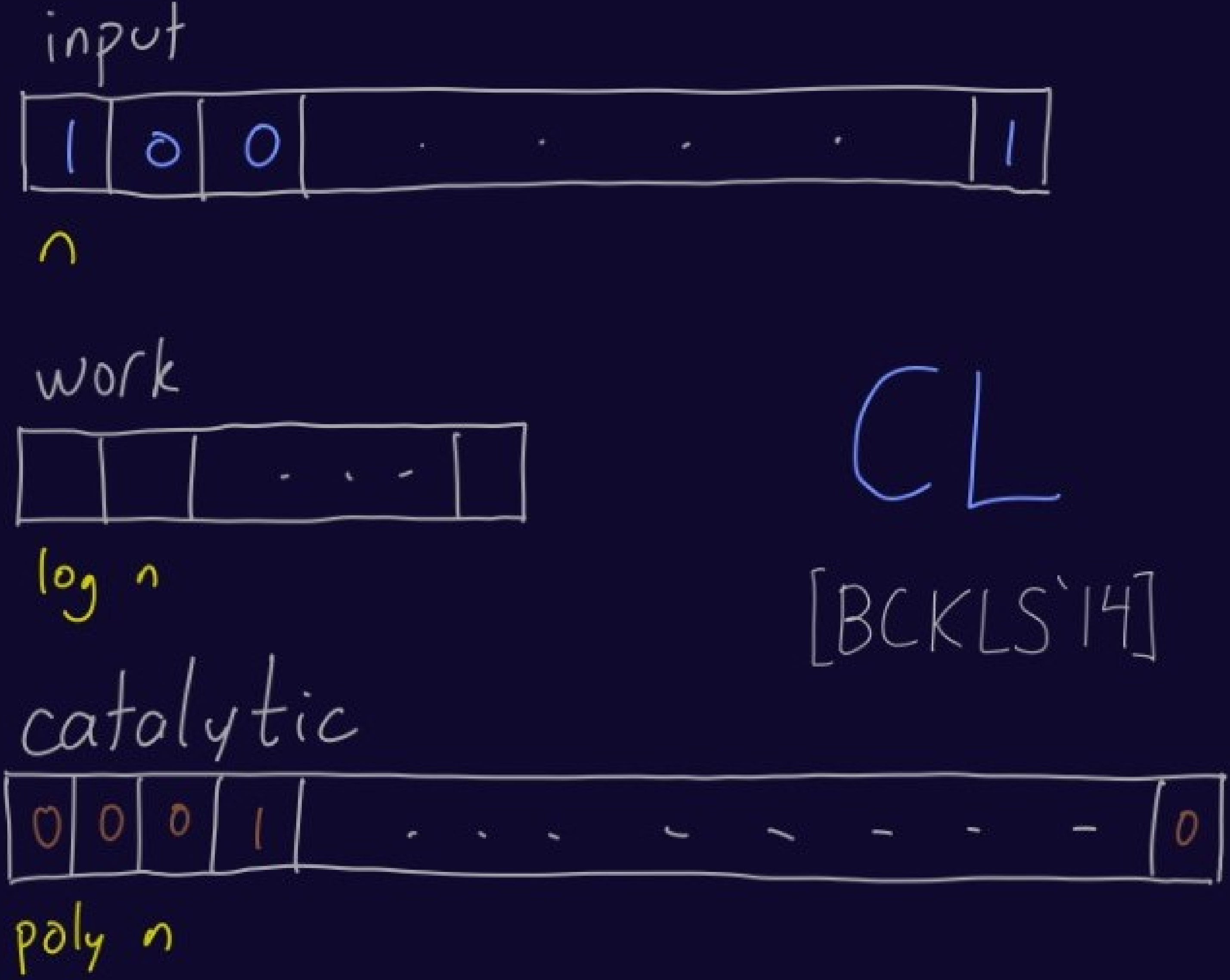
vs

$$\text{SPACE}(\text{COMPUTE } f) + \text{SPACE}(\text{REMEMBER } z)$$

CATALYTIC COMPUTING



CATALYTIC COMPUTING



CL

[BCKLS'14]

CATALYTIC COMPUTING

$$\mathrm{SPACE}\begin{pmatrix}\text{COMPUTE } f \\ + \\ \text{REMEMBER } z\end{pmatrix}$$

$$=$$

$$\mathrm{SPACE}(\text{COMPUTE } f)$$

$$+$$

$$\mathrm{SPACE}(\text{REMEMBER } z)$$

$$\approx$$

$$\mathrm{CL} = \mathrm{L}$$

CATALYTIC COMPUTING

$$\text{SPACE}\begin{pmatrix}\text{COMPUTE } f \\ + \\ \text{REMEMBER } z\end{pmatrix} = \text{SPACE}(\text{COMPUTE } f) + \text{SPACE}(\text{REMEMBER } z)$$

$$\approx \quad CL = L$$

PROBABLY FALSE!

Catalytic Computing

[BCKLS'14]:

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

CATALYTIC COMPUTING

[BCKLS'14]:

$$\begin{matrix} \mathsf{BPL} \\ \mathsf{NL} \end{matrix} \subseteq \mathsf{CL} \subseteq \mathsf{ZPP}$$

CATALYTIC COMPUTING

[BCKLS'14]:

$$\begin{matrix} \mathsf{BPL} \\ \mathsf{NL} \end{matrix} \subseteq \mathsf{CL} \subseteq \mathsf{ZPP}$$

ZP* : zero-sided error

BP* : two-sided error

Catalytic Computing

structural theory

BPCL [DGJST'20]

NCL [BKLS'18]

non-uniformity

catalytic branching programs

[GKM'15, P'17, RZ'22, CM'22, CM'24]

techniques

compress-or-random [D'15, M'23, P'23, DPT'23]

register programs [BCKLS'14, CM'20, CM'24]

applications

derandomization [DT'23, P'23, DPT'23]

TEP [CM'20, CM'24]

Catalytic Computing

Questions [M'23]:

Catalytic Computing

Questions [M'23]:

1) Is CL contained in P?

CATALYTIC COMPUTING

QUESTIONS [M'23]:

1) Is CL contained in P?

2) Which resources can further strengthen the power of CL?

CATALYTIC COMPUTING

QUESTIONS [M'23]:

1) Is CL contained in P?
   new progress!
2) Which resources can further strengthen the power of CL?

CATALYTIC COMPUTING

QUESTIONS [M'23]:

1) Is CL contained in P?

new progress!

2) Which resources can further strengthen the power of CL?

randomness does not help!

CATALYTIC COMPUTING

1) Is CL contained in P?

$$\mathrm{CLP} := \text{poly time } \mathrm{CL}$$

CATALYTIC COMPUTING

1) Is CL contained in P?

$$\mathsf{CLP} := \text{poly time } \mathsf{CL}$$

Observation:

$$\mathsf{CL} = \mathsf{CLP} \rightarrow \mathsf{CL} \subseteq \mathsf{P}$$

CATALYTIC COMPUTING

1) Is CL contained in P?

$$CLP := \text{poly time } CL$$

Observation:

converse not known!

$$CL = CLP \rightarrow CL \subseteq P$$

CATALYTIC COMPUTING

1) Is CL contained in P?

Theorem 1: $\mathsf{CLP} = \mathsf{CL} \cap \mathsf{P}$

CATALYTIC COMPUTING

2) Which resources can further strengthen the power of CL?

$$\mathrm{BPCL} := \text{randomized CL}$$

CATALYTIC COMPUTING

2) Which resources can further strengthen the power of CL?

BPCL := randomized CL

[DGJST`20]:

$$\mathrm{BPCL} \subseteq \mathrm{ZPP}$$

Catalytic Computing

2) Which resources can further strengthen the power of CL?

BPCL := randomized CL

[DGJST`20]:

BPCL = CL

assuming some derandomization hypothesis

CATALYTIC COMPUTING

2) Which resources can further strengthen the power of CL?

Theorem 2: $\mathsf{BPCL} = \mathsf{CL}$

CATALYTIC COMPUTING

2) Which resources can further strengthen the power of CL?

Theorem 2: BPCL = CL

A rare unconditional uniform derandomization result!

Catalytic Computing

$$\begin{array}{ccccccc}
\mathrm{L} & \subseteq & \mathrm{CL} & (\subseteq) & \mathrm{P} & \subseteq & \mathrm{PSPACE} \\
\rotatebox{90}{$\subseteq$} & & \rotatebox{90}{$\subseteq$} & & \rotatebox{90}{$\subseteq$} & & \rotatebox{90}{$\subseteq$} \\
\mathrm{BPL} & \subseteq & \mathrm{BPCL} & \subseteq & \mathrm{BPP} & \subseteq & \mathrm{BPPSPACE}
\end{array}$$

(all derandomizations believed)

Catalytic Computing

$$
\begin{array}{ccccccc}
L & \subseteq & CL & (\subseteq) & P & \subseteq & PSPACE \\
\rotatebox[origin=c]{-90}{$\subseteq$} & & \rotatebox[origin=c]{-90}{$\subseteq$} & & \rotatebox[origin=c]{-90}{$\subseteq$} & & \rotatebox[origin=c]{-90}{$\subseteq$} \\
BPL & \subseteq & BPCL & \subseteq & BPP & \subseteq & BPPSPACE
\end{array}
$$

equal

Catalytic Computing

$$\begin{array}{ccccccc}
L & \subseteq & CL & (\subseteq) & P & \subseteq & PSPACE \\
\rotatebox[origin=c]{-90}{$\subseteq$} & & \rotatebox[origin=c]{-90}{$\subseteq$} & & \rotatebox[origin=c]{-90}{$\subseteq$} & & \rotatebox[origin=c]{-90}{$\subseteq$} \\
BPL & \subseteq & BPCL & \subseteq & BPP & \subseteq & BPPSPACE \\
\text{open} & & & & & & \text{equal}
\end{array}$$

Catalytic Computing

$$
\begin{array}{ccccccc}
L & \subseteq & CL & (\subseteq) & P & \subseteq & PSPACE \\
\rotatebox{-90}{$\subseteq$} & & \rotatebox{-90}{$\subseteq$} & & \rotatebox{-90}{$\subseteq$} & & \rotatebox{-90}{$\subseteq$} \\
BPL & \subseteq & BPCL & \subseteq & BPP & \subseteq & BPPSPACE
\end{array}
$$

open		wide open	equal

Catalytic Computing

$$
\begin{array}{ccccccc}
\mathsf{L} & \subseteq & \mathsf{CL} & (\subseteq) & \mathsf{P} & \subseteq & \mathsf{PSPACE} \\
\rotatebox{-90}{$\subseteq$} & & \rotatebox{-90}{$\subseteq$} & & \rotatebox{-90}{$\subseteq$} & & \rotatebox{-90}{$\subseteq$} \\
\mathsf{BPL} & \subseteq & \mathsf{BPCL} & \subseteq & \mathsf{BPP} & \subseteq & \mathsf{BPPSPACE} \\
\text{open} & & \text{equal} & & \text{wide} & & \text{equal} \\
 & & & & \text{open} & &
\end{array}
$$

Catalytic Computing

$\mathsf{BPCL} := \text{randomized } \mathsf{CL}$

Catalytic Computing

BPCL := randomized CL

Abstract

In the catalytic logspace (**CL**) model of (Buhrman et. al. STOC 2013), we are given a small work tape, and a larger catalytic tape that has an arbitrary initial configuration. We may edit this tape, but it must be exactly restored to its initial configuration at the completion of the computation. This model is of interest from a complexity-theoretic perspective as it gains surprising power over traditional space. However, many fundamental structural questions remain open.

We substantially advance the understanding of the structure of **CL**, addressing several questions raised in prior work. Our main results are as follows:

1. We unconditionally derandomize catalytic logspace: **CBPL** = **CL**.
2. We show time and catalytic space bounds can be achieved separately if and only if they can be achieved simultaneously: any problem in **CL** ∩ **P** can be solved in polynomial time-bounded **CL**.
3. We characterize deterministic catalytic space by the intersection of randomness and time: **CL** is equivalent to polytime-bounded, *zero-error* randomized **CL**.

Our results center around the *compress–or–random* framework. For the second result, we introduce a simple yet novel *compress–or–compute* algorithm which, for any catalytic tape, either compresses the tape or quickly and successfully computes the function at hand. For our first result, we further introduce a *compress–or–compress–or–random* algorithm that combines runtime compression with a second compress–or–random algorithm, building on recent work on distinguish-to-predict transformations and pseudorandom generators with small-space deterministic reconstruction.

CATALYTIC COMPUTING

BPCL := randomized CL

[6] While all published works on the subject of randomized catalytic space [DGJ+20, Mer23, Pyn23, DPT24] put C before e.g. BP in **CBPSPACE** $[s]$, they first appear in an older, yet unpublished, work by Dulek, which reverses the order.

CATALYTIC COMPUTING

BPCL := randomized CL

[6]While all published works on the subject of randomized catalytic space [DGJ+20, Mer23, Pyn23, DPT24] put C before e.g. BP in **CBPSPACE** $[s]$, they first appear in an older, yet unpublished, work by Dulek, which reverses the order. Theorem 1.1, thankfully, all but obviates the need to solve this nomenclature issue.

CATALYTIC COMPUTING

Theorem 1: $\mathrm{CLP} = \mathrm{CL} \cap \mathrm{P}$

Theorem 2: $\mathrm{BPCL} = \mathrm{CL}$

CATALYTIC COMPUTING

Theorem 1: $CLP = CL \cap P$

Theorem 2: $BPCL = CL$

Theorem 3: $ZPCLP = CL$

CATALYTIC COMPUTING

Theorem 1: $\mathsf{CLP} = \mathsf{CL} \cap \mathsf{P}$

Theorem 2: $\mathsf{BPCL} = \mathsf{CL}$

Theorem 3: $\mathsf{ZPCLP} = \mathsf{CL}$

randomness

time

Catalytic Computing

Theorem 3: $\mathsf{ZPCLP} = \mathsf{CL}$

CATALYTIC COMPUTING

Theorem 3: $ZPCLP = CL$

Observation: $CL = CLP \rightarrow CL \subseteq P$

CATALYTIC COMPUTING

Theorem 3: $\mathsf{ZPCLP} = \mathsf{CL}$

Corollary: $\mathsf{ZPCL} = \mathsf{ZPCLP}$

Observation: $\mathsf{CL} = \mathsf{CLP} \rightarrow \mathsf{CL} \subseteq \mathsf{P}$

CATALYTIC COMPUTING

Theorem 3: $\mathsf{ZPCLP} = \mathsf{CL}$

Observation: $\mathsf{ZPP} = \mathsf{P} \rightarrow \mathsf{CL} \subseteq \mathsf{P}$

CATALYTIC COMPUTING

Theorem 3: $\mathsf{ZPCLP} = \mathsf{CL}$

Corollary: $\mathsf{ZPP} = \mathsf{P} \rightarrow \mathsf{ZPCLP} = \mathsf{CLP}$

Observation: $\mathsf{ZPP} = \mathsf{P} \rightarrow \mathsf{CL} \subseteq \mathsf{P}$

CATALYTIC COMPUTING

Theorem 1: $\mathsf{CLP} = \mathsf{CL} \cap \mathsf{P}$

Theorem 2: $\mathsf{BPCL} = \mathsf{CL}$

Catalytic Computing

Theorem 1: $\mathsf{CLP} = \mathsf{CL} \cap \mathsf{P}$

Theorem 2: $\mathsf{BPCL} = \mathsf{CL}$

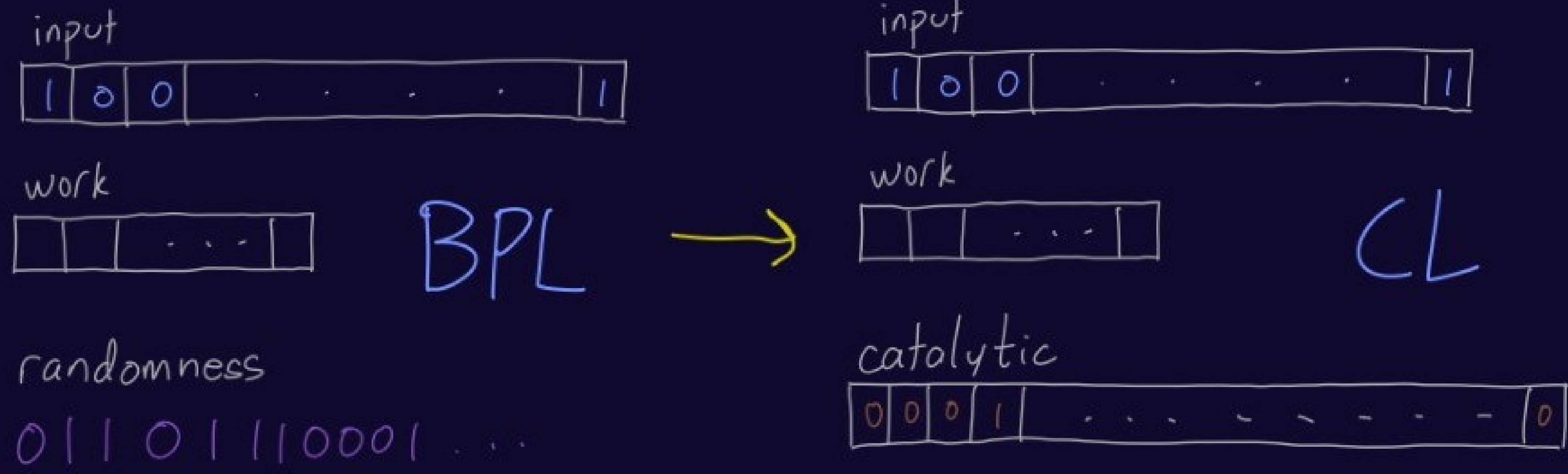
Proof: a completely new twist on an old technique

Compress-or-Random

[D'15, BCKLS'14]: $\mathsf{BPL} \subseteq \mathsf{CL}$

Compress-or-Random

[D'15, BCKLS'14]: $\mathrm{BPL} \subseteq \mathrm{CL}$



COMPRESS-OR-RANDOM

[D'15, BCKLS'14]: $\mathsf{BPL} \subseteq \mathsf{CL}$

If catalytic tape is random enough for L, then done!

Compress-or-Random

[D'15, BCKLS'14]: $\mathsf{BPL} \subseteq \mathsf{CL}$

If catalytic tape is random enough for L, then done!

[N'94]: sufficient (and efficient) test

Compress-or-Random

[D'15, BCKLS'14]: $\mathsf{BPL} \subseteq \mathsf{CL}$

If catalytic tape is not random enough for L . . .

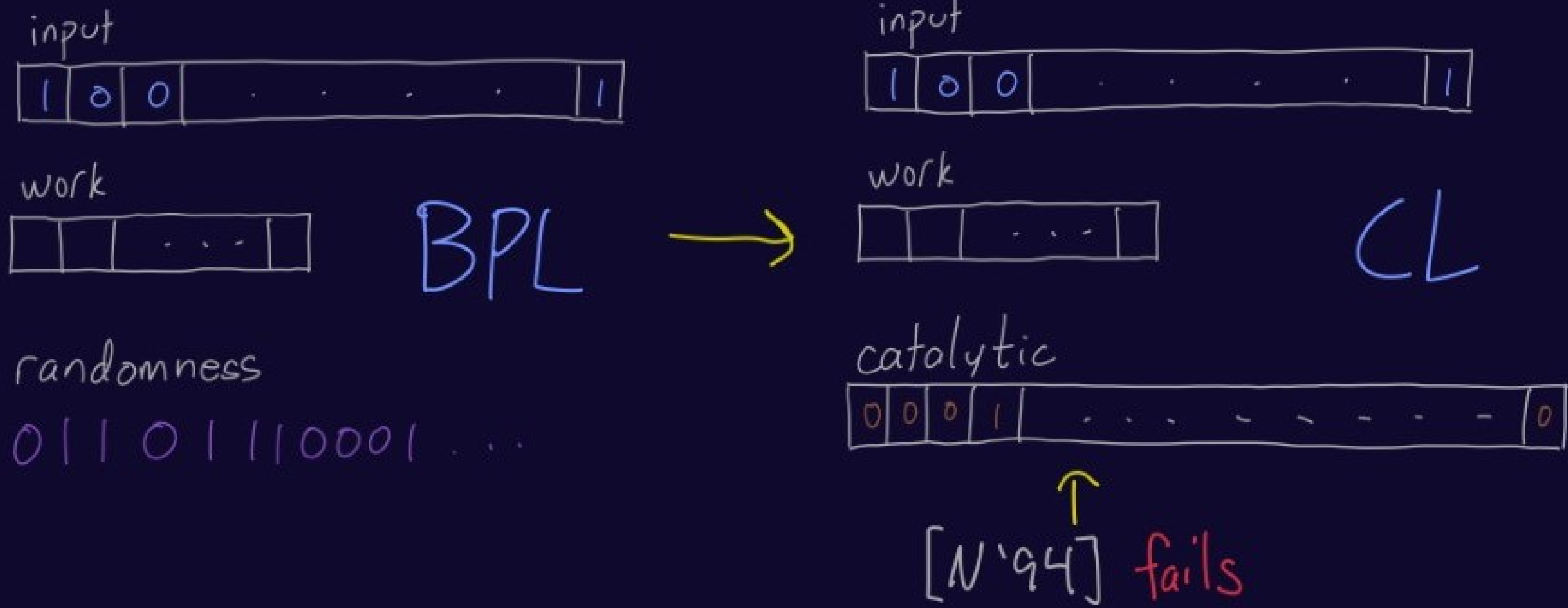
COMPRESS-OR-RANDOM

[D'15, BCKLS'14]: $\mathsf{BPL} \subseteq \mathsf{CL}$

If catalytic tape is not random enough for L, compress it!

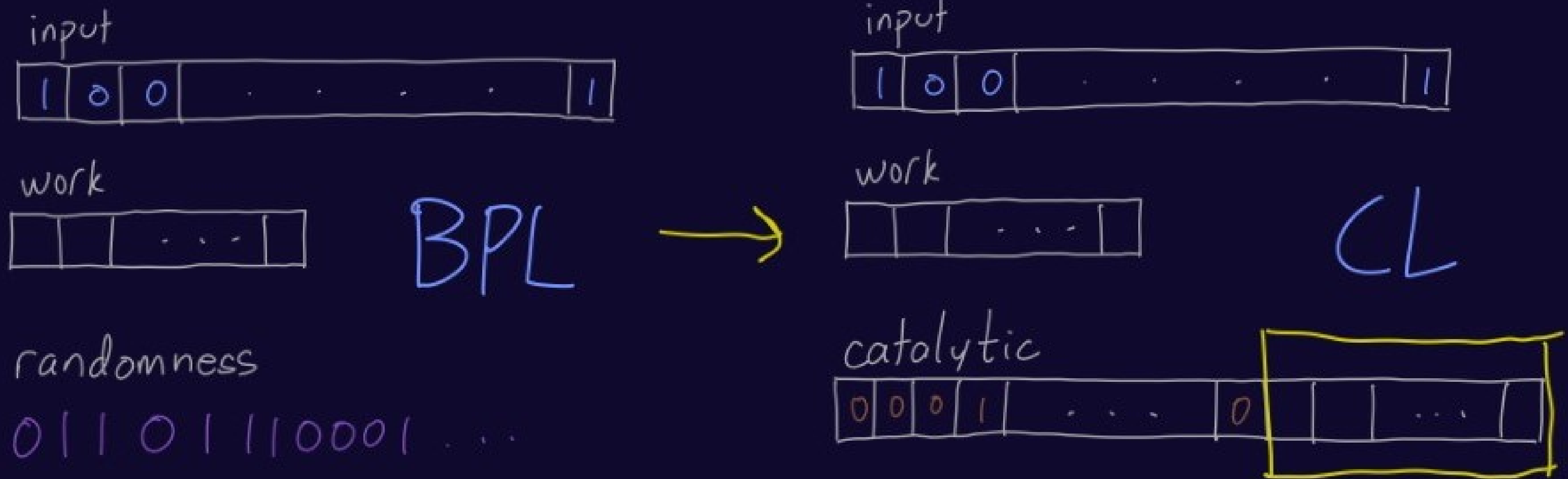
Compress-or-Random

[D'15, BCKLS'14]: $\mathsf{BPL} \subseteq \mathsf{CL}$



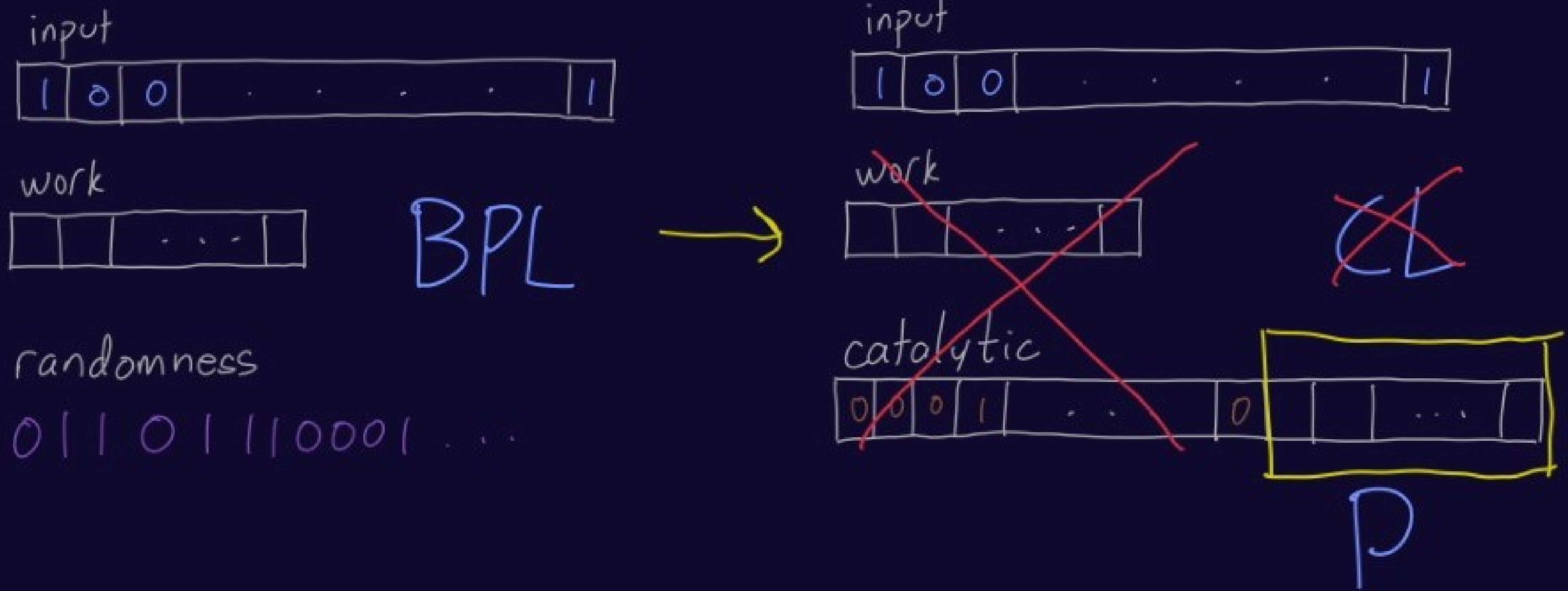
Compress-or-Random

[D'15, BCKLS'14]: $\mathsf{BPL} \subseteq \mathsf{CL}$



Compress-or-Random

[D'15, BCKLS'14]: $BPL \subseteq CL$



COMPRESS-OR-RANDOM

[D'15, BCKLS'14]: $\mathsf{BPL} \subseteq \mathsf{CL}$

Algorithm:

- run Nisan's test on the catalytic tape
- if (pass): random (use to simulate BPL)
- if (fail): compress (use to brute force, then decompress)

COMPRESS-OR-RANDOM

$CL \cap P \subseteq CLP$? $\quad BPCL \subseteq CL$?

Compress-or-Random

$CL \cap P \subseteq CLP$? $BPCL \subseteq CL$?

- randomness doesn't help us with $CL \cap P$

- compress case takes too long for CLP

Compress-or-Random

$CL \cap P \subseteq CLP$?

- randomness doesn't help us with $CL \cap P$

- compress case takes too long for CLP

$BPCL \subseteq CL$?

- Nisan's test doesn't work against $BPCL$

- configuration graph can become very large

COMPRESS-OR-RANDOM

$CL \cap P \subseteq CLP$?

- randomness doesn't help us with $CL \cap P$
- ~~compress case takes too long for CLP~~

use P algorithm

$BPCL \subseteq CL$?

- Nisan's test doesn't work against $BPCL$
- configuration graph can become very large

COMPRESS-OR-RANDOM

$CL \cap P \subseteq CLP$?

- randomness doesn't help us with $CL \cap P$
- ~~compress case takes too long for CLP~~

use P algorithm

$BPCL \subseteq CL$?

more sophisticated tools

- ~~Nisan's test doesn't work against $BPCL$~~
- configuration graph can become very large

COMPRESS-OR-RANDOM

KEY IDEA:

a compression argument which fails

exactly when

we can efficiently simulate
a catalytic computation

Compress-or-Random

$CL \cap P \subseteq CLP$?

- randomness doesn't help us with $CL \cap P$
- ~~compress case takes too long for CLP~~

use P algorithm

$BPCL \subseteq CL$?

more sophisticated tools

- ~~Nisan's test doesn't work against $BPCL$~~
- configuration graph can become very large

Compress-or-Random

$CL \cap P \subseteq CLP$?

"random" → CLP simulation

- ~~randomness doesn't help us with~~ ~~$CL \cap P$~~
- ~~compress case takes too long for CLP~~

use P algorithm

$BPCL \subseteq CL$?

more sophisticated tools

- ~~Nisan's test doesn't work against BPCL~~
- configuration graph can become very large

Compress-or-Random

$CL \cap P \subseteq CLP$?

"random" → CLP simulation

- ~~randomness doesn't help us with~~ ~~$CL \cap P$~~
- ~~compress case takes too long for~~ ~~CLP~~

use P algorithm

$BPCL \subseteq CL$?

more sophisticated tools

- ~~Nisan's test doesn't work against~~ ~~BPCL~~
- ~~configuration graph can become very large~~

"random" → small graph case

Compress-or-Random

$\mathsf{CL} \cap \mathsf{P} \subseteq \mathsf{CLP}$ $\mathsf{BPCL} \subseteq \mathsf{CL}$

COMPRESS-OR-RANDOM

$\mathsf{CL} \cap \mathsf{P} \subseteq \mathsf{CLP}$ $\mathsf{BPCL} \subseteq \mathsf{CL}$

compress-or-compute

compression succeeds:
run P algorithm

compression fails:
simulate CL machine
in poly time

Compress-or-Random

$CL \cap P \subseteq CLP$

$BPCL \subseteq CL$

compress-or-
compress-or-random

compression succeeds:
solve w/ brute force

compression fails:
run "standard" CoR on
small configuration graph

Compress-or-Random

$\mathsf{CL} \cap \mathsf{P} \subseteq \mathsf{CLP}$

compress-or-compute

compression succeeds:

run P algorithm

compression fails:

simulate CL machine in poly time

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

compress-or-compress-or-random

compression succeeds:

solve w/ brute force

compression fails:

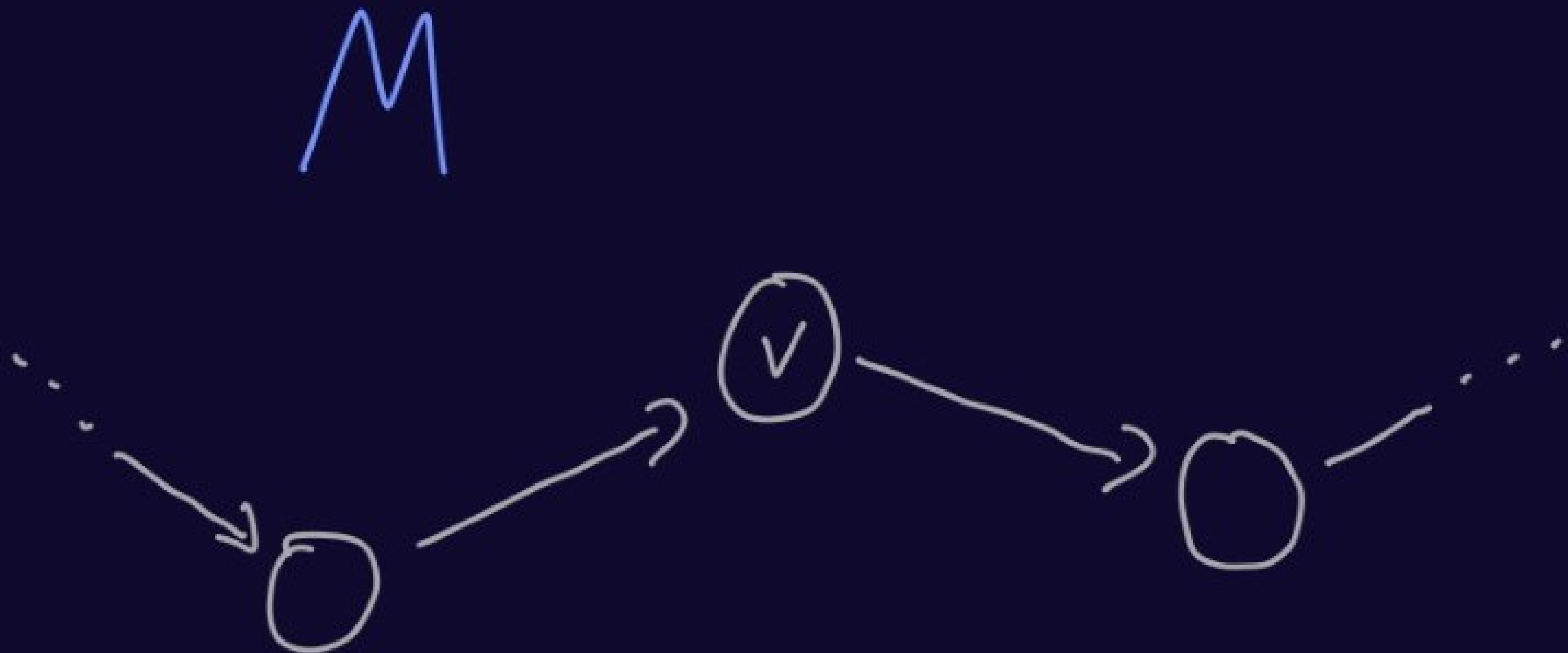
run "standard" CoR on small configuration graph

Aside: Reversibility

[LMT'00]: space is reversible

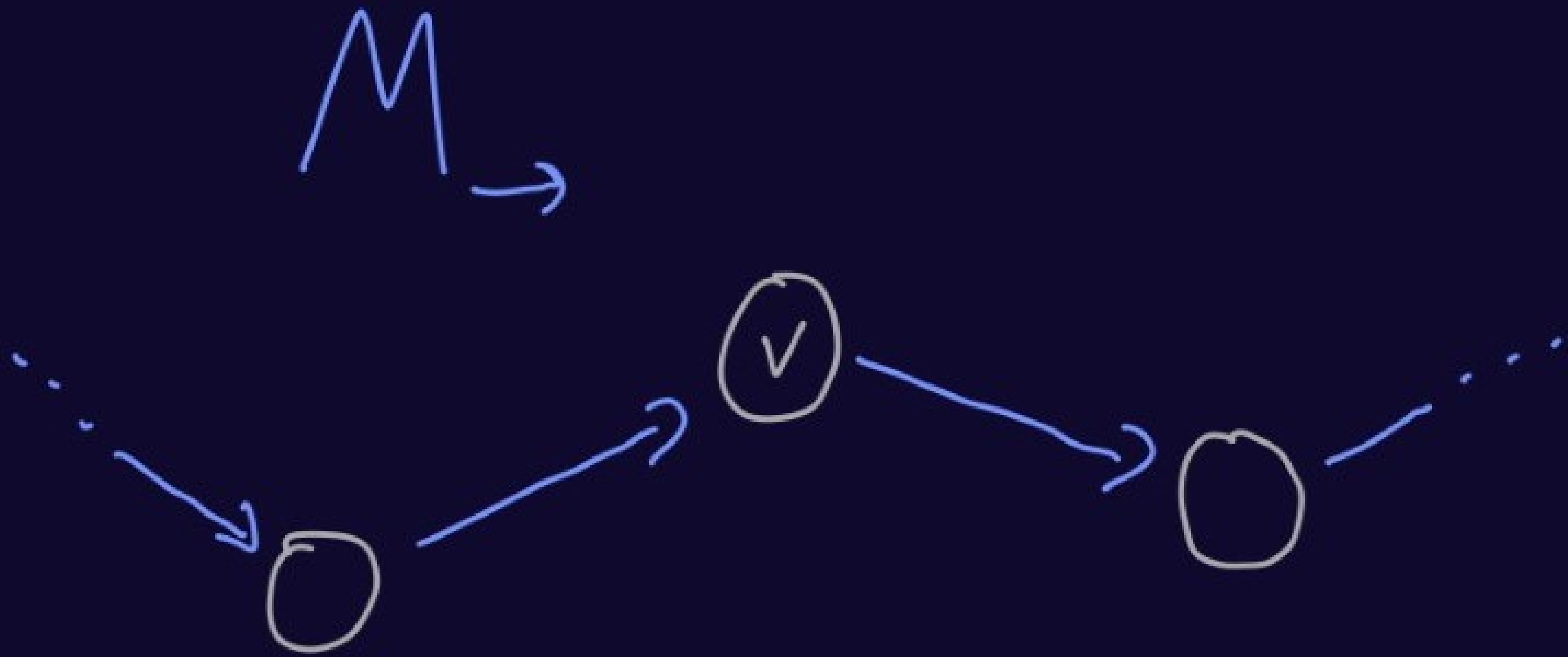
Aside: Reversibility

[LMT'00]: space is reversible



Aside: Reversibility

[LMT'00]: space is reversible



Aside: Reversibility

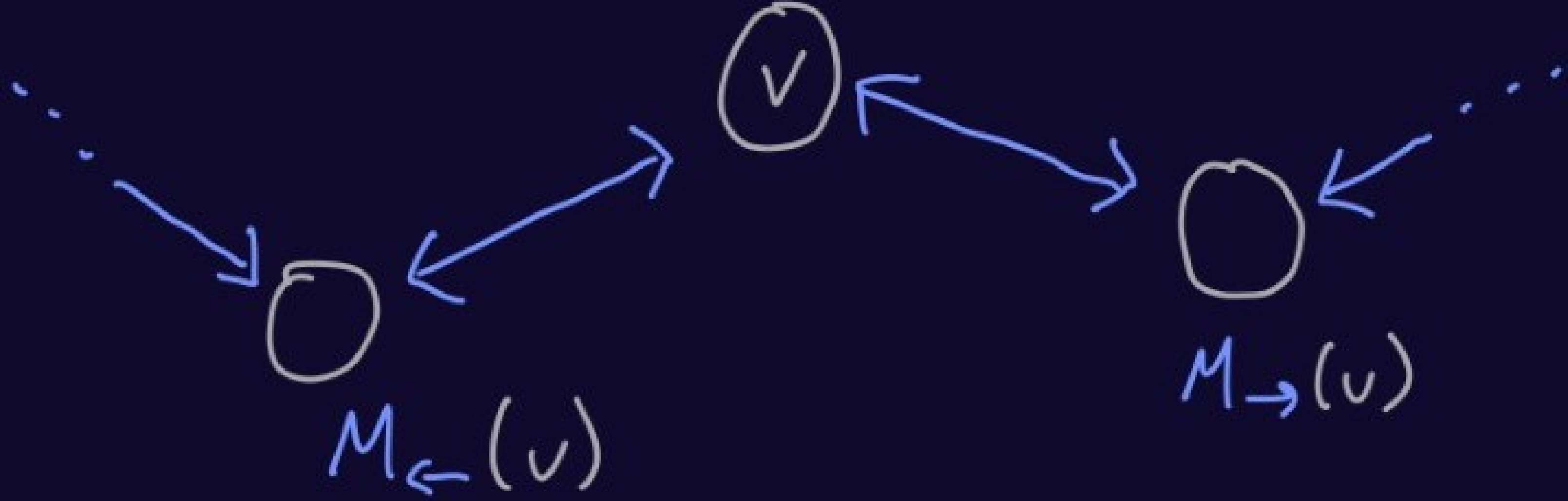
[LMT'00]: space is reversible

$M_{\leftarrow}$

Aside: Reversibility

[LMT'00]: space is reversible

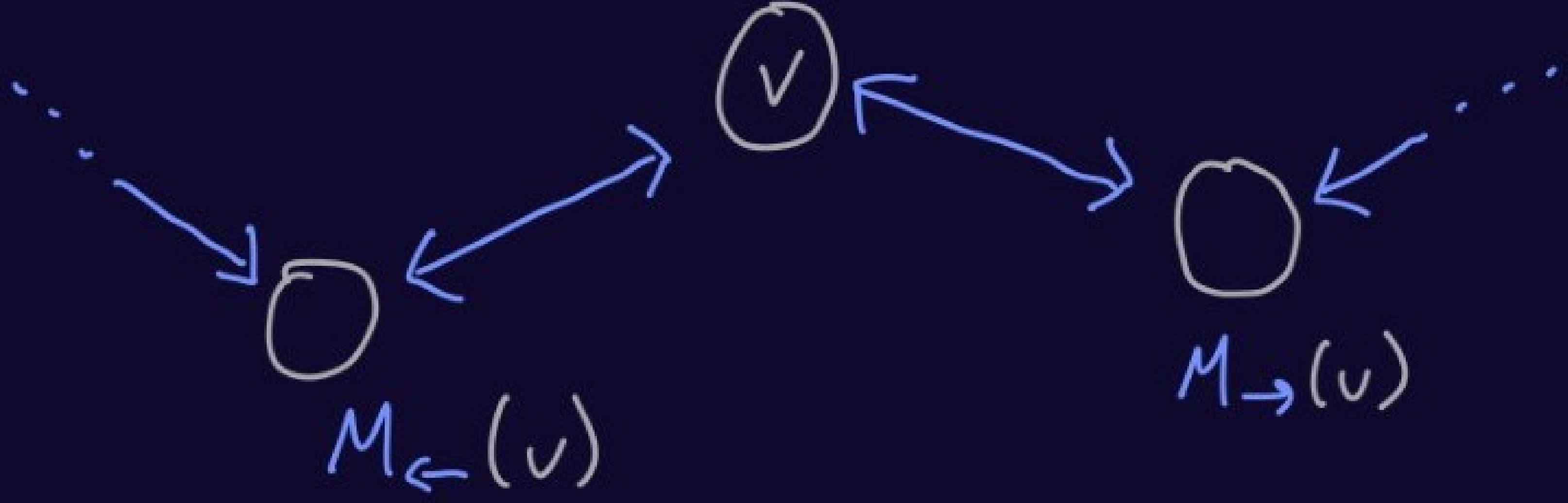
$$M_{\rightarrow}(M_{\leftarrow}(v)) = v = M_{\leftarrow}(M_{\rightarrow}(v))$$



Aside: Reversibility

[D'15]: catalytic space is reversible

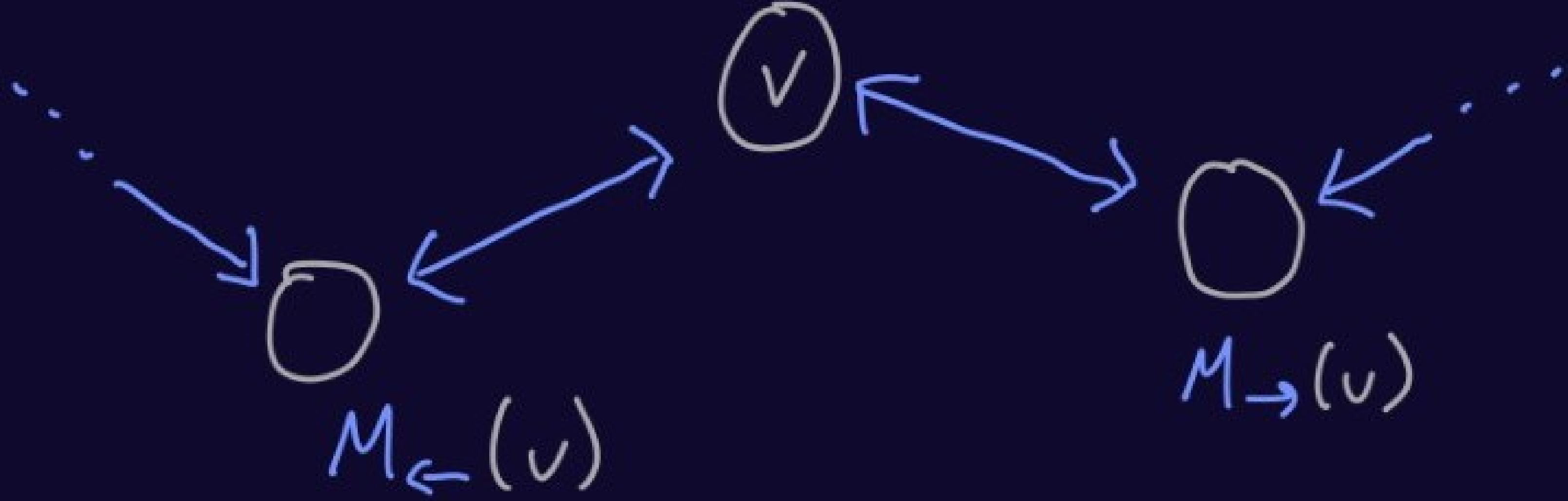
$$M_{\rightarrow}(M_{\leftarrow}(v)) = v = M_{\leftarrow}(M_{\rightarrow}(v))$$



Aside: Reversibility

[CLMP]: randomized catalytic space is reversible**

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



ASIDE: REVERSIBILITY

[CLMP]: randomized catalytic space is reversible**

[D'15]: catalytic space is reversible**

Aside: Reversibility

[CLMP]: randomized catalytic space is reversible**

[D'15]: catalytic space is reversible**

**: for any state (τ', v) reached by $M_{\rightarrow}$ on start state $(\tau, 0)$,

Aside: Reversibility

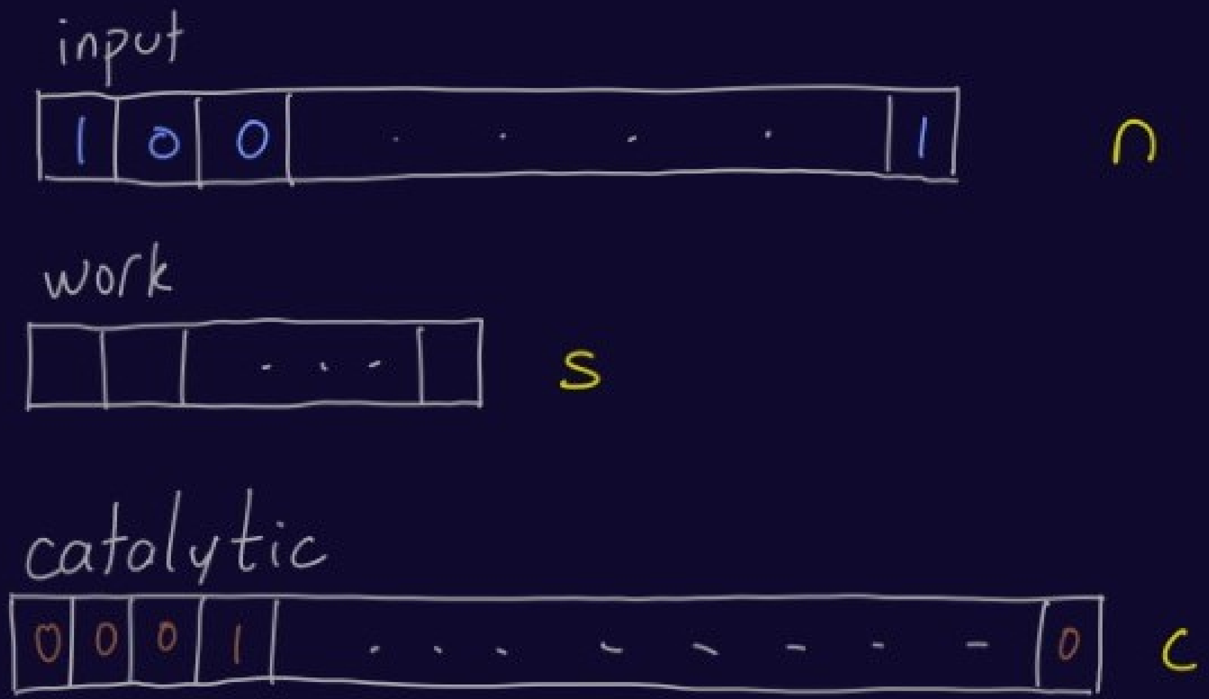
[CLMP]: randomized catalytic space is reversible**

[D'15]: catalytic space is reversible**

**: for any state (τ', v) reached by $M_{\rightarrow}$ on start state $(\tau, 0)$, $M_{\leftarrow}$ on (τ', v) can only ever reach start state $(\tau, 0)$

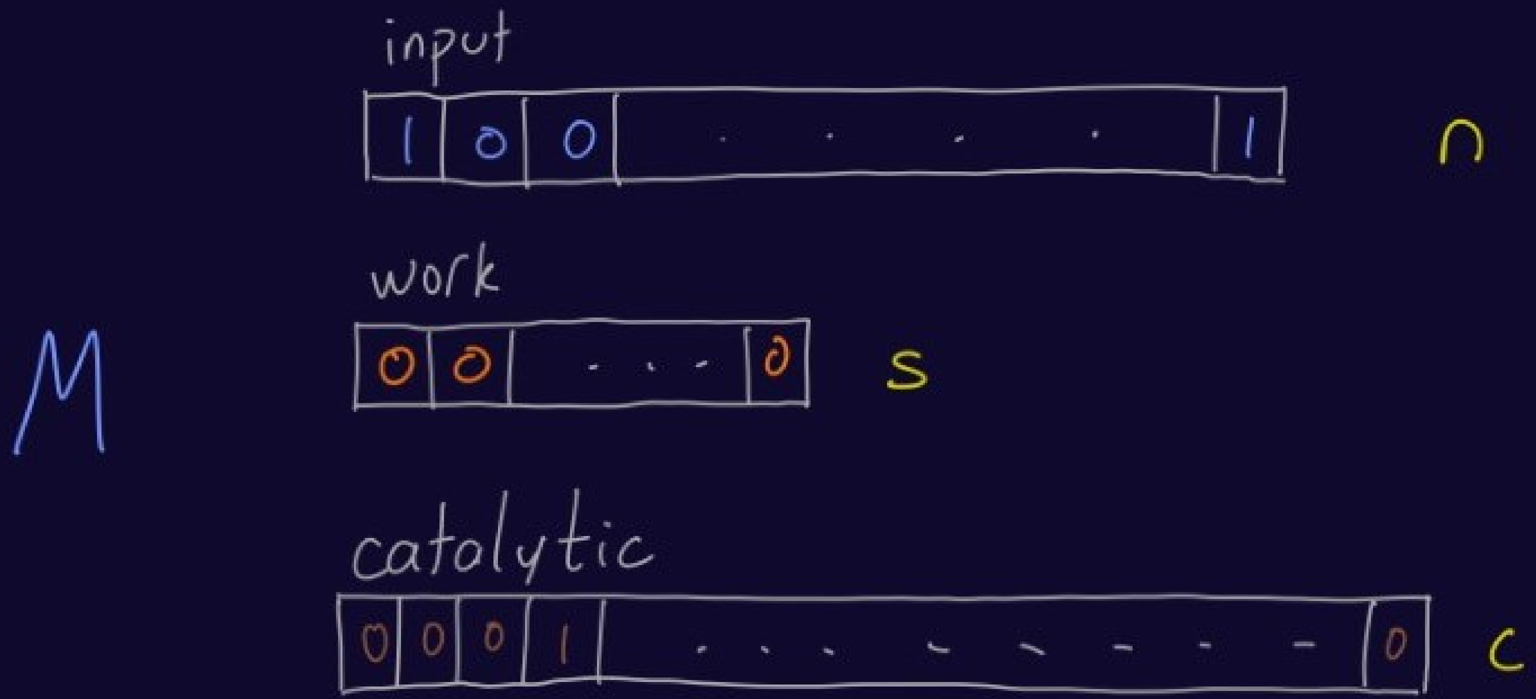
Timestep Compression

CL machine M, starting tape $\tau \in \{0,1\}^c$



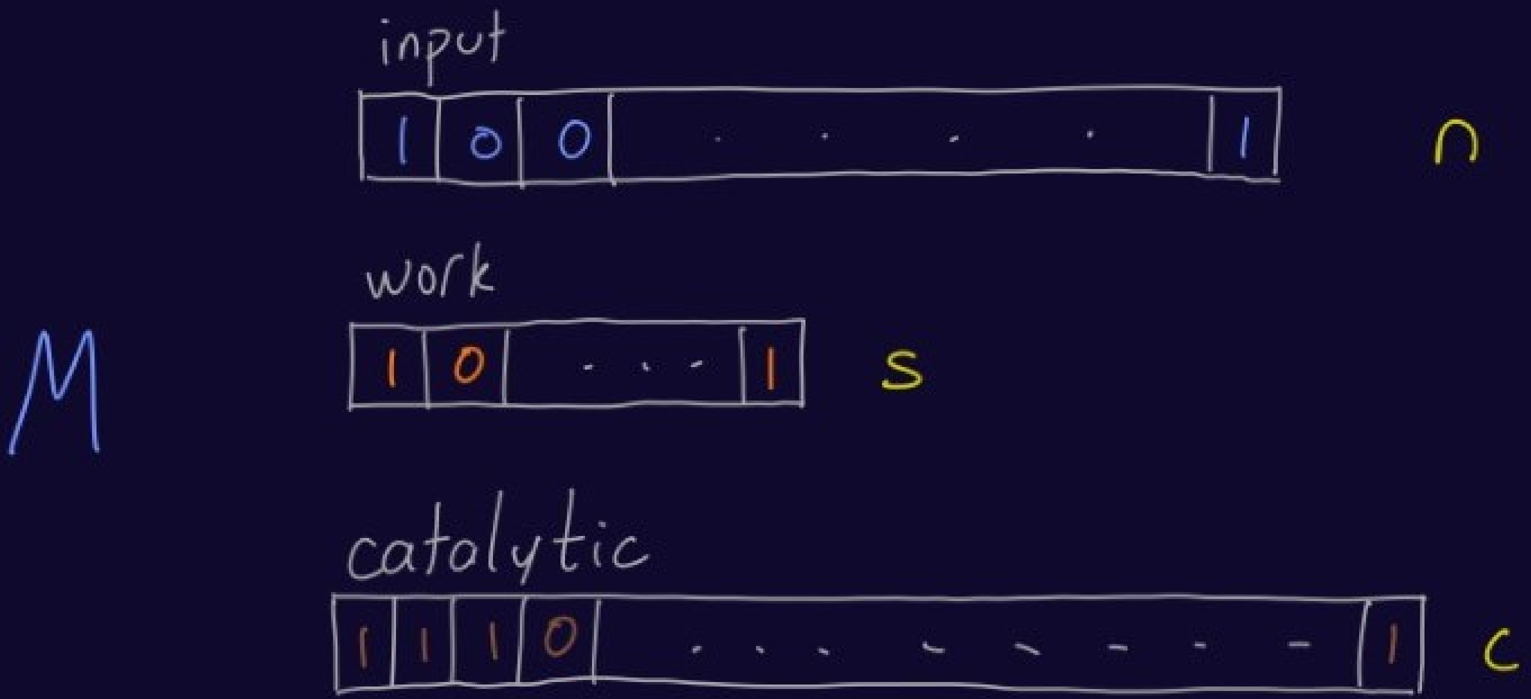
Timestep Compression

$(\tau, 0)$



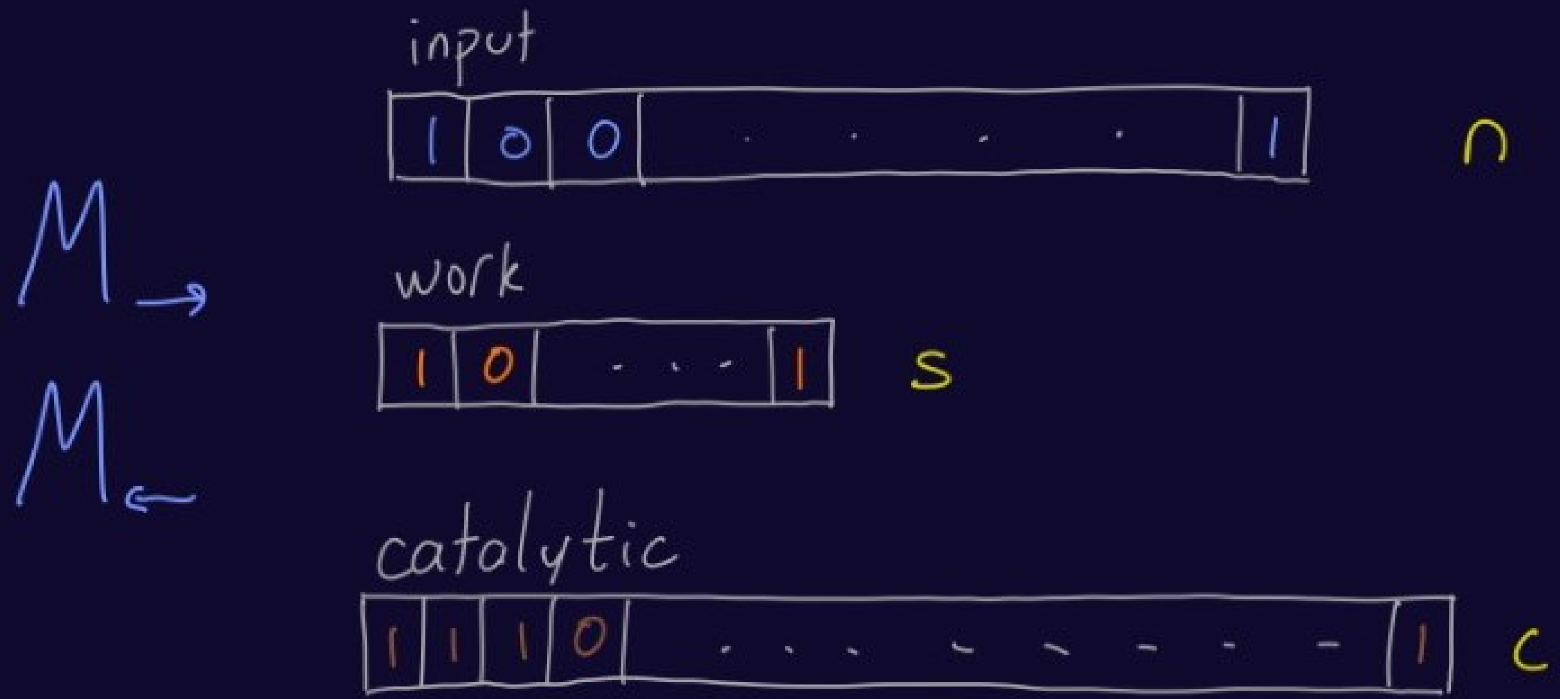
Timestep Compression

$$(\tau, 0) \xrightarrow{i \text{ steps}} (\tau_i, v_i)$$



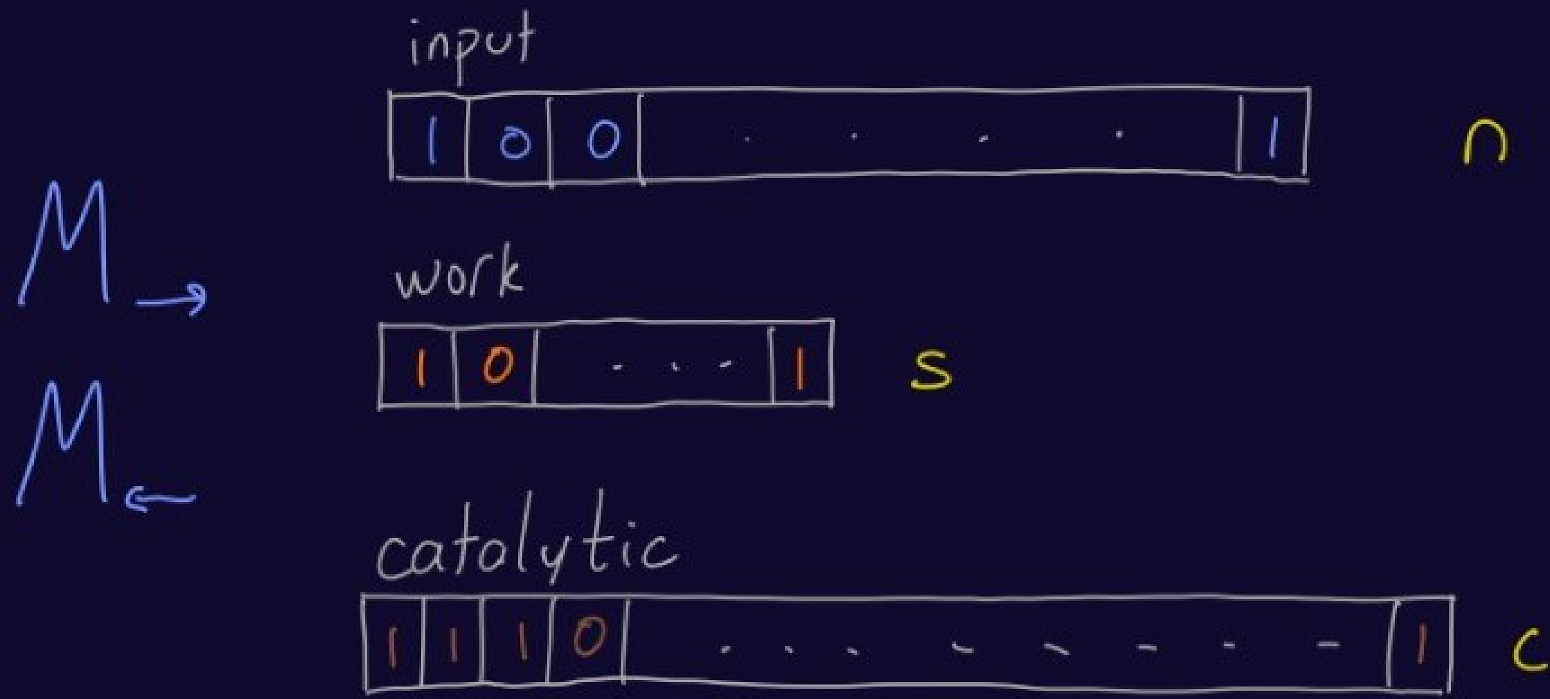
Timestep Compression

$$(\tau, 0) \xleftrightarrow{i \text{ steps}} (\tau_i, v_i)$$



Timestep Compression

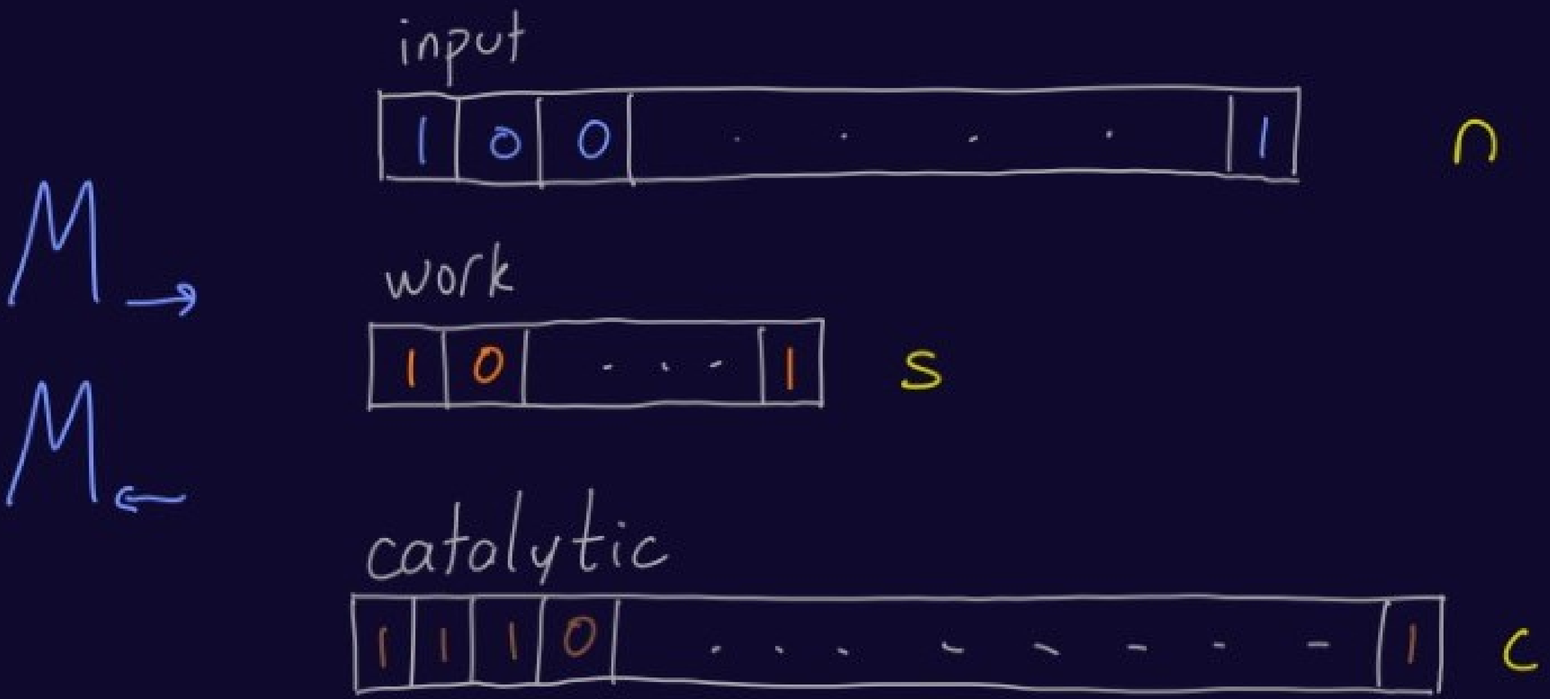
$$\langle \overset{c}{\tau}, i \rangle \underset{M_\leftarrow}{\overset{M_\to}{\rightleftarrows}} \langle \overset{c}{\tau_i}, \overset{s}{V_i} \rangle$$



Timestep Compression

compression:
$M_{\rightarrow}(\tau, 0)$ for i steps

$$\langle \overset{c}{\tau, i} \rangle \underset{M_{\leftarrow}}{\overset{M_{\rightarrow}}{\rightleftarrows}} \langle \overset{c}{\tau_i}, \overset{s}{V_i} \rangle$$



Timestep Compression

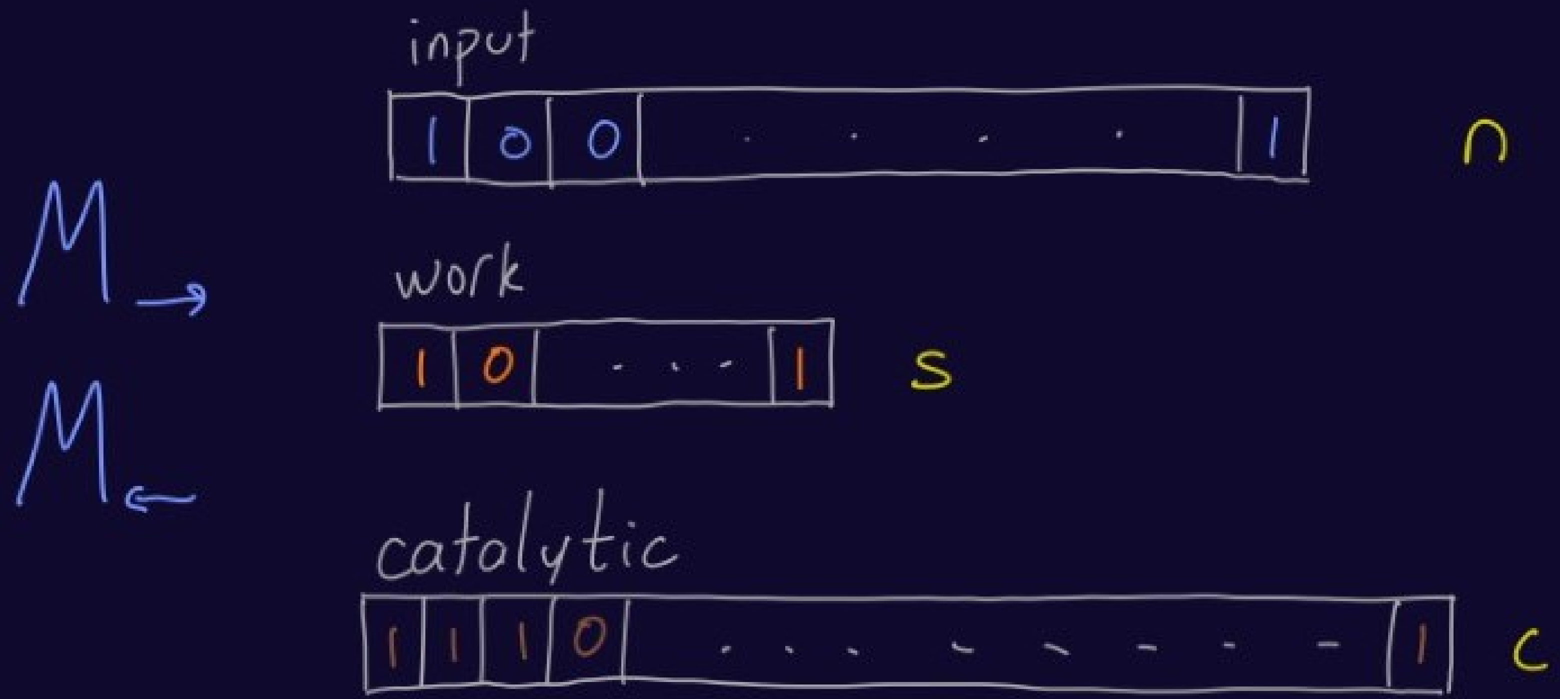
$$\langle \overset{c}{\tau}, i \rangle \underset{M_\leftarrow}{\overset{M_\rightarrow}{\rightleftarrows}} \langle \overset{c}{\tau_i}, \overset{s}{v_i} \rangle$$

compression:

$M_\rightarrow(\tau, 0)$ for i steps

decompression:

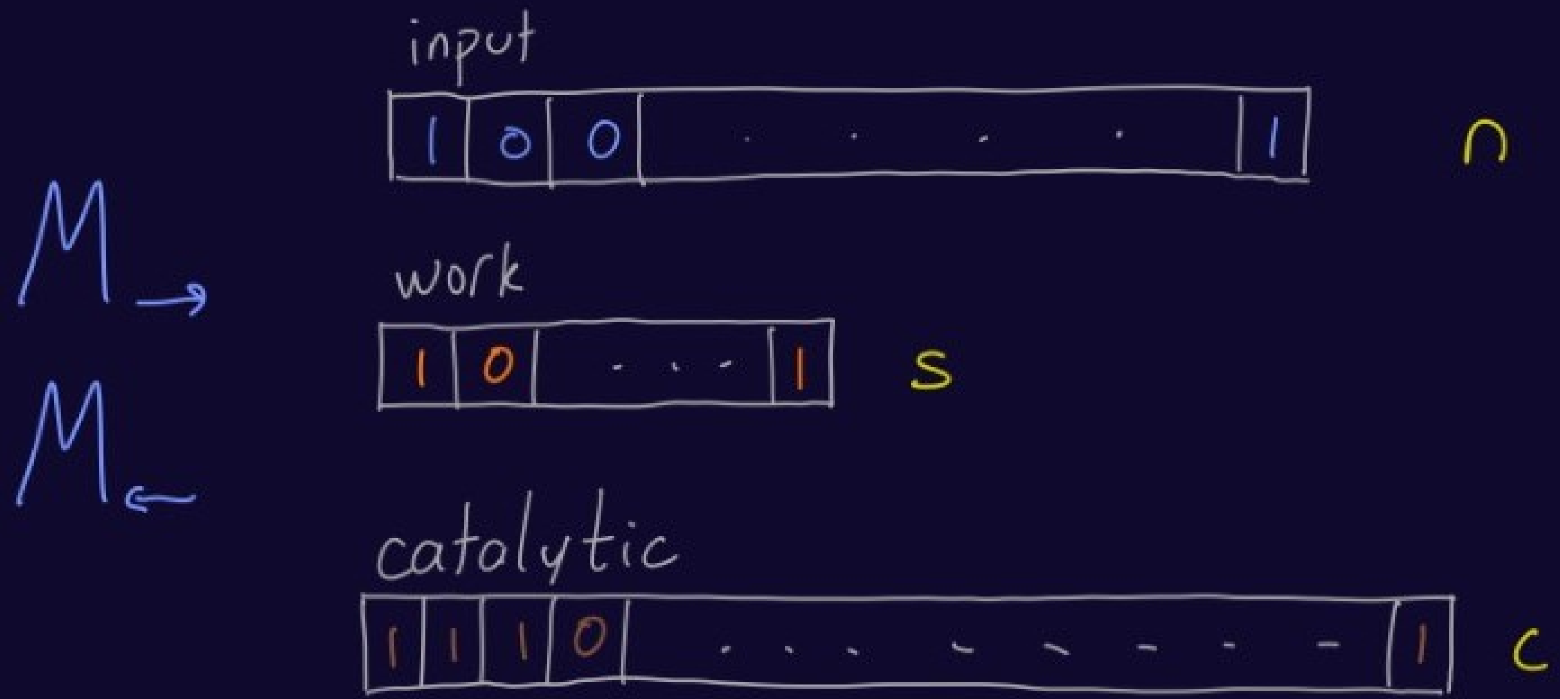
$M_\leftarrow(\tau_i, v_i)$ until we hit start state, count # steps



Timestep Compression

$$\langle \overset{c}{\tau}, i \rangle \underset{M_{\leftarrow}}{\overset{M_{\rightarrow}}{\rightleftarrows}} \langle \overset{c}{\tau_i}, \overset{s}{V_i} \rangle$$

$i \in \{0,1\}^{s+1} \rightarrow$ compression!



Timestep Compression

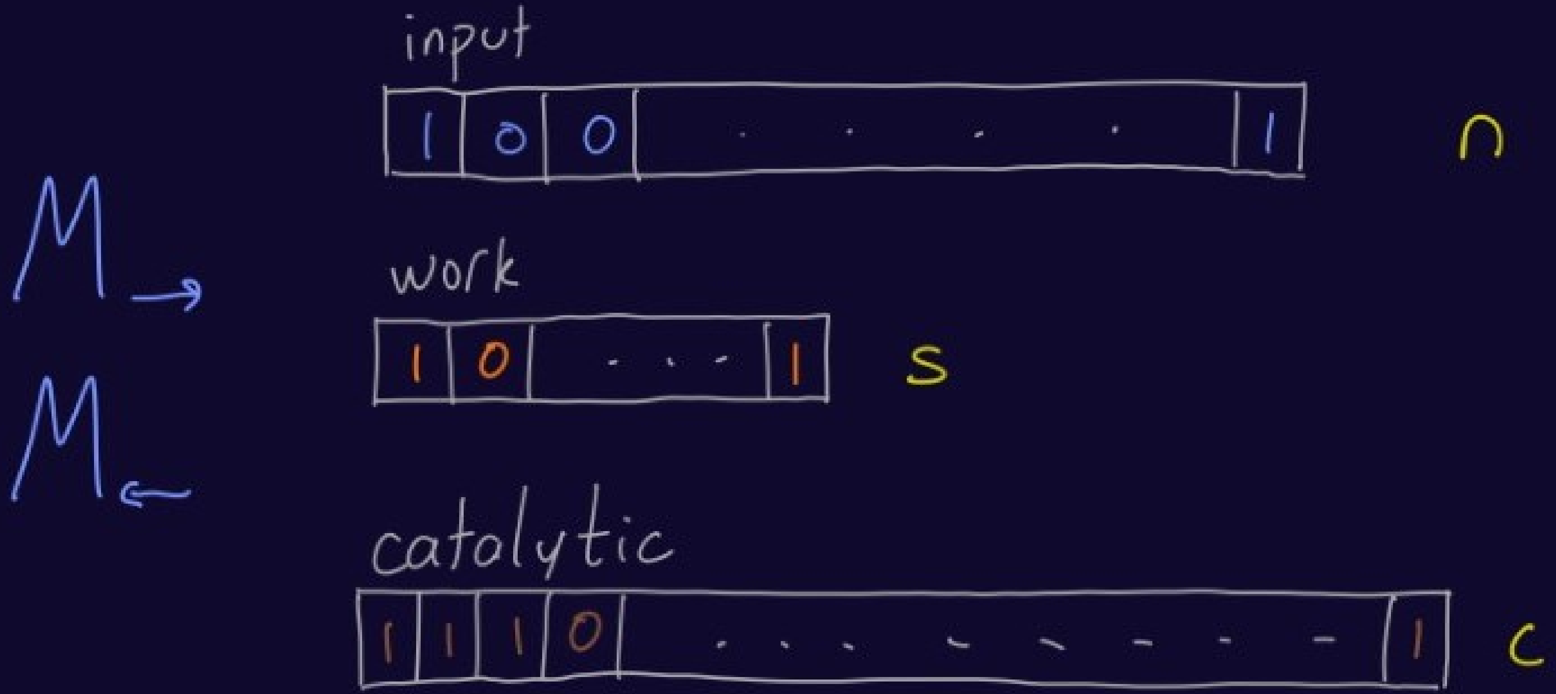
$$\langle \overset{c}{\tau}, i \rangle \underset{M_{\leftarrow}}{\overset{M_{\rightarrow}}{\rightleftarrows}} \langle \overset{c}{\tau_i}, \overset{s}{V_i} \rangle$$

$i \in \{0,1\}^{s+1} \longrightarrow$

compression!

unless M takes

$< i$ steps



Timestep Compression

$$\langle \overset{c}{\tau}, i \rangle \underset{M_\leftarrow}{\overset{M_\rightarrow}{\rightleftarrows}} \langle \overset{c}{\tau_i}, \overset{s}{V_i} \rangle$$

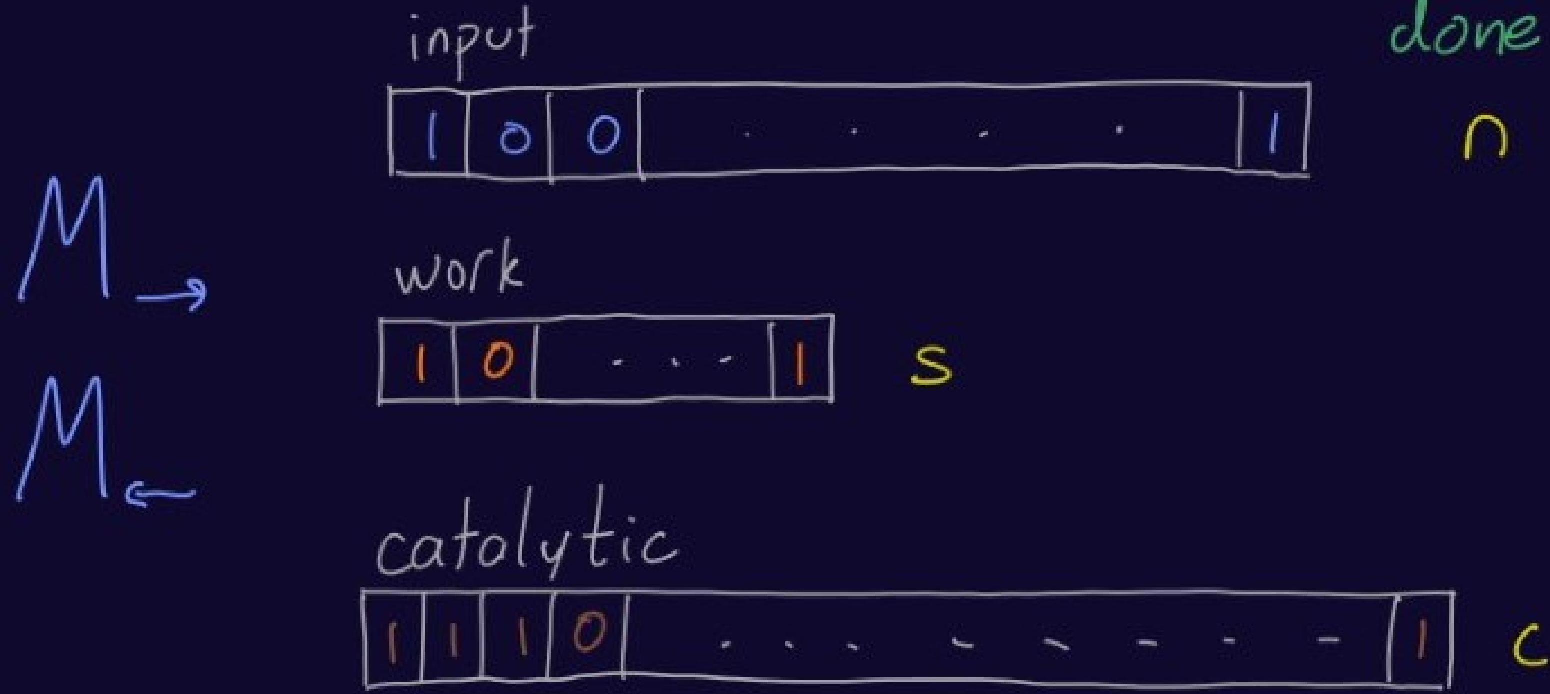
$i \in \{0,1\}^{s+1} \rightarrow$

compression!

unless M takes

$< i$ steps $\rightarrow$

done in time 2^{s+1}



TIMESTEP COMPRESSION

Theorem 1: $CLP = CL \cap P$

Theorem 2: $BPCL = CL$

TIMESTEP COMPRESSION

$CL \cap P \subseteq CLP$: compress-or-compute

Timestep Compression

$CL \cap P \subseteq CLP$: compress-or-compute

M_{CLP} (assume $\exists M_{CL}, M_P$ computing f)

TIMESTEP COMPRESSION

$CL \cap P \subseteq CLP$: compress-or-compute

M_{CLP} (assume $\exists M_{CL}, M_P$ computing f)

- repeat $time(M_P)$ times:

TIMESTEP COMPRESSION

$\mathsf{CL} \cap \mathsf{P} \subseteq \mathsf{CLP}$: compress-or-compute

M_{CLP} (assume $\exists M_{CL}, M_P$ computing f)

- repeat $time(M_P)$ times:
  - $(\tau, i) :=$ first $c + (s+1)$ bits of cat. tape

Timestep Compression

$\mathsf{CL} \cap \mathsf{P} \subseteq \mathsf{CLP}$: compress-or-compute

M_{CLP} (assume $\exists M_{CL}, M_P$ computing f)

- repeat $time(M_P)$ times:
  - $(\tau, i) :=$ first $c + (s+1)$ bits of cat. tape
  - run M_{CL} on $(\tau, 0)$ for i steps

TIMESTEP COMPRESSION

$CL \cap P \subseteq CLP$: compress-or-compute

M_{CLP} (assume $\exists M_{CL}, M_P$ computing f)

- repeat $time(M_P)$ times:
  - $(\tau, i) :=$ first $c + (s+1)$ bits of cat. tape
  - run M_{CL} on $(\tau, 0)$ for i steps
    - if halts, save answer and revert

TIMESTEP COMPRESSION

$CL \cap P \subseteq CLP$: compress-or-compute

M_{CLP} (assume $\exists M_{CL}, M_P$ computing f)

- repeat $time(M_P)$ times:
  - $(\tau, i) :=$ first $c+(s+1)$ bits of cat. tape
  - run M_{CL} on $(\tau, 0)$ for i steps
    - if halts, save answer and revert
    - else, replace (τ, i) with $(\tau_i, v_i, 0)$

move to end

Timestep Compression

$CL \cap P \subseteq CLP$: compress-or-compute

M_{CLP} (assume $\exists M_{CL}, M_P$ computing f)

- repeat $time(M_P)$ times:
  - $(\tau, i) :=$ first $c + (s+1)$ bits of cat. tape
  - run M_{CL} on $(\tau, 0)$ for i steps
    - if halts, save answer and revert
    - else, replace (τ, i) with $(\tau_i, v_i, 0)$
- run M_P on $\vec{0}$ space, save answer, and revert $\square$

TIMESTEP COMPRESSION

Theorem 1: $CLP = CL \cap P$ ✓

Theorem 2: $BPCL = CL$

Timestep Compression

Theorem 1: $CLP = CL \cap P$ ✓

Theorem 2: $BPCL = CL$

Theorem 3: $ZPCLP = CL$

$\subseteq$: Theorem 2

Timestep Compression

$\mathsf{CL} \subseteq \mathsf{ZPCLP}$: compress-or-compute

TIMESTEP COMPRESSION

$CL \subseteq ZPCLP$: compress-or-compute

M_{ZPCLP} (assume $\exists M_{CL}, M_{ZPP}$ computing f)

(recall $CL \subseteq ZPP$)

Timestep Compression

$CL \subseteq ZPCLP$: compress-or-compute

M_{ZPCLP} (assume $\exists M_{CL}, M_{ZPP}$ computing f)

(recall $CL \subseteq ZPP$)

- repeat $time(M_{ZPP})$ times:
  - $(\tau, i) :=$ first $c + (s+1)$ bits of cat. tape
  - run M_{CL} on $(\tau, 0)$ for i steps
    - if halts, save answer and revert
    - else, replace (τ, i) with $(\tau_i, v_i, 0)$

      move to end
- run M_{ZPP} on $\vec{0}$ space, save answer, and revert $\square$

TIMESTEP COMPRESSION

Theorem 1: $\mathsf{CLP} = \mathsf{CL} \cap \mathsf{P}$ ✓

Theorem 2: $\mathsf{BPCL} = \mathsf{CL}$

Theorem 3: $\mathsf{ZPCLP} = \mathsf{CL}$ ✓

$\supseteq$: done $\subseteq$: Theorem 2

Timestep Compression

Theorem 1: $\mathsf{CLP} = \mathsf{CL} \cap \mathsf{P}$ ✓

<u>Theorem 2: $\mathsf{BPCL} = \mathsf{CL}$</u>

Theorem 3: $\mathsf{ZPCLP} = \mathsf{CL}$ ✓

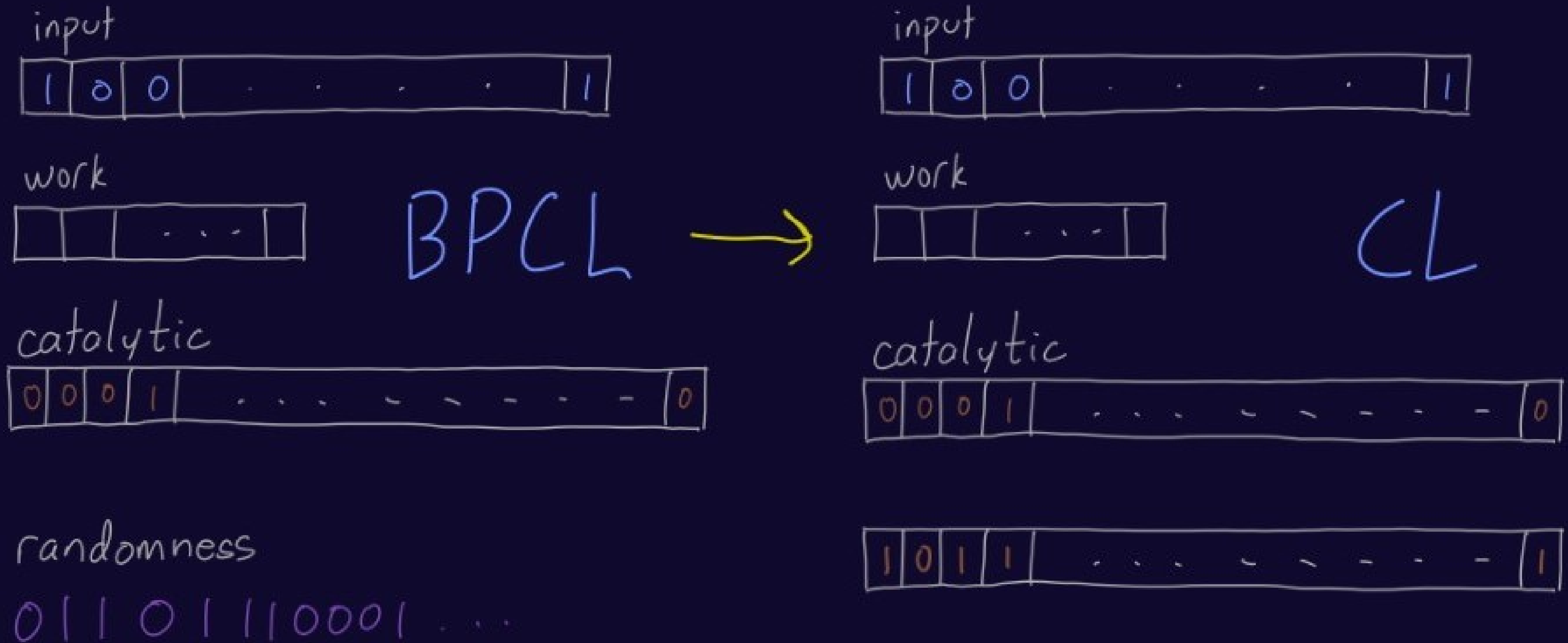
$\supseteq$: done $\qquad$ $\subseteq$: Theorem 2

DERANDOMIZING BPCL

COMPRESS-OR-COMPRESS-OR-RANDOM

DERANDOMIZING BPCL

COMPRESS-OR-<u>COMPRESS-OR-RANDOM</u>



DERANDOMIZING BPCL

COMPRESS-OR-COMPRESS-OR-RANDOM

If catalytic tape is random enough for CL, then done!

If catalytic tape is not random enough for CL, compress it!

DERANDOMIZING BPCL

COMPRESS-OR-COMPRESS-OR-RANDOM

Issues:

- Nisan's test doesn't work against BPCL
- configuration graph can become very large

Derandomizing BPCL

COMPRESS-OR-<u>COMPRESS-OR-RANDOM</u>

Issues:

- ~~Nisan's test doesn't work against BPCL~~

more sophisticated tools available

- configuration graph can become very large

DERANDOMIZING BPCL

COMPRESS-OR-COMPRESS-OR-RANDOM

Issues:

- ~~Nisan's test doesn't work against BPCL~~

more sophisticated tools available

(- configuration graph can become very large)

for later

DERANDOMIZING BPCL

COMPRESS-OR-<u>COMPRESS-OR-RANDOM</u>

Issues:

- ~~Nisan's test doesn't work against BPCL~~

more sophisticated tools available

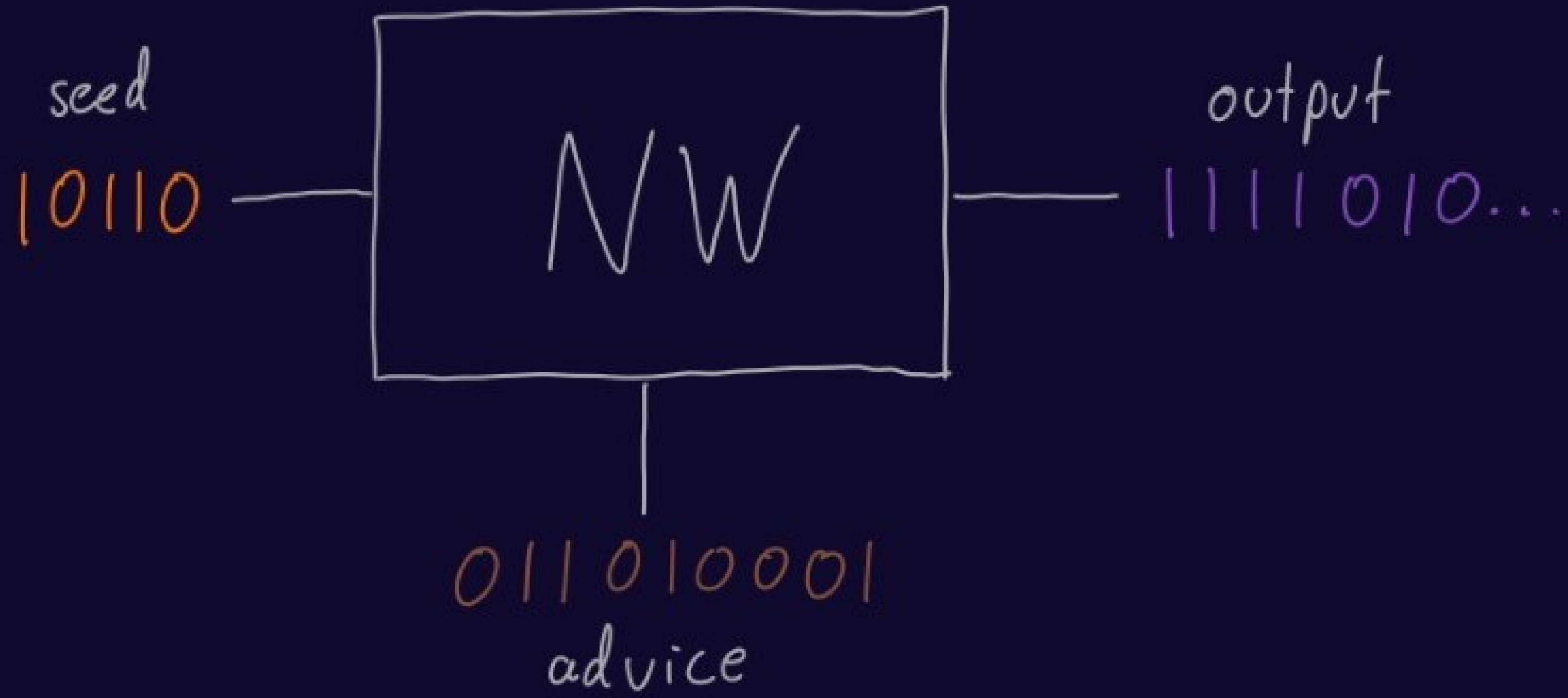
space-bounded Nisan-Wigderson generator

(- configuration graph can become very large)

for later

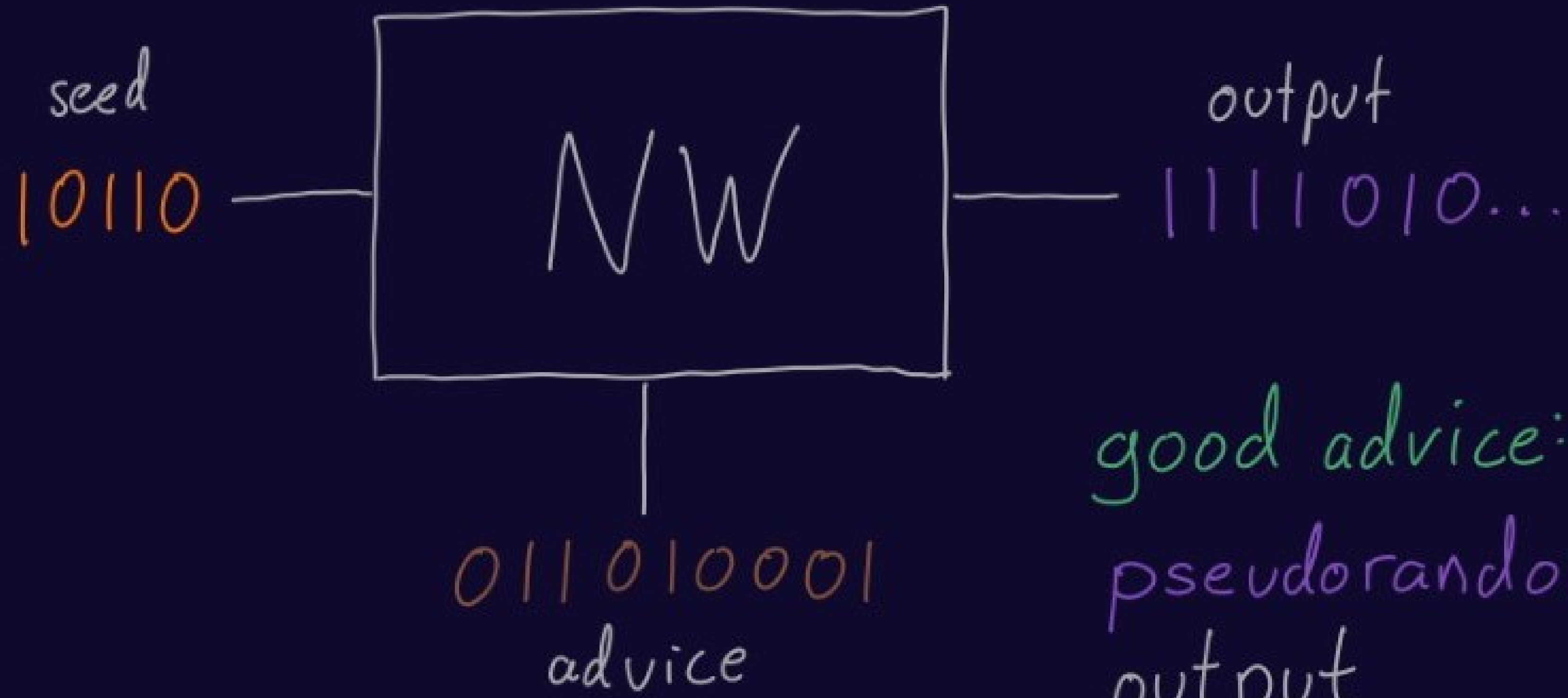
Derandomizing BPCL

COMPRESS-OR-COMPRESS-OR-RANDOM



DERANDOMIZING BPCL

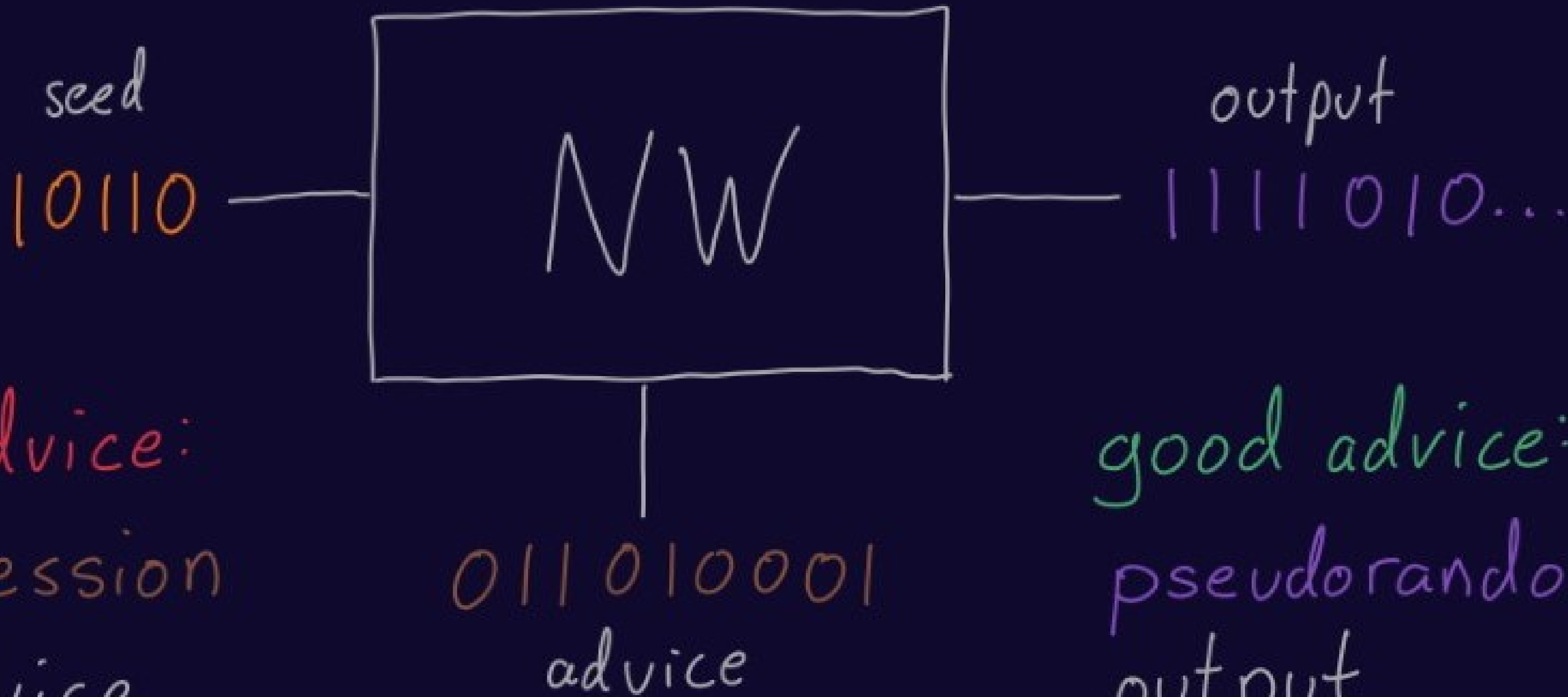
COMPRESS-OR-COMPRESS-OR-RANDOM



good advice: pseudorandom output

DERANDOMIZING BPCL

COMPRESS - OR - COMPRESS - OR - RANDOM



bad advice:
compression for advice

good advice:
pseudorandom output

DERANDOMIZING BPCL

COMPRESS-OR-COMPRESS-OR-RANDOM

DERANDOMIZING BPCL

COMPRESS-OR-COMPRESS-OR-RANDOM

DERANDOMIZING BPCL

COMPRESS - OR - COMPRESS - OR - RANDOM

ASSUME:

only explore a small number of states in total

DERANDOMIZING BPCL

COMPRESS - OR - COMPRESS - OR - RANDOM

ASSUME:
only explore a small number of states in total

- want D s.t. $\mathbb{E}[D(\mathcal{U})] \not\approx \mathbb{E}[D(NW^m(\mathcal{U}))]$

DERANDOMIZING BPCL

ASSUME: only explore a small number of states in total

COMPRESS-OR-COMPRESS-OR-RANDOM

- want D s.t. $\mathbb{E}[D(\mathcal{U})] \not\approx \mathbb{E}[D(NW^m(\mathcal{U}))]$
- our test: do you explore the graph like NW^m?

DERANDOMIZING BPCL

ASSUME:
only explore a small number of states in total

COMPRESS-OR-<u>COMPRESS-OR-RANDOM</u>

- want D s.t. $\mathbb{E}[D(\mathcal{U})] \not\approx \mathbb{E}[D(NW^m(\mathcal{U}))]$
- our test: do you explore the graph like NW^m?
  - if distinguishes, then compress

Derandomizing BPCL

COMPRESS-OR-<u>COMPRESS-OR-RANDOM</u>

ASSUME:

only explore a small number of states in total

- want D s.t. $\mathbb{E}[D(\mathcal{U})] \not\approx \mathbb{E}[D(NW^m(\mathcal{U}))]$
- our test: do you explore the graph like NW^m?
  - if distinguishes, then compress
  - if not, then graph restricted to NW^m is random, can take majority vote

Derandomizing BPCL

COMPRESS-OR-<u>COMPRESS-OR-RANDOM</u>

ASSUME:
only explore a small number of states in total

- want D s.t. $\mathbb{E}[D(\mathcal{U})] \not\approx \mathbb{E}[D(NW^m(\mathcal{U}))]$
- our test: do you explore the graph like NW^m?
  - if distinguishes, then compress
  - if not, then graph restricted to NW^m is random, can take majority vote
- small graph $\rightarrow$ can be carried out in CL

DERANDOMIZING BPCL

COMPRESS-OR-COMPRESS-OR-RANDOM

ASSUME:

only explore a small number of states in total

DERANDOMIZING BPCL

ASSUME:

only explore a small number of states in total

COMPRESS-OR-COMPRESS-OR-RANDOM

- repeat time(M_p) times:
  - $(\tau, i) :=$ first $c+(s+1)$ bits of cat. tape
  - run M_{CL} on $(\tau, 0)$ for i steps
    - if halts, save answer and revert
    - else, replace (τ, i) with $(\tau_i, v_i, 0)$

      move to end
- run M_p, save answer and revert

Derandomizing BPCL

~~Assume only explore a small number of states in total~~

COMPRESS-OR-COMPRESS-OR-RANDOM

- repeat space(M_{BPCL}) times:
  - $(\tau, i) :=$ first $c+(s+1)$ bits of cat. tape
  - run M_{BPCL} on $(\tau, 0)$ with randomness NW^m
    - if $< i$ states explored:
      - run D on NW^m and either compress or take a majority vote and revert
    - else, replace (τ, i) with $(\tau_i, v_i, 0)$
- brute force M_{BPCL}, save answer and revert $\square$

Derandomizing BPCL

The fine print:

- need to consider multiple NW^{m_i} for decompression-related reasons
- distinguisher → "previous bit predictor"
- have to count and compare states, implement distinguisher(s), ... (straightforward)

Wrapping Up

Theorem 1: $CLP = CL \cap P$ ✓

Theorem 2: $BPCL = CL$ ✓

Theorem 3: $ZPCLP = CL$ ✓

Wrapping Up

Open Problems:

Wrapping Up

Open Problems:

1) $CL \overset{?}{\subseteq} P$

Wrapping Up

Open Problems:

1) $CL \overset{?}{=} P$

2) $NCL \overset{?}{=} CL$

Wrapping Up

Open Problems:

1) $CL \subseteq P$

2) $NCL = CL$

3) $BPL = L$


That's all Folks!