Electronic Colloquium on Computational Complexity, Report No. 19 (2025)

Collapsing Catalytic Classes

Michal Koucký*
Charles University
koucky@iuuk.mff.cuni.cz

Ian Mertz*
Charles University
iwmertz@iuuk.mff.cuni.cz

Edward Pyne†
MIT
epyne@mit.edu

Sasha Sami*
Charles University
sashasami@iuuk.mff.cuni.cz

February 27, 2025

Abstract

A catalytic machine is a space-bounded Turing machine with additional access to a second, much larger work tape, with the caveat that this tape is full, and its contents must be preserved by the computation. Catalytic machines were defined by Buhrman et al. (STOC 2014), who, alongside many follow-up works, exhibited the power of catalytic space (CSPACE) and in particular catalytic logspace machines (CL) beyond that of traditional space-bounded machines.

Several variants of CL have been proposed, including non-deterministic and co-non-deterministic catalytic computation by Buhrman et al. (STACS 2016) and randomized catalytic computation by Datta et. al. (CSR 2020). These and other works proposed several questions, such as catalytic analogues of the theorems of Savitch and Immerman and Szelepcsényi. Catalytic computation was recently derandomized by Cook et al. (STOC 2025), but only in certain parameter regimes.

We settle almost all questions regarding randomized and non-deterministic catalytic computation, by giving an optimal reduction from catalytic space with additional resources to the corresponding non-catalytic space classes. One main consequence of this is

$$\mathsf{CL} = \mathsf{CNL}$$

i.e. with access to a large filled hard-drive, non-determinism provides no additional power.

Our results build on the *compress-or-compute framework* of Cook et al. (STOC 2025). Despite proving broader and stronger results, our framework is simpler and more modular.

*Partially supported by the Grant Agency of the Czech Republic under the grant agreement no. 24-10306S and by Charles Univ. project UNCE 24/SCI/008.

†Supported by the NSF GRFP.

ISSN 1433-8092

1 Introduction

The catalytic space model, introduced by Buhrman, Cleve, Koucký, Loff, and Speelman [BCK+14], studies the question of whether full space can be useful to computation. In $\mathsf{CSPACE}[s, c]$ we consider a typical $\mathsf{SPACE}[s]$ machine augmented with a second work tape, called the catalytic tape, which has length c. We think of c to be much larger than s, often exponentially larger; however, this tape is initialized to some arbitrary string τ, and at the end of the computation our machine must reset the catalytic tape to the starting τ.

Despite this restriction, [BCK+14] show that such machines are unexpectedly powerful. Focusing on the class called catalytic logspace ($\mathsf{CL} := \mathsf{CSPACE}[O(\log n), \mathrm{poly}(n)]$), they show that the catalytic tape is at least as powerful as randomization and non-determinism (BPL and NL, respectively), and contains problems (e.g. determinant) which are thought to be in neither.

Catalytic computation appeared in the context of composition for space-bounded functions, where it was unknown whether computing multiple instances of a function causes the space complexity to scale up linearly in tandem. Such techniques and insights were crucially used in the recent result of Cook and Mertz [CM21, CM22, CM24] on the tree evaluation problem, which was later used in a breakthrough by Williams [Wil25] showing $\mathsf{TIME}[t]$ can be simulated in only $\sqrt{t \log t}$ space.

In light of the surprising power of catalytic space, follow-up works have proposed several variants of the base model, such as non-deterministic catalytic computation [BKLS18, GJST19, MS24], randomized catalytic computation [DGJ+20, CLMP25], non-uniform catalytic computation [GKM15, Pot17, RZ21, CM22], and error-prone catalytic computation [GJST24, FMST25], to name a few (see surveys of Koucký [Kou16] and Mertz [Mer23] for an overview).

1.1 Non-Deterministic and Randomized Catalytic Computation

Non-deterministic space has a long yet unresolved history. The first major result, due to Savitch in 1970 [Sav70], states that determinism can simulate non-determinism with only a quadratic space overhead. Much later, Nisan [Nis92] and Saks and Zhou [SZ99] proved that randomness can similarly be simulated with an even smaller blowup showing $\mathsf{BPSPACE}[s] \subseteq \mathsf{SPACE}[s^{3/2}]$.

While derandomization continues to see vigorous work—the exponent has since been further improved by Hoza [Hoz21] by a $o(1)$ factor—Savitch's Theorem remains the best known simulation for non-determinism to date. Subsequent results by Immerman and Szelepscényi [Imm88, Sze88] and Reinhardt and Allender [RA00] show progress from a different angle, namely by showing that NL is closed under complement ($\mathsf{NL} = \mathsf{coNL}$), and NL can be made unambiguous ($\mathsf{NL} = \mathsf{UL}$) assuming strong circuit lower bounds, respectively.

Non-deterministic catalytic space was introduced by Buhrman et al. [BKLS18], who showed that, assuming pseudorandom generators, catalytic non-deterministic logspace (CNL) is also closed under complement ($\mathsf{CNL} = \mathsf{coCNL}$). Later work of [GJST19] extended this by showing, again assuming pseudorandom generators, that CNL can also be made unambiguous ($\mathsf{CNL} = \mathsf{CUL}$). The question of proving such statements unambiguously, or of obtaining a catalytic analogue of Savitch's Theorem, was put forth several times as an open question [BKLS18, GJST19, Kou16, Mer23, CLMP25].

Randomized catalytic space was introduced by Datta et al. [DGJ+20], where they showed that catalytic randomized logspace (CBPL) equals CL under similar pseudorandomness assumptions. This was recently shown unconditionally by Cook et al. [CLMP25]; however, their result only holds for $\mathsf{CSPACE}[s, c]$ when $c = 2^{\Theta(s)}$, while the general case is of interest in other settings [BDS22, Pyn24, FMST25].

1.2 Our Results

We settle essentially every question regarding non-deterministic and randomized catalytic computation. Our most striking result is that, with access to a large pre-filled hard drive, access to non-determinism

gives no additional power:

Theorem 1.

$$\mathsf{CL} = \mathsf{CNL}.$$

Before our result there were no non-trivial connections between non-deterministic and deterministic catalytic computation, even under assumptions. Combined with the result of [CLMP25] that $\mathsf{CBPL} = \mathsf{CL}$, we have that in the context of space-bounded computation, catalytic space acts as one resource to rule them all.

We also have a number of results for other values of s and c. First, a catalytic Savitch's Theorem holds, with overhead matching that of the non-catalytic case:

Theorem 2. *For all $s := s(n)$, $c := c(n)$ such that $\log n \leq s \leq c \leq 2^s$,*

$$\mathsf{CNSPACE}[s, c] \subseteq \mathsf{CSPACE}[O(s^2), O(c)].$$

No results of the above form were known before, even under assumptions.

Second, $\mathsf{CNSPACE}[s, c]$ is closed under complement:

Theorem 3. *For all $s := s(n)$, $c := c(n)$ such that $\log n \leq s \leq c \leq 2^s$,*

$$\mathsf{coCNSPACE}[s, c] \subseteq \mathsf{CNSPACE}[O(s), O(c)].$$

This result was previously known to follow from strong lower bounds [BKLS18].

Finally, $\mathsf{CBPSPACE}[s, c]$ collapses to $\mathsf{CSPACE}[O(s), O(c)]$ for every $c \geq s^2$:

Theorem 4. *For all $s := s(n)$, $c := c(n)$ such that $\log n \leq s \leq c \leq 2^s$,*

$$\mathsf{CBPSPACE}[s, c] \subseteq \mathsf{CSPACE}[O(s), O(c + s^2)].$$

This strongly extends the result of [CLMP25] as well as that of Pyne [Pyn24], which showed that $\mathsf{BPSPACE}[s] \subseteq \mathsf{CSPACE}[O(s), O(s^2)]$.

2 Catalytic Machines

We first define catalytic Turing machines:

Definition 1 (Catalytic space)**.** A catalytic Turing Machine with free space $s := s(n)$ and catalytic space $c := c(n)$ is a Turing machine with the following tapes:

1. a read-only input tape of length n which is initialized to $x \in \{0,1\}^n$
2. a read-write work tape of length s which is initialized to 0^s
3. a read-write catalytic tape of length $c \leq 2^s$ which is initialized to some $\tau \in \{0,1\}^c$

At each time step, a non-deterministic/randomized machine has two (not necessarily distinct) choices for its transition: the 0-choice and 1-choice. Sometimes we can think of the choices as being selected according to the content of an auxiliary non-deterministic/random tape that provides read-only one-way access to its content. This view will be useful later to define the space bounded hierarchy. For machines that compute binary functions we think of the machine as outputting 1 if it reaches an accepting state, and outputting 0 if it reaches a rejecting state. We equip machines that output a value from a larger range with an output tape that provides write-only one-way access. The machine is expected to write its output on this output tape. Oracle machines are equipped with a write-only one-way access oracle tape. To make an oracle query, the machine writes its query on the oracle tape, issues a query request to its oracle, the tape is reset to empty and the machine transitions into its next state depending on the query answer. A catalytic Turing machine $\mathcal{M}$ is said to be valid if for every x and τ, the machine halts in finite time with the catalytic tape containing τ regardless of its non-deterministic/random choices. (In particular, the machine is not allowed to loop forever.)

Definition 2 (Variants of catalytic space)**.** For a boolean function f, we say a catalytic machine computes f if

- (deterministic machine) for every x, the machine outputs $f(x)$.
- (non-deterministic machine) for every x, if $f(x) = 1$ then there is a sequence on non-deterministic choices where the machine accepts x, and if $f(x) = 0$ then for any sequence on non-deterministic choices the machine rejects. A co-non-deterministic machine must always accept when $f(x) = 1$ and sometimes reject when $f(x) = 0$.
- (randomized machine) for every x, the machine outputs $f(x)$ with probability at least $2/3$ over its random choices.

Definition 3 (Complexity classes)**.** A catalytic Turing machine decides a language L if it computes the characteristic function of L.

We define $\mathsf{CSPACE}[s, c]$ to be the class of languages which can be computed by a catalytic Turing machine with free space $s := s(n)$ and catalytic space $c := c(n)$.

We define $\mathsf{CNSPACE}[s, c]$ ($\mathsf{coCNSPACE}[s, c]$) to be the class of languages which can be computed (co-computed) by a non-deterministic catalytic Turing machine with free space $s := s(n)$ and catalytic space $c := c(n)$.

We define $\mathsf{CBPSPACE}[s, c]$ to be the class of languages which can be computed by a randomized catalytic Turing machine with free space $s := s(n)$ and catalytic space $c := c(n)$.

As discussed above, our main focus is on class catalytic logspace, where we fix the parameters s and c to be logarithmic and polynomial in n, respectively.

Definition 4 (Catalytic logspace classes)**.** We define the following instantiations:

- $\mathsf{CL} := \bigcup_{d\in\mathbb{N}} \mathsf{CSPACE}[d \log n, n^d]$
- $\mathsf{CNL} := \bigcup_{d\in\mathbb{N}} \mathsf{CNSPACE}[d \log n, n^d]$
- $\mathsf{coCNL} := \bigcup_{d\in\mathbb{N}} \mathsf{coCNSPACE}[d \log n, n^d]$
- $\mathsf{CBPL} := \bigcup_{d\in\mathbb{N}} \mathsf{CBPSPACE}[d \log n, n^d]$

3 Main Technical Theorem

All of our results follow from a generic reduction from catalytic space with additional resource $\mathcal{B}$, to the corresponding non-catalytic space class:

Theorem 5. *Let $\mathcal{B} \in \{\mathsf{N}, \mathsf{coN}, \mathsf{BP}\}$. Then for all $s := s(n)$, $c := c(n)$ such that $\log n \le s \le c \le 2^s$,*

$$\mathsf{C}\mathcal{B}\mathsf{SPACE}[s, c] \subseteq \mathsf{CSPACE}[O(s), O(c)]^{\mathcal{B}\mathsf{SPACE}[O(s)]}$$

where queries made to the oracle are on inputs of length $2^{O(s)}$, and for $\mathcal{B} = \mathsf{BP}$ the oracle is for the corresponding promise class. Moreover, there is just a single oracle query, and regardless of the query output, the machine still correctly resets the catalytic tape.

3.1 Derivation of Results

Then our main results follow directly from existing simulations of randomized and non-deterministic (standard) space, which we now recall:

Theorem 6 ([BCK+14]). *For all $s := s(n) \geq \log n$,*

$$\mathsf{NSPACE}[O(s)] \subseteq \mathsf{CSPACE}\left[O(s), 2^{O(s)}\right].$$

Theorem 7 ([Sav70]). *For all $s := s(n) \geq \log n$,*

$$\mathsf{NSPACE}[O(s)] \subseteq \mathsf{SPACE}[O(s^2)].$$

Theorem 8 ([Imm88, Sze88]). *For all $s := s(n) \geq \log n$,*

$$\mathsf{coNSPACE}[O(s)] \subseteq \mathsf{NSPACE}[O(s)].$$

Theorem 9 ([Pyn24]). *For all $s := s(n) \geq \log n$,*

$$\mathsf{BPSPACE}[O(s)] \subseteq \mathsf{CSPACE}\left[O(s), O(s^2)\right].$$

From this we can immediately derive all our corollaries using Theorem 5. In particular: Theorem 1 follows from Theorem 6; Theorem 2 follows from Theorem 7; Theorem 3 follows from Theorem 8; and Theorem 4 follows from Theorem 9.

3.2 Discussion

Before going into the proof of Theorem 5, we note some corollaries and extensions of our main results.

Catalytic hierarchies. All our results can be scaled up to the *non-deterministic catalytic hierarchy*, defined by classes $\Sigma_k^{\mathsf{CSPACE}}$ and Π_k^{CSPACE}, as well as the *randomized non-deterministic catalytic hierarchy*, defined by classes $\mathsf{MA}_k^{\mathsf{CSPACE}}$ and $\mathsf{AM}_k^{\mathsf{CSPACE}}$. While $\Sigma_k^{\mathsf{L}} = \mathsf{NL} \subseteq \mathsf{CL}$ for all k, and Sdroievski [MS24] showed that MAL is contained in CL, there are no previously known results connecting the catalytic non-deterministic hierarchies to CL, even under the assumption that $\mathsf{CNL} = \mathsf{coCNL}$.

Theorem 10.

$$\mathsf{CL} = \bigcup_{k\in\mathbb{N}} \Sigma_k^{\mathsf{CL}} = \bigcup_{k\in\mathbb{N}} \mathsf{MA}_k^{\mathsf{CL}} \quad (= \bigcup_{k\in\mathbb{N}} \Pi_k^{\mathsf{CL}} = \bigcup_{k\in\mathbb{N}} \mathsf{AM}_k^{\mathsf{CL}}).$$

Theorem 11. *For all $s := s(n)$, $c := c(n)$ such that $\log n \leq s \leq c \leq 2^s$ and for all $k \in \mathbb{N}$,*

$$\Sigma_k^{\mathsf{CSPACE}}[s,c], \Pi_k^{\mathsf{CSPACE}}[s,c] \subseteq \mathsf{CNSPACE}[O(s), O(c)]$$
$$\mathsf{MA}_k^{\mathsf{CSPACE}}[s,c], \mathsf{AM}_k^{\mathsf{CSPACE}}[s,c] \subseteq \mathsf{CNSPACE}[O(s), O(c+s^2)].$$

Of note, space-bounded hierarchy machines need to be defined carefully, as too much access to the various quantifiers at different points of time can result in a sharp increase in power. We define these machines as taking quantified variables $y_1 \dots y_k$, with the appropriate notions of accepting or rejecting over the choices of y_i, but with the additional restriction that the quantified variables are written from outermost to innermost on the non-deterministic tape, meaning each y_j can be accessed in a read-once fashion, and no y_j can be read after $y_{j'}$ for $j' > j$.

Catalytic and non-catalytic space. While results in catalytic space have been more forthcoming than their classical space counterparts in recent years, it is unclear whether proving connections between ordinary space classes is *formally* any harder (or easier) than proving connections between the corresponding catalytic space classes.

A corollary of our reduction is that ordinary space and catalytic space now share the same fate with regards to the power of additional resources:

Corollary 12. *Let $\mathcal{B}_1, \mathcal{B}_2 \in \{\bot, \mathsf{N}, \mathsf{coN}, \mathsf{U}, \mathsf{BP}\}$. Then*

$$\mathcal{B}_1\mathsf{SPACE}[O(s)] \subseteq \mathcal{B}_2\mathsf{SPACE}[O(s)] \quad \textit{iff}$$
$$\forall c \geq s, \quad \mathcal{B}_1\mathsf{CSPACE}[O(s), O(c)] \subseteq \mathcal{B}_2\mathsf{CSPACE}[O(s), O(c)]$$

where $\bot$ indicates no additional resources.

We make a note about unambiguity here. The proof of Theorem 5 extends to $\mathsf{CUSPACE}$ as well, but as this does not prove any new results we did not include it in our statement; however, it does allow us to get the relevant extension in Corollary 12. This gives the consequence that $\mathsf{NL} = \mathsf{UL}$ holds iff $\mathsf{CNSPACE}[s, c] \subseteq \mathsf{CUSPACE}[O(s), O(c)]$ for every s and c.

Lossy catalytic space. We lastly note one use of our result in the context of another catalytic model, namely *lossy* catalytic space [GJST24, FMST25]. Folkertsma et al. [FMST25] showed that allowing errors when resetting the catalytic tape of a non-deterministic or randomized catalytic machine is equivalent to giving the machine extra free space instead; unfortunately they could not take the further step of equating these error-free classes to deterministic ones, as 1) for non-determinism no such connections were known, and 2) the results of [CLMP25] could no longer be applied to derandomize after adding this extra space. Theorems 2 and 4 are robust to $s \gg \omega(\log c)$, however, and so we immediately get the following results (for definitions and motivation see [FMST25]):

Corollary 13. *For all $s := s(n)$, $c := c(n)$, $e := e(n)$ such that $\log n \leq s \leq c \leq 2^s$ and $e \leq \sqrt{c}$,*

$$\mathsf{LCNSPACE}[s, c, e] \subseteq \mathsf{CSPACE}[O((s + e\log c)^2), O(c)]$$
$$\mathsf{LCBPSPACE}[s, c, e] \subseteq \mathsf{CSPACE}[O(s + e\log c), O(c + (s + e\log c)^2)]$$

4 Proof of Main Result

In this section we prove our central technical theorem. We begin with a discussion of the structure of catalytic machines, followed by an overview of our approach, and lastly we fill in the details to formally prove Theorem 5.

4.1 Configuration Graphs of Catalytic Machines

Let $[n] = \{0, 1, \cdots, n-1\}$. For a graph $\mathcal{G}$, we denote its vertex set by $V(\mathcal{G})$. We use $x \cdot y$ to represent the concatenation of strings x and y.

Let $\mathcal{M}$ be a valid catalytic machine computing f. We will assume without loss of generality that all auxiliary information about the current configuration of $\mathcal{M}$, i.e. the state of $\mathcal{M}$'s internal DFA, the current positions of tape heads for the input, work, and catalytic tapes are all automatically recorded in a designated part of the worktape.[1] (The contents of the output/oracle tape and its head position is not considered to be a part of a machine configuration.)

Definition 5. Let $\mathcal{M}$ be a catalytic machine with work space s and catalytic space c. We denote by $\langle \pi, u \rangle$ the configuration of $\mathcal{M}$ where $\pi \in \{0,1\}^c$ is contained on the catalytic tape and $u \in \{0,1\}^s$ is on its work tape.

Consider the execution of $\mathcal{M}$ on some fixed input x and initial catalytic tape contents τ. Each configuration of $\mathcal{M}$ can be uniquely represented by $\langle \pi, u \rangle$ for some $\pi \in \{0,1\}^c$ and $u \in \{0,1\}^s$. Without loss of generality we define $\text{start}_{\mathcal{M},x,\tau} := \langle \tau, 0^s \rangle$ to be the start configuration and $\text{acc}_{\mathcal{M},x,\tau} := \langle \tau, 1 \cdot 1 \cdot 0^{s-2} \rangle$ to be the unique accepting halt configuration, and $\text{rej}_{\mathcal{M},x,\tau} := \langle \tau, 1 \cdot 0 \cdot 0^{s-2} \rangle$ to be the unique rejecting halt configuration.

It will often be useful to talk about the configuration graph defined by such executions.

[1] Altogether this additional information technically requires additional space $\log n + \log s + \log c + O(1) \leq 3s$, we can handle this by replacing s with $4s$ throughout the proofs, which we omit for clarity.

Definition 6 (Configuration graphs)**.** The configuration graph $\mathcal{G}_{\mathcal{M},x}$ is the directed acyclic graph where each node corresponds to a configuration of $\mathcal{M}$ on input x, where there is a directed edge from $\langle \pi, u \rangle$ to $\langle \pi', u' \rangle$ iff $\langle \pi', u' \rangle$ can be reached from $\langle \pi, u \rangle$ in one execution step of $\mathcal{M}$. The out-degree of every vertex in $\mathcal{G}_{\mathcal{M},x}$ is at most 2, and there is some fixed constant $d_{\mathcal{M}}$ depending only on $\mathcal{M}$ such that each vertex in $\mathcal{G}_{\mathcal{M},x}$ has in-degree at most $d_{\mathcal{M}} - 2$. We call the outgoing edges forward edges of $\langle \tau, v \rangle$. The remaining edges of $\langle \tau, v \rangle$ are backward edges. A halting configuration has no forward edges in $\mathcal{G}_{\mathcal{M},x}$.

We say an edge (v, v') is labeled with $b \in \{0, 1\}$ if it corresponds to a non-deterministic/randomized b-choice of the machine. For deterministic transitions we label the edge with both 0 and 1.

For every catalytic tape τ let $\mathcal{G}_{\mathcal{M},x,\tau}$ be the subgraph of $\mathcal{G}_{\mathcal{M},x}$ induced on configurations of $\mathcal{G}_{\mathcal{M},x}$ that are reachable from $\text{start}_{\mathcal{M},x,\tau}$. Clearly $\mathcal{G}_{\mathcal{M},x,\tau}$ has one source node, namely $\text{start}_{\mathcal{M},x,\tau}$, and up to two sink nodes, namely $\text{acc}_{\mathcal{M},x,\tau}$ and $\text{rej}_{\mathcal{M},x,\tau}$.

Our main method of exploring non-deterministic graphs will be to simply set our non-deterministic sequence to the all-zeroes string:

Definition 7 (0-graph of configuration graphs)**.** Given a configuration graph $\mathcal{G}_{\mathcal{M},x}$, we let the 0-graph $\mathcal{G}^0_{\mathcal{M},x}$ be the undirected graph where only edges with label 0 are retained, and we forget the direction of each edge. Observe that for every τ and $v \in \mathcal{G}_{\mathcal{M},x,\tau}$, $\mathcal{G}^0_{\mathcal{M},x}(v)$ is a tree as each node has at most one forward edge corresponding to the edge labeled 0 in $\mathcal{G}_{\mathcal{M},x}$. Given a configuration v, we let $\mathcal{G}^0_{\mathcal{M},x}(v)$ be the connected component of $\mathcal{G}^0_{\mathcal{M},x}$ containing v.

The following fact is immediate:

Fact 14. *Let v, v' be such that $v \in \mathcal{G}^0_{\mathcal{M},x}(v')$. Then $\mathcal{G}^0_{\mathcal{M},x}(v) = \mathcal{G}^0_{\mathcal{M},x}(v')$ and hence $v' \in \mathcal{G}^0_{\mathcal{M},x}(v)$.*

4.2 Proof Overview

Given a machine $\mathcal{M}$, input x, and starting catalytic tape τ, our focus will be on the 0-graphs reachable from our unique halting states $\text{acc}_{\mathcal{M},x,\tau}, \text{rej}_{\mathcal{M},x,\tau}$. These graphs together include all states reachable from the start configuration:

$$V(\mathcal{G}_{\mathcal{M},x,\tau}) \subseteq V(\mathcal{G}^0_{\mathcal{M},x}(\text{acc}_{\mathcal{M},x,\tau})) \cup V(\mathcal{G}^0_{\mathcal{M},x}(\text{rej}_{\mathcal{M},x,\tau})). \tag{1}$$

This follows as for every configuration v that can be reached from running forward from $\text{start}_{\mathcal{M},x,\tau}$, it must be the case that running forward from v on the all-0s auxiliary input reaches $\text{acc}_{\mathcal{M},x,\tau}$ or $\text{rej}_{\mathcal{M},x,\tau}$ (as otherwise the machine would not be valid). Moreover, we can deterministically and reversibly explore $\mathcal{G}^0_{\mathcal{M},x}(\text{acc}_{\mathcal{M},x,\tau})$ and $\mathcal{G}^0_{\mathcal{M},x}(\text{rej}_{\mathcal{M},x,\tau})$, since both subgraphs are trees and the roots, which are halting configurations, can be identified by examining the worktape contents. If the total size of $\mathcal{G}^0_{\mathcal{M},x}(\text{acc}_{\mathcal{M},x,\tau}) \cup \mathcal{G}^0_{\mathcal{M},x}(\text{rej}_{\mathcal{M},x,\tau})$ is bounded by $2^{O(s)}$, then we could use our $\mathcal{B}$SPACE oracle to solve our function on this graph by identifying each node in the union with its index in the exploration of the two graphs.

Unfortunately, if both graphs are large, this exploration may produce a graph which is too large even to write to the query tape. To avoid this issue, we adopt an idea of Cook et al. [CLMP25] to design what they dubbed a *compress-or-compute* argument. We observe that the average component size is bounded:

$$\mathbb{E}_\tau[|V(\mathcal{G}^0_{\mathcal{M},x}(\text{acc}_{\mathcal{M},x,\tau})) \cup V(\mathcal{G}^0_{\mathcal{M},x}(\text{rej}_{\mathcal{M},x,\tau}))|] \leq 2^s. \tag{2}$$

Similar average-case bounds [BCK+14, DGJ+20, CLMP25] have been observed before, but with an important difference – they all focused on bounding the average size of the forward-reachable graph $\mathcal{G}_{\mathcal{M},x,\tau}$.

The strategy of [CLMP25] for a deterministic CL machine worked as follows. We pad the catalytic tape τ with $s + 1$ bits, producing a new tape (τ, i), and interpret $i \in [2^{s+1}]$ as a counter. They then explore $\mathcal{G}^0_{\mathcal{M},x}(\text{start}_{\mathcal{M},x,\tau})$. If this exploration reaches a halt configuration within $2^{s+1} = \text{poly}(n)$

steps, we successfully decide the language, and revert τ. Otherwise, let $\langle \pi, u \rangle$ be the i-th configuration encountered on this traversal. Note that $|\pi, u| = |\tau, i| - 1$, and hence we can compress our initial catalytic tape by 1 bit by storing (π, u). To revert the tape, we run the machine backwards from $\langle \pi, u \rangle$ until we encounter the start state (which reverts τ), and count the number of steps to reach it (which is i).

However, for a randomized catalytic machine traversing $\mathcal{G}^0_{\mathcal{M},x}$ from the start state may not explore all vertices required to decide the language, even if $\mathcal{G}_{\mathcal{M},x,\tau}$ is small, as it ignores all 1 transitions. To prove $\mathsf{CBPL} = \mathsf{CL}$, Cook et al. [CLMP25] built a much more complicated argument based on taking random walks from the start state, where these walks were themselves generated by a reconstructive PRG [NW94, DPT24]. This added substantial complexity and limited their findings to the case where $c = \text{poly}(2^s)$. Furthermore, their results could not accommodate nondeterministic catalytic machines, as missing even a single state in the random walks could make it impossible to decide the language.

We observe that all of this complexity can be completely removed if we instead reversibly traverse from the *halt* states. For a tape τ, we explore $\mathcal{G}^0_{\mathcal{M},x}(\text{acc}_{\mathcal{M},x,\tau})$ and $\mathcal{G}^0_{\mathcal{M},x}(\text{rej}_{\mathcal{M},x,\tau})$. If both graphs are of size at most 2^{s+1}, we can construct the union graph and determine connectivity in $\mathcal{G}_{\mathcal{M},x,\tau}$ (or whatever else is required to decide the language) via our oracle. Otherwise, suppose WLOG that $|V(\mathcal{G}^0_{\mathcal{M},x}(\text{acc}_{\mathcal{M},x,\tau}))| \geq 2^{s+1}$. Then we adopt the strategy of [CLMP25]. We pad our tape to (τ, i) and let $\langle \pi, u \rangle$ be the i-th configuration in this traversal. We compress

$$(\tau, i) \to (\pi, u, 0)$$

and revert essentially as in [CLMP25], by traversing backwards and counting steps until we reach a halting configuration.

An iterative application of this idea either decides the language or frees c bits on the catalytic tape. If the latter occurs, we can brute force over tapes τ' until we find one for which this graph is small—such a τ' must exist by Equation (2)—which we can then provide to the oracle. The only difference between models is that the oracle is solving a different problem on the configuration graph.

4.3 Formal Proof

We now formalize our discussion from Section 4.2. Fix a valid catalytic machine $\mathcal{M}$ using catalytic space c and work space s computing a language $L \in \mathsf{C}\mathcal{B}\mathsf{SPACE}[s, c]$, and fix an n-bit input x. The following fact is immediate:

Fact 15. *Each tree in $\mathcal{G}^0_{\mathcal{M},x}$ contains at most one halting configuration.*

It follows, then, that the trees in $\mathcal{G}^0_{\mathcal{M},x}$ are pairwise disjoint:

Fact 16. *Let $\tau \neq \tau' \in \{0,1\}^c$ be two distinct contents of the catalytic tape of a catalytic machine $\mathcal{M}$ on an input x. The vertex sets of $\mathcal{G}^0_{\mathcal{M},x}(\text{acc}_{\mathcal{M},x,\tau})$, $\mathcal{G}^0_{\mathcal{M},x}(\text{acc}_{\mathcal{M},x,\tau'})$, $\mathcal{G}^0_{\mathcal{M},x}(\text{rej}_{\mathcal{M},x,\tau})$ and $\mathcal{G}^0_{\mathcal{M},x}(\text{rej}_{\mathcal{M},x,\tau'})$ are pairwise disjoint.*

An immediate consequence of the above is that the average size of these components is bounded:

Lemma 17. *Let $\mathcal{M}$ be a catalytic machine with work space $s := s(n)$ and catalytic space $c := c(n)$, where $\log n \leq s \leq c \leq 2^s$. Then*

$$\mathop{\mathbf{E}}_{\tau \in \{0,1\}^c}[|V(\mathcal{G}^0_{\mathcal{M},x}(\text{acc}_{\mathcal{M},x,\tau}))|] \leq 2^s \text{ and } \mathop{\mathbf{E}}_{\tau \in \{0,1\}^c}[|V(\mathcal{G}^0_{\mathcal{M},x}(\text{rej}_{\mathcal{M},x,\tau}))|] \leq 2^s.$$

Finally, the forward-reachable graph from every state is contained in the trees rooted at the two possible halt states:

Lemma 18. *Let $\mathcal{M}$ be a valid catalytic machine, and let $\langle \pi, u \rangle$ be an arbitrary node in $\mathcal{G}_{\mathcal{M},x,\tau}$. Then, $\langle \pi, u \rangle \in V(\mathcal{G}^0_{\mathcal{M},x}(\text{acc}_{\mathcal{M},x,\tau})) \cup V(\mathcal{G}^0_{\mathcal{M},x}(\text{rej}_{\mathcal{M},x,\tau}))$.*

Proof. Since $\langle \pi, u \rangle$ is reachable by $\mathcal{M}$ on x from $\text{start}_{\mathcal{M},x,\tau}$, there exists some non-deterministic sequence σ such that $\mathcal{M}$ reaches configuration $\langle \pi, u \rangle$. Now consider the non-deterministic sequence $\sigma \cdot 0^*$, which takes us to $\langle \pi, u \rangle$ and subsequently uses 0 as its non-deterministic choices. Since we started from $\text{start}_{\mathcal{M},x,\tau}$, this must eventually either reach $\text{acc}_{\mathcal{M},x,\tau}$ or $\text{rej}_{\mathcal{M},x,\tau}$, and since the latter part of this walk resides entirely inside $\mathcal{G}^0_{\mathcal{M},x}(\langle \pi, u \rangle)$, either $\mathcal{G}^0_{\mathcal{M},x}(\text{acc}_{\mathcal{M},x,\tau})$ or $\mathcal{G}^0_{\mathcal{M},x}(\text{rej}_{\mathcal{M},x,\tau})$ must contain $\langle \pi, u \rangle$. □

Exploration of Graphs. We formally define the notation related to graphs for the purpose of graph exploration. For an undirected graph $G = (V, E)$ and $v \in V$, we let $G(v)$ be the component of v. Let d be the maximum degree of G.

We assume that there is a cyclic ordering of edges at each vertex that define a rotation map $\textsc{Rot} : V \times [d] \to V \times [d]$, such that $\textsc{Rot}(v, i) = (u, j)$ if the i-th edge of v is the j-th edge of u (if $i \geq \deg(v)$ then $\textsc{Rot}(v, i) = (v, i)$). If we identify the i-th directed edge leaving v by (v, i), then $\textsc{Rot}(v, i)$ flips the direction of the edge.

We will consider walks on G starting from a given edge (v, i) that follow an Eulerian tour of $G(v)$. For $v \in V$ and $i \in [\deg(v)]$, the next edge of the walk is given by $\textsc{Next}(v, i) = (u, j+1 \mod \deg(v))$ where $\textsc{Rot}(v, i) = (u, j)$. A step back is taken by $\textsc{StepBack}(u, j) = \textsc{Rot}(u, j - 1 \mod \deg(u))$.

We will index edges at each vertex (configuration) v in $\mathcal{G}^0_{\mathcal{M},x}$ by $[\deg(v)]$ so that the forward edge gets index 0 (if there is a forward edge). We fix the rotation map $\textsc{Rot}^0$ of $\mathcal{G}^0_{\mathcal{M},x}$ arbitrarily otherwise.

Defining Catalytic Subroutines. We can define a catalytic subroutine $\textsc{Rot}(\langle \pi, u \rangle, i)$ which uses space $O(s)$ that, given $\pi \in \{0,1\}^c$ on the catalytic tape, and $u \in \{0,1\}^s$ and $i \in [d_{\mathcal{M}}]$ on the worktape, replaces $(\langle \pi, u \rangle, i)$ by $(\langle \pi', u' \rangle, j) = \textsc{Rot}^0(\langle \pi, u \rangle, i)$. Clearly, $\textsc{Next}(\langle \pi, u \rangle, i)$ and $\textsc{StepBack}(\langle \pi', u' \rangle, j)$ can be implemented by catalytic subroutines working in space $O(s)$.

Let $S = 2^{O(s)}$ be a (easily computable) function of n. We can also define a catalytic subroutine $\textsc{Walk}(\langle \tau, v \rangle, i, t)$ which applies the subroutine $\textsc{Next}(\cdot)$ t times on $(\langle \tau, v \rangle, i)$ where $t \leq S$. The procedure uses an additional work space of size $O(s)$, and the input $(\langle \tau, v \rangle, i)$ is replaced by the output $(\langle \pi, u \rangle, j)$ on the respective tapes.

We will call the subroutine $\textsc{Walk}(\langle \tau, v \rangle, i, t)$ for halting configurations $\langle \tau, v \rangle$ and $i = 0$. For these calls, we can define an inverse subroutine $\textsc{CountStepsBack}(\langle \pi, u \rangle, j)$ which calculates $\ell \leq S$, the number of times we need to apply $\textsc{StepBack}(\cdot)$ on $(\langle \pi, u \rangle, j)$ before reaching $(\langle \tau, v \rangle, 0)$ for some halting configuration $\langle \tau, v \rangle$. This subroutine uses extra $O(s)$ work space, it replaces $(\langle \pi, u \rangle, j)$ by $(\langle \tau, v \rangle, 0)$, and returns the count to a designated area of the work space. If the count is bigger than S it returns ∞.

Combining $\textsc{Walk}(\langle \tau, v \rangle, i, t)$ with $\textsc{CountStepsBack}(\langle \pi, u \rangle, j)$ we can create a catalytic subroutine $\textsc{ConfBit}(b, \langle \tau, v \rangle, t)$ which, given a halting configuration $\langle \tau, v \rangle$, $b \leq c + s$ and $t \leq S$, determines the b-th bit of the configuration reached by $\textsc{Walk}(\langle \tau, v \rangle, 0, t)$. For a halting configuration $\langle \tau, v \rangle$, $\textsc{ConfBit}(\langle \tau, v \rangle, t)$ preserves $\langle \tau, v \rangle$ and t on its tape when it finishes its computation. Additionally, we define a subroutine $\textsc{Canon}(\langle \tau, v \rangle, i, t)$ which preserves $\langle \tau, v \rangle, i, t$ and returns 1, if the edge $(\langle \pi, u \rangle, j)$ reached by $\textsc{Walk}(\langle \tau, v \rangle, i, t)$ has $j = 0$, and it returns 0 otherwise. We think of t as the *canonical* index of the configuration $\langle \pi, u \rangle$ within $\mathcal{G}^0_{\mathcal{M},x}(\langle \tau, v \rangle)$.

Similarly, we can define a catalytic subroutine $\textsc{Size}(\langle \tau, v \rangle)$, which, for a halting configuration $\langle \tau, v \rangle$, determines the minimum number of steps $t \geq 1$, such that $\textsc{Walk}(\langle \tau, v \rangle, 0, t)$ returns back to $(\langle \tau, v \rangle, 0)$. If t is at most S, it outputs t; otherwise, it outputs ∞. Since $\mathcal{G}^0_{\mathcal{M},x}(\langle \tau, v \rangle)$ forms a tree for a halting configuration $\langle \tau, v \rangle$, the subroutine returns twice the number of edges of $\mathcal{G}^0_{\mathcal{M},x}(\langle \tau, v \rangle)$ iff $2 \leq |V(\mathcal{G}^0_{\mathcal{M},x}(\langle \tau, v \rangle))| \leq \frac{S}{2} + 1$. It returns 1 or ∞ otherwise. Additionally, the subroutine uses $O(s)$ extra work space.

Note that all the above procedures should ignore any portions of the machine tapes not directly referenced therein.

The Main Subroutine. Our plan is to use the Compress-or-Compute strategy. Given a starting catalytic tape τ for a machine $\mathcal{M}$, we will either use $\mathcal{G}^0_{\mathcal{M},x}(\text{acc}_{\mathcal{M},x,\tau})$ and $\mathcal{G}^0_{\mathcal{M},x}(\text{rej}_{\mathcal{M},x,\tau})$ to construct a small graph that determines the outcome of the computation of $\mathcal{M}$ on x, or we will use vertices in $\mathcal{G}^0_{\mathcal{M},x}(\text{acc}_{\mathcal{M},x,\tau})$ and $\mathcal{G}^0_{\mathcal{M},x}(\text{rej}_{\mathcal{M},x,\tau})$ to compress the catalytic tape. We can state the main Compress-or-Compute lemma:

Lemma 19. *Let $\mathcal{M}$ be a catalytic machine with work space $s := s(n)$ and catalytic space $c := c(n)$, where $\log n \le s \le c \le 2^s$, and let $x \in \{0,1\}^n$ be an input for $\mathcal{M}$ written on the input tape. Let $B = 2s$ and $S = 2^B$, and let $\tau \in \{0,1\}^c$ and $\texttt{tar} \in \{0,1\}^B$ be given on the catalytic tape. There is a catalytic subroutine* ComputeOrCompress$(\tau, \texttt{tar})$ *which takes one of the following two actions:*

1. *Compute: If both $\mathcal{G}^0_{\mathcal{M},x}(\text{acc}_{\mathcal{M},x,\tau})$ and $\mathcal{G}^0_{\mathcal{M},x}(\text{rej}_{\mathcal{M},x,\tau})$ are of size at most $S/2+1$, then it outputs a directed graph G and two vertices r and t such that the forward reachable graph from r is isomorphic to $\mathcal{G}_{\mathcal{M},x,\tau}$, with $\text{start}_{\mathcal{M},x,\tau}$ mapping to r and $\text{acc}_{\mathcal{M},x,\tau}$ mapping to t.*
2. *Compress: Otherwise, it replaces τ by $\pi \in \{0,1\}^c$ and $\texttt{tar}$ by $(u, j, 0^{s-\log d_{\mathcal{M}}})$, where $u \in \{0,1\}^s$ and $j \in [d_{\mathcal{M}}]$ have the property that* CountStepsBack$(\langle \pi, u\rangle, j)$ *replaces π by τ and returns $\texttt{tar}$ as the number of steps.*

The subroutine returns a bit indicating which action it took, and the procedure leaves other portions of the tapes unchanged. Furthermore ComputeOrCompress$(\tau, \texttt{tar})$ *uses additional space $O(s)$.*

Proof. Recall that $\mathcal{G}_{\mathcal{M},x,\tau}$ is a the subgraph of $\mathcal{G}_{\mathcal{M},x}$ induced on configurations of $\mathcal{G}_{\mathcal{M},x}$ reachable from $\text{start}_{\mathcal{M},x,\tau}$, and that by Lemma 18 we have

$$V(\mathcal{G}_{\mathcal{M},x,\tau}) \subseteq V(\mathcal{G}^0_{\mathcal{M},x}(\text{acc}_{\mathcal{M},x,\tau}) \cup V(\mathcal{G}^0_{\mathcal{M},x}(\text{rej}_{\mathcal{M},x,\tau})).$$

In brief, if both $\mathcal{G}^0_{\mathcal{M},x}(\text{acc}_{\mathcal{M},x,\tau})$ and $\mathcal{G}^0_{\mathcal{M},x}(\text{rej}_{\mathcal{M},x,\tau})$ are of size at most $\frac{S}{2}+1$, we can explore them completely using Walk$(\cdot)$, and reconstruct a graph G containing $\mathcal{G}_{\mathcal{M},x,\tau}$. If $\mathcal{G}^0_{\mathcal{M},x}(\text{acc}_{\mathcal{M},x,\tau})$ or $\mathcal{G}^0_{\mathcal{M},x}(\text{rej}_{\mathcal{M},x,\tau})$ is large we can compress τ.

Initial check: We first check the sizes of $\mathcal{G}^0_{\mathcal{M},x}(\text{acc}_{\mathcal{M},x,\tau})$ and $\mathcal{G}^0_{\mathcal{M},x}(\text{rej}_{\mathcal{M},x,\tau})$ using calls to Size$(\text{acc}_{\mathcal{M},x,\tau})$ and Size$(\text{rej}_{\mathcal{M},x,\tau})$. Since $\text{acc}_{\mathcal{M},x,\tau} = \langle \tau, 1\cdot 1\cdot 0^{s-2}\rangle$ and $\text{rej}_{\mathcal{M},x,\tau} = \langle \tau, 1\cdot 0\cdot 0^{s-2}\rangle$, both states are easy to prepare given $\text{start}_{\mathcal{M},x,\tau}$, and Size can be run in $O(s)$ space. If either of the sizes exceeds $S/2+1$, meaning if either call to Size returns ∞, we move to the *compress case*; otherwise, we proceed to the *compute case*.

Compute case: If both the graphs have a size of at most $S/2+1$, we can explore configurations of $\mathcal{G}^0_{\mathcal{M},x}(\text{acc}_{\mathcal{M},x,\tau})$ and $\mathcal{G}^0_{\mathcal{M},x}(\text{rej}_{\mathcal{M},x,\tau})$ using ConfBit$(\cdot)$. We will index the configurations of G by $[S]\times[2]$. The configuration indexed (i,b) is the configuration reached by Walk$(\text{acc}_{\mathcal{M},x,\tau}, 0, i)$ if $b = 0$ and by Walk$(\text{rej}_{\mathcal{M},x,\tau}, 0, i)$ otherwise.

For each $(i,b),(j,d) \in [S]\times[2]$, we can check whether there is an edge from the configuration (i,b) to (j,d) in $\mathcal{G}_{\mathcal{M},x}$ by comparing them bit-by-bit using ConfBit$(\cdot)$. If i and j are canonical indexes of their respective configurations (which can be checked by calling Canon$(\cdot)$) we connect them by an edge in G. Hence, we output a graph G on $[S]\times[2]$ where the connectivity between the canonical indexes of configurations from $\mathcal{G}_{\mathcal{M},x,\tau}$ is the same as in $\mathcal{G}_{\mathcal{M},x,\tau}$. By checking each $(i,b) \in [S]\times[2]$, we can locate a canonical copy of a configuration $\text{start}_{\mathcal{M},x,\tau}$ and $\text{acc}_{\mathcal{M},x,\tau}$, and output them as r and t.

This computation will use at most $O(s)$ space on the work tape to run Walk and ConfBit, and it will preserve τ and $\texttt{tar}$ on the catalytic tape.

Compress case: Consider without loss of generality the case where Size$(\text{acc}_{\mathcal{M},x,\tau})$ returns ∞. We prepare $\text{acc}_{\mathcal{M},x,\tau} = \langle \tau, v\rangle$ where $v = 1\cdot 1\cdot 0^{s-2}$, and run Walk$(\text{acc}_{\mathcal{M},x,\tau}, i, \texttt{tar})$ with i set to 0,

treating $\mathtt{tar}$ as a natural number evaluated in base-2, plus one. The result of Walk will be to replace τ by some π, v by some $u \in \{0,1\}^s$, and i by some j. We replace $\mathtt{tar}$ by $(u, j, 0^{s-\log d_{\mathcal{M}}})$ and end the procedure.

This computation utilizes at most $O(s)$ workspace, which is all that is needed for the subroutine Walk. Since $\textsc{Size}(\mathrm{acc}_{\mathcal{M},x,\tau})$ returns ∞, it indicates that during the first S steps prescribed by Walk, we do not return to the edge $(\mathrm{acc}_{\mathcal{M},x,\tau}, 0)$. Therefore, given that $\mathtt{tar} \le S$ (with $\mathtt{tar}$ treated as a natural number), calling $\textsc{CountStepsBack}(\langle \pi, u\rangle, j)$ replaces π with τ and returns $\mathtt{tar}$ as the number of steps taken. Consequently, the output possesses the required properties. □

We now finish the proof of Theorem 5 using the compression and decompression procedures from above.

Proof of Theorem 5. Let $\mathcal{M}$ be our $\mathsf{CBSPACE}[c, s]$ machine and fix an n-bit input x. Define $B := 2s$ and $S := 2^B$.

Our goal is to output a directed graph G and two vertices r and t where t is reachable from r in G iff $\mathcal{M}$ accepts x. The graph G will be obtained by the Compress-or-Compute subroutine of Lemma 19 which will be run for a suitable choice of τ. Given such a graph G, we can clearly obtain the answer to our function by appealing to our oracle, as it will be a graph of size at most $2S = 2^{2s+1}$—thus it can be analyzed by a $\mathcal{B}\mathsf{SPACE}[O(s)]$ machine—which represents $\mathcal{G}_{\mathcal{M},x,\tau}$.

We let $k \ge 2 + 2c/s$, and we think of our catalytic tape as consisting of blocks

$$(\tau, \mathtt{tar}_0, \mathtt{tar}_1, \ldots, \mathtt{tar}_{k-1})$$

where $\tau \in \{0,1\}^c$ and $\mathtt{tar}_i \in \{0,1\}^B$. Note that this gives a total catalytic length of $c + (2 + 2c/s) \cdot 2s \le 10c$ as desired.

We iterate over $i \in [k]$ and call $\textsc{ComputeOrCompress}(\tau_i, \mathtt{tar}_i)$, where τ_i is the first c bits of the catalytic tape at the time when we begin the i-th iteration; thus $\tau_0 := \tau$. Each call $\textsc{ComputeOrCompress}(\tau_i, \mathtt{tar}_i)$ either outputs the desired graph G or compresses $\mathtt{tar}_i$. In the former case, we obtain the graph G on which we can run our oracle to obtain the solution to our original function, at which point we can decompress (see below). In the latter case, τ_i is replaced by some π, which we refer to as τ_{i+1}, and $\mathtt{tar}_i$ is replaced by some $(u_i, j_i, 0^{s-\log d_{\mathcal{M}}})$; we then move on to the $(i+1)$-st iteration.

If none of the calls gives the desired graph, then since we free at least $s/2$ bits of the catalytic tape during each iteration, we free at least $c + s$ bits of space on the catalytic tape in total. We can use this space to iterate over all possible $\tau_k \in \{0,1\}^c$ and set $\mathtt{tar}_k = 1^B$, and see for which one $\textsc{ComputeOrCompress}(\tau_k, \mathtt{tar}_k)$ falls into the compute case. Whenever it does not do so, i.e. whenever it falls into the compress case, then it replaces the current τ_k by some π and $\mathtt{tar}_k$ by some $(u, j, 0^{s-\log d_{\mathcal{M}}})$; we will revert it back to τ_k and $\mathtt{tar}_k$ by running $\textsc{CountStepsBack}(\langle \pi, u\rangle, j)$, which will replace π by τ_k and return $\mathtt{tar}_k$ as the number of steps. We increment τ_k viewed as a binary counter and continue for our new τ_k.

By Lemma 17, $\mathbf{E}_{\tau \in \{0,1\}^c}[|V(\mathcal{G}^0_{\mathcal{M},x}(\mathrm{acc}_{\mathcal{M},x,\tau}))|] \le 2^s$ and $\mathbf{E}_{\tau \in \{0,1\}^c}[|V(\mathcal{G}^0_{\mathcal{M},x}(\mathrm{rej}_{\mathcal{M},x,\tau}))|] \le 2^s$. Thus, for at least half of the possible starting states τ, we have that $|V(\mathcal{G}^0_{\mathcal{M},x}(\mathrm{acc}_{\mathcal{M},x,\tau}))| \le 4 \cdot 2^s$ and $|V(\mathcal{G}^0_{\mathcal{M},x}(\mathrm{acc}_{\mathcal{M},x,\tau}))| \le 4 \cdot 2^s$. In particular, there must exist some $\tau \in \{0,1\}^c$ for which both $V(\mathcal{G}^0_{\mathcal{M},x}(\mathrm{acc}_{\mathcal{M},x,\tau}))$ and $V(\mathcal{G}^0_{\mathcal{M},x}(\mathrm{rej}_{\mathcal{M},x,\tau}))$ are smaller than $\frac{S}{2}$, and on this $\tau_k = \tau$ we reach the compute case and output the desired graph G.

Recall that once we find a graph G in the compute case, we can appeal to our oracle to obtain the answer to our function. If we do so via the τ_k loop above we then erase $(\tau_k, \mathtt{tar}_k)$ on our tape. We are now left at the state immediately following $\textsc{ComputeOrCompress}(\tau_i, \mathtt{tar}_i)$ for some $i \in [k]$; our last step is to decompress each round of $\textsc{ComputeOrCompress}(\tau_i, \mathtt{tar}_i)$, in reverse order, that we executed until the final call.

To decompress $\pi = \tau_{i+1}$ and $(u_i, j_i, 0^{s-\log d_{\mathcal{M}}})$, we call $\textsc{CountStepsBack}(\langle \tau_{i+1}, u_i\rangle, j_i)$ which will replace τ_{i+1} by τ_i and return $\mathtt{tar}_i$ as the number of steps. Hence we can restore $\mathtt{tar}_i$, and τ_i,

and then we move on to $i-1$. Our final state will once again be the initial catalytic tape

$$(\tau, \mathtt{tar}_0, \mathtt{tar}_1, \ldots, \mathtt{tar}_{k-1})$$

at which point we return our saved answer and halt. □

Acknowledgements

We thank Ninad Rajgopal for discussions relating to the catalytic hierarchies.

References

[BCK+14] Harry Buhrman, Richard Cleve, Michal Kouckỳ, Bruno Loff, and Florian Speelman. Computing with a full memory: catalytic space. In *ACM Symposium on Theory of Computing (STOC)*, pages 857–866, 2014. doi:10.1145/2591796.2591874.

[BDS22] Sagar Bisoyi, Krishnamoorthy Dinesh, and Jayalal Sarma. On pure space vs catalytic space. *Theoretical Computer Science (TCS)*, 921:112–126, 2022. doi:10.1016/J.TCS.2022.04.005.

[BKLS18] Harry Buhrman, Michal Kouckỳ, Bruno Loff, and Florian Speelman. Catalytic space: Non-determinism and hierarchy. *Theory of Computing Systems (TOCS)*, 62(1):116–135, 2018. doi:10.1007/S00224-017-9784-7.

[CLMP25] James Cook, Jiatu Li, Ian Mertz, and Edward Pyne. The structure of catalytic space: Capturing randomness and time via compression. In *ACM Symposium on Theory of Computing (STOC)*, 2025.

[CM21] James Cook and Ian Mertz. Encodings and the tree evaluation problem. *Electronic Colloquium on Computational Complexity (ECCC)*, TR21-054, 2021. URL: https://eccc.weizmann.ac.il/report/2021/054.

[CM22] James Cook and Ian Mertz. Trading time and space in catalytic branching programs. In *IEEE Conference on Computational Complexity (CCC)*, volume 234 of *Leibniz International Proc. in Informatics (LIPIcs)*, pages 8:1–8:21, 2022. doi:10.4230/LIPIcs.CCC.2022.8.

[CM24] James Cook and Ian Mertz. Tree evaluation is in space O(log n · log log n). In *ACM Symposium on Theory of Computing (STOC)*, pages 1268–1278. ACM, 2024. doi:10.1145/3618260.3649664.

[DGJ+20] Samir Datta, Chetan Gupta, Rahul Jain, Vimal Raj Sharma, and Raghunath Tewari. Randomized and symmetric catalytic computation. In *CSR*, volume 12159 of *Lecture Notes in Computer Science (LNCS)*, pages 211–223, 2020. doi:10.1007/978-3-030-50026-9_15.

[DPT24] Dean Doron, Edward Pyne, and Roei Tell. Opening up the distinguisher: A hardness to randomness approach for BPL = L that uses properties of BPL. In *ACM Symposium on Theory of Computing (STOC)*, pages 2039–2049, 2024.

[FMST25] Marten Folkertsma, Ian Mertz, Florian Speelman, and Quinten Tupker. Fully characterizing lossy catalytic computation. In *Innovations in Theoretical Computer Science Conference (ITCS)*, volume 325 of *LIPIcs*, pages 50:1–50:13, 2025.

[GJST19] Chetan Gupta, Rahul Jain, Vimal Raj Sharma, and Raghunath Tewari. Unambiguous catalytic computation. In *Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, volume 150 of *Leibniz International Proc. in Informatics (LIPIcs)*, pages 16:1–16:13, 2019. doi:10.4230/LIPIcs.FSTTCS.2019.16.

[GJST24] Chetan Gupta, Rahul Jain, Vimal Raj Sharma, and Raghunath Tewari. Lossy catalytic computation. *Computing Research Repository (CoRR)*, abs/2408.14670, 2024.

[GKM15] Vincent Girard, Michal Koucký, and Pierre McKenzie. Nonuniform catalytic space and the direct sum for space. *Electronic Colloquium on Computational Complexity (ECCC)*, TR15-138, 2015.

[Hoz21] William M. Hoza. Better pseudodistributions and derandomization for space-bounded computation. In *Proceedings of the 25th International Conference on Randomization and Computation (RANDOM)*, pages 28:1–28:23, 2021.

[Imm88] Neil Immerman. Nondeterministic space is closed under complementation. *SIAM Journal on Computing (SICOMP)*, 17(5):935–938, 1988.

[Kou16] Michal Koucký. Catalytic computation. *Bulletin of the EATCS (B.EATCS)*, 118, 2016.

[Mer23] Ian Mertz. Reusing space: Techniques and open problems. *Bulletin of the EATCS (B.EATCS)*, 141:57–106, 2023.

[MS24] Nicollas Mocelin Sdroievski. *Derandomization vs. Lower Bounds for Arthur-Merlin Protocols*. PhD thesis, University of Wisconsin–Madison, Madison, WI, 2024.

[Nis92] Noam Nisan. Pseudorandom generators for space-bounded computation. *Combinatorica*, 12(4):449–461, 1992.

[NW94] Noam Nisan and Avi Wigderson. Hardness vs. randomness. *Journal of Computer and System Sciences (J.CSS)*, 49(2):149–167, 1994.

[Pot17] Aaron Potechin. A note on amortized branching program complexity. In *IEEE Conference on Computational Complexity (CCC)*, volume 79 of *Leibniz International Proc. in Informatics (LIPIcs)*, pages 4:1–4:12, 2017. doi:10.4230/LIPIcs.CCC.2017.4.

[Pyn24] Edward Pyne. Derandomizing logspace with a small shared hard drive. In *IEEE Conference on Computational Complexity (CCC)*, volume 300 of *LIPIcs*, pages 4:1–4:20, 2024.

[RA00] Klaus Reinhardt and Eric Allender. Making nondeterminism unambiguous. *SIAM Journal on Computing (SICOMP)*, 29(4):1118–1131, 2000.

[RZ21] Robert Robere and Jeroen Zuiddam. Amortized circuit complexity, formal complexity measures, and catalytic algorithms. In *IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 759–769. IEEE, 2021. doi:10.1109/FOCS52979.2021.00079.

[Sav70] Walter J. Savitch. Relationships between nondeterministic and deterministic tape complexities. *Journal of Computer and System Sciences (J.CSS)*, 4(2):177–192, 1970. doi:10.1016/S0022-0000(70)80006-X.

[SZ99] Michael E. Saks and Shiyu Zhou. $\mathbf{BP_HSPACE}[S] \subseteq \mathbf{DSPACE}[S^{3/2}]$. *JCSS*, 58(2):376–403, 1999.

[Sze88] Róbert Szelepcsényi. The method of forced enumeration for nondeterministic automata. *Acta Informatica*, 26(3):279–284, 1988. URL: http://dx.doi.org/10.1007/BF00299636, doi:10.1007/bf00299636.

[Wil25] Ryan Williams. Simulating time in square-root space. *Electron. Colloquium Comput. Complex.*, TR25-017, 2025. URL: https://eccc.weizmann.ac.il/report/2025/017/.

ECCC ISSN 1433-8092

https://eccc.weizmann.ac.il