Understanding Robust Catalytic Computing

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

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

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

Abstract

Catalytic computing concerns space bounded computation which starts with memory full of data that have to be restored by the end of the computation. Lossy catalytic computing, defined by Gupta et al. [GJST24] and fully characterized by Folkertsma et al. [FMST25], is the study of allowing a small number of errors when resetting the catalytic tape at the end of a computation. Such a notion is useful when considering the robust use of catalytic techniques in the study of ordinary space-bounded algorithms. To that end however, defining and characterizing less strict notions of error was left open by [FMST25] and other works [Mer23].

We expand the definition of possible resetting error in three natural ways:

1. randomized catalytic computation which can *completely destroy* the catalytic tape with some probability over the randomness
2. randomized catalytic computation which makes a bounded number of errors *in expectation over the randomness*
3. deterministic catalytic computation which makes a bounded number of errors *in expectation over the initial catalytic tape itself*

We show a near complete characterization of the above models, both in the general case and in the logspace polynomial-time regime, by showing equivalences either between one another, to errorless catalytic space models, or to standard time or space complexity classes. Under a derandomization assumption, we show a near full collapse of all existing catalytic classes in the logspace regime.

Contents

*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.

1 Introduction

1.1 Catalytic computation

When designing space-bounded algorithms, a typical assumption has been that memory being used for storage is locked away and should not be touched while running unrelated subroutines. A recent paradigm called *catalytic computing* has challenged this assumption, showing that given memory which is full with *arbitrary data*, a machine can use this memory to aid computations while still *perfectly resetting it* at the end of the computation.

Since the formulation of the model by Buhrman, Cleve, Koucký, Loff, and Speelman [BCK+14], many works have studied the properties of computing with the assistance of a large full hard drive, known as *catalytic space*. The original result of [BCK+14] showed that *catalytic logspace* (CL) is seemingly strictly more powerful than NL, BPL, or even stronger classes; this was later strengthened to include important problems such as bipartite matching [AM25], linear matroid intersection [AAV26], and matrix powering [AFM+25]. A structural theory of catalytic space has emerged regarding the power of randomness [DGJ+20, CLMP25, Pyn25, KMPS25], non-determinism [BKLS18, GJST19, CLMP25, KMPS25], and non-uniformity [Pot17, RZ21, CM22, CM25]. Finally there have been attempts to situate catalytic space in other settings, such as time-space efficient algorithms [CP26], alternative models [BDS22, BFM+25, CGM+25], and, most important, the study of non-catalytic space.

With regards to this final point, the techniques of using full memory were also used in the context of ordinary space-bounded computation, in particular in the realms of space-bounded derandomization [DPT24, LPT24, Pyn25, DPTW25, GTS25] and recently on the relationship of space and time [CM20, CM21, CM25, Wil25, Sha25]. These works used the aforementioned framework to repeatedly compute subroutines without a commensurate blowup in space, thus validating a key motivation for the introduction of catalytic space in the first place.

1.2 Robust catalytic computation

The ultimate goal of studying catalytic computation in the context of non-catalytic space is to utilize memory during a computation for both storage and other subcomputations. To that end, we may eject some of the restrictions of fully general catalytic computing by considering the problem and stored data at hand. For example, Bisoyi et al. [BDRS25] study the question of resetting the catalytic tape only when it contains some worthwhile information, or alternatively of ending with the tape in a string which shares some property of interest with its initial configuration.

Gupta et al. [GJST24] took a different approach, asking whether the model is *robust* to allowing a small number of errors when resetting the catalytic tape, a variant they called *lossy catalytic computing*. This latter question was answered by Folkertsma et al. [FMST25] when the number of errors is bounded for any run of the machine, showing that e errors on a catalytic tape of length c is equivalent in power to having $\Theta(e \log c)$ bits of additional free memory.

The more general question of failing to reset the catalytic tape, such as having a bounded number of errors in *expectation* over a generic run of the machine, were posed by Mertz [Mer23] and remain open. Understanding the more rich landscape of potential robustness conditions on the catalytic tape will allow us to understand when memory can and cannot be reused outside of the strict catalytic context, where averaging arguments and other structural results are known.

1.3 Our results

In this paper we take a systematic approach to understand lossy catalytic computation. To do so we introduce a host of different lossy catalytic models and show how the resulting complexity classes group around one another and around established classes. Our three variants are

1. $\mathsf{BP}^{\epsilon}\mathsf{C}^{\delta}\mathsf{SPACE}(s,c)$: randomized catalytic computation which, in addition to succeeding with probability $\frac{1}{2}+\epsilon$, has a probability δ to *arbitrarily fail* to reset the catalytic tape, and which faithfully resets the catalytic tape otherwise

2. $\mathbb{E}_{\mathsf{Rand}}\mathsf{BPLCSPACE}(s,c,e)$: randomized catalytic computation which makes at most e resetting errors *in expectation over the randomness*

3. $\mathbb{E}_{\mathsf{Cat}}\mathsf{LCSPACE}(s,c,e)$: *deterministic* catalytic computation which makes at most e resetting errors *in expectation over the initial contents of the catalytic tape itself*

Here s denotes the amount of work space, c denotes the amount of catalytic space and e the number of bit errors one can make when restoring the catalytic tape.
We focus on the *logspace* versions of these classes, which we denote the same way as CL; therein we can also study the *polynomial-time* bounded variants of the catalytic logspace classes, which we denote by appending P. For example $\mathsf{BP}^{\epsilon}\mathsf{C}^{\delta}\mathsf{LP}$ is the class $\mathsf{BP}^{\epsilon}\mathsf{C}^{\delta}\mathsf{SPACE}(O(\log n), n^{O(1)})$ where machines are restricted to run always in polynomial time.

The classes $\mathsf{BP}^{\epsilon}\mathsf{C}^{\delta}\mathsf{SPACE}(s,c)$ and $\mathbb{E}_{\mathsf{Rand}}\mathsf{BPLCSPACE}(s,c,e)$ capture two seemingly incomparable notions of randomized error. The former makes $\Omega(c)$ errors in expectation but does so in a highly structured way, whereas the latter has more leeway about how the errors appear but only permits e (typically $O(1)$) on average. Our third class $\mathbb{E}_{\mathsf{Cat}}\mathsf{LCSPACE}(s,c,e)$ appears to be much different, lacking randomness per se but being allowed to behave in a less careful way when the catalytic tape is "far from random".

We show a near full characterization of the above models. Focusing on the logspace setting we obtain the following:

- we show three regimes for $\mathsf{BP}^{\epsilon}\mathsf{C}^{\delta}\mathsf{L}$: 1) $\mathsf{BP}^{\epsilon}\mathsf{C}^{\delta}\mathsf{L}_{\mathsf{high}}$ when $\delta > 2\epsilon$; 2) $\mathsf{BP}^{\epsilon}\mathsf{C}^{\delta}\mathsf{L}_{\mathsf{low}}$ when $\delta < 2\epsilon < 1$; and 3) $\mathsf{BP}^{\frac{1}{2}}\mathsf{C}^{\delta}\mathsf{L}$, i.e. $\delta < 2\epsilon = 1$. While the third is an intermediate class with no clear connections, the first can capture the full power of PSPACE, while the second is equivalent to the $\mathbb{E}_{\mathsf{Rand}}\mathsf{BPLCL}[e]$ model for $e = O(1)$ (up to some parameter slackness).

- with the additional poly-time restriction, the three aforementioned regimes of $\mathsf{BP}^{\epsilon}\mathsf{C}^{\delta}\mathsf{LP}$ correspond exactly to BPP, CL (and $\mathbb{E}_{\mathsf{Rand}}\mathsf{BPLCL}[O(1)]\mathsf{P}$), and poly-time CL respectively.

- $\mathbb{E}_{\mathsf{Cat}}\mathsf{LCL}[e]$ captures (seemingly) more power than $\mathbb{E}_{\mathsf{Rand}}\mathsf{BPLCL}[e]$ when no time bound is imposed; we show that it exactly corresponds to (zero-error) randomized catalytic computing with *two-way access* to its randomness, a class which, if shown to be equal to CL, would imply major non-catalytic derandomizations as well.

- with the additional poly-time restriction, $\mathbb{E}_{\mathsf{Cat}}\mathsf{LCL}[e]\mathsf{P}$ and $\mathbb{E}_{\mathsf{Rand}}\mathsf{BPLCL}[e]\mathsf{P}$ become incomparable, with the former being in P for all e and capturing the intersection of $\mathbb{E}_{\mathsf{Cat}}\mathsf{LCL}[e]$ and P in the case of $e = O(1)$ (note that $\mathbb{E}_{\mathsf{Rand}}\mathsf{BPLCL}[O(1)]\mathsf{P} = \mathsf{CL}$ is not known to be in P)

- under a standard derandomization assumption about space (Conjecture 43), all of the aforementioned models collapse to CL and even poly-time CL (besides $\mathsf{BP}^{\epsilon}\mathsf{C}^{\delta}\mathsf{L}_{\mathsf{high}} = \mathsf{PSPACE}$ and $\mathsf{BP}^{\epsilon}\mathsf{C}^{\delta}\mathsf{LP}_{\mathsf{high}} = \mathsf{BPP} = \mathsf{P}$).

See Figures 1 and 2 for a summary of our new classes and subsequent results, respectively.

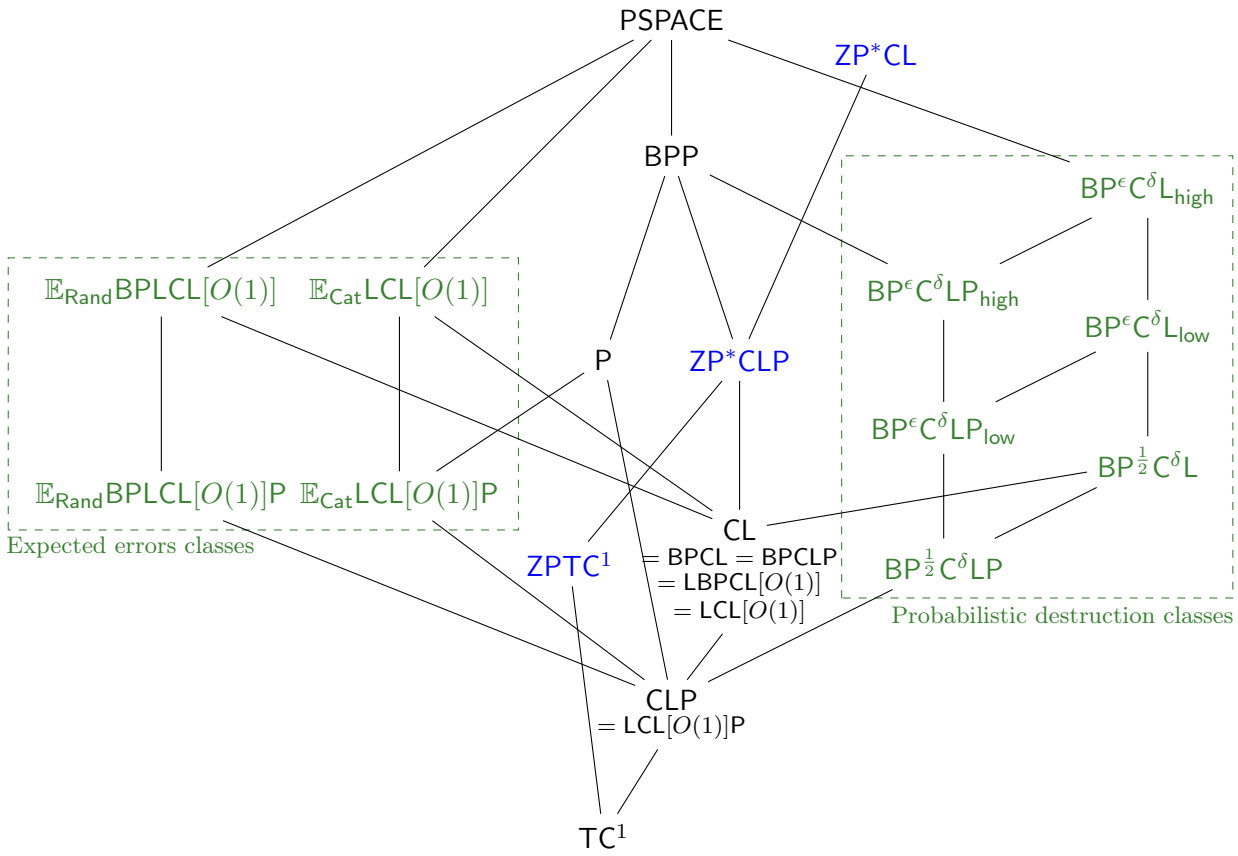

Figure 1: Basic class structure for logspace catalytic classes. Color coding: black = old classes; green + rectangle = robust catalytic classes introduced in this work; blue = reference classes which were implicitly previously defined but which have not been previously studied. Upwards lines indicate known or definitional inclusions.

1.4 Open questions

The main gap in our results is to show that $\mathsf{BP}^{\epsilon}\mathsf{C}^{\delta}\mathsf{L}_{\mathsf{low}}$ and $\mathbb{E}_{\mathsf{Rand}}\mathsf{BPLCL}[O(1)]$ both collapse to CL; this would seem most doable for $\mathsf{BP}^{\frac{1}{2}}\mathsf{C}^{\delta}\mathsf{L}$ as a warm-up. The only other stray class is $\mathbb{E}_{\mathsf{Cat}}\mathsf{LCL}[O(1)]\mathsf{P}$, whose complexity remains a mystery which would be useful to solve but may require new ideas in catalytic computing.

Our results seem to capture all remaining natural models of resetting error. One could always take further combinations of the aforementioned relaxations, such as allowing expected error over both the initial tape τ as well as randomness, but we expect these to collapse to one of our clusters without much difficulty. Alternatively, one could use this framework to study *average-case catalytic computation*, i.e. where the resetting error is in expectation over the *input*; as a setting this seems orthogonal to catalytic robustness, but perhaps similar techniques can be of use.

1.5 Our techniques

Because we have many results related to a host of different catalytic classes, there are various techniques at play. However, a key throughline through many of our proofs is an analysis of the *configuration graph* of our robust catalytic machines. Thus we will briefly discuss the common factors

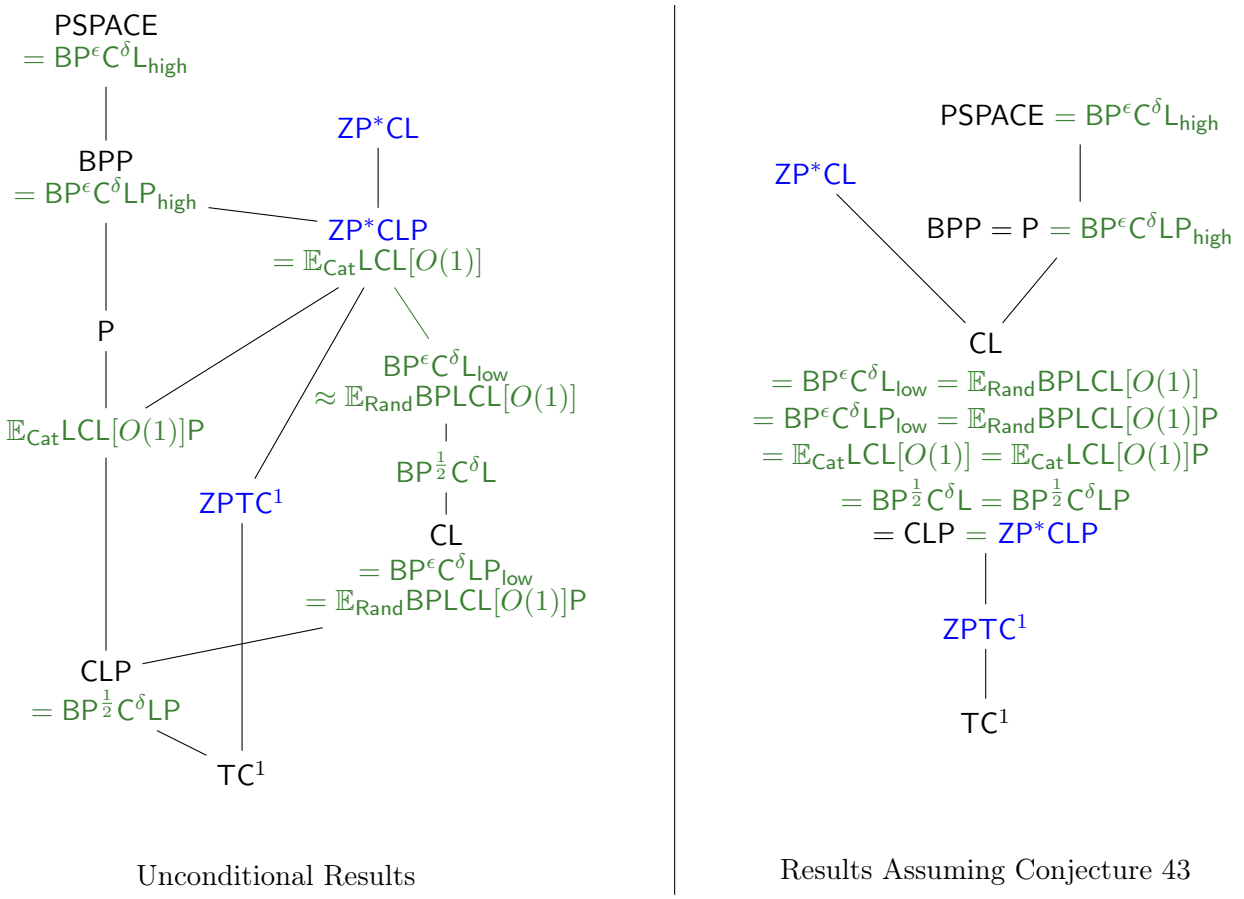

Figure 2: Summary of results (all green equalities and containments). In the unconditional setting, all new classes are either 1) equal to a preexisting class; 2) find themselves in two new clusters (note that $\approx$ means the parameters are not tight enough for full equality); or 3) are one of two stray classes. In the conditional setting, almost all new classes collapse to CL and all classes are comparable with the sole exception of $\mathsf{ZP^*CL}$.

of these results at a high level.

Configuration graphs. Configuration graphs of catalytic machines are one of the key tools in understanding catalytic machines, underlying results such as $\mathsf{CL} = \mathsf{CBPL}$ [CLMP25], $\mathsf{CL} = \mathsf{CNL}$ [KMPS25] and $\mathsf{CL} \cap \mathsf{P} = \mathsf{CLP}$ [CLMP25]. The collapses to CL of both CBPL [CLMP25, KMPS25] and CNL [KMPS25] are facilitated by efficient procedures that explore and analyze the configuration graph on a given input. The configuration graph of a catalytic machine that uses work space s and catalytic space c on a given input x might have up-to 2^{s+c} configurations, one for each $\langle \pi, u \rangle$ where the machine has π on its catalytic tape and u on its work tape.[1] When $s = O(\log n)$ and $c = n^{O(1)}$, the graph is exponentially large, which presents a challenge to analyze it.

For a machine M and input x, we denote the resulting configuration graph by $G_{M,x}$. For a given initial setting τ of the catalytic tape, we let $G_{M,x,\tau}$ be the induced subgraph of $G_{M,x}$ on configurations reachable from the starting configuration $\mathrm{start}_{M,x,\tau}$. The computation of any catalytic machine is always required to reach a final configuration with τ back on the catalytic tape. Without

[1] Here we assume that the head positions and internal state are automatically recorded as part of the work space.

loss of generality, we can assume that M has two final configurations, $\mathrm{acc}_{M,x,\tau}$ and $\mathrm{rej}_{M,x,\tau}$, which are easy to recognize; these configurations are sinks of $G_{M,x,\tau}$.

For deterministic machines M, it follows that $G_{M,x,\tau}$ and $G_{M,x,\tau'}$ are disjoint for distinct τ and τ'. Thus, on average (over τ), $G_{M,x,\tau}$ is of size at most $2^s = \mathrm{poly}(n)$. This fact is heavily used in many previous results on catalytic computation. Most important for our algorithms is that any τ that has a much larger size is far from average, i.e., it cannot be very random.

Using this logic, Cook et al. [CLMP25] coined a technique called *compress-or-random*. Their approach is to analyze $G_{M,x,\tau}$ and compress τ if the algorithm detects that $G_{M,x,\tau}$ has size $\gg 2^s$, and then restart the analysis with a new section of the catalytic tape. This eventually leads to either finding a τ that has a small configuration graph $G_{M,x,\tau}$, or freeing up so much space on the catalytic tape that we can run an ordinary space-unbounded algorithm for our original problem. Crucially, we are required to design efficient compression and decompression methods for τ in order to carry out this argument.

In our setting, the configuration graphs are more complicated. Since the machine does not always have to restore the content of the catalytic tape, two different configuration graphs $G_{M,x,\tau}$ and $G_{M,x,\tau'}$ might intersect for $\tau \neq \tau'$. This presents a challenge for analyzing $G_{M,x}$ as the above averaging argument does not work.

Compression for lossy configuration graphs. Our key technical lemmas Lemma 23a) and Lemma 23b) address this challenge. For a randomized machine M, we call a configuration of $G_{M,x,\tau}$ a τ^β-node if, from that configuration, we reset τ correctly with a probability of at least β. Clearly, the same configuration can be a τ^β-node for at most $1/\beta$ distinct values of τ; if the machine M restores the catalytic tape with a probability of at least $1-\delta$, then the probability of reaching a node that is *not* a τ^β-node is at most $\delta/(1-\beta)$. Thus, for $\beta < 1/2$, if the randomized machine resets correctly the catalytic tape with high probability, then it must be visiting only τ^β-nodes with high probability. Therefore, to understand the behavior of M on x, we only need to analyze the subgraph of $G_{M,x,\tau}$ consisting of τ^β-nodes, and so our goal while exploring $G_{M,x,\tau}$ will be to identify all τ^β-nodes, of which we expect to have at most $2^s/\beta$, and analyze the subgraph consisting of them.

As observed in [KMPS25], it is beneficial to explore $G_{M,x,\tau}$ *backwards*, starting from the final configurations $\mathrm{acc}_{M,x,\tau}$ and $\mathrm{rej}_{M,x,\tau}$; this allows us to explore the graph while always ensuring that we can return to the correct $\mathrm{acc}_{M,x,\tau}$ or $\mathrm{rej}_{M,x,\tau}$. Since we do not have space to store the explored portion of the graph, and we cannot use randomness for the exploration, as that could lead us to a place where we cannot reset, we will explore the graph using a *pseudo-random generator*. We will use a modification of Nisan's pseudo-random generator [Nis92] based on hash functions, where we will use as many hash functions in a sequence as we are generating bits. The sequence of hash functions will be taken from a portion of the catalytic tape, and we will use the compression technique of [Pyn25] to select good hash functions.

We will use the bits from the generator to guide us backwards from the final nodes $\mathrm{acc}_{M,x,\tau}$ and $\mathrm{rej}_{M,x,\tau}$, adding more hash functions to extend our horizon. Keeping the same hash functions for the later portion of the walk is useful because it fixes the part of the graph we have already explored. The pseudo-random generator also allows us to estimate well the probability of reaching either $\mathrm{acc}_{M,x,\tau}$ or $\mathrm{rej}_{M,x,\tau}$, hence determining whether a node is a τ^β-node or not.

Our ultimate goal is to find the node $\mathrm{start}_{M,x,\tau}$, classify it as a τ^β-node, and estimate the probability of reaching $\mathrm{acc}_{M,x,\tau}$. As we extend the horizon, we will always check how many nodes we see in total using the pseudo-random walks. If we encounter too many nodes, we will be able to use the large graph compression of [CLMP25] to compress our current catalytic tape and restart the exploration. Similarly, if we see sufficiently many τ^β-nodes, we will also be able to compress the

catalytic tape.

The only issue that will arise is if the start node $\text{start}_{M,x,\tau}$ is very far from the final nodes on average. In that case, we will run out of bits produced by our pseudo-random generator, and yet we will not be able to compress to the best of our knowledge. In Lemma 23b), we avoid this case altogether as the computation time is bounded by some fixed polynomial, but in the general case, we will give up, as this will be a rare occurrence over the choice of τ; this will determine the equivalences of our various error-prone catalytic classes.

For our conditional results, we can overcome the issue of such τ's by using strong pseudo-random generators and XORing τ with their output to obtain a more "typical" starting tape similar to the technique of [BKLS18]. Under a standard derandomization assumption, the Nisan-Wigderson pseudo-random generator is strong enough to guarantee that most of its output XOR-ed with τ avoids the aforementioned bad case.

2 Preliminaries

Let $[n] = \{1, \ldots, n\}$. For a graph G, we denote its vertex set by $V(G)$. Unless stated otherwise, by the size of a graph G we mean $|V(G)|$. We denote the length of a string y by $|y|$. The notation $x \circ y$ stands for the concatenation of the strings x and y. For a string $y \in \{0,1\}^n$, we write y_i for the i-th bit of y (where $i \in [n]$), $y_{\leq i}$ for the prefix of y consisting of its first i bits, and $y_{\geq i}$ for the suffix of y starting at y_i.

2.1 Catalytic computation

For the remainder of the paper, we will define functions $s \coloneqq s(n)$, $c \coloneqq c(n)$, and sometimes $e \coloneqq e(n)$, such that all are non-decreasing logspace-constructible functions satisfying 1) $s \geq \log n$; 2) $s \leq c \leq 2^s$; and 3) $e \leq c$. Note that putting all the above bounds together implies that all functions can be computed in space $O(s)$.

Our paper concerns the *catalytic computing* model, as defined by Buhrman et al. [BCK+14]:

Definition 1 (Catalytic computation)**.** *A* catalytic *Turing machine is a machine M with four tapes: 1) a read-only input tape; 2) a write-only output tape; 3) a read-write work tape; and 4) a read-write* catalytic *tape. M has the property that for any initializations x to the input tape and τ to the catalytic tape, M halts with τ in the catalytic tape.*

Such catalytic machines naturally give rise to complexity classes based on the available resource parameters.

Definition 2. *We define $\mathsf{CSPACE}(s, c)$ to be the class of languages L that can be decided by a catalytic machine with s bits of work space and c bits of catalytic space on inputs of length n, i.e., on input $x \in \{0,1\}^n$ and initial catalytic tape $\tau \in \{0,1\}^c$, M halts with $L(x)$ in the output tape and τ in the catalytic tape.*

The most important instantiation of catalytic space class is *catalytic logspace*. In this context we can also talk about an additional *polynomial time* restriction.

Definition 3 (Catalytic logspace (CL) and polytime CL)**.** *We define the class* catalytic logspace *as*

$$\mathsf{CL} \coloneqq \bigcup_{k \in \mathbb{N}} \mathsf{CSPACE}(k \log n, n^k)$$

We will also use the suffix P *to impose an additional constraint to run in polynomial time, i.e.*

$$\mathsf{CLP} \coloneqq \bigcup_{k\in\mathbb{N}} \mathsf{CTIMESPACE}(n^k, k\log n, n^k)$$

where $\mathsf{CTIMESPACE}(t, s, c)$ *is defined as* $\mathsf{CSPACE}(s, c)$ *machines which always halt in time* t*.*

An important relaxation to the base catalytic model is allowing additional access to *randomness*.

Definition 4 (Randomized catalytic computation)**.** *We define* $\mathsf{BPCSPACE}(s, c)$ *to be the class of languages* L *for which there exists a randomized catalytic machine* M *with* s *bits of work space,* c *bits of catalytic space, and access to* 2^c *random bits in a read-only read-once fashion, such that* $M(x) = L(x)$ *with probability* $\geq \frac{2}{3}$ *over the choice of the randomness of* M*.*

The focus of this paper will be on *lossy* catalytic computing, another relaxation defined by Gupta et al. [GJST24] where we may make errors on the catalytic tape.

Definition 5 (Lossy catalytic computation)**.** *A* lossy catalytic Turing machine *with* e *errors is a Turing machine* M *with the same four tapes as a catalytic Turing machine, but where on any input* x *and initial catalytic tape* τ*, the machine must halt with some* τ' *in the catalytic tape such that* τ *and* τ' *differ in at most* e *locations.*

Let $n \in \mathbb{N}$ *and let* $s \coloneqq s(n)$*,* $c \coloneqq c(n)$*,* $e \coloneqq e(n)$*. We say a language is in* $\mathsf{LCSPACE}(s, c, e)$ *if it can be decided by a lossy catalytic Turing machine with free space* s*, catalytic space* c*, and making at most* e *errors on inputs of length* n*.*

We also define BPCL, BPCLP, $\mathsf{LCL}[e]$, and $\mathsf{LCL}[e]\mathsf{P}$ analogously to CL, as well as $\mathsf{BPLCSPACE}(s, c, e)$ and $\mathsf{LBPCL}[e]$ as the intersection of both restrictions. In this context, we have a number of collapses to CL and CLP by connecting results of Cook et al. [CLMP25] and Koucký et al. [KMPS25] for CBPL with those of Folkertsma et al. [FMST25] for $\mathsf{LCL}[e]$ (reflected in Figure 1).

Lemma 6 ([CLMP25, KMPS25, FMST25])**.**

$$\mathsf{CL} = \mathsf{BPCL} = \mathsf{BPCLP} = \mathsf{LCL}[O(1)] = \mathsf{LBPCL}[O(1)] = \mathsf{LBPCL}[O(1)]\mathsf{P}$$

$$\mathsf{CLP} = \mathsf{LCL}[O(1)]\mathsf{P}$$

In the case of lossy catalytic computation, we will use a slightly more flexible fact due to Folkertsma et al. [FMST25], which in fact implies the corresponding results in Lemma 6:

Lemma 7. *If* $e \leq c^{1-\Omega(1)}$*, then*

$$\mathsf{LCSPACE}(\Theta(s), \Theta(c), e) = \mathsf{CSPACE}(\Theta(s + e\log c), \Theta(c))$$

and analogous results hold for $\mathsf{BPCSPACE}(s, c)$*,* $\mathsf{CSPACE}(s, c)\mathsf{P}$*, and* $\mathsf{BPCSPACE}(s, c)\mathsf{P}$*.*

Our paper focuses on the case of two-sided randomness, i.e., BPCL, but we also define catalytic computing with *zero-error randomness*, including a non-standard extension to the *two-way randomness access* setting. The latter definition has implicitly existed in the literature but has seen no study thus far; however, while these classes are not on their face relevant to robust catalytic computing, they find their way into our characterizations.

Definition 8. *We define* $\mathsf{ZPCSPACE}(s, c)$ *to be the same as* $\mathsf{BPCSPACE}(s, c)$ *but where the machine outputs* $L(x)$ *with probability* $\geq \frac{1}{2}$ *and otherwise outputs a special symbol* $\perp$*; in particular, it never outputs* $\overline{L(x)}$*.*

We define $\mathsf{ZP}^*\mathsf{CSPACE}(s, c)$ *to be the same as* $\mathsf{ZPCSPACE}(s, c)$ *but with* two-way *access to its randomness. We restrict the amount of randomness to* 2^c *bits and require that our machine always halts.*

2.2 Configuration graphs

As in most models of space-bounded computation, we view catalytic space through the syntactic characterization of the *configuration graph* of a machine M.

Definition 9. *Let M be any catalytic machine with work space s and catalytic space c. We denote by $\langle \pi, u \rangle$ the configuration of M where $\pi \in \{0,1\}^c$ is contained on the catalytic tape and $u \in \{0,1\}^s$ is on its work tape.*

Here, we assumed without loss of generality that all auxiliary information about the current configuration of M, i.e., the state of M's internal DFA, and the current positions of the tape heads for the input, work, and catalytic tapes are all automatically recorded in a designated part of the work tape.[2]

Consider the execution of M on some fixed input x and initial catalytic tape contents τ. Each configuration of 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 $\text{start}_{M,x,\tau} := \langle \tau, 0^s \rangle$ to be the start configuration, $\text{acc}_{M,x,\tau} := \langle \tau, 1 \circ 1 \circ 1 \circ 0^{s-3} \rangle$ to be the unique accepting halt configuration, and $\text{rej}_{M,x,\tau} := \langle \tau, 1 \circ 1 \circ 0 \circ 0^{s-3} \rangle$ to be the unique rejecting halt configuration. In the case that M is a zero-error machine (see Definition 8), we also define the unique don't-know halt configuration as $\bot_{M,x,\tau} := \langle \tau, 1 \circ 0 \circ 0^{s-2} \rangle$. Thus, starting and halting configurations can be easily recognized. It will often be useful to discuss the configuration graph defined by such executions.

Definition 10 (Configuration graphs)**.** *The configuration graph $G_{M,x}$ is a directed acyclic graph where each node corresponds to a configuration of the machine M on input x. There is a directed edge from $\langle \pi, u \rangle$ to $\langle \pi', u' \rangle$ if and only if $\langle \pi', u' \rangle$ can be reached from $\langle \pi, u \rangle$ in one execution step of M. If M is a probabilistic machine, the out-degree of every vertex in $G_{M,x}$ is at most 2; if M is deterministic, the out-degree is at most 1. A halting configuration has no outgoing edges in $G_{M,x}$. Furthermore, there exists a fixed constant d_M, which depends solely on M, such that each vertex in $G_{M,x}$ has at most d_M incident edges.*

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

For every catalytic tape τ, let $G_{M,x,\tau}$ represent the subgraph of $G_{M,x}$ that is induced by the configurations reachable from the starting configuration $\text{start}_{M,x,\tau}$. It is clear that $G_{M,x,\tau}$ contains one source node, which is $\text{start}_{M,x,\tau}$, and up to two sink nodes, namely $\text{acc}_{M,x,\tau}$ and $\text{rej}_{M,x,\tau}$. If M is a zero-error machine, there can be a third sink node, $\bot_{M,x,\tau}$. Observe that $G_{M,x,\tau}$ forms a path when M is a deterministic machine.

2.2.1 Traversal of configuration graphs

Let M be a deterministic catalytic machine, and $v \in V(G_{M,x,\tau})$, for some τ. Then, the connected (in the undirected sense) component of $G_{M,x}$ that contains v is a tree which we will denote by $G_{M,x}(v)$. This tree has a unique sink: the halting configuration in which M halts when run from v (or from $\text{start}_{M,x,\tau}$). This tree can be traversed, and vertices output in a pre-order fashion, where we think of the traversal as cyclic (upon completing a traversal, we start a new one). Then, the following procedures give us a way to traverse $G_{M,x}(v)$:

[2] 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.

Lemma 11. *Let M be a deterministic catalytic machine that utilizes s bits of work space and c bits of catalytic space. We have the following catalytic subroutines, which use extra $O(s)$ bits of work space*

- ***Next**. Given $\langle \pi, u \rangle$ on catalytic tape and work tape, this subroutine replaces $\langle \pi, u \rangle$ with $\langle \pi', u' \rangle$, which is the next vertex in (cyclic) pre-order traversal of $G_{M,x}(\langle \pi, u \rangle)$.*
- ***StepBack**. Given $\langle \pi, u \rangle$ on catalytic tape and work tape, this subroutine replaces $\langle \pi, u \rangle$ with $\langle \pi', u' \rangle$, which is the previous vertex in pre-order traversal of $G_{M,x}(\langle \pi, u \rangle)$.*
- ***Size**. Given $\langle \pi, u \rangle$ on catalytic tape and work tape, and an integer $S \leq 2^{O(s)}$, this subroutine checks whether $G_{M,x}(\langle \pi, u \rangle)$ is of size at most S. It returns the content of $\langle \pi, u \rangle$ unchanged.*

*The procedure **Size** runs in time $2^{O(s)}$, and the procedures **Next** and **StepBack** run in time $2^{O(s)} \cdot D$ where D is the depth of $G_{M,x}(\langle \pi, u \rangle)$.*

We will usually use **Next** and **StepBack** only when $G_{M,x}(\langle \pi, u \rangle)$ is of depth $2^{O(s)}$. In that case, both procedures take time $2^{O(s)}$.

Our subroutines can be easily obtained from those in [KMPS25]. Similar procedures in [KMPS25] take an additional parameter $h \in \{1, \ldots, d_M\}$ which is a label of an incident edge. For both of our procedures **Next** and **StepBack**, we can run their procedures repeatedly, starting with an edge label 1, and wait until we get into another vertex with the edge label 1. When 1 corresponds to the incoming edge in the directed version of $G_{M,x}(\langle \pi, u \rangle)$, the process will list configurations in a pre-order traversal, and the number of steps we have to take to reach the next vertex in the pre-order fashion is at most the depth of the graph.

The **Size** procedure runs the Eulerian tour traversal [KMPS25] of $G_{M,x}(\langle \pi, u \rangle)$ starting from $\langle \pi, u \rangle$ for $6S$ steps. As $G_{M,x}(\langle \pi, u \rangle)$ always contains an easy to recognize unique final configuration of M if the tour does not visit it at least twice with an edge label 1 we know $\langle \pi, u \rangle$ is larger that S. Otherwise, we count the number of edges labeled 1 that we traverse between successive visits to the final configuration. This number corresponds to the size of $G_{M,x}(\langle \pi, u \rangle)$, so we compare it with S and decide on the answer. Using the reverse traversal of the Eulerian tour for $6S$ steps, we return to the initial $\langle \pi, u \rangle$.

Since the tour given by Next/StepBack follows a depth-first search (DFS) traversal, we will commonly refer to it as such.

We will need all of the above procedures to apply to graphs that are restrictions of configuration graphs of *randomized* machines. In those cases, a sequence y of bits will be specified, which determines the random choices of a machine M. We will make a leveled version of the graph (level i leading into level $i-1$). In level i, we will use the i-th bit of y from the end to resolve the random choice of M at that time step. The leveled graph will be restricted only to have levels $|y|, \ldots, 0$, and a node in this graph will be specified by a configuration of $G_{M,x}$ and a level index. If the bits of y can be accessed via some efficiently computable procedure, then **Next**, **StepBack**, and **Size** on such a graph can be guaranteed to behave as described above, where the space and time used need to account also for the space and time needed to calculate individual bits of y (keeping track of the current level is easy based on the direction of the traversed edge in the directed graph $G_{M,x}$). Typically, the space needed to calculate individual bits of y will be $O(s)$ and time $2^{O(s)}$. So, it will not affect the asymptotics of our space or running time. For a binary sequence y, we will refer to the traversal of such a graph by the three procedures as y-DFS traversal.

3 New Robust Catalytic Classes

Our chief object of study in this paper is a number of expansions of the basic $\mathsf{LCSPACE}(s, c, e)$ definition of lossy catalytic space. Thus we begin with all definitions of interest. Every machine in this section will use s bits of work space, c bits of catalytic space, and any randomized machines have access to a read-only read-once random string $r \in \{0,1\}^{2^c}$ (recall our conditions on s, c, and e from Section 2).

In our first model, a randomized catalytic machine may arbitrarily destroy its catalytic tape with some probability over the randomness, but must perfectly reset otherwise:

Definition 12. *Let $0 \leq \delta < 1$ and $0 < \epsilon \leq \frac{1}{2}$. A language L is in $\mathsf{BP}^\epsilon\mathsf{C}^\delta\mathsf{SPACE}(s, c)$ if there is a randomized machine M such that on any input $x \in \{0,1\}^n$ and initial setting of the catalytic tape $\tau \in \{0,1\}^c$, M always halts, outputs $L(x)$ with probability at least $\frac{1}{2} + \epsilon$, and resets the catalytic tape to τ with probability at least $\geq 1 - \delta$, where both probabilities are over the machine's randomness r.*

We further use the notation $\mathsf{BP}^\epsilon\mathsf{C}^\delta\mathsf{SPACE}(s, c)_{\mathsf{high}}$ to denote $\delta > 2\epsilon$ and $\mathsf{BP}^\epsilon\mathsf{C}^\delta\mathsf{SPACE}(s, c)_{\mathsf{low}}$ to denote $\delta < 2\epsilon$ (< 1).

Our second and third models have to do with a bounded expected number of errors on the catalytic tape. The first version does so in expectation over the randomness of the machine:

Definition 13. *A language L is in $\mathbb{E}_{\mathsf{Rand}}\mathsf{BPLCSPACE}(s, c, e)$ if there is a randomized machine M such that on any input $x \in \{0,1\}^n$ and initial setting of the catalytic tape $\tau \in \{0,1\}^c$, M outputs $L(x)$ with probability at least $\frac{2}{3}$ and resets the catalytic tape to τ' such that $\mathbb{E}_r[d(\tau, \tau')] \leq e$, i.e., we make at most e errors in expectation over the machine's randomness r.*

The last model again makes a bounded number of errors in expectation, but now the expectation is over the initial catalytic tape, with the machine acting deterministically:

Definition 14. *A language L is in $\mathbb{E}_{\mathsf{Cat}}\mathsf{LCSPACE}(s, c, e)$ if there is a deterministic machine M such that on any input $x \in \{0,1\}^n$ and initial setting of the catalytic tape $\tau \in \{0,1\}^c$, M outputs $L(x)$ and resets the catalytic tape to τ' such that $\mathbb{E}_\tau[d(\tau, \tau')] \leq e$, i.e., we make at most e errors in expectation over the initial configuration τ.*

In this paper we focus on the logspace variant of all the above classes, both with and without the polynomial-time restriction, and we use notation in accordance with all previous classes.

A note on the remainder of the paper: while some results will follow from short proofs and/or minor tweaks to ones which appear earlier, for the sake of highlighting where collapses occur, we will use Theorem or Corollary exactly for those results which relate complexity classes together.

4 Model 1: $\mathsf{BP}^\epsilon\mathsf{C}^\delta\mathsf{SPACE}(s, c)$

We first address the case of $\mathsf{BP}^\epsilon\mathsf{C}^\delta\mathsf{SPACE}(s, c)$, which we stratify into the *high error case* when $\delta > 2\epsilon$ (i.e. $\mathsf{BP}^\epsilon\mathsf{C}^\delta\mathsf{L}_{\mathsf{high}}$), the *low error case* when $\delta < 2\epsilon$ (i.e. $\mathsf{BP}^\epsilon\mathsf{C}^\delta\mathsf{L}_{\mathsf{low}}$), and the *zero error case* when $\epsilon = \frac{1}{2}$ (i.e. $\mathsf{BP}^{\frac{1}{2}}\mathsf{C}^\delta\mathsf{L}$).

4.1 High error case: $\mathsf{BP}^\epsilon\mathsf{C}^\delta\mathsf{L}_{\mathsf{high}}$

In the high error regime, our chance of destroying the catalytic tape is greater than our advantage of finding the correct answer over random chance. We can exploit this fact to randomly choose whether to erase the tape and brute force the function or simply output a random answer.

Theorem 15. *Let δ, ϵ be constants such that $\delta > 2\epsilon$. Then*

$$\mathsf{BP}^{\epsilon}\mathsf{C}^{\delta}\mathsf{SPACE}(s,c) = \mathsf{SPACE}(\mathrm{poly}(s,c))$$

Proof. For the forward inclusion, $\mathsf{BP}^{\epsilon}\mathsf{C}^{\delta}\mathsf{SPACE}(s,c) \subseteq \mathsf{BPSPACE}(s+c) \subseteq \mathsf{SPACE}((s+c)^{3/2})$, where the first inclusion follows from using a "normal" read-write tape instead of the catalytic tape, and the second from the derandomization of bounded space due to Saks and Zhou [SZ99].

Now let $L \in \mathsf{SPACE}(S)$ and let N being the space-S machine deciding L. As dyadic rationals are dense in the reals [Rud76], there exists a rational number α such that $2\epsilon < \alpha < \delta$, and it has the form $\frac{p}{2^q}$ for $p, q \in \mathbb{N}$; since δ and ϵ are constants, α can be described with constantly many bits. Our $\mathsf{BP}^{\epsilon}\mathsf{C}^{\delta}\mathsf{SPACE}(\log S, S)$ machine M will first destroy the tape with probability α, and if it does then we run N in the catalytic memory and output the answer; if not, we flip a fair coin and output the answer.

M does not reset the catalytic tape with a probability of at most $\alpha < \delta$, while the probability that it correctly decides the input is

$$\frac{1}{2}(1-\alpha) + \alpha = \frac{1}{2} + \frac{\alpha}{2} \geq \frac{1}{2} + \epsilon$$

as required. □

Our main consequence will be for catalytic logspace, where we exactly capture PSPACE:

Corollary 16. *Let δ, ϵ be constants such that $\delta > 2\epsilon$. Then*

$$\mathsf{BP}^{\epsilon}\mathsf{C}^{\delta}\mathsf{L} = \mathsf{PSPACE}$$

The same logic applies when we have a polynomial time bound, with the only change being the power of the machine when we destroy the catalytic tape:

Theorem 17. *Let δ, ϵ be constants such that $\delta > 2\epsilon$. Then*

$$\mathsf{BP}^{\epsilon}\mathsf{C}^{\delta}\mathsf{LP} = \mathsf{BPP}$$

Proof. For the forward inclusion, $\mathsf{BP}^{\epsilon}\mathsf{C}^{\delta}\mathsf{LP} \subseteq \mathsf{BPTIME}(\mathrm{poly}(n))$ since the latter uses a "normal" read-write tape instead of the catalytic tape. Now let $L \in \mathsf{BPTIME}(T)$ and let N being the randomized time-T machine deciding L. Again choose α such that $2\epsilon < \alpha < \delta$ of the form $\frac{p}{2^q}$ for $p, q \in \mathbb{N}$. Our $\mathsf{BP}^{\epsilon}\mathsf{C}^{\delta}\mathsf{LP}$ machine M uses catalytic memory of size T and is identical to Theorem 15, with the note that we can run N in our catalytic memory in time $T = \mathrm{poly}(n)$ and space T. □

4.2 Low error case: $\mathsf{BP}^{\epsilon}\mathsf{C}^{\delta}\mathsf{L}_{\mathsf{low}}$

For the case of $\mathsf{BP}^{\epsilon}\mathsf{C}^{\delta}\mathsf{L}_{\mathsf{low}}$, our main result will be the following:

Theorem 18. *Let ϵ, δ be constants such that $\delta < 2\epsilon < 1$. Then*

$$\mathsf{BP}^{\epsilon}\mathsf{C}^{\delta}\mathsf{L}_{\mathsf{low}}\mathsf{P} = \mathsf{CL}$$

We will not discuss the general time-unbounded $\mathsf{BP}^{\epsilon}\mathsf{C}^{\delta}\mathsf{SPACE}(s,c)_{\mathsf{low}}$, but will return to this question in great detail in Section 5.

To prove Theorem 18, we will need to analyze the configuration graph of a $\mathsf{BP}^{\epsilon}\mathsf{C}^{\delta}\mathsf{L}_{\mathsf{low}}\mathsf{P}$ graph. Buhrman et al. [BCK+14] demonstrated that for a $\mathsf{CSPACE}(s,c)$ machine, the average size of the configuration graph over the initial catalytic tape is $2^{O(s)}$, but this relies on the fact that such a machine always resets the catalytic tape. In the case of a $\mathsf{BP}^{\epsilon}\mathsf{C}^{\delta}\mathsf{SPACE}(s,c)$ machine M, the relevant concept to consider is that of a τ^{β}-graph.

Definition 19 (τ^β-graph)**.** *Consider a* $\mathsf{BP}^\epsilon\mathsf{C}^\delta\mathsf{SPACE}(s,c)$ *machine* M *and its configuration graph* $G_{M,x,\tau}$*, for input* x *and initial catalytic tape* τ*. We define a configuration* v *in this graph to be a* τ^β*-node if the probability of reaching* $\text{acc}_{M,x,\tau}$ *or* $\text{rej}_{M,x,\tau}$ *from* v *is at least* β*. The* τ^β*-graph is then defined as follows:*

- *If* $\text{start}_{M,x,\tau}$ *is not a* τ^β*-node, we define the* τ^β*-graph to be empty. Otherwise, the vertex set consists of all the* τ^β*-nodes in* $G_{M,x,\tau}$ *that can be reached from* $\text{start}_{M,x,\tau}$ *via only* τ^β*-nodes.*
- *The transitions/edges are the same as* $G_{M,x,\tau}$*, except that all transitions going to a non-*τ^β*-node are routed to a (null)* $\perp$ *node.*

Let δ, ϵ be constants such that $\delta < 2\epsilon < 1$, and consider a suitable constant β, $0 < \beta < 1 - \delta$. Note that as M resets the catalytic tape correctly with probability $\geq 1 - \delta$, by Definition 19 and $\beta < 1 - \delta$, we have that $\text{start}_{M,x,\tau}$ is in the τ^β-graph. Since M always halts by definition, and given that $\beta > 0$, it follows from a simple averaging argument that at least one of $\text{acc}_{M,x,\tau}$ or $\text{rej}_{M,x,\tau}$ is in the vertex set of the τ^β-graph. Moreover, it follows from Definition 19 and $\beta > 0$ that the τ^β-graph does not contain $\text{acc}_{M,x,\tau'}$ or $\text{rej}_{M,x,\tau'}$ for $\tau \neq \tau'$.

Lemma 20. *Let* v *be a configuration in* $G_{M,x}$*, then there are at most* $\frac{1}{\beta}$ *different values for the initial catalytic setting* τ *such that* v *is in the* τ^β*-graph.*

Proof. If v is in the τ^β-graph for some τ, then by Definition 19 it is a τ^β-node and, with a probability of $\geq \beta$, reaches a halt state with catalytic contents τ. Since we assume $\beta > 0$, the lemma follows. □

Lemma 21. *Let the size of a* τ^β*-graph be defined as the number of vertices in it (excluding* $\perp$*). Then the average (over* τ*) size of a* τ^β*-graph is at most* 2^{4s}*.*

Proof. The number of bits required to describe any configuration is:

$$s + c + (\log n + \log c + \log s + O(1)) \leq 3s + c + 2\log s$$

which includes the bits required to store the internal state and the tape head locations. Since by Lemma 20, any configuration can be a part of a τ^β-graph for at most $\frac{1}{\beta}$ different values of τ, the average size is at most $\frac{1}{\beta} \cdot \frac{s^2 \cdot 2^{3s+c}}{2^c} \leq \frac{s^2 \cdot 2^{3s}}{\beta} \leq 2^{4s}$ (for large enough s); where we used the fact that β is a constant. □

Lemma 22. *The probability* q *of reaching* $\text{acc}_{M,x,\tau}$ *or* $\text{rej}_{M,x,\tau}$ *from* $\text{start}_{M,x,\tau}$ *within the* τ^β*-graph is at least* $1 - \frac{\delta}{(1-\beta)}$*. In other words, the probability of reaching* $\perp$ *from* $\text{start}_{M,x,\tau}$ *in the* τ^β*-graph is at most* $\frac{\delta}{(1-\beta)}$*.*

Proof. Firstly, note that by our assumption $0 \leq \delta < 1 - \beta < 1$, and thus the probabilities make sense. We have that

$$\begin{aligned} q &= 1 - \mathbf{Pr}[\text{Going from } \text{start}_{M,x,\tau} \text{ to } \perp \text{ in } \tau^\beta\text{-graph}] \\ &= 1 - \mathbf{Pr}[\text{Reaching a non-}\tau^\beta\text{-node from } \text{start}_{M,x,\tau} \text{ in } G_{M,x,\tau}] \end{aligned}$$

By the definition of a non-τ^β-node, we get

$$\begin{aligned} &(1-\beta) \cdot \mathbf{Pr}[\text{Reaching a non-}\tau^\beta\text{-node from } \text{start}_{M,x,\tau} \text{ in } G_{M,x,\tau}] \\ &\qquad \leq \mathbf{Pr}[\text{Not resetting catalytic tape to } \tau \text{ from } \text{start}_{M,x,\tau} \text{ in } G_{M,x,\tau}] \leq \delta \end{aligned}$$ □

If we take β to be a constant value such that $0 < \beta < 1 - \frac{\delta}{2\epsilon} \leq 1 - \delta$ (such a value exists as $\delta < 2\epsilon \leq 1$), then the aforementioned observations still hold. We set $\beta = \frac{1}{2}\left(1 - \frac{\delta}{2\epsilon}\right)$. Then, by Lemma 22, the probability of reaching $\perp$ from $\text{start}_{M,x,\tau}$ (in the τ^β-graph) is at most $\gamma = \frac{2\delta}{1+\frac{\delta}{2\epsilon}} < 2\epsilon$. In other words, the probability of exiting the τ^β-graph via $\perp$ is at most γ. When x is not in the language recognized by M, by definition, M accepts it with a probability of at most $\frac{1}{2} - \epsilon$. On the other hand, consider the case when x is in the language. Even if we restrict ourselves to the τ^β-graph, we know that the probability of going from $\text{start}_{M,x,\tau}$ to $\text{acc}_{M,x,\tau}$ is at least $\frac{1}{2} + \epsilon - \gamma$, which is a constant that is bounded away from $\frac{1}{2} - \epsilon$.

In summary, since the τ^β-graph has no halt configurations other than $\text{acc}_{M,x,\tau}$ and $\text{rej}_{M,x,\tau}$, the machine M behaves like a purely catalytic machine under this restriction. Moreover, even with this restriction, we can still decide the input. Although we do not yet know how to handle the case where the τ^β-graph is "large"—which would help us de-randomize the class $\mathsf{BP}^\epsilon\mathsf{C}^\delta\mathsf{SPACE}(s, c)$—we do know how to deal with the case when the graph is small. This is often the case, as indicated in Lemma 21.

When the τ^β-graph is small, we can either decide the input or compress the catalytic contents. This serves as the basis for the primary lemma in this work, and we defer the proof to Appendix B:

Lemma 23a). *Let ϵ, δ be constants such that $\delta < 2\epsilon$. Then, for every $L \in \mathsf{BP}^\epsilon\mathsf{C}^\delta\mathsf{SPACE}(s, c)$ there exist **deterministic** catalytic subroutines $\mathcal{F}^L$ and $\mathcal{R}^L$, which given input x behave as follows:*

1. *$\mathcal{F}^L$ and $\mathcal{R}^L$ use $O(s)$ work space, $2^{O(s)}$ catalytic space and run in time $2^{O(s)}$.*
2. *For initial catalytic contents τ, $\mathcal{F}^L(x, \tau)$ either outputs (don't-know) $\perp$ or correctly decides if x is in L, or changes the catalytic tape to $\tau' \circ 0$. In the first two cases, it does not change the contents of the catalytic tape.*
3. *For τ' from the previous step, $\mathcal{R}^L(x, \tau' \circ 0)$ resets the catalytic tape back to τ.*
4. *The fraction of initial catalytic contents τ, for which $\mathcal{F}^L$ outputs $\perp$ is at most $\frac{1}{2^{4s}}$.*

For $\delta = 0$, M always resets the catalytic tape. In this case, the proof of the lemma above provides a de-randomization of $\mathsf{BPCSPACE}(s, c)$, which is already known due to Cook et al. [CLMP25]; in this case, $\mathcal{F}^L$ never outputs $\perp$. This gives the following lemma, whose proof we again defer to Appendix B:

Lemma 23b). *Let ϵ, δ be constants such that $\delta < 2\epsilon$. Then, for every $L \in \mathsf{BP}^\epsilon\mathsf{C}^\delta\mathsf{LP}$, there exist **deterministic** catalytic subroutines $\mathcal{F}^L$ and $\mathcal{R}^L$ that behave exactly the same as those in Lemma 23a); except that now $\mathcal{F}^L$ never outputs $\perp$.*

Note that Lemma 23a) and Lemma 23b) hold trivially when $\delta = 0$ and $\epsilon = \frac{1}{2}$, as by definition $\mathsf{BP}^{\frac{1}{2}}\mathsf{C}^0\mathsf{SPACE}(s, c) = \mathsf{CSPACE}(s, c)$ and $\mathsf{BP}^{\frac{1}{2}}\mathsf{C}^0\mathsf{LP} = \mathsf{CLP}$.

We now have all the lemmas needed to prove our main result for $\mathsf{BP}^\epsilon\mathsf{C}^\delta\mathsf{L}_{\mathsf{low}}$.

Proof of Theorem 18. For the reverse direction, it is known that $\mathsf{CL} = \mathsf{ZPCLP}$ [CLMP25]. Given a ZPCLP machine, we can run it $\left\lceil \log \frac{1}{\frac{1}{2}-\epsilon} \right\rceil$ times, and we output any non-$\perp$ answer we see; if none exists, we output 0. Clearly, we always reset the catalytic tape and answer incorrectly with a probability of at most $\frac{1}{2} - \epsilon$.

Now we prove the forward direction. Let $L \in \mathsf{BP}^\epsilon\mathsf{C}^\delta\mathsf{LP}$, and let the corresponding machine be N which runs in time n^d (and hence uses catalytic space at most n^d) for $d \in \mathbb{N}$. Given input x, we run subroutine $\mathcal{F}^L$ from Lemma 23b) on n^{10d} different chunks of polynomial-sized catalytic tapes. If $\mathcal{F}^L$ correctly decides x for any chunk we output the result; otherwise, $\mathcal{F}^L$ frees up the last bit of each

chunk, and hence n^{10d} bits, on the catalytic tape. In this case, we do a brute force simulation of N in the freed-up space, using the fact that $N \in \mathsf{BPTIME}(s+c) \subseteq \mathsf{SPACE}(s+c)$. Finally, before outputting the result, we run $\mathcal{R}^L$ from Lemma 23b) to reset all the catalytic tape chunks. Our space bound follows because $\mathcal{F}^L$, $\mathcal{R}^L$, and N each use $O(\log n)$ work space. □

4.3 Zero error case: $\mathsf{BP}^{\frac{1}{2}}\mathsf{C}^{\delta}\mathsf{L}$

A $\mathsf{BP}^{\frac{1}{2}}\mathsf{C}^{\frac{1}{poly(n)}}\mathsf{L}$ machine always outputs the correct answer but has a non-negligible probability of destroying the catalytic tape; specifically, this probability is an inverse polynomial in the input length (for a sufficiently large polynomial). Currently, we do not know how to even show unconditionally that $\mathsf{CL} = \mathsf{BP}^{\frac{1}{2}}\mathsf{C}^{\frac{1}{poly(n)}}\mathsf{L}$. However, if the probability of not resetting the tape is exponentially small (for a sufficiently large exponential), then the class collapses to CL.

Observation 24. *For any constant $0 < \epsilon \leq \frac{1}{2}$,*

$$\cup_{d\in\mathbb{N}}\mathsf{BP}^{\epsilon}\mathsf{C}^{\frac{1}{2^{4n^d+3}}}\mathsf{SPACE}(d\log n, n^d) = \mathsf{CL}$$

Proof. The reverse direction follows by definition, while we defer the forward direction to Appendix C. □

Although for the case of $\mathsf{BP}^{\frac{1}{2}}\mathsf{C}^{\delta}\mathsf{L}$, we cannot say anything in the time-unrestricted case (it falls between CL and $\mathsf{BP}^{\frac{1}{2}}\mathsf{C}^{\delta}\mathsf{L}_{\mathsf{low}}$ by definition), in the polynomial-time world, we can show a loss of power even beyond $\mathsf{BP}^{\frac{1}{2}}\mathsf{C}^{\delta}\mathsf{LP}_{\mathsf{low}}$:

Theorem 25. *Let $\delta < 1$ be a constant. Then*

$$\mathsf{BP}^{\frac{1}{2}}\mathsf{C}^{\delta}\mathsf{LP} = \mathsf{CLP}$$

Proof. Clearly $\mathsf{BP}^{\frac{1}{2}}\mathsf{C}^{\delta}\mathsf{LP} \supseteq \mathsf{CLP}$ by definition. For the forward direction we closely follow the proof of Theorem 18. Let $L \in \mathsf{BP}^{\frac{1}{2}}\mathsf{C}^{\delta}\mathsf{LP}$, and let the corresponding machine be N. Given input x, we run subroutine $\mathcal{F}^L$ from Lemma 23b) on n^{10d} different chunks of polynomial-sized catalytic tapes, where $d \in \mathbb{N}$ is such that N runs in time n^d. If $\mathcal{F}^L$ correctly decides x for any chunk we output the result; otherwise, $\mathcal{F}^L$ frees up the last bit of each chunk, and hence n^{10d} bits, on the catalytic tape. In this case, we simulate N using the freed-up space on the all-zero random string. As N runs in polynomial time and always correctly decides x, we can also answer correctly in polynomial time. Finally, before outputting the result, we run $\mathcal{R}^L$ from Lemma 23b) to reset all the catalytic tape chunks. $\mathcal{F}^L$ and $\mathcal{R}^L$ use $O(\log n)$ work space and run in polynomial time, while N runs in polynomial time and uses the compressed space. □

4.4 Afterword: tradeoffs between ϵ and δ

It may seem strange that our choice of ϵ and δ is largely irrelevant as long as we understand where they fit in the above trichotomy. To close, we note that for the *one-sided error* regime, we can precisely scale both ϵ and δ in tandem, with a barrier to reaching $\epsilon = \frac{1}{2}$

Lemma 26. *Define $\mathsf{R}^{\epsilon}\mathsf{C}^{\delta}\mathsf{SPACE}(s,c)$ to be the class of languages L for which there exists a randomized catalytic machine M with s bits of work space, c bits of catalytic space, and access a string r of 2^c random bits in a read-only read-once fashion, such that 1) if $L(x) = 1$ then $M(x)$ outputs 1; 2) if $L(x) = 0$ then $M(x)$ outputs 0 with probability ϵ over r; 3) M resets the catalytic tape with probability $1-\delta$ over r.*

Then for all ϵ, δ,

$$\mathsf{R}^{\epsilon}\mathsf{C}^{\delta}\mathsf{SPACE}(s,c) \subseteq \mathsf{R}^{\epsilon/2}\mathsf{C}^{\delta/2}\mathsf{SPACE}(s,c)$$

$$\mathsf{R}^{\epsilon}\mathsf{C}^{\delta}\mathsf{SPACE}(s,c) \subseteq \mathsf{R}^{2\epsilon-\epsilon^2}\mathsf{C}^{2\delta-\delta^2}\mathsf{SPACE}(s,c)$$

Proof. Let M be a $\mathsf{R}^{\epsilon}\mathsf{C}^{\delta}\mathsf{SPACE}(s,c)$ machine. First, our $\mathsf{R}^{\epsilon/2}\mathsf{C}^{\delta/2}\mathsf{SPACE}(s,c)$ machine $M_{\frac{1}{2}}$ flips a fair coin, and if it returns heads we output 1, otherwise we run M. Clearly 1) remains fulfilled since we never output 0 when M outputs 1. Since with probability $\frac{1}{2}$ we output 1 and do not destroy the tape, for 2) we have $\mathbf{Pr}[M_{\frac{1}{2}}(x) = 0] = (\frac{1}{2}) \cdot \epsilon$, while for 3) our probability of destroying the tape is at most $(\frac{1}{2}) \cdot \delta$.

Second, our $\mathsf{R}^{2\epsilon-\epsilon^2}\mathsf{C}^{2\delta-\delta^2}\mathsf{SPACE}(s,c)$ machine M_2 runs M twice and outputs 1 if either run outputs 1, otherwise we output 0. Again 1) remains fulfilled by definition. For 2), given x such that $L(x) = 0$ we have

$$\mathbf{Pr}[M_2(x) = 0] = (1-\epsilon)^2 = 1 - 2\epsilon + \epsilon^2$$

while for 3) we have

$$\mathbf{Pr}[M_2(x) \text{ destroys } \tau] = \delta + (1-\delta)\delta = 2\delta - \delta^2$$

which completes the lemma. □

While the role of ϵ is different in $\mathsf{BP}^{\epsilon}\mathsf{C}^{\delta}\mathsf{SPACE}(s,c)$ and hence we do not have the same tradeoff, intuitively Lemma 26 shows that we have flexibility in choosing δ and ϵ but we cannot cross the boundary between $\delta < 2\epsilon$ and $\delta > 2\epsilon$, and that $\epsilon = 1$ is still not reachable in the latter case.

5 Model 2: $\mathbb{E}_{\mathsf{Rand}}\mathsf{BPLCSPACE}(s,c,e)$

In this section, we analyze the class $\mathbb{E}_{\mathsf{Rand}}\mathsf{BPLCSPACE}(s,c,e)$ and ultimately show a loose relationship to the class $\mathsf{BP}^{\epsilon}\mathsf{C}^{\delta}\mathsf{SPACE}(s,c)$.

Recall that an $\mathbb{E}_{\mathsf{Rand}}\mathsf{BPLCSPACE}(s,c,e)$ machine decides the input correctly with a probability of $\geq \frac{2}{3}$ and makes e errors on average on the catalytic tape over the random bits of the machine, and this holds for *every* input and initial catalytic setting. We can trivially boost the factor of $\frac{2}{3}$ to an arbitrary constant < 1, at the expense of worsening the work space and the parameter e by a constant factor:

Lemma 27. *Let $L \in \mathbb{E}_{\mathsf{Rand}}\mathsf{BPLCSPACE}(s,c,e)$. Then, for every constant $\frac{2}{3} \leq q < 1$, there exists an $\mathbb{E}_{\mathsf{Rand}}\mathsf{BPLCSPACE}(O(s),c,O(e))$ machine N, for L, such that for every input and initial catalytic tape N correctly decides the input with probability $\geq q$.*

Proof. Let M be the $\mathbb{E}_{\mathsf{Rand}}\mathsf{BPLCSPACE}(s,c,e)$ machine for L. Our machine N runs M $k = \lceil 300 \log \frac{1}{(1-q)} \rceil$ times independently, using the same s bits of free memory and c bits of catalytic tape, and takes the majority of the outputs. By a Chernoff bound, N incorrectly decides the input with a probability of $\leq 1 - q$, and by linearity of expectation and the triangle inequality, M makes at most $k \cdot e$ errors when resetting the catalytic tape. □

We will utilize the following error-correction scheme used by Folkertsma et al. [FMST25] in the context of lossy catalytic computation.

Lemma 28 (Folkertsma et al. [FMST25])**.** *There exists a compression scheme (Enc_{BCH}, Dec_{BCH}) such that:*

- ***Encoding:*** *Enc_{BCH} takes as input a string S of length c, plus an additional $(2e+1)\lceil\log(c+e)\rceil$ bits initialized to zero, and outputs a codeword S_{enc}:*

$$S + [0]^{(2e+1)\lceil\log(c+e)\rceil} \xrightarrow{Enc_{BCH}} S_{enc}.$$

Furthermore, all outputs S_{enc} generated this way have minimum distance $\delta := 2e+1$ from one another.

- ***Decoding:*** *Dec_{BCH} takes as input a string S'_{enc} of length*

$$c + (2e+1)\lceil\log(c+e)\rceil,$$

with the promise that there exists a string S of length c such that

$$Enc_{BCH}\big(S + [0]^{(2e+1)\lceil\log(c+e)\rceil}\big)$$

differs from S'_{enc} in at most $\delta/2 - 1 = e$ locations, and outputs this string S:

$$S'_{enc} \xrightarrow{Dec_{BCH}} S + [0]^{(2e+1)\lceil\log(c+e)\rceil}.$$

Furthermore, both Enc_{BCH} and Dec_{BCH} can be computed in space $O(e\log c)$.

5.1 Relationship to $\mathsf{BP}^{\epsilon}\mathsf{C}^{\delta}\mathsf{SPACE}(s,c)_{\mathsf{low}}$

We now establish a connection between $\mathbb{E}_{\mathsf{Rand}}\mathsf{BPLCSPACE}(s,c,e)$ and $\mathsf{BP}^{\epsilon}\mathsf{C}^{\delta}\mathsf{SPACE}(O(s+e\log c),c)_{\mathsf{low}}$. In one direction, we can use any δ and ϵ in the low error regime.

Theorem 29. *Let δ,ϵ be constants such that $\delta < 2\epsilon < 1$. Then*

$$\mathbb{E}_{\mathsf{Rand}}\mathsf{BPLCSPACE}(s,c,e) \subseteq \mathsf{BP}^{\epsilon}\mathsf{C}^{\delta}\mathsf{SPACE}(O(s+e\log c),c)$$

Proof. Let $L \in \mathbb{E}_{\mathsf{Rand}}\mathsf{BPLCSPACE}(s,c,e)$. By assumption, $\frac{1}{2}+\epsilon < 1$, and thus using Lemma 27, there exists a $\mathbb{E}_{\mathsf{Rand}}\mathsf{BPLCSPACE}(O(s),c,O(e))$ machine N for L that, for any input x and initial catalytic setting τ, decides the input correctly with probability $\geq \frac{1}{2}+\epsilon$. Moreover, N makes at most $e' = O(e)$ errors on the catalytic tape on average (where the average is over its random bits), for any τ. Therefore, by Markov's inequality, the probability that N makes more than $\frac{e'}{\delta}$ errors is $\leq \delta$.

We will simulate N by reusing the idea from [FMST25]. Using c bits of catalytic space and $O(\frac{e'}{\delta}\log c) = O(e\log c)$ work space, we run the encoding algorithm $\mathrm{Enc}_{\mathrm{BCH}}$ from Lemma 28. This gives us a codeword $\tau' \circ w$, where τ' is written on the catalytic tape, and w (which takes $O(\frac{e'}{\delta}\log c$ bits)) is stored in the work space. We simulate N for input x and catalytic contents τ' using an additional $O(s)$ work space (and read-once randomness). Before outputting the result, we run the algorithm $\mathrm{Dec}_{\mathrm{BCH}}$ from Lemma 28 on our final catalytic tape $\tau'' \circ w$, and then we output the result of the computation.

Since, with probability $\geq 1-\delta$, N makes $\leq \frac{e'}{\delta}$ errors; with probability $\geq 1-\delta$, the Hamming distance is $|\tau'' - \tau'| \leq \frac{e'}{\delta}$. Lemma 28 tells us that we can correct up to $\frac{e'}{\delta}$ errors. Thus, with a probability of $\geq 1-\delta$, running $\mathrm{Dec}_{\mathrm{BCH}}$ on $\tau'' \circ w$ resets the tape back to τ. As we use $O(s+e\log c)$ work space in total and c bits of catalytic space, the theorem follows. □

Corollary 30. *Let δ,ϵ be constants such that $\delta < 2\epsilon < 1$. Then*

$$\mathbb{E}_{\mathsf{Rand}}\mathsf{BPLCL}[O(1)] \subseteq \mathsf{BP}^{\epsilon}\mathsf{C}^{\delta}\mathsf{L}$$

For the reverse direction, we can only capture extremely small resetting errors δ, thus leading to a gap in our characterization but still establishing a connection.

Theorem 31. *For any constant $\epsilon < 1/2$,*

$$\mathsf{BP}^{\epsilon}\mathsf{C}^{\frac{1}{c}}\mathsf{SPACE}(s,c) \subseteq \mathbb{E}_{\mathsf{Rand}}\mathsf{BPLCSPACE}(O(s),c,O(1))$$

Proof. Let $L \in \mathsf{BP}^{\epsilon}\mathsf{C}^{\frac{1}{c}}\mathsf{SPACE}(s,c)$, and let N be the machine that decides the input correctly with probability $\frac{1}{2}+\epsilon$ using s bits of work space and c bits of catalytic space, such that N does not reset the catalytic tape with probability $\frac{1}{c}$. As in Lemma 27, we run N a constant number of times to increase the probability of deciding correctly to at least $\frac{2}{3}$, re-using the same c bits of catalytic space each time. Let's say this constant (which depends only on ϵ) is k_ϵ. By a union bound, the probability that we do not reset the catalytic tape is $\leq \frac{k_\epsilon}{c}$, and so on average over the read-once randomness we make at most $c \cdot \frac{k_\epsilon}{c} \leq k_\epsilon$ errors on the catalytic tape. □

Corollary 32. *Let $\epsilon < 1/2$ be a constant. Then,*

$$\mathsf{BP}^{\epsilon}\mathsf{C}^{\frac{1}{poly(n)}}\mathsf{L} \subseteq \mathbb{E}_{\mathsf{Rand}}\mathsf{BPLCL}[O(1)]$$

According to Corollary 32, the class $\mathsf{BP}^{\frac{1}{2}}\mathsf{C}^{\frac{1}{\mathrm{poly}(n)}}\mathsf{L}$ is contained within $\mathbb{E}_{\mathsf{Rand}}\mathsf{BPLCL}[O(1)]$, which means we currently do not have a method to de-randomize it. However, with the additional restriction of polynomial time, we can show that, similar to the class $\mathsf{BP}^{\epsilon}\mathsf{C}^{\delta}\mathsf{L}$ (as stated in Theorem 18), $\mathbb{E}_{\mathsf{Rand}}\mathsf{BPLCL}[O(1)]$ is indeed equivalent to CL.

Theorem 33. *Let δ, ϵ be constants such that $\delta < 2\epsilon < 1$. Then,*

$$\mathsf{CL} = \mathbb{E}_{\mathsf{Rand}}\mathsf{BPLCL}[O(1)]\mathsf{P} = \mathsf{BP}^{\epsilon}\mathsf{C}^{\delta}\mathsf{LP}$$

Proof. By Theorem 18 we have $\mathsf{CL} = \mathsf{BP}^{\epsilon}\mathsf{C}^{\delta}\mathsf{LP}$, while by [CLMP25] we have $\mathsf{CL} = \mathsf{BPCLP} \subseteq \mathbb{E}_{\mathsf{Rand}}\mathsf{BPLCL}[O(1)]\mathsf{P}$. Finally, $\mathbb{E}_{\mathsf{Rand}}\mathsf{BPLCL}[O(1)]\mathsf{P} \subseteq \mathsf{BP}^{\epsilon}\mathsf{C}^{\delta}\mathsf{LP}$ by the same proof as Theorem 29, as the proof only requires re-running the relevant machine a constant number of times. □

6 Model 3: $\mathbb{E}_{\mathsf{Cat}}\mathsf{LCSPACE}(s,c,e)$

In this section, we first explore the relationship between $\mathbb{E}_{\mathsf{Cat}}\mathsf{LCSPACE}(s,c,e)$ and other classes.

6.1 Relationship to randomized error

Our first set of results connect $\mathbb{E}_{\mathsf{Cat}}\mathsf{LCSPACE}(s,c,e)$ to $\mathsf{BP}^{\epsilon}\mathsf{C}^{\delta}\mathsf{SPACE}(s,c)$, and by extension to $\mathbb{E}_{\mathsf{Rand}}\mathsf{BPLCSPACE}(s,c,e)$. We specifically show that, in the low-error regime, $\mathsf{BP}^{\epsilon}\mathsf{C}^{\delta}\mathsf{L}_{\mathsf{low}}$ is a subset of $\mathbb{E}_{\mathsf{Cat}}\mathsf{LCL}[o(1)]$.

Theorem 34. *Let δ, ϵ be constants such that $\delta < 2\epsilon$. Then*

$$\mathsf{BP}^{\epsilon}\mathsf{C}^{\delta}\mathsf{SPACE}(s,c) \subseteq \mathbb{E}_{\mathsf{Cat}}\mathsf{LCSPACE}(O(s), 2^{O(s)}, o(1))$$

Proof. Let $L \in \mathsf{BP}^{\epsilon}\mathsf{C}^{\delta}\mathsf{SPACE}(s,c)$, and let N be the corresponding machine for L. Our machine M will use a catalytic tape consisting of $r \cdot t$ chunks, where $r = 2^{2s}$ and $t = s$. We divide the tape into t layers, each consisting of r chunks.

We invoke $\mathcal{F}^L$ from Lemma 23a) on all the chunks one by one, resetting each chunk after its call using $\mathcal{R}^L$ if it was compressed. If $\mathcal{F}^L$ correctly decides the input for any chunk we output the result.

Otherwise, assume there exists a layer where the $\mathcal{F}^L$ frees up the last bit of each chunk in that layer. Using $\mathcal{F}^L$ and $\mathcal{R}^L$, we figure out such a layer and free up the last bit of each chunk in the layer. Using the freed-up 2^{2s} bits, we run a space-inefficient algorithm to simulate N deterministically and find the answer, resetting all the chunks in the layer afterwards.

If no such layer exists, then every layer has at least one chunk where the subroutine outputted $\perp$. For a random catalytic tape this occurs with a probability of $\leq \left(\frac{r}{2^{4s}}\right)^t = \frac{1}{2^{2s^2}}$, and so we can destroy the catalytic tape and use a space-inefficient algorithm to find the correct answer. Thus, we always find the right answer, and the average number of errors we make over all initial catalytic tapes is

$$\leq \frac{t.r.2^{ds}}{2^{2s^2}} \leq \frac{s2^{(d+2)s}}{2^{2s^2}} = o(1)$$

Our catalytic usage is at most $rt = s \cdot 2^{2s}$, while we use at most $O(s)$ work space to run $\mathcal{F}^L$ and $\mathcal{R}^L$ by Lemma 23a). □

Corollary 35. *Let δ, ϵ be constants such that $\delta < 2\epsilon$. Then*

$$\mathsf{BP}^\epsilon\mathsf{C}^\delta\mathsf{L} \subseteq \mathbb{E}_{\mathsf{Cat}}\mathsf{LCL}[o(1)]$$

6.2 Equivalence to read-multiple randomness

We now show a connection between $\mathbb{E}_{\mathsf{Cat}}\mathsf{LCSPACE}(s, c, e)$ and read-multiple errorless randomized catalytic computing. To do this we first introduce another new catalytic class, which will be convenient for facilitating our results.

Definition 36. *A language L is in $\mathsf{ZP}_{\mathsf{Cat}}\mathsf{CSPACE}(s, c)$ if there is a deterministic catalytic machine M with free space s and catalytic space c which that always resets the catalytic tape, and outputs $L(x)$ with probability $\frac{1}{2}$ over the initial catalytic tape and $\perp$ otherwise.*

This is the same form of zero-error randomness as introduced in Definition 8, and in fact we will show that all these definitions coincide in short order. We begin with connecting $\mathsf{ZP}_{\mathsf{Cat}}\mathsf{CSPACE}(s, c)$ to $\mathbb{E}_{\mathsf{Cat}}\mathsf{LCSPACE}(s, c, e)$.

Theorem 37.

$$\mathbb{E}_{\mathsf{Cat}}\mathsf{LCSPACE}(s, c, e) \subseteq \mathsf{ZP}_{\mathsf{Cat}}\mathsf{CSPACE}(O(s + e \log c), c)$$

Proof. Let M be the $\mathbb{E}_{\mathsf{Cat}}\mathsf{LCSPACE}(s, c, e)$ (deterministic) machine, x the input, and τ the initial catalytic contents. Then by Markov's inequality, there are at most a $\frac{1}{10}$ fraction of initial catalytic tapes for which M makes more than $10e$ errors.

Using $O(e \log c)$ space, we enumerate all the $\sum_{i=0}^{10e} \binom{c}{i} \leq c^{10e+1}$ possibilities where $\leq 10e$ errors can be made on the catalytic tape. Let each possibility be represented by its characteristic error vector, denoted by z. For each z, we perform a DFS starting from the halt states $\text{acc}_{M,x,\tau\oplus z}$ and $\text{rej}_{M,x,\tau\oplus z}$, using the subroutine NEXT in Lemma 11, running until we return to the unique halt state from which we started. If we encounter *any* start state during the DFS from $\text{acc}_{M,x,\tau\oplus z}$ we accept, and likewise we reject if we encounter any start state during the DFS from $\text{rej}_{M,x,\tau\oplus z}$. If we never see any start state, we output $\perp$.

Each DFS can be performed using $O(s)$ work space and c bits of catalytic space, and hence we use $O(s + e \log c)$ work space and c catalytic space If τ is such that M makes at most $10e$ errors with tape contents τ, then some error vector z' would correspond to exactly where these errors are made. Therefore, $\text{start}_{M,x,\tau}$ will be encountered in either DFS from $\text{acc}_{M,x,\tau\oplus z'}$ or from $\text{rej}_{M,x,\tau\oplus z'}$ (depending on whether x is in the language or not); furthermore, it cannot happen that we encounter

start states in both the DFS from $\text{acc}_{M,x,\tau\oplus z}$ and $\text{rej}_{M,x,\tau\oplus z}$. Thus, we output the correct answer for such an error vector z'. Since M makes more than $10e$ errors only for $\frac{1}{10}$ initial tapes, we output $\perp$ for at most this fraction of initial tapes. □

We note that the proof of Theorem 37 essentially uses the same idea as that used by Folkertsma et al. [FMST25] in one of their proofs for $\mathsf{LCSPACE}(s, c, e) \subseteq \mathsf{CSPACE}(O(s + e \log c), c)$.

Theorem 38.

$$\mathsf{ZP}_{\mathsf{Cat}}\mathsf{CSPACE}(s, c) \subseteq \mathbb{E}_{\mathsf{Cat}}\mathsf{LCSPACE}(O(s), O(c \log c), o(1))$$

Proof. Let M be the $\mathsf{ZP}_{\mathsf{Cat}}\mathsf{CSPACE}(s, c)$ deterministic machine that outputs the correct answer on at least $\frac{1}{2}$ of all possible initial tapes and outputs $\perp$ on the remaining tapes. We run M for $\log c^2$ iterations on the given input, using a different segment of length c from the catalytic tape for each iteration. If any run returns an output besides $\perp$ we return that output; otherwise, we erase the first c bits of the catalytic tape and perform a brute-force search to find an initial tape τ for which M does not output $\perp$. Clearly we use $O(s)$ work space and $O(c \log c)$ bits on the catalytic tape, we always output the correct answer according to the definition of M, and we destroy the tape with a probability of at most $\frac{1}{2^{\log c^2}} = \frac{1}{c^2}$. Thus, we make at most $\frac{c}{c^2} = o(1)$ errors on average (over the tape). □

In order to connect $\mathsf{ZP}_{\mathsf{Cat}}\mathsf{CL}$ to $\mathsf{ZP}^*\mathsf{CL}$, we need to move to the polynomial time variant, which, as it turns out, can be done for the former class without loss of generality:

Theorem 39.

$$\mathsf{ZP}_{\mathsf{Cat}}\mathsf{CL} = \mathsf{ZP}_{\mathsf{Cat}}\mathsf{CLP}$$

Proof. We need only prove the forward direction, as the reverse holds by definition. Let M be the (deterministic) $\mathsf{ZP}_{\mathsf{Cat}}\mathsf{CL}$ machine, and let x be the given input. Also, let M use s bits of work space and c bits of catalytic space. As discussed in subsubsection 2.2.1, we know that for any τ, the graph $G_{M,x}(\text{start}_{M,x,\tau})$ forms a tree, with its sink being one of the three halt configurations: $\text{acc}_{M,x,\tau}$, $\text{rej}_{M,x,\tau}$, or $\perp_{M,x,\tau}$, depending on whether M outputs accept, reject, or $\perp$ on input x.

Since M always restores the catalytic tape to its initial contents, the trees corresponding to different starting catalytic tapes are pairwise vertex-disjoint. By applying an averaging argument similar to that used by Buhrman et al. [BCK+14], we find that the average size of a tree (over τ) is at most 2^{4s}. Using Markov's inequality, the probability (over τ) that a tree has a size of at least 10.2^{4s} is at most $\frac{1}{10}$. Additionally, since M outputs a non-$\perp$ answer with a probability (over τ) of at least $\frac{1}{2}$, we conclude that for at least $\frac{1}{2} - \frac{1}{10} = 0.4$ fraction of the τ, M outputs a non-$\perp$ answer while the tree $G_{M,x}(\text{start}_{M,x,\tau})$ has a size smaller than 10.2^{4s}.

Our $\mathsf{ZP}_{\mathsf{Cat}}\mathsf{CLP}$ machine operates as follows: Given a catalytic tape τ, it first utilizes the subroutine Size from Lemma 11 to determine if the size of the tree $G_{M,x}(\text{start}_{M,x,\tau})$ is less than 10.2^{4s}. If it is, the machine employs the subroutine Next (also from Lemma 11) to perform a walk starting from $\text{start}_{M,x,\tau}$ over $G_{M,x}(\text{start}_{M,x,\tau})$ for 10.2^{4s} steps. The output is determined based on the halting configuration observed during this walk: we accept if we see $\text{acc}_{M,x,\tau}$, reject if we see $\text{rej}_{M,x,\tau}$, and output $\perp$ if we see $\perp_{M,x,\tau}$. Before producing the final answer, the machine uses the subroutine StepBack to return to $\text{start}_{M,x,\tau}$, thereby resetting the catalytic tape. If the size of the tree $G_{M,x}(\text{start}_{M,x,\tau})$ is $\geq 10.2^{4s}$, the machine outputs $\perp$.

Using Lemma 11, we know our machine can use the mentioned subroutines using $O(s)$ workspace, and c bits of catalytic space (which is used to simulate that of M). Moreover, our machine runs in time $2^{O(s)} = \text{poly}(n)$. Furthermore, our machine never outputs an incorrect answer (which follows from the definition of M). We know that for at least 0.4 fraction of the initial catalytic tapes τ, the

tree $G_{M,x}(\text{start}_{M,x,\tau})$ has size smaller than $10 \cdot 2^{4s}$, such that the unique sink/halting configuration in the tree is either $\text{acc}_{M,x,\tau}$ or $\text{rej}_{M,x,\tau}$. Thus, our machine outputs a non-$\perp$ answer for at least 0.4 fraction of τ.

Finally, we note that we can boost this probability to any constant by simply running the machine a constant number of times, each time using a different section of the catalytic tape. □

Finally, we can connect the poly-time variants of $\mathsf{ZP}_{\mathsf{Cat}}\mathsf{CL}$ and $\mathsf{ZP}^*\mathsf{CL}$, thus successfully characterizing $\mathbb{E}_{\mathsf{Cat}}\mathsf{LCL}[O(1)]$:

Theorem 40.

$$\mathbb{E}_{\mathsf{Cat}}\mathsf{LCL}[O(1)] = \mathsf{ZP}_{\mathsf{Cat}}\mathsf{CLP} = \mathsf{ZP}^*\mathsf{CLP}$$

Proof. The first equality is a corollary of Theorem 37, Theorem 38, and Theorem 39. Thus we show both directions of the second equality.

($\Rightarrow$) Let M be a $\mathsf{ZP}_{\mathsf{Cat}}\mathsf{CLP}$ machine that uses c bits of catalytic tape. For a given catalytic tape τ (of length c), we read c two-way random bits, which we denote by r. We then simulate M on the given input using catalytic tape $\tau \oplus r$ and output the result of this simulation. Since M always restores the catalytic tape to its initial contents, we can likewise restore our tape by re-reading the bits of r. Furthermore, we know M outputs a non-$\perp$ (and correct) answer with probability $\frac{1}{2}$ over the initial tape. Since $\tau \oplus r$ gives us a uniformly random tape, we output a non-$\perp$ answer with the same probability over the randomness r.

($\Leftarrow$) Let the $\mathsf{ZP}^*\mathsf{CLP}$ machine N use $m = poly(n)$ random bits on inputs of length n, and c bits of catalytic tape. We construct a machine that uses a catalytic tape of size $c+m$ bits: the first c bits are used to emulate the catalytic tape of N, and the remaining m bits are for the randomness used by N. We output the answer we obtain by simulating N. Since, for *every* initial tape configuration, N produces a non-$\perp$ output with probability $\geq \frac{1}{2}$ over its randomness, our machine will likewise produce a non-$\perp$ output with the same probability over our catalytic tape. □

We close by noting that this connection to $\mathsf{ZP}^*\mathsf{CLP}$ gives us a number of results which do not obviously follow for $\mathbb{E}_{\mathsf{Cat}}\mathsf{LCSPACE}(s, c, O(1))$. First is a containment in randomized polynomial time for the non-polynomial time variant of $\mathbb{E}_{\mathsf{Cat}}\mathsf{LCL}[O(1)]$:

Theorem 41.

$$\mathbb{E}_{\mathsf{Cat}}\mathsf{LCL}[O(1)] \subseteq \mathsf{BPP}$$

Proof. By Theorem 40, $\mathbb{E}_{\mathsf{Cat}}\mathsf{LCL}[O(1)] \subseteq \mathsf{ZP}^*\mathsf{CLP}$. Note that for *every* initial catalytic tape, a $\mathsf{ZP}^*\mathsf{CLP}$ machine runs in polynomial time, uses polynomial space between its work and catalytic tapes, and outputs a non-$\perp$ answer with a probability of at least $\frac{1}{2}$. Thus $\mathsf{ZP}^*\mathsf{CLP}$ is trivially contained in $\mathsf{ZPP} \subseteq \mathsf{BPP}$. □

Second is a characterization of $\mathbb{E}_{\mathsf{Cat}}\mathsf{LCL}[O(1)]\mathsf{P}$ as exactly being the intersection of $\mathbb{E}_{\mathsf{Cat}}\mathsf{LCL}[O(1)]$ with no time restriction and P with no catalytic restriction; this was previously proven for CLP and $\mathsf{CL} \cap \mathsf{P}$ by Cook et al. [CLMP25]:

Theorem 42.

$$\mathbb{E}_{\mathsf{Cat}}\mathsf{LCL}[O(1)]\mathsf{P} = \mathbb{E}_{\mathsf{Cat}}\mathsf{LCL}[O(1)] \cap \mathsf{P}$$

Proof. The forward direction follows from the definitions of the classes. For the reverse direction, consider an arbitrary language $L \in \mathbb{E}_{\mathsf{Cat}}\mathsf{LCL}[(O(1))] \cap \mathsf{P}$; it follows from Theorem 40 that $L \in \mathsf{ZP}_{\mathsf{Cat}}\mathsf{CLP} \cap \mathsf{P}$. Let M be the $\mathsf{ZP}_{\mathsf{Cat}}\mathsf{CLP}$ (deterministic) machine for L, such that M uses a catalytic

tape of size n^d on input length n (for some constant d). Without loss of generality, we assume that L has a polynomial-time algorithm with a runtime of n^d.

Our $\mathbb{E}_{\mathsf{Cat}}\mathsf{LCL}[(O(1))]\mathsf{P}$ machine N uses a catalytic tape of size n^{2d}, which is broken into n^d chunks, each of size n^d. N works as follows: given input x, it simulates M for input x with each chunk serving as the starting catalytic tape for M. Note that, by the definition of M, after each simulation of M, the contents of the respective chunk are reset to their original state. If N receives a non-$\perp$ answer from even one chunk during the simulation of M, it outputs that answer. Otherwise, it destroys its catalytic tape and uses the space to run the polynomial-time algorithm for L.

Thus, N always decides the input correctly. Since M runs in polynomial time, N does as well. Lastly, since M outputs a non-$\perp$ answer for at least $\frac{1}{2}$ of its initial tapes, the probability (over the uniform choice of its starting catalytic tape) that N never sees M output a non-$\perp$ answer for any of the chunks is $\leq \frac{1}{2^{n^d}}$. Thus, the average number of errors made by N (averaged over its starting tape) is $\frac{n^{2d}}{2^{n^d}} = o(1)$. □

Lastly, the equivalence to $\mathsf{ZP}^*\mathsf{CLP}$ gives evidence that proving $\mathbb{E}_{\mathsf{Cat}}\mathsf{LCL}[(O(1))] = \mathsf{CL}$ is *much harder* than $\mathbb{E}_{\mathsf{Rand}}\mathsf{BPLCL}[O(1)]$, as it would show novel derandomizations, such as reducing ZPTC^1 to the *lossy coding problem* (see [CLMP25] for further discussion). On the flip side, an appropriate derandomization assumption would remove this barrier and give reason to believe that the collapse of $\mathbb{E}_{\mathsf{Cat}}\mathsf{LCL}[O(1)]$, and hence all previously discussed classes, could indeed occur; we validate this idea in the upcoming section.

7 Further Results Assuming Derandomization

We now move to conditional results, in particular assuming a fairly standard hardness assumption:

Conjecture 43. *There exists a constant $\gamma > 0$ such that $\mathsf{DSPACE}(n) \not\subseteq \mathsf{SIZE}(2^{\gamma n})$.*

Conjecture 43 is sufficient to construct *pseudorandom generators* (PRGs); we will take the following construction due to Impagliazzo and Wigderson [IW97]:

Lemma 44 (Derandomization assumption)**.** *If Conjecture 43 holds, then for all constants d, there exists a constant d' and a function $G : \{0,1\}^{d' \log n} \to \{0,1\}^n$ such that for any circuit C of size n^d, G $\frac{1}{n}$-fools the circuit C, i.e.,*

$$\left|\mathbf{Pr}_{r\in\{0,1\}^n}[C(r) = 1] - \mathbf{Pr}_{s\in\{0,1\}^{d' \log n}}[C(G(s) = 1]\right| < \frac{1}{n}$$

and G is computable in space logarithmic in n.

Note that Lemma 44 is sufficient to show $\mathsf{BPP} = \mathsf{P}$, which extends to $\mathsf{BP}^\epsilon\mathsf{C}^\delta\mathsf{L}_{\mathsf{high}}\mathsf{P}$ by Theorem 17:

Theorem 45. *Let δ, ϵ be constants such that $\delta > 2\epsilon$. If Conjecture 43 holds, then*

$$\mathsf{BPP} = \mathsf{BP}^\epsilon\mathsf{C}^\delta\mathsf{LP} = \mathsf{P}$$

Since $\mathsf{CL} \subseteq \mathsf{ZPP}$, this also immediately implies that $\mathsf{CL} \subseteq \mathsf{P}$. Combining this with a result of Cook et al. [CLMP25] which shows $\mathsf{CLP} = \mathsf{CL} \cap \mathsf{P}$, we can extend this to the following:

Theorem 46. *If Conjecture 43 holds, then*

$$\mathsf{CL} = \mathsf{CLP}$$

It is not surprising that one can strengthen Lemma 7, particularly the fact that $\mathsf{CL} = \mathsf{LCL}[O(1)]$, from allowing at most e errors for every initial catalytic setting to allowing at most e errors on average (averaged over the initial catalytic tape), under the de-randomization assumption.

Lemma 47. *If $e \leq c^{1-\Omega(1)}$ and Conjecture 43 holds, then*

$$\mathbb{E}_{\mathsf{Cat}}\mathsf{LCSPACE}(\Theta(s), \Theta(c), e) = \mathsf{CSPACE}(\Theta(s + e \log c), \Theta(c))$$

Proof. The reverse direction holds by Lemma 7, and so we focus on the forward direction. Let $L \in \mathbb{E}_{\mathsf{Cat}}\mathsf{LCSPACE}(s, c, e)$, and hence $L \in \mathsf{ZP}_{\mathsf{Cat}}\mathsf{CSPACE}(O(s + e \log c), c)$ by Theorem 37. Let M be the deterministic machine for L as per the definition of this class, and let x be the input.

We define an initial catalytic setting τ and the configuration graph $G_{M,x,\tau}$ as *good* if and only if M produces a non-$\perp$ answer (i.e., M halts in either $\text{acc}_{M,x,\tau}$ or $\text{rej}_{M,x,\tau}$) when executed with input x and catalytic tape τ. By the definition of M, we know that at least $\frac{1}{2}$ of the initial catalytic tapes are good. Since M always resets the catalytic tape, we also know that the configuration graphs for different initial catalytic tapes are vertex-disjoint.

Therefore, based on the argument by Buhrman et al. [BCK+14], we can conclude that the average size (number of vertices) of the configuration graph $G_{M,x,\tau}$ is at most 2^{4s}. By Markov's inequality, the probability (over τ) that the configuration graph has size $\geq 10.2^{4s}$ is at most $\frac{1}{10}$. Therefore, the probability (over τ) that the configuration graph is both good and has a size smaller than 10.2^{4s} is at least $\frac{1}{2} - \frac{1}{10} = 0.4$.

Consider the circuit $C_{x,\tau}$, which takes another catalytic setting w as input and outputs 1 if and only if the configuration graph $G_{M,x,\tau\oplus w}$ is both good and smaller than $T = 10 \cdot 2^{4s}$ in size. As M is a deterministic machine, $G_{M,x,\tau\oplus w}$ is simply a path. Thus, the circuit can check the configuration after T simulation steps of M and output 1 if it is either $\text{acc}_{M,x,\tau\oplus w}$ or $\text{rej}_{M,x,\tau\oplus w}$. Therefore, using the fact that $e \leq c \leq 2^s$, the size of the circuit can be bounded by a sufficiently large polynomial in T. Thus, we get that $C_{x,\tau}$ outputs 1 for at least 0.4 fraction of inputs w. Note that we can give the circuit a longer input, say $w \circ r$, where r is a redundant part that the circuit ignores; such that the length of input $w \circ r$ is, say, $O(T)$. The circuit size is now polynomial in its input size.

Under the derandomization assumption in Lemma 44, there exists a PRG P that has a seed length of $O(\log(T)) = O(s)$ and is computable in space $O(\log(T)) = O(s)$, such that P $\frac{1}{10}$-fools the circuit $C_{x,\tau}$. Thus, for any τ, fraction of seeds of P making the circuit $C_{x,\tau}$ output 1 is at least 0.3.

Finally, we describe the $\mathsf{CSPACE}(O(s + e \log c), c)$ machine N for L. The input is x, and the initial catalytic setting is τ. N simply does the following: For every seed q of the PRG P, it computes $P(q)$ and simulates M with the catalytic tape $\tau \oplus P(q)$. If, for any seed, we see M give a non-$\perp$ answer, N outputs that; otherwise, it outputs ERROR.

The machine N can simulate the machine M using a work space of $O(s + e \log c)$ and a catalytic space c. Additionally, since the PRG P has a seed length of $O(s)$ and is computable in space $O(s)$, N can exhaustively try all possible seeds. Furthermore, because M always restores the catalytic tape to its initial state, N can also restore the catalytic tape by XORing it back with $P(q)$ after running the simulation for seed q. Observe that, there exists a seed q such that when M is run with input x and catalytic tape $\tau \oplus P(q)$, it outputs a non-$\perp$ answer. By the definition of M, this will be the correct output. Therefore, N always provides the correct answer and never outputs ERROR. □

Lemma 47 is enough to show that all the catalytic classes discussed in this paper, with the sole exceptions of $\mathsf{BP}^{\epsilon}\mathsf{C}^{\delta}\mathsf{L}_{\mathsf{high}}(\mathsf{P})$, are equivalent, since $\mathbb{E}_{\mathsf{Cat}}\mathsf{LCL}[e]$ is the highest class in the chain of containments.

Corollary 48. *If Conjecture 43 holds, then*

$$\mathbb{E}_{\mathsf{Cat}}\mathsf{LCL}[O(1)] = \mathsf{CLP}$$

Besides giving a full characterization of all classes discussed in this paper, another interesting way of interpreting Corollary 48 is that it "scales down" a result showing e.g. $\mathsf{P} = \mathsf{ZPP}$ to show $\mathsf{CLP} = \mathsf{ZP}^*\mathsf{CLP}$.

We also make a note about more general parameters. When derandomizing $\mathsf{BP}^\epsilon\mathsf{C}^\delta\mathsf{SPACE}(s, c)$ into $\mathbb{E}_{\mathsf{Cat}}\mathsf{LCSPACE}(s, c, o(1))$ using Theorem 34, we end up using catalytic space $2^{O(s)}$ regardless of the value of c. However, in Appendix A we show that Conjecture 43 allows us to avoid $\mathbb{E}_{\mathsf{Cat}}\mathsf{LCSPACE}(s, c, o(1))$ and derandomize directly to $\mathsf{CSPACE}(s, c)$ without incurring this overhead:

Theorem 49. *Let δ, ϵ be constants such that $\delta < 2\epsilon$. Then assuming Conjecture 43 holds we have*

$$\mathsf{BP}^\epsilon\mathsf{C}^\delta\mathsf{SPACE}(s, c) \subseteq \mathsf{CSPACE}(O(s), O(c))$$

This is a more general "scaling down" of a result due to [KMPS25], which shows that Conjecture 43, which implies $\mathsf{BPL} = \mathsf{L}$, thus implies $\mathsf{BPCSPACE}(s, c) \subseteq \mathsf{CSPACE}(O(s), O(c))$ (the former class being equivalent to $\mathsf{BP}^\epsilon\mathsf{C}^0\mathsf{SPACE}(s, c)$).

References

[AAV26] Aryan Agarwala, Yaroslav Alekseev, and Antoine Vinciguerra. Linear matroid intersection is in catalytic logspace. In *Innovations in Theoretical Computer Science Conference (ITCS)*, volume 362 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 3:1–3:23. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2026.

[AFM+25] Yaroslav Alekseev, Yuval Filmus, Ian Mertz, Alexander Smal, and Antoine Vinciguerra. Catalytic computing and register programs beyond log-depth. In *Symposium on Mathematical Foundations of Computer Science (MFCS)*, volume 345 of *LIPIcs*, pages 6:1–6:18, 2025.

[AM25] Aryan Agarwala and Ian Mertz. Bipartite matching is in catalytic logspace. In *IEEE Symposium on Foundations of Computer Science (FOCS)*. IEEE Computer Society, 2025.

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

[BDRS25] Sagar Bisoyi, Krishnamoothy Dinesh, Bhabya Rai, and Jayalal Sarma. Almost-catalytic computation. In *CIAC*, volume 15680 of *Lecture Notes in Computer Science (LNCS)*, pages 35–51, 2025.

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

[BFM+25] Harry Buhrman, Marten Folkertsma, Ian Mertz, Florian Speelman, Sergii Strelchuk, Sathyawageeswar Subramanian, and Quinten Tupker. Quantum catalytic space. In *TQC*, volume 350 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 11:1–11:24. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2025.

[BKLS18] Harry Buhrman, Michal Koucký, Bruno Loff, and Florian Speelman. Catalytic space: Non-determinism and hierarchy. *Theory of Computing Systems (TOCS)*, 62(1):116–135, 2018.

[CGM+25] James Cook, Surendra Ghentiyala, Ian Mertz, Ted Pyne, and Nathan Sheffield. The structure of in-place space-bounded computation. *Computing Research Repository (CoRR)*, abs/2510.12005, 2025.

[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)*, pages 554–564. Association for Computing Machinery (ACM), 2025.

[CM20] James Cook and Ian Mertz. Catalytic approaches to the tree evaluation problem. In *ACM Symposium on Theory of Computing (STOC)*, pages 752–760. Association for Computing Machinery (ACM), 2020.

[CM21] James Cook and Ian Mertz. Encodings and the tree evaluation problem. *Electronic Colloquium on Computational Complexity (ECCC)*, TR21-054, 2021.

[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 Proceedings in Informatics (LIPIcs)*, pages 8:1–8:21, 2022.

[CM25] James Cook and Ian Mertz. Tree evaluation is in space O(log n · log log n). *SIAM Journal on Computing (SICOMP)*, Special Section STOC 2024:130–160, 2025.

[CP26] James Cook and Edward Pyne. Efficient catalytic graph algorithms. In *Innovations in Theoretical Computer Science Conference (ITCS)*, volume 362 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 43:1–43:22. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2026.

[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. Springer, 2020.

[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.

[DPTW25] Dean Doron, Edward Pyne, Roei Tell, and R. Ryan Williams. When connectivity is hard, random walks are easy with non-determinism. In *ACM Symposium on Theory of Computing (STOC)*, pages 1108–1117. Association for Computing Machinery (ACM), 2025.

[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 *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 50:1–50:13. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 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 Proceedings in Informatics (LIPIcs)*, pages 16:1–16:13. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019.

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

[GTS25] Chetan Gupta, Raghunath Tewari, and Vimal Raj Sharma. Efficient isolation of perfect matching in o(log n) genus bipartite graphs. *CoRR*, abs/2511.21217, 2025.

[IW97] Russell Impagliazzo and Avi Wigderson. $\mathbf{P} = \mathbf{BPP}$ if $\mathbf{E}$ requires exponential circuits: derandomizing the XOR lemma. In *ACM Symposium on Theory of Computing (STOC)*, pages 220–229, 1997.

[KMPS25] Michal Koucký, Ian Mertz, Ted Pyne, and Sasha Sami. Collapsing catalytic classes. In *IEEE Symposium on Foundations of Computer Science (FOCS)*. IEEE Computer Society, 2025.

[LPT24] Jiatu Li, Edward Pyne, and Roei Tell. Distinguishing, predicting, and certifying: On the long reach of partial notions of pseudorandomness. In *IEEE Symposium on Foundations of Computer Science (FOCS)*, to appear, 2024.

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

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

[Pot17] Aaron Potechin. A note on amortized branching program complexity. In *IEEE Conference on Computational Complexity (CCC)*, volume 79 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 4:1–4:12. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017.

[Pyn25] Edward Pyne. Derandomizing logspace with a small shared hard drive. *Computational Complexity (CC)*, 34(2):13, 2025.

[Rud76] Walter Rudin. *Principles of Mathematical Analysis*. McGraw-Hill, 3rd edition, 1976.

[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 Computer Society, 2021.

[Sha25] Yakov Shalunov. Improved bounds on the space complexity of circuit evaluation. *CoRR*, abs/2504.20950, 2025.

[SZ99] Michael E. Saks and Shiyu Zhou. $BP_HSPACE[S] \subseteq DSPACE[S^{3/2}]$. *Journal of Computer and System Sciences (J.CSS)*, 58(2):376–403, 1999.

[Wil25] R. Ryan Williams. Simulating time with square-root space. In *ACM Symposium on Theory of Computing (STOC)*, pages 13–23. Association for Computing Machinery (ACM), 2025.

A Proof of Theorem 49

For constants $0 \leq \delta < 2\epsilon$, we consider an arbitrary language $L \in \mathsf{BP}^\epsilon\mathsf{C}^\delta\mathsf{SPACE}(s, c)$, with M being the corresponding machine according to the definition of the class. Going forward, we will work with a fixed input x. Without loss of generality[3], we can assume δ and ϵ are rational.

[3]Let δ', ϵ' be rationals such that $\delta < \delta' < 2\epsilon' < 2\epsilon$. Then, by definition $\mathsf{BP}^\epsilon\mathsf{C}^\delta\mathsf{SPACE}(s, c) \subseteq \mathsf{BP}^{\frac{\epsilon'}{2}}\mathsf{C}^{\delta'}\mathsf{SPACE}(s, c)$.

The proof of the following lemma demonstrates that the class $\mathsf{BP}^\epsilon\mathsf{C}^\delta\mathsf{SPACE}(s,c)$ (low-error regime) is contained within the class of languages that can be solved by a bounded-probabilistic machine running in time $2^{O(s)}$. This will be useful for proving Theorem 49. Recall our discussion from Section 4.2 about the τ^β-graph, in which we had set β to be the constant $\frac{1}{2}\left(1-\frac{\delta}{2\epsilon}\right)$. It followed from Lemma 22 that, for this value of β, the probability of exiting the τ^β-graph via $\perp$ is at most the constant $\gamma = \frac{2\delta}{1+\frac{\delta}{2\epsilon}}$, which is less than 2ϵ.

Lemma 50. *Let δ, ϵ be constants such that $\delta < 2\epsilon$. Then, we have that $\mathsf{BP}^\epsilon\mathsf{C}^\delta\mathsf{SPACE}(s,c) \subseteq \mathrm{BPTIME}[\frac{1}{2}-\epsilon, \frac{1}{2}-\epsilon+\eta](2^{10s})$, for some constant $\eta > 0$.*

Proof. Consider the algorithm $\mathcal{A}$ which works as follows (given input x): it first selects τ, an initial setting for the catalytic contents, uniformly at random. Then, it takes a random walk of length $T \cdot 2^{4s}$ from $\mathrm{start}_{M,x,\tau}$ in $G_{M,x,\tau}$. If it reaches $\mathrm{acc}_{M,x,\tau}$, it accepts; otherwise, it rejects. Here, T is a constant that we set to $\lceil 10(1+\frac{\frac{1}{2}-\epsilon}{(2\epsilon-\gamma)})\rceil$. As the size of τ is $c \leq 2^s$, being generous, it follows that $\mathcal{A}$ is a probabilistic algorithm that runs in time 2^{10s}.

If the input $x \notin L$, then regardless of τ, the probability of reaching $\mathrm{acc}_{M,x,\tau}$ from $\mathrm{start}_{M,x,\tau}$ is $\leq \frac{1}{2}-\epsilon$; thus $\mathcal{A}$ accepts with a probability of $\leq \frac{1}{2}-\epsilon$. Next, consider the case where $x \in L$. Using Lemma 21 and the Markov inequality, we know that with a probability (over τ) of at least $1-\frac{1}{T}$, the size of the τ^β-graph is less than $T \cdot 2^{4s}$. In such a scenario, since the length of the walk is greater than the size of the τ^β-graph, the random walk will either exit the τ^β-graph to a non-τ^β-node or halt at either $\mathrm{acc}_{M,x,\tau}$ or $\mathrm{rej}_{M,x,\tau}$ within the τ^β-graph. Using Lemma 22 we know that the probability the walk leaves the τ^β-graph (to a non-τ^β-node) is $\leq \gamma < 2\epsilon$, for our choice of β. By the definition of M, the probability that the walk goes to $\mathrm{rej}_{M,x,\tau}$ (in the τ^β-graph) is $\leq \frac{1}{2}-\epsilon$. Thus, $\mathcal{A}$ accepts with a probability of at least

$$\begin{aligned}
\left(1-\frac{1}{T}\right)\left(1-\left(\frac{1}{2}-\epsilon\right)-\gamma\right) &\geq \left(1-\frac{2\epsilon-\gamma}{10(\frac{1}{2}+\epsilon-\gamma)}\right)\left(\frac{1}{2}+\epsilon-\gamma\right) \\
&\geq \frac{5+8\epsilon-9\gamma}{10} \\
&\geq \frac{1}{2}+\frac{8\epsilon-9\gamma}{10} \\
&= \frac{1}{2}-\epsilon+\frac{9(2\epsilon-\gamma)}{10}
\end{aligned}$$

Since $(2\epsilon-\gamma) > 0$ is a constant, the probability that $\mathcal{A}$ accepts when $x \in L$ is bounded away from $\frac{1}{2}-\epsilon$ by a positive constant; hence, the lemma follows by setting $\eta = \frac{9(2\epsilon-\gamma)}{10}$. □

Corollary 51. *Let δ, ϵ be constants such that $\delta < 2\epsilon$. Then, we have that $\mathsf{BP}^\delta\mathsf{C}^\epsilon\mathsf{L} \subseteq \mathrm{BPP}$.*

Remark 52. *Using Corollary 35 and Theorem 41 we already know that for constants $\delta < 2\epsilon$, $\mathsf{BP}^\epsilon\mathsf{C}^\delta\mathsf{L} \subseteq \mathrm{ZPP}$.*

Consider the algorithm $\mathcal{A}'$ that takes as input x and an initial catalytic setting τ. It behaves exactly like the algorithm $\mathcal{A}$ mentioned in the proof above, but it performs the walk over $G_{M,x,\tau\oplus w}$, where w is chosen uniformly at random. As we were generous with time estimates, $\mathcal{A}'$ also runs in probabilistic time 2^{10s}. As for any τ, $\tau \oplus w$ also gives a uniform distribution over all catalytic settings, $\mathcal{A}'$ decides x with the same probability separation as $\mathcal{A}$. Let $C_{x,\tau}$ denote the circuit that performs the computation of $\mathcal{A}'$ for given random bits as input; i.e., the circuit takes an input of

length $N = c + T \cdot 2^{4s}$, where the first c bits are for the random catalytic tape setting; and the latter bits are for the random walk. Since $c \leq 2^s$, the size of the circuit can be bounded by a large enough polynomial in 2^{10s}. Thus, the size of the circuit is polynomial in the size of the input; therefore, by Lemma 44, there exists a PRG $P : \{0,1\}^{d' \log N} \to \{0,1\}^N$ that $\frac{1}{N}$-fools the circuit $C_{x,\tau}$ and can be computed in the space $O(\log N) = O(s)$ (where d' is some constant).

Observation 53. *The circuit $C_{x,\tau}$ outputs 1 (accepts) on at most a $\frac{1}{2} - \epsilon$ fraction of inputs when $x \notin L$, and on at least a $\frac{1}{2} - \epsilon + \frac{9(2\epsilon-\gamma)}{10}$ fraction of inputs when $x \in L$.*

Proof. Follows from the definition of the circuit $C_{x,\tau}$ and Lemma 50. □

Before giving the proof of Theorem 49, we will need a few definitions.

Definition 54 (Layered Configuration Graph)**.** *Let $k \in \mathbb{N}$. We define the layered configuration graph with k layers associated with $G_{M,x}$ as follows:*

- *Its vertex set consists of pairs $\langle v, i \rangle$ for every $v \in V(G_{M,x})$ and $i \in [k] \cup \{0\}$.*
- *For every directed edge (v, v') in $G_{M,x}$ with label $b \in \{0,1\}$ and $i \in [k]$, the layered graph contains a directed edge from $\langle v, i \rangle$ to $\langle v', i-1 \rangle$, carrying the same label b.*
- *Additionally, for every halt configuration $v \in V(G_{M,x})$ and $i \in [k]$, the layered graph includes a directed edge from $\langle v, i \rangle$ to $\langle v, i-1 \rangle$ that is labeled with both 0 and 1.*

One can imagine the layers arranged from left to right, numbered from k to 0. Note that the vertices in the last layer (layer 0) have no outgoing edges.

Thus, every vertex in the layered configuration has at most two outgoing edges, with each edge label assumed to occur with a probability $\frac{1}{2}$. Notice that the probability of transitioning from $\langle v, i \rangle$ to $\langle v', 0 \rangle$ in the layered graph, where v' is a halting configuration, is the same as the probability (over the read-once randomness of M) of reaching the configuration v' within i steps when the machine M is executed starting from v.

Remark 55 (Notation)**.** *The vertices of the layered configuration graph, which we will simply refer to as configurations, are denoted by $\langle v, i \rangle$. In this context, $v \in V(G_{M,x})$ is represented as $\langle \pi, u \rangle$, where $\pi \in \{0,1\}^c$ and $u \in \{0,1\}^s$. To avoid nested brackets, we will also denote $\langle v, i \rangle$ as $\langle \pi, u, i \rangle$.*

Definition 56 (y-tree)**.** *Let $y \in \{0,1\}^m$. Consider the subgraph of the layered configuration graph of $G_{M,x}$, that uses the same vertex set but includes only specific edges. For each $i \in [m]$ and for every vertex $\langle v, i \rangle$ in layer i, we include only the outgoing edge labeled with y_{m-i+1}. For $i > m$ we include no outgoing edges.*

For any vertex $\langle v', j \rangle$, the connected component of this subgraph that contains $\langle v', j \rangle$ is referred to as the y-tree of $\langle v', j \rangle$, which is denoted as y-tree($\langle v', j \rangle$).

Note that by definition y-tree($\langle v, j \rangle$) is the isolated vertex $\langle v, j \rangle$ for $j > m$. Whereas for $j \leq m$, y-tree($\langle v, j \rangle$) is a directed tree whose unique sink is a vertex in layer 0 that can be reached from $\langle v, j \rangle$ by following the edge labels obtained by reading the bits y_{m-j+1} to y_m.

Remark 57. *Assuming we have oracle access to y, the subroutines outlined in Lemma 11 can be easily extended to traverse y-tree($\langle v, j \rangle$) while maintaining the time and space guarantees provided in Lemma 11. This is possible as long as $|y| \leq 2^{O(s)}$, which will be the case in our usage; since we only need an additional $O(s)$ work space space to keep track of the layer number of the current vertex/configuration we are at while performing a walk. Furthermore, since a y-tree($\langle v, 0 \rangle$) has $\langle v, 0 \rangle$ as its sink, the guarantees from Lemma 11 for a halting configuration also hold for the sink $\langle v, 0 \rangle$.*

Definition 58 (y-DFS)**.** *When we refer to performing a y-DFS starting from $\langle v, j\rangle$, we mean that we execute the walk defined by* Next *(in Lemma 11) over y-tree($\langle v, j\rangle$), starting from $\langle v, j\rangle$. Conversely, when we say we conduct a y-DFS from $\langle v, j\rangle$ in a reversible manner, we mean that we perform the walk in reverse order using* StepBack *(also in Lemma 11).*

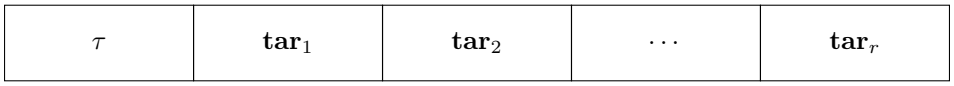

Figure 3: Catalytic tape

To prove Theorem 49, we describe a deterministic catalytic algorithm $\mathcal{B}$, which always resets the tape and decides the language L. $\mathcal{B}$ has the following catalytic tape structure: it consists of c bits denoted by τ in Figure 3, followed by r strings $\mathbf{tar}_1$ to $\mathbf{tar}_r$, called targets. We set $r = \lceil \frac{c}{s} \rceil$, and the length of each string $\mathbf{tar}_j$ is $10d \cdot s$, where d is a constant such that $d \geq 5$ and $d \cdot s$ is an upper bound to the space required to compute the PRG P (described above) for a given seed. This includes the space to hold a seed. At a high level, $\mathcal{B}$ employs a *compress-or-compute* approach introduced by Cook et al. [CLMP25]. Here, $\mathcal{B}$ either compresses the catalytic tape to create enough space for an inefficient brute-force algorithm or is able to decide the input quickly.

Assume that we are processing $\mathbf{tar}_j$ and the first c bits of the catalytic tape at this point are π. $\mathcal{B}$ works as follows:

1. We iterate through all the seeds q. For each seed q, $P(q)$ gives us a string of length $N = c + T \cdot 2^{4s}$; where we treat the first c bits as a catalytic setting w and the latter bits as choices for a random walk, denoted by y.

2. For each w, y and $i \in [T \cdot 2^{4s}]$, we call Size from Lemma 11 on $y_{\leq i}$-tree($\langle \mathrm{acc}_{M,x,\pi\oplus w}, 0\rangle$) to see if any of the trees is larger than $2^{10d \cdot s}$.

3. If for some w, y, i the size of $y_{\leq i}$-tree($\langle \mathrm{acc}_{M,x,\pi\oplus w}, 0\rangle$) is larger than $2^{10d \cdot s}$, we compress the contents of the catalytic tape using time-step compression [CLMP25] as follows. Let $val(\mathbf{tar}_j)$ represent the integer value of the binary string $\mathbf{tar}_j$ incremented by one. Thus, $val(\mathbf{tar}_j)$ is an integer in $[2^{10d \cdot s}]$. We call the walk procedure Next for $val(\mathbf{tar}_j)$ times to get from $\langle \mathrm{acc}_{M,x,\pi\oplus w}, 0\rangle$ to some configuration $\langle \pi', u, k\rangle$ where π' is the catalytic part, u is work space part, and $k \leq T \cdot 2^{4s} < 2^{5s}$ is the layer index. In place of $\mathbf{tar}_j$ on the catalytic tape, we write $q \circ u \circ k \circ i \circ 0^{10d \cdot s - (ds + 5s + 5s + 5s)}$ where q is the seed corresponding to w, y. Here, we used the fact that q can be specified using ds bits and k, i using $5s$ bits each. Also, the work space part of the configuration u (which includes head locations, etc.) can be specified using $5s$ bits. Since $d \geq 5$, we managed to free up at least $6d \cdot s$ bits. The walk also replaced $\pi \oplus w$ with π' on the catalytic tape. After compression, we move on to the next target $\mathbf{tar}_{j+1}$.

   This step is reversible, as we can always decompress as follows: we walk back using the subroutine StepBack in Lemma 11, from $\langle \pi', u, k\rangle$ over the $y_{\leq i}$-tree of $\langle \mathrm{acc}_{M,x,\pi\oplus w}, 0\rangle$, where $y_{\leq i}$ can be computed using i and q. We count the number of steps it takes for us to reach the unique halting/sink state $\langle \mathrm{acc}_{M,x,\pi\oplus w}, 0\rangle$. This count is precisely the value of $\mathbf{tar}_j$. Thus, we recover $\mathbf{tar}_j$. At this point, the first c bits of the catalytic tape are $\pi \oplus w$. Finally, we can compute w from q and XOR it with the first c bits of the catalytic tape, resetting it to π.

4. If the sizes of all the $y_{\leq i}$-tree($\langle \mathrm{acc}_{M,x,\pi\oplus w}, 0\rangle$) are smaller than $2^{10d \cdot s}$, we count the number of pairs (w, y) such that for some i, the $y_{\leq i}$-DFS from $\langle \mathrm{acc}_{M,x,\pi\oplus w}, 0\rangle$ visits $\langle \mathrm{start}_{M,x,\pi\oplus w}, i\rangle$.

We call such a pair (w, y) *accepting*. If the fraction of seeds q for which the corresponding pair w, y is accepting is at least $\frac{1}{2} - \epsilon + \frac{27(2\epsilon-\gamma)}{40}$, we accept; otherwise, we reject. In this step, we essentially do not change the catalytic contents π (each time we XOR it with some w, we XOR it back before moving to the next seed). Thus, before outputting the final answer, we just decompress all the previous targets as described in step 3.

5. In the case where we compress all the targets without hitting step 4, we free up the last $6d \cdot s$ bits of each target, thus freeing up $6d \cdot s \cdot r \geq 6d \cdot s \cdot \frac{c}{s} > c$ bits in total on the catalytic tape. We claim that this is sufficient to decide the input. Again, before we provide the final output, we decompress all targets as described in step 3.

The seed length $q = O(s)$ and $P(q)$ can be computed using space $O(s)$. Additionally, any DFS we conduct is for at most $2^{O(s)}$ steps. Therefore, using Lemma 11, we can see that all steps utilize $O(s)$ space, except for step 5. The space considerations for this step are established in Observation 60 below. The size of the catalytic tape is given by $c + 10d \cdot s \cdot \lceil \frac{c}{s} \rceil = O(c)$. Furthermore, we always reset the catalytic tape since the compression described in step 3 is reversible. The correctness of the algorithm $\mathcal{B}$ can be established through the following two observations:

Observation 59. *In case $\mathcal{B}$ hits step 4, it correctly decides the input x.*

Proof. Recall that the PRG P $\frac{1}{N}$-fools the circuit $C_{x,\pi}$ (for every π). Since ϵ and γ are constants satisfying $(2\epsilon - \gamma) > 0$, for large enough N, we have that $\frac{1}{N} < \frac{9(2\epsilon-\gamma)}{40}$. Consequently, by applying Observation 53, we conclude that if $x \in L$, then at least a $\frac{1}{2} - \epsilon + \frac{27(2\epsilon-\gamma)}{40}$ fraction of the seeds cause the circuit to output 1; whereas if $x \notin L$, then at most a $\frac{1}{2} - \epsilon + \frac{9(2\epsilon-\gamma)}{40}$ fraction of the seeds make the circuit output 1. Recall that the circuit accepts the input $P(q) \coloneqq w \circ y$ iff the walk as per y from $\text{start}_{M,x,\pi\oplus w}$ reaches $\text{acc}_{M,x,\pi\oplus w}$ (in $G_{M,x,\pi\oplus w}$). Because all the $y_{\leq i}$-DFS are small—specifically, they return to $\langle \text{acc}_{M,x,\pi\oplus w}, 0\rangle$ within $2^{10d\cdot s}$ steps—Lemma 11 guarantees that for each $i \in [T \cdot 2^{4s}]$, every vertex in the $y_{\leq i}$-tree$(\langle \text{acc}_{M,x,\pi\oplus w}, 0\rangle)$ is visited at least once. Consequently, we can equivalently state that the circuit accepts the input $w \circ y$ if and only if the $y_{\leq i}$-DFS starting from $\langle \text{acc}_{M,x,\pi\oplus w}, 0\rangle$ reaches $\langle \text{start}_{M,x,\pi\oplus w}, i\rangle$ for some $i \in [T \cdot 2^{4s}]$. Hence, it suffices to check the proportion of accepting pairs (w, y) and accept if and only if this fraction is at least $\frac{1}{2} - \epsilon + \frac{27(2\epsilon-\gamma)}{40}$.

□

Observation 60. *In case $\mathcal{B}$ reaches step 5 and frees up c bits, this is enough to decide the input using $O(s)$ work space.*

Proof. Assume that there exists a catalytic setting π such that the size of $y_{\leq i}$-DFS from $\langle \text{acc}_{M,x,\pi\oplus w}, 0\rangle$ is small for all w, y, i (where $w \circ y$ denotes the output of P for a seed and $i \in [T \cdot 2^{4s}]$). We can use the freed-up c bits on the catalytic tape to find such a π by brute-force, and then use step 4 to correctly decide the input; where the correctness follows from observation 59. This clearly takes $O(s)$ work space. Now we argue that such a π exists.

From the proof of Observation 21, we know that $G_{M,x}$ has at most 2^{c+4s} configurations. Hence, its layered configuration graph (with $T \cdot 2^{4s}$ layers) contains at most $T \cdot 2^{c+8s}$ configurations in total. Fix a seed q of P such that $P(q) = w \circ y$ and fix some $i \in [T \cdot 2^{4s}]$. For any two distinct τ', τ'', the vertex sets of $y_{\leq i}$-tree$(\langle \text{acc}_{M,x,\tau'\oplus w}, 0\rangle)$ and $y_{\leq i}$-tree$(\langle \text{acc}_{M,x,\tau''\oplus w}, 0\rangle)$ are disjoint. The reason is that every configuration in the first tree reaches the unique sink vertex $\langle \text{acc}_{M,x,\tau'\oplus w}, 0\rangle$ under a $y_{\leq i}$-DFS, while every configuration in the second tree reaches the unique sink $\langle \text{acc}_{M,x,\tau''\oplus w}, 0\rangle$. Consequently, the expected size (over a uniformly random choice of π') of the vertex set of $y_{\leq i}$-tree$(\langle \text{acc}_{M,x,\pi'\oplus w}, 0\rangle)$ is at most $T \cdot \frac{2^{c+8s}}{2^c} = T \cdot 2^{8s}$. Let us choose π' uniformly at random. Then we have

$$\begin{aligned}
&\mathbf{Pr}[\exists i \in [T \cdot 2^{4s}], q \text{ such that } G(q) \coloneqq w \circ y \text{ and } y_{\leq i}\text{-DFS from } \langle \mathrm{acc}_{M,x,\pi' \oplus w}, 0\rangle \text{ has size } \geq 2^{10d \cdot s}] \\
&\leq \sum_{q,i} \mathbf{Pr}[G(q) \coloneqq w \circ y \text{ and } y_{\leq i}\text{-DFS from } \langle \mathrm{acc}_{M,x,\pi' \oplus w}, 0\rangle \text{ has size } \geq 2^{10d \cdot s}] \\
&\leq \sum_{q,i} \frac{T \cdot 2^{8s}}{2^{10d \cdot s}} \leq \sum_{q,i} \frac{1}{2^{8d \cdot s}} \leq \frac{2^{ds} \cdot T \cdot 2^{4s}}{2^{8d \cdot s}} \leq \frac{1}{2^{6d \cdot s}} \leq \frac{1}{2^{30s}} < 1
\end{aligned}$$

where the first inequality follows from the union bound; the second from the Markov inequality; and the subsequent ones from the fact that T is a constant, $d \geq 5$, and the size of the seed q is at most $d \cdot s$ bits. Thus, there is a non-zero probability of ending up with a π' such that all the corresponding $y_{\leq i}$-DFS are small. □

B Proof of Lemma 23a) and Lemma 23b)

We will begin by proving Lemma 23a). The proof of Lemma 23b) will follow from this and will be discussed in Section B.5. Let δ and ϵ be constants such that $\delta < 2\epsilon$. Without loss of generality, we can assume that both δ and ϵ are rational numbers. Initially, we will assume that $\delta > 0$ and will address the case where $\delta = 0$ separately in Section B.4.

Let $L \in \mathsf{BP}^{\epsilon}\mathsf{C}^{\delta}\mathsf{SPACE}(s, c)$, and let M be the machine that decides it. Let x be an arbitrary fixed input. We will describe how the subroutine $\mathcal{F}^L$ operates. There are various scenarios in which $\mathcal{F}^L$ compresses the catalytic tape, freeing at least the last bit in the process (as we will demonstrate, we will free more than this, up to $O(s)$ bits). For each of these scenarios, we will outline the method for decompressing the tape. Additionally, we will be able to determine which decompression method to apply based on a given compressed tape. This describes $\mathcal{R}^L$.

The structure of the catalytic tape used by $\mathcal{F}^L$ is depicted in Figure 4. The tape starts with c cells denoted by τ (which will be used to simulate the catalytic tape of M), followed by $\boldsymbol{l}$ hash functions, each specified using $\boldsymbol{2m}$ bits; followed by a *target* string **tar** of length $\boldsymbol{3m}$.

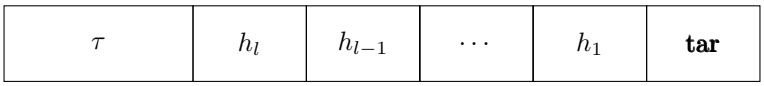

Figure 4: Catalytic tape

The hash functions on the catalytic tape are derived from a 2-independent hash family $\mathcal{H} = \{h : \{0,1\}^m \to \{0,1\}^m\}$ with a size of 2^{2m}, as in Nisan's PRG [Nis92]; where each hash function can be computed in $O(m)$ space.

B.1 The parameters

The parameters of interest are:

- $\boldsymbol{m}$ the seed length of the PRGs that will be built, which we will set later. We can think of $m = d \cdot s$, for a large constant $d \geq 500$.
- $\boldsymbol{l}$ the number of hash functions; with each specified by $2m$ bits. We set $l = 2^{20s}$.
- length of the string **tar**, which we set to $3m$.

- $\boldsymbol{H}$ (called upper-threshold), which we set to 2^{3m}.
- $\boldsymbol{T}$ (called lower-threshold), which we set to 2^{100s}.

Thus, the size of the tape used by $\mathcal{F}^L$ is

$$c + l \cdot 2m + 3m \leq 2^s + O(s) \cdot 2^{20s} + O(s) = 2^{O(s)}$$

as desired.

B.2 Building PRGs

The broad idea is as follows. $\mathcal{F}^L$ will build a sequence of pseudo-random generators: $\mathrm{PRG}_0, \mathrm{PRG}_1, \cdots, \mathrm{PRG}_l$, where PRG_0 is defined to output the empty string for every seed. For $1 \leq i \leq l$, PRG_i outputs a string of length i and is defined as

$$\mathrm{PRG}_i(seed) \coloneqq h_i(seed) \circ h_{i-1}(seed) \circ \cdots \circ h_1(seed) \tag{1}$$

where the hash functions are read from the catalytic tape. Here, we abuse the notation and use $h_i(seed) \circ h_{i-1}(seed) \circ \cdots \circ h_1(seed)$ to denote the i-bit length string obtained by concatenating the first bits outputted by each of the i hash functions. Note that since the hash functions can be computed in space $O(m) = O(s)$, we can compute these PRGs using $O(s)$ work space, since l, the number of hash functions, is $2^{O(s)}$. We would like PRG_i to approximate the probability of going from *any* configuration in layer i in the layered configuration graph of $G_{M,x}$ (recall Definition 54) to $\langle \mathrm{acc}_{M,x,\tau}, 0\rangle$ and $\langle \mathrm{rej}_{M,x,\tau}, 0\rangle$. To be precise, we want γ_i^{acc} and γ_i^{rej} to be "small", where

$$\begin{aligned}\gamma_i^{acc} \coloneqq \max_{\langle \pi,u,i\rangle} \Big\{ \big| &\mathbf{Pr}[\text{Going from } \langle \pi,u,i\rangle \text{ to } \langle \mathrm{acc}_{M,x,\tau}, 0\rangle] - \\ &\mathbf{Pr}_{seed}[y\text{-DFS from } \langle \mathrm{acc}_{M,x,\tau}, 0\rangle \text{ visits } \langle \pi,u,i\rangle \text{ for } y \leftarrow \mathrm{PRG}_i(\mathrm{seed})] \big| \Big\}\end{aligned} \tag{2}$$

and γ_i^{rej} is defined analogously. Here, $\langle \pi,u,i\rangle$ denotes an arbitrary configuration in layer i. In the equation above, the first probability refers to the likelihood of transitioning from $\langle \pi,u,i\rangle$ to $\langle \mathrm{acc}_{M,x,\tau}, 0\rangle$ in the layered configuration graph of $G_{M,x}$. The second probability is taken over a uniformly random seed for PRG_i, such that the y-DFS (see Definition 58) starting from $\langle \mathrm{acc}_{M,x,\tau}, 0\rangle$ visits $\langle \pi,u,i\rangle$, where y denotes the output of the PRG_i on a seed. Note that, by definition, the y-trees (see Definition 56) y-tree($\langle \mathrm{acc}_{M,x,\tau}, 0\rangle$) and y-tree($\langle \mathrm{rej}_{M,x,\tau}, 0\rangle$) for an empty string y are simply the isolated vertices $\langle \mathrm{acc}_{M,x,\tau}, 0\rangle$ and $\langle \mathrm{rej}_{M,x,\tau}, 0\rangle$. Additionally, using Definition 54, we know that vertices within the same layer do not have edges between them. Therefore, $\gamma_0^{acc} = \gamma_0^{rej} = 0$.

We define PRG_0 to be *good*, and we inductively define PRG_i to be good for $1 \leq i \leq l$ if PRG_{i-1} is good and h_i is *good*. We will define what it means for a hash function to be good later. We will describe $\mathcal{F}^L$ in an iterative fashion by assuming that for every $j \leq i$ (for some $0 \leq i < l$), PRG_j is good and that $\mathcal{F}^L$ has not yet compressed the catalytic tape. We will break down this description into various cases. As we will see, $\mathcal{F}^L$ may compress the catalytic tape if h_{i+1} is considered bad, among other possible scenarios, and then stop. Otherwise, it will iterate to the next value of i and repeat the same process. However, if $i = l$, $\mathcal{F}^L$ will either be able to determine the input x or will output $\perp$. We assume at this point the first c bits of the catalytic tape read τ (as in Figure 4).

Case 1: Size $> H$.

Definition 61 (Size of y-DFS)**.** *For a y-DFS from $\langle acc_{M,x,\tau}, 0\rangle$/$\langle rej_{M,x,\tau}, 0\rangle$, we define the size of the y-DFS to be the number of steps required to return to the halting configuration from which we initiated the y-DFS. In other words, it is the size of the y-tree($\langle acc_{M,x,\tau}, 0\rangle$)/$y$-tree($\langle rej_{M,x,\tau}, 0\rangle$).*

The first scenario occurs when the y-DFS from either $\langle \text{acc}_{M,x,\tau}, 0\rangle$ or $\langle \text{rej}_{M,x,\tau}, 0\rangle$ has a size that is greater than $H = 2^{3m}$ for some value of y (the output of PRG_i on a seed). In this situation, we will use timestamp compression [CLMP25] to compress the target string. We will check the size of a y-DFS (starting from $\langle \text{acc}_{M,x,\tau}, 0\rangle$ or $\langle \text{rej}_{M,x,\tau}, 0\rangle$) using the subroutine Size from Lemma 11 with the size parameter H.

Compression. By trying all the y-DFS (from $\langle \text{acc}_{M,x,\tau}, 0\rangle$ and $\langle \text{rej}_{M,x,\tau}, 0\rangle$), we can check if any of them has a size $> H$, while reusing the workspace required for Size. Without loss of generality, let the seed for which the y-DFS is large be q (i.e., $y \leftarrow \text{PRG}_i(q)$), and assume it was performed from $\langle \text{acc}_{M,x,\tau}, 0\rangle$. Let $val(\mathbf{tar})$ represent the integer value of the binary string **tar** incremented by one. Thus, $val(\mathbf{tar})$ is an integer in $[2^{3m}]$. Let $\langle \pi, u, k\rangle$ be the configuration reached by performing a y-DFS from $\langle \text{acc}_{M,x,\tau}, 0\rangle$ for $val(\mathbf{tar})$ steps using Next from Lemma 11. We compress by replacing the contents of the string **tar** with $00 \circ q \circ k \circ i \circ u \circ 0^{2m-41s-2}$. Note that q takes m bits, $0 \le k \le i \le l \le 2^{20s}$, and each can be described using $20s$ bits, while u (workspace description) takes s bits. We also replace τ with π in the first c cells of the catalytic tape. The prefix 00 indicates that we applied compression under the scenario "size $> H$" (since there will be other compression scenarios). Thus, we compress, freeing at least the last s bits of the catalytic tape, given that $m \ge \frac{42s+3}{2}$.

Decompression. To decompress, we perform a reverse y-DFS from $\langle \pi, u, k\rangle$, using StepBack from Lemma 11, counting the number of steps required to reach the unique halting configuration ($\langle \text{acc}_{M,x,\tau}, 0\rangle$ or $\langle \text{rej}_{M,x,\tau}, 0\rangle$). This count is equal to $val(\mathbf{tar})$. Therefore, we recover **tar** and reset the first c bits of our catalytic tape back to τ upon reaching $\langle \text{acc}_{M,x,\tau}, 0\rangle$.

Case 2: Size $\le H$

We now address the scenario in which every y-DFS according to PRG_i (from $\langle \text{acc}_{M,x,\tau}, 0\rangle$ or $\langle \text{rej}_{M,x,\tau}, 0\rangle$) has a size of at most H.

Definition 62. *We define V_i as the set of vertices, i.e., configurations $\langle v, i\rangle$ in layer i such that the following condition holds:*

$$\mathbf{Pr}_{seed}[y\text{-DFS from } \langle acc_{M,x,\tau}, 0\rangle \text{ or } \langle rej_{M,x,\tau}, 0\rangle \text{ visits } \langle v, i\rangle \text{ for } y \leftarrow PRG_i(seed)] > 0$$

In other words, V_i is the collection of vertices in layer i that have a non-zero probability (over seed) of being reached via a y-DFS, as determined by PRG_i, starting from $\langle acc_{M,x,\tau}, 0\rangle$ or $\langle rej_{M,x,\tau}, 0\rangle$.

Definition 63. *We define S_i as the set of vertices, i.e., configurations $\langle v, i\rangle$ in layer i such that the following holds:*

$$\mathbf{Pr}_{seed}[y\text{-DFS from } \langle acc_{M,x,\tau}, 0\rangle \text{ or } \langle rej_{M,x,\tau}, 0\rangle \text{ visits } \langle v, i\rangle \text{ for } y \leftarrow PRG_i(seed)] \ge \frac{1}{2^{30s}}.$$

In other words, S_i is the set of vertices in layer i that can be reached from $\langle acc_{M,x,\tau}, 0\rangle$ or $\langle rej_{M,x,\tau}, 0\rangle$ for at least $\frac{1}{2^{30s}}$ fraction of the y-DFS, as determined by PRG_i.

Note that by definition, $S_i \subseteq V_i$.

If $|S_i| \geq T$, we will proceed to subcase 2.1; otherwise, we will move to subcase 2.2. Before diving into these subcases, we will describe some subroutines that will help us check this condition and will also be useful in subsequent steps. These subroutines use the first c bits of the catalytic tape to simulate that of M.

- $\textsc{Count}_{V_i}$. Computes $|V_i|$.
- $\textsc{Index}_{V_i}$. Given $j \in [\textsc{Count}_{V_i}]$, the procedure outputs a seed q, an integer $t \geq 0$, and a binary value $b \in \{0, 1\}$. This indicates that the j-th vertex in V_i (according to a specific numbering) can be reached by performing a y-DFS from $\langle \text{acc}_{M,x,\tau}, 0\rangle$ for t steps if $b = 1$. Conversely, if $b = 0$, the j-th vertex can be reached by performing the y-DFS from $\langle \text{rej}_{M,x,\tau}, 0\rangle$ for t steps, where y is given by $\text{PRG}_i(q)$. Moreover, t is upper-bounded by the size of the corresponding y-DFS.
- $\textsc{Count}_{S_i}$. Computes $|S_i|$.
- $\textsc{Index}_{S_i}$. Given $j \in [\textsc{Count}_{S_i}]$, it works analogously to $\textsc{Index}_{V_i}$.

Claim 64. *Given that all the y-DFS from $\langle acc_{M,x,\tau}, 0\rangle$ and $\langle rej_{M,x,\tau}, 0\rangle$, as per PRG_i, have sizes of at most $B = 2^{O(s)}$[4], the subroutines $\textsc{Count}_{V_i}$, $\textsc{Count}_{S_i}$, $\textsc{Index}_{V_i}$, and $\textsc{Index}_{S_i}$ can be executed using $O(s)$ work space. Furthermore, these subroutines run in time $2^{O(s)}$. Additionally, they always restore the first c bits of the catalytic tape they utilize, to their original contents.*

Proof. First, given i, a bit b, seed g, $t \leq B$, and $k \in [c + O(s)]$, we can determine the k-bit of the vertex reached by procedure Next (from Lemma 11) repeated t-times starting from $\langle \text{acc}_{M,x,\tau}, 0\rangle$ if $b = 1$ or $\langle \text{rej}_{M,x,\tau}, 0\rangle$ if $b = 0$ in the y-DFS, where $y \leftarrow \text{PRG}_i(g)$. Then, by running StepBack t-times, we can also restore the catalytic tape. This uses $O(s)$ work space and time $2^{O(s)}$. Hence, for any two given triples (b, g, t) and (b', g', t'), we can determine whether they refer to the same vertex by comparing the two vertices bit by bit.

To count the size of V_i, we just need to determine how many triples (b, g, t) give distinct vertices. To do so, we initialize a counter ***count*** to 0 and then enumerate over triples (b, g, t) in lexicographical order and check for each of them whether some lexicographically smaller triple gives the same vertex. If not, and the vertex given by (b, g, t) is in layer i (which can be determined from its bit description), we increment ***count***. In either case, we proceed to the next triple (b, g, t). Clearly, this counts the size of V_i. This can be implemented using a workspace of size $O(s)$ via two nested for-loops over the triples. The catalytic space is restored to its initial content, and there are no other modifications to the space other than the internal workspace. To identify the triple (b, g, t) corresponding to a given index $j \in [\textsc{Count}_{V_i}]$, we can run the same procedure as above but terminate it once ***count*** reaches j. At that point, we have the triple (b, g, t) on our tape. This implements the two procedures $\textsc{Count}_{V_i}$ and $\textsc{Index}_{V_i}$.

To determine whether a given triple (b, g, t) refers to a vertex from S_i we need to check for how many seeds g' there is some $t' \leq B$ and $b' \in \{0, 1\}$, so that (b, g, t) and (b', g', t') refer to the same vertex. Again, this condition can be checked using internal work space $O(s)$. By going over $j \in [\textsc{Count}_{V_i}]$, finding its corresponding triple (b, g, t) using $\textsc{Index}_{V_i}$ and testing whether it belongs to S_i we can count the size of S_i using internal work space $O(s)$. This implements $\textsc{Count}_{S_i}$. To implement $\textsc{Index}_{S_i}$, we can output (b, g, t) that is reached when the counter to compute $\textsc{Count}_{S_i}$ reaches the desired value. □

[4]where B is logspace-constructible.

Since all the y-DFS from $\langle \text{acc}_{M,x,\tau}, 0\rangle$ and $\langle \text{rej}_{M,x,\tau}, 0\rangle$ have sizes of at most $H = 2^{3m} = 2^{O(s)}$, using the previous claim, we can utilize $\textsc{Count}_{S_i}$ to check if $|S_i| \geq T = 2^{100s}$ using $O(s)$ workspace and in $2^{O(s)}$ time without altering the contents of the catalytic tape.

Case 2.1: $|S_i| \geq T$. In this scenario, we again compress the target string. We use $val(\mathbf{tar}_{\leq \log T})$ to denote the integer value of the first $\log T$ bits of **tar**, incremented by 1, which is an integer in $[T]$. Here, we assume that $3m \geq \log T = 100s$. Let $\langle \pi, u, i\rangle$ be the vertex/configuration in S_i with index $val(\mathbf{tar}_{\leq \log T})$, as per the subroutine $\textsc{Index}_{S_i}$.

Let us consider performing a y-DFS starting from $\langle \pi, u, i\rangle$ for each y generated by $\text{PRG}_i(seed)$, using all possible seeds, with each run limited to H steps. For every y-DFS, we will visit at most one (distinct) configuration in **layer** 0, which will be the unique sink of the corresponding y-tree. Let TAU represent the set of all c-bit catalytic settings such that $\pi' \in \text{TAU}$ iff, for at least $\frac{1}{2^{30s}}2^m$ seeds, the y-DFS from $\langle \pi, u, i\rangle$ visits either $\langle \text{acc}_{M,x,\pi'}, 0\rangle$ or $\langle \text{rej}_{M,x,\pi'}, 0\rangle$, within H steps. Then, it follows that $|\text{TAU}| \leq 2^{30s}$. We know that for at least $\frac{1}{2^{30s}}$ fraction of the seeds, the y-DFS from $\langle \text{acc}_{M,x,\tau}, 0\rangle$ or $\langle \text{rej}_{M,x,\tau}, 0\rangle$ visits $\langle \pi, u, i\rangle$, since this configuration is in S_i. Let us denote this set of seeds as I—the set of seeds for which the y-DFS from $\langle \text{acc}_{M,x,\tau}, 0\rangle$ or $\langle \text{rej}_{M,x,\tau}, 0\rangle$ reaches $\langle \pi, u, i\rangle$. By definition, we have $|I| \geq \frac{1}{20^{30s}}2^m$. Since we are in a scenario where all y-DFS from $\langle \text{acc}_{M,x,\tau}, 0\rangle$ and $\langle \text{rej}_{M,x,\tau}, 0\rangle$ have a size no greater than H, by applying Lemma 11, we can conclude that we will visit $\langle \text{acc}_{M,x,\tau}, 0\rangle$ or $\langle \text{rej}_{M,x,\tau}, 0\rangle$ from $\langle \pi, u, i\rangle$ while performing a y-DFS within H steps, with y being the output of PRG_i for seeds in I. This implies that $\tau \in \text{TAU}$. We will index the catalytic settings π' in TAU according to the lexicographic order of the first seed for which $\langle \text{acc}_{M,x,\pi'}, 0\rangle$ or $\langle \text{rej}_{M,x,\pi'}, 0\rangle$ is visited while performing a y-DFS from $\langle \pi, u, i\rangle$ (for H steps).

Compression. First, we run the subroutine $\textsc{Index}_{S_i}$ with input $val(\mathbf{tar}_{\leq \log T})$. This gives a seed q, an integer $t \geq 0$, and a binary value b (which we, without loss of generality, assume to be 1). Since all the y-DFS are of size at most H, it follows from Claim 64 that we can save q, t using $O(s)$ work space. This allows us to visit the configuration $\langle \pi, u, i\rangle$, which changes the contents τ on the catalytic tape to π. With q and t stored in our work space, we can always return to $\langle \text{acc}_{M,x,\tau}, 0\rangle$ by performing a y-DFS from this point, corresponding to the seed q. Effectively, we are now "standing" at the configuration $\langle \pi, u, i\rangle$. Let the index of τ in TAU be denoted as idx; therefore, idx can be specified using $30s$ bits. We will outline how to compute idx.

(a) For any seed, we can check if the y-DFS from $\langle \pi, u, i\rangle$ visits a halting configuration in layer 0 within H steps and then walk back to $\langle \pi, u, i\rangle$ by performing the y-DFS in reverse order, all the while keeping track of the number of steps we walk in any direction. If we see a halting configuration in layer 0, we will, for ease, call the corresponding seed *halting*.

(b) For any two halting seeds $seed_1$ and $seed_2$ (which can be verified using (a)), we can determine if both the y_1-DFS and y_2-DFS—where $y_1 \leftarrow \text{PRG}_i(seed_1)$ and $y_2 \leftarrow \text{PRG}_i(seed_2)$—encounter a halting configuration in layer 0 with the same catalytic contents. To do this, we first modify point (a) so that, in addition to checking if a seed is halting, it also reports the j-th bit of the catalytic part of the halting configuration that is observed, for any index $j \in [c]$. We can then perform a bit-by-bit comparison of the catalytic parts of the halting configurations from the two y-DFS, while continuously returning to $\langle \pi, u, i\rangle$. If the catalytic contents match, we classify the two seeds as being *similar*.

(c) For any halting seed, using (b), we can count how many seeds are similar to it (including it). If there are more than $\frac{1}{2^{30s}}2^m$, we call the seed *relevant*.

(d) We keep a counter ***count*** initialized to 0. For each seed, we check if it is halting, relevant, and not similar to any seed that is lexicographically smaller than it. If so, we increment *count*. Let us suppose we are at a seed for which the counter was increased. Since it's a halting seed, let the contents of the halt configuration seen be π'. Then, by definition, $\pi' \in$ TAU (since the seed is relevant), and the value of *count* at this point is its index in TAU (since it is not similar to any lexicographically smaller seed).

(e) Finally, we compute idx as follows. Let $seed_1$ be any seed for which we have incremented *count*. Let the catalytic contents of the halt configuration observed during a y_1-DFS from $\langle \pi, u, i \rangle$, using $y_1 \leftarrow \mathrm{PRG}_i(seed_1)$, be π'. We can verify whether π' is identical to τ again in a bit-by-bit manner, similar to the method described in part (b). This verification is feasible because we have q and t stored, which allows us to move back and forth between $\langle \mathrm{acc}_{M,x,\tau}, 0 \rangle$ and $\langle \pi, u, i \rangle$. If $\pi' = \tau$, we declare idx as the value of *count*.

To compress, we replace the first c cells of the catalytic tape with π (recall we are effectively standing at $\langle \pi, u, i \rangle$); and we replace $\mathbf{tar}_{\leq \log T}$ with $01 \circ idx \circ i \circ u \circ 0^{49s-2}$. This is feasible as $\log T = 100s$ and

$$2 + |idx| + |i| + |u| \leq 2 + 30s + 20s + s = 51s + 2$$

where we used that $i \leq l = 2^{20s}$ and thus can be specified using $20s$ bits. The leading 01 specifies this type of compression.

Decompression. Given the compressed tape, we find ourselves in the configuration $\langle \pi, u, i \rangle$. Knowing the value of i provides us access to PRG_i. By using step (d) of the compression process, we can determine a seed q such that the y-DFS from $\langle \pi, u, i \rangle$, for $y \leftarrow \mathrm{PRG}_i(q)$, will visit either $\langle \mathrm{acc}_{M,x,\tau}, 0 \rangle$ or $\langle \mathrm{rej}_{M,x,\tau}, 0 \rangle$, within H steps. Without loss of generality, we can assume it visits $\langle \mathrm{acc}_{M,x,\tau}, 0 \rangle$. To be precise, q is the seed for which the counter *count* in step (d) of the compression becomes equal to idx, which we know. As a result, we can effectively recover the τ part of the catalytic tape. Furthermore, we can move back and forth between the configurations $\langle \pi, u, i \rangle$ and $\langle \mathrm{acc}_{M,x,\tau}, 0 \rangle$. We are now left with the task of recovering the contents of **tar**. For this, we utilize the subroutines: $\textsc{Count}_{S_i}$ and $\textsc{Index}_{S_i}$. For each $j \in [\textsc{Count}_{S_i}]$, we invoke the subroutine $\textsc{Index}_{S_i}$, which provides us with a seed q', an integer $t \geq 0$, and $b \in \{0, 1\}$ (without loss of generality, we can assume $b = 1$). Consequently, we can move back and forth between $\langle \mathrm{acc}_{M,x,\tau}, 0 \rangle$ and the j-th vertex in S_i, denoted as $\langle \pi', u', i \rangle$. Thus, we can check in a bit-by-bit manner whether $\langle \pi', u', i \rangle$ is the same as $\langle \pi, u, i \rangle$—that is, we verify if $\pi = \pi'$ and $u = u'$. Once we find such a j, we know its value is $val(\mathbf{tar}_{\leq \log T})$, which allows us to recover the compressed contents of **tar** as well.

We observe that, since we are in a situation where all the y-DFS from $\langle \mathrm{acc}_{M,x,\tau}, 0 \rangle$ and $\langle \mathrm{rej}_{M,x,\tau}, 0 \rangle$ are small, using Claim 64, it can be verified that we can perform the compression and decompression processes using $O(s)$ workspace and with time $2^{O(s)}$. Also note that we can always change the compression to free up the last bit(s) on the catalytic tape by swapping the freed-up 0^{49s-2} bits with the last $49s - 2$ bits of **tar**.

Case 2.2: $|S_i| \leq T$. In this scenario, we will check whether the hash function h_{i+1} is good (defined shortly). If the hash function is bad, we will compress the catalytic tape again. Otherwise, we will move to the next iteration $i + 1$. Before we define the goodness of a hash function, we will need the following definitions.

Definition 65. *Let $\langle v, i \rangle$ and $\langle z, i + 1 \rangle$ be two arbitrary vertices/configurations in layers i and $i + 1$, respectively, of the layered configuration graph of $G_{M,x}$. We introduce the following sets:*

- $A_v^{acc} = \{$*seeds in* $\{0,1\}^m$ *for which y-DFS from* $\langle acc_{M,x,\tau}, 0\rangle$ *visits* $\langle v, i\rangle$*, for* $y \leftarrow PRG_i(seed)\}$. *We similarly define* A_v^{rej}*, where the definition uses* $\langle rej_{M,x,\tau}, 0\rangle$ *instead of* $\langle acc_{M,x,\tau}, 0\rangle$.
- $B_{vz} = \{t \in \{0,1\}^m \mid$ *there is an edge with label* t_1 *(first bit of* t*) from* $\langle z, i+1\rangle$ *to* $\langle v, i\rangle\}$. *Thus,* $|B_{vz}|$ *is either* 0*,* 2^{m-1}*, or* 2^m.

Definition 66 (Neighborhood)**.** *We define* $N_{i+1}(\cdot)$ *as the function that identifies the neighborhood in layer* $i+1$*. Specifically, for any subset of vertices* Y *in layer* i*,* $N_{i+1}(Y)$ *denotes the set of vertices* $\langle w, i+1\rangle$ *in layer* $i+1$ *that have at least one outgoing edge directed toward a vertex in* Y*. Moreover, we will use the notation* $N_{i+1}(\langle v, i\rangle)$ *to represent* $N_{i+1}(\{\langle v, i\rangle\})$*, where* $\langle v, i\rangle$ *is a vertex in layer* i.

Note that it follows from the definitions of S_i, V_i (Definition 62 and Definition 63) that $N_{i+1}(S_i) \subseteq N_{i+1}(V_i)$.

Definition 67 ((A, B, α)-independence [Nis92])**.** *Let* $\mathcal{H} = \{h : \{0,1\}^m \to \{0,1\}^m\}$ *be a pairwise independent hash family, and let* $A, B \subseteq \{0,1\}^m$*. A hash function* $h : \{0,1\}^m \to \{0,1\}^m$ *is called* (A, B, α)*-independent if*

$$\left|\mathbf{Pr}_{x\in\{0,1\}^m}[x \in A \text{ and } h(x) \in B] - \rho(A)\rho(B)\right| \leq \alpha$$

Here $\rho(A) = \frac{|A|}{2^m}$*, and* $\rho(B) = \frac{|B|}{2^m}$.

Definition 68 (Good hash function)**.** *A hash function from the hash family is considered* good *if it is*

1. $(A_v^{acc}, B_{vz}, \alpha)$*-independent for every* $\langle v, i\rangle \in S_i$ *and* $\langle z, i+1\rangle \in N_{i+1}(\langle v, i\rangle)$.
2. $(A_v^{rej}, B_{vz}, \alpha)$*-independent for every* $\langle v, i\rangle \in S_i$ *and* $\langle z, i+1\rangle \in N_{i+1}(\langle v, i\rangle)$.

where $\boldsymbol{\alpha}$ *is a parameter whose value will be set later.*

We now break down the analysis further into two sub-cases, depending on whether h_{i+1} is good.

Case 2.2.1**:** h_{i+1} **is good.** In this case, we increment i and start over with Case 1. However, if i becomes l, we will proceed to Case 3. This assumes that we can determine whether h_{i+1} is good; we will discuss this in more detail in the next sub-case. Before we proceed, we need to analyze how the approximation errors γ_{i+1}^{acc} and γ_{i+1}^{rej}, as described in Equation 2, grow. This analysis will help us determine the parameter α and the seed length m later on. We will focus our analysis on γ_{i+1}^{acc}, as the analysis for γ_{i+1}^{rej} is identical.

Consider an arbitrary vertex $\langle z, i+1\rangle$ in layer $i+1$. Let the two outgoing transitions from $\langle z, i+1\rangle$ into layer i go to vertices $\langle a, i\rangle$ and $\langle b, i\rangle$ (for the ease of analysis, we assume $a \neq b$; though they can be the same). Then, we know that:

$$\begin{aligned}&\mathbf{Pr}[\text{Going from } \langle z, i+1\rangle \text{ to } \langle \text{acc}_{M,x,\tau}, 0\rangle] = \\ &\frac{1}{2}\mathbf{Pr}[\text{Going from } \langle a, i\rangle \text{ to } \langle \text{acc}_{M,x,\tau}, 0\rangle] + \frac{1}{2}\mathbf{Pr}[\text{Going from } \langle b, i\rangle \text{ to } \langle \text{acc}_{M,x,\tau}, 0\rangle]\end{aligned}$$

whereby the first probability refers to the probability of transitioning from $\langle z, i+1\rangle$ to $\langle \text{acc}_{M,x,\tau}, 0\rangle$ in the layered configuration graph of $G_{M,x}$ (see Definition 54); the other probabilities are similarly defined.

Remark 69 (Notation)**.** *When we write a quantity $q \pm \varepsilon$, we mean a value that differs from q by at most ε.*

Using Equation 2, Definition 65, and the definition of $\rho(\cdot)$ as per Definition 67, we have

$$\begin{aligned}\mathbf{Pr}[\text{Going from } \langle z, i+1\rangle \text{ to } \langle \text{acc}_{M,x,\tau}, 0\rangle] &= \frac{1}{2}\Big(\rho(A_a^{acc}) \pm \gamma_i^{acc}\Big) + \frac{1}{2}\Big(\rho(A_b^{acc}) \pm \gamma_i^{acc}\Big) \\ &\frac{1}{2}\rho(A_a^{acc}) + \frac{1}{2}\rho(A_b^{acc}) \pm \gamma_i^{acc}\end{aligned} \tag{3}$$

Note, by definition B_{az} consists of strings starting with 0, and B_{bz} consists of strings starting with 1 (or the other way around); therefore $B_{az} \cap B_{bz} = \phi$. Using the fact that $\text{PRG}_{i+1}(seed) = h_{i+1}(seed) \circ \text{PRG}_i(seed)$ it follows from the definition of y-DFS (Definition 58) that

$$\begin{aligned}&\mathbf{Pr}_{seed}[y\text{-DFS from } \langle \text{acc}_{M,x,\tau}, 0\rangle \text{ visits } \langle z, i+1\rangle \text{ for } y \leftarrow \text{PRG}_{i+1}(\text{seed})] \\ &= \mathbf{Pr}_{seed}[seed \in A_a^{acc} \text{ and } h_{i+1}(seed) \in B_{az}] + \mathbf{Pr}_{seed}[seed \in A_b^{acc} \text{ and } h_{i+1}(seed) \in B_{bz}] \\ &\leq \mathbf{Pr}_{seed}[seed \in A_a^{acc}] + \mathbf{Pr}_{seed}[seed \in A_b^{acc}] \\ &= \rho(A_a^{acc}) + \rho(A_b^{acc})\end{aligned} \tag{4}$$

To simplify our notation, we define $actual = \mathbf{Pr}[\text{Going from } \langle z, i+1\rangle \text{ to } \langle \text{acc}_{M,x,\tau}, 0\rangle]$ and $approx = \mathbf{Pr}_{seed}[y\text{-DFS from } \langle \text{acc}_{M,x,\tau}, 0\rangle \text{ visits } \langle z, i+1\rangle \text{ for } y \leftarrow \text{PRG}_{i+1}(seed)]$. Hence, we are interested in upper-bounding $|actual - approx|$, which provides an upper-bound on γ_{i+1}^{acc} since $\langle z, i+1\rangle$ is an arbitrary vertex in layer $i+1$. We consider an exhaustive list of possibilities:

- $\boldsymbol{\langle z, i+1\rangle \notin N_{i+1}(V_i)}$. Then, $\langle a, i\rangle, \langle b, i\rangle \notin V_i$. Thus, it follows from Definition 62 and Definition 65 that $\rho(A_a^{acc}) = \rho(A_b^{acc}) = 0$. Therefore, $actual \leq \gamma_i^{acc}$ by Equation 3. Moreover, by Equation 4 we get that $approx = 0$. Therefore, we have:

$$|actual - approx| \leq \gamma_i^{acc} \tag{5}$$

- $\boldsymbol{\langle z, i+1\rangle \in N_{i+1}(V_i) \backslash N_{i+1}(S_i)}$ **and** $\boldsymbol{\langle a, i\rangle, \langle b, i\rangle \in V_i \backslash S_i}$. Using Equation 3 we know that

$$|\ actual - \frac{1}{2}\big(\rho(A_a^{acc}) + \rho(A_b^{acc})\big)\ | \leq \gamma_i^{acc} \tag{6}$$

  Thus, we have:

$$\begin{aligned}|actual - approx| &= |actual - \frac{1}{2}\big(\rho(A_a^{acc}) + \rho(A_b^{acc})\big) + \frac{1}{2}\big(\rho(A_a^{acc}) + \rho(A_b^{acc})\big) - approx| \\ &\leq |\frac{1}{2}\big(\rho(A_a^{acc}) + \rho(A_b^{acc})\big) - approx| + |actual - \frac{1}{2}\big(\rho(A_a^{acc}) + \rho(A_b^{acc})\big)| \\ &\leq \frac{1}{2}(\rho(A_a^{acc}) + \rho(A_b^{acc})) + \gamma_i^{acc} \\ &\leq \frac{1}{2^{30s}} + \gamma_i^{acc}\end{aligned} \tag{7}$$

  The second-to-last inequality follows from Equation 4, the fact that approx ≥ 0, and Equation 6. The final inequality is grounded in the observation that $\rho(A_a^{acc})$ and $\rho(A_b^{acc})$ are both less than $\frac{1}{2^{30s}}$. This conclusion follows from the definitions of S_i and the fact that $\langle a, i\rangle$ and $\langle b, i\rangle$ belong to $V_i \setminus S_i$.

- $\boldsymbol{\langle z, i+1\rangle \in N_{i+1}(V_i)\backslash N_{i+1}(S_i)}$ **such that wlog.** $\boldsymbol{\langle a, i\rangle \in V_i\backslash S_i}$ **and** $\boldsymbol{\langle b, i\rangle \notin V_i}$. Since $\langle b, i\rangle \notin V_i$, $\rho(A_b^{acc}) = 0$. Equation 3 gives us that

$$|actual - \frac{1}{2}\rho(A_a^{acc})| \leq \gamma_i^{acc}$$

and Equation 4 gives that

$$approx \leq \rho(A_a^{acc})$$

Therefore, similar to the previous case, we get

$$\begin{aligned} |actual - approx| &\leq \frac{1}{2}\rho(A_a^{acc}) + \gamma_i^{acc} \\ &\leq \frac{1}{2} \cdot \frac{1}{2^{30s}} + \gamma_i^{acc} \end{aligned} \tag{8}$$

where the last inequality follows from the fact that $\langle a, i\rangle \in V_i\backslash S_i$ and thus $\rho(A_a^{acc}) < \frac{1}{2^{30s}}$.

- $\boldsymbol{\langle z, i+1\rangle \in N_{i+1}(S_i)}$ **and** $\boldsymbol{\langle a, i\rangle, \langle b, i\rangle \in S_i}$. Since by our assumption a and b are different configurations, $|B_{az}| = |B_{bz}| = 2^{m-1}$. Thus we can rewrite Equation 3 as

$$actual = \rho(B_{az})\rho(A_a^{acc}) + \rho(B_{bz})\rho(A_b^{acc}) \pm \gamma_i^{acc} \tag{9}$$

Since $\langle z, i+1\rangle \in N_{i+1}(\langle a, i\rangle) \cap N_{i+1}(\langle b, i\rangle)$ and both $\langle a, i\rangle$ and $\langle b, i\rangle$ are in S_i, and since we are in the case where h_{i+1} is good, using Definition 68 we obtain that

$$\begin{aligned} |\mathbf{Pr}_{seed}[seed \in A_a^{acc} \text{ and } h_{i+1}(seed) \in B_{az}] - \rho(B_{az})\rho(A_a^{acc})| \leq \alpha \\ |\mathbf{Pr}_{seed}[seed \in A_b^{acc} \text{ and } h_{i+1}(seed) \in B_{bz}] - \rho(B_{bz})\rho(A_b^{acc})| \leq \alpha \end{aligned} \tag{10}$$

Using Equation 9 and Equation 10 we get

$$\begin{aligned} actual = \\ &\mathbf{Pr}_{seed}[seed \in A_a^{acc} \text{ and } h_{i+1}(seed) \in B_{az}] \\ &+ \mathbf{Pr}_{seed}[seed \in A_b^{acc} \text{ and } h_{i+1}(seed) \in B_{bz}] \pm (\gamma_i^{acc} + 2\alpha) \end{aligned} \tag{11}$$

Using Equation 4 and Equation 11 we get

$$|actual - approx| \leq \gamma_i^{acc} + 2\alpha \tag{12}$$

- $\boldsymbol{\langle z, i+1\rangle \in N_{i+1}(S_i)}$ **such that wlog.** $\boldsymbol{\langle a, i\rangle \in S_i}$ **and** $\boldsymbol{\langle b, i\rangle \notin V_i}$. As $\langle b, i\rangle \notin V_i$, $\rho(A_b^{acc}) = 0$, and thus Equation 9 gives us that

$$actual = \rho(A_a^{acc})\rho(B_{az}) \pm \gamma_i^{acc}$$

and using Equation 4 we have

$$approx = \mathbf{Pr}_{seed}[seed \in A_a^{acc} \text{ and } h_{i+1}(seed) \in B_{az}]$$

As $\langle z, i+1\rangle \in N_{i+1}(\langle a, i\rangle)$ and $\langle a, i\rangle \in S_i$, by the goodness of h_{i+1}, we have

$$|\mathbf{Pr}_{seed}[seed \in A_a^{acc} \text{ and } h_{i+1}(seed) \in B_{az}] - \rho(B_{az})\rho(A_a^{acc})| \leq \alpha$$

Therefore, using the above, we get

$$actual = \mathbf{Pr}_{seed}[seed \in A_a^{acc} \text{ and } h_{i+1}(seed) \in B_{az}] \pm (\gamma_i^{acc} + \alpha)$$

i.e.,

$$|actual - approx| \leq \gamma_i^{acc} + \alpha$$

- $\boldsymbol{\langle z, i+1\rangle \in N_{i+1}(S_i)}$ **such that wlog** $\boldsymbol{\langle a, i\rangle \in S_i}$ **and** $\boldsymbol{\langle b, i\rangle \in V_i \backslash S_i}$. Let $Y = \big(\rho(B_{az})\rho(A_a^{acc}) + \rho(B_{bz})\rho(A_b^{acc})\big)$. We have that

$$\begin{aligned}
|actual - approx| &= |actual - Y + Y - approx| \\
&\leq |approx - Y| + |actual - Y| \\
&\leq |approx - Y| + \gamma_i^{acc} \\
&= | \ \mathbf{Pr}[seed \in A_a^{acc} \text{ and } h_{i+1}(seed) \in B_{az}] + \\
&\quad + \mathbf{Pr}[seed \in A_b^{acc} \text{ and } h_{i+1}(seed) \in B_{bz}] - Y \ | + \gamma_i^{acc} \\
&\leq |\mathbf{Pr}[seed \in A_a^{acc} \text{ and } h_{i+1}(seed) \in B_{az}] - \rho(B_{az})\rho(A_a^{acc})| \\
&\quad + |\mathbf{Pr}[seed \in A_b^{acc} \text{ and } h_{i+1}(seed) \in B_{bz}] - \rho(B_{bz})\rho(A_b^{acc})| + \gamma_i^{acc} \\
&\leq \alpha + |\mathbf{Pr}[seed \in A_b^{acc} \text{ and } h_{i+1}(seed) \in B_{bz}] - \rho(B_{bz})\rho(A_b^{acc})| + \gamma_i^{acc} \\
&= |\mathbf{Pr}[seed \in A_b^{acc} \text{ and } h_{i+1}(seed) \in B_{bz}] - \frac{1}{2}\rho(A_b^{acc})| + \alpha + \gamma_i^{acc} \\
&\leq \max\{\mathbf{Pr}[seed \in A_b^{acc} \text{ and } h_{i+1}(seed) \in B_{bz}], \frac{1}{2}\rho(A_b^{acc})\} + \alpha + \gamma_i^{acc} \\
&\leq \max\{\rho(A_b^{acc}), \frac{1}{2}\rho(A_b^{acc})\} + \alpha + \gamma_i^{acc} \\
&= \rho(A_b^{acc}) + \alpha + \gamma_i^{acc} \\
&\leq \frac{1}{2^{30s}} + \alpha + \gamma_i^{acc}
\end{aligned} \tag{13}$$

where the second inequality follows from Equation 9; the second equality from Equation 4; the fourth inequality from the goodness of h_{i+1}; third equality from the fact that $|B_{bz}| = 2^{m-1}$; fifth inequality from the fact that the terms in absolute difference are non-negative quantities; and the last inequality from the fact that $\langle b, i\rangle \in V_i \backslash S_i$ and thus $\rho(A_b^{acc}) < \frac{1}{2^{30s}}$.

Thus, in summary, if h_{i+1} is good, we get that $\gamma_{i+1}^{acc} \leq \frac{1}{2^{30s}} + \gamma_i^{acc} + 2\alpha$. By considering $\langle \text{rej}_{M,x,\tau}, 0\rangle$ instead of $\langle \text{acc}_{M,x,\tau}, 0\rangle$ and repeating the same analysis as above, one can show that the same recurrence relation holds for γ_{i+1}^{rej} as well.

Setting parameters α **and** m. We pause our description of further cases to first establish the values for the seed length m and the parameter α in Definition 68 with foresight. In the scenario where $\mathcal{F}^L$ does not compress the catalytic tape, and all the hash functions h_1 to h_l are good in their respective iterations, we would like to ensure that $\gamma_l = \max\{\gamma_l^{acc}, \gamma_l^{rej}\}$—which we refer to as the *error* of PRG_l—is small. Since in this case all hash functions are good, we can use the recurrence $\gamma_{i+1}^{acc} \leq \frac{1}{2^{30s}} + \gamma_i^{acc} + 2\alpha$ (and the same for γ_{i+1}^{rej}). Since $\gamma_0^{acc} = \gamma_0^{rej} = 0$, we derive that γ_l is at most

$$l \cdot \left(2\alpha + \frac{1}{2^{30s}}\right) \tag{14}$$

By setting $\alpha = \frac{1}{2l^2} = \frac{1}{2 \cdot 2^{40s}}$, we find from Equation 14 that γ_l is at most $\frac{1}{2^{20s}} + \frac{2^{20s}}{2^{30s}}$. This is at most $\frac{1}{2^{5s}}$ for sufficiently large s. The following lemma assists in bounding the fraction of bad hash functions, which will aid in compressing the catalytic tape if h_{i+1} is bad.

Lemma 70 (Nisan [Nis92])**.** *Let $\mathcal{H} = \{h : \{0,1\}^m \to \{0,1\}^m\}$ be a pairwise independent hash family, and let $A, B \subseteq \{0,1\}^m$. Then, the probability h chosen from $\mathcal{H}$ uniformly at random is not (A, B, α)-independent is less than or equal to $\frac{\rho(A)\rho(B)(1-\rho(B))}{2^m\alpha^2}$.*

Corollary 71. *The probability that a hash function (chosen uniformly at random from the family) is bad is upper-bounded by $\frac{2d_M \cdot T}{2^m\alpha^2}$, where d_M is a constant that depends on the machine M.*

Proof. In the layered configuration graph of $G_{M,x}$, the number of incoming edges to any vertex or configuration is bounded by the constant d_M, which represents the number of edges incident to any vertex. Thus, given that we are in the case where $|S_i| \leq T$, we can apply Lemma 70 and use the union bound to upper-bound the probability that a randomly chosen hash function from the family is not good by:

$$\frac{1}{2^m\alpha^2}\left(2d_M|S_i|\right) \leq \frac{2d_M \cdot T}{2^m\alpha^2} \tag{15}$$

□

We set $m = 500s$. Thus, substituting the values of T, α, and m in Corollary 71, we can upper bound the fraction of hash functions that are not good (in a specific iteration) by

$$\frac{8 \cdot d_M \cdot 2^{180s}}{2^{500}} \tag{16}$$

which is at most $\frac{1}{2^{200s}}$ for sufficiently large s.

Note that our choice of seed length satisfies the constraints we encountered previously, as $m > \max\left\{\frac{42s+3}{2}, \frac{100s}{3}\right\}$. Additionally, with this seed length, we have $L = 2^{100s} < H = 2^{3m} = 2^{1500s}$, ensuring that the thresholds make sense.

We now resume the description of the further cases.

Case 2.2.2**: h_{i+1} is bad.** We now consider the scenario when the hash function h_{i+1} is bad. In this case, we compress the catalytic tape by reusing the idea from Pyne [Pyn25], which was used to compress bad hash functions in the context of de-randomizing BPL in a catalytic setting.

We first assume that we can efficiently check whether a hash function from the family is good or bad, and describe the compression scheme.

Compression. We iterate over all the hash functions from the family up to h_{i+1} in lexicographical order and count how many bad hash functions exist. This allows us to determine the index idx of h_{i+1} among the set of bad hash functions, which corresponds to the total count at the end of this process. Each hash function is represented by $2m = 1000s$ bits; therefore, we can iterate through them using $O(s)$ space and in $2^{O(s)}$ time. Based on our choice of seed length, we know that the fraction of bad functions, as indicated in Equation 16, is at most $\frac{1}{2^{200s}}$. Consequently, the index idx can be represented using $800s$ bits, which frees up $200s$ bits. In other words, we replace h_{i+1} with $idx \circ 0^{200s}$ on the catalytic tape. As we cannot remember which hash function we compressed, we copy the contents of the first $20s + 2$ bits of **tar** (recall that the length of **tar** is $3m = 1500s$), given by $\mathbf{tar}_{\leq 20s+2}$, into the freed-up $200s$ bits. That is, we change the contents (in place of h_{i+1}) to $idx \circ \mathbf{tar}_{\leq 20s+2} \circ 0^{180s-2}$. In place of the original $\mathbf{tar}_{\leq 20s+2}$, we write $10 \circ i$. This is feasible because $i \leq l$, so it can be specified using $20s$ bits. Here, the 10 in the beginning indicates this compression type. We further copy the last $180s - 2$ bits of **tar**, given by $\mathbf{tar}_{\geq 1320s+3}$, into the remaining free $180s - 2$ bits. Thus, finally changing the contents in place of h_{i+1} to $idx \circ \mathbf{tar}_{\leq 20s+2} \circ \mathbf{tar}_{\geq 1320s+3}$; which effectively frees up the last $180s - 2$ bits of **tar** (and thus the catalytic tape). Notice that in this compression, we do not change the first c bits of the catalytic tape.

Decompression. We decompress as follows: We read the first $20s+2$ bits of $\mathbf{tar}$, which gives us i and thus indicates the location on the catalytic tape where h_{i+1} was written. Consequently, we now have access to the contents $idx \circ \mathbf{tar}_{\leq 20s+2} \circ \mathbf{tar}_{\geq 1320s+3}$. We then loop through all hash functions again in lexicographical order. When we find a bad one with index idx, we know it is h_{i+1}. We copy $\mathbf{tar}_{\leq 20s+2}$ and $\mathbf{tar}_{\geq 1320s+3}$ to their original locations and replace $idx \circ \mathbf{tar}_{\leq 20s+2} \circ \mathbf{tar}_{\geq 1320s+3}$ with h_{i+1}.

We now describe in a step-by-step manner how we can check if a hash function is bad:

(a) Since we are in the sub-case where all the y-DFS from $\langle \mathrm{acc}_{M,x,\tau}, 0\rangle$ and $\langle \mathrm{rej}_{M,x,\tau}, 0\rangle$ have a size of at most $H = 2^{1500s}$, and where additionally $|S_i| \leq T = 2^{100s}$, we can utilize the subroutines $\textsc{Count}_{S_i}$ and $\textsc{Index}_{S_i}$ from Claim 64. This enables us to visit each configuration $\langle v, i\rangle \in S_i$. Furthermore, for any $j \in [c+s]$, we can output the j-th bit in the description of v while ensuring that the subroutines reset the first c bits of the catalytic tape back to τ. According to Claim 64, this entire process can be accomplished using $O(s)$ workspace and requires $2^{O(s)}$ time.

(b) For any $\langle v, i\rangle$ in S_i (which can be visited using (a)), we know that there are at most d_M (a constant) elements in $N_{i+1}(\langle v, i\rangle)$. While at $\langle v, i\rangle$, we can take a peek at any of its neighbors $\langle z, i+1\rangle$ in layer $i+1$. We can keep track of the local changes made to move from $\langle v, i\rangle$ to $\langle z, i+1\rangle$ using $O(s)$ space, which can be undone. This allows us to effectively visit $\langle z, i+1\rangle$ and determine the label(s) for the edge from $\langle z, i+1\rangle$ to $\langle v, i\rangle$, which can be 0, 1, or both. Consequently, we can figure out if B_{vz} is the set of seeds with the first bit as 0 or 1 or the set of all possible seeds, and thus we can compute $\rho(B_{vz})$, which will be either $\frac{1}{2}$ or 1.

(c) Now we describe how to check the goodness of a hash function. It suffices to demonstrate how to determine if a given hash function h is (A_v^{acc}, B_{vz})-independent for some $\langle v, i\rangle \in S_i$ and $\langle z, i+1\rangle \in N_{i+1}(\langle v, i\rangle)$. The same approach can be used to check (A_v^{rej}, B_{vz})-independence. We can then iterate through all possible $\langle v, i\rangle$ and $\langle z, i+1\rangle$ using steps (a) and (b) to check the goodness of h.

We will keep two counters, both initialized to zero. Given that all y-DFS from $\langle \mathrm{acc}_{M,x,\tau}, 0\rangle$ are of size at most H, we do the following for every seed (of PRG_i): We run the y-DFS from $\langle \mathrm{acc}_{M,x,\tau}, 0\rangle$ for H steps. For each configuration visited, we move back and forth via $\langle \mathrm{acc}_{M,x,\tau}, 0\rangle$ to compare it bit-by-bit with $\langle v, i\rangle$. This allows us to check if the y-DFS from $\langle \mathrm{acc}_{M,x,\tau}, 0\rangle$ visits $\langle v, i\rangle$. If it does, we increase the first counter. If, for such a seed, $h(seed) \in B_{vz}$, we increase the second counter as well. Notice that after executing the y-DFS for each seed, the value of the first counter, as per Definition 65, is $|A_v^{acc}|$. Meanwhile, the value of the second counter is $2^m \mathbf{Pr}_{seed}[seed \in A_v^{acc} \text{ and } h(seed) \in B_{vz}]$. Consequently, we can check if h is $(A_v^{acc}, B_{vz}, \alpha)$-independent (as per Definition 67) for our value of $\alpha = \frac{1}{2 \cdot 2^{40s}}$; using the value of the second counter, $|A_v^{acc}|$, and $|B_{vz}|$ computed using (b).

The hash functions from the family can be computed using space $O(s)$, and the subroutines we employed also use space $O(s)$ while requiring time $2^{O(s)}$. This allows us to efficiently determine whether a hash function is bad. Additionally, after performing the check, we reset the first c bits of the catalytic tape (used by the subroutines) back to the value τ.

Case 3: Can build upto PRG_l

The final case is where $\mathcal{F}^L$ does not achieve compression anywhere, and all the hash functions h_1 to h_l are good in their respective iterations. In this case, $\mathcal{F}^L$ computes two fractions:

$$\begin{aligned} f_{acc} &\coloneqq \mathbf{Pr}_{seed}[y\text{-DFS from } \langle \mathrm{acc}_{M,x,\tau}, 0\rangle \text{ visits } \langle \mathrm{start}_{M,x,\tau}, l\rangle \text{ for } y \leftarrow \mathrm{PRG}_l], \\ f_{rej} &\coloneqq \mathbf{Pr}_{seed}[y\text{-DFS from } \langle \mathrm{rej}_{M,x,\tau}, 0\rangle \text{ visits } \langle \mathrm{start}_{M,x,\tau}, l\rangle \text{ for } y \leftarrow \mathrm{PRG}_l]. \end{aligned} \tag{17}$$

Here, f_{acc} denotes the fraction of seeds for which the y-DFS from $\langle \mathrm{acc}_{M,x,\tau}, 0\rangle$ according to PRG_l, visits $\langle \mathrm{start}_{M,x,\tau}, l\rangle$, while f_{rej} is similarly defined. Let the constant $\zeta \coloneqq \frac{2\delta}{1+\frac{\delta}{2\epsilon}} < 2\epsilon$ (follows from the assumption that $\delta < 2\epsilon$). If $f_{acc} \geq \frac{1}{2} + \epsilon - \zeta$, then $\mathcal{F}^L$ accepts the input x. Conversely, if $f_{rej} \geq \frac{1}{2} + \epsilon - \zeta$, then $\mathcal{F}^L$ rejects. If neither condition is satisfied, it outputs $\perp$.

Note that all the y-DFS from $\langle \mathrm{acc}_{M,x,\tau}, 0\rangle$ and $\langle \mathrm{rej}_{M,x,\tau}, 0\rangle$, as per PRG_{l-1}, have a size of at most H in this situation. Otherwise, $\mathcal{F}^L$ would have compressed the catalytic tape. It follows that the size of each y-DFS from $\langle \mathrm{acc}_{M,x,\tau}, 0\rangle$ and $\langle \mathrm{rej}_{M,x,\tau}, 0\rangle$, as per PRG_l, is at most $d_M H$, where d_M is a constant that denotes the maximum number of edges incident to any vertex in the layered configuration graph of $G_{M,x}$. Thus, to compute the fraction f_{acc}, we can do the following: For each seed (of PRG_l), we perform the y-DFS from $\langle \mathrm{acc}_{M,x,\tau}, 0\rangle$ for $d_M H$ steps. For each step, we check if the configuration visited is a starting configuration in layer l. If yes, we can move back and forth between the visited configuration and $\langle \mathrm{acc}_{M,x,\tau}, 0\rangle$ to check if the catalytic contents of the configuration are τ. This way, we can count the number of seeds for which y-DFS from $\langle \mathrm{acc}_{M,x,\tau}, 0\rangle$ (as per PRG_l) visits $\langle \mathrm{start}_{M,x,\tau}, l\rangle$, and thus compute f_{acc}. This can be done using $O(s)$ workspace and in time $2^{O(s)}$. The fraction f_{rej} can be computed similarly.

B.3 Correctness

It can be verified that $\mathcal{F}^L$ (and the decompression algorithms, i.e., $\mathcal{R}^L$) use only $O(s)$ workspace and run in time $2^{O(s)}$. Moreover, in all cases, $\mathcal{F}^L$ frees up the last bit on the catalytic tape, except when it ends up in Case 3, where it outputs accept, reject, or $\perp$. We will first argue the correctness of these outputs.

Consider the case when $f_{acc} \geq \frac{1}{2} + \epsilon - \zeta$. Using Equation 14, we know the error by PRG_l is at most $\frac{1}{2^{5s}}$. That is,

$$\left| f_{acc} - \mathbf{Pr}[\text{Going from } \langle \mathrm{start}_{M,x,\tau}, l\rangle \text{ to } \langle \mathrm{acc}_{M,x,\tau}, 0\rangle \text{ in the layered graph of } G_{M,x}] \right| \leq \frac{1}{2^{5s}} \tag{18}$$

Thus, it follows from Definition 54

$$\left| f_{acc} - \mathbf{Pr}[\text{Going from } \mathrm{start}_{M,x,\tau} \text{ to } \mathrm{acc}_{M,x,\tau} \text{ within } l \text{ steps in } G_{M,x}] \right| \leq \frac{1}{2^{5s}}$$

Therefore, we get

$$\begin{aligned} &\mathbf{Pr}[M \text{ accepts } x, \text{ given initial catalytic contents } \tau] \\ &\geq \mathbf{Pr}[\text{Going from } \mathrm{start}_{M,x,\tau} \text{ to } \mathrm{acc}_{M,x,\tau} \text{ within } l \text{ steps in } G_{M,x}] \\ &\geq f_{acc} - \frac{1}{2^{5s}} \\ &\geq \frac{1}{2} + \epsilon - \zeta - \frac{1}{2^{5s}} \\ &= \frac{1}{2} - \epsilon + (2\epsilon - \zeta) - \frac{1}{2^{5s}} \end{aligned}$$

Since the constant $(2\epsilon - \zeta) > 0$, it holds for large enough s that $\frac{1}{2^{5s}} < \frac{(2\epsilon-\zeta)}{2}$; thus, the probability is lower-bounded by $\frac{1}{2} - \epsilon + \frac{(2\epsilon-\zeta)}{2}$. But this means the input x is in the language. This is because if it

weren't the case, by the definition of M, the probability M that accepts x is $\leq \frac{1}{2} - \epsilon$, leading to a contradiction. Similarly, if $f_{rej} \geq \frac{1}{2} + \epsilon - \zeta$, it implies x is not in the language. Thus, whenever $\mathcal{F}^L$ outputs accept/reject, it is correct.

We are left to show that the fraction of initial catalytic tapes for which $\mathcal{F}^L$ outputs $\perp$ is upper-bounded by $\frac{1}{2^{4s}}$. Let τ be the contents of the initial c bits of the catalytic tape used by $\mathcal{F}^L$. Then, recalling Definition 19 and the discussion from Appendix A, we know that for $0 < \beta < 1 - \delta$ (this is valid as $\delta < 2\epsilon \leq 1$), the τ^β-graph consists of $\text{start}_{M,x,\tau}$, at least one of $\text{acc}_{M,x,\tau}$ and $\text{rej}_{M,x,\tau}$, and does not contain $\text{acc}_{M,x,\tau'}/\text{rej}_{M,x,\tau'}$, for $\tau \neq \tau'$. Taking β to be

$$0 < \beta = \frac{1}{4}\left(1 - \frac{\delta}{2\epsilon}\right) < 1 - \frac{\delta}{2\epsilon} \leq 1 - \delta$$

and using Lemma 22, we get that the probability of leaving the τ^β-graph to (null vertex) $\perp$, starting the walk from $\text{start}_{M,x,\tau}$ is

$$\leq \frac{\delta}{(1-\beta)} = \frac{2\delta}{1.5 + \frac{\delta}{4\epsilon}} \coloneqq \eta$$

That is with probability $\geq 1 - \eta$ we reach $\text{acc}_{M,x,\tau}$ or $\text{rej}_{M,x,\tau}$ within (i.e., without leaving it) the τ^β-graph, starting the walk from $\text{start}_{M,x,\tau}$. Consider the case when x is in the language. Then, by definition, M rejects it with probability $\leq \frac{1}{2} - \epsilon$. Thus, the probability of reaching $\text{acc}_{M,x,\tau}$ from $\text{start}_{M,x,\tau}$ within the τ^β-graph is at least

$$\begin{aligned} 1 - \left(\frac{1}{2} - \epsilon\right) - \eta &= \frac{1}{2} + \epsilon - \eta \\ &= \frac{1}{2} + \epsilon - \zeta + (\zeta - \eta) \end{aligned} \tag{19}$$

Similarly, when $x \notin L$, the probability of reaching $\text{rej}_{M,x,\tau}$ from $\text{start}_{M,x,\tau}$ within the τ^β-graph is at least $\frac{1}{2} + \epsilon - \zeta + (\zeta - \eta)$. Plugging in the values for η, ζ we get that

$$\begin{aligned} (\zeta - \eta) &= \frac{2\delta}{1 + \frac{\delta}{2\epsilon}} - \frac{2\delta}{1.5 + \frac{\delta}{4\epsilon}} \\ &= 2\delta \cdot \frac{0.5 - \frac{\delta}{4\epsilon}}{\left(1 + \frac{\delta}{2\epsilon}\right)\left(1.5 + \frac{\delta}{4\epsilon}\right)} \\ &> 0 \end{aligned}$$

where the last inequality follows from the assumption that $2\epsilon > \delta > 0$. Finally, using Lemma 21 we know that the average size of τ^β-graph is at most 2^{4s}. Thus, by Markov's inequality, the probability (over the uniform choice of τ) that the τ^β-graph has size $\geq 2^{9s}$ is at most $\frac{1}{2^{5s}}$. In other words, the probability (over τ) that there exists a path of length $\geq 2^{9s}$ from $\text{start}_{M,x,\tau}$ to $\text{acc}_{M,x,\tau}/\text{rej}_{M,x,\tau}$ within the τ^β-graph is at most $\frac{1}{2^{5s}}$. Since $l = 2^{20s}$, using Equation 19 and Definition 19 we have that with probability (over τ) $\geq 1 - \frac{1}{2^{5s}}$:

$x \in L \Rightarrow$

$$\textbf{Pr}[\text{Going from } \text{start}_{M,x,\tau} \text{ to } \text{acc}_{M,x,\tau} \text{ within } l \text{ steps in } G_{M,x}] \geq \frac{1}{2} + \epsilon - \zeta + (\zeta - \eta)$$

$x \notin L \Rightarrow$

$$\mathbf{Pr}[\text{Going from start}_{M,x,\tau} \text{ to rej}_{M,x,\tau} \text{ within } l \text{ steps in } G_{M,x}] \geq \frac{1}{2} + \epsilon - \zeta + (\zeta - \eta)$$

Since $(\zeta - \eta) > 0$, for large enough s, $\frac{1}{2^{5s}} \leq \frac{(\zeta-\eta)}{2}$. Since PRG_l has an error of $\leq \frac{1}{2^{5s}}$, using Equation 18 and the above, we have that, with a probability (over τ) of at least $1 - \frac{1}{2^{5s}}$,

$$\begin{aligned}
&x \in L \Rightarrow \\
&\qquad f_{acc} \geq \frac{1}{2} + \epsilon - \zeta + (\zeta - \eta) - \frac{1}{2^{5s}} \geq \frac{1}{2} + \epsilon - \zeta + \frac{(\zeta - \eta)}{2} > \frac{1}{2} + \epsilon - \zeta \\
&x \notin L \Rightarrow \\
&\qquad f_{rej} \geq \frac{1}{2} + \epsilon - \zeta + (\zeta - \eta) - \frac{1}{2^{5s}} \geq \frac{1}{2} + \epsilon - \zeta + \frac{(\zeta - \eta)}{2} > \frac{1}{2} + \epsilon - \zeta
\end{aligned}$$

In other words, we have that $\mathcal{F}^L$ outputs $\perp$ with probability (over τ) $\leq \frac{1}{2^{5s}}$. This completes the proof of Lemma 23a) for the case $\delta > 0$.

B.4 When $\delta = 0$.

For $\delta = 0$, $\mathcal{F}^L$ behaves exactly as it does for $\delta > 0$, except when it falls into Case 3, where it behaves slightly differently. In this case, it outputs accept if $f_{acc} \geq \frac{1}{2} + \frac{\epsilon}{2}$ and reject if $f_{rej} \geq \frac{1}{2} + \frac{\epsilon}{2}$ (recall that, by definition, $\epsilon > 0$). If neither of these conditions is true, as we show, it will compress the catalytic tape. Thus, $\mathcal{F}^L$ never outputs $\perp$.

Firstly, recall that when $\mathcal{F}^L$ ends up in Case 3, all the hash functions from h_1 to h_l are good, and thus the error by PRG_l is at most $\frac{1}{2^{5s}}$ (using Equation 14). Therefore, it can be argued, as done in Section B.3, that when $\mathcal{F}^L$ does output accept or reject, it is correct. Next, observe that when $\delta = 0$, by Definition 19, the τ^β-graph ($\beta > 0$) is precisely the configuration graph $G_{M,x,\tau}$. In other words, the τ^β-graph has no null vertex ($\perp$), since it follows from Lemma 22 that the probability of leaving the τ^β-graph (from $\text{start}_{M,x,\tau}$) to the $\perp$-vertex is 0, for $\delta = 0$. Moreover, repeating the argument from Section B.3, one can verify that if the longest path from $\text{start}_{M,x,\tau}$ to $\text{acc}_{M,x,\tau}$ or $\text{rej}_{M,x,\tau}$ within the τ^β-graph (which is the same as $G_{M,x,\tau}$ in this scenario) has length $< 2^{9s}$, then $\mathcal{F}^L$ outputs accept or reject, as either f_{acc} or f_{rej} is at least $\frac{1}{2} + \frac{\epsilon}{2}$. We now describe how $\mathcal{F}^L$ compresses the catalytic tape in the scenario in which neither of these fractions is at least $\frac{1}{2} + \frac{\epsilon}{2}$.

Let P be the longest path starting from $\text{start}_{M,x,\tau}$ in $G_{M,x,\tau}$. Then, we know that the length of P is at least 2^{9s}. Focus on the last 2^{9s} configurations along P, which we label as $v_1, v_2, \ldots, v_{2^{9s}}$, where each v_i is defined as $\langle \pi_i, u_i \rangle$. Then, by the definition of P the longest path from v_i (in $G_{M,x,\tau}$) to either of the halt configurations $\text{acc}_{M,x,\tau}/\text{rej}_{M,x,\tau}$ has a length of $\leq 2^{9s}$. Furthermore, for every $i \in [2^{9s}]$, M always resets the catalytic tape to τ, starting from v_i (follows from the fact that $\delta = 0$). Consider an arbitrary $i \in [2^{9s}]$. Then, without loss of generality, the probability that M, when run from v_i, reaches $\text{acc}_{M,x,\tau}$ within 2^{9s} steps is at least $\frac{1}{2}$. However, because PRG_l has an error of $\leq \frac{1}{2^{5s}}$ and $l = 2^{20s}$, using Equation 2, it follows that the fraction of seeds for which y-DFS (as per PRG_l) from $\langle \text{acc}_{M,x,\tau}, 0 \rangle$ visits $\langle \pi_i, u_i, l \rangle$ is at least $\frac{1}{2} - \frac{1}{2^{5s}}$. In general, for every i, the fraction of seeds for which the y-DFS from $\langle \text{acc}_{M,x,\tau}, 0 \rangle$ or $\langle \text{rej}_{M,x,\tau}, 0 \rangle$ visits $\langle \pi_i, u_i, l \rangle$ is at least $\frac{1}{2} - \frac{1}{2^{5s}} \geq \frac{1}{4}$, for large enough s.

Consider the following set S'_l, which is defined similarly to the set S_l (Definition 63), with the difference that we now consider a configuration $\langle v, l \rangle$ to be in the set S'_l iff for at least $\frac{1}{4}$ (instead of $\frac{1}{2^{30s}}$) fraction of the seeds, the y-DFS (as per PRG_l) from $\langle \text{acc}_{M,x,\tau}, 0 \rangle$ or $\langle \text{rej}_{M,x,\tau}, 0 \rangle$ visits $\langle v, l \rangle$. That is, a configuration $\langle v, l \rangle$ (in layer l) is in S'_l iff:

$$\mathbf{Pr}_{\text{seed}}[y\text{-DFS from } \langle \text{acc}_{M,x,\tau}, 0\rangle \text{ or } \langle \text{rej}_{M,x,\tau}, 0\rangle \text{ visits } \langle v, l\rangle \text{ for } y \leftarrow \text{PRG}_l(\text{seed})] \geq \frac{1}{4}.$$

Since, for every $i \in [2^{9s}]$, the configuration $\langle \pi_i, u_i, l\rangle$ lies in S'_l, we have that $|S'_l| \geq 2^{9s}$. Similar to subroutines $\textsc{Count}_{S_l}$ and $\textsc{Index}_{S_l}$ (introduced in Case 2), we can implement $\textsc{Count}_{S'_l}$ and $\textsc{Index}_{S'_l}$ to compute the size of the set S'_l and to index every configuration in it. Recall that all the y-DFS from $\langle \text{acc}_{M,x,\tau}, 0\rangle$ and $\langle \text{rej}_{M,x,\tau}, 0\rangle$, as per PRG_{l-1}, have a size of at most H in this situation. Otherwise, $\mathcal{F}^L$ would have compressed the catalytic tape (and we would not have ended up in Case 3). Thus, the size of each y-DFS from $\langle \text{acc}_{M,x,\tau}, 0\rangle$ and $\langle \text{rej}_{M,x,\tau}, 0\rangle$, according to PRG_l, is at most $d_M H$. Therefore, referring to Claim 64, we can implement $\textsc{Count}_{S'_l}$ and $\textsc{Index}_{S'_l}$ using $O(s)$ workspace and in time $2^{O(s)}$.

The compression we describe is identical to the one used in Case 2.1, with minor modifications.

Compression. Let $val(\mathbf{tar}_{\leq 9s})$ denote the integer value of the first $9s$ bits of **tar**, incremented by 1, which is an integer in $[2^{9s}]$. Let $\langle \pi, u, l\rangle$ be the configuration in S'_l with index $val(\mathbf{tar}_{\leq 9s})$. Using $\textsc{Index}_{S'_l}$, we can visit $\langle \pi, u, l\rangle$. Similar to Case 2.1, we define the set TAU with minor changes. Now, TAU is the set of all c-bit catalytic settings such that $\pi' \in$ TAU iff, for at least $\frac{2^m}{4}$ seeds (instead of $\frac{2^m}{2^{30s}}$), the y-DFS from $\langle \pi, u, l\rangle$ visits either $\langle \text{acc}_{M,x,\pi'}, 0\rangle$ or $\langle \text{rej}_{M,x,\pi'}, 0\rangle$ within $d_M H$ steps. From the definition of TAU, it follows that $|\text{TAU}| \leq 4$. Additionally, as in Case 2.1, we can argue that τ (the string representing the first c bits of our catalytic tape) is in TAU. We can also determine the index idx of τ in the set TAU. To achieve compression, we replace τ with π (by visiting $\langle \pi, u, l\rangle$) and $\mathbf{tar}_{\leq 9s}$ with $11 \circ idx \circ u \circ 0^{8s-4}$, freeing $8s-4$ bits (where 11 denotes this compression type). This is feasible since idx takes up 2 bits and u requires s bits.

Decompression. The decompression is also the same as Case 2.1. Given the compressed tape, we are effectively standing at $\langle \pi, u, l\rangle$ and know idx. As in Case 2.1, using idx, we can recover τ, such that after recovering τ, we can move back and forth between $\langle \text{acc}_{M,x,\tau}, 0\rangle / \langle \text{rej}_{M,x,\tau}, 0\rangle$ and $\langle \pi, u, l\rangle$. Finally, as in Case 2.1, we can figure out $val(\mathbf{tar}_{\leq 9s})$ (and hence recover **tar**), by using $\textsc{Count}_{S'_l}$ and $\textsc{Index}_{S'_l}$ to compute the index of $\langle \pi, u, l\rangle$ in the set S'_l.

Note that, as before, we can always copy the last $8s-4$ bits of **tar** into the freed-up bits so that we have free bits at the end of the catalytic tape.

The correctness of the compression scheme and the fact that the compression/decompression takes $O(s)$ space and runs in time $2^{O(s)}$ can be argued similarly to Case 2.1.

B.5 Proof of Lemma 23b)

Let $L \in \mathsf{BP}^{\epsilon}\mathsf{C}^{\delta}\mathsf{LP}$ where ϵ, δ are constants such that $\delta < 2\epsilon$. Without loss of generality, let the machine for L use $s = d \log n$ bits of work space, $c = n^d$ bits of catalytic space; and let it run in time n^d for constants d and input length n. Then, the proof of Lemma 23b) follows from the proof of Lemma 23a). This is because any path from $\text{start}_{M,x,\tau}$ to $\text{acc}_{M,x,\tau}/\text{rej}_{M,x,\tau}$ in $G_{M,x,\tau}$ has length $\leq n^d = 2^s < 2^{9s}$. Thus, following our discussion from Section B.3, $\mathcal{F}^L$ never outputs $\perp$.

C Proof of Observation 24

We present a proof sketch here, assuming that the reader is already familiar with the de-randomization proof of the class BPCL, as described by Cook et al. [CLMP25]. Their proof uses a variant of the Nisan-Wigderson pseudorandom generator that enables random walks on the configuration graph. A key observation they make is that, after running a BPCL machine from some starting configuration

$\text{start}_{M,x,\tau}$, using the output of the PRG as random bits, we can return to $\text{start}_{M,x,\tau}$ by performing a y-DFS (see Definition 58) from the resulting configuration (which can be assumed to lie in layer 0), for y being all possible strings (and their prefixes) produced by the PRG. One of these y-DFS is guaranteed to encounter $\text{start}_{M,x,\tau}$. Moreover, because the configuration graphs corresponding to different initial catalytic settings are vertex-disjoint, we can be confident that no other starting configuration apart from $\text{start}_{M,x,\tau}$ is encountered in any of these y-DFS. However, this assurance fails when dealing with machines that do not always reset the catalytic tape, as the configuration graphs may no longer be disjoint.

Let M be a $\mathsf{BP}^{\epsilon}\mathsf{C}_{\frac{1}{2^{4n^d+3}}}\mathsf{SPACE}(d\log n, n^d)$ machine (for some constant d). The claim is that we can still use the same approach as in [CLMP25], where they use walks of length at most $4n^d$.

Claim 72. *Let v be a configuration obtained from $\text{start}_{M,x,\tau}$ by following a walk of length $4n^d$ (in $G_{M,x,\tau}$). Then, while performing a y-DFS starting from $\langle v, 0\rangle$, for some y with $|y| \leq 4n^d$, we cannot encounter a different starting configuration $\langle \text{start}_{M,x,\tau'}, *\rangle$ with $\tau' \neq \tau$.*

Proof. Since v is reachable from $\text{start}_{M,x,\tau}$ within $4n^d$ steps, we assert that v must be a $\tau^{\frac{3}{4}}$-node (see Definition 19). Suppose this is not the case. Then, beginning from $\text{start}_{M,x,\tau}$, we would be able to destroy the catalytic tape with a probability of at least $\frac{1}{2^{4n^d}} \cdot \frac{1}{4} = \frac{1}{2^{4n^d+2}}$. This, however, would violate the definition of M, which permits destruction of the catalytic tape only with a probability of at most $\frac{1}{2^{4n^d+3}}$.

Furthermore, if performing a y-DFS from $\langle v, 0\rangle$ can lead us to $\langle \text{start}_{M,x,\tau'}, *\rangle$, it suggests that v is reachable from $\text{start}_{M,x,\tau'}$ within $4n^d$ steps. This means that v is a $\tau'^{\frac{3}{4}}$-node as well. However, this implies that starting from v, we could reach a halt configuration with catalytic tape contents τ, with a probability of $\frac{3}{4}$—and similarly for τ'—which is not possible. □