NON-UNIFORM LANGUAGE GENERATION WITH REPLAY

Anonymous authors
Paper under double-blind review

ABSTRACT

In the framework of *language generation in the limit* introduced by Kleinberg & Mullainathan (2024), an algorithm receives example strings from an unknown target language, which is chosen from a known collection of potential languages. The goal for the algorithm is to eventually generate only strings that belong to the target and have not appeared among the examples. The stronger notion of *non-uniform generation* requires correct generation after a finite number of distinct examples that may depend on the target language but not on its enumeration (Li, Raman, and Tewari, 2025; Charikar and Pabbaraju, 2025a). Racca, Valko, and Sanyal (2026) introduced a *replay* model in which the adversary can also provide the algorithm's previous outputs as examples, even when those outputs lie outside the target language. They showed that some countable collections cannot be non-uniformly generated with replay, and left finding a characterization as an open question. In this paper, we resolve this question by giving a complete characterization of non-uniform generation with replay for arbitrary collections over a countable universe. For countable collections, we obtain a simpler characterization: generation is possible if and only if, for each language L, every intersection of L with finitely many other languages is either infinite or has size bounded by a constant depending only on L. Beyond determining whether non-uniform generation is possible, we characterize how many distinct inputs an algorithm needs before it must generate correctly. For any assignment of a positive integer $d(L)$ to each language L, we determine an exact characterization for when there exists an algorithm that generates correctly from every target L once it has received at least $d(L)$ distinct inputs.

1 INTRODUCTION

A central question in machine learning is understanding when an algorithm can learn to produce correct new strings after being shown a sequence of example strings. We study when correct generation remains possible if the algorithm's previous outputs can themselves be returned as examples. Our starting point is the framework of *language generation in the limit*, introduced by Kleinberg & Mullainathan (2024). In this model, there is a known collection $\mathcal{C}$ of infinite languages over a countable universe U. An adversary privately chooses a target language $L \in \mathcal{C}$ and presents its strings in an infinite enumeration. The algorithm observes one example string in each round and responds with a string of its own. The goal is that, after some finite time, every output belongs to L and has not previously been presented by the adversary. Kleinberg & Mullainathan (2024) showed that every countable collection can be generated in the limit.

A natural question is how many examples the algorithm needs before it must generate correctly. In the original model, this number can depend on both the target language and its enumeration. Li, Raman, and Tewari (2025) introduced the stronger notions of uniform and non-uniform generation. In *uniform generation*, there is a single finite number of distinct examples that suffices for every target language and every enumeration. In *non-uniform generation*, this number may depend on the target language, but it must still work for every enumeration of that language. We call this number the target's *sample threshold*. Li, Raman, and Tewari (2025) and Charikar & Pabbaraju (2025a) showed that every countable collection is non-uniformly generatable. Thus one algorithm can eventually generate correctly for every target, and the number of distinct examples it needs can be bounded separately for each language.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

We study non-uniform generation when the adversary can return the algorithm's own outputs as examples. This *replay* model was introduced by Racca, Valko, and Sanyal (2026). In each round, the adversary may provide either a string from the target language or any string previously output by the algorithm. The adversary need not enumerate the entire target and may replay outputs forever. The algorithm is not told whether an input was drawn from the target or replayed from an earlier output. In particular, the adversary may return a string that the algorithm previously output outside the target. The sample threshold now counts all distinct inputs, including replayed outputs. Racca, Valko, and Sanyal (2026) showed that replay does not affect uniform generation: a collection has the same optimal uniform threshold with and without replay. However, they constructed a countable collection that cannot be non-uniformly generated under replay. Since every countable collection is non-uniformly generatable without replay, this gives a separation between the two models. They left finding a characterization of non-uniform generation under replay as an open question.

1.1 Our Results

In this paper, we give a complete characterization for non-uniform generation under replay, answering an important open question raised by Racca, Valko, and Sanyal (2026). When generation is possible, we also ask how many distinct inputs the algorithm needs before it must generate correctly. Our second characterization answers this question while allowing a different number of inputs for each target.

Characterizing non-uniform generation. To explain our characterization, first suppose the algorithm has received a finite set S of strings from the target language L. Every language containing S could be the target. If the algorithm outputs a string common to all of these languages, its output belongs to the target regardless of which language containing S the adversary chose. The set of all such strings is called the *closure* of S (Li et al., 2025). An infinite closure always contains a target string that has not yet appeared as an input. However, non-uniform generation can be possible even when every finite sample has finite closure; we give such an example in Example 3.1. Our characterization therefore uses a different intersection. For each language H, choose a finite set E_H of strings outside H. Given a sample S, first select the original languages H containing S, and then intersect their enlarged sets $H \cup E_H$. Since L contains S, the set $L \cup E_L$ is one of the sets being intersected. Every string in the intersection therefore belongs to L or to the finite set E_L. We characterize generation by asking whether these intersections can be made infinite for all samples from each target whose size exceeds a bound depending only on that target.

Informal theorem (Theorem 3.2)**.** *A collection is non-uniformly generatable under replay if and only if one can choose a finite set E_H for each language H with the following property. For each target L, there is a positive integer n_L such that every finite sample $S \subseteq L$ with $|S| \geq n_L$ satisfies*

$$\left| \bigcap_{H \in \mathcal{C}:\, S \subseteq H} (H \cup E_H) \right| = \infty.$$

For countable collections, we also obtain a simpler characterization of non-uniform generation under replay (Corollary 5.1).

Characterizing sample thresholds. Our first theorem determines whether a non-uniform generator exists. We next ask when the algorithm can be required to start generating correctly. Let $d(L)$ be the number of distinct inputs after which every output must be correct and unseen when L is the target. These positive integers may differ from one language to another; together, they form a function d called a *threshold profile*. The following theorem characterizes when one algorithm can meet these requirements for all targets.

Informal theorem (Theorem 4.1)**.** *A threshold profile d is achievable under replay if and only if the following condition holds. Consider any nonempty subfamily $\mathcal{A} \subseteq \mathcal{C}$ whose languages have finitely many strings in common and whose assigned thresholds have a common finite upper bound. The number of strings common to these languages must be smaller than $d(L)$ for every $L \in \mathcal{A}$.*

For comparison, in Appendix A we derive a characterization of achievable threshold profiles for generation without replay. For a nonempty subfamily with bounded thresholds and finite intersection,

generation without replay requires the intersection size to be smaller than the *largest* threshold; replay requires it to be smaller than *every* threshold. The two characterizations therefore identify exactly which profiles can be achieved without replay but become impossible when replay is allowed.

Extensions. In Appendix C, we combine replay with the noise model introduced by Raman and Raman (2025). In addition to replaying outputs, the adversary may introduce at most a fixed finite number b of distinct *noisy strings* outside the target, without first waiting for the algorithm to output them. We use the variant introduced by Bai, Panigrahi, and Zhang (2026), in which the algorithm knows the noise level b. The algorithm is not told which inputs are noisy and these noisy inputs count toward the sample threshold. We extend our profile characterization to this model for arbitrary collections.

There are countable collections that can be non-uniformly generated under replay but cannot be generated if the adversary may introduce even one noisy string. However, a countable collection that can be non-uniformly generated when one noisy string is allowed can also be generated for every fixed positive finite noise level b. The algorithm and its thresholds may depend on b. Finally, we consider randomized algorithms for which, with probability one, every output is correct and unseen once the number of distinct inputs reaches the target's threshold. For arbitrary collections and every fixed $b \geq 0$, these algorithms achieve exactly the same threshold profiles as deterministic algorithms.

1.2 Other Related Work

We review related results on the number of examples needed for generation, the range of strings generated, and noise in the inputs. Li, Raman, and Tewari (2025) characterized ordinary uniform generation using closure dimension, which is defined from the sizes of finite samples whose closures are finite. For ordinary non-uniform generation, Charikar & Pabbaraju (2025b) studied assignments of thresholds for which no language's threshold can be lowered without increasing another language's threshold. Other work studies the breadth of generation, asking how much of the target language an algorithm can produce (Charikar & Pabbaraju, 2025a; Kleinberg & Wei, 2025; 2026).

Another line of work studies noise or omissions in the examples (Raman & Raman, 2025; Bai et al., 2026; Li & Zhang, 2026; Kleinberg & Wei, 2026; Mehrotra et al., 2026; Charikar & Pabbaraju, 2025b; Kleinberg et al., 2026). External noise allows the adversary to introduce a string outside the target language even if the algorithm has never output that string. Our extension combines this form of noise with replay.

2 Preliminaries

We define the replay model of Racca, Valko, and Sanyal (2026), including how inputs are counted and when outputs must be correct. We also recall the notion of closure from Li, Raman, and Tewari (2025), which will be used in both characterizations.

Let $U = \{u_1, u_2, \dots\}$ be a countably infinite universe with a fixed enumeration without repetition. A language is an infinite subset of U, and $\mathcal{C}$ is a nonempty collection of languages. We allow $\mathcal{C}$ to be either countable or uncountable. We write $\mathbb{N} = \{1, 2, \dots\}$ and take the intersection of an empty family to be U.

Generation under replay. The collection $\mathcal{C}$ is known to the algorithm. As in Racca, Valko, and Sanyal (2026), we impose no computability or running-time restriction on the algorithm: it may be any function from finite input sequences to output strings. An adversary privately chooses a target language $L \in \mathcal{C}$. In round t, the adversary supplies a string x_t, and the algorithm responds with a string $o_t \in U$. The adversary may choose its input using all previous outputs of the algorithm, subject to the requirement

$$x_t \in L \cup \{o_1, \dots, o_{t-1}\}.$$

Thus each input must either belong to the target or be a string previously output by the algorithm. The algorithm is not told which of these possibilities holds. The adversary may repeat inputs, and it is not required to enumerate all strings in L. In particular, after supplying its first input from L, it may choose to replay outputs forever. We note that not requiring the adversary to enumerate all strings in L matches the model of non-uniform generation with replay introduced by Racca, Valko, and Sanyal

(2026). In the setting without replay, Bai, Panigrahi, and Zhang (2026) showed that requiring the adversary to eventually present every string in L does not change which collections are non-uniformly generatable.

Let $S_t := \{x_1, \ldots, x_t\}$ be the set of inputs received through round t, and let $q_t := |S_t|$ be the number of distinct inputs. An output o_t is *valid and unseen* if $o_t \in L \setminus S_t$. A repeated output can still be valid and unseen: it must belong to L and must not have appeared as an input. The sample threshold is measured by q_t, so replayed outputs count in the same way as other inputs.

In non-uniform generation, the number of distinct inputs needed for correctness may depend on the target language (Li et al., 2025; Charikar & Pabbaraju, 2025a). For each language L, fix a positive integer $d(L)$. One algorithm must generate correctly from L whenever $q_t \geq d(L)$, for every choice of L and every adversary allowed in the replay model.

Definition 2.1 (Threshold profile)**.** A *threshold profile* is a function $d\colon \mathcal{C} \to \mathbb{N}$. The profile is *achievable* if there is a deterministic algorithm such that, for every target $L \in \mathcal{C}$ and every valid replay adversary,

$$q_t \geq d(L) \quad \Longrightarrow \quad o_t \in L \setminus S_t \qquad \text{in every round } t.$$

The collection $\mathcal{C}$ is *non-uniformly generatable under replay* if at least one threshold profile is achievable.

Correctness is required in the round that first reaches $d(L)$ distinct inputs, as well as in every later round. If the adversary never supplies this many distinct inputs, the definition imposes no correctness requirement. The assigned value $d(L)$ is an upper bound on how many distinct inputs the algorithm needs; the algorithm is allowed to generate correctly before that bound is reached. We consider deterministic algorithms in the main text. In Appendix C, we also consider randomized algorithms and require that, with probability one, $o_t \in L \setminus S_t$ in every round with $q_t \geq d(L)$.

Consistent languages and closure. We use the notion of closure from Li, Raman, and Tewari (2025). Suppose a finite set S is known to consist of target strings. The languages that could be the target are exactly those containing every string in S. For a subcollection $\mathcal{B} \subseteq \mathcal{C}$, define

$$\mathcal{B}[S] := \{H \in \mathcal{B} : S \subseteq H\}, \qquad \mathrm{cl}_{\mathcal{B}}(S) := \bigcap_{H \in \mathcal{B}[S]} H.$$

We call the languages in $\mathcal{B}[S]$ *consistent with* S, and call their intersection the *closure* of S in $\mathcal{B}$. If $\mathcal{B}$ contains the actual target L and $S \subseteq L$, then $L \in \mathcal{B}[S]$. Every string in $\mathrm{cl}_{\mathcal{B}}(S)$ therefore belongs to L. Only finitely many strings have appeared as inputs at any given round, so an infinite closure contains a target string outside the current input set S_t. In the replay model, the full input set S_t need not be contained in L, because it may contain strings outside L that the adversary replayed from earlier outputs. The algorithms below will ensure that the sample used to form an intersection contains only target strings.

3 Characterizing Non-Uniform Generation

In this section, we characterize when a collection can be non-uniformly generated under replay. Our condition uses intersections of languages after finitely many strings have been added to each language. To explain why these additions are needed, we first give an example of a generatable collection for which every finite sample has finite closure. The collection is the family of co-singletons considered by Charikar and Pabbaraju (2025b).

Example 3.1. Let $U = \mathbb{N}$ and $H_j := \mathbb{N} \setminus \{j\}$ for each $j \in \mathbb{N}$. For $\mathcal{C} = \{H_j : j \in \mathbb{N}\}$, every finite sample S satisfies

$$\mathrm{cl}_{\mathcal{C}}(S) = \bigcap_{j \notin S} (\mathbb{N} \setminus \{j\}) = S.$$

Thus the closure contains no string outside the sample. Nevertheless, consider the algorithm that, in round t, outputs any integer larger than q_t that does not belong to S_t. The only integer outside H_j is j. Once $q_t \geq j$, the algorithm's output is larger than j and hence belongs to $H_j \setminus S_t$. This algorithm therefore achieves the profile $d(H_j) = j$ under replay.

The example shows why requiring every output to lie in the ordinary closure is too restrictive. The algorithm may output j when H_j is the target, provided it stops doing so before q_t reaches j. Adding j back to each H_j makes all these enlarged languages equal to U, whose intersection is infinite. We extend this idea by allowing finitely many additional strings for each language when forming the intersection. Choose a finite set $E_H \subseteq U \setminus H$ for each $H \in \mathcal{C}$, and write $E = (E_H)_{H \in \mathcal{C}}$. For a finite sample S, define

$$K_E(S) := \bigcap_{H \in \mathcal{C}[S]} (H \cup E_H). \tag{1}$$

The languages included in this intersection are exactly those that contain S before any strings are added. If $S \subseteq L$, the target L is included in the intersection, so $K_E(S) \subseteq L \cup E_L$. Thus a string chosen from $K_E(S)$ can lie outside the target only if it belongs to the finite set E_L. The additions are used only to define this intersection; the target languages and the adversary's permitted inputs remain unchanged.

For a target L, we need a bound on the size of a sample after which these intersections are always infinite. We therefore define

$$D_E(L) := \sup\{|S| : S \subseteq L \text{ is finite and } |K_E(S)| < \infty\}. \tag{2}$$

We take this supremum to be 0 if there are no such samples. This is a per-language version of closure dimension (Li et al., 2025), using $K_E(S)$ in place of ordinary closure. When $D_E(L)$ is finite, every finite sample $S \subseteq L$ with $|S| \geq D_E(L) + 1$ has an infinite intersection $K_E(S)$. Moreover, $D_E(L) + 1$ is the least positive sample-size bound with this property.

Theorem 3.2. *The collection $\mathcal{C}$ is non-uniformly generatable under replay if and only if there exist finite sets $E_H \subseteq U \setminus H$, one for each $H \in \mathcal{C}$, such that $D_E(L) < \infty$ for every $L \in \mathcal{C}$.*

The same sets E_H must work for every sample. The bound $D_E(L) + 1$ ensures that every sample from L of at least that size has an infinite intersection. Correct generation requires the algorithm to avoid the added strings in E_L as well.

Proof of sufficiency in Theorem 3.2. Fix finite sets E_H for which every $D_E(L)$ is finite, and let L be the actual target. We construct an algorithm that first obtains a sample A contained in the target with $K_E(A)$ infinite, and then chooses all subsequent outputs from $K_E(A)$. To obtain such a sample, it outputs only x_1 while $K_E(S_t)$ is finite, as in the uniform replay algorithm of Racca, Valko, and Sanyal (2026, Section 4). Since x_1 belongs to the target, every input during this part of the interaction must also belong to the target: it is either a target string or a replay of x_1. At the first round with $K_E(S_t)$ infinite, the algorithm saves $A := S_t$. This sample contains only target strings. Moreover, this round occurs no later than the first round with $D_E(L) + 1$ distinct inputs, if that many distinct inputs are ever provided. Indeed, if the algorithm were still outputting x_1 in that round, then $S_t \subseteq L$ and the definition of $D_E(L)$ would make $K_E(S_t)$ infinite.

For the actual target L, every string in $K_E(A)$ belongs to $L \cup E_L$. We must therefore stop outputting strings in E_L after a number of distinct inputs depending only on L. The algorithm does not know L or E_L, but it can exclude the first q_t strings of the fixed enumeration $u_1, u_2, \ldots$ of U. Since E_L is finite, this exclusion removes every string in E_L once q_t is at least the largest index of a string in E_L. Starting in the round when A is saved, the algorithm outputs the first string in

$$K_E(A) \setminus \big(S_t \cup \{u_1, \ldots, u_{q_t}\}\big).$$

Such a string always exists, since $K_E(A)$ is infinite and the set removed from it is finite.

The algorithm keeps the same sample A in every later round. Updating the sample to include all later inputs could be unsafe, because an early output may belong to E_L. If the adversary replays such a string, the new input set is no longer contained in L. Intersecting languages containing that new set could exclude L, whereas the saved sample continues to give $K_E(A) \subseteq L \cup E_L$.

To ensure that the excluded initial segment contains every string in E_L, define

$$m(L) := \max\big(\{j : u_j \in E_L\} \cup \{0\}\big), \qquad d(L) := \max\{D_E(L) + 1, m(L)\}.$$

In every round with $q_t \geq d(L)$, the algorithm has retained a sample and excludes all of E_L through the initial segment $\{u_1, \ldots, u_{q_t}\}$. Its output belongs to $K_E(A) \subseteq L \cup E_L$ and avoids both E_L and S_t, so it lies in $L \setminus S_t$. Thus d is an achievable profile. □

Proof of necessity in Theorem 3.2. Suppose an algorithm achieves a profile d. We will construct finite sets E_H such that $K_E(S)$ is infinite whenever $S \subseteq L$ and $|S| \geq d(L)$. This will prove $D_E(L) \leq d(L) - 1$ for every target L.

List without repetition all nonempty finite samples S such that $|S| \geq d(H)$ for at least one language H containing S, and denote this list by $S_1, S_2, \ldots$. This list is countable because U is countable. It is infinite because, for any fixed target L, it includes every finite subset of L of size at least $d(L)$.

For each i, consider a separate interaction in which the adversary first presents the elements of S_i once in their universe order, and thereafter replays the algorithm's latest output. Write $p_i(1)$ for the output after the last element of S_i is presented. The adversary then supplies $p_i(1)$, receives $p_i(2)$, supplies $p_i(2)$, and continues in this way. By the choice of S_i, some language L_i contains S_i and satisfies $|S_i| \geq d(L_i)$. This interaction is valid for target L_i, and the distinct-input count is already at least $d(L_i)$ when $p_i(1)$ is produced. Its output $p_i(1)$ must therefore be unseen. Replaying that string introduces a new distinct input, so the input count is still at least $d(L_i)$ and the next output must again be unseen. Inductively, the strings $p_i(1), p_i(2), \ldots$ are pairwise distinct, and the number of distinct inputs when $p_i(k)$ is produced is $|S_i| + k - 1$.

Now fix any language H containing S_i. The initial inputs belong to H, and every later input is a previous output, so the same interaction is a valid replay interaction with target H. The algorithm is not told which target was chosen, and the adversary follows the same input rule, so the outputs $p_i(k)$ are unchanged when H is the target. Since $|S_i| \geq 1$, the number of distinct inputs when $p_i(k)$ is produced is at least k. The guarantee for H therefore gives

$$S_i \subseteq H, \quad k \geq d(H) \quad \Longrightarrow \quad p_i(k) \in H. \tag{3}$$

For each sample S_i, we want an infinite set B_i contained in $H \cup E_H$ for every language H containing S_i. It will then be contained in $K_E(S_i)$, making this intersection infinite. We can obtain the required containments by adding every string of $B_i \setminus H$ to E_H. However, choosing B_i to contain the entire output sequence could require infinitely many additions: each of infinitely many samples might contribute a different string outside H.

To prevent this, for sample S_i we discard the first $i - 1$ outputs and keep $B_i := \{p_i(k) : k \geq i\}$. Each B_i is infinite because the outputs in the sequence are pairwise distinct. For a fixed H containing S_i, only outputs with $k < d(H)$ can lie outside H by Equation (3). If $i \geq d(H)$, discarding the first $i - 1$ outputs therefore leaves no string outside H. Only the finitely many samples with $i < d(H)$ can contribute additions, and each contributes finitely many. Consequently the set

$$E_H := \left(\bigcup_{i : S_i \subseteq H} B_i \right) \setminus H$$

is finite for every H.

Finally, for every i and every H containing S_i, the definition of E_H ensures that $B_i \subseteq H \cup E_H$. Intersecting over these languages gives $B_i \subseteq K_E(S_i)$, so $K_E(S_i)$ is infinite. Every finite $S \subseteq L$ with $|S| \geq d(L)$ occurs on the list, and its enlarged intersection is therefore infinite. This proves $D_E(L) \leq d(L) - 1$ for every $L \in \mathcal{C}$, as required. □

4 Characterizing Threshold Profiles

In this section, we characterize exactly which threshold profiles can be achieved under replay. Theorem 3.2 determines whether any threshold profile is achievable; here we ask whether an algorithm can achieve particular thresholds specified in advance. For each language L, let $d(L)$ be the required number of distinct inputs: whenever L is the target and $q_t \geq d(L)$, the algorithm must output a string in $L \setminus S_t$. The next theorem answers this question using intersections of the original languages and the integers $d(L)$.

Theorem 4.1. *A profile $d \colon \mathcal{C} \to \mathbb{N}$ is achievable under replay if and only if every nonempty subfamily $\mathcal{A} \subseteq \mathcal{C}$ whose intersection is finite and whose thresholds have a common finite upper bound satisfies*

$$\left| \bigcap_{L \in \mathcal{A}} L \right| < \min_{L \in \mathcal{A}} d(L). \tag{4}$$

The condition applies to finite or infinite subfamilies whose intersection is finite and whose thresholds have a common finite upper bound. If common inputs can reach even one target's threshold, every subsequent output must be unseen. The adversary can then increase the number of distinct inputs in every round by replaying the latest output. This interaction remains valid for every language in the subfamily. Because their thresholds have a common finite upper bound, eventually each output must belong to every one of them. A finite intersection cannot contain the infinitely many distinct outputs that this interaction requires.

Appendix A gives the corresponding characterization without replay: for every nonempty subfamily with bounded thresholds and finite intersection, the intersection size must be smaller than the largest threshold in that subfamily.

Proof. We first prove necessity. Suppose an algorithm achieves d, and let $\mathcal{A}$ be a nonempty subfamily whose intersection is finite and whose thresholds have a finite upper bound. Write $K \coloneqq \bigcap_{L\in\mathcal{A}} L$ and $s \coloneqq \min_{L\in\mathcal{A}} d(L)$, choose $L_0 \in \mathcal{A}$ with $d(L_0) = s$, and let m be an integer bounding all thresholds in $\mathcal{A}$. Assume for contradiction that $|K| \geq s$.

The adversary first presents s distinct elements of K and then replays the algorithm's latest output in every subsequent round. This is a valid adversary for each target $L \in \mathcal{A}$: the first s inputs belong to L, and every later input is a previous output, regardless of whether that output belongs to L. The input rule is the same for all these targets, so the algorithm produces the same transcript for each of them. At round s, the algorithm has received $s = d(L_0)$ distinct inputs and must output an unseen string. Replaying this output increases the number of distinct inputs from s to $s + 1$, so the guarantee for L_0 again requires an unseen output in the next round. Continuing inductively, we have $q_t = t$ for all $t \geq s$, and the outputs from round s onward are pairwise distinct. For every $t \geq m$ and every $L \in \mathcal{A}$, we have $q_t \geq m \geq d(L)$. Since this same transcript is valid for target L, its guarantee requires $o_t \in L$. Hence every output from round m onward belongs to K. This gives infinitely many distinct strings in the finite set K, a contradiction.

We next assume that Equation (4) holds for every subfamily specified in the theorem, and construct an algorithm achieving d. After n distinct inputs, correctness is required for a target H only if $d(H) \leq n$. We write

$$\mathcal{C}_{\leq n} \coloneqq \{H \in \mathcal{C} : d(H) \leq n\}$$

for the collection of these targets. For the prescribed values $d(L)$, we can ignore a possible target H until the count reaches $d(H)$. Our algorithm will therefore intersect only the languages in $\mathcal{C}_{\leq q_t}$ that contain a saved sample A of target strings. Unlike the algorithm of Theorem 3.2, this algorithm changes its intersection: as q_t increases, more languages are included, so the intersection may become smaller. We must keep this sample fixed for the same reason as in the proof of Theorem 3.2: while $q_t < d(L)$ for the actual target L, an output may lie outside L and later return as a replayed input. Using all subsequent inputs to determine which languages are consistent could therefore exclude the target.

We will retain the current input set as A at the first round for which some language H contains S_t and satisfies $d(H) \leq q_t$. Until then, the algorithm will output only x_1. Each input during this initial part must either belong to the target or replay x_1, which also belongs to the target, so the retained sample A contains only target strings. For this rule to work, the intersection determined by A must remain infinite as q_t grows and more languages are included in $\mathcal{C}_{\leq q_t}$. Let s denote the number of distinct inputs when the sample is saved, and let $m \geq s$ be any possible later number of distinct inputs. The following implication states that the intersection stays infinite for all such m: for every s-element sample S and every pair of integers $m \geq s \geq 1$,

$$\mathcal{C}_{\leq s}[S] \neq \varnothing \quad \Longrightarrow \quad |\mathrm{cl}_{\mathcal{C}_{\leq m}}(S)| = \infty. \tag{5}$$

To prove the claim, suppose $\mathcal{C}_{\leq s}[S] \neq \varnothing$, choose L_0 from this collection, and consider $\mathcal{A} \coloneqq \mathcal{C}_{\leq m}[S]$. This subfamily contains L_0, all its thresholds are at most m, and its intersection contains S. If that intersection were finite, Equation (4) would give

$$s \leq |\bigcap_{H\in\mathcal{A}} H| < \min_{H\in\mathcal{A}} d(H) \leq d(L_0) \leq s,$$

a contradiction. This proves Equation (5).

We now specify the algorithm. Initially, it has no retained sample. After receiving x_t, if no sample has been retained and $\mathcal{C}_{\leq q_t}[S_t] \neq \varnothing$, it retains $A := S_t$ and uses this same A in every subsequent round. If no sample has been retained, it outputs x_1. Otherwise, including in the round when A is retained, it outputs the first string in

$$\mathrm{cl}_{\mathcal{C}_{\leq q_t}}(A) \setminus S_t.$$

To see that this choice is always possible, let $a := |A|$. The rule for retaining A ensures that $\mathcal{C}_{\leq a}[A] \neq \varnothing$. Since $q_t \geq a$ in that round and every later round, Equation (5) implies that $\mathrm{cl}_{\mathcal{C}_{\leq q_t}}(A)$ is infinite. Removing the finite input set S_t therefore leaves at least one string to output.

Finally, fix an arbitrary target L and a valid replay adversary. Before retaining a sample, the algorithm has output only $x_1 \in L$. Consequently, every input up to and including the round in which A is retained belongs to L, so $A \subseteq L$. If no sample has been retained when q_t first reaches $d(L)$, then $S_t \subseteq L$ and $L \in \mathcal{C}_{\leq q_t}[S_t]$, so the algorithm retains a sample in that round. Thus, in any round with $q_t \geq d(L)$, a sample $A \subseteq L$ has been retained. The target L belongs to $\mathcal{C}_{\leq q_t}[A]$ because it contains A and satisfies $d(L) \leq q_t$. Every string in $\mathrm{cl}_{\mathcal{C}_{\leq q_t}}(A)$ therefore belongs to L. The algorithm also excludes S_t, so its output lies in $L \setminus S_t$, as required. □

5 Countable and Uncountable Collections

For countable collections, we derive a simpler characterization that considers only intersections of finitely many languages. We then show that this condition is not sufficient for uncountable collections.

5.1 Characterization for Countable Collections

We derive the countable characterization by choosing thresholds so that any subfamily with bounded thresholds is finite. To do this, enumerate the languages and assign each one a threshold at least as large as its index. Only finitely many languages can then have thresholds below any fixed bound, so Theorem 4.1 requires us to examine only finite subfamilies.

To state the resulting criterion, fix a target L and consider a finite subfamily containing L. Its thresholds have a finite upper bound because it contains only finitely many languages. If its intersection is also finite, Theorem 4.1 requires the threshold for L to exceed the size of that intersection. We therefore ask whether these intersection sizes are bounded as the finite subfamily varies. Define

$$\kappa_0(L) := \sup \left\{ \left| \bigcap_{H \in \mathcal{F}} H \right| : L \in \mathcal{F} \subseteq \mathcal{C},\ |\mathcal{F}| < \infty,\ \left| \bigcap_{H \in \mathcal{F}} H \right| < \infty \right\}. \tag{6}$$

We set the supremum to 0 when no such family exists. If such finite intersections exist and their sizes are bounded, $\kappa_0(L)$ is their largest size. If their sizes are unbounded, $\kappa_0(L)$ is infinite. The next result shows that, for a countable collection, finiteness of every $\kappa_0(L)$ also suffices to construct an algorithm that generates under replay.

Corollary 5.1. *A countable collection $\mathcal{C}$ is non-uniformly generatable under replay if and only if $\kappa_0(L) < \infty$ for every $L \in \mathcal{C}$. Every achievable profile satisfies $d(L) > \kappa_0(L)$ for all $L \in \mathcal{C}$. Conversely, a profile satisfying these inequalities is achievable if, for every n, only finitely many languages have threshold at most n.*

The proof appears in Appendix B.1.

Optimal individual thresholds. Suppose $\mathcal{C}$ is non-uniformly generatable under replay. The lower bound $\kappa_0(L) + 1$ can be attained for any fixed target L by an algorithm that still has a finite threshold for every other target. More generally, for any prescribed finite $Q \subseteq \mathcal{C}$, one achievable profile has $d(L) = \kappa_0(L) + 1$ for all $L \in Q$. We prove this in Corollary B.1. However, a single algorithm need not attain these minima for all languages at once. For the co-singletons $H_j = \mathbb{N} \setminus \{j\}$, each individual minimum is 1, but no algorithm can give threshold 1 to every H_j. The proof and example appear in Appendix B.2.

5.2 Uncountable Collections

We now show that an uncountable collection can satisfy $\kappa_0(L) < \infty$ for every target and yet fail to admit non-uniform generation under replay. The lower bound $d(L) > \kappa_0(L)$ follows by applying Theorem 4.1 to finite subfamilies, so it remains necessary without countability. The converse for countable collections uses thresholds for which each $\mathcal{C}_{\le n}$ is finite. Every language has a finite integer threshold, so $\mathcal{C} = \bigcup_{n\ge 1} \mathcal{C}_{\le n}$. If every $\mathcal{C}_{\le n}$ were finite, this union would be countable. Our example has $\kappa_0(L) = 0$ for every target.

Proposition 5.2. *There is an uncountable collection $\mathcal{C}$ with $\kappa_0(L) = 0$ for every $L \in \mathcal{C}$ that is not non-uniformly generatable under replay.*

Proof of Proposition 5.2. We construct the universe from countably many disjoint infinite columns. Each column is itself a language. We also include every language obtained from the whole universe by deleting one point from each column. Formally, let $U = \mathbb{N}^2$, let $A_i = \{i\} \times \mathbb{N}$ be column i, and define

$$B_f := U \setminus \{(i, f(i)) : i \in \mathbb{N}\}, \qquad \mathcal{C} := \{A_i : i \in \mathbb{N}\} \cup \{B_f : f\colon \mathbb{N} \to \mathbb{N}\}.$$

The function f specifies the point deleted from each column. Every language is infinite, and distinct functions give distinct languages, so $\mathcal{C}$ is uncountable.

We first check that every finite subfamily has either empty or infinite intersection. If the subfamily contains two different columns, its intersection is empty because the columns are disjoint. If it contains exactly one column, each of its other languages deletes at most one point from that column. Finitely many such deletions leave an infinite intersection. If it contains no column languages, its members delete only finitely many points from each column, again leaving an infinite intersection. Consequently, $\kappa_0(L) = 0$ for every L.

To prove that $\mathcal{C}$ is not generatable, suppose that we can choose finite sets $E_H \subseteq U \setminus H$ for all $H \in \mathcal{C}$ so that $D_E(L)$ is finite for every target L. By Theorem 3.2, such a choice would have to exist if $\mathcal{C}$ were generatable. Fix a column A_i and a nonempty finite sample $S \subseteq A_i$. Consider a point (i, j) in this column but outside S. Every B_f deleting (i, j) contains all of S, since (i, j) is the only point it deletes from this column. This holds regardless of what f deletes from the other columns. Therefore (i, j) can belong to $K_E(S)$ only if every B_f deleting it adds it back. This condition depends on the point (i, j), but not on the other points of the sample. For each column, define

$$P_i := \{j \in \mathbb{N} : (i, j) \in E_{B_f} \text{ for every } f \text{ with } f(i) = j\}.$$

We will show that each P_i is infinite. Choosing one index from each P_i will then give a language B_f that must add back a point in every column, contradicting finiteness of E_{B_f}.

We claim that for every nonempty finite sample $S \subseteq A_i$,

$$K_E(S) \cap A_i = S \cup (\{i\} \times P_i). \tag{7}$$

Every point in S belongs to all the original languages consistent with S, and hence belongs to $K_E(S)$. Now fix $(i, j) \in A_i \setminus S$. The only column language consistent with the nonempty sample S is A_i, which contains this point. Among the other languages, exactly those B_f with $f(i) = j$ omit the point. Each such language contains all of S, because the point it deletes from column i is outside S. Therefore (i, j) belongs to $K_E(S)$ exactly when all these languages add it back, which is the condition $j \in P_i$. This proves Equation (7).

Since $D_E(A_i)$ is finite, choose a finite sample $S \subseteq A_i$ with $|S| > D_E(A_i)$. The definition of $D_E(A_i)$ makes $K_E(S)$ infinite. On the other hand, consistency of A_i gives $K_E(S) \subseteq A_i \cup E_{A_i}$, so only finitely many points of $K_E(S)$ can lie outside A_i. Its intersection with A_i must therefore be infinite. By Equation (7), this infinite intersection is $S \cup (\{i\} \times P_i)$. Since S is finite, P_i must be infinite.

We can now choose a function f by taking $f(i)$ to be the least member of P_i in every column. The language B_f deletes $(i, f(i))$ from column i. Since $f(i) \in P_i$, the definition of P_i requires B_f to add this point back. Thus $(i, f(i)) \in E_{B_f}$ for every i, giving infinitely many distinct additions to B_f. This contradiction proves the proposition. □

AI USE STATEMENT

Generative AI tools assisted with formulating mathematical claims, developing and checking proofs, and drafting and revising prose. We have reviewed all AI-assisted work and take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Yannan Bai, Debmalya Panigrahi, and Ian Zhang. Language generation in the limit: Noise, loss, and feedback. In Kasper Green Larsen and Barna Saha (eds.), *Proceedings of the 2026 Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2026, Vancouver, BC, Canada, January 11-14, 2026*, pp. 794–816. SIAM, January 2026. doi: 10.1137/1.9781611978971.31. URL https://doi.org/10.1137/1.9781611978971.31.

Moses Charikar and Chirag Pabbaraju. Exploring facets of language generation in the limit. In Nika Haghtalab and Ankur Moitra (eds.), *The Thirty Eighth Annual Conference on Learning Theory, 30-4 July 2025, Lyon, France*, volume 291 of *Proceedings of Machine Learning Research*, pp. 854–887. PMLR, June 2025a. URL https://proceedings.mlr.press/v291/charikar25a.html.

Moses Charikar and Chirag Pabbaraju. Pareto-optimal non-uniform language generation, October 2025b. URL https://arxiv.org/abs/2510.02795. To appear in ALT 2026.

Jon Kleinberg, Charlotte Peale, and Omer Reingold. Mistake-bounded language generation, May 2026. URL https://arxiv.org/abs/2605.10809.

Jon M. Kleinberg and Sendhil Mullainathan. Language generation in the limit. In Amir Globerson, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang (eds.), *Advances in Neural Information Processing Systems 37: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, December 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/7988e9b3876ad689e921ce05d711442f-Paper-Conference.pdf.

Jon M. Kleinberg and Fan Wei. Density measures for language generation. In *66th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2025, Sydney, Australia, December 14-17, 2025*, pp. 620–658. IEEE, December 2025. doi: 10.1109/FOCS63196.2025.00034. URL https://doi.org/10.1109/FOCS63196.2025.00034.

Jon M. Kleinberg and Fan Wei. Language generation and identification from partial enumeration: Tight density bounds and topological characterizations. In Aditya Bhaskara and Artur Czumaj (eds.), *Proceedings of the 58th Annual ACM Symposium on Theory of Computing, STOC 2026, Salt Lake City, UT, USA, June 22-26, 2026*, pp. 1684–1691. ACM, June 2026. doi: 10.1145/3798129.3800876. URL https://doi.org/10.1145/3798129.3800876.

Aaron Li and Ian Zhang. Characterizing the effect of noise in language generation in the limit, June 2026. URL https://arxiv.org/abs/2601.21237. To appear in ICML 2026.

Jiaxun Li, Vinod Raman, and Ambuj Tewari. Generation through the lens of learning theory. In Nika Haghtalab and Ankur Moitra (eds.), *The Thirty Eighth Annual Conference on Learning Theory, 30-4 July 2025, Lyon, France*, volume 291 of *Proceedings of Machine Learning Research*, pp. 4740–4776. PMLR, June 2025. URL https://proceedings.mlr.press/v291/raman25a.html.

Anay Mehrotra, Grigoris Velegkas, Xifan Yu, and Felix Zhou. Language generation with infinite contamination. In Steve Hanneke and Tor Lattimore (eds.), *The Thirty Ninth Annual Conference on Learning Theory, June 29 - July 3, 2026, San Diego, California, USA*, volume 336 of *Proceedings of Machine Learning Research*, pp. 5053–5112. PMLR, June 2026. URL https://proceedings.mlr.press/v336/mehrotra26a.html.

Giorgio Racca, Michal Valko, and Amartya Sanyal. Language generation with replay: A learning-theoretic view of model collapse, March 2026. URL https://arxiv.org/abs/2603.11784. To appear in ICML 2026.

Ananth Raman and Vinod Raman. Generation from noisy examples. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Forty-second International Conference on Machine Learning, ICML 2025, Vancouver, BC, Canada, July 13-19, 2025*, Proceedings of Machine Learning Research. PMLR / OpenReview.net, July 2025. URL https://proceedings.mlr.press/v267/raman25a.html.

A THRESHOLD PROFILES WITHOUT REPLAY

In this appendix, we characterize the threshold profiles achievable without replay. Comparing this characterization with Theorem 4.1 identifies exactly which profiles become impossible when replay is allowed. We derive that characterization here using the closure method of Li, Raman, and Tewari (2025).

In ordinary generation, the adversary must provide a string from the target in every round, so $x_t \in L$. We keep the same requirement on outputs as in Definition 2.1: the algorithm must output a string in $L \setminus S_t$ whenever $q_t \geq d(L)$, including the first round with $d(L)$ distinct inputs. The characterization below applies both when the adversary must enumerate the entire target and when it may provide any infinite sequence of target strings.

Proposition A.1. *A profile $d\colon \mathcal{C} \to \mathbb{N}$ is achievable without replay if and only if every nonempty $\mathcal{A} \subseteq \mathcal{C}$ with finite intersection satisfies*

$$|\bigcap_{L\in\mathcal{A}} L| < \sup_{L\in\mathcal{A}} d(L).$$

If the thresholds in a subfamily are unbounded, the displayed inequality holds automatically. When the thresholds have a common finite upper bound, their supremum is their largest value. For these subfamilies, ordinary generation requires the intersection size to be smaller than the largest threshold, whereas Theorem 4.1 requires it to be smaller than the smallest threshold under replay.

Proof. Suppose first that an algorithm achieves d, and assume that a nonempty subfamily $\mathcal{A}$ violates the stated condition. Its intersection K is finite and satisfies $\sup_{L\in\mathcal{A}} d(L) \leq |K|$. Since all thresholds are positive, K is nonempty. The adversary presents every point of K once. This gives the same input sequence for every possible target in $\mathcal{A}$. The algorithm is deterministic and is not told the target, so it also produces the same output after this sequence for every $L \in \mathcal{A}$. In the final round of this sample, the number of distinct inputs is $|K| \geq d(L)$ for every $L \in \mathcal{A}$. The algorithm's output must therefore belong to every language in $\mathcal{A}$ and must be unseen. This is impossible: their intersection is K, and every point of K has already appeared as an input. This finite prefix can be extended to a complete enumeration of each target separately, so the same argument applies when the model requires complete enumerations.

Conversely, suppose the condition holds. Every input now belongs to the target, so the algorithm may use the entire current sample to test consistency without excluding the target. After receiving x_t, it considers the languages that both contain S_t and have thresholds at most q_t:

$$\mathcal{B} := \mathcal{C}_{\leq q_t}[S_t].$$

If $\mathcal{B}$ is empty, the algorithm outputs x_1. Otherwise, every member contains all q_t observed strings, so their intersection has size at least q_t. If this intersection were finite, the assumed condition would require its size to be less than $\sup_{H\in\mathcal{B}} d(H) \leq q_t$, a contradiction. The intersection is therefore infinite, and the algorithm outputs its first string outside S_t.

To verify correctness, fix the actual target L and a round with $q_t \geq d(L)$. Since $S_t \subseteq L$, the language L belongs to $\mathcal{B}$ in this round. The algorithm's output belongs to the intersection of $\mathcal{B}$ and hence to L; its choice outside S_t also makes it unseen. The algorithm therefore achieves d. □

To understand this difference, consider a subfamily whose thresholds have a common finite upper bound, and an input sequence that is valid for every language in that subfamily. In ordinary generation, every input must lie in the common intersection. The algorithm is required to output a string in every language only once the number of distinct common inputs has reached every language's threshold, including the largest one. Under replay, the adversary can continue with previous outputs even when they do not lie in the common intersection. Once the count is at least the smallest $d(L)$ in the subfamily, the requirement for that language forces every output to be unseen. Replaying these outputs then increases the count until it is at least $d(L)$ for every language in the subfamily.

For example, consider a collection $\{L_1, L_2\}$ of two infinite languages with $|L_1 \cap L_2| = 5$. The ordinary criterion permits $d(L_1) = 3$ and $d(L_2) = 8$, since $5 < 8$. A sample contained in both languages has at most 5 strings, so it never requires correctness for L_2. Under replay, the adversary

can present 3 common strings and then replay each output. After these 3 distinct inputs, correctness for L_1 requires every subsequent output to be unseen, so the number of distinct inputs keeps increasing. These replays are permitted for both targets, even when an earlier output does not belong to both languages. Once the count reaches 8, correctness for both targets requires each new output to belong to their intersection. Continuing to replay these outputs would require infinitely many distinct strings in a set of size 5. Thus the same pair of thresholds is impossible under replay.

B Proofs for Countable Collections

We prove Corollary 5.1, which characterizes non-uniform generation for countable collections. We then show that, when generation is possible, each target's smallest achievable threshold is $\kappa_0(L) + 1$. These minima can also be achieved simultaneously on any prescribed finite set of targets.

B.1 Proof of the Countable Characterization

We first bound each $\kappa_0(L)$ using an achievable profile. We then construct an achievable profile whenever all the parameters $\kappa_0(L)$ are finite.

Proof of Corollary 5.1. Suppose first that an algorithm achieves the profile d. Fix $L \in \mathcal{C}$ and a finite subfamily $\mathcal{F}$ containing L whose intersection is finite. Because $\mathcal{F}$ contains only finitely many languages, their thresholds have a finite upper bound. We may therefore apply Theorem 4.1 to obtain $|\bigcap_{H\in\mathcal{F}} H| < d(L)$. Every intersection counted in $\kappa_0(L)$ consequently has size at most $d(L) - 1$. Taking their supremum gives $\kappa_0(L) \le d(L) - 1$.

Conversely, suppose $d(L) > \kappa_0(L)$ for every target and each $\mathcal{C}_{\le n}$ is finite. To verify Theorem 4.1, consider a nonempty subfamily $\mathcal{A}$ with bounded thresholds and finite intersection. The bound on its thresholds places $\mathcal{A}$ inside some finite collection $\mathcal{C}_{\le n}$, so $\mathcal{A}$ itself is finite. Choose a language $L_0 \in \mathcal{A}$ with the smallest threshold. Since $\mathcal{A}$ is one of the subfamilies counted in $\kappa_0(L_0)$, its intersection has size at most $\kappa_0(L_0) < d(L_0)$. This is the inequality required by Theorem 4.1.

Finally, suppose $\kappa_0(L)$ is finite for every target. We construct a profile that satisfies both conditions just used. Enumerate the collection without repetition as $\{L_i : i \in I\}$, where $I \subseteq \mathbb{N}$, and assign

$$d(L_i) := \max\{i, \kappa_0(L_i) + 1\}.$$

The second term ensures $d(L_i) > \kappa_0(L_i)$. The first ensures that a language with threshold at most n has index at most n, so $\mathcal{C}_{\le n}$ is finite. The preceding argument proves that this profile is achievable. □

The same parameter also explains the countable impossibility example of Racca, Valko, and Sanyal (2026). On $U = \mathbb{Z}$, consider $\mathcal{C} = \{L_\infty\} \cup \{L_n : n \in \mathbb{N}\}$, where $L_\infty = \mathbb{N}$ and $L_n = \{1, \dots, n\} \cup \mathbb{Z}_{<0}$. The pair $\{L_\infty, L_n\}$ has intersection $\{1, \dots, n\}$, so $\kappa_0(L_\infty) = \infty$. No finite threshold for L_∞ can exceed all these intersection sizes, so Corollary 5.1 recovers their impossibility result.

B.2 Optimal Individual Thresholds

For a countable collection that can be non-uniformly generated under replay, we show that the thresholds $\kappa_0(L) + 1$ can be attained simultaneously on any prescribed finite set of targets. Requiring each language's threshold to be at least its index in an enumeration can make that threshold larger than $\kappa_0(L) + 1$. If we wish to optimize the threshold for one particular language, we can remove the index requirement for that language and keep it for all others. The same argument works for any prescribed finite set of languages: only finitely many exceptions are added to each collection $\mathcal{C}_{\le n}$.

Corollary B.1. *Suppose $\mathcal{C}$ is countable and non-uniformly generatable under replay. For each target L, the smallest value of $d(L)$ among all achievable profiles d is $\kappa_0(L) + 1$. Moreover, for every finite $Q \subseteq \mathcal{C}$, there exists one achievable profile d with $d(L) = \kappa_0(L) + 1$ for every $L \in Q$.*

Proof. Assign $d(L) = \kappa_0(L) + 1$ to every $L \in Q$. Enumerate the remaining languages and assign $d(L_i) = \max\{i, \kappa_0(L_i) + 1\}$ to them. For each n, a language with threshold at most n either belongs to Q or has index at most n. Thus $\mathcal{C}_{\le n}$ is finite, and Corollary 5.1 proves that the profile is achievable. The lower bound in that corollary rules out any smaller threshold for a language in Q. □

For the co-singleton languages $H_j = \mathbb{N} \setminus \{j\}$ from Example 3.1, every finite subfamily has infinite intersection, so $\kappa_0(H_j) = 0$ for every j. It follows that one algorithm can achieve threshold 1 on any prescribed finite set of these targets. This conclusion does not extend to all the targets at once. Indeed, for a fixed k, the infinite subfamily $\{H_j : j \neq k\}$ has intersection $\{k\}$. Assigning threshold 1 to all its members would violate Theorem 4.1, since the intersection has size equal to their smallest threshold. Thus each language can have optimal threshold 1 even though no algorithm achieves all of these optima together.

C Noise and Randomization

We extend the profile characterization to a model that permits both replay and a known finite number of distinct noisy strings outside the target. For countable collections, we also determine the optimal threshold for each target and compare what can be achieved at different noise levels. Finally, we show that randomization cannot improve achievable profiles when correctness is required with probability one.

The replay model allows inputs outside the target only when they have previously been output by the algorithm. We now combine replay with the noisy generation model introduced by Raman and Raman (2025). The variant in which the algorithm knows the noise level was introduced by Bai, Panigrahi, and Zhang (2026) and further studied by Li & Zhang (2026). In addition to replaying outputs, the adversary may introduce noisy strings outside the target, subject to a known finite bound on the number of distinct strings introduced this way. Throughout this appendix, the universe is countably infinite and every language is infinite. The collection of languages may be uncountable unless we explicitly assume otherwise.

C.1 Replay with a Known Noise Level

We first specify which inputs the adversary may provide when replay and external noise are both allowed. The noise level is a fixed integer $b \geq 0$ known to the algorithm. In addition to providing target strings and replaying outputs, the adversary may introduce a string outside the target that the algorithm has not previously output. Across the interaction, it may introduce at most b distinct strings in this way. We call these strings *noisy strings*. Formally, a transcript is valid for a target L under *replay with noise level* b if some fixed set $E \subseteq U \setminus L$ with $|E| \leq b$ satisfies

$$x_t \in L \cup E \cup \{o_1, \ldots, o_{t-1}\} \qquad \text{for every round } t.$$

Repeating a noisy string does not increase the number of distinct noisy strings. Replaying an algorithm output outside the target does not count toward the noise bound. The algorithm knows b, but it knows neither L nor E and receives no indication of which source produced a given input.

A profile $d\colon \mathcal{C} \to \mathbb{N}$ requires $o_t \in L \setminus S_t$ in every round with $q_t \geq d(L)$, for every target and every valid adaptive adversary. The count $q_t = |S_t|$ includes all distinct inputs, including noisy strings and replayed outputs. Thus the adversary can use these inputs to reach the threshold, but the algorithm must still generate a string in the target that has not appeared as an input. Taking $b = 0$ recovers the replay model studied in the main text.

Before defining the characterization, we adapt consistency and closure to allow these noisy strings, following Raman and Raman (2025). Suppose a finite sample S consists only of target strings and noisy strings. A language L is *consistent with* S *at noise level* b if $|S \setminus L| \leq b$: the strings in $S \setminus L$ are the noisy strings needed for that explanation. For a subfamily $\mathcal{B} \subseteq \mathcal{C}$, we therefore define

$$\mathcal{B}_b[S] := \{L \in \mathcal{B} : |S \setminus L| \leq b\}, \qquad \mathrm{cl}_{\mathcal{B}}^{(b)}(S) := \bigcap_{L \in \mathcal{B}_b[S]} L.$$

As before, the intersection is U if no language in $\mathcal{B}$ is consistent with S. The closure consists of the strings belonging to every language that can explain the sample using at most b noisy strings. At $b = 0$, these are exactly the consistency and closure definitions from the main text. After the algorithm has output strings outside the target, however, the adversary may replay those strings without these replays counting toward the noise bound. The full observed sample can then have more than b points outside the true target. This is why our algorithm will again retain an earlier sample for all future consistency tests.

When $b = 0$, a sample is consistent with every language in a subfamily only if it lies in their intersection. With noise, the sample can be larger: up to b sample strings may lie outside each language. We need to measure the largest size of a sample that all these languages can explain. For every nonempty $\mathcal{F} \subseteq \mathcal{C}$, define its *capacity* by

$$c_b(\mathcal{F}) := \sup\{|S| : S \subseteq U \text{ finite}, \ |S \setminus L| \leq b \text{ for all } L \in \mathcal{F}\}. \tag{8}$$

The capacity is a nonnegative integer, or infinity if such samples have unbounded size. When the capacity is finite, some sample attains it, since all sample sizes are integers. The same sample may require different sets of noisy strings for different languages; only the bound of b noisy strings must hold for every language. When $b = 0$, all sample points must lie in the common intersection, giving $c_0(\mathcal{F}) = |\bigcap_{L \in \mathcal{F}} L|$.

C.2 Threshold Profiles with Noise

We characterize exactly which threshold profiles can be achieved under replay with noise level b. The following theorem replaces the intersection size in Theorem 4.1 with the capacity $c_b(\mathcal{F})$ of the subfamily. The condition that the subfamily's intersection be finite remains unchanged. The third equivalent condition describes the intersections used by the algorithm. If an s-element sample is consistent with a language whose threshold is at most s, then for every finite bound $m \geq s$, the consistent languages with thresholds at most m must have an infinite intersection. We continue to write $\mathcal{C}_{\leq n} = \{L \in \mathcal{C} : d(L) \leq n\}$.

Theorem C.1. *Fix $b \geq 0$ and a profile $d \colon \mathcal{C} \to \mathbb{N}$. The following are equivalent.*

1. *The profile d is achievable under replay with noise level b.*
2. *Every nonempty $\mathcal{F} \subseteq \mathcal{C}$ with finite intersection and $\sup_{L \in \mathcal{F}} d(L) < \infty$ satisfies*
$$c_b(\mathcal{F}) < \min_{L \in \mathcal{F}} d(L).$$
3. *For every pair of integers $1 \leq s \leq m$ and every s-element sample $S \subseteq U$,*
$$(\mathcal{C}_{\leq s})_b[S] \neq \varnothing \quad \Longrightarrow \quad |\mathrm{cl}^{(b)}_{\mathcal{C}_{\leq m}}(S)| = \infty. \tag{9}$$

Proof. We first prove that the first condition implies the second. Suppose an algorithm achieves d, and let $\mathcal{F}$ be a nonempty subfamily with finite intersection and thresholds bounded by an integer m. Write $s = \min_{L \in \mathcal{F}} d(L)$. If $c_b(\mathcal{F}) \geq s$, some sample of size at least s is consistent with every member of $\mathcal{F}$. Select s of its points to obtain a sample S with the same consistency property. The adversary presents each point of S once and then replays the latest algorithm output in every round. This strategy is valid for every $L \in \mathcal{F}$: the initial sample uses only the noisy strings $S \setminus L$, and all subsequent inputs are replays. The algorithm sees the same transcript for all these possible targets.

A target with threshold s requires the output in round s to be unseen. Replaying that output increases the number of distinct inputs to $s+1$, at which point the next output must again be unseen. Inductively, $q_t = t$ for every $t \geq s$, and the outputs from round s onward are pairwise distinct. In every round $t \geq m$, the number of distinct inputs is at least $d(L)$ for every $L \in \mathcal{F}$. The algorithm must therefore output a string belonging to every one of these languages. This would give infinitely many distinct points in a finite set, a contradiction.

We next show that the second condition implies the third. Fix $s \leq m$ and an s-element sample S satisfying the premise of Equation (9). Consider all languages with threshold at most m that are consistent with this sample, namely $\mathcal{F} = (\mathcal{C}_{\leq m})_b[S]$. This family is nonempty, and its minimum threshold is at most s by the premise. Its capacity is at least s, because S is consistent with all its members. If its intersection were finite, the second condition would give $s \leq c_b(\mathcal{F}) < \min_{L \in \mathcal{F}} d(L) \leq s$, a contradiction. Thus its intersection, which is the closure in the third condition, is infinite.

Finally, suppose the third condition holds. We construct an algorithm using the same idea as in Section 4. The algorithm initially repeats x_1, so its outputs cannot introduce any new string into the transcript. It continues until the current sample is consistent with some language H and contains at least $d(H)$ distinct points. It then retains that sample permanently. Before the sample is retained,

every input belongs to the target or the set of noisy strings: the only string that can be replayed is x_1, which already belongs to one of those sets. Thus the retained sample contains at most b points outside the target.

More precisely, let τ be the first round for which $(\mathcal{C}_{\leq q_\tau})_b[S_\tau] \neq \varnothing$. The algorithm outputs x_1 before round τ and retains $A = S_\tau$ in that round. From round τ onward, including round τ itself, it outputs the first point of

$$\mathrm{cl}^{(b)}_{\mathcal{C}_{\leq q_t}}(A) \setminus S_t$$

in the fixed enumeration of U. If no such round τ occurs, it continues to output x_1. Once A is retained, only the threshold bound q_t and the set of inputs to avoid change; the algorithm always tests consistency against A. To check that an output is available, write $a = |A|$. The rule for retaining A gives $(\mathcal{C}_{\leq a})_b[A] \neq \varnothing$. In the round when A is retained and every later round, $q_t \geq a$. Therefore Equation (9), applied with $s = a$ and $m = q_t$, makes the relevant closure infinite. Removing the finite set S_t therefore leaves at least one possible output.

To prove correctness, fix an arbitrary target L and a valid adversary with set E of noisy strings. The first input belongs to $L \cup E$, because no earlier output is available to replay. Before retaining a sample, the algorithm outputs only this first input. Thus every input up to and including round τ lies in $L \cup E$, giving $|A \setminus L| \leq b$ whenever a sample is retained. If the number of distinct inputs reaches $d(L)$ before any sample has been retained, the same argument shows that S_t is consistent with L in that round. Since L also has threshold at most q_t, the algorithm must retain a sample by that round. Consequently, in every round with $q_t \geq d(L)$, a sample A has been retained and L belongs to $(\mathcal{C}_{\leq q_t})_b[A]$. The chosen output lies in this family's intersection and outside S_t, so it belongs to $L \setminus S_t$, as required. □

C.3 Countable Collections and Optimal Target Thresholds

For countable collections, we characterize when non-uniform generation with noise level b is possible and determine the smallest achievable threshold for each target. As in the case without noise, we can choose thresholds so that only finitely many languages have thresholds below any fixed bound. We will then need to check the capacity condition only for finite subfamilies. We first verify that the capacity of each such subfamily is finite whenever its intersection is finite. Let $\mathcal{F}$ be a finite family with finite intersection K, and let S be consistent with every member. Each point of $S \setminus K$ is omitted by at least one language in $\mathcal{F}$, and each language omits at most b sample points. Counting these omissions gives

$$|S \setminus K| \leq \sum_{H \in \mathcal{F}} |S \setminus H| \leq b|\mathcal{F}|.$$

The sample also contains at most $|K|$ points in K, so

$$c_b(\mathcal{F}) \leq |K| + b|\mathcal{F}| < \infty. \tag{10}$$

Although each of these capacities is finite, their values may grow without bound as we vary the finite subfamily while keeping one target L in it. Just as in the case $b = 0$, such growth would rule out every finite threshold for L. We therefore define

$$\kappa_b(L) := \sup \left\{ c_b(\mathcal{F}) : L \in \mathcal{F} \subseteq \mathcal{C},\ \mathcal{F} \text{ finite}, |\bigcap_{H \in \mathcal{F}} H| < \infty \right\}. \tag{11}$$

We assign value 0 if there is no such family. At $b = 0$, capacity equals the size of the common intersection, so this definition agrees with $\kappa_0(L)$ from Corollary 5.1.

Corollary C.2. *A countable collection $\mathcal{C}$ is non-uniformly generatable under replay with noise level b if and only if $\kappa_b(L) < \infty$ for every $L \in \mathcal{C}$. Every achievable profile satisfies $d(L) > \kappa_b(L)$ at every target. Conversely, a profile satisfying these inequalities is achievable if every $\mathcal{C}_{\leq n}$ is finite. If all $\kappa_b(L)$ are finite, any finite set of targets can simultaneously receive their optimal thresholds $\kappa_b(L) + 1$.*

Proof. Suppose first that d is achievable. Fix a finite subfamily $\mathcal{F}$ containing L and having finite intersection. Its thresholds are bounded because it is finite, so Theorem C.1 gives $c_b(\mathcal{F}) <$

$\min_{H \in \mathcal{F}} d(H) \leq d(L)$. Each of these capacities is an integer at most $d(L) - 1$, so their supremum satisfies $\kappa_b(L) \leq d(L) - 1$.

Conversely, suppose $d(L) > \kappa_b(L)$ for every target and every $\mathcal{C}_{\leq n}$ is finite. Consider any nonempty subfamily with bounded thresholds and finite intersection. It lies inside some $\mathcal{C}_{\leq n}$ and hence is finite. If H is a member with the smallest threshold, the definition of $\kappa_b(H)$ bounds this subfamily's capacity by $\kappa_b(H) < d(H)$. The profile therefore satisfies Theorem C.1 and is achievable.

If every $\kappa_b(L)$ is finite, we obtain such a profile by enumerating the collection without repetition as $\{L_i : i \in I\}$, where $I \subseteq \mathbb{N}$, and setting $d(L_i) = \max\{i, \kappa_b(L_i) + 1\}$. The second term makes the required inequality strict, and the first ensures that only languages with index at most n can belong to $\mathcal{C}_{\leq n}$. To attain the lower bounds on a prescribed finite set Q, instead assign $d(L) = \kappa_b(L) + 1$ on Q and apply the enumeration rule to $\mathcal{C} \setminus Q$. Each $\mathcal{C}_{\leq n}$ gains only finitely many exceptions from Q, so it remains finite. This proves achievability, and the lower bound already proved gives optimality on Q. □

The last assertion allows the optimal thresholds for any prescribed finite set of targets to be achieved by one algorithm. As in Corollary B.1, it does not imply that a single algorithm achieves all individual optima when the collection is infinite.

C.4 Comparing Positive Noise Levels

We show that a countable collection can be non-uniformly generated with noise level 1 if and only if it can be generated with every fixed finite noise level $b \geq 1$. We prove this by comparing the parameters $\kappa_b(L)$ that determine each target's optimal threshold. In particular, the bound below shows that if $\kappa_1(L)$ is finite, then $\kappa_b(L)$ is finite for every fixed finite $b \geq 1$.

Theorem C.3. *For every collection $\mathcal{C}$, every $L \in \mathcal{C}$, and every integer $b \geq 1$,*

$$\kappa_1(L) \leq \kappa_b(L) \leq b\kappa_1(L).$$

The factor b is optimal.

Proof. Any sample consistent with a language at noise level 1 remains consistent with it at noise level $b \geq 1$. Thus every capacity in the definition of $\kappa_1(L)$ is at most its counterpart in the definition of $\kappa_b(L)$, proving the first inequality.

To prove the upper bound, fix a finite subfamily $\mathcal{F}$ containing L and having finite intersection. We will reduce to a subfamily in which each language other than L omits a point contained in all the others. These points will form a sample consistent with every language in the subfamily at noise level 1. To obtain this subfamily, remove any language other than L whose removal still leaves a finite intersection, and continue until no further removal is possible. Let $\mathcal{F}' \subseteq \mathcal{F}$ be the resulting subfamily. Removing languages only removes consistency requirements, so $c_b(\mathcal{F}) \leq c_b(\mathcal{F}')$. Write $K = \bigcap_{H \in \mathcal{F}'} H$, $k = |K|$, and $r = |\mathcal{F}'|$. Since a single language is infinite while K is finite, we have $r \geq 2$.

Each language $H \in \mathcal{F}' \setminus \{L\}$ is now necessary to keep the intersection finite. Removing H leaves an infinite intersection. Since K is finite, this intersection contains a point outside K, and that point belongs to every other member of $\mathcal{F}'$. Choose such a point for each H:

$$y_H \in \left(\bigcap_{J \in \mathcal{F}' \setminus \{H\}} J \right) \setminus K.$$

Because y_H lies outside the common intersection but belongs to every member other than H, it must be omitted by H. The points chosen for different languages are distinct: y_H belongs to any other language J, whereas y_J does not belong to J. Thus adding all these points to K makes each member of $\mathcal{F}' \setminus \{L\}$ omit exactly one point, while L omits none. The sample $K \cup \{y_H : H \in \mathcal{F}' \setminus \{L\}\}$ is therefore consistent with every member when one noisy string is allowed. Its size is $k + r - 1$, giving $\kappa_1(L) \geq k + r - 1$.

We now ask whether a point belongs to every member of $\mathcal{F}' \setminus \{L\}$ but not to L. If such a point z exists, it is distinct from K and all the y_H, since those points belong to L. Adding z gives a sample

of size $k + r$ that still requires only one noisy string for each language. Hence $\kappa_1(L) \geq k + r$, and Equation (10) gives

$$c_b(\mathcal{F}') \leq k + br \leq b(k + r) \leq b\kappa_1(L).$$

If no such point z exists, every point outside K is omitted by at least one language other than L. We can therefore bound the size of a sample consistent with every member of $\mathcal{F}'$ by counting omissions from only these $r - 1$ languages. Each omits at most b sample points, so at most $b(r - 1)$ sample points lie outside K. Combining this bound with $\kappa_1(L) \geq k + r - 1$ gives

$$c_b(\mathcal{F}') \leq k + b(r - 1) \leq b(k + r - 1) \leq b\kappa_1(L).$$

In either case, $c_b(\mathcal{F}) \leq c_b(\mathcal{F}') \leq b\kappa_1(L)$. Taking the supremum over the finite subfamilies $\mathcal{F}$ proves the upper bound. If no finite subfamily containing L has finite intersection, both parameters are zero by definition.

To see that the factor b is optimal, take a collection consisting of two disjoint infinite languages A and B on $U = A \sqcup B$. The pair $\{A, B\}$ is its only nonempty subfamily with finite intersection. For a sample to be consistent with A, it must contain at most b points from B; consistency with B similarly allows at most b points from A. A sample of size $2b$ is possible by choosing b points from each language. Consequently, $\kappa_b(A) = 2b = b\kappa_1(A)$. □

Corollary C.4. *A countable collection is non-uniformly generatable under replay with noise level* 1 *if and only if it is non-uniformly generatable under replay with noise level* b *for every fixed finite* $b \geq 1$.

Proof. If the collection is generatable with one noisy string, Corollary C.2 makes $\kappa_1(L)$ finite for every target. For each fixed finite $b \geq 1$, Theorem C.3 then makes $\kappa_b(L)$ finite for every target as well. Applying Corollary C.2 again gives generation with noise level b. The reverse implication includes the choice $b = 1$. □

The corollary asserts existence separately for each known noise level; both the algorithm and its thresholds may depend on b. Indeed, increasing the noise level can increase the optimal thresholds. For the two disjoint languages used above, each optimal threshold is $2b + 1$. For comparison, Li and Zhang (2026) proved that positive finite noise levels are equivalent for generation without replay over arbitrary collections, both when thresholds may depend on the target and when one threshold must work for every target. Our result allows replay and uses countability to pass from the parameter bound to the existence of an algorithm.

C.5 Separation from Replay without Noise

We construct a countable collection that can be non-uniformly generated under replay without noise, but cannot be non-uniformly generated when even one noisy string is allowed. The following construction gives finite subfamilies with empty intersection whose members each omit at most one point of a large sample. Each subfamily contains the same fixed target, and the sizes of these samples are unbounded. At the same time, every finite subfamily has either empty or infinite intersection, so Corollary 5.1 still permits generation without noise.

Example C.5. There is a countable collection $\mathcal{C}$ with $\kappa_0(L) = 0$ for every $L \in \mathcal{C}$ and $\kappa_1(L_\star) = \infty$ for some target $L_\star \in \mathcal{C}$. Thus it is non-uniformly generatable under replay without noise but not under replay with noise level 1.

Proof. We need arbitrarily large samples that are consistent with several languages using one noisy string per language, even though those languages have empty intersection. For each $n \geq 2$, we therefore introduce an n-element set S_n and n languages, each of which contains all but a different point of S_n. Taken together, these languages have no common point in S_n.

We must also make every language infinite without creating large finite intersections, since those would obstruct generation even without noise. To do this, we use a separate infinite set T of sequences. Define

$$T \coloneqq \{(a_2, a_3, \dots) : a_n \in \{1, \dots, n\}, a_n = 1 \text{ for all but finitely many } n\}.$$

Every sequence in T agrees with the all-ones sequence after some coordinate N. For each fixed N, only finitely many such sequences are possible, so T is countable. It is infinite because changing

coordinate n to 2, for each $n \geq 2$, gives distinct members of T. Let $S_n = \{s_{n,1}, \ldots, s_{n,n}\}$ for each $n \geq 2$, choosing these sets to be mutually disjoint and disjoint from T. Define $U = T \sqcup \bigsqcup_{n \geq 2} S_n$ and include the whole universe as a target $L_\star = U$. This target will belong to each subfamily witnessing a large value of $\kappa_1(L_\star)$. The other languages are

$$L_{n,i} \coloneqq \{a \in T : a_n \neq i\} \cup (S_n \setminus \{s_{n,i}\}), \qquad n \geq 2, \quad 1 \leq i \leq n,$$

and the collection is $\mathcal{C} = \{L_\star\} \cup \{L_{n,i} : n \geq 2,\ 1 \leq i \leq n\}$. Since $n \geq 2$, we can choose a value other than i for coordinate n. Keeping that value fixed and changing one other coordinate at a time produces infinitely many sequences in T belonging to $L_{n,i}$.

We now check that every finite subfamily has either infinite or empty intersection. Its languages impose finitely many restrictions of the form $a_n \neq i$ on sequences in T; including $L_\star$ adds no restriction. Suppose first that at least one allowed value remains at every coordinate. Choose one such value at each constrained coordinate and set all other coordinates to 1. This gives a sequence in the intersection. Only finitely many coordinates are constrained. For each remaining coordinate, changing that coordinate alone from 1 to 2 gives another sequence in T that satisfies every restriction. These sequences are all distinct, so the intersection is infinite.

Otherwise, the subfamily excludes every value at some coordinate n, which means it contains all of $L_{n,1}, \ldots, L_{n,n}$. These n languages have no common sequence in T. Outside T, every one of them contains points only from S_n, and each point $s_{n,i}$ is omitted by $L_{n,i}$. They therefore have empty intersection in the whole universe. Any larger subfamily also has empty intersection. This proves that the only finite intersection size arising from a finite subfamily is zero, and hence $\kappa_0(L) = 0$ for every target.

Finally, for each n, consider the finite family $\mathcal{F}_n = \{L_\star, L_{n,1}, \ldots, L_{n,n}\}$. We have just shown that its intersection is empty. Nevertheless, every member is consistent with the n-element sample S_n when one noisy string is allowed: $L_\star$ contains the whole sample, and $L_{n,i}$ omits only $s_{n,i}$. Thus $\kappa_1(L_\star) \geq c_1(\mathcal{F}_n) \geq n$ for every n, so $\kappa_1(L_\star)$ is infinite. By Corollary C.2, the collection is generatable under replay without noise but not when one noisy string is allowed. □

C.6 Almost-Sure Derandomization

We show that randomization does not enlarge the set of achievable threshold profiles when correctness is required with probability one. We require the randomized algorithm to satisfy the same guarantee as before with probability one: for each target L and each valid adaptive adversary, every output must belong to $L \setminus S_t$ once the number of distinct inputs is at least $d(L)$. Formally, the algorithm achieves d *almost surely* if

$$\Pr[o_t \in L \setminus S_t \text{ for every round } t \text{ with } q_t \geq d(L)] = 1.$$

The probability is over the algorithm's private randomness. The adversary can observe its outputs but not the random choices that produced them. If a transcript never contains $d(L)$ distinct inputs, the event in the display holds vacuously, just as the deterministic guarantee does. The following theorem shows that randomization cannot improve any of the thresholds under this requirement.

Theorem C.6. *For every fixed $b \geq 0$ and every collection over a countable universe, a profile is achievable almost surely under replay with noise level b if and only if it is achievable deterministically.*

The collection of targets may be uncountable, so we cannot simply intersect the probability-one correctness events for all targets.

Proof. Suppose a randomized algorithm achieves d almost surely. By Theorem C.1, it suffices to verify Equation (9). Fix integers $1 \leq s \leq m$, a sample S of size s, and a language $L_0 \in (\mathcal{C}_{\leq s})_b[S]$. Thus S is consistent with L_0 and already contains at least $d(L_0)$ distinct inputs. The adversary presents the points of S once each and then returns the latest algorithm output as its next input forever. This strategy is valid for every $L \in (\mathcal{C}_{\leq m})_b[S]$, using $S \setminus L$ as the set of noisy strings. The algorithm is not told which target was chosen, and the adversary's rule does not depend on that choice. Its inputs and outputs therefore have the same joint distribution for every such target.

The guarantee for L_0 implies that, with probability one, every output from round s onward is unseen. On this event, each replay adds a new point to the observed set, so $q_t = t$ for every $t \geq s$. These outputs are also pairwise distinct, since each previous output has already been replayed when the

algorithm chooses the next one. In particular, for every $t \geq m$ and every $L \in (\mathcal{C}_{\leq m})_b[S]$, we have $q_t \geq m \geq d(L)$ with probability one. Applying the guarantee for this target L gives $\Pr[o_t \in L] = 1$.

Fix one round $t \geq m$. To turn these separate membership statements into membership in the common intersection, consider the set of strings that the algorithm outputs with positive probability in this round:

$$P_t \coloneqq \{x \in U : \Pr[o_t = x] > 0\}.$$

Every point of P_t belongs to every language in $(\mathcal{C}_{\leq m})_b[S]$. Indeed, if a language L omitted some $x \in P_t$, then $\Pr[o_t \notin L] \geq \Pr[o_t = x] > 0$, contradicting the membership statement just proved. Thus $P_t \subseteq \mathrm{cl}^{(b)}_{\mathcal{C}_{\leq m}}(S)$. For every $x \in U \setminus P_t$, the event $\{o_t = x\}$ has probability zero. The event $\{o_t \notin P_t\}$ is the union of these countably many events, so it too has probability zero, giving $\Pr[o_t \in P_t] = 1$. Therefore

$$\Pr\left[o_t \in \mathrm{cl}^{(b)}_{\mathcal{C}_{\leq m}}(S)\right] = 1.$$

There are only countably many rounds $t \geq m$, so with probability one all these outputs belong to the closure. Intersecting this event with the probability-one event of pairwise distinct outputs from round s onward, we obtain infinitely many distinct points in the closure. The closure is therefore infinite, as required by Equation (9). The deterministic characterization now supplies an algorithm achieving d. Conversely, any deterministic algorithm is a randomized algorithm that ignores its random choices, so deterministic achievability implies almost-sure achievability. □