# Generating in the Limit with Infinitely Many Hallucinations

**Irene Strauss***
University of Cambridge
is657@cam.ac.uk

**Alexandra Butoi**
ETH Zürich
abutoi@ethz.ch

**Ryan Cotterell**
ETH Zürich
rcotterell@ethz.ch

## Abstract

The classic paradigm of language identification in the limit models learning as a game between an adversary, who reveals strings from an unknown target language, and a learner tasked with identifying that language. The recently introduced framework of language generation in the limit shifted the objective to better reflect modern language modeling, requiring the learner to produce valid, unseen strings from the target language. Related work highlighted a fundamental tension: a broad coverage of the target often comes at the cost of validity. We introduce a new notion of precision and recast this problem within a precision–recall framework. We analyze generation in the limit under varying constraints on enumeration, novelty, and validity, aimed at reflecting settings closer to those encountered by large language models. A key contribution is our analysis of learners that are not eventually valid: we allow infinitely many mistakes, provided their frequency tends to zero so that precision remains one. We show that this relaxation can strictly increase recall when the adversary permanently withholds a large portion of the target language. We also study a continuous relaxation of the novelty constraint that requires only a fixed fraction of outputs to be novel. Taken together, our results move toward a more realistic model of language generation where occasional errors and repetitions are unavoidable, but their rates are controlled.

## 1 Introduction

Language identification in the limit is an influential learning paradigm (Gold, 1967; Angluin, 1980). In the classic Gold–Angluin model, a learner observes an infinite stream of strings from a target language $T$, where $T$ is assumed to belong to a countable collection of languages $\mathcal{T}$. At each timestep, the learner outputs a hypothesis in $\mathcal{T}$; it succeeds if there exists some finite (but unknown) time after which its hypotheses stabilize to the correct language $T$. In full generality, identification in this setting is impossible for most interesting collections of languages, i.e., even for regular languages. Angluin (1980) gave a characterization of the countable collections $\mathcal{T}$ that allow identification. In recent work, Kleinberg & Mullainathan (2024) introduced the paradigm of language *generation* in the limit, inspired by modern language models. Again, the learner is given access to a countable collection of languages $\mathcal{T}$. Instead of identifying the target language $T$, it must produce, after some finite timestep, only *valid* (belonging to $T$), previously unseen strings. The authors showed that generation in the limit is tractable for any countable collection of languages $\mathcal{T}$.

While the algorithm of Kleinberg & Mullainathan (2024) ensures validity, it leaves the question of *coverage* unexplored, i.e., how much of $T$ is generated. A learner can satisfy the validity requirement by generating from some arbitrarily small, albeit still infinite, subset of $T$. In doing so, the learner avoids errors but fails to cover most of $T$. Subsequent work formalized this tension, showing a trade-off between coverage and validity (Charikar & Pabbaraju, 2025) and establishing that achieving

*Work done while at ETH Zürich.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

full coverage is as hard as language identification (Kalavasis et al., 2025). Recently, Kleinberg & Wei (2025a) introduced a notion of *density*. In follow-up work (Kleinberg & Wei, 2025b), they prove that, in a partial enumeration setting, i.e., when the learner is shown strings from a subset of the target language $T$ with density $\alpha$, a language can be generated with density $\alpha/2$.

We recast this problem as a recall–precision problem, starting from the observation that the coverage from previous work is equivalent to recall. We introduce a complementary notion of precision that quantifies the learner's error rate, better reflecting modern language-modeling settings where occasional errors are unavoidable and the key question is their rate. Because the standard definitions of precision and recall are ill-defined for infinite sets, we define them via exhaustions, i.e., sequences of finite sets that approximate an infinite set. We also formalize the validity requirement by introducing a notion that we call tail precision.

We derive precision and recall bounds across a range of settings (summarized in Figure 1), obtained by varying: (i) the fraction of strings that must be novel, (ii) whether the adversary reveals a full or only a partial exhaustion of the target language, and (iii) whether the learner is required to be eventually valid (tail precision of one). Several of these settings have not been studied previously. Our main contributions and improvements over prior work are as follows. First, in the partial enumeration setting with novelty and perfect tail precision, the best known recall bound is due to Kleinberg & Wei (2025b). We show that relaxing tail precision—allowing hallucinations at a vanishing rate—strictly improves recall whenever the adversary is sufficiently sparse. In the most extreme case, where the adversary reveals almost nothing, our algorithm achieves coverage approaching one, while the no-hallucination bound approaches zero. Second, we introduce a continuous novelty parameter that interpolates between the strict novelty constraint and the no-novelty setting, and give an algorithm with precision one and recall guarantees across the full spectrum. This yields a new setting not studied in prior work: any fixed allowance of non-novel outputs ($\gamma < 1$) allows generation in the limit with recall one and precision one. As technical contributions enabling these results, we introduce an algorithmic framework that interleaves *safe* algorithms with *exploration* rounds that might contain hallucinations, and we generalize the algorithm of Kleinberg & Wei (2025b) to a batched setting where the adversary can reveal multiple strings per round.

## 2 Approximate language learning

### 2.1 Preliminaries

An **alphabet** is a finite, non-empty set $\Sigma$. A **string**, denoted by $\boldsymbol{y}$ or $\boldsymbol{x}$, is a finite sequence of symbols. We write $\mathbb{N} \stackrel{\text{def}}{=} \{1, 2, \ldots\}$ for the positive integers and $\mathbb{N}_0 \stackrel{\text{def}}{=} \{0, 1, 2, \ldots\}$ for the non-negative integers. The Kleene closure of an alphabet $\Sigma^*$ is the set of all strings over $\Sigma$. A **language** $S$ is a subset of $\Sigma^*$. We write $(\Sigma^*)^{<\omega}$ for the set of all finite subsets of $\Sigma^*$. We call a language **recursively enumerable** if there exists an algorithm (Turing machine) that lists exactly its elements (possibly with repeats), possibly never halting. An **exhaustion** is a sequence of finite sets $\boldsymbol{S} \stackrel{\text{def}}{=} \{S_n\}_{n=0}^{\infty}$ such that $S_0 = \emptyset$ and $S_n \subseteq S_{n+1}$ for all $n$. We define $\Delta S_n \stackrel{\text{def}}{=} S_n \setminus S_{n-1}$. We write $S_n \uparrow S \stackrel{\text{def}}{=} \bigcup_{n=0}^{\infty} S_n$ to denote the limit of an exhaustion. When the limit $S$ is of relevance, we call $\boldsymbol{S}$ an exhaustion *of* $S$. An **enumeration** of a language $S$ is an exhaustion $\boldsymbol{S} = \{S_n\}_{n=0}^{\infty}$ of $S$ with the property that $|\Delta S_n| = 1$ for all $n \in \mathbb{N}$. Given a function $f\colon \mathbb{N} \to \mathbb{N}_0$ with $\lim_{N\to\infty} \sum_{n=1}^{N} f(n) = \infty$, an **$f$-bounded** exhaustion is an exhaustion $\boldsymbol{S} = \{S_n\}_{n=0}^{\infty}$ of $S$ with the property that $|\Delta S_n| \leq f(n)$ for all $n \in \mathbb{N}$. If $f(n) = 1$, we call the exhaustion a **single-step** exhaustion. If $f(n) = c$, for a constant $c$, we call the exhaustion a **$c$-step** exhaustion. Given a language $S$, we write $\mathcal{E}(S)$ to denote the set of all exhaustions of $S$, $\mathcal{E}_e(S)$ to denote the set of all enumerations of $S$, and $\mathcal{E}_f(S)$ to denote the set of all $f$-bounded exhaustions of $S$.

### 2.2 Approximate language learning

In this paper, we are interested in understanding how close one language is to another. However, languages are, in general, *infinite* sets. Consequently, cardinalities do not distinguish between small and large numbers of disagreements. Moreover, since $\Sigma^*$ is infinite, it admits no uniform distribution. Thus, the error rate $\Pr_{\boldsymbol{y}\sim D}[\boldsymbol{y} \in S \setminus S']$, for two languages $S, S' \subseteq \Sigma^*$, is not well defined without choosing a particular distribution $D$ over $\Sigma^*$.

| | Perfect tail precision | | Relaxed tail precision | |
|---|---|---|---|---|
| | **Partial** | **Full** | **Partial** | **Full** |
| **Novelty** | $\widehat{\pi}_*(T, \boldsymbol{G}) = 1$<br>$\widehat{\rho}_*(\boldsymbol{T}, G) \geq \alpha/2$<br>Theorem 3.5 (Kleinberg & Wei, 2025b) | $\widehat{\pi}_*(T, \boldsymbol{G}) = 1$<br>$\widehat{\rho}_*(\boldsymbol{T}, G) \geq 1/2$<br>Theorem 3.5 (Kleinberg & Wei, 2025b) | $\widehat{\pi}_*(T, \boldsymbol{G}) = 1$<br>$\widehat{\rho}_*(\boldsymbol{T}, G) \geq \frac{\max(1-\beta,}{\alpha/3)}$<br>Theorem 4.8 (ours) | $\widehat{\pi}_*(T, \boldsymbol{G}) = 1$<br>$\widehat{\rho}_*(\boldsymbol{T}, G) \geq 1/2$<br>Theorem 3.5 (Kleinberg & Wei, 2025b) |
| **$\gamma$-Novelty** | $\widehat{\pi}_*(T, \boldsymbol{G}) = 1$<br>$\widehat{\rho}_*(\boldsymbol{T}, G) \geq \alpha$<br>Theorem I.3 (ours) | $\widehat{\pi}_*(T, \boldsymbol{G}) = 1$<br>$\widehat{\rho}_*(\boldsymbol{T}, G) = 1$<br>Corollary I.4 (ours) | $\widehat{\pi}_*(T, \boldsymbol{G}) = 1$<br>$\widehat{\rho}_*(\boldsymbol{T}, G) = 1$<br>Theorem 4.9 (ours) | $\widehat{\pi}_*(T, \boldsymbol{G}) = 1$<br>$\widehat{\rho}_*(\boldsymbol{T}, G) = 1$<br>Corollary I.2 (ours) |

Figure 1: Overview of precision ($\widehat{\pi}_*$) and recall ($\widehat{\rho}_*$) bounds across settings defined by three constraints: novelty fraction with $\gamma \in [0, 1)$ vs. novelty, partial vs. full adversarial exhaustion, and perfect tail precision ($\widehat{\tau}_*(T, \boldsymbol{G}) = 1$) vs. relaxed tail precision ($\widehat{\tau}_*(T, \boldsymbol{G})$ can be 0). Existing results are in blue and new results in green. $T$ is the target language and $\boldsymbol{T}$ is an enumeration of $T$. The adversary reveals a language $A \subseteq T$ via an exhaustion $\boldsymbol{A}$. The learner outputs a guess language $G$ via the exhaustion $\boldsymbol{G}$. In the full exhaustion setting, the adversary is restricted to revealing full exhaustions; in the partial setting, it may reveal a partial exhaustion. $\boldsymbol{A}$ is characterized by lower and upper recall bounds $\widehat{\rho}_*(\boldsymbol{T}, A) \geq \alpha$ and $\widehat{\rho}^*(\boldsymbol{T}, A) \leq \beta$.

**A warm-up.** We first consider *finite* languages as a special case. Let $T \subset \Sigma^*$ be a finite target language and let $G \subset \Sigma^*$ be a finite guess language. Under the standard definitions (Manning et al., 2008), the **precision** ($\pi$) of $G$ with respect to $T$ is the fraction of predicted strings that are correct, and the **recall** ($\rho$) is the fraction of ground truth positives (strings in $T$) that are recovered,

$$\pi(T, G) \stackrel{\text{def}}{=} \frac{|T \cap G|}{|G|}, \tag{1}$$

$$\rho(T, G) \stackrel{\text{def}}{=} \frac{|T \cap G|}{|T|}. \tag{2}$$

The ratios used in precision and recall are not directly applicable to infinite sets since $\frac{\infty}{\infty}$ is not well-defined in our setting. Hence, we will make use of exhaustions to evaluate performance on increasing finite sets. Intuitively, an exhaustion of $G$ models a learner's successive predictions, while an exhaustion of $T$ corresponds to the finite subsets of the target language on which we assess the learner. For the rest of the paper, we assume $G$ and $T$ to be infinite.

**Exhaustion-level precision.** Let $\boldsymbol{T} = \{T_n\}_{n=0}^{\infty}$ be an exhaustion of the target language $T$, called a **target exhaustion**, and let $\boldsymbol{G} = \{G_n\}_{n=0}^{\infty}$ be an exhaustion of the guess language $G$, called **guess exhaustion**. Then, define the **upper precision** $\pi^*$ and the **lower precision** $\pi_*$ as

$$\pi^*(\boldsymbol{T}, \boldsymbol{G}) \stackrel{\text{def}}{=} \limsup_{n \to \infty} \frac{|T_n \cap G_n|}{|G_n|}, \tag{3}$$

$$\pi_*(\boldsymbol{T}, \boldsymbol{G}) \stackrel{\text{def}}{=} \liminf_{n \to \infty} \frac{|T_n \cap G_n|}{|G_n|}. \tag{4}$$

When the upper and lower precision coincide, i.e., $\pi_*(\boldsymbol{T}, \boldsymbol{G}) = \pi^*(\boldsymbol{T}, \boldsymbol{G})$, we define the **precision** simply as $\pi(\boldsymbol{T}, \boldsymbol{G}) \stackrel{\text{def}}{=} \lim_{n \to \infty} \frac{|T_n \cap G_n|}{|G_n|}$. We show that precision does not always exist in Example 2.

**Exhaustion-level recall.** Analogously to precision, we define the **upper recall** $\rho^*$ and **lower recall** $\rho_*$ as

$$\rho^*(\boldsymbol{T}, \boldsymbol{G}) \stackrel{\text{def}}{=} \limsup_{n \to \infty} \frac{|T_n \cap G_n|}{|T_n|}, \tag{5}$$

$$\rho_*(\boldsymbol{T}, \boldsymbol{G}) \stackrel{\text{def}}{=} \liminf_{n \to \infty} \frac{|T_n \cap G_n|}{|T_n|}. \tag{6}$$

When these coincide, we define the **recall** as $\rho(\boldsymbol{T}, \boldsymbol{G}) \stackrel{\text{def}}{=} \lim_{n \to \infty} \frac{|T_n \cap G_n|}{|T_n|}$. Just like precision, recall does not always exist (see Example 3).

**Membership-based precision.** The precision notions above compare two *exhaustions*. Validity, however, depends on membership in $T$ rather than membership in a particular finite prefix $T_n$ of some exhaustion. To eliminate this arbitrary dependence on how $T$ is exhausted, we define precision based on membership in the target language by taking the supremum over all exhaustions of $T$. Since the limit need not exist, we use the lower and upper variants:

$$\widehat{\pi}_*(T, \boldsymbol{G}) \stackrel{\text{def}}{=} \sup_{\boldsymbol{T} \in \mathcal{E}(T)} \pi_*(\boldsymbol{T}, \boldsymbol{G}), \qquad (7) \qquad \widehat{\pi}^*(T, \boldsymbol{G}) \stackrel{\text{def}}{=} \sup_{\boldsymbol{T} \in \mathcal{E}(T)} \pi^*(\boldsymbol{T}, \boldsymbol{G}). \qquad (8)$$

We primarily work with $\widehat{\pi}_*(T, \boldsymbol{G})$ as a robust guarantee; all statements have direct analogues for the upper precision. As we show next, this definition coincides with the intuitive membership-based notion: it equals the lower asymptotic fraction of guess strings that lie in $T$.

**Theorem 2.1.** *Let $f(n)\colon \mathbb{N} \to \mathbb{N}$ be a function with $\lim_{N\to\infty} \sum_{n=1}^{N} f(n) = \infty$. Let $T \subseteq \Sigma^*$ be the target language and let $\boldsymbol{G} = \{G_n\}_{n=0}^{\infty}$ be an $f$-bounded exhaustion of the guess $G$. Then,*

$$\sup_{\boldsymbol{T} \in \mathcal{E}_f(T)} \pi_*(\boldsymbol{T}, \boldsymbol{G}) = \liminf_{n\to\infty} \frac{|T \cap G_n|}{|G_n|}. \qquad (9)$$

The proof is given in §B.2.

**Coverage-based recall.** For recall, we are interested in understanding how many strings in the target language $T$ the learner actually recovers. Hence, we check for each string in $T$ if it is included in $G$. This is achieved by taking the supremum over $G$. We define

$$\widehat{\rho}_*(\boldsymbol{T}, G) = \sup_{\boldsymbol{G} \in \mathcal{E}(G)} \rho_*(\boldsymbol{T}, \boldsymbol{G}), \qquad (10) \qquad \widehat{\rho}^*(\boldsymbol{T}, G) = \sup_{\boldsymbol{G} \in \mathcal{E}(G)} \rho^*(\boldsymbol{T}, \boldsymbol{G}). \qquad (11)$$

This definition assumes that we are given a fixed target exhaustion $\boldsymbol{T}$. Again, we are mostly concerned with the lower recall as a robust guarantee; the following lemma holds analogously for upper recall.

**Lemma 2.2.** *Let $T \subseteq \Sigma^*$ be the target language with an exhaustion $\boldsymbol{T} = \{T_n\}_{n=0}^{\infty}$ and let $G$ be the guess language. Then,*

$$\sup_{\boldsymbol{G} \in \mathcal{E}(G)} \rho_*(\boldsymbol{T}, \boldsymbol{G}) = \liminf_{n\to\infty} \frac{|T_n \cap G|}{|T_n|}. \qquad (12)$$

The proof is given in §C.2.

**Tail precision.** The precision notions above are global: they measure the fraction of correct strings among *all* strings in $G_n$. Now we define **tail precision**, which considers only the increment $\Delta G_n$.

Let $\boldsymbol{G} = \{G_n\}_{n=0}^{\infty}$ be an exhaustion. For each $n$, define the step-wise tail precision $t_n$ as

$$t_n \stackrel{\text{def}}{=} \begin{cases} 1, & \text{if } |\Delta G_n| = 0, \\ \frac{|T \cap \Delta G_n|}{|\Delta G_n|}, & \text{otherwise.} \end{cases} \qquad (13)$$

**Definition 2.3.** *Given a target language $T$ and a guess language $G$ with exhaustion $\boldsymbol{G}$, we define the **lower tail precision** and **upper tail precision** as*

$$\widehat{\tau}_*(T, \boldsymbol{G}) \stackrel{\text{def}}{=} \liminf_{n\to\infty} t_n, \qquad (14) \qquad \widehat{\tau}^*(T, \boldsymbol{G}) \stackrel{\text{def}}{=} \limsup_{n\to\infty} t_n. \qquad (15)$$

Tail precision measures the asymptotic fraction of newly added strings that are in $T$.

**Proposition 2.4.** *Let $T$ be the target language and $\boldsymbol{G}$ a single-step exhaustion of $G$. Then, $\widehat{\tau}_*(T, \boldsymbol{G})$ is either $0$ or $1$. Moreover, for any $c$-step exhaustion $\boldsymbol{G}$, $\widehat{\tau}_*(T, \boldsymbol{G})$ is attained after finitely many steps.*[2]

We give a proof in §D. For a function $f$ with non-constant growth, one can distinguish between finite-time stabilization to some constant and asymptotic convergence; the following example illustrates this.

[2]This means that there exists a timestep $n < \infty$, such that the step-wise tail precision $t_n = \widehat{\tau}_*(T, \boldsymbol{G})$ and $t_N \geq t_n$ for $N \geq n$.

**Example 1.** *Let $|\Delta G_n| \leq n$, i.e., at step $n$ the exhaustion $\boldsymbol{G}$ may grow by up to $n$ new strings. Suppose that at every step the increment contains exactly $n-1$ strings from $T$ and one string not in $T$. Then,*

$$\widehat{\tau}_*(T, \boldsymbol{G}) = \liminf_{n\to\infty} \frac{|T \cap \Delta G_n|}{|\Delta G_n|} = \liminf_{n\to\infty} \frac{n-1}{n} = 1, \tag{16}$$

*i.e., the lower tail precision converges asymptotically to 1, even though an error occurs at every step. However, this ratio is never exactly 1 at any finite step. Finite-time stabilization to 1 would require that, from some step onward, every newly generated string lies in $T$, i.e., $\Delta G_n \subseteq T$ for all sufficiently large $n$.*

**Relation to Kleinberg & Wei (2025a).** Our notion of coverage-based recall is nearly identical to the density measure of Kleinberg & Wei (2025a). In their setup, density is evaluated along a fixed enumeration $\boldsymbol{T} = \{T_n\}_{n=0}^{\infty}$. In Lemma 2.2, we derive Kleinberg & Wei's (2025a) density measure of one language $G$ in another $T$, corresponding to our definitions of lower (Eq. (10)) and upper (Eq. (11)) coverage-based recall. We extend their framework in two ways: (i) we allow *any* exhaustion rather than committing to a single fixed enumeration, and (ii) we introduce a complementary *precision* notion by exhausting the guessed language instead of the target. In the next section, we instantiate these notions in the generation-in-the-limit framework of Kleinberg & Mullainathan (2024). In that setting, a generator's output stream induces increasing prefix sets $G_n$ of generated strings, and hence naturally defines an exhaustion of $G$. We will also formalize the eventual-validity requirement from prior work by showing in Proposition 3.3 that it corresponds to *finite-time stabilization* of tail precision to 1.

## 3 Generation in the limit

We now introduce Kleinberg & Mullainathan's (2024) generation in the limit framework. Let $\mathcal{T}$ be a countable collection of languages $\mathcal{T} \subseteq \mathcal{P}(\Sigma^*)$. We assume that each language $S \in \mathcal{T}$ is infinite.[3]

An adversary chooses a target language $T \in \mathcal{T}$ and produces an $f$-bounded exhaustion $\boldsymbol{A} = \{A_n\}_{n=0}^{\infty}$ such that $A_n \subseteq T$ for all $n$, and $A_n \uparrow A$ with $A \subseteq T$. Kleinberg & Mullainathan (2024) assume a single-step exhaustion; we extend their framework to $f$-bounded exhaustions. The adversary can repeat strings, and is not required to reveal a new string at each timestep; therefore, $|\Delta A_n|$ can be 0. We call $\boldsymbol{A}$ a **full** exhaustion of $T$ if $A = T$, and a **partial** exhaustion otherwise ($A \subsetneq T$). In line with Kleinberg & Wei (2025b), we assume $A$ to be infinite. A **generator** is a function $\mathcal{G}\colon (\Sigma^*)^{<\omega} \times (\Sigma^*)^{<\omega} \to (\Sigma^*)^{<\omega}$. Given an adversarial exhaustion $\boldsymbol{A} = \{A_n\}_{n=0}^{\infty}$, the generator induces an $f$-bounded **generated exhaustion** $\boldsymbol{G} = \{G_n\}_{n=0}^{\infty}$ via

$$G_0 \stackrel{\text{def}}{=} \emptyset, \quad G_n \stackrel{\text{def}}{=} G_{n-1} \cup \mathcal{G}(A_n, G_{n-1}) \quad (n \in \mathbb{N}). \tag{17}$$

At time $n$ the generator observes the currently revealed set $A_n$ and the previously generated set $G_{n-1}$, and appends one or multiple new strings. Kleinberg & Mullainathan (2024) demand that one new string is generated that is not in $A_n$. Furthermore, their proposed algorithms also do not repeat strings in $G_{n-1}$. We write $G_n \uparrow G$ for the guess language.

**Definition 3.1** (Generation in the limit)**.** *A generator $\mathcal{G}$ **generates in the limit** with precision $\lambda_p$, recall $\lambda_r$ and tail precision $\lambda_t$ for a collection of languages $\mathcal{T}$ if for every $T \in \mathcal{T}$, every adversarial exhaustion $\boldsymbol{A} = \{A_n\}_{n=0}^{\infty}$ of $A \subseteq T$, and $\boldsymbol{G}$ defined as in (17), it holds that*

$$\widehat{\pi}_*(T, \boldsymbol{G}) \geq \lambda_p, \tag{18a}$$
$$\widehat{\rho}_*(\boldsymbol{T}, G) \geq \lambda_r, \tag{18b}$$
$$\widehat{\tau}_*(T, \boldsymbol{G}) \geq \lambda_t. \tag{18c}$$

### 3.1 Perfect tail precision

Kleinberg & Mullainathan (2024) require that after some finite timestep, all generated strings are valid (i.e., belong to $T$). We refer to this property as *eventual validity*.

**Definition 3.2** (Eventual Validity)**.** *A generator $\mathcal{G}$ is **eventually valid** if there exists an $n^\star$ such that for all $n \geq n^\star$,*

$$\mathcal{G}(A_n, G_{n-1}) \subseteq T. \tag{19}$$

[3] This is in line with Kleinberg & Mullainathan's work and makes the generation of infinitely many unseen strings possible.

Equivalently, $|G \setminus T| < \infty$. We show that eventual validity corresponds to finite-time stabilization of tail precision to 1.

**Proposition 3.3.** *Let $\mathcal{G}$ be a generator and let $\boldsymbol{G}$ be the exhaustion it produces against the adversarial exhaustion $\boldsymbol{A}$ of $A \subseteq T$. Then $\mathcal{G}$ is eventually valid (Def. 3.2) if and only if $\widehat{\tau}_*(T, \boldsymbol{G}) = 1$ is attained in finite time.*

*Proof sketch.* Intuitively, if the step-wise tail precision is 1 from some finite time onward, then every newly generated string after that time lies in the target language $T$. Hence, only finitely many mistakes can occur, and the generator is eventually valid. Conversely, if only finitely many mistakes occur, then after the last mistake every newly generated string is valid, so the step-wise tail precision is 1 from that point onward. ■

The full proof is in §D. In general, tail precision 1 automatically implies precision 1 (see Lemma D.1). On the other hand, one may have precision 1 without convergence of tail precision to 1: for example, if the generator outputs exactly one invalid string at times $n = 2^m$ for $m \in \mathbb{N}$ and a valid string otherwise, then the number of mistakes up to time $n$ is $\Theta(\log n)$, so precision tends to 1 even though mistakes occur infinitely often, which yields lower tail precision 0.

### 3.2 Generation with novelty

**Definition 3.4** (Novelty constraint)**.** *A generator $\mathcal{G}$ is **novel** if for every $n \geq 1$,*

$$\mathcal{G}(A_n, G_{n-1}) \cap (A_n \cup G_{n-1}) = \emptyset. \tag{20}$$

In words, once a string has been seen—meaning that it has been either revealed in some $A_n$ or it has been generated by some $G_n$—it becomes permanently unavailable to the generator. This modeling choice makes achieving high recall non-trivial. Without it, a generator could simply copy the adversary and achieve perfect recall and perfect precision whenever the adversary fully enumerates $T$ (see Theorem E.1). Under a novel generator, however, the adversary can *steal* strings by revealing them first, which imposes an inherent ceiling on recall. We restate, in our notation, the following recall bounds from Kleinberg & Wei (2025b).

**Theorem 3.5.** *[Kleinberg & Wei 2025b, Thm. 3.5] Let $\mathcal{T}$ be a countable collection of languages and $T \in \mathcal{T}$ a target language. Fix an enumeration $\boldsymbol{T} = \{T_n\}_{n=0}^{\infty}$ of $T$. There exists a novel generator $\mathcal{G}$ that generates in the limit for $T$ with $\widehat{\tau}_*(T, \boldsymbol{G}) = 1$, such that the following holds: For every single-step adversarial exhaustion $\boldsymbol{A} = \{A_n\}_{n=0}^{\infty}$ with $A_n \uparrow A \subseteq T$, the generator $\mathcal{G}$ induces a single-step exhaustion $\boldsymbol{G} = \{G_n\}_{n=0}^{\infty}$ with $G_n \uparrow G$ satisfying*

$$\widehat{\rho}_*(\boldsymbol{T}, G) \geq \frac{1}{2}\, \widehat{\rho}_*(\boldsymbol{T}, A). \tag{21}$$

By Proposition 2.4, $\widehat{\tau}_*(T, \boldsymbol{G})$ converges in finite time to 1 in this setting. As a special case, Theorem 3.5 implies that for a full, single-step exhaustion of the target language, an algorithm can generate with recall $\frac{1}{2}$.

**Theorem 3.6.** *[Kleinberg & Wei 2025b, Thm. 3.3] The constant $\frac{1}{2}$ in Theorem 3.5 is optimal: for every novel generator $\mathcal{G}$ there exist a countable collection $\mathcal{T}$, a target language $T \in \mathcal{T}$ with enumeration $\boldsymbol{T}$, and a single-step adversarial exhaustion $\boldsymbol{A} = \{A_n\}_{n=0}^{\infty}$ with $A_n \uparrow A \subseteq T$ such that, if $\mathcal{G}$ induces a single-step exhaustion $\boldsymbol{G}$, then*

$$\widehat{\rho}_*(\boldsymbol{T}, G) \leq \frac{1}{2}\, \widehat{\rho}_*(\boldsymbol{T}, A). \tag{22}$$

In practice, forbidding the generator from outputting previously revealed strings is not always desirable: repeating training data can be necessary (e.g., when quoting a source). Motivated by this observation, we introduce the following continuous relaxation of the novelty constraint.

**Definition 3.7** ($\gamma$-novelty)**.** *Let $\gamma \in [0, 1]$. Given an adversarial exhaustion $\boldsymbol{A} = \{A_n\}_{n=0}^{\infty}$ and the generated exhaustion $\boldsymbol{G} = \{G_n\}_{n=0}^{\infty}$, define*

$$N_0 := \emptyset, \qquad N_n := N_{n-1} \cup \big(\Delta G_n \setminus A_n\big). \tag{23}$$

*We say that the generator is $\gamma$-novel if, for every $n$ with $|G_n| > 0$,*

$$\frac{|N_n|}{|G_n|} \geq \gamma. \tag{24}$$

In words, $\gamma$-novelty requires that every $G_n$ contains at least a $\gamma$-fraction of strings that were novel at the time they were first generated, i.e., strings that had not already been revealed by the adversary and had not already been generated. Equivalently, at most a $(1-\gamma)$-fraction of the strings in any $G_n$ may be non-novel. We define $\gamma$-novelty for $\gamma \in [0,1]$. For $\gamma = 0$, this imposes no novelty restriction, while the endpoint $\gamma = 1$ coincides with the strict novelty constraint of Definition 3.4. We treat novelty ($\gamma = 1$) separately from the relaxed regime $\gamma < 1$ (see Figure 1).

We study $\gamma$-novel generation in the limit under varying tail-precision assumptions in §4. We show that the relaxation $\gamma < 1$ is already sufficient to recover all adversarially revealed strings, and, when perfect tail precision is not required, to achieve both precision and recall equal to 1.

## 4 Generating with infinitely many hallucinations

Existing results aim to maximize recall for novel generators with perfect tail precision, yielding guarantees such as the $\frac{1}{2}$ bound of Kleinberg & Wei (2025b) (restated in Theorems 3.5 and 3.6). In this section, we ask how these limits change once we relax the tail precision constraint: rather than requiring that only finitely many hallucinations occur, we allow infinitely many hallucinations provided they occur at a vanishing rate. In this regime, tail precision can even be 0, but precision is 1 in the limit.

In line with Kleinberg & Wei (2025b), we assume a fixed enumeration $\boldsymbol{T} = \{T_n\}_{n=0}^{\infty}$ and single-step exhaustions $\boldsymbol{A} = \{A_n\}_{n=0}^{\infty}$ with $A_n \uparrow A \subseteq T$ and $\boldsymbol{G} = \{G_n\}_{n=0}^{\infty}$ with $G_n \uparrow G$, i.e., both the adversary and the generator are allowed to produce at most one string per round. In contrast, an enumeration adds exactly one new string at every round. Our results for generation without novelty ($\gamma = 0$) hold for a more general setting where both $\boldsymbol{A}$ and $\boldsymbol{G}$ are $f$-bounded. We characterize the adversary's exhaustion by the lower and upper recall:

$$\widehat{\rho}_*(\boldsymbol{T}, A) \geq \alpha, \qquad \widehat{\rho}^*(\boldsymbol{T}, A) \leq \beta. \tag{25}$$

Intuitively, the lower recall measures how much of the target language the adversary eventually reveals, while the upper recall captures how large the revealed set can appear along subsequences. We fix an enumeration of $\Sigma^*$ and let $\prec$ denote the induced order. Throughout this section, we assume that the fixed target enumeration $\boldsymbol{T}$ is induced by this order, i.e., $T_n$ consists of the $n$ smallest elements of $T$ under $\prec$.

We first analyze this setting without and with the novelty constraint ($\gamma = 0$ vs. $\gamma = 1$), and then give a $\gamma$-novel algorithm and guarantees for $\gamma \in [0,1)$. We propose an algorithmic template that most of our constructions share: We run a **safe** generator (outputting from $T$) most of the time, and interleave it with rare **exploration** rounds that enumerate $\Sigma^*$ to recover strings the adversary never reveals. Exploration rounds may output invalid strings, but we schedule them sparsely, yielding precision 1. To achieve this, we will make use of **exploration sets**.

**Definition 4.1** (Exploration set)**.** *An **exploration set** is an infinite set $R \subset \mathbb{N}$ such that $1 \notin R$ and no two consecutive integers belong to $R$. For $m \in \mathbb{N}$, define $r(m) \stackrel{\text{def}}{=} \big|R \cap \{1, \ldots, m\}\big|$. We require that $r(m)/m \to 0$ as $m \to \infty$.*

An example is $R = \{2^i \colon i \in \mathbb{N}\}$.

For the $\gamma$-novel algorithm, we need exploration rounds to be sparse not only asymptotically, but in every finite prefix. We therefore introduce the following strengthened notion.

**Definition 4.2** ($\gamma$-admissible exploration set)**.** *Let $\gamma \in [0,1)$. An infinite exploration set $R \subseteq \mathbb{N}$ is **$\gamma$-admissible** if, for every $m \in \mathbb{N}$, $r(m) \leq (1-\gamma)m$.*

We show the existence of such sets for every $\gamma < 1$ in Lemma I.1.

### 4.1 Generation without novelty ($\gamma = 0$)

We first consider $\gamma = 0$, dropping the novelty constraint entirely, which serves as a useful baseline. In this regime, repeating the adversary's strings is allowed at any time, and the generator can combine (i) parroting the adversary to keep precision high and (ii) sparse exploration to achieve perfect recall, even for partial exhaustions $\boldsymbol{A}$.

**Theorem 4.3.** *Let $\mathcal{T}$ be a countable collection of languages and $T \in \mathcal{T}$ a target language. Fix an enumeration $\boldsymbol{T} = \{T_n\}_{n=0}^{\infty}$ of $T$. There exists a generator $\mathcal{G}$, such that, for any $f$-bounded adversarial exhaustion $\boldsymbol{A} = \{A_n\}_{n=0}^{\infty}$ with $A_n \uparrow A \subseteq T$, the generated exhaustion $\boldsymbol{G}$ is $f$-bounded and satisfies:*

$$\widehat{\rho}_*(\boldsymbol{T}, G) = 1, \qquad \widehat{\pi}_*(T, \boldsymbol{G}) = 1. \tag{26}$$

The proof is in §E.2. Note that this theorem holds for both full and partial exhaustion of the target $T$.

### 4.2 Generation with novelty ($\gamma = 1$)

We now reinstate novelty (Definition 3.4), considering the other extreme of the spectrum, where it is not allowed to repeat any string that has been revealed before. Hence, the adversary can *steal* strings by revealing them first, making perfect recall unattainable. Our goal is to understand how far recall can be pushed when we drop perfect tail precision, but aim to generate with precision 1. Our generator combines two complementary mechanisms:

1. **Exploration rounds** recover the non-stolen portion. If a string is never revealed by the adversary (i.e., it lies in $T \setminus A$), then it is never blocked by novelty. Hence, an exploration routine that enumerates $\Sigma^*$ while skipping already-seen strings will eventually output it.
2. **Safe rounds** recover recall proportional to $\alpha$, when $\widehat{\rho}_*(\boldsymbol{T}, A) \geq \alpha$. We run a Kleinberg & Wei (2025b)-style generator whose tail precision converges to 1, which will be the key ingredient for achieving precision 1.

In this setting, there can be up to two adversarial revelations between two safe outputs. Therefore, we first need to adapt Kleinberg & Wei's (2025b) algorithm to a so-called $k$-batched adversarial setting (with $k = 2$ here).

**Definition 4.4.** *Consider a setting in which the adversary is $k$-step bounded, i.e., it can reveal at most $k$ new strings per step $n$, and the generator is single-step bounded. We call this setting a $k$-batched adversarial setting.*

Algorithm 2 shows Kleinberg & Wei's (2025b) algorithm for generation in the limit.[4] At time $n$, consider all hypotheses that are consistent with the strings revealed by the adversary, i.e., all languages $S \in \mathcal{T}$ with $A_n \subseteq S$. Intersecting these consistent hypotheses in a global order, given by a fixed listing $(S'_i)_{i \geq 1}$ of all languages in $\mathcal{T}$, yields a decreasing *intersection chain* $C_n$ (see Algorithm 3). Early sets in the chain are larger (better for recall), but riskier, while later sets are smaller (safer), but might miss many valid strings. The *identified intersection* procedure (see Algorithm 4) selects a particular member $\mathcal{I}^{(n)} \in C_n$ that behaves stably over time: after some finite time, it is a subset of the true target $T$ (eventual validity), it is comparable across rounds, and it is *full* (contains $A$) infinitely often. Intuitively, $\mathcal{I}^{(n)}$ is designed to be a tractable candidate set of strings deemed safe given $A_n$. However, outputting only from $\mathcal{I}^{(n)}$ can hurt recall, because $\mathcal{I}^{(n)}$ may shrink temporarily when the adversary reveals new strings, making the generator too conservative in those rounds. To mitigate this, the algorithm chooses an *aggressive set* $\widetilde{\mathcal{I}}^{(n)}$, which is either the current identified intersection or a carefully chosen larger fallback from recent rounds (see Algorithm 5). After some finite time—once the identified intersections are valid and comparable—this fallback simplifies to $\widetilde{\mathcal{I}}^{(n)} \in \{\mathcal{I}^{(n)}, \mathcal{I}^{(n-1)}\} \subseteq T$. Thus $\widetilde{\mathcal{I}}^{(n)}$ is also eventually valid, but can be temporarily larger than $\mathcal{I}^{(n)}$, allowing the generator to improve recall. However, $\widetilde{\mathcal{I}}^{(n)}$ itself evolves, so a string that appears safe at round $n$ might no longer be among preferred candidates at round $n+1$ if $\widetilde{\mathcal{I}}^{(n)}$ shrinks. This problem is addressed by **pods**. In each round $n$, the algorithm creates a pod $P_n$ of size $s_n$ with the $s_n$ $\prec$-smallest *unused* strings of $\widetilde{\mathcal{I}}^{(n)}$ (excluding strings already revealed, output, or in previous pods) and adds those strings into a global *pod pool*. In each round, the generator outputs the $\prec$-smallest element from the pool. This pool is crucial for achieving high recall: Even if $\widetilde{\mathcal{I}}^{(n)}$ shrinks later, strings that were previously added to pods are not lost and can eventually be output.

We first generalize Kleinberg & Wei's (2025b) pods algorithm (see Algorithm 2) and the corresponding recall bound for a $k$-batched setting, assuming that $\boldsymbol{A}$ is a $k$-step exhaustion.

[4] Note that they require $\boldsymbol{A}$ and $\boldsymbol{G}$ to be single-step exhaustions.

**Lemma 4.5.** *Assume a $k$-batched adversarial setting, with $k \geq 2$. For any $T \in \mathcal{T}$ and any $k$-batched exhaustion $\boldsymbol{A}$ of $A \subseteq T$ with lower recall $\widehat{\rho}_*(\boldsymbol{T}, A)$, the exhaustion $\boldsymbol{G}$ generated by applying Algorithm 2 satisfies the novelty constraint and achieves*

$$\widehat{\rho}_*(\boldsymbol{T}, G) \geq \frac{1}{k+1}\widehat{\rho}_*(\boldsymbol{T}, A), \tag{27a}$$

$$\widehat{\pi}_*(T, \boldsymbol{G}) = 1, \qquad \widehat{\tau}_*(T, \boldsymbol{G}) = 1. \tag{27b}$$

*Proof sketch.* First, note that Algorithm 2 outputs only from the current pod pool $\mathcal{P}_n$, which is populated with $\prec$-small unused elements from aggressive sets. Thus, the generator is novel. Denote by $\mathcal{P}$ the set of all strings that are ever placed in a pod. We relate the adversary's revealed strings to the generator's outputs via a two-step charging argument. First, consider strings in $A$ that are neither output nor ever placed into a pod (not in $\mathcal{P}$). Using that (after finite steps) the identified intersection is valid and is full infinitely often, one can show: for each such missed string, earlier pods contain many $\prec$-smaller unused candidates. Formally, we can charge each missed $\boldsymbol{x}$ to a pod index $\rho(\boldsymbol{x})$ whose pod contains at least $s$ elements all $\prec \boldsymbol{x}$, and because the adversary introduces at most $k$ new strings per round, each pod index receives at most $k$ charges. Hence the total number of such missed strings up to $T_n$ is at most $\frac{k}{s}|\mathcal{P} \cap T_n| + \mathcal{O}(n)$. Second, we control the backlog in the pod pool. Since the algorithm outputs one element per round and the adversary reveals at most $k$ new strings per round, the number of adversary strings that can accumulate in the pool without being output is at most a factor $k$ larger than the number of outputs. We can charge every pooled string that appears in $A$ but not in $G$ to a nearby generator output. Each output receives at most $k$ such charges. Combining this with the first charging step yields $|A \cap T_n| \leq (k+1+\mathcal{O}(1))|G \cap T_n|$. Dividing by $|T_n|$, taking $\liminf$, and letting the pod-size parameter $s \to \infty$ gives $\widehat{\rho}_*(\boldsymbol{T}, G) \geq \frac{1}{k+1}\widehat{\rho}_*(\boldsymbol{T}, A)$. The tail precision and precision are 1, as Kleinberg & Wei's (2025b) algorithm guarantees that from a finite step onward every $\Delta G_n$ is in $T$. ■

Full proof in §G. The following lemma provides a matching upper bound on the lower recall of any novel generator, generalizing Kleinberg & Wei (2025b, Theorem 3.3) to the $k$-batched setting. Together with Lemma 4.5, this shows that the factor $\frac{1}{k+1}$ is worst-case optimal.

**Lemma 4.6.** *Assume a $k$-batched adversarial setting. There exist countable collections of languages $\mathcal{T}$, such that for a target $T$ and an adversarial exhaustion $\boldsymbol{A}$ with $A_n \uparrow A$, and $A \subseteq T$, the recall of any novel generator that satisfies $\widehat{\tau}_*(T, \boldsymbol{G}) = 1$ is bounded by*

$$\widehat{\rho}_*(\boldsymbol{T}, G) \leq \frac{\widehat{\rho}_*(\boldsymbol{T}, A)}{k+1}. \tag{28}$$

The full proof is given in §G.2. We now move to examining the contribution of exploration rounds: if a generator eventually outputs every string in the target language that the adversary never reveals, then recall is at least $1 - \beta$.

**Lemma 4.7.** *Let $T \in \mathcal{T}$ be the target, and $\boldsymbol{A}$ an exhaustion produced by the adversary with $\widehat{\rho}^*(\boldsymbol{T}, A)$. Assume that $\boldsymbol{G}$ contains every element in $T \setminus A$ at some point. Then*

$$\widehat{\rho}_*(\boldsymbol{T}, G) \geq 1 - \widehat{\rho}^*(\boldsymbol{T}, A). \tag{29}$$

*Proof.* As $T \setminus A \subseteq G$, we get

$$\frac{|G \cap T_n|}{|T_n|} \geq \frac{|T \setminus A \cap T_n|}{|T_n|} \geq 1 - \frac{|A \cap T_n|}{|T_n|}. \tag{30}$$

Taking $\liminf_{n\to\infty}$ on the left and using $\liminf_{n\to\infty}(1 - a_n) = 1 - \limsup_{n\to\infty}(a_n)$ gives

$$\widehat{\rho}_*(\boldsymbol{T}, G) \geq 1 - \limsup_{n\to\infty} \frac{|A \cap T_n|}{|T_n|} \tag{31a}$$

$$= 1 - \widehat{\rho}^*(\boldsymbol{T}, A). \tag{31b}$$

■

We now present Algorithm 6, which alternates safe rounds with exploration rounds at times $m \in R$. On safe rounds, we execute one step of the batched Kleinberg & Wei (2025b) routine: update the consistent chain, add a new pod from the aggressive set to the global pod pool, and output the $\prec$-smallest unused pool element. On exploration rounds, we output the next string in the fixed enumeration $\boldsymbol{E}$ of $\Sigma^*$, skipping previously seen strings to enforce novelty. Since exploration does not advance the safe routine, the safe routine perceives a 2-batched adversary.

**Theorem 4.8.** *For any $T \in \mathcal{T}$ and any single-step adversarial exhaustion $\boldsymbol{A}$ with lower recall $\widehat{\rho}_*(\boldsymbol{T}, A)$ and upper recall $\widehat{\rho}^*(\boldsymbol{T}, A)$, Algorithm 6 generating $\boldsymbol{G} = \{G_n\}_{n=0}^{\infty}$ with $G_n \uparrow G$ achieves precision 1 and recall*

$$\widehat{\rho}_*(\boldsymbol{T}, G) \geq \max\left(1 - \widehat{\rho}^*(\boldsymbol{T}, A), \frac{\widehat{\rho}_*(\boldsymbol{T}, A)}{3}\right). \tag{32}$$

*Proof sketch.* By construction, Algorithm 6 eventually outputs every string in $T \setminus A$, which yields recall at least $1 - \beta$ (cf. Lemma 4.7). Moreover, the safe routine is invoked in a 2-batched manner, implying an additional recall guarantee of at least $\alpha/3$ from the safe rounds (Lemma 4.5). These two contributions cannot, in general, be added: it may occur that both the safe and the exploration rounds output only strings from $T \setminus A$. Precision 1 follows because the safe routine outputs only strings from $T$ after some finite time, and the exploration rounds are scheduled at a vanishing rate. ■

Full proof in §H.1. The term $1 - \beta$ dominates when the adversary reveals only a small portion of the target along some subsequence (i.e., $\beta$ is small), since exploration can then recover almost all strings in $T$ that are never stolen. Overall, Algorithm 6 can improve recall, particularly in regimes where $\alpha$ and $\beta$ are small. Finally, in the setting where the adversary presents a full exhaustion of $T$, the original Kleinberg & Wei (2025b) bound cannot be improved.

### 4.3 Generation with $\gamma$-novelty for $\gamma \in [0, 1)$

With Algorithm 7, we provide an adaptation of Algorithm 6 that satisfies $\gamma$-novelty for $\gamma \in [0, 1)$ and leads to higher recall guarantees. Concretely, Algorithm 7 differs from Algorithm 6 in two ways: (i) the exploration set $R$ is required to be $\gamma$-admissible, and (ii) on exploration rounds, the algorithm skips only strings that have already been generated, rather than also skipping strings revealed by the adversary. Thus, an exploration output may be non-novel if it has previously been revealed by the adversary. Since $R$ is $\gamma$-admissible, the number of such non-novel outputs remains within the allowed quota. We show that for every $\gamma \in [0, 1)$, precision and recall equal to 1 can be achieved.

**Theorem 4.9.** *Fix any $\gamma \in [0, 1)$. For any $T \in \mathcal{T}$ and any single-step adversarial exhaustion $\boldsymbol{A}$ with $A_n \uparrow A \subseteq T$, Algorithm 7 generates a $\gamma$-novel exhaustion $\boldsymbol{G} = \{G_n\}_{n=0}^{\infty}$ with $G_n \uparrow G$ such that*

$$\widehat{\rho}_*(\boldsymbol{T}, G) = 1, \qquad \widehat{\pi}_*(T, \boldsymbol{G}) = 1. \tag{33}$$

*Proof sketch.* The modified exploration rounds enumerate all of $\Sigma^*$, skipping only strings that have already been generated. Hence, every string in $T$ is eventually generated, which yields recall 1. Precision is 1 because safe rounds are eventually valid, while exploration rounds occur at a vanishing rate. Finally, non-novel outputs can occur only in exploration rounds. Since the exploration set is $\gamma$-admissible, the fraction of novel outputs in $G_n$ for each $n$ is at least $\gamma$ and hence the generated exhaustion is $\gamma$-novel for every fixed $\gamma < 1$. The full proof is given in §I.1. ■

## 5 Conclusion

We recast generation in the limit as a precision–recall problem. We introduce a precision metric based on asymptotic error rates, which remains informative even under infinitely many errors, and formalize eventual validity as the *finite-time* stabilization of tail precision to 1. We study relaxations of novelty and perfect tail precision, moving toward settings that better reflect large language model generation. To this end, we introduce an algorithmic template that interleaves safe rounds with sparse exploration rounds. We show that allowing hallucinations at a vanishing rate improves recall over eventually valid generators, but only when the adversary permanently withholds a large portion of the target language. In this regime, exploration rounds recover hidden strings while keeping the error rate controlled. Finally, we introduce a continuous novelty parameter and show that any fixed allowance of non-novel strings enables perfect precision and recall when perfect tail precision is relaxed.

## Acknowledgements

The authors thank Ron Jost and the anonymous reviewers for their helpful feedback and discussions. Alexandra Butoi is supported by the SNSF Grant No. 10.001.775. The authors declare no conflicts of interest.

## References

Angluin, D. Inductive inference of formal languages from positive data. *Information and Control*, 45 (2):117–135, 1980. URL https://doi.org/10.1016/S0019-9958(80)90285-5.

Charikar, M. and Pabbaraju, C. Exploring facets of language generation in the limit. In *Proceedings of Thirty Eighth Conference on Learning Theory*, volume 291 of *Proceedings of Machine Learning Research*, pp. 854–887. PMLR, 2025. URL https://proceedings.mlr.press/v291/charikar25a.html.

Gold, E. M. Language identification in the limit. *Information and Control*, 10(5):447–474, 1967. URL https://www.sciencedirect.com/science/article/pii/S0019995867911655.

Kalavasis, A., Mehrotra, A., and Velegkas, G. On characterizations for language generation: Interplay of hallucinations, breadth, and stability, 2025. URL http://arxiv.org/abs/2412.18530.

Kleinberg, J. and Mullainathan, S. Language generation in the limit. In *Advances in Neural Information Processing Systems*, volume 37, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/7988e9b3876ad689e921ce05d711442f-Paper-Conference.pdf.

Kleinberg, J. and Wei, F. Density measures for language generation. In *2025 IEEE 66th Annual Symposium on Foundations of Computer Science (FOCS)*, pp. 620–658, 2025a. URL https://ieeexplore.ieee.org/document/11369227.

Kleinberg, J. and Wei, F. Language generation and identification from partial enumeration: Tight density bounds and topological characterizations, 2025b. URL https://arxiv.org/abs/2511.05295.

Manning, C. D., Raghavan, P., and Schütze, H. *Introduction to Information Retrieval*. Cambridge University Press, 2008. URL https://www.cambridge.org/highereducation/books/introduction-to-information-retrieval/669D108D20F556C5C30957D63B5AB65C.

Radó, T. On non-computable functions. *Bell System Technical Journal*, 41(3):877–884, 1962. URL https://doi.org/10.1002/j.1538-7305.1962.tb00480.x.

## A Computability of exhaustions

We call an exhaustion $\boldsymbol{S} \stackrel{\text{def}}{=} \{S_n\}_{n=0}^{\infty}$ computable if there exists an algorithm (Turing machine) that emits $S_n$ in order. Note that an exhaustion may fail to be computable even if its limit language is. Let $S$ be an infinite computable language, and write $\boldsymbol{y}_1 \prec \boldsymbol{y}_2 \prec \cdots$ for the elements of $S$ in lexicographic order. Define an exhaustion $\boldsymbol{S} \stackrel{\text{def}}{=} \{S_n\}_{n=0}^{\infty}$ by $S_n \stackrel{\text{def}}{=} \{\boldsymbol{y}_1, \ldots, \boldsymbol{y}_{\mathrm{BB}(n)}\}$, where $\mathrm{BB}(n)$ denotes the busy beaver function. If $\boldsymbol{S}$ were computable, we could run a Turing machine emitting $S_n$ in order until it outputs the $n^{\text{th}}$ set. Returning its cardinality $|S_n|$ would compute the busy beaver function, contradicting the fact that it is not computable (Radó, 1962). However, if $f$ and the limit language $S$ are computable, one can construct a computable $f$-bounded exhaustion of $S$, provided that $f$ allows for enough cumulative capacity.

We denote the set of computable exhaustions of $S$ by $\mathcal{E}_{\mathrm{C}}(S)$. Computable exhaustions imply that the language is recursively enumerable, as shown in the following lemma.

**Lemma A.1.** *Let $S \subseteq \Sigma^*$ and let $\boldsymbol{S} = \{S_n\}_{n=0}^{\infty}$ be an exhaustion of $S$. If $\boldsymbol{S}$ is computable, then $S$ is recursively enumerable.*

*Proof.* We will construct a Turing Machine that enumerates $S$. Since $\boldsymbol{S}$ is computable, there exists a Turing machine $M$ that emits (in order) a finite description of $S_1, S_2, \ldots$. This can be each $S_n$ output as a finite list of strings with separators and an end marker.

We construct an enumerator $E$ for $S$ as follows. Run $M$. Whenever $M$ finishes emitting the description of $S_n$, decode this finite description and print every string in $S_n$ (to an output tape), then continue running $M$ to obtain $S_{n+1}$, and so on.

We show that $E$ enumerates exactly $S$.

(*Soundness*) Every string printed by $E$ belongs to some $S_n$, hence belongs to $\bigcup_{k=1}^{\infty} S_k = S$.

(*Completeness*) Let $\boldsymbol{x} \in S$. Since $S = \bigcup_{n=1}^{\infty} S_n$, there exists $N$ such that $\boldsymbol{x} \in S_N$. When $M$ emits $S_N$, the machine $E$ prints all elements of $S_N$, in particular $\boldsymbol{x}$. Thus every element of $S$ is eventually printed by $E$.

Therefore $E$ lists exactly the elements of $S$ (possibly with repeats), so $S$ is recursively enumerable. ■

## B Precision

### B.1 Exhaustion-level precision

**Example 2** (Precision need not exist)**.** *Let $\Sigma = \{a, b\}$. Define the target language $T \stackrel{\text{def}}{=} \{a^n \mid n \in \mathbb{N}\}$ with an exhaustion $\boldsymbol{T}$,*

$$T_n \stackrel{\text{def}}{=} \{a^i \mid 1 \le i \le n\}. \tag{34}$$

*Let the guess language be $G \stackrel{\text{def}}{=} T \cup \{b^n \mid n \in \mathbb{N}\}$ and define an exhaustion*

$$G_n \stackrel{\text{def}}{=} \{a^1, \ldots, a^n\} \cup \{b^1, \ldots, b^{2^{\lfloor \log_2 n \rfloor}}\}, \tag{35}$$

*i.e., at time $n$ we include all $a$-strings up to length $n$ and all $b$-strings up to the largest power of 2 not exceeding $n$. Then $G_n \subseteq G_{n+1}$ and $G_n \uparrow G$. Because $T_n \subseteq G_n$ and the $a$-strings and $b$-strings are disjoint,*

$$|T_n \cap G_n| = |T_n| = n, \tag{36}$$

$$|G_n| = n + 2^{\lfloor \log_2 n \rfloor}, \tag{37}$$

*and therefore, we obtain*

$$p_n \stackrel{\text{def}}{=} \frac{|T_n \cap G_n|}{|G_n|}, \tag{38}$$

$$p_n = \frac{n}{n + 2^{\lfloor \log_2 n \rfloor}}. \tag{39}$$

*Evaluating $p_n$ along two subsequences gives different limit points, as we can see below.*

$$\lim_{m\to\infty} p_{2^m} = \lim_{m\to\infty} \frac{2^m}{2^m + 2^m} = \frac{1}{2}, \tag{40a}$$

$$\lim_{m\to\infty} p_{2^{m+1}-1} = \lim_{m\to\infty} \frac{2^{m+1} - 1}{(2^{m+1} - 1) + 2^m} = \frac{2}{3}. \tag{40b}$$

*Thus, by the definitions of* $\liminf$ *and* $\limsup$*,*

$$\pi_*(\boldsymbol{T}, \boldsymbol{G}) = \liminf_{n\to\infty} p_n \le \frac{1}{2}, \tag{41a}$$

$$\pi^*(\boldsymbol{T}, \boldsymbol{G}) = \limsup_{n\to\infty} p_n \ge \frac{2}{3}, \tag{41b}$$

*so the limit defining precision does not exist.*

### B.2 Membership-based precision

**Lemma B.1.** *Let $T \subseteq \Sigma^*$ be the target language and let $\boldsymbol{G} = \{G_n\}_{n=0}^{\infty}$ be an exhaustion of the guess language $G$. Then,*

$$\sup_{\boldsymbol{T} \in \mathcal{E}(T)} \pi_*(\boldsymbol{T}, \boldsymbol{G}) = \liminf_{n\to\infty} \frac{|T \cap G_n|}{|G_n|}. \tag{42}$$

*Proof.* We prove the lemma in two parts.

**Part 1** ($\sup_{\boldsymbol{T}\in\mathcal{E}(T)} \pi_*(\boldsymbol{T}, \boldsymbol{G}) \le \liminf_{n\to\infty} \frac{|T\cap G_n|}{|G_n|}$)**.** Choose $\boldsymbol{T} \in \mathcal{E}(T)$ arbitrarily. Because $T_n \subseteq T$ for all $n$, we have $|T_n \cap G_n| \le |T \cap G_n|$ for every $n$. Dividing by $|G_n|$ and taking $\liminf$ yields

$$\pi_*(\boldsymbol{T}, \boldsymbol{G}) = \liminf_{n\to\infty} \frac{|T_n \cap G_n|}{|G_n|} \le \liminf_{n\to\infty} \frac{|T \cap G_n|}{|G_n|}. \tag{43}$$

Taking the supremum over $\boldsymbol{T} \in \mathcal{E}(T)$ proves

$$\sup_{\boldsymbol{T} \in \mathcal{E}(T)} \pi_*(\boldsymbol{T}, \boldsymbol{G}) \le \liminf_{n\to\infty} \frac{|T \cap G_n|}{|G_n|}. \tag{44}$$

**Part 2** ($\liminf_{n\to\infty} \frac{|T\cap G_n|}{|G_n|} \le \sup_{\boldsymbol{T}\in\mathcal{E}(T)} \pi_*(\boldsymbol{T}, \boldsymbol{G})$)**.**

Fix an enumeration $\boldsymbol{Y} = \{Y_n\}_{n=0}^{\infty}$ with $Y_n \uparrow T$. We define an exhaustion $\boldsymbol{T}^\star$. For each $n \in \mathbb{N}$, define

$$T_n^\star \stackrel{\text{def}}{=} (T \cap G_n) \cup Y_n. \tag{45}$$

Note that each $T_n^\star$ is finite, $T_n^\star \subseteq T_{n+1}^\star$, and $T_n^\star \uparrow T$. Thus, $\boldsymbol{T}^\star \in \mathcal{E}(T)$. Moreover,

$$T_n^\star \cap G_n = \big((T \cap G_n) \cup Y_n\big) \cap G_n = (T \cap G_n) \cup (Y_n \cap G_n) = (T \cup Y_n) \cap G_n = T \cap G_n. \tag{46}$$

where the last equality uses $Y_n \subseteq T$ for all $n \geq 0$. Hence, $\frac{|T_n^\star \cap G_n|}{|G_n|} = \frac{|T \cap G_n|}{|G_n|}$ for all $n \geq 0$. Taking $\liminf$ yields

$$\pi_*(\boldsymbol{T}^\star, \boldsymbol{G}) = \liminf_{n\to\infty} \frac{|T_n^\star \cap G_n|}{|G_n|} = \liminf_{n\to\infty} \frac{|T \cap G_n|}{|G_n|}, \tag{47}$$

which proves the reverse inequality in Eq. (42) as $\pi_*(\boldsymbol{T}^\star, \boldsymbol{G}) \leq \sup_{\boldsymbol{T}\in\mathcal{E}(T)} \pi_*(\boldsymbol{T}, \boldsymbol{G})$. ■

An analogous property holds for upper precision.

**Lemma B.2.** *Let $T \subseteq \Sigma^*$ be the target language and let $\boldsymbol{G} = \{G_n\}_{n=0}^{\infty}$ be an exhaustion of the guess language $G$. Then,*

$$\sup_{\boldsymbol{T}\in\mathcal{E}(T)} \pi^*(\boldsymbol{T}, \boldsymbol{G}) = \limsup_{n\to\infty} \frac{|T \cap G_n|}{|G_n|}. \tag{48}$$

*Proof.* Analogous to proof of Lemma B.1. ■

**Theorem 2.1.** *Let $f(n)\colon \mathbb{N} \to \mathbb{N}$ be a function with $\lim_{N\to\infty} \sum_{n=1}^{N} f(n) = \infty$. Let $T \subseteq \Sigma^*$ be the target language and let $\boldsymbol{G} = \{G_n\}_{n=0}^{\infty}$ be an $f$-bounded exhaustion of the guess $G$. Then,*

$$\sup_{\boldsymbol{T}\in\mathcal{E}_f(T)} \pi_*(\boldsymbol{T}, \boldsymbol{G}) = \liminf_{n\to\infty} \frac{|T \cap G_n|}{|G_n|}. \tag{9}$$

*Proof.* We prove the theorem in two parts.

**Part 1 ($\sup_{\boldsymbol{T}\in\mathcal{E}_f(T)} \pi_*(\boldsymbol{T}, \boldsymbol{G}) \leq \liminf_{n\to\infty} \frac{|T\cap G_n|}{|G_n|}$).** This is analogous to the proof of Part 1 in Lemma B.1.

**Part 2 ($\liminf_{n\to\infty} \frac{|T\cap G_n|}{|G_n|} \leq \sup_{\boldsymbol{T}\in\mathcal{E}_f(T)} \pi_*(\boldsymbol{T}, \boldsymbol{G})$).** Fix an enumeration $\boldsymbol{Z} = \{Z_n\}_{n=0}^{\infty}$ of $T \setminus G$.[5] Let $h_n \stackrel{\text{def}}{=} |\Delta G_n \setminus T|$ and $H_n = \sum_{i=1}^{n} h_i$. We construct $\boldsymbol{T}^\star = \{T_n^\star\}_{n=0}^{\infty} \in \mathcal{E}_f(T)$ and distinguish two cases.

**Case 1.** $\lim_{n\to\infty} H_n = \infty$.

Define for each $n \geq 0$,

$$T_n^\star \stackrel{\text{def}}{=} (T \cap G_n) \cup Z_{H_n}. \tag{49}$$

**Claim 1.** $T_n^\star$ is an $f$-bounded exhaustion of $T$.

To start, note $T_n^\star$ is finite and $T_n^\star \subseteq T$. Moreover, $T_n^\star \subseteq T_{n+1}^\star$ since $G_n \subseteq G_{n+1}$ implies $T \cap G_n \subseteq T \cap G_{n+1}$, and since $H_n \leq H_{n+1}$ holds. Furthermore, by construction

$$|\Delta T_n^\star| = |T \cap \Delta G_n| + h_n \leq f(n). \tag{50}$$

Finally, because $\limsup H_n = \infty$, every $Z_k$ appears in $T_n^\star$ for sufficiently large $n$. Also, $(T \cap G_n) \uparrow (T \cap G)$. Hence $T_n^\star \uparrow T$ as desired.

**Claim 2.** $T_n^\star \cap G_n = T \cap G_n$.

Finally, for every $n$,

$$T_n^\star \cap G_n = (T \cap G_n) \cup (Z_{H_n} \cap G_n) = T \cap G_n, \tag{51}$$

[5] If $T \setminus G$ is finite, we can fix a single-step exhaustion that converges to a finite language.

because $Z_{H_n} \subseteq T \setminus G$ is disjoint from $G_n$. We have that

$$\pi_*(\boldsymbol{T}^\star, \boldsymbol{G}) = \liminf_{n\to\infty} \frac{|T_n^\star \cap G_n|}{|G_n|} = \liminf_{n\to\infty} \frac{|T \cap G_n|}{|G_n|}. \tag{52}$$

**Case 2.** $\limsup_n H_n < \infty$.

First, we argue that the following $\liminf$ is 1.

$$\liminf_{n\to\infty} \frac{|T \cap G_n|}{|G_n|} = \liminf_{n\to\infty} \left(1 - \frac{H_n}{|G_n|}\right) = 1. \tag{53}$$

This holds as $\limsup_n H_n < \infty$ and $\boldsymbol{G}$ exhausts an infinite language $G$. Next, we show that we can construct an exhaustion that achieves the limit. Choose an exploration set $R \subset \mathbb{N}$ according to Definition 4.1 and write $r(m) \stackrel{\text{def}}{=} |R \cap \{1, \ldots, m\}|$.

Fix an enumeration $\boldsymbol{Z} = \{Z_j\}_{j=0}^\infty$ of $T \setminus G$ with $Z_0 \stackrel{\text{def}}{=} \emptyset$. Fix an enumeration $\boldsymbol{Y} = \{Y_j\}_{j=0}^\infty$ of $T \cap G$ such that

$$Y_{|G_n \cap T|} = G_n \cap T \qquad \text{for all } n. \tag{54}$$

Note that the enumerations are disjoint.

Let $m_n \stackrel{\text{def}}{=} |G_n|$. We define the exhaustion $\boldsymbol{T}^\star = \{T_n^\star\}_{n=0}^\infty$ by

$$T_n^\star \stackrel{\text{def}}{=} Y_{m_n - r(m_n)} \cup Z_{r(m_n)}. \tag{55}$$

Then $|T_n^\star| = (m_n - r(m_n)) + r(m_n) = m_n = |G_n|$, and thus $|\Delta T_n^\star| = m_n - m_{n-1} \leq f(n)$. Furthermore, $T_n^\star \subseteq T$ for all $n$. Moreover, since $m_n$, $r(m_n)$ and $m_n - r(m_n)$ are nondecreasing, also $T_n^\star \subseteq T_{n+1}^\star$. Finally, $m_n \to \infty$, $m_n - r(m_n) \to \infty$ and $r(m_n) \to \infty$ (because $R$ is infinite, but contains no consecutive integers), we have $T_n^\star \uparrow T$. As $H_n$ is bounded and $r(m_n) \to \infty$, for large enough $n$ we have

$$|T_n^\star \cap G_n| \;\geq\; |Y_{m_n - r(m_n)}| \;=\; m_n - r(m_n). \tag{56}$$

Dividing by $m_n = |G_n|$ and taking $\liminf$ yields

$$\pi_*(\boldsymbol{T}^\star, \boldsymbol{G}) = \liminf_{n\to\infty} \frac{|T_n^\star \cap G_n|}{|G_n|} \tag{57}$$

$$\geq\; \liminf_{n\to\infty} \left(1 - \frac{r(m_n)}{m_n}\right) = 1, \tag{58}$$

using $r(m_n)/m_n \to 0$. Together with Eq. (53), this yields $\pi_*(\boldsymbol{T}^\star, \boldsymbol{G}) = 1$. ■

# C Recall

## C.1 Exhaustion-level recall

**Example 3** (Recall need not exist)**.** *Let $\Sigma = \{a\}$. Define the target language $T \overset{\text{def}}{=} \{a^n \mid n \in \mathbb{N}\}$ with exhaustion $\boldsymbol{T}$ given by*

$$T_n \overset{\text{def}}{=} \{a^i \mid 1 \le i \le n\}. \tag{59}$$

*Define the guess language $G \subseteq T$ by alternating blocks: for each $m \ge 1$ let*

$$I_m \overset{\text{def}}{=} \left(2^{2m-1},\, 2^{2m}\right] \tag{60}$$

*and*

$$E_m \overset{\text{def}}{=} \left(2^{2m},\, 2^{2m+1}\right]. \tag{61}$$

*Then include all $a^n$ with $n \in I_m$ and exclude $a^n$ with $n \in E_m$. Formally,*

$$G \overset{\text{def}}{=} \{a^n \mid n \in \bigcup_{m \ge 1} I_m\}. \tag{62}$$

*Let $\boldsymbol{G}$ be the canonical exhaustion induced by $\boldsymbol{T}$,*

$$G_n \overset{\text{def}}{=} G \cap T_n. \tag{63}$$

*Then $G_n \subseteq G_{n+1}$ and $G_n \uparrow G$.*

*Since $G_n \subseteq T_n$, we have*

$$|T_n \cap G_n| = |G_n|, \qquad |T_n| = n. \tag{64}$$

*Defining*

$$r_n \overset{\text{def}}{=} \frac{|T_n \cap G_n|}{|T_n|}, \tag{65}$$

*we obtain*

$$r_n = \frac{|G_n|}{n}. \tag{66}$$

*Let $S_m \overset{\text{def}}{=} \sum_{j=1}^{m} |I_j| = \sum_{j=1}^{m} 2^{2j-1} = \frac{2}{3}\left(4^m - 1\right)$. Then at the end of an included block we have $|G_{2^{2m}}| = S_m$, hence*

$$r_{2^{2m}} = \frac{S_m}{2^{2m}} = \frac{\frac{2}{3}(4^m - 1)}{4^m} = \frac{2}{3}\left(1 - 4^{-m}\right) \xrightarrow[m\to\infty]{} \frac{2}{3}. \tag{67}$$

*At the end of the subsequent excluded block the numerator is still $S_m$, so*

$$r_{2^{2m+1}} = \frac{S_m}{2^{2m+1}} = \frac{\frac{2}{3}(4^m - 1)}{2 \cdot 4^m} = \frac{1}{3}\left(1 - 4^{-m}\right) \xrightarrow[m\to\infty]{} \frac{1}{3}. \tag{68}$$

*Thus $r_n$ has two different limit points, and therefore, by the definitions of* $\liminf$ *and* $\limsup$*,*

$$\rho_*(\boldsymbol{T}, \boldsymbol{G}) = \liminf_{n\to\infty} r_n \le \frac{1}{3}, \tag{69}$$

$$\rho^*(\boldsymbol{T}, \boldsymbol{G}) = \limsup_{n\to\infty} r_n \ge \frac{2}{3}, \tag{70}$$

*so the limit defining recall does not exist.*

## C.2 Coverage-based recall

**Lemma 2.2.** *Let $T \subseteq \Sigma^*$ be the target language with an exhaustion $\boldsymbol{T} = \{T_n\}_{n=0}^{\infty}$ and let $G$ be the guess language. Then,*

$$\sup_{\boldsymbol{G} \in \mathcal{E}(G)} \rho_*(\boldsymbol{T}, \boldsymbol{G}) = \liminf_{n\to\infty} \frac{|T_n \cap G|}{|T_n|}. \tag{12}$$

*Proof.* We prove the lemma in two parts.

**Part 1 ($\sup_{\boldsymbol{G}\in\mathcal{E}(G)} \rho_*(\boldsymbol{T},\boldsymbol{G}) \le \liminf_{n\to\infty} \frac{|T_n\cap G|}{|T_n|}$).** Choose $\boldsymbol{G}\in\mathcal{E}(G)$ arbitrarily. Because $G_n\subseteq G$ for all $n$, we have $|T_n\cap G_n|\le|T_n\cap G|$ for every $n$. Dividing by $|T_n|$ and taking $\liminf$ yields

$$\rho_*(\boldsymbol{T},\boldsymbol{G}) = \liminf_{n\to\infty} \frac{|T_n\cap G_n|}{|T_n|} \le \liminf_{n\to\infty} \frac{|T_n\cap G|}{|T_n|}. \tag{71}$$

Taking the supremum over $\boldsymbol{G}\in\mathcal{E}(G)$ proves

$$\sup_{\boldsymbol{G}\in\mathcal{E}(G)} \rho_*(\boldsymbol{T},\boldsymbol{G}) \le \liminf_{n\to\infty} \frac{|T_n\cap G|}{|T_n|}. \tag{72}$$

**Part 2 ($\liminf_{n\to\infty} \frac{|T_n\cap G|}{|T_n|} \le \sup_{\boldsymbol{G}\in\mathcal{E}(G)} \rho_*(\boldsymbol{T},\boldsymbol{G})$).**

To show the reverse inequality, fix an exhaustion $\boldsymbol{Y}=\{Y_n\}_{n=0}^\infty$ with $\boldsymbol{Y}\uparrow G$. Then, we define the following exhaustion $\boldsymbol{G}^\star$ of $G$: for each $n\in\mathbb{N}$, let

$$G_n^\star \stackrel{\text{def}}{=} (G\cap T_n)\ \cup\ Y_n. \tag{73}$$

Then each $G_n^\star$ is finite, $G_n^\star\subseteq G_{n+1}^\star$, and $G_n^\star\uparrow G$, so $\boldsymbol{G}^\star\in\mathcal{E}(G)$. Moreover,

$$\begin{aligned} G_n^\star\cap T_n &= \big((G\cap T_n)\cup Y_n\big)\cap T_n && (74)\\ &= (G\cap T_n)\cup(Y_n\cap T_n) && (75)\\ &= G\cap T_n, && (76)\end{aligned}$$

where the last equality uses $Y_n\cap T_n\subseteq G\cap T_n$. Hence, for all $n$,

$$\frac{|G_n^\star\cap T_n|}{|T_n|} = \frac{|G\cap T_n|}{|T_n|}. \tag{77}$$

Taking $\liminf$ yields

$$\rho_*(\boldsymbol{T},\boldsymbol{G}^\star) = \liminf_{n\to\infty} \frac{|G_n^\star\cap T_n|}{|T_n|} = \liminf_{n\to\infty} \frac{|G\cap T_n|}{|T_n|}, \tag{78}$$

which proves the reverse inequality in Eq. (12). ■

The same property can be proven for upper recall in an analogous way.

**Lemma C.1.** *Let $T\subseteq\Sigma^*$ be the target language and let $\boldsymbol{G}=\{G_n\}_{n=0}^\infty$ be an exhaustion of the guess language $G$. Then,*

$$\sup_{\boldsymbol{G}\in\mathcal{E}(G)} \rho^*(\boldsymbol{T},\boldsymbol{G}) = \limsup_{n\to\infty} \frac{|T_n\cap G|}{|T_n|}. \tag{79}$$

Furthermore, we formulate a theorem analogous to Theorem 2.1 for the recall.

**Theorem C.2.** *Let $f(n)\colon\mathbb{N}\to\mathbb{N}$ be a function with $\lim_{N\to\infty}\sum_{n=1}^N f(n)=\infty$. Let $G\subseteq\Sigma^*$ be the guess language and let $\boldsymbol{T}=\{T_n\}_{n=0}^\infty\in\mathcal{E}_f(T)$ be an $f$-bounded exhaustion of the target language $T$. Then,*

$$\sup_{\boldsymbol{G}\in\mathcal{E}_f(G)} \rho_*(\boldsymbol{T},\boldsymbol{G}) = \liminf_{n\to\infty} \frac{|T_n\cap G|}{|T_n|} \tag{80}$$

*Proof.* Analogous to proof of Theorem 2.1. ■

## D Tail precision

**Proposition 2.4.** *Let $T$ be the target language and $\boldsymbol{G}$ a single-step exhaustion of $G$. Then, $\widehat{\tau}_*(T, \boldsymbol{G})$ is either $0$ or $1$. Moreover, for any $c$-step exhaustion $\boldsymbol{G}$, $\widehat{\tau}_*(T, \boldsymbol{G})$ is attained after finitely many steps.*[6]

*Proof.* For each $n$, recall the definition of the step-wise tail precision:

$$t_n \overset{\text{def}}{=} \begin{cases} 1, & \text{if } |\Delta G_n| = 0, \\ \frac{|T \cap \Delta G_n|}{|\Delta G_n|}, & \text{otherwise.} \end{cases} \tag{81}$$

Assume first that at each step at most one new string is added, i.e. $|\Delta G_n| \leq 1$ (single-step exhaustion). Then $t_n \in \{0, 1\}$ for all $n$. Let $E \overset{\text{def}}{=} \{n \in \mathbb{N} : t_n = 0\}$ be the set of error steps. If $E$ is infinite, then $t_n = 0$ occurs infinitely often and thus

$$\liminf_{n \to \infty} t_n = 0. \tag{82}$$

If $E$ is finite, let $N_0 > \max E$. Then $t_n = 1$ for all $n \geq N_0$, and hence

$$\liminf_{n \to \infty} t_n = 1. \tag{83}$$

In either case, $\widehat{\tau}_*(T, \boldsymbol{G}) = \liminf_{n \to \infty} t_n$ is reached after finitely many steps and equals either $0$ or $1$.

More generally, suppose there is a constant $c \geq 1$ such that $|\Delta G_n| \leq c$ for all $n$. Then for every $n$ we have

$$t_n \in F_c \overset{\text{def}}{=} \left\{ \frac{i}{j} : i, j \in \mathbb{N}_0, 1 \leq j \leq c, 0 \leq i \leq j \right\}. \tag{84}$$

Let $\ell \overset{\text{def}}{=} \liminf_{n \to \infty} t_n$. Since $t_n$ takes values in the finite set $F_c$, there exists a minimum value among those that occur infinitely often; moreover this minimum equals $\ell$. In particular, $\ell \in F_c$ and $t_n = \ell$ occurs infinitely often.

Let $B \overset{\text{def}}{=} \{x \in F_c : x < \ell\}$. Every value $x \in B$ can occur only finitely many times, otherwise we would have $\liminf_{n \to \infty} t_n \leq x < \ell$, a contradiction. Hence there exists $N_0$ such that $t_n \notin B$ for all $n \geq N_0$, i.e. $t_n \geq \ell$ for all $n \geq N_0$. Since $t_n = \ell$ occurs infinitely often, there is also some $n \geq N_0$ with $t_n = \ell$. Therefore,

$$\inf_{m \geq N_0} t_m = \ell, \tag{85}$$

so the lower tail precision $\widehat{\tau}_*(T, \boldsymbol{G}) = \ell$ is attained from time $n \geq N_0$ onward. ■

**Proposition 3.3.** *Let $\mathcal{G}$ be a generator and let $\boldsymbol{G}$ be the exhaustion it produces against the adversarial exhaustion $\boldsymbol{A}$ of $A \subseteq T$. Then $\mathcal{G}$ is eventually valid (Def. 3.2) if and only if $\widehat{\tau}_*(T, \boldsymbol{G}) = 1$ is attained in finite time.*

*Proof.* Recall the definition of the step-wise tail precision:

$$t_n \overset{\text{def}}{=} \begin{cases} 1, & |\Delta G_n| = 0, \\ \frac{|T \cap \Delta G_n|}{|\Delta G_n|}, & |\Delta G_n| > 0. \end{cases} \tag{86}$$

By definition, $\widehat{\tau}_*(T, \boldsymbol{G}) = \liminf_{n \to \infty} t_n$.

($\Rightarrow$) If $\mathcal{G}$ is eventually valid, then there exists $n^\star$ such that for all $n \geq n^\star$,

$$\mathcal{G}(A_n, G_{n-1}) \subseteq T. \tag{87}$$

Equivalently, for all $n \geq n^\star$, every newly generated string lies in $T$, i.e., $\Delta G_n \subseteq T$. Thus $t_n = 1$ for all $n \geq n^\star$, which means the tail-precision sequence attains $1$ from some finite time onward.

[6] This means that there exists a timestep $n < \infty$, such that the step-wise tail precision $t_n = \widehat{\tau}_*(T, \boldsymbol{G})$ and $t_N \geq t_n$ for $N \geq n$.

($\Leftarrow$) Conversely, suppose $\widehat{\tau}_*(T, \boldsymbol{G}) = 1$ is attained in finite time. This implies that there exists $n^\star$ such that $t_n = 1$ for all $n \geq n^\star$. For any such $n$, if $|\Delta G_n| > 0$ then

$$\frac{|\Delta G_n \cap T|}{|\Delta G_n|} = 1 \quad \Longrightarrow \quad \Delta G_n \subseteq T, \tag{88}$$

and if $|\Delta G_n| = 0$ then there is nothing to show. Hence $\Delta G_n \subseteq T$ for all $n \geq n^\star$, i.e., $\mathcal{G}(A_n, G_{n-1}) \subseteq T$ for all $n \geq n^\star$. This is exactly eventual validity. ■

**Lemma D.1** (Tail precision 1 implies precision 1)**.** *Let $T$ be the target language and let $\boldsymbol{G} = \{G_n\}_{n=0}^{\infty}$ be an exhaustion of $G$. Then*

$$\widehat{\tau}_*(T, \boldsymbol{G}) = 1 \implies \widehat{\pi}_*(T, \boldsymbol{G}) = 1. \tag{89}$$

*Proof.* For $n \geq 1$, define

$$s_n \stackrel{\text{def}}{=} |\Delta G_n|, \qquad e_n \stackrel{\text{def}}{=} |\Delta G_n \setminus T|. \tag{90}$$

If $s_n = 0$, let $t_n = 1$, otherwise let

$$t_n \stackrel{\text{def}}{=} \frac{|\Delta G_n \cap T|}{|\Delta G_n|} = 1 - \frac{e_n}{s_n} \qquad \text{and} \qquad p_n \stackrel{\text{def}}{=} \frac{|G_n \cap T|}{|G_n|} = 1 - \frac{|G_n \setminus T|}{|G_n|}. \tag{91}$$

The assumption $\widehat{\tau}_*(T, \boldsymbol{G}) = 1$ means $\liminf_{n\to\infty} t_n = 1$. Since always $t_n \leq 1$, this implies $\lim_{n\to\infty} t_n = 1$, hence

$$\lim_{n\to\infty} \frac{e_n}{s_n} = 0. \tag{92}$$

Now let $E_n \stackrel{\text{def}}{=} |G_n \setminus T|$ be the cumulative number of errors up to time $n$. We have

$$E_n = \sum_{i=1}^{n} e_i \qquad \text{and} \qquad |G_n| = \sum_{i=1}^{n} s_i. \tag{93}$$

Fix $\varepsilon > 0$. Since $\lim_{n\to\infty} e_n/s_n = 0$, there exists $N$ such that $e_n \leq \varepsilon s_n$ for all $n \geq N$. Then for any $n \geq N$,

$$E_n = \sum_{i=1}^{N-1} e_i + \sum_{i=N}^{n} e_i \leq \sum_{i=1}^{N-1} e_i + \varepsilon \sum_{i=N}^{n} s_i \leq \sum_{i=1}^{N-1} e_i + \varepsilon |G_n|. \tag{94}$$

Dividing by $|G_n|$ gives

$$\frac{E_n}{|G_n|} \leq \frac{\sum_{i=1}^{N-1} e_i}{|G_n|} + \varepsilon. \tag{95}$$

Since $G_n \uparrow G$ and $G$ is an infinite set, the first term tends to 0, so $\limsup_{n\to\infty} \frac{E_n}{|G_n|} \leq \varepsilon$. As $\varepsilon > 0$ can be chosen arbitrarily, we conclude $\lim_{n\to\infty} \frac{E_n}{|G_n|} = 0$, and therefore

$$\lim_{n\to\infty} p_n = \lim_{n\to\infty} 1 - \frac{E_n}{|G_n|} \to 1. \tag{96}$$

Hence $\widehat{\pi}_*(T, \boldsymbol{G}) = \liminf_{n\to\infty} p_n = 1$. ■

# E Generation without novelty

## E.1 Generation without novelty but perfect tail precision

**Theorem E.1.** *Let $\mathcal{T}$ be a countable collection of languages and let $T \in \mathcal{T}$ be the target language. Fix an enumeration $\boldsymbol{T} = \{T_n\}_{n=0}^{\infty}$ of $T$. There exists a generator $\mathcal{G}$ such that, for any $f$-bounded adversarial exhaustion $\boldsymbol{A} = \{A_n\}_{n=0}^{\infty}$ with $A_n \uparrow A \subseteq T$ and $\widehat{\rho}_*(\boldsymbol{T}, A) \geq \alpha$, the generated exhaustion $\boldsymbol{G}$ is $f$-bounded and satisfies:*

$$\widehat{\rho}_*(\boldsymbol{T}, G) \geq \alpha, \qquad \widehat{\tau}_*(T, \boldsymbol{G}) = 1, \qquad \widehat{\pi}_*(T, \boldsymbol{G}) = 1. \tag{97}$$

*Moreover, for $c$-step exhaustions the recall guarantee is tight in an adversarial sense.*

*Proof.* Define $\mathcal{G}$ by simply copying the adversary:

$$\mathcal{G}(A_n, G_{n-1}) \stackrel{\text{def}}{=} \Delta A_n \qquad \text{for all } n \geq 0. \tag{98}$$

Hence, $\Delta G_n = \Delta A_n$, and $G_n = A_n$. Then $\boldsymbol{G}$ is $f$-bounded whenever $\boldsymbol{A}$ is.

**Precision and tail precision.** Since $G_n = A_n \subseteq T$ for all $n$, we have

$$\widehat{\pi}_*(T, \boldsymbol{G}) = \liminf_{n\to\infty} \frac{|G_n \cap T|}{|G_n|} = \liminf_{n\to\infty} \frac{|G_n|}{|G_n|} = 1. \tag{99}$$

The step-wise tail precision $t_n = 1$ whenever $\Delta G_n = \emptyset$.

Furthermore, $\Delta G_n = \Delta A_n \subseteq T$, so

$$\widehat{\tau}_*(T, \boldsymbol{G}) = \liminf_{n\to\infty} \frac{|T \cap \Delta G_n|}{|\Delta G_n|} = 1. \tag{100}$$

**Recall.** Because $G_n = A_n$ for all $n$,

$$\widehat{\rho}_*(\boldsymbol{T}, G) = \widehat{\rho}_*(\boldsymbol{T}, A) \geq \alpha. \tag{101}$$

**Tightness of the recall bound.** We construct a case in which the recall cannot exceed $\alpha$.

Assume $f$ enforces a $c$-step exhaustion. Fix $T \in \mathcal{T}$ and a strict sublanguage $A \subset T$ with $\widehat{\rho}_*(\boldsymbol{T}, A) = \alpha < 1$, and assume also $A \in \mathcal{T}$.

Consider a $c$-step exhaustion of $A$. Any generator $\mathcal{G}$ that is required to satisfy $\widehat{\tau}_*(T, \boldsymbol{G}) = 1$ for *every* possible true language in $\mathcal{T}$ must in particular satisfy it when the true language is $A$. Recall that for each $n$, we have

$$t_n = \begin{cases} 1, & \text{if } |\Delta G_n| = 0, \\ \frac{|A \cap \Delta G_n|}{|\Delta G_n|}, & \text{otherwise.} \end{cases} \tag{102}$$

Since $|\Delta G_n| \leq c$, each $t_n$ belongs to the finite set

$$F_c \stackrel{\text{def}}{=} \Big\{ \frac{i}{j} : i, j \in \mathbb{N}_0, 1 \leq j \leq c, 0 \leq i \leq j \Big\}. \tag{103}$$

If $\liminf_{n\to\infty} t_n = 1$, then we must have $t_n = 1$ for all sufficiently large $n$ because $F_c$ is finite: otherwise some value $< 1$ would occur infinitely often, forcing the $\liminf$ to be $\leq \max(F_c \backslash \{1\}) < 1$. Hence, when the true language is $A$, the generator can output strings outside $A$ only finitely many times.

Now note that the adversary exhaustion can be identical whether the true language is $A$ or $T$. It follows that for the instance where the target language is $T$, the generator must also eventually restrict to $A$, as it cannot possibly know if the true language is $T$ or $A$. Consequently it cannot guarantee asymptotic recall exceeding that of $A$ inside $T$, i.e., it cannot guarantee $\widehat{\rho}_*(\boldsymbol{T}, G) > \alpha$ in general. ■

**Algorithm 1** Language generation in the limit without novelty or perfect tail precision. **Input:** adversarial exhaustion $\{A_n\}_{n=0}^{\infty}$ with $A_n \uparrow A \subseteq T$, enumeration $\boldsymbol{E}$ of $\Sigma^*$, infinite exploration set $R \subset \mathbb{N}$. **Ensures:** increasing exhaustion $\boldsymbol{G} = \{G_n\}_{n=0}^{\infty}$.

$G_0 \leftarrow \emptyset$, $G_1 \leftarrow A_1$, $j \leftarrow 1$
$r_{\text{next}} \leftarrow \min(R)$
**for** $n = 1, 2, 3, \ldots$ **do**
  **if** $|G_n| \geq r_{\text{next}}$ **and** $\Delta A_{n+1} \neq \emptyset$ **then**
    $G_{n+1} \leftarrow G_n \cup \Delta E_j$ ▷ exploration round
    $j \leftarrow j + 1$
    $r_{\text{next}} \leftarrow \min\{r \in R : r > r_{\text{next}}\}$
  **else**
    $G_{n+1} \leftarrow G_n \cup \Delta A_{n+1}$ ▷ safe round
  **end if**
**end for**

### E.2 Generation without novelty and without perfect tail precision

**Theorem 4.3.** *Let $\mathcal{T}$ be a countable collection of languages and $T \in \mathcal{T}$ a target language. Fix an enumeration $\boldsymbol{T} = \{T_n\}_{n=0}^{\infty}$ of $T$. There exists a generator $\mathcal{G}$, such that, for any $f$-bounded adversarial exhaustion $\boldsymbol{A} = \{A_n\}_{n=0}^{\infty}$ with $A_n \uparrow A \subseteq T$, the generated exhaustion $\boldsymbol{G}$ is $f$-bounded and satisfies:*

$$\widehat{\rho}_*(\boldsymbol{T}, G) = 1, \qquad \widehat{\pi}_*(T, \boldsymbol{G}) = 1. \tag{26}$$

*Proof.* Since novelty is not required, the generator may repeat the adversary's outputs. We will construct an algorithm that mimics the adversary's strings and occasionally introduces new, potentially incorrect strings at a vanishing rate. We call these insertions exploration rounds. Fix an arbitrary enumeration $\boldsymbol{E} = \{E_n\}_{n=0}^{\infty}$ of $\Sigma^*$. Choose an exploration set $R \subset \mathbb{N}$ according to Definition 4.1, and let $r(m) \stackrel{\text{def}}{=} |R \cap \{1, \ldots, m\}|$. We use $R$ to trigger exploration rounds whenever $|G_n|$ reaches an $r \in R$. We construct an $f$-bounded $\boldsymbol{G} = \{G_n\}_{n=0}^{\infty}$ iteratively as described in Algorithm 1. The exhaustion $\boldsymbol{G}$ is $f$-bounded because, on a safe round, it adds exactly the strings in $\Delta A_{n+1}$, while an exploration round adds only one string and is taken only if $\Delta A_{n+1} \neq \emptyset$. In the latter case, $f(n+1) \geq 1$, and hence the exploration output also respects the bound.

By construction, $G_n$ is increasing. Moreover, we ensure that exploration happens only whenever $|G_n|$ reaches the next value in $R$. This will ensure that for $G_n$ with $|G_n| = m_n$, the number of explorative strings is at most $r(m_n)$.

**Recall.** Every $\Delta E_n$ gets added eventually, hence $G_n \uparrow \Sigma^*$ and

$$\widehat{\rho}_*(\boldsymbol{T}, G) = \liminf_{n\to\infty} \frac{|T_n \cap G|}{|T_n|} = \liminf_{n\to\infty} \frac{|T_n|}{|T_n|} = 1. \tag{104}$$

**Precision.** For each $G_n$ with $|G_n| = m_n$ the number of outputs from explorations (and hence possible hallucinations) is at most $r(m_n)$; all other outputs are parroted adversary elements and thus lie in $T$ because $A \subseteq T$. Hence

$$|G_n \cap T| \geq m_n - r(m_n) \tag{105}$$

and therefore

$$\widehat{\pi}_*(T, \boldsymbol{G}) = \liminf_{n\to\infty} \frac{|G_n \cap T|}{|G_n|} \tag{106}$$

$$\geq \liminf_{n\to\infty} \left(1 - \frac{r(m_n)}{m_n}\right) = 1. \tag{107}$$

■

## F Pods algorithm

In this section, we give Kleinberg & Wei's (2025b) algorithm and describe its main ideas. We fix a countable collection of languages $\mathcal{T} \subseteq \mathcal{P}(\Sigma^*)$ together with (i) a global order on languages in $\mathcal{T}$, yielding an ordered sequence $(S'_i)_{i\geq 1}$ and (ii) an enumeration of $\Sigma^*$ that induces a total order $\prec$. For a string $\boldsymbol{x} \in T$ we define its successor by $\mathrm{succ}_T(\boldsymbol{x})$ in $(T, \prec)$.[7]

We recall the two main technical primitives from Kleinberg & Wei (2025b): the *identified intersection* procedure and the *pods* construction.

**Consistency and intersection chains.** Let $T \in \mathcal{T}$ be the target language. An adversary reveals an exhaustion $\boldsymbol{A}$. Note that in Kleinberg & Wei (2025b) this exhaustion is single-step-bounded. However, we write Algorithm 2 in a way such that it can also be run with a $k$-batched adversary, a setting that we analyze in §G.1.

**Definition F.1.** *Let $\boldsymbol{A} = \{A_n\}_{n=0}^{\infty}$ be an adversarial exhaustion and $S$ an arbitrary language. We call $S$ **consistent** with $\boldsymbol{A}$ up to timestep $n$ if $A_n \subseteq S$.*

**Definition F.2.** *Let the consistent languages at time $n$ be listed in the global order as $S_1^{(n)}, S_2^{(n)}, S_3^{(n)}, \ldots$, where $(S_i^{(n)})_{i\geq 1}$ is obtained by scanning $S_1, S_2, \ldots$ and keeping exactly those that contain $A_n$. If only finitely many, say $k$, consistent languages remain at time $n$, we pad the sequence by repeating the last language, namely*

$$S_l^{(n)} = S_k^{(n)}, \qquad l \geq k \tag{108}$$

*We define the descending **chain of intersections** at time $n$ by*

$$\mathcal{I}_i^{(n)} \stackrel{\text{def}}{=} \bigcap_{j\leq i} S_j^{(n)}, \qquad C_n \stackrel{\text{def}}{=} \mathcal{I}_1^{(n)} \supseteq \mathcal{I}_2^{(n)} \supseteq \mathcal{I}_3^{(n)} \supseteq \cdots. \tag{109}$$

Note that $C_n$ is always an infinite chain (Kleinberg & Wei, 2025b, Eq. 3). See also Algorithm 3.

**Identified intersection.** Algorithm 4 describes how to find the best guess $\mathcal{I}^{(n)}$ of the target language by intersecting the remaining consistent hypotheses. $\mathcal{I}^{(n)}$ is called **identified intersection**. The algorithm chooses an index $i(n) \in \mathbb{N}$ such that the output $\mathcal{I}^{(n)} \stackrel{\text{def}}{=} \mathcal{I}_{i(n)}^{(n)}$ is an infinite set (except in finitely many cases) and moves in a controlled way over time as $n$ increases. Concretely, if the descending chain of intersections remains the same between rounds $n$ and $n+1$, the algorithm either keeps $i(n+1) = i(n)$ or increases it by one (moving one step deeper in the intersection chain) provided the next intersection is still infinite. If the set of consistent hypotheses changes, the algorithm resets $i(n+1)$ to the deepest level at which the old and new intersection chains still agree on an infinite set.

We use the following terminology matching Kleinberg & Wei, 2025b. We say that $\mathcal{I}^{(n)}$ is **valid** if $\mathcal{I}^{(n)} \subseteq T$; and **full** if $A \subseteq \mathcal{I}^{(n)}$.

Kleinberg & Wei (2025b, Lem. 2.5) establish the following structural properties of $\mathcal{I}^{(n)}$: there exists a finite time $N_0$ such that for all $n \geq N_0$,

1. (*eventual validity*) $\mathcal{I}^{(n)}$ is valid;
2. (*eventual comparability*) either $\mathcal{I}^{(n)} \subset \mathcal{I}^{(n+1)}$ or $\mathcal{I}^{(n+1)} \subset \mathcal{I}^{(n)}$;
3. (*full infinitely many times*) the event $\mathcal{I}^{(n)}$ *is full* occurs infinitely often.

Intuitively, $\mathcal{I}^{(n)}$ is the algorithm's current safe proxy for $T$. Once $\mathcal{I}^{(n)}$ is valid, every string in it lies in $T$, so outputting from $\mathcal{I}^{(n)}$ preserves validity. Eventual comparability means that after finitely many steps the sequence evolves keeping some inclusion relation between consecutive identified intersections, which the later analysis repeatedly exploits to control how the identified intersection changes over time. Finally, the fact that $\mathcal{I}^n$ is full infinitely often guarantees that the process repeatedly returns to an intersection that contains $A$, a property which can be exploited to support high recall.

[7] Note that $\prec$ can be applied to any language $S$ in $\Sigma^*$, as $S \subseteq \Sigma^*$.

**Algorithm 2** Pods algorithm for language generation in the limit (after Kleinberg & Wei, 2025b). UPDATECHAIN takes the adversary's input and a sequence of languages and returns an updated sequence of languages and a chain of intersections of consistent languages; IDINTERSECT takes the timestep and a level index, two intersection chains and an identified intersection and returns the next level index and the next identified intersection; AGGRSET takes two identified intersections and two intersection chains and returns an aggressive set, i.e., a superset of the current identified intersection. **Input:** adversarial exhaustion $\{A_n\}_{n=0}^{\infty}$ with $A_n \uparrow A \subseteq T$, pod size sequence $(s_n)_{n\geq 1}$, ordered sequence $(S'_i)_{i\geq 1}$ of languages in $\mathcal{T}$ **Ensures:** single-step exhaustion $\boldsymbol{G} = \{G_n\}_{n=0}^{\infty}$.

$G_0 \leftarrow \emptyset$, $U_0 \leftarrow \emptyset$ {unavailable strings}, $\mathcal{P}_0 \leftarrow \emptyset$ {pod pool}, $(S_i^{(0)})_{i\geq 1} \leftarrow (S'_i)_{i\geq 1}$ {consistent languages}, $k_0 \leftarrow 1$, $\mathcal{I}^{(0)} \leftarrow \emptyset$, $C_0 \leftarrow \emptyset$
**for** $n = 1, 2, 3, \ldots$ **do**
  Observe $A_n$
  $\big((S_i^{(n)})_{i\geq 1}, C_n\big) \leftarrow \text{UPDATECHAIN}\big(\Delta A_n, (S_i^{(n-1)})_{i\geq 1}\big)$
  $(\mathcal{I}^{(n)}, k_n) \leftarrow \text{IDINTERSECT}(n, k_{n-1}, \mathcal{I}^{(n-1)}, C_{n-1}, C_n)$
  $\widetilde{\mathcal{I}}^{(n)} \leftarrow \text{AGGRSET}(n, \mathcal{I}^{(n-1)}, \mathcal{I}^{(n)}, C_{n-1}, C_n)$
  $U_n \leftarrow U_{n-1} \cup A_n$.
  $P_n \leftarrow$ the $s_n$ $\prec$-smallest elements of $\widetilde{\mathcal{I}}^{(n)} \setminus U_n$.
  $U_n \leftarrow U_n \cup P_n$
  $\mathcal{P}_n \leftarrow \mathcal{P}_{n-1} \cup P_n$.
  $G_n \leftarrow G_{n-1} \cup \min_{\prec}(\mathcal{P}_n \setminus (A_n \cup G_{n-1}))$.
  $U_n \leftarrow U_n \cup G_n$.
**end for**

**Pods generator algorithm.** Kleinberg & Wei introduce a generating procedure (see Algorithm 2) that achieves the properties in Theorem 3.5. In each timestep $n$ it decides on an **aggressive set** $\widetilde{\mathcal{I}}^{(n)}$ (see Algorithm 5, as defined in Kleinberg & Wei (2025b, Sec. 3.2)), which consists of strings that could be part of $T$, though including them may risk overshooting. Intuitively, $\widetilde{\mathcal{I}}^{(n)}$ is either the current identified intersection $\mathcal{I}^{(n)}$ or a controlled fallback superset from earlier rounds. Furthermore, in each round, a **pod** $P_n$ is determined. A pod $P_n$ is a set of strings that have priority to be output. The pod sizes are fixed as a growing sequence, for example $(s_n)_{n\geq 1}$ with $s_n = n$. The algorithm maintains a set $U_n$ of strings that are *used*, meaning they have been revealed by the adversary, already output, or already placed into previous pods:

$$U_n \stackrel{\text{def}}{=} A_n \cup G_{n-1} \cup \bigcup_{\nu<n} P_\nu. \tag{110}$$

The **pod** $P_n$ is formed by the $s_n$ $\prec$-smallest elements of $\widetilde{\mathcal{I}}^{(n)} \setminus U_n$. Furthermore, the algorithm maintains a **pod pool** $\mathcal{P}_n$ of available pod elements waiting to be output. The algorithm outputs the smallest element in $\mathcal{P}_n \setminus (A_n \cup G_{n-1})$. By construction, the generated exhaustion $\boldsymbol{G}$ satisfies the novelty constraint.

**Algorithm 3** UPDATECHAIN($\Delta A_n$, $(S_i^{(n-1)})_{i\geq 1}$), **Input:** current adversarial increment $\Delta A = A_n \setminus A_{n-1}$, previous consistent list $(S_i^{(n-1)})_{i\geq 1}$ **Ensures:** updated consistent list $(S_i^{(n)})_{i\geq 1}$, intersection chain $C_n = (\mathcal{I}_i^{(n)})_{i\geq 1}$

$t \leftarrow 0$
**for** $i = 1, 2, 3, \ldots$ **do**
  **if** $\Delta A_n \subseteq S_i^{(n-1)}$ **then**
    $t \leftarrow t + 1$
    $S_t^{(n)} \leftarrow S_i^{(n-1)}$
    **if** $t = 1$ **then**
      $\mathcal{I}_1^{(n)} \leftarrow S_1^{(n)}$
    **else**
      $\mathcal{I}_t^{(n)} \leftarrow \mathcal{I}_{t-1}^{(n)} \cap S_t^{(n)}$
    **end if**
  **end if**
**end for**
**return** $\big((S_i^{(n)})_{i\geq 1}, (\mathcal{I}_i^{(n)})_{i\geq 1}\big)$

**Algorithm 4** IDINTERSECT($n, k_{n-1}, \mathcal{I}^{(n-1)}, C_{n-1}, C_n$) **Input:** timestep $n$, identified intersection $\mathcal{I}^{(n-1)}$, level index $k_{n-1}$, descending chains $C_{n-1}$ and $C_n$ **Ensures:** updated identified intersection $\mathcal{I}^{(n)}$ and updated level index $k_n$.

**if** $n = 1$ **then**
  **return** $(\mathcal{I}_1^{(1)}, 1)$
**end if**
$k \leftarrow k_{n-1}$
**if** $C_{n-1} = C_n$ **then**
  **if** $|\mathcal{I}_{k+1}^{(n-1)}| = \infty$ **then**
    **return** $(\mathcal{I}^{(n)} \leftarrow \mathcal{I}_{k+1}^{(n-1)}, k_n \leftarrow k + 1)$ {move down one level}
  **else**
    **return** $(\mathcal{I}^{(n)} \leftarrow \mathcal{I}_k^{(n-1)}, k_n \leftarrow k)$ {stay if next set becomes finite}
  **end if**
**else**
  $k^\star \leftarrow 0$
  **for** $j = 1, 2, 3, \ldots$ **do**
    **if** $\mathcal{I}_j^{(n-1)} = \mathcal{I}_j^{(n)}$ **and** $|\mathcal{I}_j^{(n-1)}| = \infty$ **then**
      $k^\star \leftarrow j$
    **else**
      **break**
    **end if**
  **end for**
  **if** $k^\star \geq 1$ **then**
    **return** $(\mathcal{I}^{(n)} \leftarrow \mathcal{I}_{k^\star}^{(n-1)}, k_n \leftarrow k^\star)$ {largest common infinite prefix}
  **else**
    **return** $(\mathcal{I}^{(n)} \leftarrow \mathcal{I}_1^{(n)}, k_n \leftarrow 1)$ {no common infinite prefix}
  **end if**
**end if**

# G Recall bounds in a batched adversary setting

## G.1 Lower recall bound

**Lemma 4.5.** *Assume a $k$-batched adversarial setting, with $k \geq 2$. For any $T \in \mathcal{T}$ and any $k$-batched exhaustion $\boldsymbol{A}$ of $A \subseteq T$ with lower recall $\widehat{\rho}_*(\boldsymbol{T}, A)$, the exhaustion $\boldsymbol{G}$ generated by applying*

**Algorithm 5** AGGRSET$(n, \mathcal{I}^{(n-1)}, \mathcal{I}^{(n)}, C_{n-1}, C_n)$ **Input:** round $n$, identified intersection $\mathcal{I}^{(n-1)}$ and $\mathcal{I}^{(n)}$, descending chains $C_{n-1}$ and $C_n$ **Ensures:** Aggressive set $\widetilde{\mathcal{I}}^{(n)}$.

**if** $n = 1$ **then**
  **return** $\mathcal{I}^{(n)}$
**end if**
**if** $\mathcal{I}^{(n)} = \mathcal{I}^{(n-1)}$ **then**
  **return** $\mathcal{I}^{(n)}$ {no fallback}
**else if** $\mathcal{I}^{(n)} \subsetneq \mathcal{I}^{(n-1)}$ **then**
  **return** $\mathcal{I}^{(n-1)}$ {aggressively guess the previous larger set}
**else if** $\mathcal{I}^{(n)} \supsetneq \mathcal{I}^{(n-1)}$ **then**
  **return** $\mathcal{I}^{(n)}$ {aggressively guess the new larger set}
**else**
  {incomparable case: search for a common predecessor in both chains}
  $\mathcal{S}_{n-1} \leftarrow \{J \in C_{n-1} \colon J \supseteq \mathcal{I}^{(n-1)}\}$
  $\mathcal{S}_n \leftarrow \{J \in C_n : J \supseteq \mathcal{I}^{(n)}\}$
  $\mathcal{S} \leftarrow \mathcal{S}_{n-1} \cap \mathcal{S}_n$
  **if** $\mathcal{S} = \emptyset$ **then**
    **return** $\mathcal{I}^{(n)}$ {no fallback possible}
  **else**
    **return** a $\subseteq$-minimal element of $\mathcal{S}$
  **end if**
**end if**

*Algorithm 2 satisfies the novelty constraint and achieves*

$$\widehat{\rho}_*(\boldsymbol{T}, G) \geq \frac{1}{k+1}\widehat{\rho}_*(\boldsymbol{T}, A), \tag{27a}$$

$$\widehat{\pi}_*(T, \boldsymbol{G}) = 1, \qquad \widehat{\tau}_*(T, \boldsymbol{G}) = 1. \tag{27b}$$

*Proof.* We use Kleinberg & Wei's (2025b) algorithm as described in Algorithm 2. By construction, $\boldsymbol{G}$ satisfies the novelty constraint, as all outputs come from pod pools $\mathcal{P}_n$, that contain only unseen strings at step $n$. The major difference to the original setting is that in each timestep the adversary reveals $k$ strings, and hence we choose the consistent languages based on $k$ strings instead of just one. The procedure of computing the intersections remains the same. Note that the properties of eventual validity, eventual comparability, and being full infinitely often transfer to this setting. The proof is analogous to the proof of Kleinberg & Wei (2025b, Lemma 2.5). The procedure of choosing the aggressive sets and pods remains also the same. Hence, our proof follows the structure of Kleinberg & Wei (2025b, Thm. 3.5) closely and differs only in certain counting steps.

**Tail precision and precision.** After finitely many timesteps $N$ the identified intersection $\mathcal{I}^{(n)}$ and the aggressive set $\widetilde{\mathcal{I}}^{(n)}$ remain valid (see Kleinberg & Wei (2025b, Lemma 2.5) and Algorithm 5). Hence, up to that point only finitely many wrong strings can be added to $\mathcal{P}_N$ and afterwards only strings from $T$ are added. This immediately implies

$$t_n = \frac{|\Delta G_n \cap T|}{|\Delta G_n|} = 1 \qquad n \geq N \tag{111}$$

and thus $\widehat{\tau}_*(T, \boldsymbol{G}) = 1$ and $\widehat{\pi}_*(T, \boldsymbol{G}) = 1$ by Lemma D.1.

**Recall.** Denote by $\mathcal{P} = \bigcup_{n \geq 1} P_n$ the set of all strings that ever appear in a pod. We define a partition into a good and bad set as in Kleinberg & Wei (2025b, Thm. 3.5):

$$B_g := \{\boldsymbol{x} \in A \setminus (\mathcal{P} \cup G) : \forall \boldsymbol{y} \in P_{\nu(\boldsymbol{x})-1},\ \boldsymbol{y} \preceq \mathrm{succ}_T(\boldsymbol{x})\}, \tag{112}$$

$$B_b := (A \setminus (\mathcal{P} \cup G)) \setminus B_g, \tag{113}$$

where $\nu(\boldsymbol{x})$ is the first round in which $\boldsymbol{x}$ appears among $A_{\nu(\boldsymbol{x})}$.

Let $N_1$ be the time from which on all identified intersections are valid. We let $N$ be sufficiently large such that at $N$ we have already observed a full intersection twice since $N_1$.

**Bounding $B_b$ and $B_g$.** We fix an $s$. By Lemma G.1, there exists a map $\rho$ from bad strings to pod indices such that $P_{\rho(\boldsymbol{x})}$ contains at least $s$ elements, all $\prec \boldsymbol{x}$, and each index has at most $k$ preimages. This implies

$$|B_b \cap T_n| \ \leq k\big(\frac{1}{s}\,|\mathcal{P} \cap T_n| \ + \mathcal{O}(N) + \mathcal{O}(s)\big). \tag{114}$$

Furthermore, we can bound $|B_g \cap T_n|$ by mapping each $\boldsymbol{x} \in B_g$ to the pod output at time $\nu(\boldsymbol{x}) - 1$. Hence, at most $k$ strings get mapped to the same pod. For a string $\boldsymbol{x}$ with $\nu(\boldsymbol{x}) \geq s$, each previous pod contains at least $s$ elements, which are $\prec \boldsymbol{x}$ by definition of $B_g$. Hence,

$$|B_g \cap T_n| \leq k\big(\frac{1}{s}|\mathcal{P} \cap T_n| + \mathcal{O}(N) + \mathcal{O}(s)\big) \tag{115}$$

**Charging pod elements to outputs.** The only further major difference appears in Kleinberg & Wei's (2025b) charging argument (their Eq. (14)). In our $k$-batched setting, at most $k$ new adversary elements can enter the pool per round, hence

$$|(\mathcal{P} \cap (A \setminus G)) \cap T_n| \ \leq \ k\,|\mathcal{P} \cap G \cap T_n| = k\,|G \cap T_n|, \tag{116}$$

as $G \subseteq \mathcal{P}$.

**Combining bounds.** Every $\boldsymbol{x} \in A$ lies in exactly one of

$$(\mathcal{P} \cap (A \setminus G)), \qquad (\mathcal{P} \cap G) \cap A, \qquad B_b, \qquad B_g. \tag{117}$$

Intersecting with $T_n$ and taking cardinalities yields

$$|A \cap T_n| = |(A \cap G) \cap T_n| + |(\mathcal{P} \cap (A \setminus G)) \cap T_n| + |B_b \cap T_n| + |B_g \cap T_n|.$$

The first term is at most $|G \cap T_n|$. The second term is bounded by $k|G \cap T_n|$ by Eq. (116). The last two terms are bounded using Eq. (114) and Eq. (115). Altogether,

$$|A \cap T_n| \ \leq \ (k+1)|G \cap T_n| + \frac{2k}{s}|\mathcal{P} \cap T_n| + \mathcal{O}(kN) + \mathcal{O}(ks) \tag{118}$$

**Bounding pool mass by outputs.** We next bound $|\mathcal{P} \cap T_n|$ in terms of $|G \cap T_n|$. Using that $G \subseteq \mathcal{P}$ (the algorithm always outputs from the pool), we may decompose

$$\mathcal{P} \cap T_n \subseteq (G \cap T_n) \ \cup \ ((\mathcal{P} \cap (A \setminus G)) \cap T_n). \tag{119}$$

Taking cardinalities and applying Eq. (116) gives

$$|\mathcal{P} \cap T_n| \ \leq \ |G \cap T_n| + |(\mathcal{P} \cap (A \setminus G)) \cap T_n| \ \leq \ (k+1)|G \cap T_n|. \tag{120}$$

Substituting Eq. (120) into Eq. (118) yields

$$|A \cap T_n| \ \leq \ \Big((k+1) + \frac{2k(k+1)}{s}\Big)|G \cap T_n| + \mathcal{O}(kN + ks). \tag{121}$$

Dividing by $n$ and taking $\liminf_{n\to\infty}$, we obtain

$$\liminf_{n\to\infty} \frac{|A \cap T_n|}{n} \leq \ \Big((k+1) + \frac{2k(k+1)}{s}\Big) \liminf_{n\to\infty} \frac{|G \cap T_n|}{n}. \tag{122}$$

Letting $s \to \infty$ gives

$$\widehat{\rho}_*(\boldsymbol{T}, G) \geq \frac{1}{k+1}\widehat{\rho}_*(\boldsymbol{T}, A) \tag{123}$$

which proves the claim. ■

**Lemma G.1** (Batched bad-string pod map)**.** *Fix $s \in \mathbb{N}$ and consider the pods generator algorithm (Algorithm 2) in the $k$-batched setting. There exists a finite time $N$ and a map*

$$\rho : \ \{\boldsymbol{x} \in B_b : \nu(\boldsymbol{x}) > N\} \to \mathbb{N} \tag{124}$$

*such that for every $\boldsymbol{x} \in B_b$ with $\nu(\boldsymbol{x}) > N$:*

1. *all elements of $P_{\rho(\boldsymbol{x})}$ are $\prec \boldsymbol{x}$ and $|P_{\rho(\boldsymbol{x})}| \geq s$;*

2. *(*bounded multiplicity*) each index $n' \in \mathbb{N}$ has at most $k$ preimages: $|\rho^{-1}(n')| \leq k$.*

*Proof.* The proof follows Kleinberg & Wei (2025b, Lem. 3.6). Let $N_1$ be the time from which on all identified intersections are valid. We let $N$ be sufficiently large such that at $N$ we have already observed a full intersection twice since $N_1$. Furthermore, we assume $n \geq s$ for the rest of the proof. Fix $\boldsymbol{x} \in B_b$ with appearance time $\nu(\boldsymbol{x})$, yielding $\boldsymbol{x} \in \mathcal{I}^{(\nu(\boldsymbol{x}))}$. $\boldsymbol{x} \in B_b$ means that $\boldsymbol{x} \notin \mathcal{P} \cup G$ and hence in no previous pod, however, a pod element beyond $\mathrm{succ}_T(\boldsymbol{x})$ is in $P_{\nu(\boldsymbol{x})-1}$. Hence, $\boldsymbol{x} \notin \mathcal{I}^{(\nu(\boldsymbol{x})-1)}$ and $\mathcal{I}^{(\nu(\boldsymbol{x})-1)}$ is inconsistent at round $\nu(\boldsymbol{x})$ and the identified intersection must make an upward move to a *superset* within the prior chain, exactly as in Kleinberg & Wei (2025b, Lem. 3.6). Consequently, there exists a time $n < \nu(\boldsymbol{x})$ such that the identified intersections at times $n$ and $\nu(\boldsymbol{x})$ coincide, and we define $\rho(\boldsymbol{x}) \coloneqq n$. If there are multiple previous times at which the identified intersections coincide with the one at $\nu(\boldsymbol{x})$, we choose the largest such index $n$.

Consider the pod $P_n$ created at time $n$. By construction, $P_n$ contains the $s_n$ smallest unused strings in the aggressive set, which is a superset of the corresponding identified intersection. Because $\boldsymbol{x}$ first appears at time $\nu(\boldsymbol{x})$ and $\boldsymbol{x} \notin \mathcal{P} \cup G$, it was unused at time $n$; however, it appeared in the identified intersection and the aggressive set at that time. Thus every element added to $P_n$ must be $\prec \boldsymbol{x}$, and since $n \geq s$ for all sufficiently large times, we also have $|P_n| = s_n \geq s$.

Finally, the only difference from Kleinberg & Wei (2025b, Lem. 3.6) is that $\rho$ is no longer injective: in the $k$-batched model, at most $k$ new strings can arrive in a single round, possibly triggering one upward move together. Each string $\boldsymbol{x} \in B_b$ from that round yields the same $\rho(\boldsymbol{x})$. However, strings arriving in later rounds will be mapped to another index. Hence, $|\rho^{-1}(n)| \leq k$ for all $n$. ■

### G.2 Upper recall bound

**Lemma 4.6.** *Assume a $k$-batched adversarial setting. There exist countable collections of languages $\mathcal{T}$, such that for a target $T$ and an adversarial exhaustion $\boldsymbol{A}$ with $A_n \uparrow A$, and $A \subseteq T$, the recall of any novel generator that satisfies $\widehat{\tau}_*(T, \boldsymbol{G}) = 1$ is bounded by*

$$\widehat{\rho}_*(\boldsymbol{T}, G) \leq \frac{\widehat{\rho}_*(\boldsymbol{T}, A)}{k+1}. \tag{28}$$

*Proof.* Assume any countable collection of languages $\mathcal{T}$. Assume a single-step novel generator $\mathcal{G}$ that satisfies $\widehat{\tau}_*(T, \boldsymbol{G}) = 1$. Let $\boldsymbol{T} = \{T_n\}_{n=0}^{\infty}$ be an enumeration of $T$.

**Full exhaustion.** Assume first that the adversary fully reveals $T$, i.e., $A_n \uparrow T$ and hence $\widehat{\rho}_*(\boldsymbol{T}, A) = 1$. We define an adversary exhaustion that forces

$$\widehat{\rho}_*(\boldsymbol{T}, G) \leq \frac{1}{k+1}. \tag{125}$$

Let $R \stackrel{\text{def}}{=} \{2^m : m \in \mathbb{N}_0\}$ be the set of so-called repetition steps and define $r(n) \stackrel{\text{def}}{=} |R \cap \{1, \ldots, n\}|$. At each timestep $n \in \mathbb{N}$, the adversary reveals a set of at most $k$ new strings as follows. If $n \notin R$, it reveals the $k$ smallest strings in $T$ (with respect to the ordering induced by the enumeration $\boldsymbol{T}$) that have not yet been used by either player, or more formally, it outputs the unused strings from $T_{i_n}$, where $i_n$ is the smallest index such that $|T_{i_n} \setminus (A_{n-1} \cup G_{n-1})| = k$ . If $n \in R$, it reveals the $(k-1)$ smallest such unused strings and, in addition, the smallest string in $T$ that has previously been output by $\mathcal{G}$ but has not yet been revealed by the adversary (if such a string exists). This exhaustion is $f$-bounded with $f(n) = k$ and satisfies $|\Delta A_n| \leq k$ by construction. Moreover, by construction, the adversary outputs all strings in $G \cap T$, and hence $A_n \uparrow T$.

Now consider the first $n$ interaction steps. The generator outputs at most $n$ strings in total. On each non-repetition step $n' \notin R$ with $n' \leq n$ the adversary reveals the $k$ smallest unused strings, so apart from the $r(n)$ repetition steps, each block of $k+1$ newly used strings can contain at most one generator output. Consequently, among the first $(k+1)n$ strings in $T_{(k+1)n}$, the generator can cover at most $n + r(n)$ of them:

$$\left|G \cap T_{(k+1)n}\right| \leq n + r(n). \tag{126}$$

Divide by $(k+1)n$ and take $\liminf_{n\to\infty}$. Since $r(n) = \mathcal{O}(\log n)$ for powers of two, we have $r(n)/n \to 0$, and thus

$$\liminf_{n\to\infty} \frac{|G \cap T_n|}{|T_n|} \leq \frac{1}{k+1}, \tag{127}$$

which is exactly Eq. (125).

**Partial exhaustion.** Now let $A \subseteq T$ be arbitrary, and let the adversary reveal only strings from $A$ using the same strategy as above, with smallest string interpreted with respect to the order induced by $\boldsymbol{T}$ restricted to $A$. Consider a collection $\mathcal{T}$ that contains both $A$ and $T$. Crucially, the revealed exhaustion is compatible with either choice of true language in $\{A, T\}$. Since the generator's behavior depends only on the revealed history, it produces the same output stream in both cases. Therefore, because the generator must eventually reach tail precision 1 when the true language is $A$, it can output only finitely many strings outside $A$. Hence, $\big|G \cap (T_n \setminus A)\big| = \mathcal{O}(1)$. We apply the same counting argument as in the full case:

$$|G \cap T_n| = |G \cap (A \cap T_n)| + \mathcal{O}(1) \leq \frac{|A \cap T_n|}{k+1} + \mathcal{O}(\log |A \cap T_n|) \tag{128}$$

Dividing by $|T_n|$ and taking $\liminf_{n\to\infty}$, the second term vanishes and thus

$$\widehat{\rho}_*(\boldsymbol{T}, G) = \liminf_{n\to\infty} \frac{|G \cap T_n|}{|T_n|} \leq \frac{1}{k+1} \liminf_{n\to\infty} \frac{|A \cap T_n|}{|T_n|} = \frac{1}{k+1}\, \widehat{\rho}_*(\boldsymbol{T}, A). \tag{129}$$

■

# H Generation with novelty

## H.1 Generation with novelty but without perfect tail precision

**Theorem 4.8.** *For any $T \in \mathcal{T}$ and any single-step adversarial exhaustion $\boldsymbol{A}$ with lower recall $\widehat{\rho}_*(\boldsymbol{T}, A)$ and upper recall $\widehat{\rho}^*(\boldsymbol{T}, A)$, Algorithm 6 generating $\boldsymbol{G} = \{G_n\}_{n=0}^{\infty}$ with $G_n \uparrow G$ achieves precision 1 and recall*

$$\widehat{\rho}_*(\boldsymbol{T}, G) \geq \max\Big(1 - \widehat{\rho}^*(\boldsymbol{T}, A), \frac{\widehat{\rho}_*(\boldsymbol{T}, A)}{3}\Big). \tag{32}$$

*Proof.* Fix an arbitrary enumeration $\boldsymbol{E} = \{E_n\}_{n=1}^{\infty}$ of $\Sigma^*$. Choose an exploration set $R \subset \mathbb{N}$ according to Definition 4.1 and let $r(m) \stackrel{\text{def}}{=} |R \cap \{1, \ldots, m\}|$. We will schedule exploration rounds on each $r \in R$. We construct $\boldsymbol{G} = \{G_n\}_{n=0}^{\infty}$ iteratively as described in Algorithm 6. By construction, $G_n$ increases by 1 in each round, as the outputs are novel. Moreover, exploration happens only whenever the current timestep reaches the next value in $R$. This will ensure that for $G_n$ with $|G_n| = n$, the number of exploratory strings is at most $r(n)$. We first prove the recall bound.

**The $1 - \widehat{\rho}^*(\boldsymbol{T}, A)$ bound.** We show that Algorithm 6 generates every string in $T \setminus A$. Fix any $\boldsymbol{x} \in T \setminus A$, and let $l$ be such that $\Delta E_l = \{\boldsymbol{x}\}$. Since $\boldsymbol{x} \notin A$, the adversary never outputs $\boldsymbol{x}$, so $\boldsymbol{x} \notin A_m$ at all times $m$. On exploration rounds, the algorithm outputs the first $\Delta E_j$ not in $G_{m-1} \cup A_m$ and advances $j$. Exploration rounds occur infinitely often and $\boldsymbol{x}$ cannot be made unavailable by $A$, as it does not occur in any $A_m$. Hence, $\boldsymbol{x}$ is either produced in a safe round or, if not, the pointer $j$ reaches $l$ eventually, and the algorithm outputs $\boldsymbol{x}$ in an exploration round. Thus $G$ contains $T \setminus A$.

Applying Lemma 4.7 yields

$$\widehat{\rho}_*(\boldsymbol{T}, G) \;\geq\; 1 - \widehat{\rho}^*(\boldsymbol{T}, A). \tag{130}$$

**The $\frac{\widehat{\rho}_*(\boldsymbol{T}, A)}{3}$ bound.** Consider only the safe rounds. Since $R$ contains no consecutive integers, at most two adversarial strings are revealed between consecutive safe rounds. Thus, the safe subroutine sees a 2-batched adversary.

Let $\mathcal{P}$ be the union of all pods created during safe rounds. Note that $G$ contain both safe and exploration outputs. We now closely follow the charging argument from Lemma 4.5 and adapt it to this setting.

First, every element of $\mathcal{P}$ is eventually either generated or revealed by the adversary. Consequently,

$$\mathcal{P} \subseteq A \cup G. \tag{131}$$

Next, consider an element $\boldsymbol{x} \in \mathcal{P} \cap (A \setminus G)$. At the safe round immediately preceding the first safe round at which $\boldsymbol{x}$ is observed, $\boldsymbol{x}$ was still available in the pod pool. Hence, the safe output at that round precedes $\boldsymbol{x}$ under $\prec$. Since at most two adversarial strings are newly observed per safe round, each safe output receives at most two such charges. Using the common ordering of $T_n$ and $\prec$, and discarding finitely many early invalid outputs, we get

$$|(\mathcal{P} \cap (A \setminus G)) \cap T_n| \leq 2|G \cap T_n| + O(1). \tag{132}$$

Together with $\mathcal{P} \subseteq A \cup G$, this implies

$$|\mathcal{P} \cap T_n| \leq 3|G \cap T_n| + O(1). \tag{133}$$

It remains to bound adversarial strings that are neither in $\mathcal{P}$ nor in $G$. As in Lemma 4.5, partition these strings into $B_g$ and $B_b$. The pod-mapping argument continues to apply: the identified intersection chain evolves only on safe rounds, with at most two new adversarial strings per update. Exploration outputs merely remove strings from the unused set; any string in $B_g \cup B_b$ remains unused throughout.

Fix $s \geq 2$. Since all sufficiently late pods have size at least $s$, the same bounded-multiplicity arguments yield

$$|B_b \cap T_n| \leq \frac{2}{s}|\mathcal{P} \cap T_n| + O_s(1) \tag{134}$$

and

$$|B_g \cap T_n| \leq \frac{2}{s-1}|\mathcal{P} \cap T_n| + O_s(1). \tag{135}$$

Partitioning $A$ into generated strings, stolen pod elements, and the sets $B_g, B_b$, we obtain

$$|A \cap T_n| \leq \left(3 + \frac{6}{s} + \frac{6}{s-1}\right) |G \cap T_n| + O_s(1). \tag{136}$$

Dividing by $n$, taking $\liminf_{n\to\infty}$, and then letting $s \to \infty$ proves

$$\widehat{\rho}_*(\boldsymbol{T}, G) \geq \frac{\widehat{\rho}_*(\boldsymbol{T}, A)}{3}. \tag{137}$$

Combining both bounds gives

$$\widehat{\rho}_*(\boldsymbol{T}, G) \geq \max\left(1 - \widehat{\rho}^*(\boldsymbol{T}, A), \frac{\widehat{\rho}_*(\boldsymbol{T}, A)}{3}\right). \tag{138}$$

Note that we cannot simply add up the bounds. This is because Kleinberg & Wei's (2025b) routine guarantees generation from $T$, not from $A$, and might hence generate solely from $T \setminus A$.

**Precision.** As all outputs are novel, $|G_n| = n$. For each $G_n$, the number of outputs from explorations (and hence possible hallucinations) is at most $r(n)$; furthermore, Kleinberg & Wei's (2025b) algorithm guarantees stabilization of tail precision to 1 in finite time, meaning that all outputs from safe rounds except finitely many lie in $T$. Hence

$$|G_n \cap T| \geq n - r(n) - \mathcal{O}(1) \tag{139}$$

and therefore

$$\widehat{\pi}_*(T, \boldsymbol{G}) = \liminf_{n\to\infty} \frac{|G_n \cap T|}{|G_n|} \tag{140}$$

$$\geq \liminf_{n\to\infty} \left(1 - \frac{r(n) + \mathcal{O}(1)}{n}\right) \tag{141}$$

$$= 1. \tag{142}$$

■

# I Generation with $\gamma$-novelty

**Lemma I.1.** *There exists a $\gamma$-admissible exploration set (Definition 4.2) for every $\gamma < 1$.*

*Proof.* Let $M \geq \max\{2, \lceil 1/(1-\gamma)\rceil\}$ and set $R = \{M2^i : i \in \mathbb{N}_0\}$. Then $R$ is infinite, has no consecutive elements and satisfies $r(m)/m \to 0$. Furthermore,

$$r(m) \leq \frac{m}{M} \leq (1-\gamma)m. \tag{143}$$

■

## I.1 Generation with $\gamma$-novelty without perfect tail precision

**Theorem 4.9.** *Fix any $\gamma \in [0, 1)$. For any $T \in \mathcal{T}$ and any single-step adversarial exhaustion $\boldsymbol{A}$ with $A_n \uparrow A \subseteq T$, Algorithm 7 generates a $\gamma$-novel exhaustion $\boldsymbol{G} = \{G_n\}_{n=0}^{\infty}$ with $G_n \uparrow G$ such that*

$$\widehat{\rho}_*(\boldsymbol{T}, G) = 1, \qquad \widehat{\pi}_*(T, \boldsymbol{G}) = 1. \tag{33}$$

*Proof.* Algorithm 7 combines safe rounds with exploration rounds scheduled according to a $\gamma$-admissible exploration set. Since exploration rounds are not consecutive and the safe routine is not advanced during exploration rounds, the safe routine sees a 2-batched adversary. The exploration rounds output elements from an enumeration $\boldsymbol{E}$ of $\Sigma^*$. In those rounds, we allow the generator to output strings that have been previously revealed by the adversary while it skips strings that already belong to $G_n$. Thus, $|G_n| = n$. An exploration round either does not influence the safe rounds by outputting a string that the safe rounds would never produce or makes the safe routine advance more quickly. The proof of precision is analogous to Theorem 4.8. It remains to show that $\gamma$-novelty is satisfied and that the recall is 1.

**Algorithm 6** Language generation in the limit with novelty and without perfect tail precision. UPDATECHAIN takes the adversary's input and a sequence of languages and returns an updated sequence of languages and a chain of intersections of consistent languages; IDINTERSECT takes the timestep and a level index, two intersection chains and an identified intersection and returns the next level index and the next identified intersection; AGGRSET takes two identified intersections and two intersection chains and returns an aggressive set, i.e., a superset of the current identified intersection. **Input:** adversarial exhaustion $\{A_n\}_{n=0}^{\infty}$ with $A_n \uparrow A \subseteq T$, enumeration $\boldsymbol{E}$ of $\Sigma^*$, infinite exploration set $R \subset \mathbb{N}$, pod size sequence $(s_n)_{n\geq 1}$, ordered sequence $(S'_i)_{i\geq 1}$ of languages in $\mathcal{T}$ **Ensures:** Increasing exhaustion $\boldsymbol{G} = \{G_n\}_{n=0}^{\infty}$.

$G_0 \leftarrow \emptyset$, $n \leftarrow 1$, $j \leftarrow 1$, $U_0 \leftarrow \emptyset$ {unavailable strings}, $\mathcal{P}_0 \leftarrow \emptyset$ {pod pool}, $(S_i^{(0)})_{i\geq 1} \leftarrow (S'_i)_{i\geq 1}$ {consistent languages}, $k_0 \leftarrow 1$, $A_{\text{prev}} \leftarrow \emptyset$, $\mathcal{I}^{(0)} \leftarrow \emptyset$, $C_0 \leftarrow \emptyset$
$r_{\text{next}} \leftarrow \min(R)$
**for** $m = 1, 2, 3, \ldots$ **do**
  Observe $A_m$
  **if** $m \neq r_{\text{next}}$ **then**
    $\big((S_i^{(n)})_{i\geq 1}, C_n\big) \leftarrow \text{UPDATECHAIN}\big(A_m \setminus A_{\text{prev}}, (S_i^{(n-1)})_{i\geq 1}\big)$
    $(\mathcal{I}^{(n)}, k_n) \leftarrow \text{IDINTERSECT}(n, k_{n-1}, \mathcal{I}^{(n-1)}, C_{n-1}, C_n)$
    $\widetilde{\mathcal{I}}^{(n)} \leftarrow \text{AGGRSET}(n, \mathcal{I}^{(n-1)}, \mathcal{I}^{(n)}, C_{n-1}, C_n)$
    $U_n \leftarrow U_{n-1} \cup A_m \cup G_{m-1}$.
    $P_n \leftarrow$ the $s_n$ $\prec$-smallest elements of $\widetilde{\mathcal{I}}^{(n)} \setminus U_n$
    $U_n \leftarrow U_n \cup P_n$
    $\mathcal{P}_n \leftarrow \mathcal{P}_{n-1} \cup P_n$.
    $G_m \leftarrow G_{m-1} \cup \min_{\prec}(\mathcal{P}_n \setminus (A_m \cup G_{m-1}))$
    $U_n \leftarrow U_n \cup G_m$
    $A_{\text{prev}} \leftarrow A_m$
    $n \leftarrow n + 1$ ▷ safe rounds
  **else**
    **while** $E_j \subseteq A_m \cup G_{m-1}$ **do**
      $j \leftarrow j + 1$
    **end while**
    $G_m \leftarrow G_{m-1} \cup \Delta E_j$
    $j \leftarrow j + 1$
    $r_{\text{next}} \leftarrow \min(\{r \in R\colon r > r_{\text{next}}\})$ ▷ exploration
  **end if**
**end for**

**$\gamma$-novelty.** The algorithm chooses a $\gamma$-admissible exploration set $R$. All safe rounds produce novel outputs, i.e., outputs that are chosen outside $A_n \cup G_{n-1}$. Therefore, non-novel outputs can occur only on exploration rounds. Let $N_n$ be the cumulative novel set from Definition 3.7. For every timestep $n$, by Definition 4.2

$$|G_n \setminus N_n| \leq r(n) \leq (1-\gamma)n. \tag{144}$$

Combined with $|G_n| = n$, it follows that

$$\frac{|N_n|}{|G_n|} = 1 - \frac{|G_n \setminus N_n|}{|G_n|} \geq \gamma. \tag{145}$$

Thus, the generator is $\gamma$-novel.

**Recall.** Let $\boldsymbol{E} = \{E_j\}_{j=1}^{\infty}$ be the fixed enumeration of $\Sigma^*$. In each exploration round, the algorithm outputs the first element $\Delta E_j$ such that $\Delta E_j \nsubseteq G_{n-1}$. Since $R$ is infinite, this exploration routine is invoked infinitely many times. We show that every element of $\Sigma^*$ is eventually generated. Fix $m \in \mathbb{N}$. For $m = 1$, either $\Delta E_1$ is output in the first exploration round, or it has already been generated by a safe round. In either case, $\Delta E_1 \in G$.

Now assume that, for some $m > 1$, all elements in $E_{m^\star}$ for every $m^\star < m$ have already been added to $G$. Consider the next exploration round. When the exploration routine scans the enumeration $\boldsymbol{E}$, it

**Algorithm 7** Language generation in the limit with $\gamma$-novelty and without perfect tail precision. UPDATECHAIN takes the adversary's input and a sequence of languages and returns an updated sequence of languages and a chain of intersections of consistent languages; IDINTERSECT takes the timestep and a level index, two intersection chains and an identified intersection and returns the next level index and the next identified intersection; AGGRSET takes two identified intersections and two intersection chains and returns an aggressive set, i.e., a superset of the current identified intersection. **Input:** adversarial exhaustion $\{A_n\}_{n=0}^{\infty}$ with $A_n \uparrow A \subseteq T$, enumeration $\boldsymbol{E}$ of $\Sigma^*$, infinite $\gamma$-admissible exploration set $R \subset \mathbb{N}$, pod size sequence $(s_n)_{n\geq 1}$, ordered sequence $(S'_i)_{i\geq 1}$ of languages in $\mathcal{T}$ **Ensures:** Increasing exhaustion $\boldsymbol{G} = \{G_n\}_{n=0}^{\infty}$.

$G_0 \leftarrow \emptyset$, $n \leftarrow 1$, $j \leftarrow 1$, $U_0 \leftarrow \emptyset$ {unavailable strings}, $\mathcal{P}_0 \leftarrow \emptyset$ {pod pool}, $(S_i^{(0)})_{i\geq 1} \leftarrow (S'_i)_{i\geq 1}$ {consistent languages}, $k_0 \leftarrow 1$, $A_{\text{prev}} \leftarrow \emptyset$, $\mathcal{I}^{(0)} \leftarrow \emptyset$, $C_0 \leftarrow \emptyset$
$r_{\text{next}} \leftarrow \min(R)$
**for** $m = 1, 2, 3, \ldots$ **do**
  Observe $A_m$
  **if** $m \neq r_{\text{next}}$ **then**
    $\big((S_i^{(n)})_{i\geq 1}, C_n\big) \leftarrow \text{UPDATECHAIN}\big(A_m \setminus A_{\text{prev}}, (S_i^{(n-1)})_{i\geq 1}\big)$
    $(\mathcal{I}^{(n)}, k_n) \leftarrow \text{IDINTERSECT}(n, k_{n-1}, \mathcal{I}^{(n-1)}, C_{n-1}, C_n)$
    $\widetilde{\mathcal{I}}^{(n)} \leftarrow \text{AGGRSET}(n, \mathcal{I}^{(n-1)}, \mathcal{I}^{(n)}, C_{n-1}, C_n)$
    $U_n \leftarrow U_{n-1} \cup A_m \cup G_{m-1}$.
    $P_n \leftarrow$ the $s_n$ $\prec$-smallest elements of $\widetilde{\mathcal{I}}^{(n)} \setminus U_n$
    $U_n \leftarrow U_n \cup P_n$
    $\mathcal{P}_n \leftarrow \mathcal{P}_{n-1} \cup P_n$.
    $G_m \leftarrow G_{m-1} \cup \min_{\prec}(\mathcal{P}_n \setminus (A_m \cup G_{m-1}))$
    $U_n \leftarrow U_n \cup G_m$
    $A_{\text{prev}} \leftarrow A_m$
    $n \leftarrow n + 1$ ▷ safe rounds
  **else**
    **while** $E_j \subseteq G_{m-1}$ **do**
      $j \leftarrow j + 1$
    **end while**
    $G_m \leftarrow G_{m-1} \cup \Delta E_j$
    $j \leftarrow j + 1$
    $r_{\text{next}} \leftarrow \min(\{r \in R\colon r > r_{\text{next}}\})$ ▷ exploration
  **end if**
**end for**

skips all $E_{m^\star}$ with $m^\star < m$, since they already belong to the current guessed language. Hence, the routine reaches $E_m$. If $\Delta E_m \nsubseteq G_{n-1}$, it is output in this exploration round. If it is not output in this exploration round, then this can only be because $\Delta E_m$ has been output in a previous safe round and thus $\Delta E_m \subseteq G_{n-1}$ already. In either case, $\Delta E_m \subseteq G$.

By induction, every element of $\Sigma^*$ is eventually contained in $G$, and therefore $\Sigma^* \subseteq G$. Since $T \subseteq \Sigma^*$, it follows that $T \subseteq G$. Therefore,

$$\widehat{\rho}_*(\boldsymbol{T}, G) = \liminf_{n\to\infty} \frac{|G \cap T_n|}{|T_n|} = 1. \tag{146}$$

■

**Corollary I.2.** *Fix any $\gamma \in [0, 1)$. For any $T \in \mathcal{T}$ and any single-step full adversarial exhaustion $\boldsymbol{A}$ with $A_n \uparrow T$, Algorithm 7 generates a $\gamma$-novel exhaustion $\boldsymbol{G} = \{G_n\}_{n=0}^{\infty}$ with $G_n \uparrow G$ such that*

$$\widehat{\rho}_*(\boldsymbol{T}, G) = 1, \qquad \widehat{\pi}_*(T, \boldsymbol{G}) = 1. \tag{147}$$

### I.2 Generation with $\gamma$-novelty and with perfect tail precision

We consider $\gamma$-novel generation under the requirement of perfect tail precision. In contrast to the strict novelty setting with $\gamma = 1$, a $\gamma$-novel generator for $\gamma \in [0, 1)$ may use a controlled fraction of

**Algorithm 8** Language generation in the limit with $\gamma$-novelty and perfect tail precision. UPDATECHAIN takes the adversary's input and a sequence of languages and returns an updated sequence of languages and a chain of intersections of consistent languages; IDINTERSECT takes the timestep and a level index, two intersection chains and an identified intersection and returns the next level index and the next identified intersection; AGGRSET takes two identified intersections and two intersection chains and returns an aggressive set, i.e., a superset of the current identified intersection. **Input:** adversarial exhaustion $\{A_n\}_{n=0}^{\infty}$ with $A_n \uparrow A \subseteq T$, novelty parameter $\gamma \in [0,1)$, pod size sequence $(s_n)_{n\geq 1}$, ordered sequence $(S'_i)_{i\geq 1}$ of languages in $\mathcal{T}$. **Ensures:** increasing exhaustion $\boldsymbol{G} = \{G_n\}_{n=0}^{\infty}$.

$L \leftarrow \left\lceil \frac{1}{1-\gamma} \right\rceil$
$G_0 \leftarrow \emptyset$, $n \leftarrow 1$, $U_0 \leftarrow \emptyset$ {unavailable strings}, $\mathcal{P}_0 \leftarrow \emptyset$ {pod pool}, $(S_i^{(0)})_{i\geq 1} \leftarrow (S'_i)_{i\geq 1}$ {consistent languages}, $k_0 \leftarrow 1$, $A_{\text{prev}} \leftarrow \emptyset$, $\mathcal{I}^{(0)} \leftarrow \emptyset$, $C_0 \leftarrow \emptyset$
**for** $m = 1, 2, 3, \ldots$ **do**
  Observe $A_m$
  **if** $m \equiv 0 \pmod{L}$ **and** $A_m \setminus G_{m-1} \neq \emptyset$ **then**
    $G_m \leftarrow G_{m-1} \cup \{\min_{\prec}(A_m \setminus G_{m-1})\}$ ▷ repetition round
  **else**
    $\big((S_i^{(n)})_{i\geq 1}, C_n\big) \leftarrow \textsc{UpdateChain}\big(A_m \setminus A_{\text{prev}}, (S_i^{(n-1)})_{i\geq 1}\big)$
    $(\mathcal{I}^{(n)}, k_n) \leftarrow \textsc{IdIntersect}(n, k_{n-1}, \mathcal{I}^{(n-1)}, C_{n-1}, C_n)$
    $\widetilde{\mathcal{I}}^{(n)} \leftarrow \textsc{AggrSet}(n, \mathcal{I}^{(n-1)}, \mathcal{I}^{(n)}, C_{n-1}, C_n)$
    $U_n \leftarrow U_{n-1} \cup A_m \cup G_{m-1}$
    $P_n \leftarrow$ the $s_n$ $\prec$-smallest elements of $\widetilde{\mathcal{I}}^{(n)} \setminus U_n$
    $U_n \leftarrow U_n \cup P_n$
    $\mathcal{P}_n \leftarrow \mathcal{P}_{n-1} \cup P_n$
    $G_m \leftarrow G_{m-1} \cup \{\min_{\prec}(\mathcal{P}_n \setminus (A_m \cup G_{m-1}))\}$
    $U_n \leftarrow U_n \cup G_m$
    $A_{\text{prev}} \leftarrow A_m$
    $n \leftarrow n+1$ ▷ safe round
  **end if**
**end for**

non-novel outputs. We use these non-novel outputs to repeat all the strings revealed by the adversary. In Algorithm 8, we interleave safe rounds with repetition rounds.

**Theorem I.3.** *Let $\mathcal{T}$ be a countable collection of languages and let $T \in \mathcal{T}$ be the target language. Let $\gamma \in [0,1)$ be the required fraction of novel strings. Fix an enumeration $\boldsymbol{T} = \{T_n\}_{n=0}^{\infty}$ of $T$. There exists a $\gamma$-novel generator $\mathcal{G}$ such that, for any single-step adversarial exhaustion $\boldsymbol{A} = \{A_n\}_{n=0}^{\infty}$ with $A_n \uparrow A \subseteq T$ and $\widehat{\rho}_*(\boldsymbol{T}, A) \geq \alpha$, the generated single-step exhaustion $\boldsymbol{G}$ satisfies:*

$$\widehat{\rho}_*(\boldsymbol{T}, G) \geq \alpha, \qquad \widehat{\pi}_*(T, \boldsymbol{G}) = 1, \qquad \widehat{\tau}_*(T, \boldsymbol{G}) = 1. \tag{148}$$

*Moreover, the recall guarantee is tight in an adversarial sense.*

*Proof.* The case $\gamma = 0$ is covered in Theorem E.1. Now let $\gamma \in (0,1)$ and let $L := \left\lceil \frac{1}{1-\gamma} \right\rceil$. We divide time into blocks of length $L$ and run Algorithm 8. In each block, we run safe rounds using the pods algorithm by Kleinberg & Wei (see Algorithm 2) for $L-1$ rounds and take one repetition round at the end of the block. The subroutine for the pods algorithm is updated only on safe rounds, using all adversarial strings revealed since the previous safe round. Since there is at most one repetition round between two consecutive safe rounds, and the adversary is single-step bounded, the safe subroutine sees a 2-batched adversary. On repetition rounds, the generator outputs the smallest adversary-revealed string that has not yet been generated. Formally, on a timestep $m$, a repetition round outputs the $\prec$-smallest element of $A_m \setminus G_{m-1}$, if this set is nonempty. If $A_m \setminus G_{m-1} = \emptyset$, the generator instead takes a safe round. The repetition outputs are always valid because $A \subseteq T$. Furthermore, the repetition rounds do not interfere with the 2-batched safe routine in any way, as they never output unused elements from pods that the safe routine might produce later.

**$\gamma$-novelty.** The safe rounds output only novel strings: the output is chosen outside $A_m \cup G_{m-1}$, i.e., outside the strings already revealed by the adversary or previously generated. Only repetition rounds may be non-novel. Since there is at most one repetition round in each block of length $L$, and since the repetition round occurs at the end of the block, every $G_m$ contains at most one non-novel output per completed block. If $A_m \setminus G_{m-1} = \emptyset$ on a repetition round, the generator takes a safe round instead, which can only increase the fraction of novel outputs.

Let $N_n$ denote the cumulative novel set from Definition 3.7. Consider the set $G_n \setminus N_n$ of non-novel strings generated up to timestep $n$. It holds that

$$|G_n \setminus N_n| \leq \left\lfloor \frac{|G_n|}{L} \right\rfloor \leq \frac{|G_n|}{L}. \tag{149}$$

Therefore, we have

$$\frac{|N_n|}{|G_n|} = 1 - \frac{|G_n \setminus N_n|}{|G_n|} \geq 1 - \frac{1}{L} \geq \gamma, \tag{150}$$

where the last inequality follows from $L \geq 1/(1-\gamma)$. Hence the generator is $\gamma$-novel.

**Tail precision.** All outputs from repetition rounds belong to $A$, and therefore to $T$. On safe rounds, we run the 2-batched Kleinberg & Wei's (2025b) algorithm. It guarantees stabilization of tail precision to 1 in finite time, meaning that all outputs from safe rounds except finitely many lie in $T$. Hence, after some finite timestep, every generated string, whether produced by a safe round or by a repetition round, lies in $T$. Thus

$$\widehat{\tau}_*(T, \boldsymbol{G}) = 1. \tag{151}$$

**Precision.** By Lemma D.1, a tail precision of 1 also implies

$$\widehat{\pi}_*(T, \boldsymbol{G}) = 1. \tag{152}$$

**Recall.** We show that every string revealed by the adversary is eventually generated. Let $\boldsymbol{y} \in A$. There exists a finite timestep $m^\star$ such that $\boldsymbol{y} \in A_m$ for all $m \geq m^\star$. From that time onward, unless $\boldsymbol{y}$ has already been generated by a safe round, it remains in $A_m \setminus G_{m-1}$. The repetition rounds continue to occur as long as $A_m \setminus G_{m-1} \neq \emptyset$. Moreover, since $\prec$ is induced by an enumeration of $\Sigma^*$, each string has only finitely many $\prec$-predecessors. Hence, there are only finitely many elements that can be output in repetition rounds before outputting $\boldsymbol{y}$, meaning that $\boldsymbol{y}$ is generated eventually. Hence, every $\boldsymbol{y} \in A$ is output eventually, which yields

$$\widehat{\rho}_*(\boldsymbol{T}, G) = \liminf_{n\to\infty} \frac{|T_n \cap G|}{|T_n|} \geq \liminf_{n\to\infty} \frac{|T_n \cap A|}{|T_n|} = \widehat{\rho}_*(\boldsymbol{T}, A) \geq \alpha. \tag{153}$$

The tightness of the recall bound in an adversarial sense follows from the same argument as in Theorem E.1. ■

The following corollary describes the special case of a full exhaustion $\boldsymbol{A} = \{A_n\}_{n=0}^{\infty}$ with $A_n \uparrow T$, meaning $\widehat{\rho}_*(\boldsymbol{T}, A) = 1$.

**Corollary I.4.** *Let $\mathcal{T}$ be a countable collection of languages and let $T \in \mathcal{T}$ be the target language. Let $\gamma \in [0, 1)$ be the required fraction of novel strings. Fix an enumeration $\boldsymbol{T} = \{T_n\}_{n=0}^{\infty}$ of $T$. There exists a $\gamma$-novel generator $\mathcal{G}$ such that, for every single-step full adversarial exhaustion $\boldsymbol{A} = \{A_n\}_{n=0}^{\infty}$ with $A_n \uparrow T$, the generated single-step exhaustion $\boldsymbol{G}$ satisfies:*

$$\widehat{\rho}_*(\boldsymbol{T}, G) = 1, \qquad \widehat{\pi}_*(T, \boldsymbol{G}) = 1, \qquad \widehat{\tau}_*(T, \boldsymbol{G}) = 1. \tag{154}$$

## NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We believe that the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discussed how the benefit of hallucinations is specific to the partial enumeration regime. When the adversary shows a large proportion of the target, our results show no benefit from allowing hallucinations.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: For each theoretical result, we provide the full set of assumptions and complete proofs, as well as proof sketches in the main text, where necessary.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [N/A]

   Justification: The paper does not include experiments.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [N/A]

Justification: The paper does not include experiments, therefore it does not use any data and code.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [N/A]

   Justification: The paper does not include experiments.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [N/A]

   Justification: The paper does not include experiments.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)

- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [N/A]

   Justification: The paper does not include experiments.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We believe that the research conducted in this paper conform, in every respect, with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

    Justification: This paper presents theoretical results in the foundations of language generation. The primary contribution is a better formal understanding of the tension between coverage and validity in generative models. We do not anticipate direct negative societal impacts from this work.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.

- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: We did not use any data or models.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [N/A]

    Justification: We did not use any such assets.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.

- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: We do not introduce any new assets.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: We did not do crowdsourcing experiments and research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: We did not do research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: We used LLMs only for writing, editing, and formatting purposes.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.