On Characterizations for Language Generation: Interplay of Hallucinations, Breadth, and Stability

Anonymous Authors[1]

Abstract

We study language generation in the limit – introduced by Kleinberg & Mullainathan (2024) – building on classical works of Gold (1967) and Angluin (1979). (Kleinberg & Mullainathan, 2024)'s main result is an algorithm for generating from any countable language collection in the limit. While their algorithm eventually generates unseen strings from the target language K, it sacrifices coverage or *breadth*, *i.e.*, its ability to generate a rich set of strings. Recent work introduces different notions of breadth and explores when generation with breadth is possible, leaving a full characterization of these notions open. Our first set of results settles this by characterizing generation for existing notions of breadth and their natural extensions. Interestingly, our lower bounds are very flexible and hold for many performance metrics beyond breadth – for instance, showing that, in general, it is impossible to train generators which achieve a higher perplexity or lower hallucination rate for K compared to other languages. Next, we study language generation with breadth and stable generators – algorithms that eventually stop changing after seeing an arbitrary but finite number of strings – and prove unconditional lower bounds for such generators, strengthening the results of Kalavasis et al. (2025) and demonstrating that generation with many existing notions of breadth becomes equally hard, when stability is required. This gives a separation for generation with approximate breadth, between stable and unstable generators, highlighting the rich interplay between breadth, stability, and consistency in language generation.

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on ICML 2025 Workshop on Reliable and Responsible Foundation Models. Do not distribute.

1. Introduction

Language generation has a rich history in computer science, dating back to the seminal work of (Shannon, 1951), culminating in today's Large Language Models (LLMs) that have revolutionized natural language processing and, more broadly, machine learning (ML). Although the problem at the core of generation – generate new and unseen strings given a sequence of examples from a target language K – is easy to state, a theoretical understanding of why LLMs are able to produce coherent text remains elusive. Recently, Kleinberg & Mullainathan (2024) formalized this problem under a simple yet elegant model of *language generation in the limit:* given a stream of strings from an unknown target language K (belonging to a known collection of languages $\mathcal{L} = \{L_1, L_2, \dots\}$), learn to generate new, previously unseen, strings also belonging to this target language.

Their model is reminiscent of online learning (Littlestone, 1988); there are two players, the generator and the adversary who play the following game: First, the adversary fixes a target language $K \in \mathcal{L}$ and an enumeration of K.[1] Then, at any round $n \geq 1$, it presents the n-th element x_n of the enumeration to the generator. The generator, given the strings $S_n = \{x_1, \dots, x_n\}$ seen so far, outputs a new string $w_n \notin S_n$ – its guess for an unseen string in K. The generator wins the game if eventually it learns "to generate from K." Formally, the generator $\mathcal{G}$ is said to generate from $\mathcal{L}$ in the limit if for all $K \in \mathcal{L}$ and any enumeration of K, there is a finite time $n^\star$ such that, for any subsequent round $n \geq n^\star$, w_n is an unseen element of K, *i.e.*, $w_n \in K \setminus S_n$.

This model has deep connections to the classical works of Gold (1967) and Angluin (1979; 1980), which studied the problem of language *identification* in the limit. In the Gold–Angluin model, like the above model, an algorithm observes an adversarially chosen enumeration of strings from some unknown target language $K = L_{i^\star}$. The only difference is that in the Gold–Angluin model the goal is to eventually *identify* the index $i^\star$ of the correct language, whereas in the Kleinberg–Mullainathan (KM) model the goal is the simpler task of *generation – i.e.*, of outputting unseen strings from K.

[1]An enumeration of K is an infinite sequence of elements (potentially including duplicates) which does not contain elements

Language identification turns out to be hard for essentially all infinite collections of languages. Indeed, Angluin showed that it is intractable for most interesting language collections, including regular languages. Surprisingly, (Kleinberg & Mullainathan, 2024) proved, in stark contrast, that language generation is tractable for *all* countable collections of languages. They provided an elegant algorithm that, given any stream of input strings from a target language K in a countable collection $\mathcal{L} = \{L_1, L_2, \ldots\}$, generates a sequence of previously unseen strings such that beyond a finite time step, all generated strings belong to the target language K.

Main Questions. The KM algorithm eventually stops *hallucinating*, as it ceases outputting elements outside of K after a finite time. However, this property comes at a cost: the KM algorithm sacrifices *breadth – i.e.*, the ability to generate diverse strings from the target language. As the algorithm eliminates hallucinations, it generates from an increasingly smaller subset of the target language, resembling mode collapse in generative adversarial networks (Arjovsky & Bottou, 2017). This observation raises a fundamental question, left open by Kleinberg & Mullainathan (2024):

> **Question #1.** *Is the trade-off between consistency and breadth inherent for generation? In other words, must any algorithm that eventually generates only valid strings from the target language necessarily sacrifice the ability to generate a broad subset of the language?*

To formalize this question, recent work (Kalavasis et al., 2025; Charikar & Pabbaraju, 2024a) relaxed the requirement that the learner outputs one element at a time and allowed it to output a whole set of elements. This also allows for the case where, at some finite point, one can stop training and generate a rich set of responses. With this change, Kalavasis, Mehrotra, and Velegkas (2025) proposed three distinct notions of breadth and showed that, for a large family of generators, language generation with breadth is as hard as language identification. Adding to this result, (Charikar & Pabbaraju, 2024a) proved the impossibility of generation with breadth for a specific language collection (with any generator). While these results suggest a fundamental tension between consistency and breadth, a complete characterization of when different notions of generation with breadth are achievable remains open.

Another intriguing direction initiated by Kalavasis et al. (2025) concerns the stability of generators: a stable generator is one that eventually stops changing its "support," *i.e.*, the set of elements it outputs, after seeing a finite number of distinct strings from the target language. Stability is a central object in online learning and has already been studied in language identification (Gold, 1967). Kalavasis et al. (2025) studied generation under stability showing that certain notions of generation with breadth are "hard" to achieve if generators (from a specific family) are required to be stable, but largely left characterizing the effect of stability on generation with other notions of breadth and with other generators outside this family open.

> **Question #2.** *How does stability interplay with consistency and breadth in language generation?*

1.1. Our Contributions and Technical Novelty

Our work is centered around answering Questions 1 and 2 in the model of language generation in the limit (Kleinberg & Mullainathan, 2024). Next, we describe our main results and techniques.

Results for Question #1. There are many notions of breadth in the literature, all attempting to quantify how much of the target language is covered by a generator. Our first set of results provides a complete characterization of all notions of breadth proposed in prior work (Section 3 and Appendix A). In the main body, we illustrate our results with two of the simplest notions of breadth: *exact breadth* and *approximate breadth* (Kalavasis et al., 2025). Exact breadth is the strongest notion, requiring that after sufficiently many examples, the learner must be able to generate all unseen elements of the target language K. Approximate breadth relaxes this condition, requiring generators to output all but *finitely* many unseen elements of K after seeing enough examples. For exact breadth, we show that:

Informal Theorem 1.1 (see Theorem 3.3)**.** *A generator $\mathcal{G}$ can generate from a collection $\mathcal{L}$ with exact breadth in the limit if and only if $\mathcal{L}$ is identifiable in the limit.*

Thus, collections $\mathcal{L}$ admitting generators with exact breadth are exactly those that are identifiable in the Gold–Angluin model; they have a combinatorial characterization due to (Angluin, 1979) that we call *Angluin's condition* (see Definition 2.2). This result strengthens Kalavasis et al. (2025)'s lower bound which only applied to generators with specific properties; since our result applies to all generators without assumptions, it requires a fundamentally different proof approach.

The above is essentially a negative result because the classes $\mathcal{L}$ satisfying Angluin's condition are known to be very limited (Kleinberg & Mullainathan, 2024). A natural follow-up question is whether relaxing the requirement to approximate breadth, where the generator can miss finitely many elements, might overcome this limitation. For this question, we show:

Informal Theorem 1.2 (see Theorem 3.8)**.** *A generator $\mathcal{G}$ can generate from a collection $\mathcal{L}$ with approximate breadth in the limit if and only if $\mathcal{L}$ satisfies*

outside K, and for every element $x \in K$ there is some position $n_x \in \mathbb{N}$ where x appears.

weak Angluin's Condition (Definition 3.7).

A few remarks are in order. First, the "weak Angluin's condition" (Definition 3.7) is a novel relaxation of Angluin's classic condition (Definition 2.2) that we introduce in this work. We prove that this requirement is strictly weaker than Angluin's original condition (Appendix F.1), establishing that approximate breadth is strictly easier to achieve than exact breadth. Nevertheless, the weak Angluin's condition remains highly restrictive – it is not even satisfied by regular languages, which are far simpler than human languages. This demonstrates that the trade-off between consistency and breadth is inherent and largely unavoidable, even when we weaken our breadth requirement.

Technical Novelty. We view generation with exact or approximate breadth as special cases of generation *properties* relative to the target language. Other such properties might include having uniquely low perplexity or hallucination rate for the target language compared to other languages (see Remark 3.6). Our characterizations of generation with breadth rely on two novel abstractions and also have consequences for other properties: The first is the uniqueness criterion (Definition 3.4) which informally states that if generator G satisfies property P for language L, it cannot satisfy P for any different language L'. We prove the following implications:

- ▷ Properties P with uniqueness can only be achieved for collections satisfying Angluin's condition.
- ▷ Exact breadth (like some other notions of breadth) satisfies uniqueness, establishing the necessity direction of Informal Theorem 1.1. Sufficiency is simpler: if $\mathcal{L}$ satisfies Angluin's condition, we can identify the target language and use its index to generate with exact breadth.

However, approximate breadth (along with some other notions of breadth) does not satisfy uniqueness, and requires our second abstraction, the *finite non-uniqueness criterion* (Definition 3.9). Informally, this weaker condition requires that if $\mathcal{G}$ satisfies a property P for L, then it can also satisfy P for another language L' only if L and L' differ on finitely many elements. We show that:

- ▷ Properties P with finite non-uniqueness can only be achieved for collections satisfying the weak Angluin's condition.
- ▷ Approximate breadth satisfies finite non-uniqueness, establishing the necessity direction of Informal Theorem 1.2. Unlike Informal Theorem 1.1, the sufficiency direction is also non-trivial: collections satisfying weak Angluin's condition are not necessarily identifiable, so we develope a novel algorithm achieving approximate breadth for any such collection.

The most technically intricate parts of these constructions are the lower bounds, which rely on careful *diagonalization* arguments. To establish the upper bounds we present several algorithms that are inspired by the work of Kleinberg & Mullainathan (2024) and the seminal work of Angluin (1980). We elaborate on these techniques in Section 3.3. In summary, these reductions are the main tools that enable us to characterize all existing notions of breadth in the literature and resolve **Question #1.**

Implications for Statistical Settings. Using reductions from prior work, our characterizations extend to statistical settings where examples are drawn from distributions rather than chosen adversarially. We provide unconditional characterizations of generation with exact and approximate breadth in the stochastic model, extending the conditional characterizations of Kalavasis et al. (2025) that were limited to a specific generator family (Remark 3.12 and Appendix D).

Results for Question #2. Next, we investigate how generation with breadth is affected by stability, where generators eventually stop changing their support (Definition 3.13), as defined by (Kalavasis et al., 2025). Our results show that stability creates a unified landscape across notions of breadth:

Informal Theorem 1.3 (see Theorem 3.14)**.** *A **stable** generator $\mathcal{G}$ can generate from a countable collection $\mathcal{L}$ with exact/approximate breadth in the limit if and only if $\mathcal{L}$ is identifiable in the limit.*

This reveals a stark separation between stable and unstable generators, as certain notions that only require the weak Angluin's condition without stability now require the full condition with stability. We also introduce further weaker notions of breadth and make significant progress in characterizing when they can be achieved under stability; due to space constraints, we defer these results to Appendix A.

Technical Novelty. Requiring stability introduces an important challenge: unlike breadth, which can be verified at specific steps t, stability requires examining the infinite future sequence of a generator's behavior. Even if a generator appears stable for arbitrarily many steps, we cannot confirm stability without seeing its entire infinite execution. This challenge in verification breaks our earlier lower bound techniques, making the proof significantly more difficult, and necessitating novel ideas (Section 3.3).

Our results comprehensively map the landscape of language generation with breadth, pinpointing when various notions are achievable and revealing the interplay between consistency, stability, and different notions of breadth. Our abstractions also extend beyond breadth, establishing impossibility results for other desirable generation properties (Remarks 3.6 and 3.11).

1.2. Related Work

Our work directly builds on the framework of Kleinberg & Mullainathan (2024), who introduced the model of language generation in the limit. Since then, a growing line of research has explored various aspects of language generation with and without breadth (*e.g.*, (Li et al., 2024; Kalavasis et al., 2025; Charikar & Pabbaraju, 2024a; Raman & Raman, 2025; Peale et al., 2025)). Here, we discuss the most relevant prior works and defer the discussion of the remaining works to Appendix H.

Language Generation with Breadth. Our work builds upon Kalavasis et al. (2025); Charikar & Pabbaraju (2024a) who study language generation with breadth. Kalavasis et al. (2025) introduced three notions of breadth: exact, approximate, and unambiguous. They explored both Kleinberg & Mullainathan (2024)'s online setting and its statistical counterpart – where the strings are sampled from a distribution instead of being adversarially generated. For specific generator family and each notion of breadth, they characterized which countable collections $\mathcal{L}$ enable generation with breadth (for the last two notions, they also require stability). Charikar & Pabbaraju (2024a) introduced exhaustive generation, another notion of breadth, and provided an unconditional lower bound by constructing a specific language collection for which no algorithm can generate exhaustively. (They also studied questions beyond breadth, discussed in Appendix H). Our work unifies and extends both approaches by providing complete characterizations for all these notions of breadth without assumptions on the generator family that hold for all countable language collections.

Independent and Concurrent Work. Independently of and concurrently to this work, the authors of (Charikar & Pabbaraju, 2024a) updated their manuscript to include a characterization of exhaustive generation Charikar & Pabbaraju (2024b) which is similar to our result on approximate breadth (Theorem 3.8). Our work provides several additional contributions beyond this shared result, including characterizations of all existing notions of breadth (Section 3.1), lower bounds for abstract properties of generation – extending beyond breadth (Section 3.1 and Remarks 3.6 and 3.11), characterizations for stable generation (Section 3.2), and characterizations for the statistical setting (Remark 3.12).

Subsequent Work. Two papers follow-up on our work to study more fine-grained notions of breadth. Peale et al. (2025) introduce "representation," a weaker notion of breadth that requires the generator's outputs to proportionally represent certain groups of (elements in) the domain. Kleinberg & Wei (2025) weaken approximate breadth by allowing generators to miss infinitely many elements from the target language, instead focusing on the output set's "density" in the target language. Both of these works address natural follow-up questions raised by our results while being orthogonal.

2. Preliminaries

In this section, we present some background on language identification and generation in the limit.

Notation. Let Σ be a finite alphabet (*e.g.*, $\{a, b, \dots, z\}$) and Σ^* the set of all finite-length strings formed by concatenating symbols from Σ. We define a language L as an infinite subset of Σ^*. A countable collection of languages is denoted by $\mathcal{L} = \{L_1, L_2, \dots\}$. We define a generating algorithm $\mathcal{G} = (\mathcal{G}_n)_{n\in\mathbb{N}}$ as a sequence of mappings $\mathcal{G}_n \colon (\Sigma^*)^n \to 2^{\Sigma^*}$ parametrized by the input size n. In words, the generator maps a finite training set to a (potentially infinite) set of elements.

Language Generation in the Limit. We now formally define language generation in the limit.

Definition 2.1 (Language Generation in the Limit (Kleinberg & Mullainathan, 2024)). *Let $\mathcal{L} = \{L_1, L_2, \dots\}$ be a collection of languages, $\mathcal{G} = (\mathcal{G}_n)$ be a generating algorithm, and $K \in \mathcal{L}$ be some target language. The algorithm $\mathcal{G}$ is said to generate from K in the limit if, for all enumerations of K, there is some $n^* \in \mathbb{N}$ such that for all steps $n \geq n^*$, the algorithm's output $\mathcal{G}_n(S_n)$ is a subset of $K \setminus S_n$, where S_n are the first n elements of the enumeration. The collection $\mathcal{L}$ allows for generation in the limit if there is an algorithm $\mathcal{G}$ that generates from K in the limit for any $K \in \mathcal{L}$.*

To gain some intuition about this definition, consider the collection $\mathcal{L} = \{\mathbb{Z}, L_1, L_{-1}, L_2, L_{-2}, \dots\}$ of thresholds over integers where, for each $i \in \mathbb{Z}$, $L_i = \{i, i+1, i+2, \dots\}$. Suppose the target language is some $K \in \mathcal{L}$ and the adversary first enumerates string x_1. The generator can deduce that $K = L_z$ for some $z \leq x_1$, *i.e.*, $K \in \{\mathbb{Z}, L_{x_1}, L_{x_1-1}, \dots\}$. Since the intersection of all of these languages is non-empty and is a strict superset of the strings enumerated so far (namely, the intersection is $\{x_1 + 1, x_1 + 2, \dots\}$), the generator can generate an element that is guaranteed to be in K: for instance, it is sufficient to output $\{x_1 + 1\}$. More generally, after seeing strings $x_1, x_2, \dots, x_i$, the generator can output a singleton containing any integer larger than $\max_i x_i$.

For the problem to be interesting, Kleinberg & Mullainathan (2024) assumed throughout that each language in the collection has infinite cardinality, *i.e.*, $|L_i| = \infty$ for all i. (Otherwise, $K \setminus S_n$ eventually becomes empty.) They showed that language generation in the limit is possible for *all* countable collections of languages – starkly contrasting results in language identification, discussed next. The KM algorithm is a key starting point for our algorithms, and we discuss it in Section 3.3.

Language Identification in the Limit. Language identification in the limit was introduced by Gold (1967) and has, since, been widely studied in learning theory. The model is slightly different from that of generation: while generation only requires producing valid examples from the target language $K = L_{i^*}$, identification requires the learner to eventually determine the exact identity i^* (index) of the target language in the collection. Despite this seemingly minor difference, identification is dramatically harder than generation: indeed, generation is possible for any countable collection (Kleinberg & Mullainathan, 2024), but identification is only possible for very limited collections (Angluin, 1979; 1980), which satisfy a certain structural property that we explain next. A formal definition of language identification appears in Appendix G but is not essential for understanding this paper.

Angluin's Condition. A key concept in our analysis is Angluin's condition – a structural property of language collections $\mathcal{L}$ that characterizes identifiability: $\mathcal{L}$ is identifiable if and only if it satisfies Angluin's condition (Angluin, 1980). Informally, a collection satisfies Angluin's condition if for any language $L \in \mathcal{L}$, there exists a finite subset T_L (called a tell-tale set) that serves as a finite "fingerprint" allowing one to distinguish L from any other language that contains T_L.

Definition 2.2 (Angluin's Condition (Angluin, 1980))**.** *Fix a language collection $\mathcal{L} = \{L_1, L_2, \dots\}$. The collection $\mathcal{L}$ is said to satisfy Angluin's condition if for any index i, there is a tell-tale, i.e., a finite set of strings T_i such that T_i is a subset of L_i, i.e., $T_i \subseteq L_i$, and the following holds:*

For all $j \geq 1$, if $L_j \supseteq T_i$, then L_j is not a proper subset of L_i.

Roughly, this condition ensures that after observing enough examples from the target language, one can rule out all incorrect languages. We refer to Figure 1 for a visualization of the condition.

3. Our Results and Techniques

In this section, we present our main results. We begin with two notions of generation with breadth from prior work, provide characterizations of generation with breadth (Section 3.1) and their implications (Remark 3.12), examine stable generation (Section 3.2), and overview our proof techniques (Section 3.3). While we focus on exact and approximate breadth in the main body, our techniques extend to all existing notions and their natural combinations; we present these extensions in Appendix A.

Notions of Breadth. Recent works have introduced various notions of breadth, capturing different aspects of how generators cover a target language. The first notion, *exact breadth* (introduced by Kalavasis et al. (2025) and studied by Charikar & Pabbaraju (2024b)). Given samples S, a generator $\mathcal{G}$ has exact breadth for K if $\mathcal{G}(S) = K \setminus S$, meaning it generates all unseen strings in K.

Definition 3.1. *Generator $\mathcal{G}$ has exact breadth for language K given samples S if $\mathcal{G}(S) = K \setminus S$.*

In words, language generation in the limit with exact breadth requires that, for any target language $K \in \mathcal{L}$ and any enumeration of K, there is an $n^* \geq 1$, such that for all $n \geq n^*$, after seeing n elements of the enumeration S_n, $\mathcal{G}$ achieves exact breadth for language K.

Recognizing that this is a strong requirement, Kalavasis et al. (2025) also introduced a natural relaxation, approximate breadth, which allows the generator to miss a finite number of elements.

Definition 3.2. *Generator $\mathcal{G}$ has approximate breadth for language K given samples S if $\mathcal{G}(S) \subseteq K$ and $|K \setminus \mathcal{G}(S)| < \infty$.*

Again, one can naturally define language generation in the limit with approximate breadth as above. Next, we present our results for these two notions of breadth. We mention that we also characterize generation under all other notions of breadth introduced in prior work (see Appendix A).

3.1. Results on Generation with Breadth

Our first result characterizes language generation with exact breadth.

Theorem 3.3 (Exact Breadth $\iff$ Angluin's Condition)**.** *For any countable collection of languages $\mathcal{L}$, there is a generator $\mathcal{G} = (\mathcal{G}_n)$ that generates with exact breadth from $\mathcal{L}$ in the limit if and only if $\mathcal{L}$ satisfies Angluin's condition.*

This result establishes that generation with exact breadth is as hard as language identification in the limit, which is a much more challenging problem than generation in the limit without breadth constraints. Our characterization generalizes previous work in several ways: it removes technical conditions on the generators needed by Kalavasis et al. (2025) and extends the unconditional lower bound of Charikar & Pabbaraju (2024a), which only held for a specific language collection.

Generalization to Any "Unique" Property. One side of this result, the upper bound, is simple: at a high level, if Angluin's condition holds, then language identification is possible (*i.e.*, one can find $i^\star$ such that $K = L_{i^\star}$), and then, one can generate with exact breadth by outputting the first unseen string from K. (That said, there are some difficulties because we do not know when we have found $i^\star$, and we handle this in our proofs.) The other side, the lower bound, is non-trivial and is actually a corollary of a much more general result concerning a property we call *uniqueness*.

Definition 3.4 (Uniqueness)**.** *A property P of generation satisfies the uniqueness criterion for a collection $\mathcal{L}$ if no*

generator $\mathcal{G}$ can simultaneously satisfy that property for two different languages $L \neq L'$ in $\mathcal{L}$, i.e., if $\mathcal{G}$ has property P for L, then it cannot have P for $L' \neq L$ and vice versa.

We prove the following lower bound for any property satisfying the uniqueness criterion.

Theorem 3.5 (Lower bound with Uniqueness)**.** *Let P be any property of generation that satisfies the uniqueness criterion. For a countable collection of languages $\mathcal{L}$, there exists an algorithm that generates with property P from $\mathcal{L}$ in the limit only if $\mathcal{L}$ satisfies Angluin's Condition.*

To gain some intuition, note that exact breadth satisfies this uniqueness criterion: if a generator $\mathcal{G}$ generates a language L with exact breadth (*i.e.*, $\mathcal{G}(S) = L \backslash S$), then it necessarily cannot generate any other language $L' \neq L$ with exact breadth. In contrast, approximate breadth does not satisfy uniqueness: for collections containing languages $L_1 \subseteq L_2$ that differ on only finitely many elements, a generator with support L_1 can simultaneously generate with approximate breadth from L_1 and L_2. Like exact breadth, other notions of breadth in the literature also satisfy the uniqueness condition and Theorem 3.5 is a powerful tool for proving lower bounds for such notions.

Remark 3.6 (Implications Beyond Generation with Breadth). The theorem also has implications well beyond breadth. Assume we require only that a generator's evaluation metric – *e.g.*, lower perplexity or hallucination rate – is *strictly* better on a target language K than on every other $L \neq K$. Even this weaker "metric-separation" goal is attainable *only* when the language collection $\mathcal{L}$ satisfies Angluin's condition; if not, then no generator can perform strictly better for the target K than the rest. This fundamental limit applies regardless of the specific metric.

Characterization of Approximate Breadth. Next, we move to approximate breadth. Since approximate breadth does not satisfy the uniqueness criteria introduced above, we cannot show a lower bound for approximate breadth based on Angluin's condition, and need new ideas. In fact, the reason why approximate breadth does not satisfy it hints towards the required relaxation that we need to impose on Angluin's condition: languages that differ on finitely many elements need to be treated differently from languages that differ on infinitely many elements. Motivated by this, we introduce a variant of Angluin's condition we call the weak Angluin's condition:

Definition 3.7 (Weak Angluin's Condition)**.** *Fix a language collection $\mathcal{L} = \{L_1, L_2, \dots\}$. The collection $\mathcal{L}$ is said to satisfy the weak Angluin's condition if for any index i, there is a tell-tale, i.e., a finite set of strings T_i such that T_i is a subset of L_i, i.e., $T_i \subseteq L_i$, and the following holds:*

For all $j \geq 1$ such that $L_j \supseteq T_i$, one of the following holds.

- *Either L_j is not a proper subset of L_i; or*
- *L_j is a proper subset and misses finitely many elements of L_i, i.e., $|L_i \setminus L_j| < \infty$.*

The tell-tale oracle is a primitive that, given an index i, outputs an enumeration of the set T_i.

For a visualization of this condition, we refer to Figure 1. This condition relaxes Angluin's condition by allowing language L_j containing the tell-tale set T_i of language L_i to be a proper subset of L_i provided L_j misses only finitely many elements (see Figure 1). We remark that this is a *strict* weakening of Angluin's condition (see Appendix F.1).

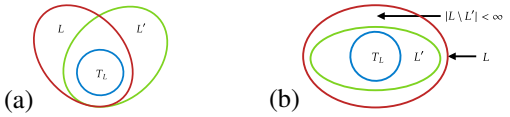


Figure 1: Figure 1a visualizes Angluin's condition: any language L' containing language L's tell-tale set T_L cannot be a strict subset of L. Our weak Angluin's condition relaxes this by allowing an additional case (Figure 1b): a language L' containing T_L can be a strict subset of L provided L' only misses finitely many elements of L (*i.e.*, $|L \setminus L'| < \infty$).

Our next result characterizes approximate breadth via the Weak Angluin's Condition.

Theorem 3.8 (Approximate Breadth $\iff$ Weak Angluin's Condition)**.** *For any countable collection of languages $\mathcal{L}$, there is a generator $\mathcal{G} = (\mathcal{G}_n)$ that generates with approximate breadth from $\mathcal{L}$ in the limit if and only if $\mathcal{L}$ satisfies the weak Angluin's condition (Definition 3.7).*

Since approximate breadth is characterized by the weak Angluin's condition, which is strictly weaker than Angluin's condition, approximate breadth is a strictly weaker requirement than exact breadth.

Unlike the characterization of exact breadth, the upper bound side of this result is not simple. This is because if a language collection $\mathcal{L}$ satisfies the weak Angluin's condition, it may not be identifiable, and hence we need a different algorithm for generation that achieves approximate breadth. We design a new algorithm based on the weak Angluin's condition and overview it in Section 3.3. Like with characterization of exact-breadth, the lower-bound side of this argument is non-trivial and a corollary of a more general result concerning a property of *finite non-uniqueness*.

Generalization to Any "Finitely Non-Unique" Property. Roughly speaking, finite non-uniqueness relaxes uniqueness by allowing properties that can hold for two languages L and L' simultaneously but only when L and L' differ on finitely many elements.

Definition 3.9 (Finite Non-Uniqueness)**.** *A property P of generation satisfies the finite non-uniqueness criterion for a collection $\mathcal{L}$ if no generator $\mathcal{G}$ can simultaneously satisfy that property for two languages $L, L' \in \mathcal{L}$ that differ in*

infinitely many elements (i.e., when $|L \triangle L'| = \infty$), i.e., if $\mathcal{G}$ has property P for L and L' both, then $|L' \triangle L| < \infty$.

To gain some intuition, note that approximate breadth satisfies this finite non-uniqueness criterion: if a generator generates with approximate breadth from two different languages L and L', then these languages can only differ on finitely many elements. This follows because the generator's support must be largely contained in both languages (with only finitely many elements missing), which is only possible when $|L \triangle L'| < \infty$.

Our next result shows that achieving any property which satisfies finite non-uniqueness is already impossible for any collection that does not satisfy the weak Angluin's condition.

Theorem 3.10 (Lower bound with Finite Non-Uniqueness)**.** *Let P be any property of generation satisfying the finite non-uniqueness criterion. For a countable collection $\mathcal{L}$, there exists an algorithm that generates with property P from $\mathcal{L}$ in the limit only if $\mathcal{L}$ satisfies the weak Angluin's Condition.*

This lets us characterize every breadth notion in the literature, including approximate breadth.

Remark 3.11 (Implications Beyond Generation with Breadth). The same reasoning as in Remark 3.6 yields lower bounds for an even milder objective: achieving *optimal* (rather than uniquely optimal) performance on K together with finitely many other languages. If $\mathcal{L}$ fails the *weak* Angluin condition, then no generator can attain the best possible perplexity – or any analogous metric – on a finite set of languages $\mathcal{L}' \subseteq \mathcal{L}$ (with $|\mathcal{L}'| < \infty$) which includes the target language K (*i.e.*, $K \in \mathcal{L}'$).

Remark 3.12 (Implications for Statistical Setting). Using Kalavasis et al. (2025)'s framework, our results extend to statistical settings where strings are sampled from distributions rather than adversarially chosen. Concretely, we provide unconditional characterizations for generation with both exact and approximate breadth in stochastic models – improving upon the earlier conditional results that applied only to a specific generator family (Kalavasis et al., 2025). See Appendix D for details.

3.2. Results on Stable Generation with Breadth

Our next set of results focuses on *stable generators* – those whose support eventually stops changing – a requirement motivated by practical algorithms that converge to a model and by Gold's original work, which also required stability. Under stability, the landscape changes dramatically:

Definition 3.13 (Stable Generating Algorithm (Kalavasis et al., 2025))**.** *A generating algorithm $\mathcal{G} = (\mathcal{G}_n)$ is stable for a language collection $\mathcal{L}$ if for any target language $K \in \mathcal{L}$ and for any enumeration of K, there is some finite $n^* \in \mathbb{N}$ such that for all $n, n' \geq n^*$, it holds that $\mathcal{G}_n(S_n) = \mathcal{G}_{n'}(S_{n'})$.*

Theorem 3.14 (Characterization for Stable Generation)**.** *Fix any countable collection of languages $\mathcal{L}$. $\mathcal{L}$ satisfies Angluin's condition if and only if one of the following two equivalent conditions hold*

- ▷ *There is a stable algorithm that generates with approximate breadth from $\mathcal{L}$.*
- ▷ *There is a stable algorithm that generates with exact breadth from $\mathcal{L}$.*

Hence, exact and approximate breadth are equivalent under stability, both requiring the (full) Angluin's condition – contrasting with our earlier result where approximate breadth only requires the weak Angluin's condition. In fact, a stronger result holds: all notions of breadth proposed in prior work collapse to this same characterization under stability. In Appendix E, we prove this and also present additional results that allow hallucinations and introduce weaker breadth notions.

3.3. Technical Overview

We now outline our proof techniques and their novelty, beginning with our lower bound results.

Lower Bounds. Our goal is to show that if a collection $\mathcal{L}$ lacks a certain property (*e.g.*, Angluin's condition), then no generator can achieve the corresponding notion of breadth (*e.g.*, exact breadth) for $\mathcal{L}$. The full proofs appear in Appendix B. First, we overview techniques in existing works.

- ▷ *Technique I: Generator-Specific Bounds.* (Kalavasis et al., 2025)'s approach require generators satisfy a technical condition (Appendix G.2) that, roughly, enables access to their "support," or the set of their outputs, allowing a reduction from language identification to generation with breadth. This, however, fails for unconditional lower bounds which make no assumptions on generators.
- ▷ *Technique II: Diagonalization for Identification.* For the related problem of language identification, the standard and only technique for proving unconditional lower bounds is *diagonalization* (*e.g.*, (Gold, 1967)). At a high level, it constructs an algorithm-dependent enumeration of target language K in phases: in the i-th phase, it enumerates L_i, and either the algorithm A fails to identify L_i or A guesses the index as i, at which point the enumeration advances to phase $i + 1$. This creates a dilemma: either a phase continues indefinitely (causing infinitely many identification errors) or infinitely many phases occur (meaning A misidentifies the language $K = L_\infty$ infinitely often).
- ▷ *Technique III: Collection-Specific Bounds.* (Charikar & Pabbaraju, 2024a) adapted the above diagonalization technique to prove generation with breadth is impossible for a specific "hard" collection $\mathcal{L}^*$ – yielding the first

unconditional lower bound for generation with breadth, albeit one limited to just one collection.

The complementary limitations of prior work raise a natural question: Can we prove lower bounds for language generation with breadth that simultaneously apply to all generators and for all collections for which generation with breadth is fundamentally impossible?

Idea 1: Universal Diagonalization. We generalize Charikar and Pabbaraju's diagonalization from a specific "hard" collection $\mathcal{L}^*$ to *all* collections violating Angluin's condition – which is a tight result since Kalavasis et al. (2025) give a generator with exact breadth for collections satisfying this condition. Here, the key insight is leveraging the structure of collections that violate Angluin's condition: Specifically, we set L_∞ to be the language witnessing this violation, and index the remaining languages $L_1, L_2, \ldots$ with finite subsets of L_∞: for each finite $T \subseteq L_\infty$, L_T is the language containing T and satisfying $L_T \subsetneq L_\infty$ (guaranteed by the violation of Angluin's condition).

Idea 2: Weak Angluin's Condition. This approach fails for approximate breadth because a generator can simultaneously achieve approximate breadth for multiple languages. We address this by introducing the weak Angluin's condition, a relaxation of the original, and proving it enables diagonalization for approximate breadth. This lower bound is also tight: we provide a novel algorithm achieving approximate breadth for any collection satisfying this weaker condition.

Challenge: Diagonalization against Stable Generators. While our previous (unconditional) lower bounds apply to stable generators, they do not yield tight characterizations for notions like approximate breadth. The core issue is that unlike breadth – which can be verified at specific steps t – verifying stability requires examining the generator's behavior over infinitely many steps. As even if a generator is stable for many steps, we cannot confirm its stabile without seeing its future behaviour.

Idea 3: Lazy Analysis of Diagonalization. To address this challenge, we introduce a "lazy analysis" of diagonalization, loosely inspired by techniques in computational complexity (Arora & Barak, 2009). Unlike the standard analysis of diagonalization where the adversary forces the generator to make "mistakes" at the end of each phase, here the adversary cannot force a mistake every round. Instead, this lazy analysis uses the fact that after waiting for sufficiently many rounds, the generator "exhausts all possibilities" and must make a mistake. Proving this requires a sophisticated technical construction which shows that a generator must either be unstable or generate without approximate breadth infinitely often. We believe this technique is of independent interest and can have further applications in the analysis of natural properties of stable generators beyond breadth.

Upper Bounds. Our upper bounds construct algorithms for generation with (different notions of) breadth that work whenever collection $\mathcal{L}$ satisfies properties like Angluin's condition. For exact breadth, one already exists in prior work (Kalavasis et al., 2025). Here, we focus on approximate breadth; we develop two algorithms for it with different access models of $\mathcal{L}$: one with unrestricted access and another with only membership access (ability to query "is $w \in L_i$?"). The membership-only algorithm is a novel adaptation of Angluin (1980)'s seminal algorithm and is presented in Appendix C.2 due to space constraints. Here, we overview the simpler unrestricted-access algorithm.

KM24's Algorithm. Kleinberg & Mullainathan (2024)'s algorithm, in every round t, creates a chain of *critical languages* $\mathcal{C}_1 \supsetneq \mathcal{C}_2 \supsetneq \cdots \supsetneq \mathcal{C}_t$ with the property that, for large enough t, the target language K enters this chain and remains in it. Now their algorithm is simple: it outputs (unseen strings from) the last critical language. Unfortunately, this algorithm loses breadth as t increases, as it keeps generating from the last element of a constantly decreasing chain.

New Analysis of KM24's Algorithm. If $\mathcal{L}$ satisfies Angluin's condition, then Kalavasis et al. (2025) have already shown that this algorithm achieves exact breadth. To achieve approximate breadth, we show that when $\mathcal{L}$ satisfies weak Angluin's condition, the last critical language, $\mathcal{C}_t$, misses at most finitely many elements of K (*i.e.*, $|K \setminus \mathcal{C}_t| < \infty$) for large enough t. This shows that the above algorithm achieves approximate breadth for such $\mathcal{L}$. This reveals an interesting best-of-three-worlds property: if $\mathcal{L}$ satisfies Angluin's condition it achieves exact breadth, if it satisfies weak Angluin's condition it achieves approximate breadth, otherwise it achieves consistent generation. This is particularly appealing as these conditions might be challenging to verify given limited access to $\mathcal{L}$. Finally, to obtain algorithms for other existing notions of breadth, we use this as a building block (Appendix C.3).

4. Concluding Remarks

In this work, we continue the study of language generation, a nascent area introduced by Kleinberg & Mullainathan (2024). On a conceptual level, our results – building on prior work – offer a resolution to the main open question of Kleinberg and Mullainathan showing that, indeed, a tension between validity and breadth is inherent in language generation, at least under all the formal notions of breadth considered in prior work (Kalavasis et al., 2025; Charikar & Pabbaraju, 2024a). On a technical level, we introduce novel diagonalization-based lower bound techniques and new algorithms that achieve generation with breadth whenever possible. Though we focus on the prompt-less setting, our techniques extend to the prompted generation setting as well (Kleinberg & Mullainathan, 2024). Our work sug-

gests several promising directions for future work: investigating weaker notions of breadth, completing the characterizations for certain novel variants of stable generation (Appendix A.3), and identifying what additional information beyond positive examples could help generators achieve both validity and breadth – an intriguing challenge given our impossibility results.

References

Angluin, D. Finding patterns common to a set of strings (extended abstract). In *Proceedings of the Eleventh Annual ACM Symposium on Theory of Computing*, STOC '79, pp. 130–141, New York, NY, USA, 1979. Association for Computing Machinery. ISBN 9781450374385. doi: 10.1145/800135.804406. URL https://doi.org/10.1145/800135.804406.

Angluin, D. Inductive inference of formal languages from positive data. *Information and Control*, 45(2):117–135, 1980. ISSN 0019-9958. doi: https://doi.org/10.1016/S0019-9958(80)90285-5. URL https://www.sciencedirect.com/science/article/pii/S0019995880902855.

Angluin, D. *Identifying Languages From Stochastic Examples*. Yale University. Department of Computer Science, 1988. URL http://www.cs.yale.edu/publications/techreports/tr614.pdf.

Arjovsky, M. and Bottou, L. Towards principled methods for training generative adversarial networks. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=Hk4_qw5xe.

Arora, S. and Barak, B. *Computational Complexity: A Modern Approach*. Cambridge University Press, 2009.

Bousquet, O., Hanneke, S., Moran, S., van Handel, R., and Yehudayoff, A. A theory of universal learning. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2021, pp. 532–541, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450380539. doi: 10.1145/3406325.3451087. URL https://doi.org/10.1145/3406325.3451087.

Charikar, M. and Pabbaraju, C. Exploring facets of language generation in the limit, 2024a. URL https://arxiv.org/pdf/2411.15364v1.

Charikar, M. and Pabbaraju, C. Exploring facets of language generation in the limit, 2024b. URL https://arxiv.org/pdf/2411.15364v2.

Gold, E. M. Language identification in the limit. *Information and Control*, 10(5):447–474, 1967. ISSN 0019-9958. doi: https://doi.org/10.1016/S0019-9958(67)91165-5. URL https://www.sciencedirect.com/science/article/pii/S0019995867911655.

Kalavasis, A., Mehrotra, A., and Velegkas, G. On the limits of language generation: Trade-offs between hallucination and mode collapse. In *Proceedings of the 57th Annual ACM Symposium on Theory of Computing (STOC'25)*, New York, NY, USA, 2025. Association for Computing Machinery. URL https://arxiv.org/abs/2411.09642.

Karbasi, A., Montasser, O., Sous, J., and Velegkas, G. (im)possibility of automated hallucination detection in large language models, 2025. URL https://arxiv.org/abs/2504.17004.

Kleinberg, J. and Mullainathan, S. Language generation in the limit. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=FGTDe6EA0B.

Kleinberg, J. and Wei, F. Density measures for language generation, 2025. URL https://arxiv.org/abs/2504.14370.

Li, J., Raman, V., and Tewari, A. Generation through the lens of learning theory, 2024. URL https://arxiv.org/abs/2410.13714.

Littlestone, N. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning*, 2(4):285–318, 1988. doi: 10.1007/BF00116827. URL https://doi.org/10.1007/BF00116827.

Peale, C., Raman, V., and Reingold, O. Representative language generation. In *Forty-second International Conference on Machine Learning*, 2025.

Raman, A. and Raman, V. Generation from noisy examples. In *Forty-second International Conference on Machine Learning*, 2025.

Shannon, C. E. The redundancy of english. In *Cybernetics; Transactions of the 7th Conference, New York: Josiah Macy, Jr. Foundation*, pp. 248–272, 1951. URL https://jontalle.web.engr.illinois.edu/uploads/537.F18/Papers/Shannon50b.pdf.

A. Summary Characterizations with Language Generation

In this section, we summarize our characterizations for language generation with all existing notions of breadth, with additional results for new notions presented in Appendix E.

Outline. We first define two additional notions of breadth from prior work (Appendix A.1), completing all definitions of breadth in prior works, alongside exact breadth (Definition 3.1) and approximate breadth (Definition 3.2) from the main body. We then provide characterizations for each notion (Appendix A.2), extending Theorems 3.3 and 3.8. Finally, we examine stable generation with breadth (Appendix A.3), extending Theorem 3.14, and consider settings allowing some hallucinations, both for unstable (Appendix A.4) and stable generators (Appendix A.5).

A.1. Remaining Notions of Breadth in Prior Work

In this section, we introduce two additional notions of breadth, unambiguous generation and exhaustive generation, completing all definitions of breadth in prior works, alongside exact breadth (Definition 3.1) and approximate breadth (Definition 3.2) from the main body.

Unambiguous Generation. This relaxation of exact breadth by Kalavasis et al. (2025) allows hallucination (outputting strings outside target language K) provided the generator performs "better" for K than for any other language in the collection.

Definition A.1 (Unambiguous Generation in the Limit (Kalavasis et al., 2025))**.** *Generator $\mathcal{G}$ unambiguously generates from language K given samples S if*

$$|\mathcal{G}(S) \triangle K| < \min_{L \in \mathcal{L}:\, L \neq K} |\mathcal{G}(S) \triangle L| \,, \tag{1}$$

where $A \triangle B \coloneqq (A \setminus B) \cup (B \setminus A)$ for sets A and B.

While unambiguous generation is seemingly weaker than exact breadth and incomparable to approximate breadth, our characterization (Theorem A.3) reveals that it is as hard to achieve as exact breadth.

Exhaustive Generation. (Charikar & Pabbaraju, 2024a) proposed exhaustive generation.[2] Their formulation treats generators as mappings from domain sequences to domain *enumerations*. For $i, n \in \mathbb{N}$, let $\mathcal{G}_n(i)$ be the i-th element in the enumeration output in round n.

Definition A.2 (Exhaustive Generation in the Limit (Charikar & Pabbaraju, 2024b))**.** *Generator $\mathcal{G}$ exhaustively generates from language K in round n if*

$$\left|\bigcup_{i=1}^{\infty} \mathcal{G}_n(i) \setminus K\right| < \infty \quad \textit{and} \quad S_n \cup \bigcup_{j=1}^{n-1} \mathcal{G}_j(1) \cup \bigcup_{i=1}^{\infty} \mathcal{G}_n(i) \supseteq K \,, \tag{2}$$

where S_n is the set of elements enumerated until round n.

Exhaustive generation is strictly weaker than exact breadth but seems incomparable to approximate breadth: it permits finite hallucinations (which approximate breadth forbids) but requires covering K using potentially all past outputs (which approximate breadth does not require). Our characterization (Theorem A.3) reveals that it is as hard to achieve as approximate breadth.

A.2. Generation with Breadth (Extension of Theorems 3.3 and 3.8 and Proof Sketch)

Our next result characterizes generation with all four existing notions of breadth in the literature.

Theorem A.3 (Characterizations of Language Generation with Breadth)**.** *For any countable collection of languages $\mathcal{L}$ the following hold:*

1. *The following are equivalent:*
   - ▷ *There is an algorithm that generates with (exact) breadth from $\mathcal{L}$ in the limit.*
   - ▷ *There is an algorithm that generates unambiguously from $\mathcal{L}$ in the limit.*

[2] The definition in (Charikar & Pabbaraju, 2024a) differs slightly from (Charikar & Pabbaraju, 2024b). We use the updated version, though our techniques also show that both properties are characterized by the same condition.

▷ *$\mathcal{L}$ satisfies Angluin's condition (Definition 2.2).*

2. *The following are equivalent:*
   - ▷ *There is an algorithm that generates with approximate breadth from $\mathcal{L}$ in the limit.*
   - ▷ *There is an algorithm that generates exhaustively from $\mathcal{L}$ in the limit.*
   - ▷ *$\mathcal{L}$ satisfies the weak Angluin's condition (Definition 3.7).*

This results generalizes Theorems 3.3 and 3.8 from the main body. Like Theorems 3.3 and 3.8, this result is *unconditional*, requiring no particular structure on the generator. Hence, it strengthens the conditional lower bounds of (Kalavasis et al., 2025). It also applies to all countable language collections, strengthening the collection-specific results of Charikar & Pabbaraju (2024a).

Proof Sketch of Theorem A.3. We outline four key components:

- *Upper bound when $\mathcal{L}$ satisfies Angluin's condition:* Since $\mathcal{L}$ is identifiable (by Angluin's result), we can convert any identification algorithm to an exact generator, as established by Kalavasis et al. (2025). Unambiguous generation follows since it is weaker than exact breadth.
- *Lower bound when $\mathcal{L}$ violates Angluin's condition:* In Appendix B.1, we prove that properties satisfying uniqueness are unachievable for collections violating Angluin's condition (see Section 3.3 for a discussion). Given this, the present result follows since exact breadth and unambiguous generation both satisfy uniqueness.
- *Upper bound when $\mathcal{L}$ satisfies weak Angluin's condition:* Since weak Angluin's condition is strictly weaker than Angluin's condition, $\mathcal{L}$ is generally not identifiable and so we cannot use algorithms from the above upper bound. We present new algorithms for this case in Appendix C.
- *Lower bound when $\mathcal{L}$ violates weak Angluin's condition:* In Appendix B.2, we prove that properties satisfying finite non-uniqueness are unachievable for collections violating weak Angluin's condition (see Section 3.3 for some discussion). The result follows from this since approximate breadth and exhaustive generation both satisfy finite non-uniqueness. □

A.3. Generation with Breadth and Stability (Extension of Theorem 3.14 and Proof Sketch)

In this section, we provide characterizations for generation with stable generators, those whose support eventually stops changing and stabilizes (Definition 3.13).

Remark A.4 (Discussion on Stability). This notion of stability stems from the original work of (Gold, 1967) on language identification in the limit, where Gold requires the learner to stabilize to a specific guess for the target language L in the above sense (see Appendix G). It is also closely related to the question of whether the algorithm can verify that it has "learned" to generate with the required notion of breadth; if the algorithm can verify that it has learned, then it can stabilize. Further, any generator that is consistent and achieves exact breadth is also stable, since after some finite point its support must become identical to the target language K and remain so.[3]

Landscape with Stable Generators. Under the stability requirement, the landscape for generation with breadth changes (compared to the one in the previous section).

Theorem A.5 (Characterizations of Stable Language Generation with Breadth)**.** *For any countable collection of languages $\mathcal{L}$, the following are equivalent:*

- *There is a stable algorithm that generates with approximate breadth from $\mathcal{L}$ in the limit.*
- *There is a stable algorithm that generates exhaustively from $\mathcal{L}$ in the limit.*
- *There is a stable algorithm that generates with (exact) breadth from $\mathcal{L}$ in the limit.*
- *There is a stable algorithm that generates unambiguously from $\mathcal{L}$ in the limit.*
- *$\mathcal{L}$ satisfies Angluin's condition (Definition 2.2).*

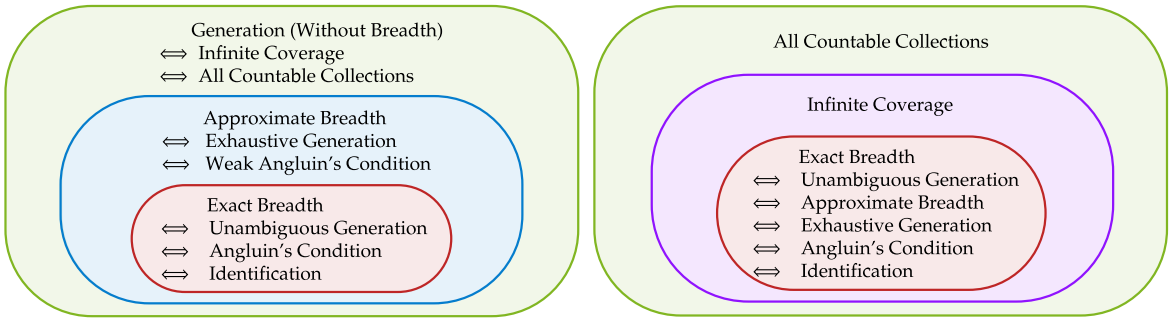
[3]Here, we use an equivalent notion of generation with exact breadth that allows for inclusion of the training set in the support: the equivalence holds because any generator $\mathcal{G}$ that generates with breadth without repeating training examples can be converted to one $\mathcal{G}'$ that generates with breadth and repeats the training examples and vice versa.

This result extends Theorem 3.14. Like Theorem 3.14, it shows that the requirement of stability makes the problem of generation with approximate breadth strictly harder (see Figure 2): there exist stable generators with this property if and only if the collection satisfies Angluin's condition for identifiability whereas before, when unstable generators were also allowed, one only required the weak Angluin's condition. As another example of the stark change in the landscape, we also show that there exists a collection that satisfies the weak Angluin's condition (hence admits a non-stable generator with approximate breadth), but for which no stable generator can achieve a much weaker requirement, which we term infinite coverage (Theorem E.7).

Proof Sketch of Theorem A.5. We outline two main components:

- *Upper bound when $\mathcal{L}$ satisfies Angluin's condition:* This follows by observing that Theorem A.3's upper bound for collections satisfying Angluin's condition constructs stable generators.
- *Lower bound when $\mathcal{L}$ violates Angluin's condition:* For exact breadth and unambiguous generation, this follows from Theorem A.3. The key technical challenge is establishing lower bounds for approximate breadth and exhaustive generation, requiring a certain "lazy analysis" of diagonalization as discussed in Section 3.3. The proof appears in Appendix B.3.

□



(a) Unconditional Characterizations

(b) Characterization With Stable Generators

Figure 2: *Comparison of Generation in the Limit with and without Requiring Stability.* Each containment illustrated by a border is *strict*, *i.e.*, for each border there is a language collection that satisfies the outer containment but not the inner containment. Concretely, in the figure on the left, there are (1) language collections that do not satisfy the Weak Angluin's Condition (Definition 3.7) (see Example E.12), (2) language collections that satisfy the Weak Angluin's Condition, but not Angluin's condition (see Example E.6), and (3) there are language collections which satisfy Angluin's Condition (Definition 2.2) (*e.g.*, all finite collections). The figure on the right depicts the characterization for stable generators. In addition to what is depicted there, there are (1) language collections that satisfy the weak Angluin's condition and for which infinite coverage is not achievable (see Theorem E.7) and (2) language collections for which infinite coverage is achievable but that do not satisfy the weak Angluin's Condition (Definition 3.7) (see Example E.12). We note that (1) and (2) *are* not depicted in the right figure.

A.4. Generation with Breadth and Hallucinations

To illustrate the generality of our techniques, we use them to obtain characterizations for several new notions of generation. In particular, we obtain characterizations for generation with breadth where we relax the requirement that the generation becomes consistent (*i.e.*, it has no hallucinations) in the limit. Instead, we allow for two cases:

▷ *Finite Hallucinations:* Generator $\mathcal{G}$ has finite hallucinations for language K if $|\mathcal{G}(S) \setminus K| < \infty$

▷ *Infinite Hallucinations:* Generator $\mathcal{G}$ has infinite hallucinations for K if $|\mathcal{G}(S) \setminus K| = \infty$.

Figure 3 summarizes our characterizations for different notions of breadth (along rows: exact breadth, approximate breadth, no breadth) and different amoungs of hallucinations (along columns: no hallucinations, finite hallucinations, and infinite hallucinations).

	No Hallucinations $\lvert\text{supp}(G)\backslash K\rvert = 0$	**Finite Hallucinations** $\lvert\text{supp}(G)\backslash K\rvert < \infty$	**Infinite Hallucinations** $\lvert\text{supp}(G)\backslash K\rvert = \infty$
Zero Missing Elements $\lvert K\backslash\text{supp}(G)\rvert = 0$	Angluin's Condition *(i.e., Exact Breadth)*	Weak Angluin's Condition	All Countable Collections
Finite Missing Elements $\lvert K\backslash\text{supp}(G)\rvert < \infty$	Weak Angluin's Condition *(i.e., Approximate Breadth)*	Weak Angluin's Condition	All Countable Collections
Infinite Present Elements $\lvert K\cap\text{supp}(G)\rvert = \infty$	All Countable Collections *(i.e., Infinite Coverage)*	All Countable Collections	All Countable Collections

Figure 3: *Characterizations of All Possible Notions of Generation:* This figure lists all possible notions of language generation (at a certain granularity) and their characterizing conditions. Rows capture breadth (*i.e.*, how many elements are missed from the target language). Columns capture the extent of hallucinations (*i.e.*, how many elements outside of the target language are included). Generation becomes easier when moving down rows and/or right along columns. The notion of infinite coverage requires $|K \cap \text{supp}(G)| = \infty$ (Definition E.3).

Proof Sketch for Results in Figure 3. To achieve notions in the last column, one can generate the whole domain (*i.e.*, ensure $\text{supp}(\mathcal{G}) = \mathcal{X}$). To achieve notions in the last row, one can use an extension of (Kleinberg & Mullainathan, 2024)'s algorithm from Proposition E.4. It remains to explain the results in the top 2×2 cells. Among these the two results in the first column are from Theorem A.3. For the remaining two results (the first two in the second column): the lower bound follows from Appendix B.2 since both of these notions satisfy finite non-uniqueness (Definition 3.9). The upper bounds are presented in Appendix C. □

A.5. Generation with Breadth, Stability, and Hallucinations

Next, as in the previous section, to illustrate the generality of our techniques. For stable generators, we use them to give necessary and/or sufficient conditions for several new notions of generation with stable generators. Figure 4 summarizes our results for different notions of breadth with stable generators (along rows: exact breadth, approximate breadth, no breadth) and different amoungs of hallucinations (along columns: no hallucinations, finite hallucinations, and infinite hallucinations). Interestingly, our results also show that if we allow for finitely many hallucinations while missing no elements from the target language, stable generation is still characterized by the weak Angluin's condition.

Unlike the case of unstable generation, we do not have a complete characterization for every cell of Figure 4. It is an interesting direction to characterize all the remaining cells.

Proof Sketch for Results in Figure 4. To achieve any notion in the last column, it is sufficient to generate the whole domain (*i.e.*, ensure $\text{supp}(\mathcal{G}) = \mathcal{X}$). Unlike the case of unstable generators, achieving the notions in the last row is non-trivial. In particular, we show that there exists a collection for which no stable algorithm can achieve infinite coverage (Theorem E.7). It remains to overview the results in the top left 2×2 cells. Among these, the two results in the first column are from Theorem A.5. For the remaining two results: the lower bound follows from Appendix B.2 since both notions satisfy finite non-uniqueness (Definition 3.9) and the upper bound algorithms is as below:

> The algorithm that achieves these notions is straightforward adaptation of Lemma C.3 that does not drop the elements $S_t \cup \{x_1, \ldots, x_t\}$ from the set it outputs.

□

Stable Generators	No Hallucinations $\lvert G(S_t)\backslash K\rvert = 0$	Finite Hallucinations $\lvert G(S_t)\backslash K\rvert < \infty$	Infinite Hallucinations $\lvert G(S_t)\backslash K\rvert = \infty$
Zero Missing Elements $\lvert K\backslash G(S_t)\rvert = 0$	Angluin's Condition [Ang 80] *(i.e., Exact Breadth)*	Weak Angluin's Condition [KMV 24b, CP 24]	All Countable Collections
Finite Missing Elements $\lvert K\backslash G(S_t)\rvert < \infty$	Angluin's Condition [Ang 80] *(i.e., Approximate Breadth)*	Weak Angluin's Condition [KMV 24b, CP 24]	All Countable Collections
Infinite Present Elements $\lvert K \cap G(S_t)\rvert = \infty$	Characterization ? *(Not all countable collections)*	Characterization ?	All Countable Collections

Figure 4: *Stability Under All Possible Notions of Generation:* This figure lists all possible notions of language generation (at a certain granularity). Rows capture the extent of breadth (*i.e.*, how many elements are missed from the target language). Columns capture the extent of hallucinations (*i.e.*, how many elements outside of the target language are included). Generation becomes easier as one moves down the rows and/or to the right along columns. For the yellow cell, we have shown that not all countable collections admit a stable generator that satisfies this notion of breadth, but we do not have a condition that characterizes it. For the gray cell, we do not know whether all collections satisfy this notion, and we do not have a characterization. The notion of infinite coverage refers to a generator whose support satisfies $|K \cap \text{supp}(G)| = \infty$ (see also Definition E.3).

B. Proofs of Lower Bounds

In this section, we prove our lower bound results.

B.1. Lower Bound with Uniqueness (Proof of Theorem 3.5)

In this section, we prove Theorem 3.5. Recall that this requires to prove that the following: if a collection $\mathcal{L}$ violates Angluin's condition, then no generator can achieve a property P satisfying uniqueness in the limit for $\mathcal{L}$.

Proof of Theorem 3.5. For any enumeration E, we use the notation $E(i)$ to denote its i-th element, $E(1:i)$ to denote its first i elements, and $E(i:\infty)$ to denote all but the first $i-1$ elements. Since $\mathcal{L}$ is not identifiable in the limit, it does not satisfy Angluin's condition (Definition 2.2). Hence, there exists a language $L^* \in \mathcal{L}$ such that the following holds:

$$\text{for all finite subsets } T \subseteq L^*\,, \quad \text{there exists a language } L_T \in \mathcal{L}\,, \quad T \subseteq L_T \quad \text{and} \quad L_T \subsetneq L^*\,. \tag{3}$$

Fix $L^* \in \mathcal{L}$ to be any language for which this holds. Let E_*^∞ be an arbitrary enumeration of L^*, without repetitions. Let K and E_K^∞ respectively denote the target language and its enumeration that we will construct to show the impossibility result.

We will show that for any generating algorithm $\mathcal{G} = (\mathcal{G}_n)$ there exists a choice of the target language K in $\mathcal{L}$ (which may be different from L^*) and an enumeration of it such that if K is the target language and the adversary provides enumeration E_K^∞ to $\mathcal{G}$, then the algorithm $\mathcal{G}$ cannot generate with breadth in the limit.

We will construct the enumeration iteratively and select K based on the generating algorithm. The construction of the enumeration proceeds in multiple (possibly infinite) phases. At any point $t \in \mathbb{N}$ of the interaction, we denote by S_t the set of elements enumerated so far.

Phase 1 of Construction. To construct the first phase, we present the generator with the first element of the enumeration of L^*, *i.e.*, $x_{i_1} := E_*^\infty(1)$. Let L_{j_1} be some language such that $x_{i_1} \in L_{j_1}$ and $L_{j_1} \subsetneq L^*$, *i.e.*, it is a proper subset of L^*. Notice that such a language is guaranteed to exist by picking $T = \{x_{i_1}\}$ in the violation of Angluin's condition (3).

- **Subphase A (Enumerate L_{j_1} Until Generator Generates with Breadth from L_{j_1}):** Consider an enumeration E_1^∞ of the language L_{j_1} that is constructed by traversing E_*^∞ and using the elements of L_{j_1} that appear in it, in the same

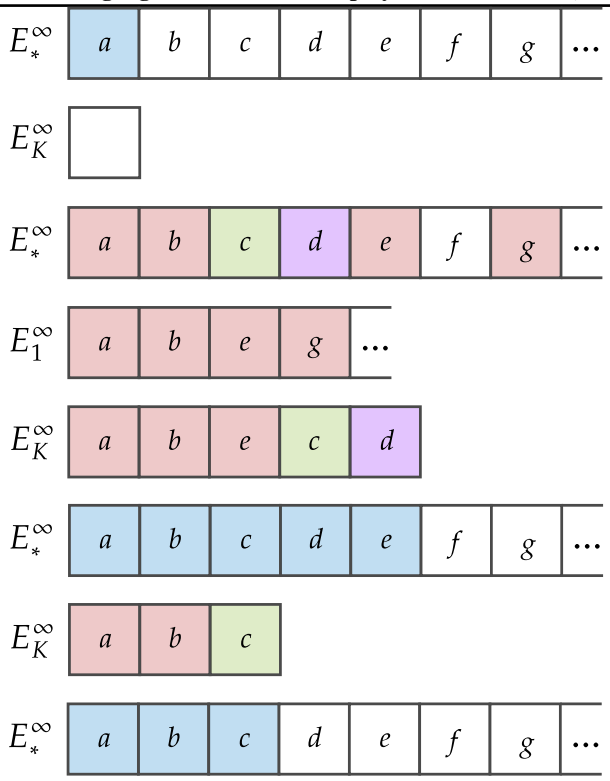


Figure 5: *Illustration of the Construction in the Proof of Theorem 3.5.* Fix any enumeration $a, b, c, d, e, f, g, \ldots$ of the language L^*, depicted in the first row. The enumeration of K is initially empty in the construction and this is depicted in the second row. To begin the construction, we apply the contrapositive to Angluin's condition with $T = \{a\}$ (*i.e.*, with the set highlighted in blue in the first row). This results in a language L_1 that contains T and is a subset of L^*. For this illustration, suppose that the enumeration of L_1 is as presented in the fourth row. The elements shared between L_1 and L^* are highlighted in red in the third row. From the fourth row, we can see that the strings in L_1's enumeration, E_1^*, follow the same relative order as in E_*^∞. Further, note that c, d, and f are skipped from the enumeration since they do not belong to L_1 (*i.e.*, they are not highlighted in red). Now, the algorithm in the proof is trained on the enumeration E_1^∞ (Subphase A), and we consider two cases: **Case (i):** Assume that after seeing element e, the algorithm achieves property P. Then we update E_K^∞ by adding all elements of E_1^∞ until e and then add all the elements that we skipped from E_*^∞; this is shown in the fifth row where we added c and d. This scenario corresponds to Subphase B.1 in the proof since at least one element from the enumeration of E_*^∞ was skipped during Subphase A. Next, we again apply the contrapositive to Angluin's condition. This time, we set $T = \{a, b, e, c, d\}$ (denoted in blue in the sixth row), and, then repeat the process. **Case (ii):** Assume that the algorithm achieves property P after seeing b. Then, we update E_K^∞ by adding a, b and then the first element that is not in L_1, *i.e.*, c. This is depicted in the seventh row. This scenario corresponds to Subphase B.2 in the proof since no strings from E_*^∞ were skipped during Subphase A. Next, we again apply the contrapositive to Angluin's condition. This time, we use $T = \{a, b, c\}$ (denoted in blue in the last row) and repeat the process.

order as they appear, *i.e.*, for every $i \in \mathbb{N}$ it holds that $E_1^\infty(i)$ is the i-th element of L_{j_1} that appears in E_*^∞. Notice that this is indeed a valid enumeration of L_{j_1} as L_{j_1} is a subset of L^*. At any round t of the first phase, the adversary presents the element $E_1^\infty(t)$ to the generator.

Consider two cases: i) either there is some finite $t_1 \in \mathbb{N}$ such that $\mathcal{G}_{t_1}$ achieves property P for L_{j_1} or ii) there is no such $t_1 \in \mathbb{N}$. In the latter case, we pick the target language $K = L_{j_1}$ and the target enumeration $E_K^\infty = E_1^\infty$, and the lower bound follows since we have found a pair of K and E_K^∞ for which the generator never achieves property P. Hence, assume that we are in the former case, and let $\widehat{x}_1$ be the first element of E_1^∞ for which the condition holds. Note that, at this point, $\mathcal{G}_{t_1}$ does *not* achieve property P for L^* since P satisfies the uniqueness criterion and $L_{j_1} \neq L^*$. Further, note that S_{t_1} is the set of strings shown to the generating algorithm after which it starts to generate with breadth from L_{j_1}.

Let $\widehat{S}_1$ be the set of elements of E_*^∞ that appear before $\widehat{x}_1$ in E_*^∞ and have not appeared in S_{t_1}. If $\widehat{S_1} \neq \emptyset$, we go to Subphase B.1 and, otherwise if $\widehat{S_1} = \emptyset$, we go to Subphase B.2.

- **Subphase B.1 (Add Any Skipped Elements):** We will use the set $\widehat{S}_1$ to extend the construction of the target enumeration E_K^∞. To do this, we enumerate the elements from $\widehat{S}_1$ in an arbitrary order and we fix the prefix of the target enumeration E_K^∞ to be $(S_{t_1}, \widehat{S}_1)$. Notice that this step is well-defined since we are only adding to the already constructed enumeration. Let $\widehat{t}_1$ be the total number of elements enumerated so far. Notice that $\widehat{t}_1 = \infty$ if and only if Case i) (from Subphase A) holds, in which case the lower bound already follows. Hence, assume for the continuation of the proof that $\widehat{t}_1 < \infty$. Now we terminate the first phase (without going to Subphase B.2).
- **Subphase B.2 (If Nothing Skipped Enumerate An Element Outside L_{j_1}):** Notice that $\widehat{S}_1 = \emptyset$ if and only if we did not skip any element of E_*^∞ during the traversal in Subphase A. If we indeed did not skip elements of E_*^∞ we continue traversing it and adding elements to E_K^∞ in the same order as we see them in E_*^∞ until we find some element that does not belong to L_{j_1}. We also include this element in the enumeration E_K^∞, we fix $\widehat{t}_1$ to be the number of elements enumerated so far and we terminate the first phase.

Notice that so far in our construction, we have enumerated the first $\widehat{t}_1$ elements of E_*^∞.

Now we continue our construction inductively for phases $\ell = 2, 3, \ldots$. Consider any $\ell \geq 2$. Suppose our construction continued from Phase 1 until Phase ℓ. Then, Phase $\ell + 1$ of our construction is as follows.

Phase $\ell + 1$ of Construction. For the $(\ell + 1)$-th phase, consider the set $E_*^\infty(1 : \widehat{t}_\ell)$ that has been enumerated so far. By construction,

$$E_*^\infty(1 : \widehat{t}_\ell) \not\subseteq L_{j_\ell}\,, \quad E_*^\infty(1 : \widehat{t}_\ell) \subseteq L^*\,, \quad \text{and} \quad E_*^\infty(1 : \widehat{t}_\ell) \text{ is finite}\,.$$

We will now apply the violation of Angluin's condition (3) with $T = E_*^\infty(1 : \widehat{t}_\ell)$. This means that there must exist some $j_{\ell+1} \notin \{j_1, j_2, \ldots, j_\ell\}$ such that

$$L_{j_{\ell+1}} \in \mathcal{L}\,, \quad L_{j_{\ell+1}} \subsetneq L^*\,, \quad \text{and} \quad E_*^\infty(1 : \widehat{t}_\ell) \subseteq L_{j_{\ell+1}}\,.$$

We now perform analogs of each subphase in Phase 1.

- **Subphase A (Enumerate $L_{j_{\ell+1}}$ Until Generator Generates with Breadth from $L_{j_{\ell+1}}$):** Consider an enumeration $E_{\ell+1}^\infty$ of $L_{j_{\ell+1}}$ whose first $\widehat{t}_\ell$ strings are $E_*^\infty(1 : \widehat{t}_\ell)$ and whose remaining strings are constructed by traversing $E_*^\infty(\widehat{t}_\ell + 1 : \infty)$ and selecting strings that belong to $L_{j_{\ell+1}}$, in the same order as they appear in E_*^∞. Notice that this is indeed a valid enumeration of $L_{j_{\ell+1}}$ as $L_{j_{\ell+1}}$ is a subset of L^*. At any round t of this phase, the adversary presents the element $E_{\ell+1}^\infty(t + \widehat{t}_\ell)$ to the generator.

  Consider two cases: i) either there is some finite $t_{\ell+1} \geq \widehat{t}_\ell + 1$ such that $\mathcal{G}_{t_{\ell+1}}$ achieves property P for $L_{j_{\ell+1}}$ or ii) there is no such $t_{\ell+1} \in \mathbb{N}$. In the latter case, we pick the target language $K = L_{j_{\ell+1}}$ and the enumeration $E_K^\infty = E_{\ell+1}^\infty$, and the lower bound follows since we have found a pair of K and E_K^∞ for which the generator never achieves property P. Hence, assume that we are in the former case, and let $\widehat{x}_{\ell+1}$ be the first element of $E_{\ell+1}^\infty$ for which the condition holds. Note that, at this point, $\mathcal{G}_{t_{\ell+1}}$ does *not* achieve property P for L^* since P satisfies the uniqueness criterion and $L_{j_{\ell+1}} \neq L^*$. Further, note that $S_{t_{\ell+1}}$ is the set of strings shown to the generating algorithm after which it starts to generate with breadth from $L_{j_{\ell+1}}$.

Let $\widehat{S}_{\ell+1}$ be the set of strings of E_*^∞ that appear before $\widehat{x}_{\ell+1}$ in E_*^∞ and have not appeared in the enumeration $S_{t_{\ell+1}}$. If $\widehat{S}_{\ell+1} \neq \emptyset$, we go to Subphase B.1 and, otherwise if $\widehat{S}_{\ell+1} = \emptyset$, we go to Subphase B.2.

- **Subphase B.1 (Add Any Skipped Elements):** We will use $\widehat{S}_{\ell+1}$ to extend the construction of the target enumeration E_K^∞. To do this, we enumerate the elements from $\widehat{S}_{\ell+1}$ in an arbitrary order and we fix the prefix of the target enumeration E_K^∞ to be $(S_{t_{\ell+1}}, \widehat{S}_{\ell+1})$. Notice that this step is well-defined since we are only adding to the already constructed enumeration. Let $\widehat{t}_{\ell+1}$ be the set of elements enumerated so far. Notice that $\widehat{t}_{\ell+1} = \infty$ if and only if Case i) (from Subphase A) holds, in which case the lower bound already follows. Hence, assume for the continuation of the proof that $\widehat{t}_{\ell+1} < \infty$. Now we terminate the $(\ell + 1)$-th phase without going to Subphase B.2.

- **Subphase B.2 (If Nothing Skipped Enumerate An Element Outside $L_{j_{\ell+1}}$):** Notice that $\widehat{S}_{\ell+1} = \emptyset$ if and only if we did not skip any element of E_*^∞ during the traversal in Subphase A. If we indeed did not skip elements of E_*^∞ we continue traversing it and adding elements to E_K^∞ in the same order as we see them in E_*^∞ until we find some element that does not belong to $L_{j_{\ell+1}}$. We also include this element in the enumeration E_K^∞, we set $\widehat{t}_{\ell+1}$ to be the number of elements enumerated so far and we terminate Phase $\ell + 1$.

Notice that so far we have enumerated the first $\widehat{t}_{\ell+1} > \widehat{t}_\ell + 1$ elements of E_*^∞.

Inductive Argument. As explained, we continue the construction of the target enumeration inductively. If there is some phase ℓ such that Case ii) (in Subphase A) is activated, then the lower bound follows. Let us now assume that Case ii) is not activated for any phase $\ell \in \mathbb{N}$. Then, we have constructed an enumeration of L^* (by construction of the sets S_{t_ℓ} and $\widehat{S}_\ell$ for each $\ell \in \mathbb{N}$) such that $\mathcal{G}_t$ does not achieve property P for L^* for infinitely many $t \in \mathbb{N}$. Now, the lower bound follows by setting the target language $K = L^*$ and the target enumeration to the one we have constructed inductively over all phases. □

B.2. Lower Bound with Finite Non-Uniqueness (Proof of Theorem 3.10)

In this section, we prove Theorem 3.10.

Proof of Theorem 3.10. The proof of this lower bound uses the construction in the proof of Theorem 3.5 with one change: now the language L_T (introduced at the start of the proof) is the language determined by the contrapositive to the weak Angluin's criterion (Definition 3.7) and not the contrapositive to the (usual) Angluin's criterion (Definition 2.2). Concretely, the contrapositive to the weak Angluin's criterion implies that there exists a language $L^* \in \mathcal{L}$ such that the following holds:

$$\forall T \subseteq L^*\,, \quad \exists L_T \in \mathcal{L}\,, \quad \text{such that} \quad T \subseteq L_T\,, \quad L_T \subsetneq L^*\,, \quad \text{and} \quad |L^* \setminus L_T| = \infty\,. \tag{4}$$

We will use this language L^* and proceed with the construction without change.

Having completed the construction, we proceed to the proof. The only place in which the proof uses a property of the criterion for breadth is when it invokes the uniqueness criterion with respect to the pair of languages L_T and L^* (once in Subphase A of each phase). Here, T is the set $E_*^\infty(1)$ in the first phase and $E_*^\infty(1 : \widehat{t}_\ell)$ in the ℓ-th phase. Now, we cannot directly invoke the uniqueness criterion since P does not satisfy it. However, since $|L^* \setminus L_T| = \infty$ and since property P satisfies the finite non-uniqueness criterion, we can conclude that no generator can achieve property P for both L^* and L_T simultaneously, as desired. Hence, we can use the finite non-uniqueness criterion in analyzing each phase of the construction and the result follows as in the proof of Theorem 3.5. □

B.3. Lower Bound for Approximate Breadth with Stability

In this section, we prove the lower bound in Theorem 3.14: we show that if a collection $\mathcal{L}$ violates Angluin's condition, then no generator can generate with approximate breadth from $\mathcal{L}$. (Note that this as a corollary implies that no generator can generate with exact breadth.)

Proof of lower bound in Theorem 3.14. We will use the following corollary of the construction in the previous section.

Corollary B.1. *Let $\mathcal{L}$ be a countable collection of languages that is not identifiable in the limit. Let $\mathcal{G} = (\mathcal{G}_n)$ be a generating algorithm. If $\mathcal{G}$ generates with approximate breadth from $\mathcal{L}$ in the limit, then there is a language $L^* \in \mathcal{L}$, an enumeration E^* of L^*, a sequence of distinct languages $L_{\ell_1}, L_{\ell_2}, \cdots \in \mathcal{L}$, and a* strictly *increasing sequence $t(1), t(2), \cdots \in \mathbb{N}$, such that the following holds.*

- *For each $i \in \mathbb{N}$, L_{ℓ_i} is a proper subset of L^*, i.e., $L_{\ell_i} \subsetneq L^*$; and*
- *Given strings from E^* as input, for each $i \in \mathbb{N}$, $\mathcal{G}_{t(i)}$ generates with approximate breadth from L_{ℓ_i}.*

Consider the construction in in the above corollary. Let $K = L^*$ and suppose that the adversary follows the enumeration E^*.

Let $C_B, C_S \colon \mathbb{N} \to \mathbb{N}$ be two counters: for each t, $C_B(t)$ counts the number of values $1 \le i \le t$ for which $\mathcal{G}_i$ does *not* generate with approximate breadth from L^* and $C_S(t)$ counts the number of values $2 \le i \le t$ for which $\mathrm{supp}(\mathcal{G}_i) \neq$

$\text{supp}(\mathcal{G}_{i-1})$. In other words, $C_B(t)$ is the number of times $\mathcal{G}$ does not generate with approximate breadth from L^* in the first t-steps and $C_S(t)$ is the number of times $\mathcal{G}$ changes its support in the first t-steps.

Toward a contradiction suppose that $\mathcal{G}$ is stable and generates with approximate breadth from K in the limit (when given the enumeration E^*). This, by definition, implies that

$$\lim_{t\to\infty} C_B(t) < \infty \qquad \text{and} \qquad \lim_{t\to\infty} C_S(t) < \infty\,. \tag{5}$$

The former implies that there are only finitely many values of $i \in \mathbb{N}$ such that $\mathcal{G}_{t(i)}$ does not generate with approximate breadth from L^* (where $t(i)$ are from Corollary B.1). Thus, after discarding a sufficiently large finite prefix of $t(i), i \in \mathbb{N}$, and re-indexing we see that there are infinitely many values, say, $\tau(1) < \tau(2) < \cdots \in \mathbb{N}$, such that, for each i, $\mathcal{G}_{\tau(i)}$ generates with approximate breadth from L^* and L_{ℓ_i}. Since $\mathcal{G}_{\tau(i)}$ generates with approximate breadth from both L^* and L_{ℓ_i} and $L_{\ell_i} \subsetneq L^*$, it follows that: for each $i \in \mathbb{N}$,

$$L_{\ell_i} = \text{supp}(\mathcal{G}_{\tau(i)}) \cup R \qquad \text{where} \qquad R \subseteq L^* \setminus \text{supp}(\mathcal{G}_{\tau(i)})\,. \tag{6}$$

Fix any i. Let

$$s(i) \coloneqq \left|L^* \setminus \text{supp}(\mathcal{G}_{\tau(i)})\right|\,.$$

Since $\mathcal{G}_{\tau(i)}$ generates with approximate breadth from L^*, $s(i) < \infty$. We claim that

$$\text{supp}(\mathcal{G}_{\tau(i)}) \neq \text{supp}(\mathcal{G}_{\tau(i+j)}) \qquad \text{for some} \qquad 1 \le j \le S(i) \coloneqq 2^{s(i)} + 1\,. \tag{7}$$

Proof of Equation (7). To see this, toward a contradiction, suppose that

$$\text{supp}(\mathcal{G}_{\tau(i)}) = \text{supp}(\mathcal{G}_{\tau(i+1)}) = \cdots = \text{supp}(\mathcal{G}_{\tau_{i+S(i)}})\,.$$

This combined with Equation (6) implies that, for each $1 \le j \le S(i)$, $L_{\ell_{(i+j)}} = \text{supp}(\mathcal{G}_{\tau(i)}) \cup R_j$ for some finite set $R_j \subseteq L^* \setminus \text{supp}(\mathcal{G}_{\tau(i)})$. Since all of $L_{\ell_1}, L_{\ell_2}, \ldots$ are different, it must hold that all of $R_1, R_2, \ldots, R_{S(i)}$ are different. This is a contradiction since each R_i is a subset of $R_i \subseteq L^* \setminus \text{supp}(\mathcal{G}_{\tau(i)})$ and there are only $S(i) - 1 = 2^{s(i)}$ such subsets.

Completing the Proof. Equation (7) shows that, for each $i \in \mathbb{N}$, starting from the $\tau(i)$-th step, the support of the generator changes after finitely many steps. Since $\tau_1, \tau_2, \ldots, \in \mathbb{N}$ is a strictly increasing and infinite sequence, this implies that the support of the generator changes infinitely often as it is provided more and more examples and, hence, $\lim_{t\to\infty} C_S(t) = \infty$ which contradicts the fact that $\mathcal{G}$ is stable (5). Hence, our assumption that $\mathcal{G}$ is stable and generates with approximate breadth from $\mathcal{L}$ in the limit must be false. Therefore, no stable generator can generate with approximate breadth from any non-identifiable collection. □

C. Proofs of Upper Bounds

In this section, we present new algorithms for generation required in our results (Theorems 3.8, A.3 and A.5 and Figures 3 and 4).

C.1. Functional Upper Bound for Generation with Approximate Breadth

In this section, we present a function[4] that generates with approximate breadth from any countable collection $\mathcal{L}$ satisfying weak Angluin's condition. This establishes the upper bound in Theorem 3.8.

Lemma C.1 (Function for Generation with Approximate Breadth)**.** *Let $\mathcal{L}$ be a countable collection of languages that satisfies Definition 3.7. Then, there exists a generating algorithm that, given access to a membership oracle for $\mathcal{L}$ and a subset oracle for $\mathcal{L}$ (that given indices i, j outputs Yes if $L_i \subseteq L_j$ and No otherwise), generates from $\mathcal{L}$ with approximate breadth in the limit.*

This proof is inspired by the proof of Theorem B.2 in (Kalavasis et al., 2025), the difference is that, instead of using Angluin's condition (Definition 2.2), we use its weakening (Definition 3.7).

[4]Using the terminology of Kleinberg & Mullainathan (2024), we refer to algorithms that have access to certain oracles (beyond membership oracle) specific to the collection $\mathcal{L}$ as functions; reserving the term algorithm for algorithms which only require membership access to languages in $\mathcal{L}$ (*i.e.*, answer to questions of the form "is $s \in L_i$?").

Proof of Lemma C.1. The algorithm $\mathcal{A}$ is illustrated below. This algorithm follows the steps of the generation algorithm of (Kleinberg & Mullainathan, 2024) (see Steps 1 to 5). The only change is in its last Step 6 where it generates a random sample from the set of interest.

for $t \in \{1, 2, \dots\}$ **do:**

1. Observe element x_t and let S_t be the set of all elements observed so far.
2. Construct a *version space* V_t consisting of all languages in $\mathcal{L}_{\leq t}$ consistent with S_t, *i.e.*,
$$V_t \coloneqq \{L_j \colon 1 \leq j \leq t\,,\ \ L_j \supseteq S_t\}\ .$$
 # Define a language $L_i \in V_t$ to be critical *if L_i is the smallest-index language in V_t or L_i is a subset of all languages preceding it in V_t, i.e., $L_i \subseteq L_j$ for all $1 \leq j < i$.*
3. If $V_t = \emptyset$, **output** an arbitrary element of $\mathcal{X}$ and **go** to the next iteration.
4. Construct the set $C_t \subseteq V_t$ of all critical languages.
 # To construct the set of critical languages C_t the algorithm needs access to the subset oracle.
5. Let L_i be the largest-indexed language in the set of critical languages C_t.
6. **output** a sample from any distribution whose support is $L_i \setminus S_t$. This can be done in a computable fashion by first sampling a natural number n from (*e.g.*, the geometric distribution on $\mathbb{N}$) and then outputting the n-th string from $L_i \setminus S_t$.

Let z be the first index such that $K = L_z$. The proposed algorithm generates with approximate breadth from K when after some finite time t^*, and for $t > t^*$, the last language in the set of critical languages C_t, $L_i = L_i(t)$, satisfies that
$$L_i \subseteq K \quad \text{and} \quad |K \setminus L_i| < \infty\,.$$
This condition is implied by the following two conditions.

(A) K is eventually included in set of critical languages C_t and is never removed after that.

(B) Eventually all the languages L_j with $j > z$ that are in C_t satisfy $L_j \subseteq K$ and $|K \setminus L_j| < \infty$.

Result (4.3) of (Kleinberg & Mullainathan, 2024) shows that there is a finite time t_A after which Condition (A) holds. We will show that there is also a finite time t_B after which Condition (B) holds. This shows that, for any $t \geq \max\{t_A, t_B\}$, $\mathcal{A}$ generates with approximate breadth from K.

Condition (B) holds after a finite time. Since $\mathcal{L}$ satisfies the weakening of Angluin's condition (Definition 3.7), $K = L_z$ has a finite tell-tale set T_z, such that, any language $L \in \mathcal{L}$ containing the tell-take T_z satisfies one of the following:

- Either L is not a proper subset of K;
- Or L is a proper subset of K and satisfies $|K \setminus L| < \infty$.

(Recall that T_z is not known to us; our proof will not need this.) Fix any $j > z$ and any time $t_B \geq t_A$ after which K is guaranteed to be a critical language and after which $S_t \supseteq T_z$ (which happens at a finite time since T_z is finite and, so, all elements of T_z appear in the enumeration of K at some finite time). Our goal is to show that for any $t \geq t_B$, and any $j > z$ for which L_j is in C_t, it holds that
$$L_j \subseteq K \quad \text{and} \quad |K \setminus L_j| < \infty\,.$$
By the definition of critical languages and the fact that L_j appears after $K = L_z$ in the set of critical languages (as $j > z$), it follows that $L_j \subseteq K$. Hence, it remains to show that $|K \setminus L_j| < \infty$. To see this, observe that since $L_j \in C_t$ and $C_t \subseteq V_t$, L_j is in the version space V_t and, hence, by the definition of V_t, $L_j \supseteq S_t$. Therefore, in particular, $L_j \supseteq T_z$ (as $S_t \supseteq T_z$). Now, Definition 3.7 combined with the observation that $L_j \subseteq K$ implies that $|K \setminus L_j| < \infty$ as required. □

Building on the result of (Kalavasis et al., 2025) (Corollary B.2 in their paper), the previous result shows that the function[5] of (Kleinberg & Mullainathan, 2024) with access to a subset query oracle achieves the "best-of-three" worlds for generation, without requiring any prior information about $\mathcal{L}$, only subset and membership oracle access.

Corollary C.2. *Let $\mathcal{L}$ be a countable collection of languages. Exactly one of the following holds for the subset-oracle-based function of (Kleinberg & Mullainathan, 2024).*

- *If $\mathcal{L}$ satisfies Angluin's condition, the function generates with exact breadth in the limit.*
- *If $\mathcal{L}$ does not satisfy Angluin's condition but satisfies the weak Angluin's condition, the function generates with approximate breadth in the limit.*
- *If $\mathcal{L}$ does not satisfy the weak Angluin's condition, the function generates with infinite coverage in the limit.*

C.2. Algorithmic Upper Bound for Generation with Approximate Breadth

Next, we give an algorithm that generates with approximate breadth without requiring access to a subset oracle. This establishes an alternate proof of the upper bound in Theorem 3.8.

Lemma C.3 (Algorithm for Generation with Approximate Breadth)**.** *Let $\mathcal{L}$ be a countable collection of languages that satisfies Definition 3.7. Then, there exists a generating algorithm that, given access to a membership oracle for $\mathcal{L}$ and the tell-tale oracle from Definition 3.7, generates from $\mathcal{L}$ with approximate breadth in the limit.*

Proof of Lemma C.3. Let S_n be the set of elements the adversary has enumerated up to round $n \in N$. For every $i, n \in \mathbb{N}$, let T_n^i be the first n elements enumerated from the tell-tale oracle when called on language L_i. Let also $x_1, x_2, \ldots,$ be an enumeration of the domain $\mathcal{X}$. Our proof is reminiscent of Angluin's approach (Angluin, 1980), and the generating algorithm requires only one extra step, namely removing the elements $x_1, \ldots, x_n$ from the support of the outputted distribution. However, due to the relaxed condition we are using, our analysis is more technically involved.

For every round $n \in \mathbb{N}$, the generating algorithm constructs the sets T_n^i using the tell-tale oracle for all languages L_i with $1 \leq i \leq n$. Let $g_n \in \mathbb{N}, 1 \leq g_n \leq n$, be the smallest number (if any) such that $S_n \subseteq L_{g_n}$ and $T_n^{g_n} \subseteq S_n$. If no such number exists, let $\mathcal{G}_n$ be some arbitrary distribution. Otherwise, let $\mathcal{G}_n$ be a distribution with $\mathrm{supp}(\mathcal{G}_n) = L_{g_n} \setminus (S_n \cup \{x_1, \ldots, x_n\})$.[6]

Fix a canonical enumeration $x_1, x_2, \ldots$ of $\mathcal{X}$.

for $n \in \{1, 2, \ldots\}$ **do:**

1. Let S_n be the set of all elements observed so far.
2. Create the list $\mathcal{L}_{\leq n} = \{L_1, \ldots, L_n\}$.
3. For each language L_i in $\mathcal{L}_{\leq n}$, let $T^i = \mathsf{TellTaleOracle}(L_i),\ i \in [n]$.
4. Truncate the outputs of the oracle and keep only their first n elements
$$T_n^i = (T^i(1), \ldots, T^i(n)),\ i \in [n]\,.$$
5. Find smallest index $g_n \in \{1, \ldots, n\}$ such that $S_n \subseteq L_{g_n}$ and $T_n^{g_n} \subseteq S_n$.

   # This is the minimum indexed language in $\mathcal{L}_{\leq n}$ that is consistent and its truncated tell-tale is contained in the observed elements.
6. If no such g_n exists, **output** an arbitrary point from $\mathcal{X}$ and **go** to the next iteration.

[5] To be precise, the function is that of (Kleinberg & Mullainathan, 2024) together with a process to sample from a language given membership access to it; see *e.g.*, Step 6 in the Algorithm of Lemma C.1.

[6] One can sample from this distribution in a computable fashion.

7. Otherwise, define a distribution $\mathcal{G}_n$ with $\text{supp}(\mathcal{G}_n) = L_{g_n} \setminus (S_n \cup \{x_1, \ldots, x_n\})$.

   # The intuition for removing the first n elements $x_1, \ldots, x_n$ of the canonical enumeration of $\mathfrak{X}$ is as follows. A bad scenario for our algorithm is that there exists some language L_{g_n} in the enumeration of $\mathfrak{L}$ before $L_z = K$ such Step 5 will be stuck on L_{g_n}. Then we can guarantee that $|L_{g_n} \setminus K| < \infty$. Since this set is finite, by removing parts of the enumeration of $\mathfrak{X}$ of increasing but finite size, we will eventually remove $|L_{g_n} \setminus K|$, and obtain a sampler that (i) is consistent and (ii) misses only finitely many elements from K.

8. **Output** a sample from the distribution $\mathcal{G}_n$.

We will show that this algorithm generates with approximate breadth in the limit. Let K be the target language and $z \in \mathbb{N}$ be the smallest number such that $L_z = K$. We consider two cases.

Case A ($z = 1$): $S_n \subseteq L_1, \forall n \in \mathbb{N}$ and since the tell-tale set T^1 of L_1 is finite and the adversary presents a complete presentation of K, it holds that $T^1_n \subseteq S_n$ for sufficiently large n. Thus, in the limit, it holds that $g_n = 1$, thus $\text{supp}(\mathcal{G}_n) = L_1 \setminus (S_n \cup \{x_1, \ldots, x_n\})$, and the proof is concluded by noting that $\text{supp}(\mathcal{G}_n) \subseteq K$ and $|S_n \cup \{x_1, \ldots, x_n\}| < \infty$, for all sufficiently large n.

Case B ($z > 1$): We now move on to the case $z > 1$. Then, for every language $L_i, 1 \leq i \leq z-1$, that precedes L_z, exactly one of the following holds:

(i) either there exists some $x_{j_i} \in L_z$ but $x_{j_i} \notin L_i$, or

(ii) $L_z \subsetneq L_i$.

If Case **(i)** holds, then there exists some $n_i \in \mathbb{N}$ such that $S_{n_i} \not\subseteq L_i$. Thus, since there are finitely many languages before z for which Case **(i)** holds, after finitely many $n \in \mathbb{N}$ all of them will have been contradicted by S_n. Thus, we consider some $n_0 \in \mathbb{N}$ large enough so that for all $n \geq n_0$ every language $L_i, 1 \leq i \leq z-1$, for which $S_n \subseteq L_i$ satisfies $L_z \subsetneq L_i$.

Let $\mathcal{I} = \{i_1, \ldots, i_\ell\}$ be the set of the indices for which the previous holds. For every $j \in \mathcal{I}$, and for all $j' \in \mathbb{N}$ for which the tell-tale set of L_j is a subset of $L_{j'}$, *i.e.*, $T^j \subseteq L_{j'}$, one of the following two cases hold by the definition of the weak Angluin's condition: (a) either $L_{j'}$ is not a proper subset of L_j or (b) $|L_j \setminus L_{j'}| < \infty$.

Consider $j' = z$ and any $j \in \mathcal{I}$. Since, by construction, $L_z \subsetneq L_j$, the previous argument shows that either **(I)** $T^j \not\subseteq L_z$ or **(II)** $|L_j \setminus L_z| < \infty$.

If j falls into Case **(I)** then for large enough n it holds that $T^j_n \not\subseteq L_z$, thus $T^j_n \not\subseteq S_n$, and due to the way we have defined g_n, $g_n \neq j$.[7] Thus, we let $\mathcal{I}'$ be the set of indices $j \in \mathbb{N}, 1 \leq j \leq z-1$, such that $T^j \subseteq L_z$ and $L_z \subsetneq L_j$ and, hence, since we fall into Case **(II)** the previous argument implies that $|L_j \setminus L_z| < \infty$ for each $j \in \mathcal{I}'$.

We consider again two cases: if $\mathcal{I}' = \emptyset$, then for large enough n it holds that $g_n = z$. Hence, the correctness follows from the previous arguments.

We now handle the more complicated case $\mathcal{I}' \neq \emptyset$. Let j^* be the first element of $\mathcal{I}'$. For large enough n, the choice of g_n will stabilize to j^*. To see this, notice that $S_n \subseteq L_{j^*}$ for all $n \in \mathbb{N}$, $T^{j^*}_n = T^{j^*}$ for sufficiently large n (since T^{j^*} is finite), and since $T^{j^*} \subseteq L_z$ (and the adversary presents a complete presentation of L_z), for large enough n it holds that $T^{j^*}_n \subseteq S_n$. Thus, indeed for all sufficiently large n it holds that $g_n = j^*$. By definition of $\mathcal{I}'$, it holds that $|L_{j^*} \setminus L_z| < \infty$. Let $x_{\ell_{j^*}}$ be the largest element of the enumeration of $\mathfrak{X}$ for which $x_{\ell_{j^*}} \in L_{j^*}$ but $x_{\ell_{j^*}} \notin L_z$ (this always exists as $j^* \in \mathcal{I}'$ and, hence, $L_z \subsetneq L_{j^*}$ and $|L_{j^*} \setminus L_z| < \infty$.). For $n \geq \ell_{j^*}$ it holds that $L_{j^*} \setminus \{x_1, \ldots, x_n\} \subseteq L_z$. This shows that, indeed, $\text{supp}(\mathcal{G}_n) \subseteq K$, for large enough n, since we set $\text{supp}(\mathcal{G}_n) = L_{j^*} \setminus (S_n \cup \{x_1, \ldots, x_n\})$. Moreover, since $L_z \subsetneq L_{j^*}$, and $|\{x_1, \ldots, x_n\}| < \infty$, it holds that $|L_z \setminus (L_{j^*} \setminus \{x_1, \ldots, x_n\})| < \infty$, for all $n \in \mathbb{N}$. Hence, the generator generates with approximate breadth from K in the limit. □

Remark C.4. The generating algorithm that achieves approximate breadth in the limit for languages that satisfy the weak version of Angluin's condition has the property that the Membership Oracle Problem is decidable. Hence, by the results of (Kalavasis et al., 2025), it cannot be stable, and, indeed, it is not since its support changes at each iteration.

[7] Observe that if we had assumed the stronger Definition 2.2 (Angluin's condition), then this step implies that we can identify L_z in the limit, since only Case (I) is valid. This is exactly how the tell-tale-based algorithm of (Angluin, 1980) works.

C.3. Extensions to Other Notions of Breadth

In this section, we generalize the results from the previous two sections to give algorithms that achieve exhaustive generation for countable collections satisfying weak Angluin's condition.

We first give a function that achieves exhaustive generation.

Lemma C.5 (Function for Exhaustive Generation)**.** *Let $\mathcal{L}$ be a countable collection of languages that satisfies Definition 3.7. Then, there exists a generating algorithm that, exhaustively generates from $\mathcal{L}$ (and is consistent with the target language) in the limit. The algorithm uses access to the following oracles:*

- ▷ *a membership oracle for $\mathcal{L}$,*
- ▷ *a subset oracle for $\mathcal{L}$ (that given indices i, j outputs Yes if $L_i \subseteq L_j$ and No otherwise),*
- ▷ *a finite difference oracle for $\mathcal{L}$ (that given indices i, j with $L_i \subset L_j$ outputs Yes if $|L_j \setminus L_i| < \infty$ and No otherwise).*

The generation in the above result satisfies a property stronger than Definition A.2:

Remark C.6. In addition to achieving exhaustive generation, the generator is consistent with the target language and, hence, does not have *any* hallucinations.

The generator in Lemma C.5 is as follows.

Fix the following: a special character $x_0 \notin \mathcal{X}$ and a canonical enumeration $x_1, x_2, \dots$ of $\mathcal{X}$.
Initialize $\ell_0 = 0$.
for $t \in \{1, 2, \dots\}$ **do:**

1. Observe element x_t and let S_t be the set of all elements observed so far.
2. Construct a *version space* V_t consisting of all languages in $\mathcal{L}_{\leq t}$ consistent with S_t, *i.e.*,
$$V_t \coloneqq \{L_j \colon 1 \leq j \leq t\,,\ \ L_j \supseteq S_t\}\ .$$
3. If $V_t = \emptyset$, **output** an arbitrary element of $\mathcal{X}$ and **go** to the next iteration.

   # Define a language $L_i \in V_t$ to be critical *if L_i is the smallest-indexed language in V_t or L_i is a subset of all languages preceding it in V_t, i.e., $L_i \subseteq L_j$ for all $1 \leq j < i$.*
4. Construct the set $C_t = \{L_{i_1^t} \supseteq L_{i_2^t} \supseteq \cdots \supseteq L_{i_j^t}\} \subseteq V_t$ of critical languages for some $j \leq t$.

   # To construct the set of critical languages C_t the algorithm needs access to the subset oracle.
5. Find the smallest indexed language $L = L(t)$ in C_t such that $|L \setminus L_{i_j^t}| < \infty$. Create the set C'_t by removing all the languages in C_t before L.

   # To perform this filtering, the algorithm needs access to the finite difference oracle.
6. If $C'_t = \emptyset$, **output** an arbitrary element of $\mathcal{X}$ and **go** to the next iteration.
7. Let $L_i = L_{i(t)}$ be the minimum indexed language in the set of filtered critical languages C'_t.
8. If $i(t) \neq i(t-1)$, set $\ell_t = 0$; else $\ell_t = \ell_{t-1} + 1$.
9. **output** the enumeration of $L_i \setminus \{x_0, \dots, x_{\ell_t}\}$ induced by the canonical enumeration of $\mathcal{X}$ fixed at the start.

Proof of Lemma C.5. We will show that the above function exhaustively generates and is consistent with the true language in the limit. Let K be the target language and $z \in \mathbb{N}$ be the smallest number such that $L_z = K$. We will use the case analysis of Lemma C.3. Fix some symbol $x_0 \notin \mathcal{X}$.

Case A ($z = 1$)**:** Since $z = 1$, the true language is the first critical language and is never filtered from C'_t. Moreover, the counters ℓ_t will never be reset (in Step 8) and, in fact, satisfy $\ell_t = t$. Hence, for each $t \in \mathbb{N}$, the algorithm $\mathcal{G}_t$ enumerates

the set $K \setminus (S_t \cup \{x_0, \ldots, x_t\})$ induced by the canonical enumeration of $\mathcal{X}$. It follows that, for each removed x_i, there is some t where it is the first element of the output enumeration. Further, the output enumeration is always consistent with K. Hence, the resulting generator exhaustively generates K. In fact, it has the stronger property that it never hallucinates.

Case B ($z > 1$)**:** Consider the languages before L_z in the enumeration of $\mathcal{L}$. There are two cases: For any $i < z$, either there exists an element that belongs to L_z but not L_i or $L_z \subseteq L_i$. If the first case holds, then eventually the distinguishing element will appear in the enumeration of K and make L_i inconsistent. Hence, let us assume that for all $i < z$, we only care about indices i for which $L_i \supsetneq L_z$. We claim that eventually the index of Step 5 stabilizes in the limit. In particular, we will show that it stabilizes to the smallest index i^* such that $L_{i^*} \supseteq L_z$ and $|L_{i^*} \setminus L_z| < \infty$; note that if there is no language $L_i \supsetneq L_z$, then i^* must be z. Before proving this claim, we show that it implies the result. Let $1 \leq i^* \leq z$ be the index that Step 5 eventually stabilizes on. We know that $L_{i^*} \supseteq K$ (by our earlier argument that any index $1 \leq i \leq z$ not satisfying this property is eliminated after a finite time) and $|L_{i^*} \setminus K| < \infty$ (by construction). We now show how to exhaustively generate K in the limit, this corresponds to Steps 8 and 9 of the above function. To see this, observe that as $|L_{i^*} \setminus K| < \infty$, after a finite number of steps $L_{i^*} \setminus \{x_0, \ldots, x_{\ell_t}\} \subseteq K$ (and, hence, the algorithm eventually stops hallucinating). Further, since at step t (for large enough t), we output the enumeration of $L_{i^*} \setminus \{x_0, \ldots, x_{\ell_t}\}$ induced by the canonical enumeration of $\mathcal{X}$, it follows, for each removed x_i, there is some t where it is the first element of the output enumeration. Hence, the resulting generator exhaustively generates K. In fact, it has the stronger property that it eventually stops making any hallucinations.

Proof of the claim. It remains to prove our claim that the index of Step 5 stabilizes in the limit. Since $\mathcal{L}$ satisfies the weak Angluin's condition, then K has a finite tell-tale set T_K. We condition on the following events: (A) K is a critical language, and (B) $S_t \supset T_K$. Condition (A) is satisfied for any $t \geq z$ and (B) is satisfied after a finite time since T_K is finite and all its elements appear at a finite point in the enumeration of K. Conditioned on these events the critical list C_t is of the form

$$L_{i_1^t} \supseteq L_{i_2^t} \supseteq \cdots \supseteq K \supseteq L_{j_1^t} \supseteq \ldots$$

First, observe that there are finitely many languages before K in this list: this is because K appears at a finite point in this list. Next, we claim that conditioned on the above events the indices $i_1^t, i_2^t, \ldots$ of the languages appearing *before* K in the list never change. The proof is via induction.

- *Base Case:* First, consider the first index i_1^t. It is defined as the smallest index language consistent with S_t. Moreover, due to the structure above it has the property that $L_{i_1^t} \supseteq K$ and, hence, it never becomes inconsistent with $S_{t'}$ for $t' \geq t$. Therefore, the index i_1^t never changes in subsequent steps.
- *Induction Step:* Next, we complete the induction argument, suppose indices $i_1^t, i_2^t, \ldots, i_r^t$ never change in subsequent steps, then we claim that the index i_{r+1}^t (if it exists) also never changes in subsequent steps. This is because i_{r+1}^t is defined as the smallest indexed language that is (1) consistent with S_t and (2) has the property that $L_{i_{r+1}^t} \subseteq L_{i_r^t}$. The former always holds for all subsequent $t' \geq t$ since $L_{i_{r+1}^t} \supseteq S_t \supseteq T_K$ and the latter holds for all subsequent $t' \geq t$ since i_r^t never changes.

Now we are ready to prove that the index $i(t)$ selected in Step 5 stabilizes. Recall that $i(t)$ is the smallest index satisfying that (1) $L_{i(t)}$ appears before K in the critical list and (2) $|L_{i(t)} \setminus L_{i_j^t}| = |L_{i(t)} \setminus K| + |K \setminus L_{i_j^t}| < \infty$. Observe that $|L_{i(t)} \setminus L_{i_j^t}| = |L_{i(t)} \setminus K| + |K \setminus L_{i_j^t}|$ and, by construction, $|K \setminus L_{i_j^t}| < \infty$ and, therefore, Condition (2) is equivalent to $|L_{i(t)} \setminus K| < \infty$. Fix any t satisfying Conditions A and B above and the corresponding $i(t)$. For all subsequent $t' \geq t$, $L_{i(t)}$ continues to appear before K in the critical list since we proved that all indices before K in the critical list stabilize. Further, $|L_{i(t)} \setminus K| < \infty$ since it is independent of t'. Therefore, $i(t) = i(t')$ since $i(t)$ satisfies both properties that determine $i(t')$. It follows that for $t' \geq t$, the index selected in Step 5 never changes. □

Moreover, a small adaptation of the proof of Lemma C.3 gives a generator that generates exhaustively (Definition A.2) in the limit provided one has access to the tell-tale oracle from Definition 3.7.

Lemma C.7 (Algorithm for Exhaustive Generation)**.** *Let $\mathcal{L}$ be a countable collection of languages that satisfies Definition 3.7. Then, there exists a generating algorithm that, given access to a membership oracle for $\mathcal{L}$ and the tell-tale oracle from Definition 3.7, exhaustively generates from $\mathcal{L}$ in the limit.*

Proof of Lemma C.7. The argument in the proof of Lemma C.3 shows that the choice of the index g_n stabilizes in the limit. Moreover, $K \subseteq L_{g_n}$ and $|L_{g_n} \setminus K| < \infty$. To achieve exhaustive generation, the only modification needed is that we keep

track of another index ℓ_n which is initialized at 0, increases by 1 in every round, and every time the choice of g_n changes, we reset $\ell_n = 0$. The enumeration we output is $L_{g_n} \setminus \{x_0, \ldots, x_{\ell_n}\}$, where we use the notational convention that x_0 is some special element that does not appear in $\mathcal{X}$. Moreover, the sequence in which the element appears in the enumeration is the natural order induced by (some canonical) enumeration of $\mathcal{X}$. Assume that n is large enough so that g_n has stabilized. It is easy to see two things: for every element $\widehat{x}$ of L_{g_n}, there exists some finite round $\widehat{n} \in \mathbb{N}$ such that $\widehat{x}$ is the first element in the enumeration we have outputted. Moreover, since $L_z \subseteq L_{g_n}$ and $|L_{g_n} \setminus L_z| < \infty$, after some finite $n \in \mathbb{N}$ it holds that $L_{g_n} \setminus \{x_0, \ldots, x_{\ell_n}\} \subseteq L_z$. Moreover, every time an element x_i is omitted from the enumeration we output, there has been some prior iteration where it has been the first element in the enumeration. These arguments show that the modified generator is an exhaustive generator for $\mathcal{L}$. □

D. Implication for the Statistical Setting

In this section, we discuss the implications of our results in the statistical setting.

In this setting, there is a countable language collection $\mathcal{L}$, a "valid" distribution $\mathcal{P}$ supported on a language $K \in \mathcal{L}$, and the generating algorithm takes as input string drawn i.i.d. from $\mathcal{P}$. For every different notion of breadth, one can define an error function for the generating algorithm $(\mathcal{G}_n)_{n\in\mathbb{N}}$ as

$$\operatorname{er}(\mathcal{G}_n) = \mathbb{1}\{\neg P(\mathcal{G}_n)\}\,, \tag{8}$$

where $P(\cdot)$ is a predicate defined based on the underlying notion of breadth and its value is True if the breadth property is achieved by $\mathcal{G}_n$ and False, otherwise.

Given this definition (8), (Kalavasis et al., 2025) define the error rate for generation with breadth via the universal rates framework of Bousquet et al. (2021).

Definition D.1 (Error Rate (Bousquet et al., 2021))**.** *Let $\mathcal{L}$ be a countable collection of languages,* er *be an error function defined in Equation (8), and $R\colon \mathbb{N} \to [0,1]$ be a rate function such that $\lim_{n\to\infty} R(n) = 0$. We say that rate $R(\cdot)$ is achievable for $\mathcal{L}$ if there exists a generating algorithm $\mathcal{G} = (\mathcal{G}_n)$ such that*

$$\forall\, \mathcal{P} \in \mathrm{Val}(\mathcal{L})\ \ \exists\, C, c > 0 \quad \textit{such that} \quad \mathbb{E}\left[\operatorname{er}(\mathcal{G}_n)\right] \le C \cdot R(c \cdot n) \quad \forall n \in \mathbb{N}\,,$$

where $\mathrm{Val}(\mathcal{L})$ the set of all valid distributions with respect to $\mathcal{L}$. Conversely, we say that no rate faster than $R(\cdot)$ is achievable for $\mathcal{L}$ if for any generating algorithm $\mathcal{G} = (\mathcal{G}_n)$ there exists a valid distribution $\mathcal{P}$ and $c, C > 0$ such that $\mathbb{E}\left[\operatorname{er}(\mathcal{G}_n)\right] \ge C \cdot R(c \cdot n)$, for infinitely many $n \in \mathbb{N}$. We say that no rate is achievable for $\mathcal{L}$ if for any generating algorithm $\mathcal{G} = (\mathcal{G}_n)$ there exists a valid distribution $\mathcal{P}$ such that $\limsup_{n\to\infty} \mathbb{E}\left[\operatorname{er}(\mathcal{G}_n)\right] > 0$.

(Kalavasis et al., 2025) proved bounds in this statistical setting for language identification, generation with exact breadth for algorithms for which the MOP is decidable,[8] and generation with approximate breadth for algorithms that are stable in the limit,[9] and for which the MOP is decidable. To get these results, (Kalavasis et al., 2025) showed connections between the online setting considered in the previous sections and the statistical setting. Using the new results in this work, and the results of (Kalavasis et al., 2025), we can get characterizations for the statistical rates under these two notions of breadth removing the requirement for decidability of the MOP oracle and stability of the generating algorithm.

Theorem D.2 (Rates for Generation with Exact Breadth)**.** *For any non-trivial collection of languages $\mathcal{L}$ no rate faster than e^{-n} is achievable for generation with exact breadth. Moreover, For any collection that is identifiable in the limit, there exists an algorithm that achieves generation with exact breadth at rate e^{-n}. Conversely, for any non-identifiable collection, no rate is achievable for generation with exact breadth.*

For the non-triviality requirement, we refer the interested reader to (Kalavasis et al., 2025). The e^{-n} lower bound and upper bound follow immediately from their results. The lower bound for no rates achievable follows from the approach of (Kalavasis et al., 2025) (with a few modifications in their construction) and Theorem 3.5. For brevity, we only sketch the modifications here:

[8] Recall this is a mild technical condition that requires that the generating algorithm can answer queries about whether a string x is in its support.

[9] Roughly speaking, stability means that after finitely many steps, the support of the distribution outputted by the generating algorithm does not change. For the formal definition, see Definition 3.13.

- (Kalavasis et al., 2025) make use of a construction of (Angluin, 1988) which connects the adversarial setting "in-the-limit" to the statistical setting "in-the-limit" (Theorem 5.6 in their paper) for language identification. A similar result can be shown for generation with exact breadth.

- (Kalavasis et al., 2025) make use of majority votes over learners that identify the target language. In Lemma 5.8 they use the voting scheme, (a modification of) Angluin's result (Angluin, 1988), and the Borel-Cantelli lemma to show that no rate is achievable for language identification, for collections that do not satisfy Angluin's criterion (Definition 2.2). The same approach can be used to derive the lower bound for generation with exact breadth, by using a slightly different majority voting scheme. At a very high level, following (Kalavasis et al., 2025)[10] we split the dataset into different batches and train the generating algorithm, and we can show that for large enough n, a c-fraction of these generators satisfies the generation with exact breadth property (for, *e.g.*, $c > 2/3$). In order to combine their outputs, we define an (implicit) distribution as follows: we keep sampling from all the batches until a c-fraction of them outputs the same element. It is not hard to see that **(i)** this process terminates in finite time,[11] **(ii)** only elements of K have positive probability of being outputted, **(iii)** every element of K has a positive probability of being outputted.

A similar result can be obtained for language generation with approximate breadth, using the criterion from Definition 3.7.

Theorem D.3 (Rates for Generation with Approximate Breadth)**.** *For any non-trivial collection of languages $\mathcal{L}$ no rate faster than e^{-n} is achievable for generation with approximate breadth. For any collection that satisfies Definition 3.7, there exists an algorithm that achieves generation with approximate breadth at rate e^{-n}. Conversely, for any collection that does not Definition 3.7, no rate is achievable for generation with exact breadth.*

The above pair of results provides statistical rates for language generation with exact and approximate breadth. Obtaining statistical rates for unambiguous generation is an interesting direction.

E. Further Results

In this section, we give results for language generation with new notions of breadth and stability.

Outline. In Appendix E.1, we introduce a notion of infinite coverage which weakens approximate breadth and show that it is achievable for all countable collections. In Appendix E.2, we study generation with infinite coverage with stable generators: (1) we show that it cannot be achieved for all countable collections (Appendix E.2.1), and (2) we give a sufficient condition to achieve it (Appendix E.2.2). In Appendix E.3, we present a strengthening of stability, which we call increasing coverage, and show that it can be achieved for certain collections.

Remark E.1 (Characterizations for Existing Notions of Breadth with Stability). We present the characterizations of existing notions of breadth with stability in Appendix A.3. In this section, we discuss characterizations for new notions of breadth and a strengthening of stability.

Remark E.2 (Results allowing for Hallucinations). We refer the reader to Appendices A.4 and A.5 for results on language generation with breadth when some amount of hallucination is allowed.

E.1. Generation with Infinite Coverage

In this section, we provide further motivation behind Definition 3.2, generation with approximate breadth. An immediate modification of the algorithm of (Kleinberg & Mullainathan, 2024) can achieve *finite coverage* of the target language, for any finite number. More concretely, for any function $f\colon \mathbb{N} \to \mathbb{N}$ and any countable collection of languages $\mathcal{L}$ there exists a generating algorithm $(\mathcal{G}_n)_{n\in\mathbb{N}}$ such that, for any target language $K \in \mathcal{L}$ and any enumeration of K the algorithm achieves in the limit

$$\mathrm{supp}(\mathcal{G}_n) \subseteq K\,, \qquad \mathrm{supp}(\mathcal{G}_n) \cap S_n = \emptyset\,, \qquad \text{and} \qquad |\mathrm{supp}(\mathcal{G}_n)| = f(n)\,,$$

where S_n is the set of elements enumerated until round n. In fact, their algorithm can achieve the stronger property of *infinite coverage* defined below.

[10] The same approach has been used extensively in the universal rates literature, starting from (Bousquet et al., 2021).

[11] One small complication is that if a c-fraction does not satisfy the desired property, the algorithm might not terminate. To fix that, in every step we either terminate with probability $1/2$ or we do the sampling strategy we described with probability $1/2$. If we terminate, we run the algorithm from (Kleinberg & Mullainathan, 2024) to generate a valid string from K.

Definition E.3 (Language Generation with Infinite Coverage in the Limit)**.** *A generating algorithm $\mathcal{G} = (\mathcal{G}_n)$ is said to generate with infinite coverage in the limit for a language collection $\mathcal{L} = \{L_1, L_2, \dots\}$ if, for any $K \in \mathcal{L}$ and enumeration of K, there is an $n^* \geq 1$, such that for all $n \geq n^*$, after seeing n elements of the enumeration (corresponding to the set S_n in round n),*

$$\operatorname{supp}(\mathcal{G}_n) \subseteq K\,, \quad \operatorname{supp}(\mathcal{G}_n) \cap S_n = \emptyset\,, \quad \textit{and} \quad |\operatorname{supp}(\mathcal{G}_n)| = \infty\,,$$

Given the above notion of infinite coverage, a simple modification to the generating algorithm of (Kleinberg & Mullainathan, 2024) gives the following result.

Proposition E.4 (Modification of (Kleinberg & Mullainathan, 2024))**.** *There is a generating algorithm with the property that for any countable collection of languages $\mathcal{L} = \{L_1, L_2, \dots\}$, any target language $K \in \mathcal{L}$, and any enumeration of K, the algorithm generates with infinite coverage from K in the limit.*

Thus, the aforementioned modification of the algorithm of (Kleinberg & Mullainathan, 2024) has the property that it does not hallucinate (*i.e.*, it does not include any elements outside of K in its support) and covers infinitely many (unseen) elements of the target language, but might, potentially, not cover infinitely many elements as well. Thus, a natural question is whether there exists an algorithm that does not hallucinate, can cover infinitely many elements of K, and also miss only *finitely* many elements of it. This is precisely the requirement of generation with approximate breadth (Definition 3.2).

Proof Sketch of Proposition E.4. We discuss a sketch of the proof for the version of the algorithm of (Kleinberg & Mullainathan, 2024) that uses a subset oracle for $\mathcal{L}$, *i.e.*, for any $L_i, L_j \in \mathcal{L}$ it can ask "Is $L_i \subseteq L_j$?". Let us first give a high-level description of their algorithm. For large enough $n \in \mathbb{N}$, it creates a (potentially infinite) sequence of languages $\mathcal{L}' = \{L_{i_1}, L_{i_2}, \dots\} \subseteq \mathcal{L}$ such that the following hold.

- **(i)** For every language $L \in \mathcal{L}'$ it holds that L is consistent, *i.e.*, $S_n \subseteq L$, where S_n is the set of elements enumerated until round n,
- **(ii)** The sequence of languages in $\mathcal{L}'$ satisfies the inclusion: $L_{i_1} \supseteq L_{i_2} \supseteq \dots$, and
- **(iii)** $K \in \mathcal{L}'$.

Then, it outputs an arbitrary string x such that $x \notin S_n$ and $x \in L_{i_\ell}$, where $i_\ell \in \mathbb{N}$ is the largest number such that $L_{i_\ell} \in \mathcal{L}'$ and $i_\ell \leq n$. The immediate modification is to output a distribution $\mathcal{G}_n$ such that $\operatorname{supp}(\mathcal{G}_n) = L_{i_\ell} \setminus S_n$. Notice that this can be done in a computable way: in order to sample from this distribution, we first sample a natural number $\widehat{n}$ (*e.g.*, from a geometric distribution on $\mathbb{N}$), and then we check if $x_{\widehat{n}} \in L_{i_\ell} \setminus S_n$. □

An analogous modification can be made to the algorithm of (Kleinberg & Mullainathan, 2024) that only has access to a membership oracle for $\mathcal{L}$. For brevity, we omit the modifications to this algorithm.

Remark E.5 (Oracle Access for Results in Figure 3). Following the phrasing of (Kleinberg & Mullainathan, 2024), we provide both *functions* and *algorithms* that generate in the limit. An algorithm only accesses $\mathcal{L}$ via a membership oracle (and potentially a tell-tale oracle). When a generator uses other types of oracles (*e.g.*, subset oracle), we call it a *function*.

E.2. Infinite Coverage with Stable Generators

In this section, we continue the study of infinite coverage, exploring when it can be achieved with stable generators.

E.2.1. A Collection for which No Stable Generator Has Infinite Coverage

In this section, we show that there is a language collection $\mathcal{L}$ for which there exists an algorithm that achieves approximate breadth in the limit, but no stable algorithm can achieve the (strictly) weaker notion of generating with infinite coverage in the limit. The collection $\mathcal{L}$ is due to (Charikar & Pabbaraju, 2024a), who observed that a trivial generating algorithm that does not get *any* input generates from $\mathcal{L}$ exhaustively in the limit. Since exhaustive generation implies, by definition, generation with approximate breadth, we only need to prove the impossibility result for generation with infinite coverage by stable generators.

We first provide the collection and then state the result.

Example E.6 ((Charikar & Pabbaraju, 2024a)). Let $\mathcal{X} = \mathbb{N}, L_\infty = \mathbb{N}$, for every $i \in \mathbb{N}$ let $L_i = \mathbb{N} \setminus \{i\}$, and let $\mathcal{L} = \{L_\infty, L_1, L_2, \ldots\}$. Notice that every pair of languages $L_i, L_j \in \mathcal{L}$ differ in at most two elements, so it follows that $\mathcal{L}$ satisfies Definition 3.7. To see that it does not satisfy Angluin's condition (Definition 2.2), consider the language L_∞. Then, for every finite subset $T \subseteq L_\infty$ there is some language L_T such that $T \subseteq L_T$ and $L_T \subsetneq L_\infty$.

We continue with the statement of the theorem.

Theorem E.7. *There exists a countable collection of languages $\mathcal{L}$ that satisfies the weak Angluin's condition (Definition 3.7), and for which no stable generating algorithm can achieve generation with infinite coverage in the limit (Definition E.3).*

Proof. Consider the collection defined in Example E.6. Since it satisfies the weak Angluin's condition (Definition 3.7), by Theorem 3.8, it follows that there exists an algorithm that achieves generation with approximate breadth in the limit.[12] Assume towards contradiction that there exists a stable generating algorithm $\mathcal{G} = (\mathcal{G}_n)_{n\in\mathbb{N}}$ that achieves generation with infinite coverage in the limit. We will pick a target language and an enumeration of it that witnesses the lower bound based on the given algorithm $\mathcal{G}$. We denote the target language by K and the target enumeration by E_K^∞. Like in the previous proofs, for any enumeration E, we use the notation $E(i)$ to denote its i-th element, $E(1:i)$ to denote its first i elements, and $E(i:\infty)$ to denote all but the first $i-1$ elements.

As in the previous proofs of the impossibility results, we consider several phases for our construction. First, we start with the enumeration $E_\mathbb{N}^\infty = (1, 2, 3, \ldots)$. Notice that this is a valid enumeration for L_∞. We consider two cases: **(I)** either there is some $n \in \mathbb{N}$ such that $|\mathrm{supp}(\mathcal{G}_n)| = \infty$, or **(II)** if there is no such n the lower bound follows immediately by picking $K = \mathbb{N}$ and the hard enumeration $E_K^\infty = E_\mathbb{N}^\infty$. For the continuation of the proof, assume that the former case holds and let n_1 denote the first timestep for which this holds. Notice that up to that point we have enumerated $(1, \ldots, n_1)$. Let $\widehat{n}_1 \in \mathbb{N}$ be the smallest number strictly greater than n_1 that is in the support of $\mathcal{G}_{n_1}$. Notice that such a number must exist because $|\mathrm{supp}(G_{n_1})| = \infty$.

We now extend the target enumeration $E_K^\infty(1:\widehat{n}_1 - 1) = (1, 2, \ldots, \widehat{n}_1 - 1)$. Notice that this is well-defined since we only add elements to the already constructed enumeration. We continue building the target enumeration by skipping the element $\widehat{n}_1$ and including the element $\widehat{n}_1 + 1$ to it, *i.e.*, the $\widehat{n}_1$-th element of the constructed enumeration is $\widehat{n}_1 + 1$. We continue adding consecutive elements to the enumeration E_K^∞ until the first timestep $n > \widehat{n}_1 + 1$ such that $\mathrm{supp}(\mathcal{G}_n) \neq \mathrm{supp}(\mathcal{G}_{n_1})$ and $|\mathrm{supp}(\mathcal{G}_n)| = \infty$. Notice that if no such n exists the lower bound already follows by picking the target language $K = L_{\widehat{n}_1}$ and the constructed target enumeration. This is because in every timestep either $\mathrm{supp}(\mathcal{G}_n) = \mathrm{supp}(\mathcal{G}_{n_1})$ (and therefore $\mathrm{supp}(\mathcal{G}_n) \not\subseteq K$ because $\widehat{n}_1 \in \mathrm{supp}(\mathcal{G}_n)$) or $|\mathrm{supp}(\mathcal{G}_n)| < \infty$, hence the algorithm does not achieve generation with infinite coverage in the limit. For the continuation of the proof, let n_2 denote the first timestep for which $\mathrm{supp}(\mathcal{G}_{n_2}) \neq \mathrm{supp}(\mathcal{G}_{n_1})$ and $|\mathrm{supp}(\mathcal{G}_{n_2})| = \infty$. We then add the element $\widehat{n}_1$ to the constructed prefix of the enumeration E_K^∞ and terminate the first phase.

Notice that at the end of the first phase we have enumerated all the elements $\{1, 2, \ldots, n_2 - 1\}$ and the support of the generating algorithm has changed at least once or we have the desired lower bound. We continue inductively in exactly the same way until **(I)** either some phase cannot be terminated in which case the lower bound follows because the property of infinite coverage in the limit is not achieved or **(II)** we construct infinitely many phases which witness infinitely many changes in the support of the generating algorithm, hence showing it cannot be stable. This concludes the proof. □

E.2.2. Sufficient Condition for Stable Generation with Infinite Coverage

In this section, we provide a sufficient condition on the language collection $\mathcal{L}$ that guarantees the existence of a stable generating algorithm that generates with infinite coverage in the limit. In particular, we can show that if a collection has finite closure dimension (Li et al., 2024), then there exists a stable generating algorithm that achieves infinite coverage in the limit. First, we give the definition of the closure dimension (Li et al., 2024), which is inspired by a result of (Kleinberg & Mullainathan, 2024) on *uniform generation*[13] from finite sets of languages.

Definition E.8 (Closure Dimension (Li et al., 2024))**.** *The closure dimension of $\mathcal{L}$, denoted by $\mathrm{d}(\mathcal{L})$, is the largest natural*

[12] As we explained, this also follows from the work of (Charikar & Pabbaraju, 2024a).

[13] The exact definition of uniform generation is not important for our work. At a high level, this condition asks whether there exists some $d \in \mathbb{N}$ such that after the generator observes d different strings from *any* target language of $\mathcal{L}$, then it can generate unseen strings that belong to K.

number $\ell \in \mathbb{N}$ for which there exist distinct $x_1, \ldots, x_\ell \in \mathcal{X}$ such that

$$V(x_1, \ldots, x_\ell) \coloneqq \{L \in \mathcal{L} \colon \{x_1, \ldots, x_\ell\} \subseteq L\} \neq \emptyset \quad \textit{and} \quad \left| \bigcap_{L \in V(x_1, \ldots, x_\ell)} L \right| < \infty \,.$$

If for every $\ell \in \mathbb{N}$ there exists a set of distinct elements that satisfies this condition we say that $\mathrm{d}(\mathcal{L}) = \infty$.

In general the closure dimension can be ∞, but due to a result of (Kleinberg & Mullainathan, 2024), we know that all collections of languages with finitely many languages have finite closure dimension. In order to design an algorithm that achieves stable infinite coverage for any collection $\mathcal{L}$ that has a finite closure dimension, we will make use of a stronger oracle for $\mathcal{L}$ than just the membership oracle to it. Namely, we define the *version space intersection* (VSI) membership oracle as follows.

Definition E.9 (Membership Oracle to Version Space Intersection (VSI))**.** *The membership oracle to VSI is a primitive that, given a set of distinct elements $x_1, \ldots, x_n \in \mathcal{X}$ and a target element $x \in \mathcal{X}$, returns*

$$\mathbb{1}\left\{x \in \cap_{L \in V(x_1, \ldots, x_n)} L\right\} \,.$$

We remark that for finite collections $\mathcal{L}$ this oracle can be computed just with membership oracle to $\mathcal{L}$, but for countable collections this oracle might not be computable.

Proposition E.10 (Adaptation of Lemma 3.2 in (Li et al., 2024))**.** *Let $\mathcal{L}$ be a collection of languages with $\mathrm{d}(\mathcal{L}) < \infty$ (Definition E.8). There exists a stable (Definition 3.13) generating algorithm $\mathcal{G} = (\mathcal{G}_n)$ for $\mathcal{L}$ that, given the value of $\mathrm{d}(\mathcal{L})$, achieves infinite coverage (Definition E.3) using access to a VSI membership oracle for $\mathcal{L}$, after taking as input $\mathrm{d}(\mathcal{L}) + 1$ distinct elements.*

In particular, since the closure dimension of any finite collection of languages is finite (Kleinberg & Mullainathan, 2024), for any finite collection of languages, there exists a stable generating algorithm that achieves infinite coverage. It is not hard to see that for such collections, the VSI oracle can be implemented using only membership oracle to languages in $\mathcal{L}$.

Corollary E.11 (Stable Generation for Finite Collections)**.** *For every finite collection of languages $\mathcal{L}$, the following hold:*

1. *There exists a stable generating algorithm that achieves generation with exact breadth in the limit, using only membership oracle access to $\mathcal{L}$.*
2. *There exists a stable generating algorithm that achieves generation with infinite coverage after taking as input $\mathrm{d}(\mathcal{L}) + 1$ distinct strings, using only membership oracle access to $\mathcal{L}$.*

Moreover, for finite collections, a stronger property is possible: the results of (Kalavasis et al., 2025) (see Proposition 3.9 in their work) show that for finite collections there exists a stable generating algorithm that achieves exact breadth in the limit (and, hence, also infinite coverage), but there might not be an upper bound on the elements needed to achieve this property.[14]

Finally, we prove Proposition E.10.

Proof of Proposition E.10. Our proof is inspired by the Lemma 3.2 from (Li et al., 2024). The only modification is that now the algorithm stops using new elements beyond the $\mathrm{d}(\mathcal{L}) + 1$ elements required to achieve infinite coverage. Moreover, we discuss the type of access to $\mathcal{L}$ needed that is sufficient to achieve this property, which was not the focus of (Li et al., 2024). Let $K \in \mathcal{L}$ be any target language and $x_1, \ldots, x_{\mathrm{d}(\mathcal{L})+1} \in K$ be any $\mathrm{d}(\mathcal{L}) + 1$ distinct elements of the target language. First, notice that since $x_1, \ldots, x_{\mathrm{d}(\mathcal{L})+1} \in K$, $V(x_1, \ldots, x_{\mathrm{d}(\mathcal{L})+1}) \neq \emptyset$, as $K \in V(x_1, \ldots, x_{\mathrm{d}(\mathcal{L})+1})$. By the definition of the closure dimension (Definition E.8) and since $|K| = \infty$ (recall that language generation is not meaningful with finite languages and, hence, throughout this work, we consider all languages are infinite),

$$\left| \bigcap_{L \in V(x_1, \ldots, x_{\mathrm{d}(\mathcal{L})+1})} L \right| = \infty \qquad \text{and} \qquad \bigcap_{L \in V(x_1, \ldots, x_{\mathrm{d}(\mathcal{L})+1})} L \subseteq K \,.$$

[14] To be precise, Proposition 3.9 in (Kalavasis et al., 2025) gives an algorithm to identify finite collections in the limit. This algorithm immediately gives an algorithm for generation with exact breadth: once we know an index z such that $K = L_z$, we can sample a natural number (from, *e.g.*, an exponential distribution on $\mathbb{N}$) and output the i-th element of L_z. The latter, in turn, can be found using the membership oracle to L_z.

Thus, the generating algorithm can stabilize its support to be $T \coloneqq \bigcap_{L \in V(x_1,\ldots,x_{\mathrm{d}(\mathcal{L})+1})} L$ and never change it from this point on during the interaction with the adversary. Notice that given access to a VSI membership oracle for $\mathcal{L}$ the learner can indeed sample from a distribution supported on T as follows: first sample a natural number $\widehat{n}$ (*e.g.*, from a geometric distribution on $\mathbb{N}$) and then query the VSI membership oracle with the set of elements $x_1, \ldots, x_{\mathrm{d}(\mathcal{L})+1}$ and the target element $x_{\widehat{n}}$.[15] Repeat the process until the oracle returns Yes. Notice that this process terminates with probability 1, and the support of the induced distribution is exactly T. □

As a final note on our discussion on stability, it is worth pointing out that there are collections that do not satisfy the weak Angluin's condition, nevertheless there is a stable generating algorithm that achieves infinite coverage after observing one example from the target language. The example is due to (Charikar & Pabbaraju, 2024a).

Example E.12 (Stable Infinite Coverage $\not\Longrightarrow$ Weak Angluin's Condition). Define the domain $\mathcal{X}$ and the language collection $\mathcal{L}$ as follows

$$\mathcal{X} = \mathbb{Z} \qquad \text{and} \qquad \mathcal{L} = \{L_\infty \coloneqq \mathbb{Z}, L_a \coloneqq \{a+i, i \in \mathbb{N}\} : a \in \mathbb{Z}\}\ ,$$

where $\mathbb{Z}$ is the set of integer numbers. Notice that both $\mathcal{X}$ and $\mathcal{L}$ are countable, and each $L \in \mathcal{L}$ is also countable. Consider the language L_∞ and any finite $T \subseteq L_\infty$. Let i_T be the smallest element of the subset T. Then, $T \subseteq L_{i_T}, L_{i_T} \subsetneq L_\infty$, and $|L_\infty \setminus L_{i_T}| = \infty$. Hence, this collection does not satisfy the weak Angluin's condition. Consider the generating algorithm $\mathcal{G}$ which in every round n outputs a distribution with $\mathrm{supp}\,(\mathcal{G}_n) = \mathbb{N} \setminus S_1$, where S_1 is the input in round 1. It is not hard to see that for any target language K, this generating algorithm achieves infinite coverage, and is, by definition, stable.

E.3. Generation with Increasing Coverage: A Strengthening of Stability

In this section, we introduce new property of generation – increasing coverage, which is a strengthening of stable generation.

A key observation in (Kleinberg & Mullainathan, 2024) is that their generator's support can decrease when it sees new strings from the target K and, in fact, for many language collections the number of valid strings omitted from its support can grow without bound, which is an extreme form of *mode collapse*. In this light, one can view stability as a property that avoids such extreme mode collapse: any stable generator can only change its support finitely many times. A natural question is whether we can achieve something stronger than stability and, yet, more tractable than breadth. To capture this phenomenon, we introduce the following notion of *generation with strictly increasing coverage*.

Definition E.13 (Generation with Strictly Increasing Coverage)**.** *Let $\mathcal{L}$ be a countable collection of languages. A generating algorithm $\mathcal{G} = (\mathcal{G}_n)$ is said to have strictly increasing coverage for $\mathcal{L}$ in the limit if, for any $K \in \mathcal{L}$ and enumeration of K, there is an $n^* \geq 1$ such that for all $n \geq n^*$, after seeing n elements of the enumeration, the following hold*

- $\mathrm{supp}\,(\mathcal{G}_n) \subseteq \mathrm{supp}\,(\mathcal{G}_{n+1})$, *and*
- *either* $\mathrm{supp}\,(\mathcal{G}_n) = K$ *or there exists some* $n' > n$ *such that* $\mathrm{supp}\,(\mathcal{G}_n) \subsetneq \mathrm{supp}\,(\mathcal{G}_{n'})$.

Intuitively, if a generator satisfies this property of strictly increasing coverage, then, at a high level, one may gather that it learns something new about the target language each time it sees a new string from it.

To gain intuition about when increasing coverage is achievable, let us consider two extremes. On the one hand, it is not hard to see that achieving approximate breadth along with strictly increasing coverage is significantly harder than achieving approximate breadth along: This is because if a generator has approximate breadth, then after seeing sufficiently many strings from K, its support only misses a finite number of strings from K and, then, if it further has strictly increasing coverage, its support eventually becomes equal to K implying exact breadth which is only achievable for collections satisfying Angluin's condition (Theorem 3.3). On the other hand, if one is not required to have infinite coverage[16] (a requirement already weaker than any notion of breadth), then it is easy to achieve strictly increasing coverage: consider the generator $\mathcal{G}$ in Proposition E.4, which achieves infinite coverage for any collection $\mathcal{L}$, and post-process the algorithm to have a support of size at most t on round t. Since eventually $\mathcal{G}$'s support has infinitely many elements (as it achieves infinite coverage), it follows that the support of the above post-processed variant increases infinitely many times, implying that the post-processed variant achieves strictly increasing coverage.

[15] To be formal, we need to use a different enumeration of the strings of $\mathcal{X}$ and the strings that define the target version space. We overload the notation for simplicity.

[16] For the subsequent discussion, we use the equivalent version of the definition of infinite coverage (Definition E.3) which allows the support of the generator to contain strings from the set S_n, which is the set of all strings enumerated so far.

Thus, the most interesting question is whether there is a generator that achieves infinite coverage – a property between breadth and consistent generation – while also having strictly increasing coverage. Our next result shows that there are collections for which this is indeed possible. The collection we use to show this result does not satisfy the weak Angluin's condition, so one cannot achieve even the weakest notion of breadth (namely, approximate breadth or equivalently exhaustive generation) for this collection.

Proposition E.14. *There exists a countable collection of languages $\mathcal{L}$ that does not satisfy the weak Angluin's condition (Definition 3.7) and for which there exists a generating algorithm $\mathcal{G} = (\mathcal{G}_n)$ that can achieve infinite coverage (Definition E.3) and has strictly increasing coverage in the limit (Definition E.13).*

Proof. Consider the collection of arithmetic progressions used in Example E.12. As we discussed, this collection does not satisfy the weak Angluin's condition. Let S_n be the set of elements enumerated up to round n and let $\widehat{t}_n$ denote the smallest element of S_n. Then, it is immediate that the generating algorithm that outputs a distribution supported on $\{\widehat{t}_n, \widehat{t}_n + 1, \ldots\}$ achieves infinite coverage and has strictly increasing coverage in the limit. □

We remark that the generating strategy in the above result uses information about the structure of $\mathcal{L}$, and not just membership access to it.

F. Additional Remarks and Discussion

In this section, we present additional remarks and discussions.

F.1. Separation between weak Angluin and Angluin's condition.

In Remark F.1, we give a collection $\mathcal{L}$, taken from (Charikar & Pabbaraju, 2024a), which witnesses that the above modification of Angluin's condition is a *strict* weakening of Definition 2.2.

Remark F.1 (Separation Between Definition 2.2 and Definition 3.7 (Charikar & Pabbaraju, 2024a)). We highlight that there is a separation between the collections of languages that satisfy Definition 2.2 and Definition 3.7, which is taken from (Charikar & Pabbaraju, 2024a). Let $\mathcal{X} = \mathbb{N}$, $L_i = \mathbb{N} \setminus \{i\}$, and $\mathcal{L} = \{\mathbb{N}, L_1, L_2, \ldots\}$. Then, $\mathcal{L}$ does not satisfy Definition 2.2 but satisfies Definition 3.7. Thus, Definition 3.7 is a strictly weaker condition than Definition 2.2.

F.2. Overview of Kleinberg and Mullainathan's Algorithm

In this section, we give a high-level description of the algorithm of Kleinberg & Mullainathan (2024). Consider some fixed language collection $\mathcal{L} = \{L_1, L_2, \ldots\}$. Now consider any enumeration the adversary gives as input to the generator. In every round $n \in \mathbb{N}$, the generation algorithm of Kleinberg & Mullainathan (2024) creates a (potentially infinite) sequence of languages $\mathcal{L}' = \{L_{i_1}, L_{i_2}, \ldots\} \subseteq \mathcal{L}$ such that the following holds:

- **(i)** For every language $L \in \mathcal{L}'$ it holds that L is consistent, *i.e.*, $S_n \subseteq L$, where S_n is the set of elements enumerated until round n,
- **(ii)** For every language $L_{i_j} \in \mathcal{L}'$ it holds that $L_{i_j} \subseteq L_{i_{j'}}, \forall j' \leq j$.

Then, it outputs an arbitrary string x such that $x \notin S_n$ and $x \in L_{i_\ell}$, where $i_\ell \in \mathbb{N}$ is the largest number such that $L_{i_\ell} \in \mathcal{L}'$ and $i_\ell \leq n$. The main ingredient of the proof is that for all n sufficiently large the target language K will be part of $\mathcal{L}'$. Moreover, languages that come after it are subsets of K. Thus, it is safe to be generating elements from these languages.

F.3. Unambiguous Generation Satisfies Uniqueness

In this section, we show that unambiguous generation satisfies the uniqueness criterion. To see this, consider any distinct languages $L \neq L'$. Suppose a generator $\mathcal{G}$ unambiguously generates from L. This implies that

$$|\text{supp}(\mathcal{G}) \triangle L| < \min_{L'' \in \mathcal{L},\ L'' \neq L} |\text{supp}(\mathcal{G}) \triangle L''| \ .$$

However, setting $L'' = L'$ implies that $|\text{supp}(\mathcal{G}) \triangle L| < |\text{supp}(\mathcal{G}) \triangle L'|$ which shows that $\mathcal{G}$ does not unambiguously generate from L'. This proves the following result.

Observation F.2. Unambiguous generation (Definition A.1) satisfies the uniqueness criterion.

F.4. Exhaustive Generation Satisfies Finite Non-Uniqueness

In this section, we show that exhaustive generation satisfies the finite non-uniqueness criterion.

Recall that in the formulation of exhaustive generation, the generating algorithm is a sequence of mappings from sequences of the domain to *enumerations* of the domain. Let $\mathcal{G}(1:\infty)$ be the set containing all the items $\mathcal{G}$ enumerates.

To see the claim, consider any pair of languages L and L' that differ in infinitely many elements, *i.e.*, $|L \triangle L'| = \infty$. Now, if a generator $\mathcal{G}$ generates exhaustively generates both L and L', then, by definition

$$|L \setminus \mathcal{G}(1:\infty)|\,, \quad |L' \setminus \mathcal{G}(1:\infty)|\,, \quad |\mathcal{G}(1:\infty) \setminus L|\,, \quad |\mathcal{G}(1:\infty) \setminus L'| \;<\; \infty\,. \tag{9}$$

This contradicts the fact that $|L \triangle L'| = \infty$ since

$$\begin{aligned} |L \triangle L'| &= |L \setminus L'| + |L' \setminus L| \\ &\leq (|(1:\infty) \triangle L'| + |L \setminus \mathcal{G}(1:\infty)|) + (|\mathcal{G}(1:\infty) \triangle L| + |L' \setminus \mathcal{G}(1:\infty)|) \\ &\leq 3 \cdot (|L \setminus \mathcal{G}(1:\infty)| + |\mathcal{G}(1:\infty) \setminus L| + |L' \setminus \mathcal{G}(1:\infty)| + |\mathcal{G}(1:\infty) \setminus L'|) \\ &\overset{(9)}{<} \infty\,. \end{aligned}$$

Observation F.3. Exhaustive generation (Definition A.2) satisfies the finite non-uniqueness criterion.

Remark F.4 (Exhaustive Generation Does Not Satisfy the Uniqueness Criterion). Note that the above proof can be made constructing – there is a generator which generates exhaustively from both L and L' provided L and L' differ in finitely many elements. This implies that exhaustive generation does not satisfy the uniqueness criteria.

G. Formal Definition of Language Identification in the Limit

In this section, we provide the formal definition of language identification in the limit.

For a fixed collection $\mathcal{L}$, an adversary and an identifier play the following game: The adversary chooses a language K from $\mathcal{L}$ without revealing it to the identifier, and it begins *enumerating* the strings of K (potentially with repetitions) $x_1, x_2, \ldots$ over a sequence of time steps $n = 1, 2, 3, \ldots$. The adversary can repeat strings in its enumeration, but the crucial point is that for every string $x \in K$, there must be at least one time step n at which it appears. At each time n, the identification algorithm I, given the previous examples $x_1, x_2, \ldots, x_n$, outputs an index i_n that corresponds to its guess for the index of the true language K. Language identification in the limit is then defined as follows.

Definition G.1 (Language Identification in the Limit (Gold, 1967))**.** *Fix some K from the language collection $\mathcal{L} = \{L_1, L_2, \ldots\}$. The identification algorithm $\mathit{I} = (I_n)$ identifies K in the limit if there is some $n^* \in \mathbb{N}$ such that for all steps $n > n^*$, the identifier's guess i_n satisfies $i_n = i_{n-1}$ and $L_{i_n} = K$. The language collection $\mathcal{L}$ is identifiable in the limit if there is an identifier that identifies in the limit any $K \in \mathcal{L}$, for any enumeration of K. In this case, we say that the identifier identifies the collection $\mathcal{L}$ in the limit.*

It is important to note that the above definition imposes some stability to the algorithm: since there can be multiple appearances of K in the enumeration of $\mathcal{L}$, an algorithm identifies K in the limit only if it eventually *stabilizes* (*i.e.*, $i_n = i_{n-1}$ for n larger than some n^*) to a correct index (*i.e.*, $L_{i_n} = K$). A natural question is which collections of languages are identifiable in the limit. Angluin (Angluin, 1980) provided a condition that characterizes such collections (see Definition 2.2).

Theorem G.2 (Characterization of Identification in the Limit (Angluin, 1980))**.** *The following holds for any countable collection of languages $\mathcal{L}$.*

1. *$\mathcal{L}$ is identifiable in the limit if it satisfies Angluin's condition and one has access to the tell-tale oracle.*
2. *If there is an algorithm that identifies $\mathcal{L}$ in the limit, then Angluin's condition is true and the tell-tale oracle can be implemented.*

The above tight characterization shows that language identification is information-theoretically impossible even for simple collections of languages, such as the collection of all regular languages. Crucially, access to the tell-tale oracle is necessary for identification in the limit (its existence alone is not sufficient) (Angluin, 1980).

G.1. Representation of Generators

Remark G.3 (Representation of the Generators). The astute reader might observe that the previous definitions allow for generating algorithms that output infinite-sized objects. However, all our generating algorithms have succinct representations and this allows for computable algorithms that sample (*i.e.*, generate) a new element, enumerate the support of all generatable elements, and, given an element, decide whether it belongs to the support (*i.e.*, whether it is part of the enumeration). On the other hand, our lower bounds are stronger, they hold for functions that might not be computable.

G.2. Membership Oracle Problem

In this section, we define the Membership Oracle Problem (MOP), which is required for the impossibility results of (Kalavasis et al., 2025), but not required for the characterizations in our work. For more details, we refer to Definitions 5 and 6 in (Kalavasis et al., 2025).

Definition G.4 (Membership Oracle Problem (Kalavasis et al., 2025))**.** *Given a generator $\mathcal{G}$, the membership oracle problem for $\mathcal{G}$, denoted as $\mathsf{MOP}(\mathcal{G})$, is defined as follows: given the description of $\mathcal{G}$ and a string x, output* Yes *if $x \in \mathrm{supp}(\mathcal{G})$ and output* No *otherwise.*

H. Detailed Related Work

Since the work of (Kleinberg & Mullainathan, 2024), a growing line of research has explored various aspects of language generation with and without breadth (*e.g.*, (Li et al., 2024; Kalavasis et al., 2025; Charikar & Pabbaraju, 2024a; Raman & Raman, 2025)). We already overview the work studying generation with breadth in the main body (Section 1.2). Here, we discuss the other lines of work and present a map between the results of (Charikar & Pabbaraju, 2024b) and some of our results.

Other Directions in Language Generation. Beyond breadth, recent work has explored other aspects of language generation. Li, Raman, and Tewari (2024) studied language generation with uncountable collections and analyzed sample complexity for generation. Raman & Raman (2025) investigated language generation in a model where an adversary can introduce errors in the inputs, developing a robust framework for noisy settings. (Karbasi et al., 2025) explored the complexity of determining if a specific generator $\mathcal{G}$ is hallucinating.

Comparison to (Charikar & Pabbaraju, 2024b). See Section 1.2 for a timeline of the works Charikar & Pabbaraju (2024a), Charikar & Pabbaraju (2024b), and the present work. In the following, we map the relevant results of Charikar & Pabbaraju (2024b) to some of our results.

- **Characterization of Generation with Exact Breadth:** Their result showing that Weak Angluin's Condition with Existence (Proposition 6.1 in their work) is necessary for exhaustive generation is comparable to the lower bound for exhaustive generation in Theorem A.3. Their result showing the sufficiency of Weak Angluin's Condition with Existence (Proposition 6.2 in their work) for exhaustive generation is comparable to the upper bound for exhaustive generation in Lemma C.5. Their result showing the sufficiency of Weak Angluin's Condition with Enumeration (Proposition 6.2 in their work) for exhaustive generation with only membership queries is comparable to Lemma C.7.
- **Characterization of Exhaustive Generation:** Their result showing that Angluin's Condition is necessary for generation with exact breadth (Proposition 5.3 in their work) is comparable to the upper bound in Theorem 3.3.

Finally, as mentioned in Section 1.2, our work provides several additional contributions for existing notions of breadth/stability beyond these shared results (see Sections 3.1 to 3.2 and Remarks 3.6, 3.11 and 3.12). Further, our work also introduces new notions of breadth/stability and provides results for them (see Appendices A.4, A.5 and E).