# Language Generation: Complexity Barriers and Implications for Learning

Marcelo Arenas
DCC UC & IMFD
marenas@uc.cl

Pablo Barceló
IMC UC, CENIA & IMFD
pbarcelo@uc.cl

Luis Cofré
Faculty of Mathematics UC
luis.cofr@uc.cl

Alexander Kozachinskiy
CENIA
alexander.kozachinskyi@cenia.cl

November 10, 2025

**Abstract**

Kleinberg and Mullainathan showed that, in principle, language generation is always possible: with sufficiently many positive examples, a learner can eventually produce sentences indistinguishable from those of a target language. However, the existence of such a guarantee does not speak to its practical feasibility. In this work, we show that even for simple and well-studied language families—such as regular and context-free languages—the number of examples required for successful generation can be extraordinarily large, and in some cases not bounded by any computable function. These results reveal a substantial gap between theoretical possibility and efficient learnability. They suggest that explaining the empirical success of modern language models requires a refined perspective—one that takes into account structural properties of natural language that make effective generation possible in practice.

## 1 Introduction

The study of language identification problems in the limit started in the 60s with the seminal work by Gold [2]. In a nutshell, a (finite or infinite) collection $\mathcal{L}$ of languages is said to be *identifiable in the limit* if there exists a learning algorithm that, given an infinite sequence of positive examples from some target language $L \in \mathcal{L}$, produces a sequence of hypotheses $H_1, H_2, \ldots$ from $\mathcal{L}$ such that, after finitely many steps, all subsequent hypotheses coincide exactly with $L$. Gold showed that even the class of regular languages is not identifiable in the limit, whereas finite families of languages are. Later, Angluin refined and deepened the theory by characterizing, in purely structural terms, the collections of languages that are identifiable in the limit [1]. Overall, the concept turned out to be extremely restrictive—very few nontrivial collections of languages satisfy these conditions—and has often been interpreted as formal support for the "poverty of the stimulus" argument in linguistics, suggesting that purely data-driven learning is insufficient to explain language acquisition without some form of innate prior structure or constraints.

The recent success of LLMs contrasts sharply with the aforementioned negative results on learnability. Even though LLMs are trained on finite samples and mostly see only positive examples, they still manage to capture many of the structural and semantic patterns of natural language. This suggests that useful generalization may be possible without explicitly identifying the underlying grammar. In this sense, the phenomenon revealed by LLMs naturally motivates the problem of *language generation in the limit*, recently studied by Kleinberg and Mullainathan [9], which seeks to formalize and understand how a computational process can produce data indistinguishable from that of a target language—shifting the focus from *identifying* the underlying rules to *generating* behavior consistent with them. Their results show that such generation is in

fact always possible, provided the collection $\mathcal{L}$ is countable, highlighting a fundamental asymmetry between learning and generation: while exact identification may be unattainable, successful imitation can always be achieved under mild assumptions. As Kleinberg and Mullainathan state, this shows that, in a certain sense, language identification and language generation are fundamentally different problems. Noticeably, the concept of language generation in the limit has been further studied and refined in several subsequent works [12, 13, 6, 7, 8, 10, 14, 3].

But while the results on language generation in the limit show that the problem is "computable", this does not necessarily mean it is "feasible". From a computational complexity perspective, feasibility depends on how many examples a learner must process before it can reliably generate strings indistinguishable from those of the target language. If this sample complexity or computational cost grows astronomically with the size or structure of the language, then the theoretical possibility of generation becomes practically meaningless. Understanding language generation through this lens—by quantifying the resources required to achieve it—is therefore essential to determine whether the phenomenon observed in systems like LLMs reflects genuine computational feasibility or merely a theoretical existence result.

One might expect that language generation becomes infeasible if considered over arbitrary computable languages, where complexity pathologies are the norm. What is less obvious—and what we show—is that the difficulty remains even when we restrict attention to the most well-behaved and extensively studied families. In particular, we analyze language generation problem for *regular* and *context-free* languages, which form the core of formal language theory and underlie fundamental applications in computer science. Despite their relatively tame structure, these classes already exhibit inherent barriers to feasible generation. This provides a more nuanced view: the challenge is not merely a consequence of working with overly expressive language families, but is intrinsic to language generation itself.

In particular, we show that generating from finite families of (deterministic) finite automata requires a double-exponential number of examples on the size of the family. This indicates that, although generation is theoretically possible, it quickly becomes intractable even for simple language classes. In contrast, for context-free languages, we prove a strong negative result: there exists no computable bound on the number of examples needed to guarantee successful generation, even when the family contains only two infinite context-free languages defined by (deterministic) pushdown automata. Hence, increasing expressive power—from regular to context-free languages—comes at an enormous computational cost. While these results do not directly imply that the notion of language generation in the limit is inadequate for explaining the success of large language models, they do suggest that if such a perspective is to be adopted, it must be done in a much more refined and careful way—one that also accounts for why the kinds of languages LLMs seem to learn in practice require far fewer examples.

Beyond these results, our work also sheds light on interesting open problems in formal language theory. In particular, it is easy to observe that the lower bound on the number of examples needed for language generation in the limit is determined by the largest size of a finite intersection of languages in the collection $\mathcal{L}$ (see Proposition 1). The study of this quantity has received little attention in the literature, yet it is closely related to fundamental questions about the complexity of language intersection. Notably, deciding if the intersection of a collection of regular languages is non-empty is PSPACE-complete [11], while the same problem becomes undecidable for context-free languages (folklore). Hence, understanding the structure and size of finite intersections provides a new perspective that bridges language learning and complexity results.

# 2 Language generation in the limit

## 2.1 Non-uniform generation

Let $\Sigma$ be a finite alphabet. A *generator* is a function $G$ that takes a finite sequence of words over $\Sigma$ and returns an infinite set of words over $\Sigma$.

**Definition 1** (Language generation in the limit.)**.** *A set $\mathcal{F}$ of infinite languages over $\Sigma$ to be* generatable in the limit*, if there exists a generator $G$ such that for any $L \in \mathcal{F}$, and for any ordering $w_1, w_2, w_3, \ldots$ of the words of $L$, there exists an integer $m \geq 1$ such that $G(w_1, \ldots, w_n)$ is an infinite subset of $L$ for all $n \geq m$.*

Kleinberg and Mullainathan show that all countable $\mathcal{F}$ are generatable in the limit [9], and Li, Raman, and Tewari [12] provide a necessary and sufficient conditions for an arbitrary $\mathcal{F}$ to be generatable in the limit.

### 2.2 Uniform generation

Notice that in the previous definition, $m$, the size of the training set needed to generate from an unknown $L \in \mathcal{F}$ depends both on $L$ and on its ordering. A natural question from the practical perspective is which families $\mathcal{F}$ admit a *uniform* bound on $m$. Formally:

**Definition 2** (Uniform generation)**.** *Let $m \geq 1$ be a fixed integer. A set $\mathcal{F}$ of infinite languages is* $m$-generatable*, if there exists a generator $G$ such that for every $L \in \mathcal{F}$ and for every $m$ distinct words $w_1, \ldots, w_m \in L$, we have that $G(w_1, \ldots, w_n)$ is an infinite subset of $L$ for every $n \geq m$.*

Li, Raman, and Tewari charecterize families $\mathcal{F}$ that are $m$-generatable for some $m \in \mathbb{N}$. We first provide a simpler and more precise version of this result.

**Proposition 1.** *A set $\mathcal{F}$ of infinite languages is not $m$-generatable if and only if there exists a non-empty $\mathcal{S} \subseteq \mathcal{F}$ such that $\bigcap_{L \in \mathcal{S}} L$ is finite but has size at least $m$.*

*Proof.* Assume first that there exists a non-empty $\mathcal{S} \subseteq \mathcal{F}$ such that $\bigcap_{L \in \mathcal{S}} L$ is finite but has size at least $m$. Assume for contradiction that $\mathcal{F}$ admits an $m$-generator $G$. Take $m$ distinct words $w_1, \ldots, w_m \in \bigcap_{L \in \mathcal{S}} L$. The set $G(w_1, \ldots, w_m)$ is infinite and hence contains some word $w \notin \bigcap_{L \in \mathcal{S}} L$. That is, we have $w \notin L$ and $w_1, \ldots, w_m \in L$ for some $L \in \mathcal{S}$. Hence, $G(w_1, \ldots, w_m)$ is not a subset of $L$, a contradiction.

Assume, in turn, that there exists no non-empty $\mathcal{S} \subseteq \mathcal{F}$ such that $\bigcap_{L \in \mathcal{S}} L$ is finite but has size at least $m$. Given $m$ distinct words $w_1, \ldots, w_m$, we define $G(w_1, \ldots, w_m)$ as the intersection of all $L \in \mathcal{F}$ such that $w_1, \ldots, w_m \in L$. Observe that by definition, for every $L \in \mathcal{F}$ and every distinct $w_1, \ldots, w_m \in L$, we have that $G(w_1, \ldots, w_m)$ is a subset of $L$. We have to show that this is an infinite subset. Define $\mathcal{S}$ to be the set of all languages from $\mathcal{F}$ that contain $w_1, \ldots, w_m$. By definition, $\mathcal{S}$ is non-empty and

$$G(w_1, \ldots, w_m) = \bigcap_{L \in \mathcal{S}} L$$

has size at least $m$, meaning that it has to be infinite. □

Observe then that, when $\mathcal{F}$ is finite, it is $m$-generatable for some $m \in \mathbb{N}$, because there are only finitely many possible sets $\mathcal{S} \subseteq \mathcal{F}$ that may arise as finite intersections. However, to understand the *feasibility* of generation, we must go beyond the mere existence of such an $m$ and instead quantify its magnitude. Finite families thus provide a natural setting: they avoid the classical impossibility phenomena associated with infinite families, while still allowing us to meaningfully examine how large $m$ must be as a function of the descriptive complexity of the languages involved.

## 3 Results for regular languages

We begin by studying $m$-generatability for finite families of regular languages. The main question we address is the following: how many examples are required to generate from an unknown regular language $L$ recognizable by a finite family of finite automata? We show that, in the worst case, a double-exponential number of examples in the size of the family is necessary.

### 3.1 Finite automata

Recall that a *finite automaton* over an alphabet $\Sigma$ is a tuple $\mathcal{A} = (Q, q_0, F, \delta)$, where (a) $Q$ is a finite set of *states*, (b) $q_0 \in Q$ is the *initial state*, (c) $F \subseteq Q$ is the set of *final* states, and (d) $\Delta \subseteq Q \times \Sigma \times Q$ is a set of *transition rules*.

A *configuration* is a triple $(q, w)$ where $q \in Q$ is the current state and $w \in \Sigma^*$ is the unread input. The transition relation $\vdash$ is defined as follows:

$$(q, aw) \vdash (q', w) \quad \text{if } (q, a, q') \in \Delta.$$

The accepted language is:

$$L(\mathcal{A}) = \{w \in \Sigma^* \mid (q_0, w) \vdash^* (q, \epsilon) \text{ for some } q \in F\}.$$

Languages accepted by finite automata are known as *regular* languages.

A *deterministic* finite automaton (DFA) is a finite automaton in which, for every state $q \in Q$ and symbol $a \in \Sigma$, there is at most one transition of the form $(q, a, q') \in \Delta$. It is a well-known fact that every regular language can be recognized by a DFA. Thus, determinism does not restrict the class of regular languages. However, translating from a finite automaton into an equivalent DFA takes exponential time.

## 3.2 Theoretical results

We establish now our first main result regarding the sample complexity of uniform language generation in the limit for regular languages specified by finite families of finite automata. Our lower bounds holds even for the class of regular languages specified as DFAs.

**Theorem 1.** *The following statements hold:*

*a) Let $n, k \in \mathbb{N}$, and let $\mathcal{F}$ be a family of infinite regular languages over the binary alphabet such that $|\mathcal{F}| = k$ and each $L \in \mathcal{F}$ is accepted by a finite automaton with at most $n$ states. Then $\mathcal{F}$ is $2^{n^k}$-generatable.*

*b) There exists $c \in \mathbb{N}$ such that, for every $n \in \mathbb{N}$ and $k \leq 2^n$, there exists a family $\mathcal{F}$ of infinite regular languages over the binary alphabet satisfying the following: $|\mathcal{F}| = k$, each $L \in \mathcal{F}$ is accepted by an $(cn)$-state DFA, and $\mathcal{F}$ is not $2^{n \cdot 2^k}$-generatable.*

*Proof.* We first establish *a)*. Let $\mathcal{F}$ be a $k$-size family of infinite regular languages, each one of them specified by $n$-state finite automata. By the standard product construction, each intersection $\bigcap_{L \in \mathcal{S}} L$, for $\mathcal{S} \subseteq \mathcal{F}$, is recognizable by a finite automaton with at most $n^k$ states. If this intersection is finite, it cannot have a word of length $n^k$ or larger. Hence, this intersection has size less than $2^{n^k}$. By Proposition 1, the family is $2^{n^k}$-generatable.

We now proceed to the proof of *b)*. We split an input word into blocks of length $n$ (if the input length is not a multiple of $n$, the word will not be accepted by any of the languages from the family). Let $w_1 \leq w_2 \leq \ldots w_{2^n}$ be the lexicographic ordering of the binary words of length $n$. The language $L_1$ is the language of all words that have at most one block, equal to $w_1$. The language $L_i$ is the language of words where there are no two occurrences of the block $w_i$ without an occurrence of some block $w_1, \ldots, w_{i-1}$ between them. All these languages can be accepted by $O(n)$-DFAs. Indeed, for $L_i$, we check an automata that looks for the first difference of the current block with $w_i$. If the current block is smaller then $w_i$ in the lexicographic order, we reset. If it is equal to $w_i$, we have to be check not to get $w_i$ again before resetting.

We first show that the intersection of $L_1, \ldots, L_k$ is finite but has a word of length at least $n \cdot 2^k$. Indeed, consider any word in the intersection. Since it is in $L_1$, it has at most one block $w_1$. Before and after $w_1$, it can have at most one block $w_2$. Now, we have at most 4 spaces where to put $w_3$:

$$w_3 w_2 w_3 w_1 w_3 w_2 w_3$$

and so on, each block can appear only finitely many time.

On the other hand, one can see that by adding blocks $w_1, w_2, w_3$ in the above manner (when we add $w_i$, we add it between all the existing blocks), we get 1 block $w_1$, 2 blocks $w_2$, 4 blocks $w_3$, and so on. In the end, we get $2^k$ blocks $w_k$, and the conditions will be satisfied since $w_k$ is a word of length $n$.

So far, we have only obtained a long word of length $n \cdot 2^k$. To construct an intersection of size $2^{n \cdot 2^k}$, we apply the same construction to the odd $n$-bit blocks, while allowing the even $n$-bit blocks to be filled arbitrarily. In addition, we impose the restriction that each language $L_i$ contains an even number of $n$-bit blocks, so that the number of odd $n$-bit blocks is equal to the number of even $n$-bit blocks in the words belonging to the intersection. In this way, since there is one word in the intersection whose odd $n$-bit blocks contain $n \cdot 2^k$ bits, we obtain $2^{n \cdot 2^k}$ words in the intersection by independently assigning 0 or 1 to each bit in the even $n$-bit blocks. □

As a corollary of Theorem 1, we obtain a triple-exponential lower bound on the number of examples required for generation in the limit for any finite family of infinite regular languages over the binary alphabet, recognizable by DFAs of linear size.

**Corollary 1.** *There exists a constant $c \in \mathbb{N}$ such that, for every $n \in \mathbb{N}$, there is a finite family $\mathcal{F}$ of infinite regular languages over the binary alphabet such that each $L \in \mathcal{F}$ is accepted by a $(cn)$-state DFA, and $\mathcal{F}$ is not $2^{n \cdot 2^{2^n}}$-generatable.*

# 4 Results for context-free languages

We now turn to $m$-generatability for finite families of context-free languages. The corresponding question is: how many examples are required to generate from an unknown language $L$ recognizable by a *deterministic pushdown automaton* of a certain size? In stark contrast to the regular case, we show that there is no computable bound on the number of examples sufficient for uniform generation, even when the family contains only two infinite context-free languages.

## 4.1 Pushdown automata

A *pushdown automaton* (PDA) over an input alphabet $\Sigma$ is a tuple $\mathcal{P} = (Q, \Sigma, \Gamma, q_0, Z_0, \Delta, F)$, where: (a) $Q$ is a finite set of *states*, (b) $\Gamma$ is a finite *stack alphabet*, (c) $q_0 \in Q$ is the *initial* state, (d) $Z_0 \in \Gamma$ is the *initial stack symbol*, (e) $F \subseteq Q$ is the set of *final states*, and (f) $\Delta \subseteq Q \times (\Sigma \cup \{\epsilon\}) \times \Gamma \times Q \times \Gamma^*$ is a finite set of *transition rules*.

A *configuration* is a triple $(q, w, \alpha)$ where $q \in Q$ is the current state, $w \in \Sigma^*$ is the unread input, and $\alpha \in \Gamma^*$ is the stack contents (the leftmost symbol is the top of the stack). The transition relation $\vdash$ is defined as follows:

$$(q, aw, X\beta) \vdash (q', w, \gamma\beta) \quad \text{if } (q, a, X, q', \gamma) \in \Delta.$$

The accepted language is:

$$L(\mathcal{P}) = \{w \in \Sigma^* \mid (q_0, w, Z_0) \vdash^* (q, \epsilon, \alpha) \text{ for some } q \in F\}.$$

Languages accepted by PDAs are known as *context-free* languages.

## 4.2 Theoretical results

**Theorem 2.** *There exists no algorithm that, given two infinite context-free languages $L_1, L_2$ defined by PDAs, outputs some $m$ such that $\{L_1, L_2\}$ is $m$-generatable (we assume that the algorithm might not halt if either $L_1$ or $L_2$ is finite).*

*Proof.* We show that if such an algorithm exists, then the halting problem is solvable.

**Lemma 1.** *There is an algorithm that, given a Turing machine $M$ without an input, constructs two PDAs accepting languages $L_1$ and $L_2$ with the following property: if $M$ halts in precisely $t$ computation steps, then $2^t = |L_1 \cap L_2|$.*

Assuming that Lemma 1 is proved and that the algorithm from Theorem 2 exists, we show the decidability of the halting problem, which leads to a contradiction. More precisely, we show that the problem of verifying, given a Turing machine $M$, whether $M$ halts on input the empty string is decidable. Given $M$, we use the algorithm in Lemma 1 to construct two context-free languages $L_1$ and $L_2$ such that if $M$, on input the empty string, halts in precisely $t$ computation steps, then $2^t = |L_1 \cap L_2|$. It is a folklore result that, given a context-free language $L$, one can compute $|L|$ (which is either a natural number or $+\infty$). We use this to compute $|L_1|$ and $|L_2|$, and we consider three cases for our proof.

1. If $L_1$ is finite, we compute some $t$ such that $2^t > |L_1| \geq |L_1 \cap L_2|$. Note that $M$ cannot halt in $t$ or more steps. So we run it for $t-1$ steps, and if it has not halted, we output that $M$ does not halt.
2. If $L_2$ is finite, then we proceed as in the previous case but considering $t$ such that $2^t > |L_2| \geq |L_1 \cap L_2|$.
3. If $L_1$ and $L_2$ are both infinite, we use the algorithm from Theorem 2 to obtain some number $m$ such that $\{L_1, L_2\}$ is $m$-generatable. We compute some $t$ such that $m \leq 2^t$. By Proposition 1, we know that $|L_1 \cap L_2| < 2^s$ for every $s \geq t$. Hence, the machine $M$ in this case cannot halt in $t$ or more steps. Again, to decide if $M$ halts, it is then enough to run it for $t-1$ steps.

*Proof of Lemma 1.* Assume that $M$ halts in $t$ computation steps, with $C_1, \ldots, C_t$ being its sequence of configurations (each configuration is a word, coinciding with the current content of the tape; the letter, corresponding to the current position of the head, additionally indicates the current state of the machine). In the construction, we use two special delimiting symbols $\#_0, \#_1$ that are not included in the alphabet of $M$. We give two context-free grammars $L_1, L_2$ whose intersection consists exactly of words of the form

$$C_1 \#_{i_1} C_2^R \#_{i_2} \ldots C_t^R \#_{i_t}$$

if $t$ is even, and

$$C_1 \#_{i_1} C_2^R \#_{i_2} \ldots C_t \#_{i_t}$$

if $t$ is odd; here $i_1, \ldots, i_t \in \{0, 1\}$ and $w^R$ denotes the reverse of a word $w$.

More specifically, we give $L_1, L_2$ via pushdown automata; first, one of the PDAs checks that $C_1$ is the initial configuration of $M$ and that $C_t$ is the only configuration in the sequence with an accepting state (the stack is not even required for this part). It remains to check that $C_{i+1}$ is obtained from $C_i$ correctly by the rules of $M$; one PDS will do that for odd $i$, and the other for even $i$. For example, the PDS for $L_1$ will first push $C_1$ to the stack. Then, while reading $C_2^R$, it will simultaneously free the stack while popping $C_1$ and comparing it with $C_2$, checking that they are almost equal, and differ only around the letter with the head, according to the transition function of $M$. After reading, $C_2^R$, the stack is free, and the first PDS repeats the same check for $C_3, C_4$, then for $C_5, C_6$ and so on, while the second PDS does the same for $C_2, C_3$, then $C_4, C_5$, and so on. As the symbols $\#_{i_j}$ can take any of the values 0 or 1, we conclude that if $M$ halts in precisely $t$ computation steps, then $|L_1 \cap L_2| = 2^t$. □

The proof of Lemma 1 concludes the proof of Theorem 2. □

# 5 Final Remarks

Kleinberg and Mullainathan highlight that language generation is, in principle, possible even under worst-case presentations of positive data, without relying on statistical regularities [9]. Our results refine this picture by showing that the feasibility of such generation can hinge critically on the structure of the language family. In particular, the lower bounds we obtain arise from the size of finite intersections among the languages in the collection: when such intersections are large, the number of examples required for successful generation can be enormous, or even beyond any computable bound. This phenomenon already appears in regular and context-free languages, indicating that the challenge is not confined to highly expressive or pathological classes. At the same time, it suggests a possible explanation for the practical tractability of generation in natural settings: if the relevant intersections are much smaller in the kinds of language structures and

distributions encountered by large language models, then generation may become feasible in practice despite the worst-case barriers. Thus, the key question is not only whether generation is possible, but under which structural conditions it can be achieved efficiently.

This perspective complements recent work showing that LLMs must, under very general assumptions, occasionally produce outputs that fall outside the target language they aim to model [5, 4]. These "hallucination" results demonstrate that even an idealized, perfectly calibrated model cannot avoid some deviations when only finite positive data are available. Taken together with our findings, this suggests a broader research direction: rather than treating generation, accuracy, and sample efficiency as independent concerns, a full account of the behavior of modern language models must understand how structural properties of language and data distributions jointly constrain (and enable) successful generalization. Developing such a unified framework remains an important open challenge.

## Acknowledgements

Barcelo and Kozachinskiy are funded by the National Center for Artificial Intelligence CENIA FB210017, Basal ANID. Barcelo is also funded by ANID Millennium Science Initiative Program Code ICN17002. Kozachinskiy is further supported by ANID Fondecyt Iniciacion grant 11250060. Cofré is supported by the Faculty of Mathematics at Pontificia Universidad Católica de Chile through the master's scholarship BMA.

## References

[1] Angluin, D. Inductive inference of formal languages from positive data. *Inf. Control. 45*, 2 (1980), 117–135.

[2] Gold, E. M. Language identification in the limit. *Inf. Control. 10*, 5 (1967), 447–474.

[3] Hanneke, S., Karbasi, A., Mehrotra, A., and Velegkas, G. On union-closedness of language generation. *CoRR abs/2506.18642* (2025).

[4] Kalai, A. T., Nachum, O., Vempala, S. S., and Zhang, E. Why language models hallucinate. *CoRR abs/2509.04664* (2025).

[5] Kalai, A. T., and Vempala, S. S. Calibrated language models must hallucinate. In *STOC* (2024), ACM, pp. 160–171.

[6] Kalavasis, A., Mehrotra, A., and Velegkas, G. Characterizations of language generation with breadth. *CoRR abs/2412.18530* (2024).

[7] Kalavasis, A., Mehrotra, A., and Velegkas, G. On the limits of language generation: Trade-offs between hallucination and mode-collapse. In *STOC* (2025), ACM, pp. 1732–1743.

[8] Karbasi, A., Montasser, O., Sous, J., and Velegkas, G. (im)possibility of automated hallucination detection in large language models. *CoRR abs/2504.17004* (2025).

[9] Kleinberg, J., and Mullainathan, S. Language generation in the limit. *Advances in Neural Information Processing Systems 37* (2024), 66058–66079.

[10] Kleinberg, J. M., and Wei, F. Density measures for language generation. *CoRR abs/2504.14370* (2025).

[11] Kozen, D. Lower bounds for natural proof systems. In *FOCS* (1977), pp. 254–266.

[12] Li, J., Raman, V., and Tewari, A. Generation through the lens of learning theory. *arXiv preprint arXiv:2410.13714* (2024).

[13] Peale, C., Raman, V., and Reingold, O. Representative language generation. *CoRR abs/2505.21819* (2025).

[14] Raman, A., and Raman, V. Generation from noisy examples. *CoRR abs/2501.04179* (2025).