

A Characterization of List Language Identification in the Limit

Moses Charikar*

Chirag Pabbaraju[†]

Ambuj Tewari[‡]

November 7, 2025

Abstract

We study the problem of language identification in the limit, where given a sequence of examples from a target language, the goal of the learner is to output a sequence of guesses for the target language such that all the guesses beyond some finite time are correct. Classical results of Gold showed that language identification in the limit is impossible for essentially any interesting collection of languages. Later, Angluin gave a precise characterization of language collections for which language identification is possible. Motivated by recent positive results for the related problem of language generation, we revisit the classic language identification problem in the setting where the learner is given the additional power of producing a list of k guesses at each time step. The goal is to ensure that beyond some finite time, one of the guesses is correct at each time step. Such list learning versions of several basic learning problems have been widely studied.

We give an exact characterization of collections of languages that can be k -list identified in the limit, based on a recursive version of Angluin’s characterization (for language identification with a list of size 1). This further leads to a conceptually appealing characterization: A language collection can be k -list identified in the limit if and only if the collection can be decomposed into k collections of languages, each of which can be identified in the limit (with a list of size 1). We also use our characterization to establish rates for list identification in the statistical setting where the input is drawn as an i.i.d. stream from a distribution supported on some language in the collection. Our results show that if a collection is k -list identifiable in the limit, then the collection can be k -list identified at an exponential rate, and this is best possible. On the other hand, if a collection is not k -list identifiable in the limit, then it cannot be k -list identified at any rate that goes to zero.

*Stanford University. Email: moses@cs.stanford.edu.

[†]Stanford University. Email: cpabbara@cs.stanford.edu.

[‡]University of Michigan, Ann Arbor. Email: tewaria@umich.edu.

1 Introduction

Modeling language learning is a complex and multi-faceted task. This paper is concerned with the model of *language identification in the limit* proposed by Gold in his seminal work [Gol67]. One of Gold’s motivations for considering such a model was to formalize the process of language acquisition in children. Viewed in a certain way, the language learning environment for children comprises of people around them constantly uttering valid sentences from a language, with few instances of negative examples. In this environment, a child supposedly infers a valid grammatical representation of the language, as the number of positive utterances increases. Moreover, linguistic studies indicate that children are rarely given feedback when they utter invalid sentences [BH04], or are not fully receptive of it when they are [Wik, McN70]. Thus, the learning environment would appear to comprise of a corpus of largely positive examples from the target language of interest, from which the child is supposed to infer a valid grammar for the language.

The formal setup that Gold proposes to model this is as follows: there is a countable collection $\mathcal{C} = (L_1, L_2, \dots)$ of languages, of which a certain language L_z is the target language of interest. The sentences, or “strings” in this language are enumerated to a learner in an online fashion, and in an arbitrary order. Concretely, the learner is provided a sequence x_1, x_2, \dots as input, and at every time step t , the learner guesses an index i_t based on the input it has seen so far. The learner is said to have identified the language L_z in the limit if beyond some finite time, all of its guesses i_t satisfy that $L_{i_t} = L_z$. If we think of the collection \mathcal{C} as comprising of different *grammars* for languages, then the criterion of identification in the limit is ensuring that the learner eventually acquires a valid grammar for the target language. We note that the input does not contain any negative examples that are outside the target language, and furthermore, the learner is never given any feedback about its guess being correct or not.

The results established by Gold for when language identification in the limit is possible in this model are drastically negative. Gold showed that essentially any interesting collection of languages that contains an infinite language cannot be identified in the limit. In particular, this rules out most formal language families of interest like regular languages, context-free languages, etc. Later follow-up work by Angluin [Ang79, Ang80] established a precise characterization of collections that may be identified in the limit. Namely, Angluin identified a strict combinatorial condition that must be satisfied by a collection for it to be identifiable in the limit, giving justification for why most formal language collections are not identifiable.

In contrast to these classical negative results, the recent work of Kleinberg and Mullainathan [KM24] establishes surprisingly general positive results for a slightly different task of language *generation* in the limit. In this model, instead of making a guess i_t of the identity of the target language at every time step, the learner is tasked with generating a string z_t . The learner successfully generates in the limit, if beyond some finite time, the string z_t generated by the learner belongs to the target language L_z , and is not one of the input strings x_1, \dots, x_t it has seen so far. [KM24] showed that this modified success criterion can be achieved for every countable collection of languages! Such a strong positive result for generation in the limit motivates revisiting Gold’s model for language identification in the limit, and asking: Could the strong negative results in the model be circumvented if we consider slightly more *relaxed notions* of identification in the limit?

Motivated by this question, in this work, we consider the setting of language identification in the limit, where instead of making a single guess at every time step, the learner may make a *short list* of at most k guesses. We will deem the learner to have successfully k -list identified the target language L_z in the limit, if beyond some finite time, every list of guesses output by the learner *contains* an index i such that $L_i = L_z$. This is a natural way to impose a slightly weaker desideratum on the learner: of its k guesses, we merely require that at least one is correct. In fact, this formulation

of “learning with a list” has received considerable attention in the literature for various natural learning problems (see [Section 1.2](#)).

1.1 Main Results

As noted above, Angluin [[Ang80](#)] provided a precise characterization of language collections that are identifiable in the limit. This characterization requires every language in the collection to have a finite *tell-tale* set which distinguishes it from every other language that is a proper subset of it. As our first main contribution, we present a tell-tale condition ([Section 4](#)) which we term the *k-Angluin condition*, that precisely characterizes *k*-list identification in the limit. The condition stipulates the existence of tell-tales for every language that satisfy a *k*-level recursive predicate. We show that this condition is both sufficient and necessary for *k*-list identification. Our characterization thus significantly generalizes Angluin’s tell-tale-style characterization for identification in the limit, which corresponds to the special case of $k = 1$, to the setting of *k*-list identification for any $k \geq 1$.

Theorem 1 (Characterization of *k*-list Identification). *A countable collection $\mathcal{C} = (L_1, L_2, \dots)$ of languages can be identified in the limit with a list of size k if and only if it satisfies the *k-Angluin condition*.*

We note that our characterization above also shows that for any k , there exist language collections that are *k*-list identifiable in the limit, but are not $(k - 1)$ -list identifiable. Furthermore, there also exist countable language collections that are not *k*-list identifiable for *any* finite value of k (see [Remarks 2](#) and [3](#)!)

Our next result is about a structural property of *k*-list identifiable collections. As guaranteed by [Theorem 1](#), any *k*-list identifiable collection must satisfy the *k-Angluin condition*. As it turns out, the recursive form of this condition allows us to decompose any *k*-list identifiable collection into a union of k collections that are each identifiable in the limit (in the standard $k = 1$ sense)!

Theorem 2 (Stratification of *k*-list Identifiable Collections). *Let $\mathcal{C} = (L_1, L_2, \dots)$ be a countable language collection. Then, \mathcal{C} is *k*-list identifiable in the limit if and only if $\mathcal{C} = \cup_{i=1}^k \mathcal{C}_i$ where \mathcal{C}_i is identifiable in the limit for every $i \in \{1, 2, \dots, k\}$.*

It is instructive to contrast the two results above with standard binary classification in machine learning. Namely, in the statistical learning setting of binary classification, the ability to output a list of k binary hypotheses from a hypothesis class, with the objective that one of them has low prediction error on a new test point, does not allow a learner to learn hypothesis classes that it cannot already learn with a *single* hypothesis (and these are precisely those classes that have finite *VC dimension* [[VC71](#), [EHKV89](#)]). Namely, if a learner outputs k hypotheses of which one has low error, then the learner may as well output the single hypothesis out of these k hypotheses that minimizes error on some additional training data. In contrast, in the context of identification in the limit where the language collection plays the role of a hypothesis class, the characterization in [Theorem 1](#) shows that the ability to output a list of k languages from the collection is *provably more powerful* than the ability to output only a single language.

Similarly, it is known that the union of k binary hypothesis classes, each of which is statistically learnable, results in a binary hypothesis class that is also statistically learnable¹ (e.g., see [[GGKM21](#)]). However, consider any collection \mathcal{C} that is *k*-list identifiable, but not identifiable in the limit. [Theorem 2](#) ensures that such a collection may always be broken down into k collections all of which are individually identifiable, but their union is *not* identifiable in the limit!

¹Online learnability, as governed by the *Littlestone dimension* [[Lit88](#), [BDPSS09](#)], also satisfies such a union closure property.

We now turn towards somewhat of a more statistical setting for language identification in the limit, where the sequence of strings input to the learner is an *infinite i.i.d. sequence* of strings drawn from a distribution supported on the target language, instead of being an arbitrary worst-case sequence. In this setting, one cares about the probability that a learner’s sequence of guesses converges to the identity of the target language. This setting was first considered by Angluin in [Ang88]. More recently, it was studied by [KMV25] in the context of determining precise *finite-sample rates* of identification. Namely, with any learner, one can associate a *rate function*, which maps any t to the probability of the learner incorrectly guessing the identity of the target language, upon seeing t i.i.d. examples from a distribution supported on the target language. The results of [KMV25] show that any collection that can be identified in the limit, can also be identified at an *exponential rate* in the statistical setting, and this is effectively the best rate possible. Furthermore, a collection that is not identifiable in the limit does not admit *any vanishing rate*.

We can also consider such a statistical setting for list identification, where the rate function measures the probability that the list output upon seeing t i.i.d. examples does not contain the identity of the target language. In this setting, we show:

Theorem 3 (Statistical Rates for k -list Identification). *Let $\mathcal{C} = (L_1, L_2, \dots)$ be any countable language collection. If \mathcal{C} is k -list identifiable in the limit, then \mathcal{C} can be k -list identified at an exponential rate, and this is the best rate possible. Furthermore, if \mathcal{C} is not k -list identifiable in the limit, then \mathcal{C} cannot be k -list identified at any rate that goes to zero.*

Our result above provides a complete characterization of finite-sample statistical rates possible for k -list identification, analogous to the results of [KMV25] for (vanilla) identification.

1.2 Other Related Work

Language Identification in the Limit. The study of *inductive inference*, starting from the works of [Gol67] and [BB75], has a rich history. Various identification criteria under different names (like **EX**, **BC**, **Fex**, etc.) have been established over the years (see [CS83, OSW86, Cas99, BCJ99]), of which **EX** (Explanatory) and **BC** (Behaviorally Correct) are arguably the two most basic notions. At a high level, the **EX** criterion requires the stronger guarantee that the sequence of guesses converges to the *same* guess (which is also the correct answer) beyond a finite time on every input sequence. On the other hand, the **BC** criterion does not require the sequence of guesses itself to converge to a single guess, but only requires that every guess beyond a finite time be a correct guess (since there could be multiple objects in the collection at different indices that all represent the target object). In this regard, the identification criterion that we are concerned with in this work (and the one described in the introduction above) is the **BC** criterion.

The most general setup for inductive inference involves a *complete presentation* of the graph of a function $f : \mathbb{N} \rightarrow \mathbb{N}$ as input, which is any infinite sequence of pairs $(x_i, f(x_i))_{i \in \mathbb{N}}$, such that for every $j \in \mathbb{N}$, $x_i = j$ for some i . Notably, the function’s values on the entire domain are eventually revealed in the input. This is *unlike* the case in language identification in the limit, where the input only comprises of *positive presentations*; namely, if we associate every language L in the collection with its indicator function $f_L : x \mapsto \mathbb{1}[x \in L]$ over the universe, then the input in the model of language identification can be seen as only comprising of those $(x, f(x))$ pairs for which $f(x) = 1$. We will distinguish these two paradigms in the study of inductive inference explicitly by referring to them as “function identification” and “language identification”.

The works most directly relevant to our setting are those on *identification with a team*, introduced by [Smi82] for function identification, and extended to language identification in the works

of [JS90, JS95a, JS95b, JS96a, JS96b, JS00]. In particular, the notion of $\mathbf{Team}_n^m \mathbf{ID}$ identification considered in these works is defined as follows: of n machines working to identify the target function/language, at least m of them \mathbf{ID} -identify the target language, where \mathbf{ID} could be one of various identification criteria of interest (like \mathbf{EX} , \mathbf{BC} , \mathbf{TxtEX} , \mathbf{TxtBC} , etc.). Translating to these terms, our criterion for k -list identification would be equivalent to the notion of $\mathbf{Team}_k^1 \mathbf{TxtBC}$ identification considered in the latter set of works, provided that the machines are also allowed to communicate with each other. More importantly though, we note that the primary focus in these works was to: 1) study and establish *separations* between collections that can be $\mathbf{Team}_n^m \mathbf{ID}$ identified, but not $\mathbf{Team}_{n'}^{m'} \mathbf{ID}$ identified for $(m, n) \neq (m', n')$, and 2) study connections of team identification with the notion of *probabilistic inductive inference* [Pit85, Pit89]. In contrast, our primary focus in this work is on establishing a *clean* and *exact* Angluin-style tell-tale characterization of k -list identification in the limit, as well as to establish precise finite-sample rates for the same. To the best of our knowledge, prior to our work, such an exact characterization for k -list identification has not been previously established.

Language Generation in the Limit. The inductive inference paradigm has recently regained a lot of attention, largely due to the milestone work of [KM24] on language generation in the limit. Since then, there have been a number of follow-ups on this topic in a very short span of time [KMV25, LRT24, CP24, KMV24, PF25, MAC⁺25]. As mentioned earlier, our motivation to revisit relaxed notions of identification is inspired from the overwhelmingly positive results possible in generation.

List Learning. List prediction, or “list-decodable learning” [BBV08], where the objective is to output a short list of predictions with the goal that at least one of the outputs in the list is accurate, is also a widely studied topic, particularly in settings where data corruptions abound, or when the statistical/computational nature of the problem is pathological in the worst case. Among many works, this includes the extensive literature on list decoding in coding theory [Gur07], works on robust mean estimation [CSV17, DKS18], list classification [CP23, MSTY23] as well as list regression [KKK19, RY20, PS25].

2 Overview of Techniques

We will now give a detailed overview of the various techniques used in establishing our results.

Characterization of k -list Identification. For the purposes of this overview, we will restrict ourselves to the case of 2-list identification, and work with canonical language collections that capture the crux of our characterizing condition, as well as our upper and lower bounds. Consider first the following language collection:

Example 4. Let \mathcal{C} be the language collection whose constituent languages are \mathbb{Z} , as well as $\mathbb{Z} \setminus \{i\}$ for every $i \in \mathbb{Z}$.

The collection in Example 4 is not identifiable in the limit (with a single guess), and it is helpful to see the direct diagonalization argument for this. Here, we will use the term “identifier” to denote a learner that seeks to identify in the limit. Fix any enumeration of the language \mathbb{Z} (e.g., $0, -1, 1, -2, 2, \dots$), and fix any $n_1 \in \mathbb{Z}$. Consider an adversary that starts enumerating $\mathbb{Z} \setminus \{n_1\}$ in the enumeration order that was fixed for \mathbb{Z} . For any valid identifier, upon seeing this sequence as input, there must exist a finite time t_1 at which it outputs a guess i_{t_1} , such that

$L_{i_{t_1}} = \mathbb{Z} \setminus \{n_1\}$. At $t_1 + 1$, the adversary inputs the leftmost number in the enumeration of \mathbb{Z} that has not been enumerated as yet. At this point, note that the adversary has only enumerated finitely many numbers. In particular, there is a number n_2 that the adversary has not yet shown in the input, such that the language $\mathbb{Z} \setminus \{n_2\}$ is consistent with the input presented so far. Hence, the adversary can pretend that it was enumerating $\mathbb{Z} \setminus \{n_2\}$ in the order of \mathbb{Z} all this time, and switch to enumerating $\mathbb{Z} \setminus \{n_2\}$ beyond time $t_1 + 1$. There must now be a finite time t_2 at which the identifier outputs a guess i_{t_2} that satisfies $L_{i_{t_2}} = \mathbb{Z} \setminus \{n_2\}$, at which point the adversary can again output the leftmost number in the enumeration of \mathbb{Z} that has not been enumerated as yet, and then switch to a different $\mathbb{Z} \setminus \{n_3\}$ beyond that. By repeating this ad infinitum, the adversary completely enumerates \mathbb{Z} (since before each “switch point”, the adversary inputs the leftmost number in \mathbb{Z} that has not yet been enumerated), while also inducing an infinite sequence of time steps $t_1 < t_2 < \dots$ at which the identifier does not guess \mathbb{Z} to be the target language. Thus, the identifier fails to identify \mathbb{Z} in the limit on this adversarially constructed input sequence.

Observe however that there is a simple identifier that identifies the collection in [Example 4](#) with a list of size 2. This identifier always maintains \mathbb{Z} as one of its guesses, and constructs its second guess as follows. The identifier fixes an enumeration σ of \mathbb{Z} for itself (e.g., $\sigma = 0, -1, 1, -2, 2, \dots$), and outputs as its second guess the language $\mathbb{Z} \setminus \{i\}$, where i is the *leftmost* number in σ that has *not* yet shown up in the input. Observe that if the target language chosen by the adversary was \mathbb{Z} , then the identifier’s first guess renders it to be correct right from the first time step. On the other hand, if the adversary chose some $\mathbb{Z} \setminus \{i\}$ as its target language, then every number before i in σ will eventually show up in the input, and i will never show up. Thus, the identifier’s second guess will stabilize to $\mathbb{Z} \setminus \{i\}$ after all the numbers before i in σ have appeared in the input. In either case, we have argued that the identifier 2-list identifies the collection in the limit.

Now consider instead the following collection:

Example 5. Let \mathcal{C} be the language collection whose constituent languages are \mathbb{Z} , $\mathbb{Z} \setminus \{i\}$ for every $i \in \mathbb{Z}$, as well as $\mathbb{Z} \setminus \{i, j\}$ for every $i, j \in \mathbb{Z}, i < j$.

Since this collection is a superset of the collection in [Example 4](#), it is also not identifiable in the limit by the same diagonalization argument sketched above. Could this collection be identified with a list of size 2? Let us try to come up with an identifier similar to the one above, where the first guess is maintained to be \mathbb{Z} , so that there is protection from diagonalization against \mathbb{Z} . However, having the second guess be $\mathbb{Z} \setminus \{i\}$ for the leftmost i that has not yet shown up in the input would make the guess converge to $\mathbb{Z} \setminus \{i\}$, *even* when the adversary is enumerating $\mathbb{Z} \setminus \{i, j\}$ for some j that is later than i in the enumeration σ maintained by the identifier. We may then try to maintain a *joint* enumeration σ' of the set \mathbb{Z} together with all the pairs (i, j) for $i < j$, and then output $\mathbb{Z} \setminus F$ for the leftmost F (which is either some $\{i\}$ or $\{i, j\}$) satisfying that F is not a subset of the input. Unfortunately, this strategy fails too: any i must be located at some finite position p_i in σ' , which means that there is some (i, j) that appears later than p_i in σ . The adversary may be enumerating $\mathbb{Z} \setminus \{i, j\}$, but the identifier’s second guess will either converge to $\mathbb{Z} \setminus \{i\}$ or $\mathbb{Z} \setminus \{j\}$, depending on whether i or j comes first in σ' . Essentially, we want the identifier to be able to place *all* the (i, j) pairs for a particular i before i , which is not possible.

As it turns out, the collection in [Example 5](#) is *not* 2-list identifiable. However, the diagonalization argument to show this is more intricate. In particular, observe that the choice of language that was adversarially enumerated to preclude identification in the limit in the case of [Example 4](#) was clear: it was the language \mathbb{Z} , and we capitalized on being able to confuse the identifier to guess a proper subset of \mathbb{Z} at infinitely many time steps. Note however that a 2-list identifier may always have one of its guesses be \mathbb{Z} ! Namely, the identifier may completely hedge against being diagonalized on \mathbb{Z} , in which case the adversary has to look towards some other language in the collection

that it can enumerate in an adversarial manner. This is where having all the $\mathbb{Z} \setminus \{i, j\}$ languages in addition to the $\mathbb{Z} \setminus \{i\}$ languages will come in handy. That is, just like how the adversary could use infinitely many $\mathbb{Z} \setminus \{i\}$ languages to fool against \mathbb{Z} , it can also use infinitely many $\mathbb{Z} \setminus \{i, j\}$ languages to fool against $\mathbb{Z} \setminus \{i\}$. Since the identifier cannot simultaneously “cover all the three bases” of \mathbb{Z} , $\mathbb{Z} \setminus \{i\}$ and $\mathbb{Z} \setminus \{i, j\}$ with just two guesses, it will necessarily be diagonalized against on one of the bases.

Lower Bound. Concretely, consider the following adversarial strategy. The adversary fixes an enumeration of all the languages in the collection, and will keep switching between enumerating different languages; in the limit however, they will have enumerated some fixed language. Whenever the adversary switches to/resumes enumerating a language, it picks back up from the leftmost not-yet-enumerated number. So suppose that: (\star) the adversary starts enumerating \mathbb{Z} . For any valid 2-list identifier, there must exist a finite time t_1 at which one of the two guesses made by the identifier is \mathbb{Z} . At this point, since the adversary has only enumerated finitely many numbers, they may: $(\star\star)$ switch to enumerating any consistent $\mathbb{Z} \setminus \{i\}$. Again, there must exist a finite time t_2 at which one of the two guesses made by the identifier is $\mathbb{Z} \setminus \{i\}$. Crucially, at this time, the adversary *inspects* the other guess made by the identifier. If this guess is not \mathbb{Z} , then the adversary has identified a time step at which none of the two guesses made by the identifier are \mathbb{Z} , and it can go back to (\star) and resume enumerating \mathbb{Z} . Otherwise, the second guess made by the identifier at this time is still maintained to be \mathbb{Z} . In this case, observe that up until t_2 , the adversary has only enumerated finitely many strings from $\mathbb{Z} \setminus \{i\}$. So, they can safely switch to enumerating any $\mathbb{Z} \setminus \{i, j\}$ that is consistent with the input so far. There must now be a finite time t_3 at which the identifier has $\mathbb{Z} \setminus \{i, j\}$ as one of its two guesses. But now the identifier is trapped, in that it must have let go of one of \mathbb{Z} or $\mathbb{Z} \setminus \{i\}$ as its other guess—this is what the adversary capitalizes on. If the second guess is still \mathbb{Z} , then the adversary goes back to $(\star\star)$, and resumes enumerating $\mathbb{Z} \setminus \{i\}$. Otherwise, the adversary goes back to (\star) , and resumes enumerating \mathbb{Z} .

If the adversary repeats this routine ad infinitum, in the limit, one of two cases happens: (i) Either the adversary stops jumping back to (\star) entirely, and keeps jumping back to $(\star\star)$ infinitely often, in which case it will have produced a valid enumeration of some fixed $\mathbb{Z} \setminus \{i\}$. But every jump back to $(\star\star)$ witnesses a time step at which the identifier did not have $\mathbb{Z} \setminus \{i\}$ in its two guesses, which means that the identifier does 2-list identify $\mathbb{Z} \setminus \{i\}$ in the limit. (ii) Or, the adversary jumps back infinitely often to (\star) . In this case, the adversary produces an infinite enumeration of \mathbb{Z} . But every jump back to (\star) witnesses a time step where the identifier did not have \mathbb{Z} in its two guesses. Thus, the identifier does not 2-list identify \mathbb{Z} in the limit.

In any case, the adversarial strategy is guaranteed to make the 2-list identifier fail on some language — either \mathbb{Z} or $\mathbb{Z} \setminus \{i\}$. Crucially, observe how the language that the adversary uses to diagonalize is *dependent* on the 2-list identifier they are interacting with, and only gets determined in the limit. This is fundamentally different from the adversarial strategy used in [Example 4](#), where the adversary can commit to diagonalizing with \mathbb{Z} against all identifiers.

The Characterizing Condition. We can inspect the structure in the collection used in [Example 5](#) more closely to extract a characterizing condition that prevents 2-list identification. Recall that the adversarial strategy finally fooled the identifier on either \mathbb{Z} or some $\mathbb{Z} \setminus \{i\}$. This adversarial strategy was made possible because of the following property: at any finite time in the process of enumerating \mathbb{Z} , the adversary could switch to a *proper subset* of \mathbb{Z} (namely some $\mathbb{Z} \setminus \{i\}$) that was consistent with the input so far (i.e., which contained the finite input enumerated so far). By itself, this property is sufficient for diagonalizing against standard identification in the limit. However,

the additional crucial property that enabled diagonalizing against identification with two guesses, was the fact that at any finite time in which the adversary was also enumerating some $\mathbb{Z} \setminus \{i\}$, they could safely switch to enumerating a *further proper subset* of $\mathbb{Z} \setminus \{i\}$ (namely some $\mathbb{Z} \setminus \{i, j\}$). In other words, the underlying property that enabled the diagonalization was: there exists a language in the collection (namely \mathbb{Z}), such that for every finite subset T of it, there exists a language L' in the collection (namely $\mathbb{Z} \setminus \{i\}$) that contains T and is a proper subset of L , such that furthermore, for every finite subset T' of L' , there exists a language L'' in the collection (namely $\mathbb{Z} \setminus \{i, j\}$) that contains T' and is a proper subset of L' . This is precisely the negation of the condition that characterizes 2-list identification, and the generalization of this condition to k levels is precisely the negation of the k -Angluin condition that we state in [Section 4](#).

Upper Bound. Let us now consider a collection that does not have the pathological structure that the collection in [Example 5](#) has. Namely, every language L contains a finite “first-level tell-tale” set T , such that every language L' that is a proper subset of L necessarily satisfies one of two properties: (1) Either L' does not contain T , or (2) L' itself contains a finite “second-level tell-tale” T' such that every language L'' that is a proper subset of L' does not contain T' . In this case, there is a natural way to utilize the existence of such layered tell-tales and state a recursive 2-list identification algorithm. The algorithm is a generalization of Angluin’s algorithm given in [\[Ang80\]](#), and operates as follows: Suppose the target language is language L_z . At any time step t , the algorithm finds the leftmost language L in the collection that is consistent with the input, and whose first-level tell-tale T is entirely contained in the input. We can then see that beyond a large enough time, this language L stabilizes to the *leftmost* language in the collection that is a *superset* of the target language L_z , and whose first-level tell-tale is contained in L_z . If the collection were in fact identifiable in the limit, we can already stop here: Angluin’s condition would immediately imply that L_z cannot be a proper subset of L , meaning that L must equal L_z . But in our case, it *is* possible for a language L to have proper subset languages that contain its tell-tale. Nevertheless, the condition above stipulates that any such language L better contain a *second-level* tell-tale. So, we continue reasoning as follows: we set L to be the first of the two guesses made by the algorithm, and then recursively run the same process on all the languages to the *right* of L , that are proper subsets of L , and contain its first-level tell-tale T ! It is not too hard to see that by virtue of the condition, either L was equal to L_z in the first place, or the second-level tell-tale of L_z suffices to discern it from all the other proper subsets of L that contained its first-level tell-tale.

Stratification of the Collection. The recursive algorithm above, as well as the layered nature of our condition, naturally also inspire a greedy procedure to stratify a 2-list identifiable collection into two separate identifiable collections. By the characterization above, any 2-list identifiable collection must satisfy our condition of having suitable first-level and second-level tell-tales. The greedy procedure is then as follows: let us construct a directed graph over the languages in the collection, where we draw a directed edge $L \rightarrow L'$ for every pair L, L' which satisfies that L' is a proper subset of L , and L' contains the first-level tell-tale of L . Intuitively, for any such pair L, L' , the first-level tell-tale does not suffice to discern between them. So, we set $\mathcal{C}_1 \subseteq \mathcal{C}$ to comprise of all the languages $L \in \mathcal{C}$, that do not have any directed edge coming into it. Then, for any pair of languages L, L' in \mathcal{C}_1 , it must be the case that neither of them contain the other’s first-level tell-tale (otherwise, one of them would have a directed edge coming into it, and would not be added to \mathcal{C}_1). In other words, within \mathcal{C}_1 , the first-level tell-tales of every language suffice as tell-tales in the standard version of Angluin’s condition; hence, \mathcal{C}_1 is identifiable in the limit. On the other hand, observe also that any language L' in $\mathcal{C} \setminus \mathcal{C}_1 := \mathcal{C}_2$ must also contain a second-level tell-tale, by virtue

of it satisfying our nested condition. This is because every language $L' \in \mathcal{C}_2$ satisfies that it is a proper subset of some other language L , and that it also contains the first-level tell-tale of L . Thus, within \mathcal{C}_2 , the second-level tell-tales of all languages suffice as tell-tales in the standard version of Angluin’s condition, making \mathcal{C}_2 also identifiable in the limit. We have thus partitioned \mathcal{C} into \mathcal{C}_1 and \mathcal{C}_2 in a way that both \mathcal{C}_1 and \mathcal{C}_2 are identifiable in the limit. We can repeat this greedy peeling procedure k times over residual collections to generalize to the case where the collection is k -list identifiable in the limit.

Statistical Rates. We will now describe our results on finite-sample rates in the statistical setting. These results extend those of [KMV25] for identification pointwise to the list identification setting.

We will first argue that any collection that is k -list identifiable in the limit can be k -list identified at an exponential rate. From our stratification result above, we know that if \mathcal{C} is k -list identifiable, it can be decomposed into k individually identifiable collections. Since [KMV25] show that every identifiable collection can be identified at an exponential rate, this is already sufficient to conclude that \mathcal{C} may be k -list identified at an exponential rate: simply break \mathcal{C} into $\mathcal{C}_1, \dots, \mathcal{C}_k$, and concatenate the outputs of individual identifiers (that achieve an exponential rate) running on each of these collections. Since the target language belongs to one of the k collections, the identifier running on that collection yields the desired exponential rate for list identification.

It is also straightforward to see that an exponential rate is essentially the best rate possible.² In particular, suppose that the collection contains $k + 1$ distinct languages that all share a string x , and consider a distribution that assigns mass $1/2$ to x . Then, there is at least a 2^{-t} chance that an i.i.d. sequence of t examples drawn from the distribution comprises only of x . In this case, since the algorithm only outputs k guesses at any time step, at least one of the $k + 1$ languages is not identified at time t if this input sequence is realized. With a couple more steps of simple reasoning, this argument can be formalized to show that a rate faster than exponential is not possible.

Similar to [KMV25], the major bulk of the work goes into showing that a collection that is not k -list identifiable in the limit cannot be k -list identified at *any* vanishing rate. Here, we note that [KMV25] were able to directly use a previous result by Angluin [Ang88] in their proof, which relates identification with respect to an i.i.d. stream to the standard online setting of identification. In fact, Angluin draws this connection by using results established by Pitt [Pit85] in the related setting of *probabilistic* identification in the limit.³ Essentially, Pitt relates identification in the limit to probabilistic identification in the limit, which Angluin in turn relates to identification with respect to an i.i.d. sequence. While it sufficed for [KMV25] to directly instantiate the end result, we are not afforded this convenience, and have to carefully re-derive the entire bridge ourselves for the setting of list identification. In particular, we show a standalone result analogous to that obtained by Pitt, relating k -list identification to probabilistic k -list identification. The main part of this result, which converts a probabilistic k -list identifier to a standard deterministic k -list identifier, requires several ideas, including a careful “Top- k ” aggregation scheme over the predictions made by different nodes in the *computation tree* of a probabilistic k -list identifier. We also redo Angluin’s part of the analysis in more detail, relating probabilistic k -list identification to k -list identification over an i.i.d. stream of examples. The end result of this bridge asserts that for any collection that is not k -list identifiable in the limit, for any k -list identifier, there exists a distribution over a target language such that with probability at least $\frac{1}{k+1}$ over an i.i.d. stream of examples drawn from

²upto “trivial” collections that can always be list-identified with a single example, see Section 8.3.

³In probabilistic identification, an identifier is allowed to be randomized, and we care about the probability (over the random coins of the identifier) that the guesses of the identifier converge to the target language on any fixed sequence.

this distribution, the k -list identifier fails to k -list identify the target language in the limit. The last part of the analysis then shows how any k -list identifier that achieves a vanishing rate with respect to arbitrary distributions may be boosted up to k -list identify from an i.i.d. stream from any distribution with probability 1, which contradicts the previous assertion.

Roadmap. With this overview, we now move on to formally deriving all our results. We first state all the required technical preliminaries in [Section 3](#). We then precisely state the k -Angluin condition for k -list identification in [Section 4](#). We derive a k -list identification algorithm for any collection that satisfies this condition in [Section 5](#). [Section 6](#) contains the diagonalization argument which shows that the condition is in fact necessary for k -list identification. In [Section 7](#), we detail the structural stratification result which decomposes any k -list identifiable collection into k identifiable collections. Finally, we derive all results related to statistical rates, together with intermediate results on probabilistic list identification, in [Section 8](#).

3 Preliminaries

$\mathbb{N} = \{1, 2, 3, \dots\}$ denotes the set of natural numbers, and $\mathbb{Z} = \{\dots, -2, -1, 0, 1, 2, \dots\}$ denotes the set of integers. We assume a countable collection $\mathcal{C} = (L_1, L_2, \dots)$ of languages, where every language is a subset of a countable universe U . An enumeration of a language L is a sequence x_1, x_2, \dots which satisfies that every $x_i \in L$, and furthermore, for every $x \in L$, there exists i such that $x_i = x$. We only assume that every language in the collection is non-empty; otherwise, it may be finite, or countably infinite. Throughout the paper, we will reserve the variable k to denote the size of the list output by a learner that is trying to do list identification, or a *list identifier*.

Definition 1 (List Identifier). *A list identifier with list size k , or a k -list identifier, is a function⁴ $\mathcal{A} : U^* \rightarrow \mathbb{N}^k$, which at any time step t , takes as input a finite ordered sequence of strings x_1, \dots, x_t , and outputs a list of indices $\mu_t = \mathcal{A}(x_1, \dots, x_t) = (i_1, \dots, i_k)$.*

For a language $L \in \mathcal{C}$, and a list $\mu \in \mathbb{N}^*$, we use the notation $L \in_{\text{id}} \mu$ to denote that $\exists i \in \mu$ such that $L_i = L$. In other words, μ contains the “identity” of language L , upto equivalent copies of L at different indices in \mathcal{C} . Similarly, $L \notin_{\text{id}} \mu$ denotes that $\forall i \in \mu, L_i \neq L$. The notation $\mu \ni_{\text{id}} L_z$ and $\mu \not\ni_{\text{id}} L_z$ represents the corresponding relations with the order of the arguments reversed.

Definition 2 (List Identification in the Limit). *A k -list identifier \mathcal{A} identifies a countable collection $\mathcal{C} = (L_1, L_2, \dots)$ in the limit if for every language $L_z \in \mathcal{C}$, and for every enumeration x_1, x_2, \dots of L_z presented to \mathcal{A} as input, there exists a finite time t^* such that for every $t \geq t^*$, the list $\mu_t = \mathcal{A}(x_1, \dots, x_t)$ output by \mathcal{A} satisfies that $L_z \in_{\text{id}} \mu_t$.*

We now define the notion of a *valid distribution*, which is any distribution that is supported on some language in the collection.

Definition 3 (Valid Distribution). *A distribution \mathcal{D} over U is said to be valid for a language L if $\mathcal{D}(x) > 0$ if and only if $x \in L$.*

We can then describe statistical rates that an identifier may attain with respect to valid distributions.

⁴In this paper, we only focus on the existence of list-valued functions that can identify in the limit; we will not be concerned about the computational power required to compute this function.

Definition 4 (List Identification Rates). *For a countable language collection \mathcal{C} , and a rate function $R : \mathbb{N} \rightarrow [0, 1]$ satisfying $\lim_{t \rightarrow \infty} R(t) = 0$, we say that:*

- \mathcal{C} can be k -list identified at rate R if there exists a k -list identifier \mathcal{A} which satisfies that for every language $L_z \in \mathcal{C}$, and for every distribution \mathcal{D} that is valid for L_z , there exist constants $c_1 = c_1(L_z, \mathcal{D}, \mathcal{C}) > 0$ and $c_2 = c_2(L_z, \mathcal{D}, \mathcal{C}) > 0$ such that for every $t \in \mathbb{N}$,

$$\Pr_{x_1, \dots, x_t \sim \mathcal{D}^t}[\mathcal{A}(x_1, \dots, x_t) \not\equiv_{\text{id}} L_z] \leq c_1 \cdot R(c_2 \cdot t). \quad (1)$$

- \mathcal{C} cannot be k -list identified at a rate faster than R if for every k -list identifier \mathcal{A} , there exists a language $L_z \in \mathcal{C}$, a distribution \mathcal{D} valid for L_z , and constants $c_1 = c_1(L_z, \mathcal{D}, \mathcal{C}) > 0$, $c_2 = c_2(L_z, \mathcal{D}, \mathcal{C}) > 0$ such that for infinitely many $t \in \mathbb{N}$,

$$\Pr_{x_1, \dots, x_t \sim \mathcal{D}^t}[\mathcal{A}(x_1, \dots, x_t) \not\equiv_{\text{id}} L_z] \geq c_1 \cdot R(c_2 \cdot t). \quad (2)$$

Furthermore, we have the following:

- (Optimal Rate) Rate R is the optimal rate at which \mathcal{C} can be k -list identified if \mathcal{C} can be k -list identified at rate R , but cannot be k -list identified at a rate faster than R .
- (No Rate) \mathcal{C} cannot be k -list identified at any rate if for every k -list identifier \mathcal{A} , there exists a language $L_z \in \mathcal{C}$ and a distribution \mathcal{D} valid for L_z such that

$$\limsup_{t \rightarrow \infty} \Pr_{x_1, \dots, x_t \sim \mathcal{D}^t}[\mathcal{A}(x_1, \dots, x_t) \not\equiv_{\text{id}} L_z] > 0. \quad (3)$$

4 The k -Angluin Condition

We begin by precisely stating the k -Angluin condition that characterizes identification in the limit with a list, and plays a central role throughout the rest of the paper. Fix a countable language collection $\mathcal{C} = (L_1, L_2, \dots)$. For language L_i at index i in \mathcal{C} and list size parameter $k \geq 1$, define the predicate $\Psi(L_i, k)$ ⁵ inductively as follows:

Base Predicate.

$$\Psi(L_i, 1) := \exists \text{ finite } T_i^{(1)} \subseteq L_i \text{ such that } \forall j \in \mathbb{N}, \text{ if } L_j \subsetneq L_i \text{ then } T_i^{(1)} \not\subseteq L_j. \quad (4)$$

Inductive Predicate. $\forall k > 1 :$

$$\Psi(L_i, k) := \exists \text{ finite } T_i^{(k)} \subseteq L_i \text{ such that } \forall j \in \mathbb{N}, \text{ if } L_j \subsetneq L_i \text{ then either } T_i^{(k)} \not\subseteq L_j \text{ or } \Psi(L_j, k-1). \quad (5)$$

The precise “ k -Angluin” condition that characterizes k -list identification is then as follows:

$$\boxed{\forall i \in \mathbb{N}, \Psi(L_i, k).} \quad (6)$$

The sets $T_i^{(k)}$ are known as *tell-tale* sets for the language L_i . We note that (6) instantiated with $k = 1$, i.e., simply having the base predicate (4) hold for every language is precisely Angluin’s condition that characterizes standard language identification in the limit without a list [Ang80].

⁵We remark that the predicate Ψ has an implicit dependence on \mathcal{C} (which we have fixed); we suppress this dependence in the notation for readability.

Remark 1 (Uniqueness of tell-tales across levels). Suppose we have that $\Psi(L_1, 2)$ and $\Psi(L_2, 2)$ for two languages L_1 and L_2 in \mathcal{C} . Then, consider some $T_1^{(2)}$ and $T_2^{(2)}$ that satisfy the respective predicates, and suppose it is the case that language $L_3 \in \mathcal{C}$ is a proper subset of both L_1 and L_2 , and also contains both of $T_1^{(2)}$ and $T_2^{(2)}$. Then, according to (5), in order for $\Psi(L_1, 2)$ to be true, $\Psi(L_3, 1)$ needs to be true, which implies the existence of a $T_3^{(1)} \subseteq L_3$ satisfying (4). Similarly, in order for $\Psi(L_2, 2)$ to be true, $\Psi(L_3, 1)$ needs to be true again, which, strictly speaking, implies the existence of a $\tilde{T}_3^{(1)} \subseteq L_3$ satisfying (4). Notably, observe that the definition allows for $T_3^{(1)}$ and $\tilde{T}_3^{(1)}$ to be different: as long as (4) is satisfied for both, there is no contradiction to either of $\Psi(L_1, 2)$ or $\Psi(L_2, 2)$. Nevertheless, observe that we can have one of $T_3^{(1)}$ or $\tilde{T}_3^{(1)}$ simultaneously satisfy both the “separate instantiations” of $\Psi(L_3, 1)$. Therefore, without loss of generality, whenever we consider an instantiation of the predicate $\Psi(L_i, k)$ to hold, we can assume it to hold with a unique tell-tale $T_i^{(k)}$. We note also that if $\Psi(L_i, k)$ holds for any language L_i , the tell-tale $T_i^{(k)}$ that makes $\Psi(L_i, k)$ hold also suffices to make $\Psi(L_i, k+1)$ hold. Thus, for any language L_i , the tell-tale $T_i^{(k)}$ for the smallest k for which $\Psi(L_i, k)$ holds suffices as a tell-tale for every predicate $\Psi(L_i, k')$ where $k' \geq k$.

Remark 2 (Canonical Collections). It is helpful to keep in mind the suite of canonical collections $(\mathcal{C}_k)_{k \geq 1}$, where $\mathcal{C}_k = \{\mathbb{Z}\} \cup \{\mathbb{Z} \setminus F : F \subseteq \mathbb{Z}, |F| \leq k\}$. Namely, every language in \mathcal{C}_k excludes a finite subset \mathbb{Z} of size at most k . We can observe that \mathcal{C}_k does not satisfy the k -Angluin condition, but satisfies the $(k+1)$ -Angluin condition. To see why \mathcal{C}_k does not satisfy the k -Angluin condition, observe that no matter what finite subset $T \subseteq \mathbb{Z}$ we try to ascribe to the language \mathbb{Z} for the purposes of satisfying $\Psi(\mathbb{Z}, k)$, there are many languages $\mathbb{Z} \setminus \{i\}$ that are proper subsets of \mathbb{Z} , contain T , and also do not satisfy $\Psi(\mathbb{Z} \setminus \{i\}, k-1)$. On the other hand, any arbitrary assignment of finite tell-tale sets $T_i^{(k')}$ for every language $L_i \in \mathcal{C}_k$ and $k' \leq k$ suffices for the purposes of satisfying the $(k+1)$ -Angluin condition for \mathcal{C}_k . To see this, notice that once we fix a tell-tale set $T_i^{(k')}$ for some language $L_i = \mathbb{Z} \setminus F$ towards the purpose of satisfying $\Psi(L_i, k')$, the languages L_j that are proper subsets of L_i necessarily satisfy that $L_j = \mathbb{Z} \setminus G$ for $|G| < |F|$. Hence, the languages L_i for which we recursively hit the base predicate $\Psi(L_i, 1)$ all satisfy that $L_i = \mathbb{Z} \setminus F$ where $|F| = 1$, and no two languages of this form are proper subsets of each other. We also note that the collection $\mathcal{C}_\infty = \{\mathbb{Z}\} \cup \{\mathbb{Z} \setminus F : F \subseteq \mathbb{Z}, |F| < \infty\}$ does not satisfy the k -Angluin collection for any finite value of k .

5 Upper Bound

We will now show that any countable collection that satisfies the k -Angluin condition can be k -list identified in the limit. Towards this, consider Algorithm 1 for k -list identification.

Observe that for any k, I, S , the size of the list output by `ListIdentify`(k, I, S) is at most k . This is because every time we recursively invoke `ListIdentify`, we append a language to the output, but also decrease k by 1; furthermore, we return \emptyset when $k = 0$. This establishes that the size of the list output is at most k .

We now claim that when the algorithm above is invoked on increasing prefixes of an input enumeration, the index i^* determined in Algorithm 1 stabilizes beyond some finite time step.

Claim 5.1 (i^* stabilizes). *Let the index of the target language being enumerated be z , and let the enumeration be (x_1, x_2, \dots) . Let I be any set of indices that contains z , and let $k \geq 1$ be such that $\forall i \in I, \Psi(L_i, k)$. Denote by S_t the set of distinct strings in a finite prefix (x_1, \dots, x_t) of*

⁶We define \min over an empty set to be ∞ .

Algorithm 1: List Identification Algorithm

Input: List size k , set of indices I satisfying $\forall i \in I : \Psi(L_i, k)$, finite dataset S

Output: A list of size at most k

```
1 Procedure ListIdentify( $k, I, S$ ):  
2   if  $I$  is empty or  $k = 0$  then  
3     return  $\emptyset$   
4   end  
6    $i^* \leftarrow \min \{i \in I : S \subseteq L_i \text{ and } S \supseteq T_i^{(k)}\}^6$   
7   if  $i^* = \infty$  then  
8     return  $\{1\}$   
9   end  
11   $I' \leftarrow \{j \in I : j > i^* \text{ and } L_j \subsetneq L_{i^*} \text{ and } T_{i^*}^{(k)} \subseteq L_j\}$   
12  return  $\{i^*\} \cup \text{ListIdentify}(k-1, I', S)$ 
```

the enumeration. Then, there exists a finite time step t^* , such that for every $t \geq t^*$, the index i^* determined in [Algorithm 1](#) of [Algorithm 1](#) upon invoking `ListIdentify`(k, I, S_t) satisfies

$$i^* = \min \left\{ i \in I : i \leq z \text{ and } L_i \supseteq L_z \text{ and } L_z \supseteq T_i^{(k)} \right\}. \quad (7)$$

Note that $i = z$ satisfies the condition, and hence $i \neq \infty$ for $t \geq t^*$.

Proof. Consider first any $i \in I$ for which $i \leq z$ and $L_i \supseteq L_z$, but $L_z \not\supseteq T_i^{(k)}$. Note that such an L_i never satisfies $S_t \supseteq T_i^{(k)}$ for any t . This is simply because $T_i^{(k)}$ contains some string $x \notin L_z$ which will never show up in the input. Thus, such an i never becomes feasible in [Algorithm 1](#).

Now consider the set $J = \{i \in I : i \leq z \text{ and } L_i \supseteq L_z \text{ and } L_z \supseteq T_i^{(k)}\}$, which is precisely the set in (7). Note first that J is a finite and non-empty set, since z belongs to it. Furthermore, for any $i \in J$, observe that $S_t \subseteq L_i$ for every t , simply because the input is an enumeration of L_z , and $L_i \supseteq L_z$. We now claim that there exists a finite time t_1 , such that for every $t \geq t_1$, $S_t \supseteq T_i^{(k)}$ for every $i \in J$. Again, this is true because $L_z \supseteq T_i^{(k)}$ for every $i \in J$, and since there are finitely many $i \in J$, and each $T_i^{(k)}$ is also a finite set, there is a finite time t_1 when all of $\cup_{i \in J} T_i^{(k)}$ shows up in the input. Thus, for every $t \geq t_1$, every L_i for $i \in J$ is feasible in [Algorithm 1](#).

Finally, observe also there exists a finite time step t_2 , such that for every $t \geq t_2$, every $i \in I$ for which $i \leq z$ and $L_i \not\supseteq L_z$ satisfies that $S_t \not\subseteq L_i$: this is simply because for every such i , L_z contains some string $x_i \notin L_i$, and this x_i appears in the enumeration at some finite time. Since there are only finitely many such i , there is a finite time t_2 when the input will contain x_i for every i . Thus, for every $t \geq t_2$, we have that every such i is infeasible in [Algorithm 1](#).

Combining the reasoning above, we can conclude that for every $t \geq \max(t_1, t_2)$, the index t^* determined in [Algorithm 1](#) satisfies

$$i^* = \min \left\{ i \in I : i \leq z \text{ and } L_i \supseteq L_z \text{ and } L_z \supseteq T_i^{(k)} \right\}.$$

□

Note that the expression for i^* in (7) does not depend on t , and hence i^* has stabilized to this value for every $t \geq t^*$.

We are now ready to argue that any collection that satisfies the k -Angluin condition can be k -list identified by [Algorithm 1](#).

Theorem 6 (Upper Bound). *Let $\mathcal{C} = (L_1, L_2, \dots)$ be a countable collection that satisfies the k -Angluin condition given in (6). Let L_z be the target language being enumerated as (x_1, x_2, \dots) , and let S_t denote the set of distinct strings in a finite prefix (x_1, \dots, x_t) of the enumeration. Then, there exists a finite t^* , such that for every $t \geq t^*$, $L_z \in_{\text{id}} \text{ListIdentify}(k, \mathbb{N}, S_t)$.*

Proof. Let $I_k = \mathbb{N}$. The set I_k contains the index z of the target language L_z , and since \mathcal{C} satisfies the k -Angluin condition, we have that $\forall i \in I_k, \Psi(L_i, k)$. Hence, the requirements of Claim 5.1 are satisfied, implying the existence of a finite time step t_k^* , such that for every $t \geq t_k^*$, the index i_k^* determined in Algorithm 1 of $\text{ListIdentify}(k, I_k, S_t)$ is

$$i_k^* = \min \left\{ i \in I_k : i \leq z \text{ and } L_i \supseteq L_z \text{ and } L_z \supseteq T_i^{(k)} \right\}.$$

Furthermore, i_k^* is always included in the output list for $t \geq t_k^*$. Now, either $L_{i_k^*} = L_z$, in which case we are done. Otherwise, $i_k^* < z$, and $L_{i_k^*}$ is a proper superset of L_z for which $L_z \supseteq T_{i_k^*}^{(k)}$. Now, since i_k^* has stabilized for $t \geq t_k^*$, the set of indices determined in Algorithm 1 also stabilizes for $t \geq t_k^*$. Denote this set by I_{k-1} , and recall that

$$I_{k-1} = \{j \in I_k : j > i_k^* \text{ and } L_j \subsetneq L_{i_k^*} \text{ and } T_{i_k^*}^{(k)} \subseteq L_j\}.$$

By our previous reasoning, $z \in I_{k-1}$. Furthermore, recall that $\Psi(L_{i_k^*}, k)$ holds, and every index j in I_{k-1} corresponds to a language L_j that is a proper subset of $L_{i_k^*}$ and also contains $T_{i_k^*}^{(k)}$. Recalling (5), this implies that $\Psi(L_j, k-1)$ holds for every $j \in I_{k-1}$.

We then repeat the above argument on the recursive invocation to $\text{ListIdentify}(k-1, I_{k-1}, S_t)$ for $t \geq t_k^*$. As argued above, the set I_{k-1} contains the index z of the target language, and $\Psi(L_j, k-1)$ holds for every $j \in I_{k-1}$. The requirements of Claim 5.1 are again satisfied, implying the existence of a finite time step t_{k-1}^* , such that for every $t \geq \max(t_k^*, t_{k-1}^*)$, the index i_{k-1}^* determined in Algorithm 1 of $\text{ListIdentify}(k-1, I_{k-1}, S_t)$ is

$$i_{k-1}^* = \min \left\{ i \in I_{k-1} : i \leq z \text{ and } L_i \supseteq L_z \text{ and } L_z \supseteq T_i^{(k-1)} \right\}.$$

Furthermore, i_{k-1}^* is always included in the output list for $t \geq \max(t_k^*, t_{k-1}^*)$. Again, if $L_{i_{k-1}^*} = L_z$, we are done. Otherwise, $i_{k-1}^* < z$, and $L_{i_{k-1}^*}$ is a proper superset of L_z for which $L_z \supseteq T_{i_{k-1}^*}^{(k-1)}$. Since i_{k-1}^* has stabilized for $t \geq \max(t_k^*, t_{k-1}^*)$, the set of indices determined in Algorithm 1 also stabilizes. Denote this set by I_{k-2} , and recall that

$$I_{k-2} = \{j \in I_{k-1} : j > i_{k-1}^* \text{ and } L_j \subsetneq L_{i_{k-1}^*} \text{ and } T_{i_{k-1}^*}^{(k-1)} \subseteq L_j\}.$$

By our previous reasoning, $z \in I_{k-2}$. Furthermore, recall that $\Psi(L_{i_{k-1}^*}, k-1)$ holds, and every index j in I_{k-2} corresponds to a language L_j that is a proper subset of $L_{i_{k-1}^*}$ and also contains $T_{i_{k-1}^*}^{(k-1)}$. Recalling (5), this implies that $\Psi(L_j, k-2)$ holds for every $j \in I_{k-2}$.

Continuing thus all the way down till $k = 1$, we would either have that $L_z \in_{\text{id}} (i_k^*, \dots, i_2^*)$ for every $t \geq \max(t_k^*, \dots, t_2^*)$. Otherwise, we consider the invocation to $\text{ListIdentify}(1, I_1, S_t)$ for $t \geq \max(t_k^*, \dots, t_2^*)$. Recall that we ensure that $\Psi(L_j, 1)$ holds for every $j \in I_1$, and that I_1 contains the index z of the target language. Invoking Claim 5.1 one final time, there exists a finite time t_1^* such that for every $t \geq \max(t_k^*, \dots, t_1^*)$, the index i_1^* determined in Algorithm 1 of $\text{ListIdentify}(1, I_1, S_t)$ is

$$i_1^* = \min \left\{ i \in I_1 : i \leq z \text{ and } L_i \supseteq L_z \text{ and } L_z \supseteq T_i^{(1)} \right\}.$$

Algorithm 2: Adversarial enumeration strategy against k -list identifier \mathcal{A}

Input: Ordered sequence of indices **chain**, input S enumerated so far

Output: A valid enumeration σ of some language in \mathcal{C}

```
1 Procedure AdvEnum(chain,  $S$ ):  
2   Suppose chain :=  $(i_1, \dots, i_\ell)$   
3   while True do  
4      $x \leftarrow$  first string in the enumeration of  $L_{i_\ell}$  that hasn't yet appeared in  $S$   
5      $S \leftarrow S \circ x$   
6     if  $L_{i_\ell} \in_{\text{id}} \mathcal{A}(S)$  then  
7       break  
8     end  
9   end  
10  if  $\forall j \in [\ell], L_{i_j} \in_{\text{id}} \mathcal{A}(S)$  then  
11    Let  $i_{\ell+1} \in \mathbb{N}$  be such that  $L_{i_{\ell+1}} \subsetneq L_{i_\ell}$ ,  $L_{i_{\ell+1}} \supseteq S$  and  $\neg\Psi(L_{i_{\ell+1}}, k - \ell)$ ; terminate if  
12     $k - \ell < 0$ , or if no such  $i_{\ell+1}$  exists  
13    chain  $\leftarrow (i_1, \dots, i_\ell, i_{\ell+1})$   
14    AdvEnum(chain,  $S$ )  
15  end  
16  else  
17     $j \leftarrow \min\{j \in [\ell] : L_{i_j} \notin_{\text{id}} \mathcal{A}(S)\}$   
18    chain  $\leftarrow (i_1, \dots, i_j)$   
19    AdvEnum(chain,  $S$ )  
20  end
```

Furthermore, i_1^* is now always included in the output list for $t \geq \max(t_k^*, \dots, t_1^*)$. We now claim that if we are indeed in this case, $L_{i_1^*}$ must indeed *equal* L_z . Otherwise, if $L_{i_1^*}$ is a proper superset of L_z , it would be the case that $L_z \subsetneq L_{i_1^*}, T_{i_1^*}^{(1)} \subseteq L_z$, which is a contradiction to $\Psi(L_{i_1^*}, 1)$ (see (4)). We conclude that beyond $t \geq \max(t_k^*, \dots, t_1^*)$, **ListIdentify**(k, \mathbb{N}, S_t) identifies L_z . \square

6 Lower Bound

We next proceed to showing that the k -Angluin condition is necessary for k -list identification in the limit.

Theorem 7 (Lower Bound). *Let $\mathcal{C} = (L_1, L_2, \dots)$ be a countable collection that does not satisfy the k -Angluin condition given in (6), for $k \geq 1$. Then, \mathcal{C} cannot be identified in the limit with a list of size k .*

Proof. For any k -list identifier \mathcal{A} , consider the adversarial enumeration strategy specified in [Algorithm 2](#).

Now, since \mathcal{C} does not satisfy the k -Angluin condition, there exists $i_1 \in \mathbb{N}$ such that $\neg\Psi(L_{i_1}, k)$ holds. Referring to (5), this means that

$$\forall \text{ finite } T \subseteq L_{i_1}, \exists i_2 \in \mathbb{N} \text{ such that } L_{i_2} \subsetneq L_{i_1} \text{ and } T \subseteq L_{i_2} \text{ and } \neg\Psi(L_{i_2}, k - 1).$$

Here, we adopt the convention that $\Psi(L_{i_2}, 0) = \text{False}$ for every $i_2 \in \mathbb{N}$.

We now claim that the list-identifier \mathcal{A} fails the list identification criterion for \mathcal{C} on the infinite sequence S populated upon invoking $\text{AdvEnum}((i_1), ())$. We first establish a few invariants of [Algorithm 2](#) upon this invocation.

Invariant 1: Whenever $\text{AdvEnum}(\text{chain}, S)$ is invoked, chain is a non-empty sequence of indices (i_1, \dots, i_ℓ) , such that $L_{i_1} \supsetneq L_{i_2} \supsetneq \dots \supsetneq L_{i_\ell}$. Furthermore, we also have that $\neg\Psi(L_{i_1}, k), \neg\Psi(L_{i_2}, k-1), \dots, \neg\Psi(L_{i_\ell}, k-(\ell-1))$, and that $S \subseteq L_{i_j}$ for every $j \in [\ell]$.

To see this, we argue inductively: observe that this is true in the base case when we invoke $\text{AdvEnum}((i_1), ())$, either vacuously or by choice of i_1 . Now, suppose that the invariant is true when $\text{AdvEnum}(\text{chain}, S)$ is invoked—we will argue that the invariant holds true at a subsequent invocation. Note that if there is a subsequent invocation at all, it must be the case that the while loop in [Algorithm 2](#) breaks. Then, if we are in the case that $\forall j \in [\ell], L_{i_j} \in_{\text{id}} \mathcal{A}(S)$, we must not terminate in [Algorithm 2](#); so we will go on to append $i_{\ell+1}$ that we found to chain , which is guaranteed to satisfy $S \subseteq L_{i_{\ell+1}}, L_{i_{\ell+1}} \subsetneq L_{i_\ell}$, as well as $\neg\Psi(L_{i_{\ell+1}}, k-\ell)$ by construction. Thus, the invariant continues to hold true at the next invocation in [Algorithm 2](#). Otherwise, if we are in the case of the else condition, we trim chain to be (i_1, \dots, i_j) ; then, the invariant continues to hold true at the next invocation in [Algorithm 2](#), due to the fact that all newly added strings to S in the while loop in the present invocation were from $L_{i_\ell} \subseteq L_{i_j}$, together with the inductive hypothesis.

The invariant above ensures that at the beginning of any invocation to $\text{AdvEnum}(\text{chain}, S)$, S is a valid *partial* enumeration of L_{i_ℓ} , and the while loop in [Algorithm 2](#) *resumes* enumerating L_{i_ℓ} (namely, if the loop goes on endlessly, S gets completed to be a valid *complete* enumeration of L_{i_ℓ}).

Invariant 2: Whenever $\text{AdvEnum}(\text{chain}, S)$ is invoked, $|\text{chain}| \leq k+1$.

To see this, note that we begin with $|\text{chain}| = 1$ at the very first invocation $\text{AdvEnum}((i_1), ())$. Thereafter, assuming no termination/infinite loops, we either append a single language to chain in the case of the if condition, or we strictly *reduce* the size of chain in the other case—this is because in [Algorithm 2](#), j is necessarily determined to be strictly smaller than ℓ , since the while loop broke with $L_{i_\ell} \in_{\text{id}} \mathcal{A}(S)$. So, it suffices to argue that at an invocation of $\text{AdvEnum}(\text{chain}, S)$ with $|\text{chain}| = k+1$, the if condition, namely $\forall j \in [\ell], L_{i_j} \in_{\text{id}} \mathcal{A}(S)$, can never be satisfied. But this is true simply because $\mathcal{A}(S)$ is a list of size at most k , and the languages at the indices in chain are all distinct (e.g., by Invariant 1). This implies that at least one index $i_j \in \text{chain}$ satisfies $L_{i_j} \notin_{\text{id}} \mathcal{A}(S)$. Thus, we have established the invariant.

Invariant 3: The termination condition in [Algorithm 2](#) is never realized during any invocation.

This follows from Invariant 1 and Invariant 2. The reasoning for Invariant 2 establishes that after the while loop breaks, the if condition $\forall j \in [\ell], L_{i_j} \in_{\text{id}} \mathcal{A}(S)$ can be satisfied only if $\ell = |\text{chain}| \leq k$; but this ensures that $k-\ell$ is non-negative. Furthermore, from Invariant 1, we also know that $\neg\Psi(L_{i_\ell}, k-(\ell-1))$ holds. By definition, this means that

$$\forall \text{ finite } T \subseteq L_{i_\ell}, \exists i_{\ell+1} \in \mathbb{N} \text{ such that } L_{i_{\ell+1}} \subsetneq L_{i_\ell} \text{ and } T \subseteq L_{i_{\ell+1}} \text{ and } \neg\Psi(L_{i_{\ell+1}}, k-\ell).$$

Taking $T = S$ ensures the existence of the required $i_{\ell+1}$, and establishes the invariant.

With these invariants established, we can now proceed towards finishing the proof. Consider first the case where the while loop in [Algorithm 2](#) never breaks in some invocation to $\text{AdvEnum}(\text{chain}, S)$. By Invariant 1, this means that the loop is validly completing the enumeration of the language L_{i_ℓ} for $i_\ell \in \text{chain}$, and L_{i_ℓ} is never identified in the list output by \mathcal{A} ; this makes \mathcal{A} an invalid list identifier.

So, assume that the while loop breaks in every invocation of $\text{AdvEnum}(\text{chain}, S)$. By Invariant 3 above, we know that there is never a termination. This means that there is an infinite sequence of invocations of $\text{AdvEnum}(\text{chain}, S)$. Let $\text{chain}_1, \text{chain}_2, \dots$ be the corresponding chain parameters that each AdvEnum is invoked with in this sequence. Define

$$\ell^* := \liminf_{n \rightarrow \infty} |\text{chain}_n|.$$

Note that ℓ^* always exists, and furthermore belongs to the set $\{1, \dots, k+1\}$, since all the numbers in the sequence belong to this finite set. That is, there exists a large enough n^* , such that for every $n \geq n^*$, $|\text{chain}_n| \geq \ell^*$. This means that the indices i_1, \dots, i_{ℓ^*} stay fixed in chain_n for every $n \geq n^*$. It is also the case that there are infinitely many $n \geq n^*$ where the invocation $\text{AdvEnum}(\text{chain}_n, S)$ satisfies $|\text{chain}_n| = \ell^*$. These invocations occur precisely when the previous invocation (which satisfies $|\text{chain}_{n-1}| > \ell^*$) determines $j = \ell^*$ in [Algorithm 2](#), meaning that $L_{i_{\ell^*}} \notin_{\text{id}} \mathcal{A}(S)$ at that time. Namely, each invocation $\text{AdvEnum}(\text{chain}_n, S)$ where $|\text{chain}_n| = \ell^*$ can be associated to a *witness* of a time step where the list output by the algorithm did not identify $L_{i_{\ell^*}}$.

Finally, in each invocation where $|\text{chain}_n| = \ell^*$, the while loop resumes enumerating $L_{i_{\ell^*}}$; in particular, it is adding at least one string in the enumeration of $L_{i_{\ell^*}}$ that hasn't yet appeared in the input to S . This means that over the infinite execution of [Algorithm 2](#), $L_{i_{\ell^*}}$ gets completely enumerated. Thus, we have argued the existence of a sequence that is a valid enumeration of some language $L_{i_{\ell^*}} \in \mathcal{C}$, for which there are infinitely many time steps where the list output by \mathcal{A} does not identify $L_{i_{\ell^*}}$. Thus, \mathcal{A} is not a valid list-identifier. \square

Remark 3. In light of [Remark 2](#), we thus have that, for any $k \geq 1$, the collection $\mathcal{C}_k = \{\mathbb{Z}\} \cup \{\mathbb{Z} \setminus F : F \subseteq \mathbb{Z}, |F| \leq k\}$ is not k -list identifiable, but is $(k+1)$ -list identifiable in the limit. Furthermore, the collection $\mathcal{C}_\infty = \{\mathbb{Z}\} \cup \{\mathbb{Z} \setminus F : F \subseteq \mathbb{Z}, |F| < \infty\}$ is not k -list identifiable for *any* finite value of k . We highlight that this is an example of a simple countable collection of languages, which, by the general positive result of [\[KM24\]](#) for countable collections, is *generatable* in the limit, but is not identifiable in the limit, even with a finite list of guesses.

7 Stratification of k -list Identifiable Collections

In this section, we show that k -list identifiable collections admit additional structure: they can be decomposed into k collections, each of which is *identifiable* in the limit in the standard sense!

Theorem 2 (Stratification of k -list Identifiable Collections). *Let $\mathcal{C} = (L_1, L_2, \dots)$ be a countable language collection. Then, \mathcal{C} is k -list identifiable in the limit if and only if $\mathcal{C} = \cup_{i=1}^k \mathcal{C}_i$ where \mathcal{C}_i is identifiable in the limit for every $i \in \{1, 2, \dots, k\}$.*

Proof. One direction is straightforward: if $\mathcal{C} = \cup_{i=1}^k \mathcal{C}_i$ where \mathcal{C}_i is identifiable in the limit for every $i \in \{1, 2, \dots, k\}$, then consider $\mathcal{A}_1, \dots, \mathcal{A}_k$ that identify $\mathcal{C}_1, \dots, \mathcal{C}_k$ respectively. We can verify that \mathcal{A} , which assigns $\mathcal{A}(x_1, \dots, x_t) := (\mathcal{A}_1(x_1, \dots, x_t), \dots, \mathcal{A}_k(x_1, \dots, x_t))$ successfully k -list identifies \mathcal{C} in the limit.

For the other direction, let \mathcal{C} be a k -list identifiable collection. By [Theorem 7](#), this must mean that \mathcal{C} satisfies the k -Angluin condition. Namely, we have that $\forall i \in \mathbb{N} : \Psi(L_i, k)$. We will construct $\mathcal{C}_k, \mathcal{C}_{k-1}, \dots, \mathcal{C}_1$ satisfying $\cup_{i=1}^k \mathcal{C}_i$ in levels, using the inductive nature of the condition.

Let $I_k = \mathbb{N}$, and consider the relation $R_k \subseteq I_k \times I_k$ constructed as follows:

$$R_k := \{(i, j) : i, j \in I_k, L_j \subsetneq L_i \text{ and } L_j \supseteq T_i^{(k)}\}. \quad (8)$$

Then, let $J_k = \{j \in I_k : (i, j) \notin R_k \ \forall i \in I_k\}$, and let $\mathcal{C}_k = \{L_j : j \in J_k\}$. Note that for any two languages L_a, L_b in \mathcal{C}_k , if $L_b \subsetneq L_a$, then it must be the case that $L_b \not\supseteq T_a^{(k)}$; otherwise, $(a, b) \in R_k$ and L_b would not have been included in \mathcal{C}_k . In other words, for every language $L_a \in \mathcal{C}_k$, $T_a^{(k)}$ serves as a tell-tale set for L_a within \mathcal{C}_k , meaning that \mathcal{C}_k satisfies Angluin's condition, and is identifiable in the limit.

Now let $I_{k-1} = I_k \setminus J_k$. Observe that for any $j \in I_{k-1}$, since L_j was not included in \mathcal{C}_k , it must be the case that there exists $i \in I_k$ for which $L_j \subsetneq L_i$ and $L_j \supseteq T_i^{(k)}$. But since $\Psi(L_i, k)$ holds for every $i \in I_k$, this must mean that $\Psi(L_j, k-1)$ holds. Thus, we have argued that $\forall j \in I_{k-1}$, $\Psi(L_j, k-1)$ holds. Furthermore, as detailed in [Remark 1](#), whenever we consider the predicate $\Psi(L_j, k-1)$ to hold, we can assume it to hold with a unique $T_j^{(k-1)}$. So, let us proceed with constructing the relation $R_{k-1} \subseteq I_{k-1} \times I_{k-1}$ along the same lines as above:

$$R_{k-1} := \{(i, j) : i, j \in I_{k-1}, L_j \subsetneq L_i \text{ and } L_j \supseteq T_i^{(k-1)}\}. \quad (9)$$

Then let $J_{k-1} = \{j \in I_{k-1} : (i, j) \notin R_{k-1} \ \forall i \in I_{k-1}\}$, and let $\mathcal{C}_{k-1} = \{L_j : j \in J_{k-1}\}$. Again, by the same reasoning as in the above paragraph, we have that for every language $L_a \in \mathcal{C}_{k-1}$, the set $T_a^{(k-1)}$ serves as a tell-tale for L_a within \mathcal{C}_{k-1} , and thus, \mathcal{C}_{k-1} is identifiable in the limit.

We proceed similarly, and construct the collections $\mathcal{C}_{k-2}, \dots, \mathcal{C}_1$. By the reasoning above, each of these collections is identifiable in the limit. We will now argue that their union is \mathcal{C} . For this, we need to show that $J_k \cup J_{k-1} \cup \dots \cup J_1 = \mathbb{N}$.

Observe that $J_k \cup J_{k-1} \cup \dots \cup J_1 = \mathbb{N}$. We will argue that $J_1 = I_1$, by arguing that the set R_1 is empty. Indeed, if R_1 contains any (i, j) for $i, j \in I_1$, then by definition, it must be the case that $L_j \subsetneq L_i$ and $L_j \supseteq T_i^1$. But recall that we maintain that for every $i \in I_1$, $\Psi(L_i, 1)$ holds. Thus, recalling the definition of the base predicate (4), the existence of $L_j \subsetneq L_i$ satisfying $L_j \supseteq T_i^{(1)}$ is a contradiction. We conclude that $J_1 = I_1$, which means that $J_k \cup J_{k-1} \cup \dots \cup J_1 = \mathbb{N}$. Thus, we have that $\cup_{i=1}^k \mathcal{C}_i = \mathcal{C}$, which completes the proof. \square

Remark 4. [Theorem 2](#) also implies an alternative k -list identifier to the one described in [Section 5](#). Namely, if a collection is k -list identifiable in the limit, then it can be decomposed into k collections each of which is identifiable in the limit. So, concatenating the outputs of identifiers run on each of these collections also successfully k -list identifies the collection in the limit.

8 List Identification Rates

We now proceed towards establishing rates for list identification in the setting where the input is drawn as an i.i.d. stream from a valid distribution supported on some language in the collection. The three subsequent sections respectively show that: (1) If a collection does not satisfy the k -Angluin condition, then it cannot be k -list identified at any rate, (2) A collection that satisfies the k -Angluin condition can be k -list identified at an exponential rate, and (3) A rate faster than an exponential rate is impossible for any (non-trivial) collection.

8.1 Condition Not Satisfied \implies No Rate

The main result in this section is the following:

Theorem 8 (k -Angluin Condition Not Satisfied \implies No Rate). *Let \mathcal{C} be a countable collection of languages that does not satisfy the k -Angluin condition (6). Then \mathcal{C} cannot be k -list identified at any rate.*

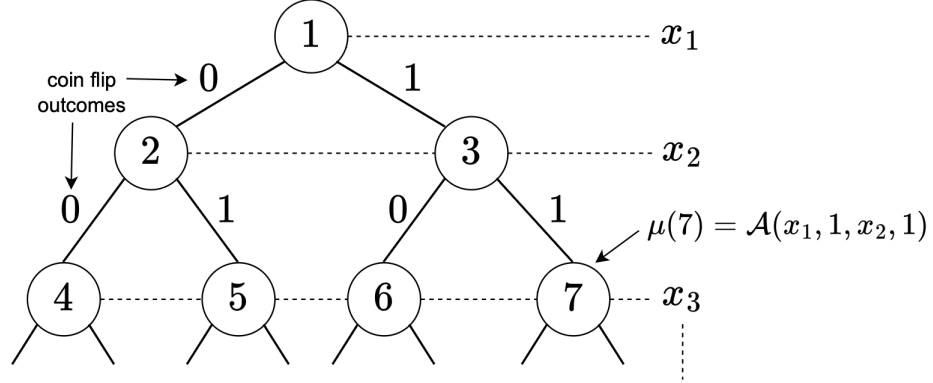


Figure 1: The tree of computation $\mathcal{T}_{\mathcal{A},\sigma}$ of a probabilistic k -list identifier on an input sequence $\sigma = (x_1, x_2, \dots)$. The labels on the edges represent coin flip outcomes. The root is at level 1. Every node is associated with the list guessed by \mathcal{A} at that node.

8.1.1 List-identifiable Collections \equiv Probabilistically List-identifiable Collections

Our discussion on k -list identification so far has been about list identifiers that are *deterministic*, in that the list output at every time step by the identifier is a deterministic function of the input up until that time step. Here, we will show that the family of collections that are deterministically k -list identifiable in the limit is precisely the family of collections that are *probabilistically* k -list identifiable in the limit. In probabilistic k -list identification, we allow the list output by the identifier at every time step to be randomized. Our discussion here is based on the work of [Pit85] on probabilistic inductive inference, although we are required to carefully derive some results differently for the purposes of list identification.

Definition 5 (Probabilistic List Identifier). A *probabilistic list identifier* with list size k , or a *probabilistic k -list identifier*, is a function $\mathcal{A} : (U \times \{0,1\})^* \rightarrow \mathbb{N}^k$, which at any time step t , takes as input a finite ordered sequence of strings x_1, \dots, x_t and the outcomes of t independent and uniformly random bits r_1, \dots, r_t , and outputs a list of indices $\mu_t = \mathcal{A}(x_1, r_1, \dots, x_t, r_t) = (i_1, \dots, i_k)$.

Definition 6 (Probabilistic List Identification in the Limit). A *probabilistic k -list identifier* \mathcal{A} identifies a countable collection $\mathcal{C} = \{L_1, L_2, \dots\}$ in the limit with probability p if for every language $L_z \in \mathcal{C}$, and for every enumeration x_1, x_2, \dots of L_z , with probability at least p over a random bit sequence r_1, r_2, \dots , there exists a finite time t^* such that for every $t \geq t^*$, the list $\mu_t = \mathcal{A}(x_1, r_1, \dots, x_t, r_t)$ output by \mathcal{A} satisfies that $L_z \in_{\text{id}} \mu_t$.

For any given input enumeration σ , we model the infinite sequence of guesses made by a probabilistic k -list identifier \mathcal{A} on σ as a path on a binary tree, that begins at the root, and descends down infinitely, where at a node at level t , the path proceeds towards the left child if the random bit $r_t = 0$, and towards the right child otherwise.

Definition 7 (Tree of Computation). Given a probabilistic k -list identifier \mathcal{A} and an input enumeration $\sigma = (x_1, x_2, \dots)$, we define the object $\mathcal{T}_{\mathcal{A},\sigma}$ to be the infinite binary tree, whose nodes are numbered $1, 2, 3, \dots$ starting from the root, and proceeding in a breadth-first order from left to right. A node is connected to its left child with an edge labeled 0, and to its right child with an edge labeled 1. A path in the tree is an infinite sequence of adjacent nodes (n_1, n_2, \dots) starting from the root (i.e., $n_1 = 1$), and proceeding downwards at every step. The level of a node n , denoted $d(n)$, is the number of edges on the path from the root to n , plus one (so the root is at level 1). We associate

level i of the tree with input x_i in the enumeration σ . We associate every node n_i at level i in a path, for $i > 1$, with the list $\mu(n_i) := \mathcal{A}(x_1, r_1, \dots, x_{i-1}, r_{i-1})$, where r_1, \dots, r_{i-1} are the edge labels on the path from the root node to n_i . Refer to [Figure 1](#) for an illustration.

We will now define what it means for a path in the tree of computation $\mathcal{T}_{\mathcal{A},\sigma}$ to *converge*.

Definition 8 (Convergence in Tree). *A path (n_1, n_2, \dots) in $\mathcal{T}_{\mathcal{A},\sigma}$ converges to a language $L_z \in \mathcal{C}$ if there exists a finite i such that for every $j \geq i$, $L_z \in_{\text{id}} \mu(n_j)$. Furthermore, we say that a path (n_1, n_2, \dots) converges to language $L_z \in \mathcal{C}$ at node n_i , for $i \geq 2$, if: (1) for every $j \geq i$, $L_z \in_{\text{id}} \mu(n_j)$, and (2) either $i = 2$, or property (1) does not hold for any $j < i$.*

In other words, a path (n_1, n_2, \dots) converges to L_z at node n_i , if n_i corresponds to the smallest i , such that the path converges to L_z at node n_i . Note that because of this, any path (n_1, n_2, \dots) that converges to L_z converges to it at a *unique* node. We denote by $P(L_z)$ the set of paths in $\mathcal{T}_{\mathcal{A},\sigma}$ that converge to language $L_z \in \mathcal{C}$. Similarly, for any node $n \in \mathcal{T}_{\mathcal{A},\sigma}$, we denote by $P(L_z, n)$ the set of paths in $\mathcal{T}_{\mathcal{A},\sigma}$ that pass through n and converge to language $L_z \in \mathcal{C}$ at node n . We can then observe that

$$P(L_z) = \cup_{n \in \mathcal{T}_{\mathcal{A},\sigma}} P(L_z, n), \quad (10)$$

and by our previous reasoning, the sets $P(L_z, n)$ participating in the union above are all mutually disjoint.

Now consider the random experiment of tracing down a path (n_1, n_2, \dots) in $\mathcal{T}_{\mathcal{A},\sigma}$ with respect to a randomly drawn sequence of bits $R := (r_1, r_2, \dots)$, where from any node n_i , we descend down the edge labeled 0 if $r_i = 0$, and down the edge labeled 1 otherwise. Note that this defines a bijection between paths in $\mathcal{T}_{\mathcal{A},\sigma}$ and sequences of random bits. In this case, if σ is an enumeration of some language $L_z \in \mathcal{C}$, then the set $P(L_z)$ defines the event that \mathcal{A} list-identifies L_z in the limit.⁷ Concretely, if \mathcal{A} k -list identifies \mathcal{C} with probability p , then we have that

$$\Pr_R[P(L_z)] = \sum_{n \in \mathcal{T}_{\mathcal{A},\sigma}} \Pr_R[P(L_z, n)] \geq p. \quad (11)$$

We will require considering truncated versions of the infinite tree $\mathcal{T}_{\mathcal{A},\sigma}$ in order to validly define a deterministic list identifier from a probabilistic list identifier. Accordingly, we require the following definition:

Definition 9. *For any input enumeration σ , consider the infinite tree $\mathcal{T}_{\mathcal{A},\sigma}$. For any node $n \in \mathcal{T}_{\mathcal{A},\sigma}$ at level $d(n) \geq 2$, and for any $d \geq d(n)$, we define $P(L_z, n, d)$, for any $L_z \in \mathcal{C}$, to be the set of paths (n_1, n_2, \dots) in $\mathcal{T}_{\mathcal{A},\sigma}$, which satisfy:*

1. *The path passes through node n , i.e., $n_i = n$ for $i = d(n)$.*
2. *Either $d(n) = 2$, or $\mu(n_{d(n)-1}) \not\in_{\text{id}} L_z$.*
3. *Every list guessed from node n onward up until level d in the path identifies L_z , i.e., for every j where $d(n) \leq j \leq d$, $L_z \in_{\text{id}} \mu(n_j)$.*

In other words, $P(L_z, n, d)$ comprises of paths that appear to have converged to L_z at node n , if one myopically looks beyond node n only until level d . We can observe that for any $d \geq d(n)$,

$$P(L_z, n) \subseteq P(L_z, n, d). \quad (12)$$

⁷The probability space is indeed well-behaved; for measure-theoretic considerations, we refer the reader to [\[Pit85\]](#).

More consequentially, observe that we can easily compute $\Pr_R[P(L_z, n, d)]$ exactly. For this, we first check if either $d(n) = 2$ or $L_z \notin \mu(n_{d(n)-1})$. If this is not the case, then $\Pr_R[P(L_z, n, d)] = 0$. Otherwise, we consider all the descendants of node n at level d —there are precisely $2^{d-d(n)}$ many of these. From each descendant n_d , we trace up a path to the root—say $(n_d, n_{d-1}, \dots, n_{d(n)}, \dots, n_2, n_1)$, and check whether $L_z \in \mu(n_i)$ for every $i \geq d(n)$. We count the number of descendants $N_{z,n,d}$ that pass this check. Then, we can see that $P(L_z, n, d)$ is precisely the event that a random path passes through one of these descendants, giving us that $\Pr_R[P(L_z, n, d)] = \frac{N_{z,n,d}}{2^{d-1}}$. This, together with the observation that $P(L_z, n, d)$ and $P(L_z, n', d)$ are disjoint for any $n \neq n'$, imply that

$$\sum_{\substack{n \in \mathcal{T}_{A,\sigma} \\ d(n) \leq d}} \Pr_R[P(L_z, n, d)] = \frac{|\{n \in \mathcal{T}_{A,\sigma} : d(n) = d, L_z \in \mu(n)\}|}{2^{d-1}}, \quad (13)$$

where the RHS above is precisely the fraction of nodes at level d in $\mathcal{T}_{A,\sigma}$ that identify L_z .

We are now ready to state the main theorem of this subsection which equates deterministic and probabilistic list identification.

Theorem 9 (Probabilistic List Identification \equiv Deterministic List Identification). *Let $\mathcal{C} = \{L_1, L_2, \dots\}$ be a countable collection of languages.*

- (1) *If \mathcal{C} is deterministically k -list identifiable in the limit, then \mathcal{C} is probabilistically k -list identifiable in the limit with probability p , for any $p \in [0, 1]$.*
- (2) *If \mathcal{C} is probabilistically k -list identifiable in the limit with probability $p > \frac{k}{k+1}$, then \mathcal{C} is deterministically k -list identifiable in the limit.*

Proof. The proof of (1) is immediate: if there exists a deterministic k -list identifier \mathcal{A}' that identifies \mathcal{C} in the limit, then the probabilistic k -list identifier \mathcal{A} , which on input $\sigma = (x_1, x_2, \dots)$, disregards any random bits, and simply outputs $\mathcal{A}(x_1, r_1, \dots, x_t, r_t) = \mathcal{A}'(x_1, \dots, x_t)$, probabilistically k -list identifies \mathcal{C} in the limit with probability 1.

Given a probabilistic k -list identifier \mathcal{A} for \mathcal{C} that succeeds with probability larger than $\frac{k}{k+1}$, we will now construct a deterministic k -list identifier \mathcal{A}' for \mathcal{C} . We will specify \mathcal{A}' by specifying its outputs on any given input enumeration $\sigma := (x_1, x_2, \dots)$. For any index $l \in \mathbb{N}$, let us define $\text{first}_{\mathcal{C}}(l) := \min\{i \in \mathbb{N} : L_i = L_l\}$. That is, $\text{first}_{\mathcal{C}}(l)$ is the index of the first occurrence of language L_l in the collection \mathcal{C} .

Deterministic List Identifier \mathcal{A}' from Probabilistic List Identifier \mathcal{A}

At time step t on input $\sigma := (x_1, x_2, \dots)$:

- (a) Let μ_1, \dots, μ_{2^t} denote the k -lists output at all the nodes at level $t+1$ in $\mathcal{T}_{A,\sigma}$.
- (b) Let S be the *multiset* of $\text{first}_{\mathcal{C}}(l)$ values for every index l in these lists. That is,

$$S := \text{multiset}\{\text{first}_{\mathcal{C}}(l) : l \in \mu_i, 1 \leq i \leq 2^t\}.$$

- (c) Return $\mathcal{A}'(x_1, \dots, x_t) := \text{Top-}k(S)$, where $\text{Top-}k(S)$ returns a list of the k most frequently occurring indices in S .

Procedure 2: Obtaining a deterministic list identifier from a probabilistic list identifier.

Let \mathcal{A} be a probabilistic k -list identifier that identifies \mathcal{C} in the limit with probability $p > \frac{k}{k+1}$; in particular, let $p = \frac{k}{k+1} + \varepsilon$ for some $\varepsilon > 0$. Fix any language $L_z \in \mathcal{C}$, and any enumeration σ of it. From (11), we have that

$$\sum_{n \in \mathcal{T}_{\mathcal{A}, \sigma}} \Pr_R[P(L_z, n)] \geq \frac{k}{k+1} + \varepsilon.$$

Because all the summands in the summation above are non-negative, there exists a finite $d^* \in \mathbb{N}$ such that for all $d \geq d^*$,

$$\sum_{\substack{n \in \mathcal{T}_{\mathcal{A}, \sigma} \\ d(n) \leq d}} \Pr_R[P(L_z, n)] \geq \frac{k}{k+1} + \frac{\varepsilon}{2}.$$

From (12), this implies that for all $d \geq d^*$,

$$\sum_{\substack{n \in \mathcal{T}_{\mathcal{A}, \sigma} \\ d(n) \leq d}} \Pr_R[P(L_z, n, d)] \geq \frac{k}{k+1} + \frac{\varepsilon}{2}. \quad (14)$$

From (13), this implies that for all $d \geq d^*$, the fraction of nodes in $\mathcal{T}_{\mathcal{A}, \sigma}$ at level d that identify L_z is at least $\frac{k}{k+1} + \frac{\varepsilon}{2}$.

So, consider any time step $t \geq d^* - 1$. By the reasoning above, we have that of the 2^t lists μ_1, \dots, μ_{2^t} considered by the deterministic list identifier \mathcal{A}' in Step (a) of [Procedure 2](#), at least a $\frac{k}{k+1} + \frac{\varepsilon}{2}$ fraction identify L_z . This means that the multiset S constructed in Step (b) contains at least $2^t \cdot \left(\frac{k}{k+1} + \frac{\varepsilon}{2}\right)$ copies of $\text{first}_{\mathcal{C}}(z)$. We then claim that $\text{Top-}k(S)$ necessarily contains $\text{first}_{\mathcal{C}}(z)$. If not, there are k distinct indices other than $\text{first}_{\mathcal{C}}(z)$, each of which occur at least $2^t \cdot \left(\frac{k}{k+1} + \frac{\varepsilon}{2}\right)$ times in S . Since S also contains $2^t \cdot \left(\frac{k}{k+1} + \frac{\varepsilon}{2}\right)$ copies of $\text{first}_{\mathcal{C}}(z)$, this would mean that there are a total of at least $2^t \cdot \left(k + \frac{\varepsilon(k+1)}{2}\right)$ numbers in S , which is not possible since S is a concatenation of at most 2^t lists of size $\leq k$. Thus, we have argued that $\text{Top-}k(S) = \mathcal{A}'(x_1, \dots, x_t)$ contains $\text{first}_{\mathcal{C}}(z)$ for every $t \geq d^* - 1$, implying that \mathcal{A}' (deterministically) list-identifies L_z in the limit on the input enumeration σ . Since the language L_z and its enumeration σ were arbitrarily chosen, we conclude that \mathcal{A}' list-identifies \mathcal{C} in the limit. \square

8.1.2 Probabilistic List Identification \equiv List Identification on Infinite Draws

In both the definitions [Definition 2](#) and [Definition 6](#), we considered the enumeration $\sigma = (x_i)_{i \in \mathbb{N}}$ of the target language L_z to be any fixed worst-case enumeration. In this subsection, we consider the setting where the enumeration $(x_i)_{i \in \mathbb{N}}$ is itself an infinite i.i.d. sequence drawn from a distribution \mathcal{D} that is valid for the language L_z (and the identifier is deterministic). Since \mathcal{D} is valid for L_z , meaning that its support is precisely L_z , every member of L_z will eventually show up in an infinite draw from \mathcal{D} , and there are no spurious strings in it. More formally, Proposition 5.2 in [\[KMV25\]](#) shows that such an infinite draw is an enumeration of L_z with probability 1.

Definition 10 (List Identification in the Limit on Infinite Draws). *A deterministic k -list identifier \mathcal{A}' identifies a countable collection $\mathcal{C} = \{L_1, L_2, \dots\}$ in the limit with probability p on infinite draws if for every language $L_z \in \mathcal{C}$, and for every distribution \mathcal{D} that is valid for L_z , with probability at least p over $(x_i)_{i \in \mathbb{N}} \sim \mathcal{D}^\infty$, there exists a finite time t^* such that for every $t \geq t^*$, the list $\mu_t = \mathcal{A}'(x_1, \dots, x_t)$ output by \mathcal{A}' satisfies that $L_z \in_{\text{id}} \mu_t$*

The main theorem of this subsection equates list identification in the limit on infinite draws and probabilistic list identification in the limit. The proof of this theorem follows the proof structure of Theorem 9 in [Ang88].

Theorem 10 (Probabilistic List Identification \equiv List Identification on Infinite Draws). *Let $\mathcal{C} = \{L_1, L_2, \dots\}$ be a countable collection of languages, and fix $p \in [0, 1]$. Then, \mathcal{C} is k -list identifiable in the limit with probability p on infinite draws if and only if \mathcal{C} is probabilistically k -list identifiable in the limit with probability p .*

Proof. First, let \mathcal{A}' be a deterministic k -list identifier that identifies \mathcal{C} in the limit with probability p on infinite draws. We will construct a probabilistic k -list identifier \mathcal{A} for \mathcal{C} that succeeds with probability p . Fix any language L_z and any enumeration $\sigma := (x_i)_{i \in \mathbb{N}}$ of it. Define a distribution \mathcal{D}_σ supported over L_z as follows:

$$\mathcal{D}_\sigma(x) = \begin{cases} \sum_{i \in \mathbb{N}: x_i = x} \frac{1}{2^i} & \text{if } x \in L_z, \\ 0 & \text{otherwise.} \end{cases}$$

Since σ is a valid enumeration of L_z , we can verify that \mathcal{D}_σ is a valid distribution for L_z . Then, consider the probabilistic k -list identifier \mathcal{A} which, given σ and an infinite random bit sequence $R := (r_i)_{i \in \mathbb{N}}$ as input, operates as follows. Using R , \mathcal{A} constructs an infinite draw $\sigma' := (x'_i)_{i \in \mathbb{N}} \sim \mathcal{D}_\sigma$ for \mathcal{D}_σ defined above.⁸ It then feeds σ' to \mathcal{A}' , and at each time step, outputs the list output by \mathcal{A}' . Since \mathcal{D}_σ is a valid distribution for L_z , we have that with probability at least p over σ' , \mathcal{A}' will converge to outputting a list that identifies L_z . Since the draw of σ' is induced by the draw of R , and \mathcal{A} returns the output of \mathcal{A}' , we conclude that with probability at least p over R , \mathcal{A} converges to outputting a list that identifies L_z on the enumeration σ .

Now, let \mathcal{A} be a probabilistic k -list identifier that identifies \mathcal{C} in the limit with probability p . We will construct a deterministic k -list identifier \mathcal{A}' that identifies \mathcal{C} in the limit with probability p on infinite draws. Fix any language $L_z \in \mathcal{C}$, and any distribution \mathcal{D} that is valid for L_z . Consider \mathcal{A}' , which on input $(x_i)_{i \in \mathbb{N}} \sim \mathcal{D}^\infty$, operates as follows. Suppose $x_1 = a$: \mathcal{A}' scans increasing prefixes (x_1, \dots, x_t) of the input up until the first time it sees some $x_t \neq a$. Up until this time, \mathcal{A}' keeps outputting the singleton list $\{L_j\}$, where j is the index of any language in \mathcal{C} for which $L_j = \{a\}$; if no such language exists \mathcal{A}' keeps outputting $\{L_1\}$ arbitrarily until it arrives at x_t . Observe that if L_z were a singleton set, then the only valid distribution for L_z is a point mass on the single element in L_z , and \mathcal{A}' identifies L_z with probability 1 from the first time step.

Otherwise, \mathcal{A}' reaches an $x_t = b \neq a$. Note that this can only happen if L_z has at least two elements. Furthermore, in this case, since \mathcal{D} is supported on all of L_z , such an x_t will be encountered with probability 1. \mathcal{A}' then feeds $(x_{2i-1})_{i \in \mathbb{N}}$ to \mathcal{A} as input. Since $(x_{2i-1})_{i \in \mathbb{N}}$ is also an infinite draw from \mathcal{D} , it is a valid enumeration of L_z with probability 1. \mathcal{A}' constructs a random bit sequence $(r_i)_{i \in \mathbb{N}}$ for \mathcal{A} , using the randomness in the input, in the following manner: denote $t = j_1$, where t was the first time where $x_t = b$. In order to determine r_1 , \mathcal{A}' scans pairs $(x_{2(j_1+1)}, x_{2(j_1+2)}), (x_{2(j_1+3)}, x_{2(j_1+4)}), \dots$ up until the time it sees a pair which is either (a, b) or (b, a) —say this happens at $(x_{2(j_1+m)}, x_{2(j_1+m+1)})$. This process also terminates with probability 1, and once it terminates, r_1 is set to 1 if the process terminates with the pair (a, b) and 0 if it terminates with (b, a) . Thereafter, r_2 is determined similarly by setting $j_2 = j_1 + m + 1$, and starting to scan pairs $(x_{2(j_2+1)}, x_{2(j_2+2)}), (x_{2(j_2+3)}, x_{2(j_2+4)}), \dots$ and so on.

⁸For example, one way to do this is as follows: to generate $x' \sim \mathcal{D}_\sigma$, \mathcal{A} scans (the rest of) R ahead and finds the position j of the first 1 it encounters (here, we imagine the first bit in the rest of R to be at position 1). \mathcal{A} then sets x' to be x_j in σ . We can verify that over the randomness of R , x' generated thus is distributed according to \mathcal{D}_σ .

Since $(x_i)_{i \in \mathbb{N}}$ is an infinite i.i.d. draw from \mathcal{D} , conditioned on any $(x_{2i-1})_{i \in \mathbb{N}}$, the sequence $(x_{2i})_{i \in \mathbb{N}}$ is also an infinite i.i.d. draw from \mathcal{D} . So, consider the distribution of $(r_i)_{i \in \mathbb{N}}$ induced by the randomness in $(x_{2i})_{i \in \mathbb{N}} \sim \mathcal{D}^\infty$. Since each r_i is determined using independent strings from \mathcal{D} , the bits are independent. Furthermore, conditioned on the first two distinct strings in $(x_{2i})_{i \in \mathbb{N}}$ being any (a, b) , the next pair in the sequence that is one of (a, b) or (b, a) is equally likely to be either. So, every r_i is either 0 or 1 with probability $1/2$ each. Thus, we have argued that the distribution of $(r_i)_{i \in \mathbb{N}}$ induced by $(x_{2i})_{i \in \mathbb{N}} \sim \mathcal{D}^\infty$ is precisely that of an infinite sequence of independent random bits.

So, \mathcal{A}' feeds $(x_{2i-1})_{i \in \mathbb{N}}$ to the probabilistic k -list identifier \mathcal{A} as input, along with the bit sequence $(r_i)_{i \in \mathbb{N}}$ that it constructs from $(x_{2i})_{i \in \mathbb{N}}$. At each time step, it outputs precisely the list that is output by \mathcal{A} . Observe that the outputs of \mathcal{A}' are deterministic. It remains to argue that \mathcal{A}' thus constructed identifies L_z with probability at least p over the draw of $(x_i)_{i \in \mathbb{N}}$. To see this, note that

$$\begin{aligned} & \Pr_{(x_i)_{i \in \mathbb{N}} \sim \mathcal{D}^\infty} [\mathcal{A}' \text{ list-identifies } L_z \text{ in the limit on input } (x_i)_{i \in \mathbb{N}}] \\ &= \mathbb{E}_{(x_{2i-1})_{i \in \mathbb{N}} \sim \mathcal{D}^\infty} \left[\Pr_{(x_{2i})_{i \in \mathbb{N}} \sim \mathcal{D}^\infty} \left[\mathcal{A}' \text{ list-identifies } L_z \text{ in the limit on input } (x_i)_{i \in \mathbb{N}} \mid (x_{2i-1})_{i \in \mathbb{N}} \right] \right] \\ &= \mathbb{E}_{(x_{2i-1})_{i \in \mathbb{N}} \sim \mathcal{D}^\infty} \left[\Pr_{(r_i)_{i \in \mathbb{N}}} \left[\mathcal{A} \text{ list-identifies } L_z \text{ in the limit on input } (x_{2i-1})_{i \in \mathbb{N}} \mid (x_{2i-1})_{i \in \mathbb{N}} \right] \right] \\ &\geq \mathbb{E}_{(x_{2i-1})_{i \in \mathbb{N}} \sim \mathcal{D}^\infty} [p] = p. \end{aligned}$$

The inequality above follows from our above reasoning, which establishes that the induced distribution of $(r_i)_{i \in \mathbb{N}}$ conditioned on any $(x_{2i-1})_{i \in \mathbb{N}}$ is that of an infinite sequence of independent random bits, together with the guarantee that \mathcal{A} , by virtue of being a probabilistic k -list identifier, identifies L_z in the limit with probability at least p over the randomness in $(r_i)_{i \in \mathbb{N}}$ on the (fixed) enumeration $(x_{2i-1})_{i \in \mathbb{N}}$ of L_z . \square

8.1.3 Condition Not Satisfied \implies No Rate

As a corollary of [Theorems 7, 9](#) and [10](#), we have the following:

Corollary 8.1. *Let \mathcal{C} be a countable collection of languages that does not satisfy the k -Angluin condition [\(6\)](#). Then, for every (deterministic) k -list identifier \mathcal{A} , there exists a language $L_z \in \mathcal{C}$ and a distribution \mathcal{D} that is valid for L_z , such that \mathcal{A} fails to k -list identify L_z in the limit on $(x_i)_{i \in \mathbb{N}} \sim \mathcal{D}^\infty$ with probability at least $\frac{1}{k+1}$. In other words,*

$$\Pr_{(x_i)_{i \in \mathbb{N}} \sim \mathcal{D}^\infty} [\exists i_1 < i_2 < i_3 < \dots : \forall j \in \mathbb{N}, \mathcal{A}(x_1, x_2, \dots, x_{i_j}) \not\equiv_{\text{id}} L_z] \geq \frac{1}{k+1}.$$

Similar to [\[KMV25\]](#), we now have all the ingredients to establish [Theorem 8](#), which shows that any collection that does not satisfy the k -Angluin condition [\(6\)](#) cannot be k -list identified at any rate. More precisely, we will show that:

Theorem 11. *Let \mathcal{C} be a countable collection of languages that does not satisfy the k -Angluin condition [\(6\)](#). Then, for every (deterministic) k -list identifier \mathcal{A} , there exists a language $L_z \in \mathcal{C}$ and a distribution \mathcal{D} that is valid for L_z , such that*

$$\limsup_{t \rightarrow \infty} \Pr_{x_1, \dots, x_t \sim \mathcal{D}^t} [\mathcal{A}(x_1, \dots, x_t) \not\equiv_{\text{id}} L_z] \geq \frac{1}{k+1}. \quad (15)$$

Proof. Assume for the sake of contradiction that there exists a k -list identifier \mathcal{A} which satisfies that for every language $L_z \in \mathcal{C}$, and for every distribution \mathcal{D} that is valid for L_z ,

$$\limsup_{t \rightarrow \infty} \Pr_{x_1, \dots, x_t \sim \mathcal{D}^t} [\mathcal{A}(x_1, \dots, x_t) \not\equiv_{\text{id}} L_z] \leq \frac{1}{k+1} - \varepsilon \quad (16)$$

for some $\varepsilon > 0$. This means that for every $L_z \in \mathcal{C}$, and every distribution \mathcal{D} that is valid for L_z , there exists a finite $t_0 = t_0(L_z, \mathcal{D})$, such that for every $t \geq t_0$,

$$\Pr_{x_1, \dots, x_t \sim \mathcal{D}^t} [\mathcal{A}(x_1, \dots, x_t) \not\equiv_{\text{id}} L_z] \leq \frac{1}{k+1} - \frac{\varepsilon}{2}. \quad (17)$$

Otherwise, if there were infinitely many t for which $\Pr_{x_1, \dots, x_t \sim \mathcal{D}^t} [\mathcal{A}(x_1, \dots, x_t) \not\equiv_{\text{id}} L_z] > \frac{1}{k+1} - \frac{\varepsilon}{2}$, then we would have $\limsup_{t \rightarrow \infty} \Pr_{x_1, \dots, x_t \sim \mathcal{D}^t} [\mathcal{A}(x_1, \dots, x_t) \not\equiv_{\text{id}} L_z] \geq \frac{1}{k+1} - \frac{\varepsilon}{2}$, which would contradict (16).

So, with a view to contradict [Corollary 8.1](#), let us fix an arbitrary $L_z \in \mathcal{C}$, and an arbitrary \mathcal{D} that is valid for L_z . We will use (17) to construct the following k -list identifier \mathcal{A}' : given $x_1, \dots, x_t \sim \mathcal{D}^t$, \mathcal{A}' splits it into $M := \frac{t}{\log t}$ batches B_1, \dots, B_M , each of size $\log t$. It then runs \mathcal{A} on each batch to obtain the lists $\mathcal{A}(B_1), \dots, \mathcal{A}(B_M)$. Then, it constructs S to be the multiset of $\text{first}_{\mathcal{C}}(l)$ values for every index l in these lists, i.e.,

$$S := \text{multiset} \{ \text{first}_{\mathcal{C}}(l) : l \in \mathcal{A}(B_j), 1 \leq j \leq M \}.$$

Finally, \mathcal{A}' outputs $\text{Top-}k(S)$. These latter steps are similar to Steps (b) and (c) in [Figure 2](#). In essence, we are boosting the guarantee in (17) by combining the predictions of \mathcal{A} on independent batches of increasing size.

Observe then that when $t \geq e^{t_0}$ for $t_0 = t_0(L_z, \mathcal{D})$ satisfying (17), we will have that every batch B_j is of size at least t_0 , which implies that

$$\Pr_{B_j} [\mathcal{A}(B_j) \equiv_{\text{id}} L_z] \geq \frac{k}{k+1} + \frac{\varepsilon}{2}.$$

Fix any $t \geq e^{t_0}$, and let $Y_j = \mathbb{1}[\mathcal{A}(B_j) \equiv_{\text{id}} L_z]$. Since the batches are independent, we have that the Y_j s are independent, and also, $\mathbb{E} \left[\sum_{j=1}^M Y_j \right] \geq M \left(\frac{k}{k+1} + \frac{\varepsilon}{2} \right)$. Then, we have that

$$\begin{aligned} \Pr_{x_1, \dots, x_t \sim \mathcal{D}^t} \left[\sum_{j=1}^M Y_j \leq M \left(\frac{k}{k+1} + \frac{\varepsilon}{4} \right) \right] &\leq \Pr_{x_1, \dots, x_t \sim \mathcal{D}^t} \left[\sum_{j=1}^M Y_j \leq \mathbb{E} \left[\sum_{j=1}^M Y_j \right] - \frac{M\varepsilon}{4} \right] \\ &\leq e^{-M\varepsilon^2/8}. \end{aligned} \quad (\text{Hoeffding's Inequality})$$

Therefore, we have that with probability at least $1 - e^{-M\varepsilon^2/8}$, it holds that $\frac{1}{M} \sum_{j=1}^M Y_j > \frac{k}{k+1} + \frac{\varepsilon}{4}$, which means that of the lists $\mathcal{A}(B_1), \dots, \mathcal{A}(B_M)$, strictly more than a $\frac{k}{k+1} + \frac{\varepsilon}{4}$ fraction identify L_z . By a similar argument as that at the end of the proof of [Theorem 9](#), this implies that $\text{Top-}k(S)$ contains $\text{first}_{\mathcal{C}}(z)$. Summarily, we have argued that for every $t \geq e^{t_0}$,

$$\Pr_{x_1, \dots, x_t \sim \mathcal{D}^t} [\mathcal{A}'(x_1, \dots, x_t) \not\equiv_{\text{id}} L_z] \leq \Pr_{x_1, \dots, x_t \sim \mathcal{D}^t} [\text{first}_{\mathcal{C}}(z) \notin \mathcal{A}'(x_1, \dots, x_t)] \leq e^{-M\varepsilon^2/8} \leq e^{-\frac{t\varepsilon^2}{8 \log t}}.$$

Thus, we have that

$$\begin{aligned}
\sum_{t \in \mathbb{N}} \Pr_{(x_i)_{i \in \mathbb{N}} \sim \mathcal{D}^\infty} [\mathcal{A}'(x_1, \dots, x_t) \not\equiv_{\text{id}} L_z] &= \sum_{t \in \mathbb{N}} \Pr_{x_1, \dots, x_t \sim \mathcal{D}^t} [\mathcal{A}'(x_1, \dots, x_t) \not\equiv_{\text{id}} L_z] \\
&\leq e^{t_0} + \sum_{t \geq e^{t_0}} \Pr_{x_1, \dots, x_t \sim \mathcal{D}^t} [\mathcal{A}'(x_1, \dots, x_t) \not\equiv_{\text{id}} L_z] \\
&\leq e^{t_0} + \sum_{t \geq e^{t_0}} e^{-\frac{t\epsilon^2}{8 \log t}} < \infty.
\end{aligned}$$

Then, by the Borel-Cantelli lemma, we have that

$$\Pr_{(x_i)_{i \in \mathbb{N}} \sim \mathcal{D}^\infty} [\exists i_1 < i_2 < i_3 < \dots : \forall j \in \mathbb{N}, \mathcal{A}'(x_1, x_2, \dots, x_{i_j}) \not\equiv_{\text{id}} L_z] = 0.$$

Since the language L_z and distribution \mathcal{D} were arbitrarily chosen, \mathcal{A}' satisfies this conclusion for every $L_z \in \mathcal{C}$ and every distribution \mathcal{D} valid for L_z . But since \mathcal{C} does not satisfy the k -Angluin condition (6), this contradicts [Corollary 8.1](#), concluding the proof. \square

8.2 Condition Satisfied \implies Exponential Rate

We will now argue that any collection that satisfies the k -Angluin condition (6) can be k -list identified at an exponential rate. While this can be shown by arguing that the list identification algorithm from [Section 5](#) achieves exponential rates, we will provide a more direct argument that uses the structural property of a k -list identifiable collection, which allows it to be decomposed into k identifiable collections.

Theorem 12 (Exponential Rate). *Let \mathcal{C} be a countable collection of languages that satisfies the k -Angluin condition (6). Then, \mathcal{C} can be k -list identified at rate $R(t) = e^{-t}$.*

Proof. Since \mathcal{C} satisfies the k -Angluin condition, [Theorem 6](#) ensures that it is k -list identifiable in the limit. Then, from [Theorem 2](#), we further know that \mathcal{C} may be expressed as $\mathcal{C} = \cup_{i=1}^k \mathcal{C}_i$, where each \mathcal{C}_i is identifiable in the limit, meaning that it satisfies Angluin's condition for identification (the base predicate given in (4)). From Proposition 3.10 in [\[KMV25\]](#), we then know that there exists an identifier \mathcal{A}_i for every \mathcal{C}_i that identifies languages in \mathcal{C}_i at an exponential rate. This immediately implies that the k -list identifier \mathcal{A} , which concatenates the outputs of $\mathcal{A}_1, \dots, \mathcal{A}_k$ that are run on the collections $\mathcal{C}_1, \dots, \mathcal{C}_k$ respectively, achieves an exponential rate for k -list identification of \mathcal{C} . \square

8.3 Exponential Rate Best Possible

We will now show that an exponential rate is effectively the best rate possible for list identification, upto a technical triviality condition. This technical condition captures the following triviality: if a collection \mathcal{C} satisfies that for every $k+1$ distinct languages $L_{i_1}, \dots, L_{i_{k+1}}$ in the collection, $\cap_{j=1}^{k+1} L_{i_j} = \emptyset$, then observe that any single string from the target language L_z pins down a list of at most k distinct languages that must necessarily identify L_z . That is, collections satisfying this property can be list-identified right from $t = 1$ (i.e., at a zero failure probability rate). So, adopting terminology from [\[KMV25\]](#), we will say that a collection is *non-trivial* for k -list identification if there exist $k+1$ distinct languages $L_{i_1}, \dots, L_{i_{k+1}}$ in \mathcal{C} which satisfy $\cap_{j=1}^{k+1} L_{i_j} \neq \emptyset$. The next theorem shows that a collection that is non-trivial for k -list identification cannot be k -list identified at a rate that is faster than exponential.

Theorem 13 (Exponential Rate Best Possible). *Let \mathcal{C} be a countable collection of languages that is non-trivial for k -list identification. Then, for every k -list identifier \mathcal{A} , there exists a language $L_z \in \mathcal{C}$ and a distribution \mathcal{D} valid for L_z such that $\Pr_{x_1, \dots, x_t \sim \mathcal{D}^t}[\mathcal{A}(x_1, \dots, x_t) \not\equiv_{\text{id}} L_z] \geq e^{-2t}$ for infinitely many $t \in \mathbb{N}$.*

Proof. Given that \mathcal{C} is non-trivial for k -list identification, there exist $k + 1$ distinct languages $L_{i_1}, \dots, L_{i_{k+1}}$ in \mathcal{C} that all share some string x . For each j , let \mathcal{D}_{i_j} be the distribution that assigns mass at least $1/2$ to x , and distributes the rest of the mass among the rest of the strings in L_{i_j} arbitrarily, in a way that every string gets assigned a positive mass. Then, for every \mathcal{D}_{i_j} , for any $t \in \mathbb{N}$, there is at least a $1/2^t$ chance that the first t strings $x_1, \dots, x_t \sim \mathcal{D}_{i_j}^t$ are all equal to x . Denoting this event by \mathcal{E}_t , we have that for every j , $\Pr_{x_1, \dots, x_t \sim \mathcal{D}_{i_j}^t}[\mathcal{E}_t] \geq \frac{1}{2^t}$. Observe now that conditioned on \mathcal{E}_t , because $\mathcal{A}(x_1, \dots, x_t)$ has size at most k ,

$$\sum_{j=1}^{k+1} \mathbb{1}[\mathcal{A}(x_1, \dots, x_t) \equiv_{\text{id}} L_{i_j}] \leq k,^9$$

which means that for some j , $\mathbb{1}[\mathcal{A}(x_1, \dots, x_t) \not\equiv_{\text{id}} L_{i_j}] = 1$. By the pigeonhole principle, it holds that for some j , $\mathbb{1}[\mathcal{A}(x_1, \dots, x_t) \not\equiv_{\text{id}} L_{i_j}] = 1$ conditioned on \mathcal{E}_t , for infinitely many $t \in \mathbb{N}$. We can therefore conclude that for such a j , for infinitely many t ,

$$\begin{aligned} \Pr_{x_1, \dots, x_t \sim \mathcal{D}_{i_j}^t}[\mathcal{A}(x_1, \dots, x_t) \not\equiv_{\text{id}} L_{i_j}] &\geq \Pr_{x_1, \dots, x_t \sim \mathcal{D}_{i_j}^t}[\mathcal{E}_t] \cdot \Pr_{x_1, \dots, x_t \sim \mathcal{D}_{i_j}^t}[\mathcal{A}(x_1, \dots, x_t) \not\equiv_{\text{id}} L_{i_j} \mid \mathcal{E}_t] \\ &\geq 2^{-t} \geq e^{-2t}. \end{aligned}$$

□

Acknowledgements

This work was supported by Moses Charikar's and Gregory Valiant's Simons Investigator Awards, and a Google PhD Fellowship.

References

- [Ang79] Dana Angluin. Finding patterns common to a set of strings. In *Proceedings of the eleventh annual ACM Symposium on Theory of Computing*, pages 130–141, 1979. [1](#)
- [Ang80] Dana Angluin. Inductive inference of formal languages from positive data. *Information and control*, 45(2):117–135, 1980. [1](#), [1.1](#), [2](#), [4](#)
- [Ang88] Dana Angluin. *Identifying languages from stochastic examples*. Yale University. Department of Computer Science, 1988. [1.1](#), [2](#), [8.1.2](#)
- [BB75] Lenore Blum and Manuel Blum. Toward a mathematical theory of inductive inference. *Information and control*, 28(2):125–155, 1975. [1.2](#)

⁹If \mathcal{A} is allowed to be randomized, we can instantiate this inequality as $\sum_{j=1}^{k+1} \Pr[\mathcal{A}(x_1, \dots, x_t) \equiv_{\text{id}} L_{i_j} \mid \mathcal{E}_t] = \mathbb{E} \left[\sum_{j=1}^{k+1} \mathbb{1}[\mathcal{A}(x_1, \dots, x_t) \equiv_{\text{id}} L_{i_j}] \mid \mathcal{E}_t \right] \leq k$, where the probability is only over the randomness in \mathcal{A} .

- [BBV08] Maria-Florina Balcan, Avrim Blum, and Santosh Vempala. A discriminative framework for clustering via similarity functions. In *Proceedings of the fortieth annual ACM symposium on Theory of computing*, pages 671–680, 2008. [1.2](#)
- [BCJ99] Ganesh R Baliga, John Case, and Sanjay Jain. The synthesis of language learners. *Information and Computation*, 152(1):16–43, 1999. [1.2](#)
- [BDPSS09] Shai Ben-David, Dávid Pál, and Shai Shalev-Shwartz. Agnostic online learning. In *COLT*, volume 3, page 1, 2009. [1](#)
- [BH04] R. Brown and C. Hanlon. Derivational complexity and order of acquisition in child speech. *First language acquisition: the essential readings*, page 155, 2004. [1](#)
- [Cas99] John Case. The power of vacillation in language learning. *SIAM Journal on Computing*, 28(6):1941–1969, 1999. [1.2](#)
- [CP23] Moses Charikar and Chirag Pabbaraju. A characterization of list learnability. In *Proceedings of the 55th Annual ACM Symposium on Theory of Computing*, pages 1713–1726, 2023. [1.2](#)
- [CP24] Moses Charikar and Chirag Pabbaraju. Exploring facets of language generation in the limit. *arXiv preprint arXiv:2411.15364*, 2024. [1.2](#)
- [CS83] John Case and Carl Smith. Comparison of identification criteria for machine inductive inference. *Theoretical Computer Science*, 25(2):193–220, 1983. [1.2](#)
- [CSV17] Moses Charikar, Jacob Steinhardt, and Gregory Valiant. Learning from untrusted data. In *Proceedings of the 49th annual ACM SIGACT symposium on theory of computing*, pages 47–60, 2017. [1.2](#)
- [DKS18] Ilias Diakonikolas, Daniel M Kane, and Alistair Stewart. List-decodable robust mean estimation and learning mixtures of spherical gaussians. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, pages 1047–1060, 2018. [1.2](#)
- [EHKV89] Andrzej Ehrenfeucht, David Haussler, Michael Kearns, and Leslie Valiant. A general lower bound on the number of examples needed for learning. *Information and Computation*, 82(3):247–261, 1989. [1.1](#)
- [GGKM21] Badih Ghazi, Noah Golowich, Ravi Kumar, and Pasin Manurangsi. Near-tight closure bounds for the littlestone and threshold dimensions. In *Algorithmic learning theory*, pages 686–696. PMLR, 2021. [1.1](#)
- [Gol67] E Mark Gold. Language identification in the limit. *Information and control*, 10(5):447–474, 1967. [1](#), [1.2](#)
- [Gur07] Venkatesan Guruswami. Algorithmic results in list decoding. *Foundations and Trends® in Theoretical Computer Science*, 2(2):107–195, 2007. [1.2](#)
- [JS90] Sanjay Jain and Arun Sharma. Language learning by a “team”. In *International Colloquium on Automata, Languages, and Programming*, pages 153–166. Springer, 1990. [1.2](#)

- [JS95a] Sanjay Jain and Arun Sharma. On aggregating teams of learning machines. *Theoretical computer science*, 137(1):85–108, 1995. 1.2
- [JS95b] Sanjay Jain and Arun Sharma. Team learning of formal languages. *WSIMLC95*, 1995. 1.2
- [JS96a] Sanjay Jain and Arun Sharma. Computational limits on team identification of languages. *Information and Computation*, 130(1):19–60, 1996. 1.2
- [JS96b] Sanjay Jain and Arun Sharma. Team learning of recursive languages. In *Pacific Rim International Conference on Artificial Intelligence*, pages 324–335. Springer, 1996. 1.2
- [JS00] Sanjay Jain and Arun Sharma. Team learning of computable languages. *Theory of Computing Systems*, 33(1):35–58, 2000. 1.2
- [KKK19] Sushrut Karmalkar, Adam Klivans, and Pravesh Kothari. List-decodable linear regression. *Advances in neural information processing systems*, 32, 2019. 1.2
- [KM24] Jon Kleinberg and Sendhil Mullainathan. Language generation in the limit. *Advances in Neural Information Processing Systems*, 37:66058–66079, 2024. 1, 1.2, 3
- [KMV24] Alkis Kalavasis, Anay Mehrotra, and Grigoris Velegkas. Characterizations of language generation with breadth. *arXiv preprint arXiv:2412.18530*, page 3, 2024. 1.2
- [KMV25] Alkis Kalavasis, Anay Mehrotra, and Grigoris Velegkas. 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*, pages 1732–1743, 2025. 1.1, 1.1, 1.2, 2, 8.1.2, 8.1.3, 8.2, 8.3
- [Lit88] Nick Littlestone. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine learning*, 2(4):285–318, 1988. 1
- [LRT24] Jiaxun Li, Vinod Raman, and Ambuj Tewari. Generation through the lens of learning theory. *arXiv preprint arXiv:2410.13714*, 2024. 1.2
- [MAC⁺25] Moses Charikar, Anay Mehrotra, Charlotte Peale, Chirag Pabbaraju, and Grigoris Velegkas. Tutorial on language generation in the limit. Tutorial presented at COLT 2025, 2025. Online: <https://languagegeneration.github.io/>. 1.2
- [McN70] David McNeill. The acquisition of language: The study of developmental psycholinguistics. 1970. 1
- [MSTY23] Shay Moran, Ohad Sharon, Iska Tsubari, and Sivan Yosebashvili. List online classification. In *The Thirty Sixth Annual Conference on Learning Theory*, pages 1885–1913. PMLR, 2023. 1.2
- [OSW86] Daniel N Osherson, Michael Stob, and Scott Weinstein. *Systems that learn: An introduction to learning theory for cognitive and computer scientists*. The MIT Press, 1986. 1.2
- [PF25] Hristo Papazov and Nicolas Flammarion. Learning algorithms in the limit. *arXiv preprint arXiv:2506.15543*, 2025. 1.2

- [Pit85] Leonard B. Pitt. *Probabilistic Inductive Inference*. Phd thesis, Yale University, 1985. Yale University Computer Science Department Technical Report TR-400. [1.2](#), [2](#), [8.1.1](#), [7](#)
- [Pit89] Leonard Pitt. Probabilistic inductive inference. *Journal of the ACM (JACM)*, 36(2):383–433, 1989. [1.2](#)
- [PS25] Chirag Pabbaraju and Sahasrajit Sarmasarkar. A characterization of list regression. In *36th International Conference on Algorithmic Learning Theory*, 2025. [1.2](#)
- [RY20] Prasad Raghavendra and Morris Yau. List decodable learning via sum of squares. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 161–180. SIAM, 2020. [1.2](#)
- [Smi82] Carl H Smith. The power of pluralism for automatic program synthesis. *Journal of the ACM (JACM)*, 29(4):1144–1165, 1982. [1.2](#)
- [VC71] VN Vapnik and A Ya Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability & Its Applications*, 16(2):264–280, 1971. [1.1](#)
- [Wik] Wikipedia. “Negative Evidence in Language Acquisition”. https://en.wikipedia.org/wiki/Negative_evidence_in_language_acquisition. Accessed: 2025-10-12. [1](#)