**UC Berkeley** **Jonathan Shafer**
**CS 294-220, Spring 2021**

**Unit 1**

# Language Identification in the Limit

## 1 Introduction

In this unit we will present a definition of learning called *language identification in the limit*. This definition was introduced in 1964 by E. Mark Gold [7, 8], and was a popular formal model of learning until the advent of the PAC model by Leslie Valiant in the 1980's. We will spend a lot of time in subsequent lectures discussing the PAC model of learning, but before we do so it is worthwhile to get acquainted with Gold's older definition. In particular, the limitations of Gold's definition and the lessons learned from it can serve as motivation for some of the choices made in the design of the PAC definition, and they highlight why the PAC definition makes sense. We will say more about this in Section 6 below, and at the beginning of the next unit.

Gold's definition of language identification in the limit is modeled on the following game. Suppose a *teacher* has in mind some arbitrary subset $L \subseteq \mathbb{N}$, and a *learner* needs to guess what the subset $L$ is. The game proceeds in steps, where in each step the teacher reveals an example $x \in L$, and then the learner makes a guess. For example:

*Teacher:* 6 *is in* $L$.
*Learner: I'm guessing that* $L = \{6\}$.
*Teacher:* 18 *is in* $L$.
*Learner: I'm guessing that* $L = \{6n : n \in \mathbb{N}\}$.
*Teacher:* 60 *is in* $L$.
*Learner: I'm guessing that* $L = \{6n : n \in \mathbb{N}\}$.
*Teacher:* 14 *is in* $L$.
*Learner: I'm guessing that* $L = \{2n : n \in \mathbb{N}\}$.
*Teacher:* 1002 *is in* $L$.
*Learner: I'm guessing that* $L = \{2n : n \in \mathbb{N}\}$.
...

In the game, the teacher never tells the learner whether the guess is correct or not. However, if the guess is correct then the learner will not see any examples $x \in L$ that are inconsistent with it, and so the learner will not have any compelling reason to change its guess from that point onward. The goal of the learner is to eventually *converge* on the correct language. Namely, that from some point onward, the learner's guess will be the correct language.

The fact that the teacher never tells the learner whether the guess is correct is akin to how scientists learn about the world. They attempt to find a hypothesis that is consistent with their observations. If they see an observation that is not consistent with their current hypothesis, they are inclined to change it in favor of one that is consistent. Nature never offers any definite confirmation that a hypothesis is correct. Instead, the hope is that scientists will eventually converge onto some good hypothesis that is consistent with all observations, and from that point onward they will no longer be compelled to change their hypothesis.[1]

Notice that in the game, the learner only receives *positive examples* of the form "$x$ is in $L$," and it does not receive negative examples of the form "$y$ is not in $L$." This was partially intended to model the way that children learn their first language. Children are typically exposed to positive examples, in which people use language correctly, but they are rarely exposed to negative examples. For example, children are likely to hear something like *don't tickle me*, but they are much less likely to hear an incorrect utterance like *don't giggle me* along with some indication that this utterance is not grammatical. See further discussion on language acquisition in Section 6 below.

## 2 Definitions

### 2.1 Preliminaries

We assume familiarity with basic notions from the theory of computation such as Turing machines, computable languages, computable functions, as well as enumerators and recursively enumerable languages. Sipser [16] is an excellent introduction to these topics.

▶ **Notation 1** (Tuples). We denote finite tuples using parentheses like $(x_1, x_2, \dots, x_n)$. A tuple of length 1 is denoted like so, $(x_1)$. The concatenation operator for tuples is denoted by $\circ$, e.g., $(x_1, \dots, x_n) \circ (y_1, \dots, y_n) = (x_1, \dots, x_n, y_1, \dots, y_n)$. For a tuple or a sequence $t$, we write $x_i \in t$ to denote that $x_i$ appears at some location in $t$, and we write $|t|$ to denote the length of $t$. If $t$ is an infinite sequence then $|t| = \infty$.

▶ **Notation 2.** $\mathbb{N} = \{1, 2, 3, \dots\}$. For any $n \in \mathbb{N}$, $[n] = (1, 2, 3, \dots, n)$.

▶ **Definition 3** (Symbols, Strings, Languages). *An* alphabet *is a finite set. If $\Sigma$ is an alphabet, then we call each member $\sigma \in \Sigma$ a* symbol. *A* string of length $n$ over $\Sigma$ *for some $n \in \mathbb{N}$ is an ordered tuple $s = (\sigma_1, \sigma_2, \dots, \sigma_n)$ such that $\sigma_1, \sigma_2, \dots, \sigma_n \in \Sigma$. We will often discard the parentheses and simply write $s = \sigma_1\sigma_2 \dots \sigma_n$. The set of all strings of length $n$ over $\Sigma$ for some fixed $n \in \mathbb{N} \cup \{0\}$ is denoted $\Sigma^n$. The (unique) string of length 0 is called* the empty string *and is denoted by $\varepsilon$. We use the* Kleene star *notation $\Sigma^* = \cup_{n=0}^{\infty}\Sigma^n$ to denote the set of all strings of finite length over $\Sigma$. For any strings $s = \sigma_1 \dots \sigma_n$ and $t = \tau_1 \dots \tau_m$ over $\Sigma$, we denote their* concatenation *by $s \circ t = \sigma_1 \dots \sigma_n\tau_1 \dots \tau_m$, and the length of each string by $|\cdot|$, namely $|s| = n$ and $|s \circ t| = n + m$. A* Language $L$ over $\Sigma$ *is a set of strings, namely $L \subseteq \Sigma^*$. We say that a language $L$ is* empty *if $L = \varnothing$. We say that $L$ is* finite *if the cardinality of the set $L$ satisfies $|L| < \infty$, and we say that $L$ is* infinite *if $|L| = \infty$.*

▶ **Definition 4** (Encodings). *For a mathematical object $x$, we write $\langle x \rangle$ to denote the encoding of $x$ as a string. Namely, $x \mapsto \langle x \rangle$ is assumed to be some reasonable and computable bijection $X \to \Sigma^*$ where $X$ is a set and $x \in X$. The details of the bijection will usually not be specified because they will not matter to us. We write $\langle x_1, \dots, x_n \rangle$ to denote the encoding of the tuple $(x_1, \dots, x_n)$.*

### 2.2 Language Identification in the Limit

The following is a formalization of the game from the introduction. The first definition describes the sequence of examples that are presented to the learner. The idea is that the

[1] This is of course a very rough illustration. In reality, science is a lot more messy than this.

examples need to be "fair" in the sense that the teacher should not omit any members of the language. If $x$ is in the language, then the teacher must at some point present $x$ as an example to the learner (otherwise, the learner would have no way of knowing that $x$ is in the language, and it would have no hope of converging onto the correct language).

▶ **Definition 5** (Positive Presentation of a Language). *Let $L \subseteq \Sigma^*$ be a language. A* positive presentation of $L$ *is an infinite sequence $s$ of strings $s_1, s_2, s_3, \dots$ such that $L = \{s_i\}_{i=1}^{\infty}$. In other words, the sequence $s$ satisfies the following two conditions:*

**(i)** *$s_i \in L$ for all $i \in \mathbb{N}$.*
**(ii)** *For any $s \in L$ there exists $i \in \mathbb{N}$ such that $s = s_i$.*

▶ Remark 6. The empty language $\varnothing$ does not have any positive presentations. Every non-empty language has a positive presentation, and any language with more than one string has infinitely many positive presentations.

The following two definitions describe the output of the learner. The idea is that the learner will output thing like "I'm guessing that the correct language is language no. 17", refering to an index of a language within some agreed-upon indexing of all the possible languages that the learner could guess.

▶ **Definition 7** (Computable Indexing). *Let $\mathcal{L}$ be a set of languages over an alphabet $\Sigma$. A* computable indexing of $\mathcal{L}$ *is an infinite sequence of languages $L_1, L_2, \dots$ such that:*

**(i)** *$\{L_i\}_{i=1}^{\infty} = \mathcal{L}$, and*
**(ii)** *There exists a Turing machine $D$ such that for any $i \in \mathbb{N}$ and any $s \in \Sigma^*$,*

$$D(\langle i, s \rangle) = \mathbb{1}(s \in L_i).$$

*Namely, if $D$ is executed with the encoding of $(i, s)$ as input then $D$ halts, and furthermore $D$ outputs $1$ if $s \in L_i$ and it outputs $0$ otherwise.*

▶ Remark 8. If a set of languages $\mathcal{L}$ has a computable indexing, then in particular each language in $\mathcal{L}$ is computable, $\mathcal{L}$ is finite or countable, and $\mathcal{L}$ is recursively enumerable in the sense that $D(\langle 1, \cdot \rangle), D(\langle 2, \cdot \rangle), \dots$ is an enumeration of deciders of the languages in $\mathcal{L}$.

▶ **Definition 9** (Sequence of Outputs, Convergence).

- *Let $L$ be a language, and let $s$ be a positive presentation of $L$. Let $M$ be a Turing machine. The* sequence of outputs of $M$ on $s$ *is $m_1, m_2, m_3, \dots$ such that*

$$m_i = M(\langle s_1, s_2, \dots, s_i \rangle)$$

*for all $i \in \mathbb{N}$. Namely, $m_i$ is the output of $M$ when executed on the input $\langle s_1, s_2, \dots, s_i \rangle$ which consists of the first $i$ strings in the positive presentation $s$. If $M$ does not halt on this input, we write $m_i = \bot$.*
- *We say that $M$ converges to a number $k \in \mathbb{N}$ on $s$, and write $M(s) \to k$, if $m_i \neq \bot$ for all $i \in \mathbb{N}$ and there exists $N \in \mathbb{N}$ such that $m_i = k$ for all $i > N$.*
- *If $M$ does not converge to any number $k$, we say that $M$* diverges on $s$.

Finally, the game itself is captured in the following definition.

▶ **Definition 10** (Language Identification in the Limit, [8]). *Let $\mathcal{L}$ be a set of languages over an alphabet $\Sigma$, let $L \in \mathcal{L}$, let $I = (L_i)_{i \in \mathbb{N}}$ be a computable indexing of $\mathcal{L}$, and let $M$ be a Turing machine. We say that $M$* identifies $L$ in the limit with respect to indexing $I$ *if for any positive presentation $s$ of $L$, $M(s) \to i$ for $i \in \mathbb{N}$ such that $L = L_i$. We say that $M$* identifies $\mathcal{L}$ in the limit with respect to indexing $I$ *if $M$ identifies every language in $\mathcal{L}$.*

*We say that a set of languages $\mathcal{L}$ (or a language $L$) is* identifiable in the limit *if there exists a Turing machine $M$ and a computable indexing $I$ of $\mathcal{L}$ such that that $M$ identifies $\mathcal{L}$ (or $L$) in the limit with respect to $I$.*

▶ Remark 11. Because $\varnothing$ does not have any positive presentations, every Turing machine identifies $\varnothing$ in the limit in a vacuous sense.

## 3 Some Simple Examples

▶ **Example 12.** *The set $\mathcal{L}$ of all finite languages over an alphabet $\Sigma$ is identifiable in the limit.*

**Proof sketch.** We define the learner $M$ to be a TM as follows. Let $L$ be the target language and let $s$ be a positive presentation of $L$. For any finite prefix $s_1, s_2, \dots, s_n$ of $s$, the learner $M$ will output $M(\langle s_1, \dots, s_n \rangle) = k$ such that $L_k = \{s_1, \dots, s_n\}$. Observe that because $s$ is a positive presentation, for every $w \in L$ there exists a minimal index $m = m(w) \in \mathbb{N}$ such that $s_m = w$. Because $L$ is finite, the value $N = \max\{m(w) : \ w \in L\}$ is a finite integer. For all $n \geq N$, the language $L_k = \{s_1, \dots, s_n\}$ as above will contain all the strings in $L$, and only strings in $L$. Thus, $M(s) \to k$ where $L_k = L$ as desired.

To complete the proof fomally, it is necessary to verify that the behavior of $M$ as specified above is indeed computable. That is, we need to show that there exists a computable indexing $(L_i)_{i=1}^{\infty}$ of $\mathcal{L}$ such that for any finite set $\{s_1, \dots, s_n\}$, the mapping $\{s_1, \dots, s_n\} \mapsto k$ such that $L_k = \{s_1, \dots, s_n\}$ is a computable mapping. That is not difficult. ◀

▶ **Example 13.** *The set $\mathcal{L}$ be the set of co-singleton sets, namely $\mathcal{L} = \{\mathbb{N} \setminus \{n\} : \ n \in \mathbb{N}\}$. Then $\mathcal{L}$ is identifiable in the limit.*

**Proof sketch.** Using the same notation as in Example 12, the learner $M$ will compute

$$c = \min\{i \in \mathbb{N} : \ i \notin \{s_1, \dots, s_n\}\},$$

and then output an index $k$ such that $L_k = \mathbb{N} \setminus \{c\}$. Assume that the target language is $L = \mathbb{N} \setminus \{t\}$ for some fixed $t \in \mathbb{N}$. Let

$$N = \max\{m(s) : \ s \in \mathbb{N} \ \wedge \ s < t\},$$

and note that $N$ is a finite integer. For all $n \geq N$ it holds that $M(\langle s_1, \dots, s_n \rangle) = t$ as desired, because all the natural numbers smaller than $t$ appear in the prefix $s_1, \dots, s_n$ (by the choice of $N$), and $t$ doesn't appear in the prefix because $t \notin L$.

Finally, to complete the proof we note that there exists a computable indexing of $\mathcal{L}$ such that the mapping $c \mapsto k$ such that $L_k = \mathbb{N} \setminus \{c\}$ is computable. Hence, $M$ is a well-defined Turing machine. ◀

▶ Remark 14. Language identification in the limit explicitly models only learning from positive examples ("$x$ is in $L$" – a "yes" instance), and the learner does not see negative examples ("$x$ is not in the language" – a "no" instance). Nonetheless, we can use the same formalism more generally to also model learning functions and sequences. If $X$ and $Y$ are finite or countable sets and $f : X \to Y$ is a function, we can model learning the function by considering the identification in the limit of the language $L = \{(x, y) : \ x \in X \ \wedge \ y = f(x)\}$. In particular, if $Y = \{0, 1\}$, we can think of examples of the form $(x, 1)$ as "yes" instances, and examples of the form $(x, 0)$ as "no" instances. Additionally, because a sequence is simply a function where the domain is the natural numbers, we can also use this identification in the limit to model a form of sequence learning (learning to complete an infinite sequence like $2, 3, 5, 7, 11, \dots$ from a finite number of examples). However, the presentation of the sequence to the learner might not be in order (e.g., the learner is informed that "the 3rd item in the sequence is 5" without necessarily being first informed about the previous items). ⌟

## 4 Negative Result: Gold's Theorem, Locking Sequences

▶ **Definition 15** (Locking Sequence for a Language, [4]). *Let $L \subseteq \Sigma^*$ be a language, let $M$ be a Turing machine, and let $k \in \mathbb{N}$. We say that a finite sequence of strings $s_1, s_2, \dots, s_n \in \Sigma^*$ is* a locking sequence for $L$ that locks $M$ onto $k$ *if the following conditions hold:*

**(i)** *$\{s_1, \dots, s_n\} \subseteq L$, and*

**(ii)** *For any (possibly empty) finite sequence $t_1, t_2, \dots, t_m \in \Sigma^*$, if $\{t_1, \dots, t_m\} \subseteq L$ then*

$$M(\langle s_1, \dots, s_n, t_1, \dots, t_m\rangle) = k.$$

▶ **Lemma 16** (Locking Sequence Lemma, [4]). *Let $\mathcal{L}$ be a set of languages with a computable indexing $(L_i)_{i\in\mathbb{N}}$, let $L \in \mathcal{L}$, and let $M$ be a Turing machine. If $M$ identifies $L$ in the limit with respect to $(L_i)_{i\in\mathbb{N}}$ then there exists a locking sequence for $L$ that locks $M$ onto $k$ such that $L = L_k$.*

**Proof.** Assume for contradiction that there does not exists a locking sequence for $L$ that locks $M$ onto any number $k \in \mathbb{N}$ such that $L_k = L$. We make two observations.

1. If there exists a locking sequence for $L$ that locks $M$ onto some number $k$, then it must be that $L_k = L$. To see this, assume for contradiction that there exists a locking sequence $s$ for $L$ that locks $M$ onto $k$ such that $L_k \neq L$. Let $p$ be a positive presentation of $L$. Then the concatenation $s \circ p$ is also a positive presentation of $L$. From the assumption that $s$ is a locking sequence, $M(s \circ p) \to k$. Seeing as $s \circ p$ is a positive presentation for $L$ and $L_k \neq L$, this is a contradiction to the assumption that $M$ identifies $L$ in the limit.
2. For any finite sequence $s$ of strings from $L$, there exists a finite sequence $\mathrm{unlock}(s)$ of strings from $L$ such that $M(\langle s\rangle) \neq M(\langle s \circ \mathrm{unlock}(s)\rangle)$. This is true because if there exists a sequence $s$ of strings from $L$ that does not have a suitable sequence $\mathrm{unlock}(s)$, then $s$ would be a locking sequence for $L$, and therefore from the first observation it must be that $s$ locks $M$ onto $k$ such that $L = L_k$, in contradiction to the assumption that no such locking sequence exists.

Now, Let $p$ be a positive presentation of $L$. We will construct a positive presentation $q$ for $L$ on which $M$ diverges, which is a contradiction to the assumption that $M$ identifies $L$ in the limit:

**Listing 1** Construction of the sequence $q$

```
$q$ ← empty sequence
for $i = 1, 2, \dots$ :
    $q$ ← $q \circ (p_i)$
    $q$ ← $q \circ \mathrm{unlock}(q)$
```

Namely, the construction alternates between extending $q$ with the next string from $p$, and extending $q$ with its own unlocking sequence. Note that:

**(a)** $q$ is a positive presentation for $L$. That is, $q$ is a well defined infinite sequence of strings, all the strings in $q$ are members of $L$ (by the definitions of $p$ and $\mathrm{unlock}(\cdot)$), and every string in $L$ appears in $q$ at some point (because $p$ is a positive presentation of $L$).

**(b)** $M$ diverges on $q$. This is true because for each step $i$ in the construction of $q$,

$$M(\langle q \rangle) \neq M(\langle q \circ \text{unlock}(q) \rangle),$$

so the output of $M$ changes infinitely many times when executing on the input $q$.

Taken together, (a) and (b) are a contradiction to the assumption that $M$ identifies $L$ in the limit. ◀

It would appear that identification in the limit is a very generous model towards the learner, in the sense that the learner gets infinitely many guesses. The learner is allowed to make any finite number of mistakes, so long as it eventually converges onto the right answer. Perhaps surprisingly, the following theorem shows that even simple collections of languages are not identifiable in the limit.

▶ **Definition 17.** *A set of languages is called* transfinite *if it contains all finite languages over some alphabet $\Sigma$ and at least one infinite language over $\Sigma$.*

▶ **Theorem 18** (Gold [8])**.** *Every transfinite set of languages is not identifiable in the limit.*

**Proof.** Let $\mathcal{L}$ be a transfinite set of languages, and assume for contradiction that there exists a Turing machine $M$ that identifies $\mathcal{L}$ in the limit with respect to some computable indexing $(L_i)_{i \in \mathbb{N}}$. Let $L_\infty \in \mathcal{L}$ be an infinite language. From Lemma 16 there exists a finite locking sequence $s$ for $L_\infty$ that locks $M$ onto $k$ such that $L_k = L_\infty$. Let $L_s$ be the (finite) set of strings in $s$. Let $s_1$ be the first string in $s$, and consider the infinite sequence $p = s \circ (s_1, s_1, s_1, \dots)$. Note that $p$ is a positive presentation for $L_s$. Because $\mathcal{L}$ is trasfinite, $L_s \in \mathcal{L}$, and therefore $M$ identifies $L_s$ in the limit. In particular, $M(p) \to t$ such $L_t = L_s$. Because $L_s \neq L_\infty$, $t \neq k$. This is a contradiction to $s$ being a locking sequence that locks $M$ onto $k$. ◀

▶ **Exercise 19.** *Show that $\mathcal{L} = \{\mathbb{N} \setminus \{i\}\}_{i=1}^{\infty} \cup \{\mathbb{N}\}$ is not identifiable in the limit.*

## 5 A Characterization of the Identifiable Sets of Languages

The proof of Gold's theorem highlights the main problem that makes language identification in the limit difficult to achieve: at some point, the learner might make up its mind that the target language is some language $L$ (e.g., if the learner has seen a locking sequence for $L$), and from that point onward, it will not change its mind unless it sees a string that does not belong to $L$. If the correct target language is actually a subset $L' \subsetneq L$, then the learner will fail to identify it. We call this phenomena *over-generalization*, because the learner fixates on the solution $L$ that is too general (too big a set), instead of identifying the correct solution $L'$ that is more particular (is a smaller set).

The following theorem basically says that over-generalization is the only problem that can make a set of languages not identifiable. A set of languages is identifiable if and only if over-generalization can be avoided.

▶ **Definition 20** (Telltale Set, [2])**.** *Let $\mathcal{L}$ be a set of languages, and let $L \in \mathcal{L}$. A finite set of strings $T \subseteq L$ is called a* telltale set for $L$ with respect to $\mathcal{L}$ *if for any $L' \in \mathcal{L}$,*

$$T \subseteq L' \Longrightarrow L' \not\subset L,$$

*namely every language $L' \in \mathcal{L}$ that is a strict subset of $L$ does not contain $T$.*

▶ Remark 21. $\varnothing$ is a telltale set of itself.

▶ **Theorem 22** (Angluin [2]). *A a set of languages $\mathcal{L}$ over an alphabet $\Sigma$ is identifiable in the limit if and only if the following two conditions are satisfied:*

**(i)** *Every language $L \in \mathcal{L}$ has a telltale set with respect to $\mathcal{L}$, and furthermore*

**(ii)** *The telltale sets are recursively enumerable in the following sense. There exists a computable indexing $(L_i)_{i\in\mathbb{N}}$ of $\mathcal{L}$ and a Turing machine $T$ such that for any $i \in \mathbb{N}$, if $T$ is executed with input $\langle i\rangle$ then $T$ enumerates a set $T_i$ that is a telltale set for $L_i$.*

**Proof.**
$\Longleftarrow$. We assume that $\mathcal{L}$ satisfies conditions (i) and (ii) and prove that $\mathcal{L}$ is identifiable in the limit. Let $(L_i)_{i\in\mathbb{N}}$ be the computable indexing of $\mathcal{L}$ from condition (ii). We write $T_i^{(n)}$ to denote the subset of the telltale set $T_i$ that is printed during the first $n$ steps of executing $T$ on input $\langle i\rangle$. Note that the mapping $(i, n) \mapsto T_i^{(n)}$ is computable. The Turing machine $M$ that identifies $\mathcal{L}$ operates as follows.

```
Input: a finite sequence of strings $s_1, s_2, \ldots, s_n$.

for $k = 1, 2, \ldots, n$:
  if $T_k^{(n)} \subseteq \{s_1, \ldots, s_n\} \subseteq L_k$:          ▷ $T_k^{(n)}$ is a subset of the telltale set for $L_k$
    output $k$ and halt

output 0 and halt
```

**Listing 2** A Turing machine $M$ that identifies $\mathcal{L}$ in the limit.

Let $L \in \mathcal{L}$, and let $s$ be a positive presentation of $L$. Let $j \in \mathbb{N}$ be the minimal number such that $L_j = L$. We show that $M(s) \to j$. Let $T_j$ be the telltale set enumerated for $L_j$ by $T$. Because $T_j \subseteq L_j = L$ and $s$ is a positive presentation of $L$, it holds that $T_j \subseteq \{s_1, \ldots, s_n\}$ for all $n$ large enough, and therefore $T_j^{(n)} \subseteq \{s_1, \ldots, s_n\}$ for all $n$ large enough. We conclude that for $k = j$ the condition

$$T_k^{(n)} \subseteq \{s_1, \ldots, s_n\} \subseteq L_k \tag{$*$}$$

is satisfied for all $n$ large enough.

To prove that $M(s) \to j$, it suffices to show that for all $k < j$, condition $(*)$ is satisfied at most finitely many times. Fix a value $k < j$. Fix $N$ large enough such that $T_k^{(n)} = T_k$ for all $n \geq N$ ($N$ exists because $T_k$ is finite). Assume that this value of $k$ satisfies $(*)$ for some fixed $n \geq N$. Then $T_k \subseteq \{s_1, \ldots, s_n\}$ (because $(*)$ is satisfied and $T_k^{(n)} = T_k$), and also $\{s_1, \ldots, s_n\} \subseteq L$ (because $s$ is a presentation of $L$). Hence, $T_k \subseteq L$. Because $T_k$ is a telltale set for $L_k$, this implies that $L \not\subset L_k$. Furthermore, from the minimality of $j$, $L \neq L_k$. Hence, there exists a string $w$ such that $w \in L$ and $w \notin L_k$. Because $s$ is a positive presentation of $L$, there exists an index $q$ such that $s_q = w$. Therefore, this value of $k$ does not satisfy $(*)$ whenever $n \geq q$. Thus, we have shown that $j$ is the minimal number that satisfies $(*)$ for all $n$ large enough, and therefore $M(s) \to j$. This shows that $\mathcal{L}$ is identifiable in the limit.

$\Longrightarrow$. We assume that $\mathcal{L}$ is identifiable in the limit and prove that $\mathcal{L}$ satisfies condition (ii), and this implies that condition (i) is also satisfied. Let $M$ be a Turing machine that identifies $\mathcal{L}$ with respect to some computable indexing $(L_i)_{i\in\mathbb{N}}$ of $\mathcal{L}$.

First, we define a Turing machine EnumerateFiniteSequences that on input $\langle i\rangle$ enumerates all the finite sequences of strings in $L_i$. It uses $D$, the Turing machine for the computable indexing $(L_i)_{i\in\mathbb{N}}$ such that $s \in L_i \iff D(\langle i, s\rangle) = 1$.

```
EnumerateFiniteSequences($i$):     ▷ enumerates finite sequences of strings from $L_i$
  for $n = 0, 1, 2, \ldots$:
    $S \leftarrow \varnothing$
    for $s$ in first $n$ strings of $\Sigma^*$:
      if $D(i, s) = 1$:
        add $s$ to $S$
    for $q$ in Helper($S, n$):
       print $q$

Helper($S, n$):          ▷ enumerates all sequences of length $\leq n$ from finite set $S$
  $Q \leftarrow \varnothing$
  if $n = 0$:
    add $()$ to $Q$                                  ▷ $()$ denotes an empty sequence
  else:
    for $q$ in Helper($S, n-1$):
      add $q$ to $Q$
      for $s$ in $S$:
        add $q \circ (s)$ to $Q$
  return $Q$
```

**Listing 3** A Turing machine that enumerates all the finite sequences of strings from a langauge $L_i$.

Second, we construct the following Turing machine to enumerate the telltale sets as desired. The construction uses the fact that for every $i \in \mathbb{N}$ it is possible to compute the members $p_1, p_2, \ldots$ of a positive presentation $p$ for $L_i$ (by enumerating $\Sigma^*$ and using $D$ to filter out strings that do not belong to $L_i$).

```
T($i$):
  $q \leftarrow ()$
  $m_0 \leftarrow M(q)$
  for $j = 1, 2, 3, \ldots$:
    for $f$ in EnumerateFiniteSequences($i$):
      $m_j \leftarrow M(q \circ f)$
      if $m_j \neq m_{j-1}$:
        for $s$ in $f \circ (p_j)$:
          print $s$
        $q \leftarrow q \circ f \circ (p_j)$
        break                                              ▷ continue to next $j$
```

**Listing 4** A Turing machine that enumerates a telltale set $T_i$ for language $L_i$.

Third, we argue that for any $i \in \mathbb{N}$, $T$ prints only a finite number of strings when executed on input $\langle i \rangle$. Assume for contradiction that $T$ prints an infinite number of strings for some $i \in \mathbb{N}$. Then the sequences $q$ and $m_1, m_2, \ldots$ (that are constructed during the computation) are both infinite. However, observe that $q$ contains only members of $L_i$, and it contains all members of $L_i$ (because in each step $j$ we append $p_j$ to $q$). Hence, $q$ is a positive presentation of $L_i$. But from the construction of $m_j$, it holds that $m_j \neq m_{j-1}$ for all $j \in \mathbb{N}$. Hence, $M(q)$ diverges, in contradiction to $M$ identifying $L_i$ in the limit. Thus, the set of strings $T_i$ that $T$ prints on input $\langle i \rangle$ is finite.

Fourth, let $t$ be the finite sequence of all the strings printed out when executing $T$ on $\langle i \rangle$. We argue that $t$ is a locking sequence for $L_i$ that locks $M$ onto some index $k$ such that $L_k = L_i$. Observe that there exists no nonempty finite sequence $f$ of strings from $L_i$ such that $M(\langle t \circ f \rangle) \neq M(\langle t \rangle)$, because if such a sequence $f$ existed then $T$ would have eventually enumerated it and then printed the strings in $f$ after it printed $t$, which is a contradiction to our choice of $t$. This implies that $t$ is a locking sequence for $L$ that locks $M$ onto some index $k = M(\langle t \rangle)$. To see that $L_k = L_i$, note that $t \circ p$ is a positive presentation of $L_i$, and so $M(t \circ p) \to j$ such that $L_j = L_i$. In particular, there exists some finite prefix $f$ of $p$ such that $M(\langle t \circ f \rangle) = j$, and so $k = M(\langle t \rangle) = M(\langle t \circ f \rangle) = j$, and $L_k = L_j = L_i$.

Finally, we argue that the set $T_i$ of strings in $t$ is indeed a telltale set for $L_i$. Clearly, $T_i \subseteq L_i$, and from our third argument it is finite. Assume for contradiction that there exists $m \in \mathbb{N}$, $m \neq i$, such that $T_i \subseteq L_m \subsetneq L_i$. Let $w$ be a positive presentation of $L_m$. Then $t \circ w$ is also a positive presentation of $L_m$. Hence $M(t \circ w) \to m$, in contradiction to $t$ being a locking sequence for $L_i$ that locks $M$ onto $i$.

Note that in the corner case $L_i = \varnothing$, it holds that $\varnothing$ is a telltale set for $L_i$, and indeed $T$ does not print out any strings on input $\langle i \rangle$. ◀

## 6 Discussion

Golds's theorem shows that in some cases, identification in the limit is not possible because of over-generalization, and Angluin's theorem tells us that over-generalization is essentially the only type of problem that can make identification in the limit not possible. However, there are good reasons to believe that when humans learn in the real world, over-generalization is a difficulty that is readily and routinely overcome. The motto is:

*Absence of evidence can be evidence of absence.*

Let's see a few examples that clarify this idea:

1. **Lunch (silly example).** Alice and Bob eat lunch together every day at the cafeteria, which serves three dishes: pasta, soup, and salad. Alice notices that Bob always ordered the pasta or the soup, but he never ordered the salad. Over time, she develops an expectation that Bob will order the pasta or the soup. If someone asks her if she thinks Bob will order salad tomorrow, she will say that it is unlikely. Alice uses the absence of evidence (no cases of Bob ordering a salad) as evidence of absence (evidence that salad is not a member of the set of dishes that Bob is likely to order). This kind of reasoning is common and can work well in the real world, but the framework of language identification in the limit doesn't fully capture this (for identification in the limit, there is no difference between the case where Bob ordered the pasta and the soup a single time each, versus the case where Bob ordered each of them a million times).

2. **Scientific induction.** In the scientific method, scientists select a particular hypothesis out of a seemingly infinite set of possible hypotheses. In particular, they appear quite good at avoiding over-generalization. Consider the choice between the hypothesis (a) *steel balls fall downwards*, versus the more general hypothesis (b) *steel balls fall downwards, except on nights with a full moon in which they can fall down or float up into the sky* (the predictions of (a) are a strict subset of the predictions of (b)). Scientists do overgeneralize sometimes, but they also appear very good at avoiding over-generalization, and at using absence of evidence as evidence of absence (absence of evidence for a hypothesis counts against it).

3. **Language acquisition.** When children learn their first language, they often go through stages of incorrect over-generalization, where they make semantic, syntactic and morphological mistakes. Following are some examples (mistakes are marked with "*" signs).

- Semantic mistakes:

  Using **doggie* to refer to all biggish animals (e.g., bears and horses).

  Using **kitten* to refer to all felines, including lions and tigers.
- Morphological mistakes:

  Saying **I goed to the store with mommy.* Should be: *went.*

  Similarly, using **knowed*, **eated*, **gived*, **sitted* instead of *knew*, *ate*, *gave*, *sat.*

  Saying **Auntie is a good cooker.* Should be: *cook.*
- Syntactical mistakes:

  Saying **Don't giggle me.* Should be: *don't make me giggle.*

  Saying **Don't say me that.* Should be: *Don't say that to me.*

All these examples can be understood as cases where the child is using a rule or word that exists in English in a situation where it is not applicable. For example:

- *Walk*, *walked* $\longrightarrow$ *go*, **goed*
- *Build*, *builder* $\longrightarrow$ *cook*, **cooker*
- *I eat*, *don't eat me* $\longrightarrow$ *I giggle*, **don't giggle me*

These mistakes are interesting because they seem to satisfy the following three conditions.

**(i)** Children probably reach these incorrect generalizations on their own. They don't hear adults making mistakes like **goed* or **eated* very often.

**(ii)** It appears that children do not get a lot of direct negative feedback informing them that these are mistakes, and when they do get feedback, they tend to ignore it. This phenomena is called *poverty of the stimulus*, and there is a wealth of literature in linguistics that documents and discusses it.

**(iii)** Children eventually stop making these types of mistakes, despite the lack of negative feedback.

It appears probable that at least part of the mechanism that allows children to evetually stop making these mistakes operates along the following lines: *walk* and *walked* are both common, *kick* and *kicked* are both common – but **goed* is not common even though *go* is common. By picking up on these frequencies, children can identify the lack of evidence in favor of **goed* being a valid verb form, and interpret that absence of evidence as evidence that **goed* is in fact not a valid form.

The discussion above suggests that it would be valuable to consider a model of learning in which absence of evidence can count as evidence of absence, and that such a model could capture instances of learning that are common in the real world. As we will see, this can be done using *probability.*

Beyond that, another issue ignored (by design) in the model of identification in the limit is the issue of *resources.* Because real-world learners have limitations on their computational space and runtime complexities, as well as the amount of training data that they receive, it makes sense to ask *what can be learned efficiently?* (namely, using a reasonable amount of time, space, and data).

To summarize, the wish list for our next definition of learning is:

1. Probabilistic (absence of evidence is evidence of absence).
2. Account for time, space and data complexity.

In the next lecture we will present the PAC definition, which satisfies this wish list.

## 7 Bibliographic Notes

We have mostly followed the presentation of Gierasimczuk and de Jongh [6] and of Angluin [2]. See also the survey by Angluin and Smith [3], and the monograph by Osherson, Stob and Weinstein [13].

Gold [8] uses the terminology *arbitrary text* instead of *positive presentation* (which is used by Angluin [2]). Some authors (including Angluin [2]) use the terminology *inductive inference* or *inferring a language from positive data* instead of *identification in the limit.*

Most authors define the learner and the sequence of outputs in a different manner then we did. In Definition 9 we considered an ordinary Turing machine that is executed repeatedly on finite prefixes of the positive presentation $s$, producing a sequence of outputs $m_i$. In contrast, most sources consider a multi-tape Turing machine where the infinite positive presentation $s$ is written sequentially on a special input tape, and for each $i$ the Turing machine reads $s_i$, and then prints out $m_i$ to a special output tape before advancing to the next $i$. Their definition has the advantage that the Turing machine is more efficient because it doesn't start the computation from scratch for each $i$. A disadvantage of their definition that it requires a more complex setup (e.g., a multi-tape Turing machine, sometimes with special *request* and *guess* states, etc.). Because in the current note we are not interested in computational complexity, we used the simpler definition.

Theorem 18 is due to Gold [8], but we presented a simpler proof based on [4].

The linguistics literature about language acquisition, over-generalization, poverty of the stimulus, and their (possible) connections to Gold's theorem is vast (and I am not an expert on it); see e.g., [15, 9, 5, 14, 12, 10, 11]. Most of the examples of over-generalization in childrens' linguistic production were taken from [1].

### References

1 Ben Ambridge, Julian M Pine, Caroline F Rowland, Franklin Chang, and Amy Bidgood. The retreat from overgeneralization in child language acquisition: Word learning, morphology, and verb argument structure. *Wiley Interdisciplinary Reviews: Cognitive Science*, 4(1):47–62, 2013.

2 Dana Angluin. Inductive Inference of Formal Languages from Positive Data. *Inf. Control.*, 45(2):117–135, 1980. `doi:10.1016/S0019-9958(80)90285-5`.

3 Dana Angluin and Carl H. Smith. Inductive inference: Theory and methods. *ACM Comput. Surv.*, 15(3):237–269, 1983. `doi:10.1145/356914.356918`.

4 Lenore Blum and Manuel Blum. Toward a Mathematical Theory of Inductive Inference. *Inf. Control.*, 28(2):125–155, 1975. `doi:10.1016/S0019-9958(75)90261-2`.

5 Alexander Clark. Unsupervised language acquisition: Theory and practice. *arXiv preprint cs/0212024*, 2002.

6 Nina Gierasimczuk and Dick de Jongh. Gold's Theorems and Locking Sequences. 2013. URL: `https://web.archive.org/web/20130503081642/http://www.ninagierasimczuk.com/flt2013/wp-content/uploads/2013/01/lecture2_handout.pdf`.

7 E. Mark Gold. Language identification in the limit. RAND Research Memorandum RM-4136-PR, 1964.

8 E. Mark Gold. Language Identification in the Limit. *Inf. Control.*, 10(5):447–474, 1967. `doi:10.1016/S0019-9958(67)91165-5`.

9 Kent Johnson. Gold's theorem and cognitive science. *Philosophy of Science*, 71(4):571–592, 2004.

10 Brian MacWhinney. The (il) logical problem of language acquisition. In *Proceedings of the Fifteenth Annual Conference of the Cognitive Science Society*, pages 61–70. Lawrence Erlbaum Associates Hillsdale, NJ, 1993.

11 Brian MacWhinney. A multiple process solution to the logical problem of language acquisition. *Journal of child language*, 31(4):883–914, 2004.

12 Martin A Nowak, Natalia L Komarova, and Partha Niyogi. Computational and evolutionary aspects of language. *Nature*, 417(6889):611–617, 2002.

13 D. Osherson, M. Stob, and S. Weinstein. *Systems that Learn: An Introduction to Learning Theory.* MIT press, 1986.

14 Steven Pinker. Clarifying the logical problem of language acquisition. *Journal of Child Language*, 31(4):949–953, 2004.

15 Douglas LT Rohde and David C Plaut. Language acquisition in the absence of explicit negative evidence: How important is starting small? *Cognition*, 72(1):67–109, 1999.

16 Michael Sipser. *Introduction to the Theory of Computation.* Cengage Learning, Boston, MA, third edition, 2013.