

2.5 Background: the Chomsky hierarchy

- (1) In linguistic theory, we idealize to simplify the problem, thinking of a language as a (non-empty) set of expressions built from some (finite, non-empty) vocabulary Σ , but actually what we are interested in is how linguistic structure is assigned to anything.

As we will see, the sets of expressions are a useful idealization in the study of the computational mechanisms for assigning structure.

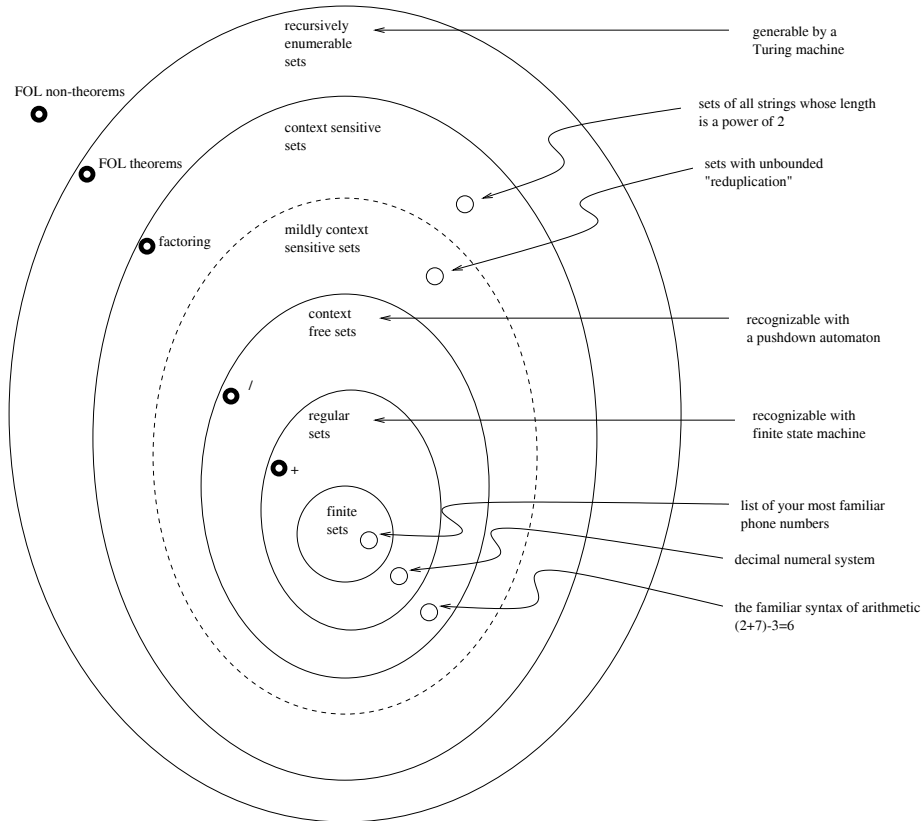
In learnability theory, we idealize to simplify the problem, thinking of what can be learned as a set of languages, but actually what we are interested in is how we come to recognized linguistic structure in anything.

The sets of expressions are a useful idealization in the study of the computational mechanisms for coming to recognize structure.

- (2) Chomsky noticed in the 1950's that we can distinguish sets of languages according to the complexity of the patterns that can occur in those languages. This idea is one of the fundamental ones in the theory of computation.
- (3) A **rewrite grammar** $G = \langle Cat, \Sigma, S, R \rangle$ where
- Cat (categories) is a finite nonempty set;
 - Σ (vocabulary) is a finite nonempty set such that $Cat \cap \Sigma = \emptyset$;
 - $S \in Cat$ (the "start" or "sentence" category);
 - $R \subseteq ((\Sigma \cup Cat)^* \times Cat \times (\Sigma \cup Cat)^*) \times (\Sigma \cup Cat)^*$ (the rewrite rules)
- (4) Let ϵ be the empty sequence. For any set S , let $S^\epsilon = S \cup \{\epsilon\}$
- (5) We often write $x \rightarrow y$ to indicate that $\langle x, y \rangle \in R$;
- (6) For any $w, z \in (\Sigma \cup Cat)^*$, we say $w \Rightarrow z$ iff there are $x, y, u, v \in (\Sigma \cup Cat)^*$ such that
- $w = uxv$,
 - $z = uyv$, and
 - $x \rightarrow y$.
- (7) The relation \Rightarrow^+ is the transitive closure of \Rightarrow .
The relation \Rightarrow^* is the reflexive, transitive closure of \Rightarrow .
- (8) Given a rewrite grammar G , $L(G) = \{s \in \Sigma^* \mid S \Rightarrow^* s\}$.
- (9) A rewrite grammar G is
- a **list grammar** iff $R \subseteq (\{S\} \times \Sigma^*)$;
 - a **regular (right branching) grammar** iff $R \subseteq ((Cat \times (\Sigma \times Cat)) \cup (Cat \times \{\epsilon\}))$;
 - a **context free grammar** iff $R \subseteq Cat \times (\Sigma \cup Cat)^*$;
 - a **context sensitive grammar** iff $R \subseteq \{\langle xAy, xwy \rangle \mid x, y \in (\Sigma \cup Cat)^*, A \in Cat, w \in (\Sigma \cup Cat)^+\}$
- (10) a. The set of languages generated by list grammars is the set \mathcal{L}_{fin} of finite languages;
- b. The set of languages generated by regular (right branching) grammars is the set \mathcal{L}_{fs} of regular, or finite state languages;
- c. The set of languages generated by context free grammars is the set \mathcal{L}_{cf} of context free languages;
- d. The set of languages generated by context sensitive grammars is the set \mathcal{L}_{cs} of context sensitive languages;
- e. The set of languages generated by all the grammars grammars is the set \mathcal{L}_{re} of recursively enumerable languages.

(11) The Chomsky hierarchy specifies proper inclusion relations among collections of languages as follows:

$$\begin{array}{c}
 \mathcal{L}_{re} , \text{ recursively enumerable languages} \\
 \subset \\
 \mathcal{L}_{rec} , \text{ recursive languages} \\
 \subset \\
 \mathcal{L}_{cs} , \text{ context sensitive languages} \\
 \subset \\
 \mathcal{L}_{cf} , \text{ context free languages} \\
 \subset \\
 \mathcal{L}_{fs} , \text{ finite state languages} \\
 \subset \\
 \mathcal{L}_{fin} , \text{ finite languages}
 \end{array}$$



Many more details and proofs of all of these results can be found in, for example, Salomaa (1973), Harrison (1978), Hopcroft and Ullman (1979). The early Chomsky papers (Chomsky, 1963; Miller and Chomsky, 1963) are dated, but still interesting.

2.6 Enumerations

(12) An enumeration is a sequence, a list, a function whose domain is the positive integers. A finite sequence is a function whose domain is $\{0, 1, \dots, n\}$ for some positive integer n .

(13) An enumerable set is one whose members can be enumerated.¹ ...By courtesy, we regard as enumerable the empty set, \emptyset , which has no members.

Sometimes an enumerable set is called “denumerable.”

(14) For any vocabulary Σ (i.e. for any finite, nonempty set), Σ^* is enumerable.

There are many enumerations of Σ^* . Let’s define one.

Let $\Sigma^0 = \{\epsilon\}$.

Let $\Sigma^1 = \{\langle x \rangle \mid x \in \Sigma\}$.

For $k \geq 1$, let $\Sigma^{k+1} = \{\langle x_1, \dots, x_k, x_{k+1} \rangle \mid \langle x_1, \dots, x_k \rangle \in \Sigma^k \text{ and } x_{k+1} \in \Sigma\}$.

Notice that each of the sets $\Sigma^0, \Sigma^1, \dots$ is finite.

Define a total order $<$ on Σ (e.g. “alphabetical” or “numeric” order).

Then extend the order to Σ^* as follows: $x < y$ iff either

- a. $|x| < |y|$ or
- b. $|x| = |y|$ and there is some i such that for all $0 \leq j < i$, the j ’th element of x is the same as the j ’th element of y and the i ’th element of $x <$ the i ’th element of y .

To enumerate Σ^* in order, list all elements of each of $\Sigma^0, \Sigma^1, \dots$, in order.

Every element of Σ^* appears in this sequence at some finite point.

(15) The set of positive rational numbers is enumerable.

There are many other ways to get an enumeration, but here is one. The following matrix has all of the natural numbers in it:

1	2	3	4	5	...
1	2	3	4	5	...
1	2	3	4	5	...
1	2	3	4	5	...
1	2	3	4	5	...
...

And listing elements of the previous matrix in the the following order, we will get to each entry at some finite point:

1	2	6	7	15	...
3	5	8	14	...	
4	9	13	...		
10	12	...			
11	...				
...					

The fact that this enumeration lists each rational number infinitely often does not matter; the thing we require for this to count as an enumeration of the rationals is just that every rational number appear (at some finite position).

(16) Using the same method, we see that for any vocabularies Σ_1, Σ_2 , the set $\Sigma_1^* \times \Sigma_2^*$ is enumerable.

¹This is the first sentence of Boolos and Jeffrey (1980). The first chapters of this book are a good place to go for more background on enumerability. Other good places (though more advanced) are the first chapter of Kleene (1952) or part I of Smullyan (1993).

- (17) The set of all grammars is enumerable.

Once again, there are many ways to do this, but let's define one enumeration in a little bit of detail. Assume that the elements of Σ are alphabetic, and order them alphabetically.

The set Cat is, by definition of grammar, finite and disjoint from Σ . Assign a total order to this set as well, and order all of Cat before all of Σ .

We can then order $((\Sigma \cup Cat)^* \times Cat \times (\Sigma \cup Cat)^*)$ in just the way we ordered Σ in (14) – these are the left sides of the rules of rewrite grammars.

We similarly order $(\Sigma \cup Cat)^*$ – the right sides of the rewrite rules.

Since rewrite rules are nothing more than left-side/right-side pairs, we can now use the construction in (15) to enumerate them all. Assuming that the categories Cat , vocabulary Σ and start symbol S are given, we can then enumerate grammars with 2 symbols, grammars with 3 symbols, etc, to get all grammars.²

- (18) The set of all sets of positive integers is not enumerable. (Cantor)

Proof: Suppose $L = \langle s_1, s_2, s_3, \dots \rangle$ is any list of sets of positive integers.

For any such list L , we define another set $nondiag(L) = \{n \mid n \notin s_n\}$.

(For example, if s_3 happens to be the even numbers, then $3 \in nondiag(L)$.)

Suppose (for contradiction) that L is a list of all sets of integers. Then for some i , $nondiag(L) = s_i$.

By the definition of $nondiag(L)$, $i \in nondiag(L)$ iff $i \notin s_i$.

But then since $nondiag(L) = s_i$, $i \in nondiag(L)$ iff $i \notin nondiag(L)$. ζ

- (19) The set of all languages is not enumerable.

2.7 Single value languages

- (20) In learnability theory, we often think of languages as sets of strings, and texts as sequences of strings, but the previous section shows that many other learning problems will have a structure which is similar in the relevant respects.

- (21) Consider the “language” which is a set of pairs of strings:

$$\langle x, y \rangle.$$

For example, if Σ had symbols to represent phonetic structures (PF) and logical forms (LF), we could have a “language” which was a set of PF-LF pairs.

Definition 2 \mathcal{L}_{sv} is the collection of r.e. languages L such that L is a function. That is, a set L of pairs is in \mathcal{L}_{sv} if and only if each first element of a pair is paired with a unique second element. These languages are “single valued.”

\mathcal{L}_{svt} is the collection of “single valued” r.e. languages which are “total” in the sense that every string occurs as a first element of a pair.

- (22) Assuming that the languages in \mathcal{L}_{svt} are r.e. means that each of these languages has a grammar, a grammar in which the expressions derived code pairs of expressions. This coding of pairs can be done in many ways: one way is just to designate a special symbol $*$ (in a vocabulary of size >2) that separates the left expression s from the right expression t in derived sentences $s * t$.

Furthermore, the grammars generating expressions of this form are enumerable. For example, take our earlier enumeration of all grammars and remove all the grammars that do not conform. (Of course, this is not a computable procedure.)

²We could let Cat and Σ be infinite enumerable sets too, and still enumerate all grammars, again using a variant of the strategy in (15).

- (23) Notice that the languages in \mathcal{L}_{svt} have the following nice property: if two languages $L, L' \in \mathcal{L}_{svt}$ differ, then there is at least one first element s such that $s * t \in L$ (for some t) but $s * t' \in L'$ where $t \neq t'$. Given this fact about \mathcal{L}_{svt} , it is easy to establish the following result:

Theorem 5 \mathcal{L}_{svt} is identifiable.

Proof: Let $\phi(T[i]) = G$ where G is the first grammar in the enumeration such that $L(G) \subseteq \text{content}(T[i])$.

Since each language L in \mathcal{L}_{svt} is represented by a machine that appears at some point in the enumeration, and this function will differ from all others at some point, we are guaranteed that in any text T there is a finite point i at which $\phi(T[i])$ will be the correct guess, and the guess will never change after that. \square

This is another kind of “induction-by-enumeration.”³

2.8 Languages defined by constraints

The learner ϕ_e just keeps a record of what has been seen so far. Since many linguistic theories make use of constraints of various kinds, it is an interesting to consider learners that pay attention to what has not been seen so far. Let’s define some simple language classes based on ruling out some of the possible structures, letting $|L|$ represent the number of elements in L :

Definition 3 Let \mathcal{L}_{co-1} be the class of languages $\{\Sigma^* - \{x\} \mid x \in \Sigma^*\}$.

For any number $k \geq 0$, let \mathcal{L}_{co-k} be the class of languages $\{\Sigma^* - L \mid |L| = k\}$.

Let \mathcal{L}_{co-fin} be the class of languages $\{\Sigma^* - L \mid L \in \mathcal{L}_{fin}\}$.

Notice that, like \mathcal{L}_{fin} , all of these classes are infinite (except for \mathcal{L}_{co-0} which is a singleton set, $\{\Sigma^*\}$). But, unlike \mathcal{L}_{fin} , every member of every one of these classes is an infinite language.

A grammar for a language in \mathcal{L}_{co-1} could be regarded as a “filter” represented by a rule of the form

$$*x$$

where x is any string. A grammar for any language in one of the classes \mathcal{L}_{co-k} could be given by exactly k filters of this kind. And a grammar in \mathcal{L}_{co-fin} would have any number of filters of this kind. That is, $\mathcal{L}_{co-fin} = \bigcup_{k \geq 0} \mathcal{L}_{co-k}$.

Theorem 6 \mathcal{L}_{co-1} is identifiable.

Proof: Consider any enumeration of Σ^* , and let ϕ be defined as follows. For any sequence of strings $T[i]$,

$$\phi(t_i) = *x$$

where x is the first string in the enumeration that does not occur in $T[i]$.

Now we just need to show that, given any text T for any language $L \in \mathcal{L}_{co-1}$, ϕ identifies T . Since $L \in \mathcal{L}_{co-1}$, L is defined by some filter $*x$. Since there are just finitely many elements that occur before x , there will be some finite point i such that $T[i]$ contains all of those elements. At that point, by the definition of ϕ , $\phi(T[i]) = *x$. Furthermore, it is easy to see that at all later points $j > i$, $\phi(T[j]) = *x$ because x will never occur in the text. \square

The previous result generalizes immediately:

Theorem 7 For any k , \mathcal{L}_{co-k} is identifiable.

³It turns out that the collection of all “single valued” languages, \mathcal{L}_{sv} , is not identifiable Osherson, Weinstein, and Stob (1986b, Cor.2.5A). Totality guarantees that the texts will contain, at some finite point, values that distinguish any two functions that differ; in \mathcal{L}_{sv} we have no such guarantee, because the relevant values might be undefined.

One learner for this class is the function that, at each point, conjectures that the first k elements of Σ^* not seen so far are the ones excluded.

The previous results do not generalize to \mathcal{L}_{co-fin} , though. In fact, we can show:

Conjecture 1 The class $\mathcal{L}_{co-1} \cup \mathcal{L}_{co-2}$ is not identifiable.

This claim is intuitive: at each point in a text, the learner has no way of knowing whether one or two elements are excluded. In particular, suppose a learner guessed correctly in some case that the language was defined by $*x$. The problem is that if from there on the text were really one for the co-2 language $*x, *y$, the learner would not see the error. To prove this, though, requires that we show that the learner would never notice the mistake, and that is slightly tricky. To show this, the result in the following section will be extremely useful.

2.9 Exercises

Do the following exercises from Osherson, Weinstein, and Stob (1986b).

1. Suppose that some particular r.e. language L is given. Show that the class of languages $\{L \cup D \mid D \in \mathcal{L}_{fin}\}$ is learnable.
2. Let $\mathcal{L} = \{L, L'\}$ be a set of r.e. languages such that $L \subset L'$. Prove that no self-monitoring learner identifies \mathcal{L} .
3. We saw that \mathcal{L}_{co-1} is identifiable. Show that no self-monitoring learner identifies \mathcal{L}_{co-1} .

2.10 Locking sequences

Here we introduce a notion that is very useful for reasoning about how learners behave on texts: the notion of a locking sequence for a learner ϕ on a language L . This is a finite sequence such that no extension of that sequence with elements of L can make ϕ change its mind. That is, it is not merely that none of the following elements *in a given text* make ϕ change, but *no* elements of L can make ϕ change. Surprisingly, whenever ϕ identifies L , there is a locking sequence of this kind. We make this idea precise as follows:

Definition 4 Given a learner ϕ that identifies a language L , a sequence t is said to be a **locking sequence for ϕ and L** iff

1. t is a finite sequence of expressions and $\#$,
2. $content(t) \subseteq L$,
3. $L(\phi(t)) = L$, and
4. for every finite sequence s such that $content(ts) \subseteq L$, $\phi(ts) = \phi(t)$.

Theorem 8 (Blum and Blum, 1975) If L is any (recursively enumerable) language and ϕ identifies L , then there is a locking sequence t for ϕ and L .

Proof: Let L be any (recursively enumerable) language that ϕ identifies. For contradiction, assume that for every finite t such that $content(t) \subseteq L$ and $L(\phi(t)) = L$, there is a finite sequence s such that $content(ts) \subseteq L$ but $\phi(ts) \neq \phi(t)$.

We will deduce a contradiction by constructing a text r for L on which ϕ does not converge.

Let $u = x_0x_1\dots$ be any text such that $content(u) = L$. Since ϕ identifies u , there is some finite u_i such that $L(\phi(u_i)) = L$. We now define a set of sequences $R = \{r^0, r^1, \dots\}$, where each r^{i+1} extends r^i :

$$r^0 = u_i$$

$$r^{i+1} \begin{cases} = r^i x_i & \text{if } L(\phi(r^i)) \neq L \\ = r^i s x_i & \text{otherwise, where } s \text{ is finite, } L(s) \subseteq L, \text{ and } \phi(r^i s) \neq \phi(r^i) \end{cases}$$

We know that, at each step, the s mentioned in this definition exists by our assumption. Notice that for all $i < j$, r^i is a prefix of r^j . Also notice that every r^i is at least i elements long. So we can define a text $r = y_1 y_2 \dots$ as follows: for all $i > 0$, y_i is the i 'th element of r^i . Notice that $\text{content}(r) = L$, since r contains every element of u and all of the sequences s are such that $\text{content}(s) \subseteq L$. However, ϕ does not converge on r , because for every $i \geq 0$, either $L(\phi(r_i)) \neq L$ or for some $j > i$, $\phi(r_i) \neq \phi(r_j)$. ζ □

2.11 Languages defined by constraints, part 2

Locking sequences provide exactly what we need to get an easy proof of our earlier Conjecture 1:

Theorem 9 The class $\mathcal{L}_{co-1} \cup \mathcal{L}_{co-2}$ is not identifiable.

Proof: Suppose for contradiction that this class is identifiable. Then there is some learner ϕ that identifies the class. Consider any $L \in \mathcal{L}_{co-1}$. By Theorem 8 there is a locking sequence t_i for L and ϕ . Since L is infinite and $\text{content}(t_i)$ is finite, there is an $x \in L - L(t_i)$. Consider any text T' for the language $L' = (L - \{x\}) \in \mathcal{L}_{co-2}$. Since $\text{content}(T') \subseteq L$, and since t_i is a locking sequence for ϕ and L , $\phi(t_i T')$ converges on a grammar for L , but this is incorrect since $L \neq \text{content}(t_i T') = (L - \{x\})$. So ϕ does not identify this text, so it does not identify $(L - \{x\})$. ζ □

We can immediately draw some more general conclusions, using the general and obvious fact:

Theorem 10 If \mathcal{L} is not identifiable, no superset of \mathcal{L} is identifiable.

Proof: If \mathcal{L}' is identifiable, then there is some ϕ which identifies every language in \mathcal{L}' . Hence it identifies every language in \mathcal{L} . ζ □

With this result, we have,

Corollary 1 \mathcal{L}_{co-fin} is not identifiable.

We can now make a number of connections to traditional formal language theory, because of the following simple fact:

Theorem 11 $\mathcal{L}_{co-fin} \subset \mathcal{L}_{fs}$

Proof: This follows immediately from the following: (i) that every finite language is finite state, (ii) Σ^* is finite state, and (iii) if $L_1, L_2 \in \mathcal{L}_{fs}$ then $L_1 - L_2 \in \mathcal{L}_{fs}$.⁴ □

Corollary 2 (Gold, 1967)

The class of finite state languages \mathcal{L}_{fs} is not identifiable.

The class of context free languages \mathcal{L}_{cf} is not identifiable.

The class of context sensitive languages \mathcal{L}_{cs} is not identifiable.

The set of recursive languages \mathcal{L}_{rec} is not identifiable.

The set of recursively enumerable languages \mathcal{L}_{re} is not identifiable.

We also have the general result:

Theorem 12 It can happen that the union of two identifiable classes fails to be identifiable.

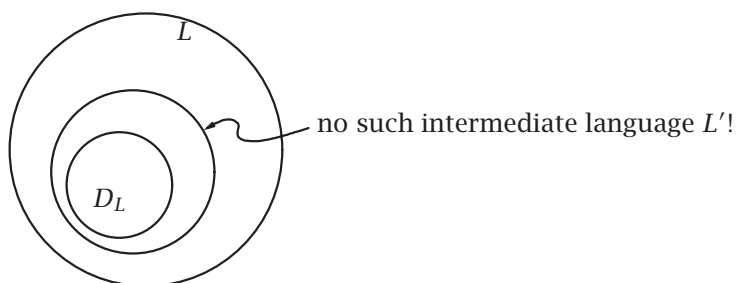
⁴These results are presented in any introductory text on formal language theory, such as Hopcroft and Ullman (1979) or Harrison (1978). We will come back to look at finite state languages and some of the other classes more carefully later.

2.12 Angluin's subset theorem

Various theorems about special cases appeared before Angluin homed in on the following very general "subset theorem."⁵

Theorem 13 (Angluin, 1980) Let \mathcal{L} be a collection of (recursively enumerable) languages. Then \mathcal{L} is identifiable iff for all $L \in \mathcal{L}$ there is a finite distinguished subset $D_L \subseteq L$ such that for all $L' \in \mathcal{L}$, if $D_L \subseteq L'$ then $L' \not\subset L$.

Angluin's theorem tells us that a collection of languages is identifiable just in case, for every language L in the collection, you can find a finite subset D_L such that no language L' that includes D_L is properly included in L :



Proof: (\Rightarrow) Suppose \mathcal{L} is identified by ϕ . Then by Theorem 8, for any $L \in \mathcal{L}$, there is a locking sequence t for ϕ and L . Since locking sequences are finite, so is $\text{content}(t)$. This finite subset of L will distinguish it: that is, we show that for all $L' \in \mathcal{L}$, if $\text{content}(t) \subseteq L'$, then $L' \not\subset L$.

Assume for contradiction that this is not so, that there is some $L' \in \mathcal{L}$ such that $\text{content}(t) \subseteq L'$ and $L' \subset L$. Then there is some text ts extending t such that $\text{content}(ts) = L'$, but since t is a locking sequence for ϕ and L , and since $\text{content}(ts') \subseteq L' \subset L$, $\phi(ts') = \phi(t)$ for any $s' = s_i$, $i \geq 0$. Thus ϕ fails to identify ts and hence fails to identify L' . ζ

(\Leftarrow) Assume that for every $L \in \mathcal{L}$, there is a finite $D_L \subseteq L$ such that for all $L' \in \mathcal{L}$, $D_L \subseteq L'$ implies $L' \not\subset L$. Suppose that we have an enumeration of all possible grammars. For each $L \in \mathcal{L}$, we fix a particular finite $D_L \subseteq L$ such that for all $L' \in \mathcal{L}$, if $D_L \subseteq L'$ then $L' \not\subset L$.

Then for every text T , define $\phi(T[i])$ to be the first grammar G in the enumeration such that $D_{L(G)} \subseteq \text{content}(T[i]) \subseteq L(G)$, if there is one. Otherwise, let $\phi(T[i])$ be the first grammar in the enumeration.

Now we show that this ϕ identifies \mathcal{L} . Consider any $L(G) \in \mathcal{L}$ where G is the first grammar in the enumeration that generates $L(G)$, and consider any text T such that $\text{content}(T) = L(G)$. Since T is a text for $L(G)$, the assumption tells us that there is a finite i such that:

- a. $D_{L(G)} \subseteq \text{content}(t_i)$, and
- b. if $G' < G$, $L(G') \in \mathcal{L}$, and $L \not\subset L(G')$, then $\text{content}(T[i]) \not\subseteq L(G')$.

We know that there is such an i because, first, $D_{L(G)}$ is finite. And second, the number of $G' < G$ such that $L(G') \in \mathcal{L}$ and $L \not\subset L(G')$ is finite. Since for each such G' there is some string s' such that $s' \in L(G')$ and $s' \notin L$, we can let i be high enough that t_i contains such a string for each such G' .

It is now easy to see that ϕ will converge on the correct grammar at such a point i . By a and the definition of ϕ , for all $j \geq i$, $\phi(T[j]) = G$ unless there is a $G' < G$ such that $L(G') \in \mathcal{L}$, and $D_{L(G')} \subseteq \text{content}(T[j]) \subseteq L(G')$. But clearly, by the choice of i , $D_{L(G)} \subseteq \text{content}(T[j]) \subseteq L(G)$, and by b, there is no earlier grammar $G' < G$ with these properties, so $\phi(T[j]) = G$. \square

⁵We follow Osherson, Weinstein, and Stob (1986b) in presenting a simple version of Angluin's result which does not establish the existence of a computable learner, but only of a learner. de Jongh and Kanazawa (1996) call this a "watered down" version of Angluin's result, because it characterizes learnability rather than effective learnability. See this paper or Angluin's original paper for the stronger result.

This main theorem can be used to establish both positive and negative results of special interest. The following theorem from Gold (1967) was given earlier as an exercise (maybe a rather challenging one), but now it can be proven as a corollary to Angluin's theorem.

Corollary 3 Any finite class of languages is identifiable.

Proof: Consider any finite collection of languages L_1, L_2, \dots, L_n , and assume that they are here listed in such a way that if $j < i$, then $L_i \not\subseteq L_j$.

Now, consider each L_i in turn. For each earlier language, L_j , $j < i$, let $x_{i,j}$ be some element that is in L_i but not in L_j . We define $D_{L_i} = \{x_{i,j} \mid j < i\}$.

Obviously, for each of the finitely many languages L_i , the corresponding subset will only have i elements in it – that is, finitely many!

Now suppose that there is some language L_k in our collection such that $D_{L_i} \subseteq L_k$. Clearly in this case, L_k is not one of the languages that appear earlier than L_i in the enumeration, and so it follows by our definition of the enumeration that $L_k \not\subseteq L_i$. So for each L_i we have a subset D_{L_i} as specified in Angluin's Theorem 13, and so the collection of languages is identifiable. \square

Another way to see that any finite collection L_1, L_2, \dots, L_n is identifiable is to specify an adequate learner directly. Assume again that if $j < i$, then $L_i \not\subseteq L_j$, and assume that these languages are defined by grammars G_1, G_2, \dots, G_n , respectively. Define ϕ so that for any t_i , $\phi(T[i])$ is just the first grammar G such that $\text{content}(T[i]) \subseteq L(G)$. This learner identifies the class.

The proof of Corollary 3 actually gives us a more general result:

Corollary 4 Consider any class \mathcal{L} that can be enumerated L_0, L_1, \dots in such a way that if $j < i$, then $L_i \not\subseteq L_j$. \mathcal{L} is identifiable.

We also derive some interesting negative results:

Corollary 5 Let \mathcal{L} be a collection of (recursively enumerable) languages such that

- i. \mathcal{L} contains an infinite language L_∞ , and
- ii. for every $L_0 \subset L_\infty$ there is some $L \in \mathcal{L}$ such that $L_0 \subseteq L \subset L_\infty$.

Then \mathcal{L} is not identifiable.

Proof: Assume \mathcal{L} satisfies (i) and (ii). By (ii), every finite subset L_0 of L_∞ is such that for some $L \in \mathcal{L}$, $L_0 \subseteq L \subset L_\infty$, and so Theorem 13 applies immediately. \square

Corollary 6 (Gold, 1967) No strict superset of \mathcal{L}_{fin} is identifiable.