The Complexity of Learning SUBSEQ($\boldsymbol{A}$)

Stephen Fenner[1], William Gasarch[2], and Brian Postow[3]

[1] University of South Carolina[†]
[2] University of Maryland at College Park[‡]
[3] Union College[§]

Abstract. Higman showed[1] that if A is *any* language then SUBSEQ(A) is regular, where SUBSEQ(A) is the language of all subsequences of strings in A. Let $s_1, s_2, s_3, \ldots$ be the standard lexicographic enumeration of all strings over some finite alphabet. We consider the following inductive inference problem: given $A(s_1), A(s_2), A(s_3), \ldots$, learn, in the limit, a DFA for SUBSEQ(A). We consider this model of learning and the variants of it that are usually studied in inductive inference: anomalies, mind-changes, teams, and combinations thereof.
This paper is a significant revision and expansion of an earlier conference version [6].

1 Introduction

In Inductive Inference [2, 4, 16] the basic model of learning is as follows.

Definition 1.1. A class $\mathcal{A}$ of decidable sets of strings[2] is in EX if there is a Turing machine M (the learner) such that if M is given $A(\varepsilon)$, $A(0)$, $A(1)$, $A(00)$, $A(01)$, $A(10)$, $A(11)$, $A(000)$, $\ldots$, where $A \in \mathcal{A}$, then M will output $e_1, e_2, e_3, \ldots$ such that $\lim_s e_s = e$ and e is an index for a Turing machine that decides A.

Note that the set A must be computable and the learner learns a Turing machine index for it. There are variants [1, 12, 14] where the set need not be computable and the learner learns something about the set (e.g., "Is it infinite?" or some other question).

Our work is based on a remarkable theorem of Higman's, [17][3] given below as Theorem 1.4.
Convention: Σ is a finite alphabet.

Definition 1.2. Let $x, y \in \Sigma^*$. We say that x is a *subsequence* of y if $x = x_1 \cdots x_n$ and $y \in \Sigma^* x_1 \Sigma^* x_2 \cdots x_{n-1} \Sigma^* x_n \Sigma^*$. We denote this by $x \preceq y$.

Notation 1.3. If A is a set of strings, then SUBSEQ(A) is the set of subsequences of strings in A.

Higman [17] showed the following using well-quasi-order theory.

[†] Department of Computer Science and Engineering, Columbia, SC 29208. `fenner@cse.sc.edu`. Partially supported by NSF grant CCF-05-15269.
[‡] Department of Computer Science and Institute for Advanced Computer Studies, College Park, MD 20742. `gasarch@cs.umd.edu`. Partially supported by NSF grant CCR-01-05413.
[§] Department of Computer Science, Schenectady, NY 12305. `postow@acm.org`.
[1] The result we attribute to Higman is actually an easy consequence of his work. See [7] for more discussion.
[2] The basic model is usually described in terms of learning computable functions; however, virtually all of the results hold in the setting of decidable sets.
[3] See footnote 1.

Theorem 1.4 (Higman [17]). *If A is any language over Σ^*, then* SUBSEQ(A) *is regular. In fact, for any language A there is a unique minimum (and finite) set S of strings such that*

$$\mathrm{SUBSEQ}(A) = \{x \in \Sigma^* : (\forall z \in S)[z \not\preceq x]\}. \tag{1}$$

Note that A is *any language whatsoever*. Hence we can investigate the following learning problem.

Notation 1.5. We let $s_1, s_2, s_3, \ldots$ be the standard length-first lexicographic enumeration of Σ^*. We refer to Turing machines as TMs.

Definition 1.6. A class $\mathcal{A}$ of sets of strings in Σ^* is in SUBSEQ-EX if there is a TM M (the learner) such that if M is given $A(s_1), A(s_2), A(s_3), \ldots$ where $A \in \mathcal{A}$, then M will output $e_1, e_2, e_3, \ldots$ such that $\lim_s e_s = e$ and e is an index for a DFA that recognizes SUBSEQ(A). It is easy to see that we can take e to be the least index of the minimum-state DFA that recognizes SUBSEQ(A). Formally, we will refer to $A(s_1)A(s_2)A(s_3)\cdots$ as being on an auxiliary tape.

Notation 1.7. For any k we let F_k denote the DFA with index k. (See also Notation 2.8.)

We give examples of elements of SUBSEQ-EX. Additional examples are given in Section 4.

Definition 1.8. *$\mathcal{F}$ is the set of all finite sets of strings.*

Proposition 1.9. $\mathcal{F} \in$ SUBSEQ-EX.

Proof. Let M be a learner that, when $A \in \mathcal{F}$ is on the tape, outputs $k_1, k_2, \ldots$, where each k_i is the index of a DFA that recognizes SUBSEQ($A \cap \Sigma^{\le i}$). Clearly, M learns SUBSEQ(A). ☐

More generally, we have

Proposition 1.10. REG $\in$ SUBSEQ-EX.

Proof. When A is on the tape, for $n = 0, 1, 2, \ldots$, the learner M

1. finds the least k such that $A \cap \Sigma^{<n} = L(F_k) \cap \Sigma^{<n}$, then
2. outputs the least ℓ such that $L(F_\ell) = \mathrm{SUBSEQ}(L(F_k))$ (see Proposition 2.7(1)).

If A is regular, then clearly M will converge to the least k such that $A = L(F_k)$, whence M will converge to the least ℓ such that $L(F_\ell) = \mathrm{SUBSEQ}(A)$. ☐

This problem is part of a general theme of research: given a language A, rather than try to learn a program for it (which is not possible if A is undecidable) learn some aspect of it. In this case we learn SUBSEQ(A). Note that we learn SUBSEQ(A) in a very strong way in that we have a DFA for it.

If $\mathcal{A} \in$ EX, then a TM can infer a Turing index for any $A \in \mathcal{A}$. The index is useful if you want to determine membership of particular strings, but not useful if you want most global properties (e.g., "Is A infinite?"). If $\mathcal{A} \in$ SUBSEQ-EX, then a TM can infer a DFA for SUBSEQ(A). The index is useful if you want to determine virtually any property of SUBSEQ(A) (e.g., "Is SUBSEQ(A) infinite?") but not useful if you want to answer almost any question about A.

We look at anomalies, mind-changes, and teams, both alone and in combination. These are standard variants of the usual model in inductive inference. See [4] and [21] for the definitions within inductive inference; however, our definitions are similar.

We list definitions and our main results.

1. Let $\mathcal{A} \in \text{SUBSEQ-EX}^a$ mean that the final DFA may be wrong on at most a strings (called *anomalies*). Also let $\mathcal{A} \in \text{SUBSEQ-EX}^*$ mean that the final DFA may be wrong on a finite number of strings (i.e., a finite number of anomalies—the number perhaps varying with A). The anomaly hierarchy collapses; that is,
$$\text{SUBSEQ-EX} = \text{SUBSEQ-EX}^*.$$
This contrasts sharply with the case of EX^a, where it was proven in [4] that $\text{EX}^a \subset \text{EX}^{a+1}$.
2. Let $\mathcal{A} \in \text{SUBSEQ-EX}_n$ mean that the TM makes at most $n+1$ conjectures (and hence changes its mind at most n times). The mind-change hierarchy separates; that is, for all n,
$$\text{SUBSEQ-EX}_n \subset \text{SUBSEQ-EX}_{n+1}.$$
This is analogous to the result proved in [4].
3. The mind-change hierarchy also separates if you allow a transfinite number of mind-changes, up to ω_1^{CK} (see "Transfinite Mind Changes and Procrastination" in Section 3.3). This is also analogous to the result in [9].
4. Let $\mathcal{A} \in [a,b]\text{SUBSEQ-EX}$ mean that there is a team of b TMs trying to learn the DFA, and we demand that at least a of them succeed (it may be a different a machines for different $A \in \mathcal{A}$).
   (a) If $1 \le a \le b$ and $q = \lfloor b/a \rfloor$, then
   $$[a,b]\text{SUBSEQ-EX} = [1,q]\text{SUBSEQ-EX}.$$
   Hence we need only look at team learning classes of the form $[1,n]\text{SUBSEQ-EX}$.
   (b) The team hierarchy separates. That is, for all b,
   $$[1,b]\text{SUBSEQ-EX} \subset [1,b+1]\text{SUBSEQ-EX}.$$
   These are also analogous to results from [15].
5. The anomaly hierarchy collapses in the presence of teams. That is, for all $1 \le a \le b$,
$$[a,b]\text{SUBSEQ-EX}^* = [a,b]\text{SUBSEQ-EX}.$$
6. There are no trade-offs between bounded anomalies and mind-changes: for all a and c,
$$\text{SUBSEQ-EX}_c^a = \text{SUBSEQ-EX}_c.$$
However, $\text{SUBSEQ-EX}_0^* \not\subseteq \text{SUBSEQ-EX}_c$ and $\text{SUBSEQ-EX}_c \not\subseteq \text{SUBSEQ-EX}_{c-1}^*$ for any $c > 0$. There *are* nontrivial trade-offs if we consider anomaly revisions (transfinite anomalies) versus mind-changes.
7. There are several interesting trade-offs between mind-changes and teams. For all $1 \le a \le b$ and $c \ge 0$,
$$[a,b]\text{SUBSEQ-EX}_c \subseteq [1,\lfloor b/a \rfloor]\text{SUBSEQ-EX}_{b(c+1)-1}$$
and $[1,q]\text{SUBSEQ-EX}_c \subseteq [a,aq]\text{SUBSEQ-EX}_c$ for $q \ge 1$. Also,
$$\text{SUBSEQ-EX}_{b(c+1)-1} \subseteq [1,b]\text{SUBSEQ-EX}_c \not\supseteq \text{SUBSEQ-EX}_{b(c+1)}.$$
Finally, if $b > 1$ and $c \ge 1$, then
$$\text{SUBSEQ-EX}_{2b(c+1)-3} \supseteq [1,b]\text{SUBSEQ-EX}_c \not\subseteq \text{SUBSEQ-EX}_{2b(c+1)-4}.$$

Note 1.11. PEX [3, 4] is like EX except that the conjectures must be for total TMs. The class SUBSEQ-EX is similar in that all the machines are total (in fact, DFAs) but different in that we learn the subsequence language, and the input need not be computable. The anomaly hierarchy for SUBSEQ-EX collapses just as it does for PEX; however, the team hierarchy for SUBSEQ-EX is proper, unlike for PEX.

2 Definitions

2.1 Definitions about subsequences

Notation 2.1. We let $\mathbb{N} = \{0, 1, 2, \ldots\}$ and $\mathbb{N}^+ = \{1, 2, 3, \ldots\}$. We assume that Σ is some finite alphabet, that $0, 1 \in \Sigma$, and that all languages are subsets of Σ^*. We identify a language with its characteristic function.

Notation 2.2. For $n \in \mathbb{N}$, we let $\Sigma^{=n}$ denote the set of all strings over Σ of length n. We also define $\Sigma^{\leq n} = \bigcup_{i \leq n} \Sigma^{=i}$ and $\Sigma^{<n} = \bigcup_{i<n} \Sigma^{=i}$. $\Sigma^{\geq n}$ and $\Sigma^{>n}$ are defined analogously.

Notation 2.3. Given a language A, we call the unique minimum set S satisfying (1) the *obstruction set* of A and denote it by $os(A)$. In this case, we also say that S *obstructs* A.

The following facts are obvious:

- The $\preceq$ relation is computable.
- For every string x there are finitely many $y \preceq x$, and given x one can compute a canonical index (see Notation 2.8) for the set of all such y.
- By various facts from automata theory, including the Myhill-Nerode minimization theorem: given a DFA, NFA, or regular expression for a language A, one can effectively compute the unique minimum state DFA recognizing A. (The minimum state DFA is given in some canonical form.)
- Given DFAs F and G, one can effectively compute DFAs for $\overline{L(F)}$, $L(F) \cup L(G)$, $L(F) \cap L(G)$, $L(F) - L(G)$, and $L(F) \triangle L(G)$ (symmetric difference). One can also effectively determine whether or not $L(F) = \emptyset$ and whether or not $L(F)$ is finite. If $L(F)$ is finite, then one can effectively find a canonical index for $L(F)$.
- For any language A, the set SUBSEQ(A) is completely determined by $os(A)$, and in fact, $os(A) = os(\text{SUBSEQ}(A))$.
- The strings in the obstruction set of a language must be pairwise $\preceq$-incomparable (i.e., the obstruction set is an $\preceq$-antichain). Conversely, any $\preceq$-antichain obstructs some language.
  For any $S \subseteq \Sigma^*$ define

$$\text{ObsBy}(S) = \{x \in \Sigma^* : (\forall z \in S)[z \not\preceq x]\}.$$

The term ObsBy(S) is an abbreviation for 'obstructed by S'. Note that $os(\text{ObsBy}(S)) \subseteq S$, and equality holds iff S is an $\preceq$-antichain.

Definition 2.4. A language $A \subseteq \Sigma^*$ is $\preceq$*-closed* if SUBSEQ$(A) = A$.

Observation 2.5. A language A is $\preceq$-closed if and only if there exists a language B such that $A = \text{SUBSEQ}(B)$.

Observation 2.6. Any infinite $\preceq$-closed set contains strings of every length.

The next proposition implies that finding $os(A)$ is computationally equivalent to finding a DFA for SUBSEQ(A).

Proposition 2.7. *The following tasks are computable:*

1. *Given a DFA F, find a DFA G such that $L(G) = \mathrm{SUBSEQ}(L(F))$.*
2. *Given the canonical index of a finite language $D \subseteq \Sigma^*$, compute a regular expression for (and hence the minimum-state DFA recognizing) the language* $\mathrm{ObsBy}(D) = \{x \in \Sigma^* : (\forall z \in D)[z \not\preceq x]\}$.
3. *Given a DFA F, decide whether or not $L(F)$ is $\preceq$-closed.*
4. *Given a DFA F, compute the canonical index of $os(L(F))$.*

Proof. We prove the fourth item and leave the first three as exercises for the reader.

Given DFA F, first compute the DFA G of Item 1. Since $os(A) = os(\mathrm{SUBSEQ}(A))$ for all languages A, it suffices to find $os(L(G))$.

Suppose that G has n states.

We claim that every element of $os(L(G))$ has length less than n. Assume otherwise, i.e., that there is some string $w \in os(L(G))$ with $|w| \geq n$. Then $w \notin L(G)$, and as in the proof of the Pumping Lemma, there are strings $x, y, z \in \Sigma^*$ such that $w = xyz$, $|y| > 0$, and $xy^iz \notin L(G)$ for all $i \geq 0$. In particular, $xz \notin L(G)$. But $xz \preceq w$ and $xz \neq w$, which contradicts the assumption that w was a $\preceq$-minimal string in $\overline{L(G)}$. This establishes the claim.

By the claim, in order to find $os(L(G))$, we just need to check each string of length less than n to see whether it is a $\preceq$-minimal string rejected by G. □

2.2 Classes of languages

We define classes of languages via the types of machines that recognize them.

Notation 2.8.

1. $D_1, D_2, \ldots$ is a standard enumeration of finite languages. (e is the *canonical index* of D_e.)
2. $F_1, F_2, \ldots$ is a standard enumeration of minimized DFAs, presented in some canonical form so that for all $i \neq j$ we have $L(F_i) \neq L(F_j)$. Let $\mathrm{REG} = \{L(F_1), L(F_2), \ldots\}$.
3. $P_1, P_2, \ldots$ is a standard enumeration of $\{0,1\}$-valued polynomial-time TMs. Let $\mathrm{P} = \{L(P_1), L(P_2), \ldots\}$. Note that these are total.
4. $M_1, M_2, \ldots$ is a standard enumeration of Turing Machines. We let $\mathrm{CE} = \{L(M_1), L(M_2), \ldots\}$, where $L(M_i)$ is the set of all x such that $M_i(x)$ halts with output 1 (i.e., $M_i(x)$ *accepts*). CE stands for "computably enumerable."[4]
5. We let $\mathrm{DEC} = \{L(N) : N \text{ is a total TM}\}$.

The notation below is mostly standard. For the notation that relates to computability theory, our reference is [22].

For separation results, we will often construct tally sets, i.e., subsets of 0^*.

Notation 2.9.

1. The empty string is denoted by ε.
2. For $m \in \mathbb{N}$, we define $0^{<m} = \{0^i : i < m\}$.
3. If $B \subseteq 0^*$ is finite, we let $m(B)$ denote the least m such that $B \subseteq 0^{<m}$, and we observe that $\mathrm{SUBSEQ}(B) = 0^{<m(B)}$.
4. If A is a set then $\mathcal{P}(A)$ is the powerset of A.

[4] These sets are also called, "recursively enumerable."

Notation 2.10. If $B, C \subseteq 0^*$ and B is finite, we define a "shifted join" of B and C as follows:

$$B \cup\!\!+ C = \{0^{2n+1} : 0^n \in B\} \cup \{\, 0^{2(m(B)+n)} : 0^n \in C\}.$$

In $B \cup\!\!+ C$, all the elements from B have odd length and are shorter than the elements from C, which have even length. We define inverses to the $\cup\!\!+$ operator:

Notation 2.11. For every $m \geq 0$ and language A, let

$$\xi(A) := \{0^n : n \geq 0 \land 0^{2n+1} \in A\},$$
$$\pi(m; A) := \{0^n : n \geq 0 \land 0^{2(m+n)} \in A\}.$$

If $B, C \subseteq 0^*$ and B is finite, then we have $B = \xi(B \cup\!\!+ C)$ and $C = \pi(m(B); B \cup\!\!+ C)$.

Notation 2.12. For languages $A, B \subseteq \Sigma^*$, we write $A \subseteq^* B$ to mean that $A - B$ is finite.

The following family of languages will be used in several places:

Definition 2.13. For all i, let R_i be the language $(0^*1^*)^i$.

Note that $R_1 \subseteq R_2 \subseteq R_3 \subseteq \cdots$, but $R_{i+1} \not\subseteq^* R_i$ for any $i \geq 1$. Also note that $\text{SUBSEQ}(R_i) = R_i$ for all $i \geq 1$.

2.3 Variants on SUBSEQ-EX

In this section, we note some obvious inclusions among the variant notions of SUBSEQ-EX. We also define relativized SUBSEQ-EX.

Obviously,

$$\text{SUBSEQ-EX}_0 \subseteq \text{SUBSEQ-EX}_1 \subseteq \text{SUBSEQ-EX}_2 \subseteq \cdots \subseteq \text{SUBSEQ-EX}. \tag{2}$$

We will extend this definition into the transfinite later. Clearly,

$$\text{SUBSEQ-EX} = \text{SUBSEQ-EX}^0 \subseteq \text{SUBSEQ-EX}^1 \subseteq \cdots \subseteq \text{SUBSEQ-EX}^*. \tag{3}$$

Finally, it is evident that if $a \geq c$ and $b \leq d$, then $[a, b]\text{SUBSEQ-EX} \subseteq [c, d]\text{SUBSEQ-EX}$.

Definition 2.14. If $X \subseteq \mathbb{N}$, then SUBSEQ-EX^X is the same as SUBSEQ-EX except that we allow the learner to be an oracle TM using oracle X.

We may combine these variants in a large variety of ways.

3 Main results

3.1 Standard learning

We start with an example of something in SUBSEQ-EX that contains nonregular languages. We'll give more extreme examples in Section 4.

Definition 3.1. For all $i \in \mathbb{N}$, let

$$\mathcal{S}_i := \{A \subseteq \Sigma^* : |os(A)| = i\}.$$

Also let

$$\mathcal{S}_{\leq i} := \mathcal{S}_0 \cup \mathcal{S}_1 \cup \cdots \cup \mathcal{S}_i = \{A \subseteq \Sigma^* : |os(A)| \leq i\}.$$

Proposition 3.2. $\mathcal{S}_i \in$ SUBSEQ-EX *for all* $i \in \mathbb{N}$.

Proof. Given A on its tape, let M behave as follows, for $n = 0, 1, 2, \ldots$:

1. Compute $N = os(A \cap \Sigma^{\leq n}) \cap \Sigma^{\leq n}$.
2. If $|N| < i$, then go on to the next n.
3. Let $x_1, \ldots, x_i$ be the i shortest strings in N. If there is a tie, i.e., if there is more than one set of i shortest strings in N, then go on to the next n.
4. Output the least index k such that $L(F_k)$ is $\preceq$-closed and $os(L(F_k)) = \{x_1, \ldots, x_i\}$.

□

It was essentially shown in [7] that DEC $\notin$ SUBSEQ-EX. The proof there can be tweaked to show the stronger result that P $\notin$ SUBSEQ-EX. We include the stronger result here.

Theorem 3.3 ([7]). P $\notin$ SUBSEQ-EX. *In fact, there is a computable function g such that for all e, setting $A = L(P_{g(e)})$, we have $A \subseteq 0^*$ and* SUBSEQ(A) *is not learned by M_e.*

Proof. Assume, by way of contradiction, that P $\in$ SUBSEQ-EX via M_e. Then we effectively construct a machine N_e that implements the following recursive polynomial-time algorithm for computing A. Let j_0 be the unique index such that $L(F_{j_0}) = 0^*$.
On input x:

1. If $x \notin 0^*$ then reject. (This will ensure that $A \subseteq 0^*$.)
2. Let $x = 0^n$. Using no more than n computational steps, recursively run N_e on inputs $\varepsilon, 0, 00, \ldots, 0^{\ell_n - 1}$ to compute $A(\varepsilon)$, $A(0)$, $A(00)$, $\ldots$, $A(0^{\ell_n - 1})$, where $\ell_n \leq n$ is largest such that this can all be done within n steps. Set $R_n := A \cap 0^{<\ell_n}$.
3. Simulate M_e for $\ell_n - 1$ steps with R_n on its tape. If M_e does not output anything within this time, then reject. [Note that M_e only has time to scan its tape on cell corresponding to inputs $\varepsilon, 0, 00, \ldots, 0^{\ell_n - 1}$ (and perhaps some inputs not in 0^*).]
4. Let k be the most recent index output by M_e within $\ell_n - 1$ steps with R_n on its tape.
5. If $k = j_0$ (i.e., if $L(F_k) = 0^*$), then reject; else accept.

This algorithm runs in polynomial time for each fixed e, and thus $A = L(N_e) \in$ P. Further, given e we can effectively compute an index i such that $A = L(P_i)$. We let $g(e) = i$.

We note the following:

- It is clear that the sequence $\ell_0, \ell_1, \ell_2, \ldots$ is monotone and unbounded.
- When M_e is simulated in step 3, it behaves the same way with R_n on its tape as with A on its tape, because it does not run long enough to examine any place on the tape where R_n and A may differ.

We now show that M_e does not learn SUBSEQ(A). Assume otherwise, and let $k_1, k_2, \ldots$ be the sequence of outputs of M_e with A on the tape. By assumption, there is a $k' = \lim_{n\to\infty} k_n$ such that $L(F_{k'}) = \mathrm{SUBSEQ}(A)$. If $L(F_{k'}) = 0^*$, then for all large enough n, the algorithm rejects 0^n in Step 5, making A finite, which makes SUBSEQ(A) finite. If $L(F_{k'}) \neq 0^*$, then the algorithm accepts 0^n in Step 5 for all large enough n, making A infinite, which makes $\mathrm{SUBSEQ}(A) = 0^*$. In either case, $L(F_{k'}) \neq \mathrm{SUBSEQ}(A)$; a contradiction. □

Corollary 3.4. $\mathrm{P} \notin$ SUBSEQ-EX. *In fact,* $\mathrm{P} \cap \mathcal{P}(0^*) \notin$ SUBSEQ-EX.

We can learn more with access to the halting problem.

Theorem 3.5. CE $\in$ SUBSEQ-EX$^{\emptyset'}$, *where* $\emptyset'$ *is the halting problem.*

Proof. Consider a learner M for all c.e. languages that behaves as follows: When the characteristic string of a c.e. language A is on the tape, M learns (with the help of $\emptyset'$) a c.e. index for A by finding, for each $n = 0, 1, 2, \ldots$, the least e such that $W_e \cap \Sigma^{\leq n} = A \cap \Sigma^{\leq n}$. Eventually M will settle on a correct e, assuming A is c.e. Let e_n be the nth index found by M. Upon finding e_n, M uses $\emptyset'$ to determine, for each $w \in \Sigma^{\leq n}$, whether or not there is a $z \in W_{e_n}$ such that $w \preceq z$. M collects the set D of all $w \in \Sigma^{\leq n}$ for which this is *not* the case, then outputs (an index for) the corresponding minimum-state DFA as in Proposition 2.7(2).

For all large enough n we have $A = W_{e_n}$, and all strings in $os(A)$ will have length at most n. Thus M eventually outputs a DFA for SUBSEQ(A). □

3.2 Anomalies

The next theorem shows that the anomalies hierarchy of Equation (3) collapses completely. In other words, allowing the DFA that is output to be wrong on (say) five places does not increase learning power.

Theorem 3.6. SUBSEQ-EX = SUBSEQ-EX*. *In fact, there is a computable h such that for all e and languages A, if M_e learns* SUBSEQ(A) *with finitely many anomalies, then $M_{h(e)}$ learns* SUBSEQ(A) *(with zero anomalies).*

Proof. Given e, we let $M_{h(e)}$ learn SUBSEQ(A) by finding better and better approximations to it: For increasing n, $M_{h(e)}$ with A on its tape approximates SUBSEQ(A) by examining its tape directly on strings in $\Sigma^{<n}$ (where there could be anomalies) and relying on $L(F)$ for strings of length $\geq n$, where F is the most recent output of M_e. Here is the algorithm for $M_{h(e)}$:

When language A is on the tape:

1. Run M_e with A. Wait for M_e to output something.
2. Whenever M_e outputs some index k, do the following:
   (a) Let n be the number of times M_e has output something thus far.
   (b) Compute a DFA G recognizing the language $\mathrm{SUBSEQ}((A \cap \Sigma^{<n}) \cup (L(F_k) \cap \Sigma^{\geq n}))$.
   (c) Output the index of G.

If M_e learns A with finite anomalies, then there is a DFA F such that, for all large enough n, M_e outputs an index for F as its nth output, and furthermore $L(F) \triangle \mathrm{SUBSEQ}(A) \subseteq \Sigma^{<n}$, that

is, all anomalies are of length less than n. For any such n, let G_n be the DFA output by $M_{h(e)}$ after the nth output of M_e. We have

$$\begin{aligned} L(G_n) &= \mathrm{SUBSEQ}((A \cap \Sigma^{<n}) \cup (L(F) \cap \Sigma^{\geq n})) \\ &= \mathrm{SUBSEQ}((A \cap \Sigma^{<n}) \cup (\mathrm{SUBSEQ}(A) \cap \Sigma^{\geq n})) \\ &= \mathrm{SUBSEQ}(A). \end{aligned}$$

Thus $M_{h(e)}$ learns SUBSEQ(A). □

One could define a looser notion of learning with finite anomalies: The learner is only required to eventually (i.e., cofinitely often) output indices for DFAs whose languages differ a finite amount from SUBSEQ(A), but these languages need not all be the same.

Definition 3.7. For a learner M and language A, say that M *weakly learns* SUBSEQ(A) *with finite anomalies* if, when A is on the tape, M outputs an infinite sequence $k_1, k_2, \ldots$ such that $\mathrm{SUBSEQ}(A) \triangle L(F_{k_i})$ is finite for all but finitely many i.

A class $\mathcal{C}$ of languages is in SUBSEQ-W-EX* if there is a learner M that, for every $A \in \mathcal{C}$, weakly learns SUBSEQ(A) with finite anomalies.

Clearly, SUBSEQ-EX$^* \subseteq$ SUBSEQ-W-EX*.

We use Theorem 3.6 to get an even stronger collapse.

Proposition 3.8. SUBSEQ-EX $=$ SUBSEQ-W-EX*. *In fact, there is a computable function b such that for all e and A, if M_e weakly learns A with finite anomalies, then $M_{b(e)}$ learns A (without anomalies).*

Proof. Let c be a computable function such that for all e and A, $M_{c(e)}$ with A on the tape simulates M_e with A on the tape, and (supposing M_e outputs $k_1, k_2, \ldots$) whenever M_e outputs k_n, $M_{c(e)}$ finds the least $j \leq n$ such that $L(F_{k_j}) \triangle L(F_{k_n})$ is finite, and outputs k_j instead. (Such a j can be computed.)

Now suppose M_e weakly learns SUBSEQ(A) with finite anomalies, and let $k_1, k_2, \ldots$ be the outputs of M_e with A on the tape. Let j be least such that $L(F_{k_j}) \triangle \mathrm{SUBSEQ}(A)$ is finite. Then for cofinitely many n, we have $L(F_{k_n}) \triangle \mathrm{SUBSEQ}(A)$ is finite, and so $L(F_{k_n}) \triangle L(F_{k_j})$ is also finite, but $L(F_{k_n}) \triangle L(F_{k_\ell})$ is infinite for all $\ell < j$. Thus $M_{c(e)}$ outputs k_j cofinitely often, and so $M_{c(e)}$ learns A with finite anomalies (not weakly!).

Now we let $b = h \circ c$, where h is the function of Theorem 3.6. If M_e weakly learns A with finite anomalies, then $M_{c(e)}$ learns A with finite anomalies, and so $M_{b(e)} = M_{h(c(e))}$ learns A. □

3.3 Mind-changes

The next theorems show that the mind change hierarchy of Equation (2) separates. In other words, if you allow more mind-changes then you give the learning device more power.

Definition 3.9. For every $i > 0$, define the class

$$\mathcal{C}_i = \{A \subseteq 0^* : |A| \leq i\}.$$

Proposition 3.10. $\mathcal{C}_i \in$ SUBSEQ-EX$_i$ *for all $i \in \mathbb{N}$. In fact, there is a single learner M that for each i learns* SUBSEQ(A) *for every $A \in \mathcal{C}_i$ with at most i mind-changes.*

Proof. Let M be as in the proof of Proposition 1.9. Clearly, M learns any $A \in \mathcal{C}_i$ with at most $|A|$ mind-changes. $\square$

Theorem 3.11. *For each $i > 0$, $\mathcal{C}_i \notin \text{SUBSEQ-EX}_{i-1}$. In fact, there is a computable function ℓ such that, for each e and $i > 0$, $M_{\ell(e,i)}$ is total and decides a unary language $A_{e,i} = L(M_{\ell(e,i)}) \subseteq 0^*$ such that $|A_{e,i}| \le i$ and M_e does not learn $\text{SUBSEQ}(A_{e,i})$ with fewer than i mind-changes.*

Proof. Given e and $i > 0$ we construct a machine $N = M_{\ell(e,i)}$ that implements the following recursive algorithm to compute $A_{e,i}$:

Given input x,

1. If $x \notin 0^*$, then reject. (This ensures that $A_{e,i} \subseteq 0^*$.) Otherwise, let $x = 0^n$.
2. Recursively compute $R_n = A_{e,i} \cap 0^{<n}$.
3. Simulate M_e for $n-1$ steps with R_n on the tape. (Note that M_e does not have time to read any of the tape corresponding to inputs $0^{n'}$ for $n' \ge n$.) If M_e does not output anything within this time, then reject.
4. Let k be the most recent output of M_e in the previous step, and let c be the number of mind-changes that M_e has made up to this point. If $c < i$ and $L(F_k) = \text{SUBSEQ}(R_n)$, then accept; else reject.

In step 3 of the algorithm, M_e behaves the same with R_n on its tape as it would with $A_{e,i}$ on its tape, given the limit on its running time.

Let $A_{e,i} = \{0^{z_0}, 0^{z_1}, \ldots\}$, where $z_0 < z_1 < \cdots$ are natural numbers.

Claim 3.12. For $0 \le j$, if z_j exists, then M_e (with $A_{e,i}$ on its tape) must output a DFA for $\text{SUBSEQ}(R_{z_j})$ within $z_j - 1$ steps, having changed its mind at least j times when this occurs.

Proof (of the claim). We proceed by induction on j: For $j = 0$, the string 0^{z_0} is accepted by N only if within $z_0 - 1$ steps M_e outputs a k where $L(F_k) = \emptyset = \text{SUBSEQ}(R_{z_0})$; no mind-changes are required. Now assume that $j \ge 0$ and z_{j+1} exists, and also (for the inductive hypothesis) that within $z_j - 1$ steps M_e outputs a DFA for $\text{SUBSEQ}(R_{z_j})$ after at least j mind-changes. We have $R_{z_j} \subseteq 0^{<z_j}$ but $0^{z_j} \in R_{z_{j+1}}$, and so $\text{SUBSEQ}(R_{z_j}) \ne \text{SUBSEQ}(R_{z_{j+1}})$. Since N accepts $0^{z_{j+1}}$, it must be because M_e has just output a DFA for $\text{SUBSEQ}(R_{z_{j+1}})$ within $z_{j+1} - 1$ steps, thus having changed its mind at least once since the z_jth step of its computation, making at least $j+1$ mind-changes in all. So the claim holds for $j+1$. *End of Proof of Claim*

First we show that $A_{e,i} \in \mathcal{C}_i$. Indeed, by Claim 3.12, z_i cannot exist, because the algorithm would explicitly reject such a string 0^{z_i} if M_e made at least i mind-changes in the first $z_i - 1$ steps. Thus we have $|A_{e,i}| \le i$, and so $A_{e,i} \in \mathcal{C}_i$.

Next we show that M_e cannot learn $A_{e,i}$ with fewer than i mind-changes. Suppose that with $A_{e,i}$ on its tape, M_e makes fewer than i mind-changes. Suppose also that there is a k output cofinitely many times by M_e. Let t be least such that $t \ge m(A_{e,i})$ and M_e outputs k within $t-1$ steps. Then $L(F_k) \ne \text{SUBSEQ}(A_{e,i})$, for otherwise the algorithm would accept 0^t and so $0^t \in A_{e,i}$, contradicting the choice of t. It follows that M_e cannot learn $A_{e,i}$ with fewer than i mind-changes. $\square$

Transfinite mind-changes and procrastination This section may be skipped on first reading. We extend the results of this section into the transfinite. Freivalds & Smith defined EX_α for all constructive ordinals α [9]. When $\alpha < \omega$, the definition is the same as the finite mind-change case above. If $\alpha \geq \omega$, then the learner may revise its bound on the number of mind changes during the computation. The learner may be able to revise more than once, or even compute a bound on the number of future revisions, and this bound itself could be revised, et cetera, depending on the size of α. After giving some basic facts about constructive ordinals, we define SUBSEQ-EX$_\alpha$ for all constructive α, then show that this transfinite hierarchy separates. Our definition is slightly different from, but equivalent to, the definition in [9]. For general background on constructive ordinals, see [19, 20].

Church defined the constructive (computable) ordinals, and Kleene defined a partially ordered set $\langle \mathcal{O}, <_{\mathcal{O}} \rangle$ of *notations* for constructive ordinals, where $\mathcal{O} \subseteq \mathbb{N}$. $\langle \mathcal{O}, <_{\mathcal{O}} \rangle$ may be defined as the least partial order that satisfies the following closure properties:

- $<_{\mathcal{O}} \subseteq \mathcal{O} \times \mathcal{O}$, and $<_{\mathcal{O}}$ is transitive.
- $0 \in \mathcal{O}$.
- If $a \in \mathcal{O}$ then $2^a \in \mathcal{O}$ and $a <_{\mathcal{O}} 2^a$.
- If M_e is total (with inputs in $\mathbb{N}$) and

$$M_e(0) <_{\mathcal{O}} M_e(1) <_{\mathcal{O}} M_e(2) <_{\mathcal{O}} \cdots,$$

  then $3 \cdot 5^e \in \mathcal{O}$ and $M_e(n) <_{\mathcal{O}} 3 \cdot 5^e$ for all $n \in \mathbb{N}$.

$\langle \mathcal{O}, <_{\mathcal{O}} \rangle$ has the structure of a well-founded tree. For $a \in \mathcal{O}$ we let $\|a\|$ be the ordinal rank of a in the partial ordering.[5] Then a is a *notation* for the ordinal $\|a\|$. An ordinal α is *constructive* if it has a notation in $\mathcal{O}$. We let ω_1^{CK} be the set of all constructive ordinals, i.e., the height of the tree $\langle \mathcal{O}, <_{\mathcal{O}} \rangle$. ω_1^{CK} is itself a countable ordinal—the least nonconstructive ordinal.

It can be shown that $\langle \mathcal{O}, <_{\mathcal{O}} \rangle$ has individual branches of height ω_1^{CK}. If $B \subseteq \mathcal{O}$ is such a branch, then every constructive ordinal has a unique notation in B. In keeping with [9], we fix a single such branch ORD $\subseteq \mathbb{N}$ of unique notations once and for all, then identify (for computational purposes) each constructive ordinal with its notation in ORD. (It is likely that the classes we define depend on the actual system ORD chosen, but our results hold for any such branch that we fix.)

We note the following basic facts about constructive ordinals $\alpha < \omega_1^{\mathrm{CK}}$:

- It is a computable task to determine whether α is zero, α is a successor, or α is a limit. ($\alpha = 0$, $\alpha = 2^a$ for some a, or $\alpha = 3 \cdot 5^e$ for some e, respectively.)
- If α is a successor, then its predecessor ($= \log_2 \alpha$) can be computed.
- If $\alpha = 3 \cdot 5^e$ is a limit, then we can compute $M_e(0), M_e(1), M_e(2), \ldots$, and this is a strictly ascending sequence of ordinals with limit α.
- We can compute the unique ordinals λ and n such that λ is zero or a limit, $n < \omega$, and $\lambda + n = \alpha$. We denote this n by $N(\alpha)$ and this λ by $\Lambda(\alpha)$.
- There is a computably enumerable set S such that for all $b \in$ ORD and $a \in \mathbb{N}$, $(a, b) \in S$ iff $a \in$ ORD and $\|a\| < \|b\|$. That is, given an ordinal $\alpha < \omega_1^{\mathrm{CK}}$, we can effectively enumerate all $\beta < \alpha$, and this enumeration is uniform in α.
- Thanks to ORD being totally ordered, the previous item implies that we can effectively determine whether or not $\alpha < \beta$ for any $\alpha, \beta < \omega_1^{\mathrm{CK}}$. That is, there is a partial computable predicate that extends the ordinal less-than relation on ORD.

[5] The usual expression for the rank of a is $|a|$, but we change the notation here to avoid confusion with set cardinality and string length.

Definition 3.13. A *procrastinating learner* is a learner M equipped with an additional *ordinal tape*, whose contents is always a constructive ordinal. Given a language on its input tape, M runs forever, producing infinitely many outputs as usual, except that just before M changes its mind, if α is currently on its ordinal tape, M is required to compute some ordinal $\beta < \alpha$ and replace the contents of the ordinal tape with β before proceeding to change its mind. (So if $\alpha = 0$, no mind-change may take place.) M may alter its ordinal tape at any other time, but the only allowed change is replacement with a lesser ordinal.

Thus a procrastinating learner must decrease its ordinal tape before each mind-change.

We abuse notation and let $M_1, M_2, \ldots$ be a standard enumeration of procrastinating learners. Such an effective enumeration exists because we can enforce the ordinal-decrease requirement for a machine's ordinal tape: if $b \in \mathrm{ORD}$ is the current contents of the ordinal tape, and the machine wishes (or is required) to alter it—say, to some value $a \in \mathbb{N}$—we first start to computably enumerate the set of all $c \in \mathrm{ORD}$ such that $\|c\| < \|b\|$ and allow the machine to proceed only when a shows up in the enumeration.

Definition 3.14. Let M be a procrastinating learner, α a constructive ordinal, and A a language. We say that M *learns* SUBSEQ(A) *with* α *mind-changes* if M learns SUBSEQ(A) with α initially on its ordinal tape.

If $\mathcal{C}$ is a class of languages, we say that $\mathcal{C} \in \text{SUBSEQ-EX}_\alpha$ if there is a procrastinating learner that learns every language in $\mathcal{C}$ with α mind-changes.

The following is straightforward and given without proof.

Proposition 3.15. *If $\alpha < \omega$, then* SUBSEQ-EX_α *is the same as the usual finite mind-change version of* SUBSEQ-EX.

Proposition 3.16. *For all $\alpha < \beta < \omega_1^{\mathrm{CK}}$,*

$$\text{SUBSEQ-EX}_\alpha \subseteq \text{SUBSEQ-EX}_\beta \subseteq \text{SUBSEQ-EX}.$$

Proof. The first containment follows from the fact that any procrastinating learner allowed α mind-changes can be simulated by a procrastinating learner, allowed β mind-changes, that first decreases its ordinal tape from β to α before the simulation. (α is hard-coded into the simulator.)

The second containment is trivial; any procrastinating learner is also a regular learner. □

In [9], Freivalds and Smith defined EX_α for constructive α and showed that this hierarchy separates using classes of languages constructed entirely by diagonalization. We take a different approach and define more "natural" (using the term loosely) classes of languages that separate the SUBSEQ-EX_α hierarchy.

Definition 3.17. For every $\alpha < \omega_1^{\mathrm{CK}}$, we define the class $\mathcal{F}_\alpha$ inductively as follows: Let $n = N(\alpha)$, and let $\lambda = \Lambda(\alpha)$.

- If $\lambda = 0$, let
$$\mathcal{F}_\alpha = \mathcal{F}_n = \{B \cup\!\!+ \emptyset : (B \subseteq 0^*) \wedge (|B| \le n)\}.$$
- If $\lambda > 0$, then λ has notation $3 \cdot 5^e$ for some TM index e. Let
$$\mathcal{F}_\alpha = \{B \cup\!\!+ C : (B, C \subseteq 0^*) \wedge (|B| \le n+1) \wedge (C \in \mathcal{F}_{M_e(m(B))})\}.$$

It is evident by induction on α that $\mathcal{F}_\alpha$ consists only of finite unary languages and that $\emptyset \in \mathcal{F}_\alpha$. Note that in the case of finite α we have the condition $|B| \le n$, but in the case of $\alpha \ge \omega$ we have the condition $|B| \le n+1$. This is not a mistake.

The next two theorems have proofs that are similar to the finite mind-change case in some ways, but very different in others.

Theorem 3.18. *For every constructive α, $\mathcal{F}_\alpha \in$ SUBSEQ-EX$_\alpha$. In fact, there is a single procrastinating learner N such that for every α, N learns every language in $\mathcal{F}_\alpha$ with α mind-changes.*

Proof. With α initially on its ordinal tape and language A on its input tape, the machine N executes the following recursive algorithm:

1. Compute $n := N(\alpha)$ and $\lambda := \Lambda(\alpha)$.
2. For $i = 0, 1, 2, \ldots$ in increasing order, do the following:
   (a) Let k_i be the index of a DFA recognizing SUBSEQ$(A \cap 0^{<i+1})$.
   (b) If $i = 0$ or $0^i \notin A$, then outputting k_i does not require a mind-change; output k_i, and proceed to the next i.
   (c) Else, we have $i > 0$ and $0^i \in A$, and so a mind-change is required before outputting k_i.
      i. If i is odd and $n > 0$, then (since α is a successor) replace α with its predecessor on the ordinal tape, decrease n by one, output k_i, and continue to the next i.
      ii. (At this point, either i is even or $n = 0$.) If $\lambda = 0$, then halt. (This never happens if $A \in \mathcal{F}_\alpha$.)
      iii. We get e such that λ has notation $3 \cdot 5^e$.
         A. If i is even, then set $B := \xi(A \cap 0^{<i})$.
         B. Otherwise, set $B := \xi(A \cap 0^{<i+1})$.
      iv. Let $C = \pi(m(B); A)$. Set $\gamma := M_e(m(B))$.
      v. If i is odd, then replace α with $\gamma + 1$ on the ordinal tape and output k_i.
      vi. Write γ on the ordinal tape. (This gives us license for the first output in the simulation below; the simulated machine might make its first output without altering its ordinal tape.)
      vii. Simulate N from the beginning with C on its input tape and γ initially on its ordinal tape:
         – If the simulation ever halts, then halt. (This never happens if $A \in \mathcal{F}_\alpha$.)
         – Whenever the simulation alters its ordinal tape, alter the ordinal tape in the same way.
         – Whenever the simulation outputs some k, and it is the case that $L(F_k) = 0^{<s}$ for some s, then output the index of a DFA recognizing SUBSEQ$(B \uplus 0^{<s})$.

         (We never get out of this step.)

We prove by induction on α that N correctly learns any $A \in \mathcal{F}_\alpha$.

If $\alpha < \omega$, then $A = B \uplus \emptyset$ for some $B \subseteq 0^*$ such that $|B| \le \alpha$. All strings in A have odd length, and because $|A| = |B| \le n = \alpha$, we have enough mind-changes available so that $n > 0$ whenever we reach step 2(c)i. This means that we never go beyond this step. For all large enough i, we have $A \subseteq 0^{<i}$, and so we output a DFA for SUBSEQ(A) in step 2b.

Now suppose $\alpha \ge \omega$. Let λ and n be as computed in step 1. Then λ has notation $3 \cdot 5^e$ for some e, and $A = B \uplus C$, where $B, C \subseteq 0^*$, $|B| \le n+1$, and $C \in \mathcal{F}_\gamma$, where $\gamma = M_e(m(B))$. When we get to step 2(c)iii, then we have "seen" all strings in A coming from B, either because (1) i is even and so 0^i is the shortest string coming from C, or (2) we have already seen n many strings from B shorter than i (causing n mind-changes) and thus 0^i is the longest string coming from B. In either case, B is correctly computed in step 2(c)iiiA or step 2(c)iiiB, and thus C and γ are correct in step 2(c)iv. We have the following situation after step 2(c)vi: N's most recent output is the index of a DFA for $B \cup\!\!+ \emptyset$, and after that output, N's ordinal tape is decreased to γ. N is then run recursively on C in step 2(c)vii. (The first output of the recursive call may constitute a mind-change for the original call, but this is okay because of the ordinal decrease in step 2(c)vi, just before the recursive call.) By the inductive hypothesis, the simulated N correctly learns SUBSEQ(C) with γ mind-changes by cofinitely often outputting the least k such that $L(F_k) = \text{SUBSEQ}(C) = 0^{<s}$ for some $s > 0$. Clearly,

$$\begin{aligned}\text{SUBSEQ}(A) &= \text{SUBSEQ}(B \cup\!\!+ C) = \text{SUBSEQ}(B \cup\!\!+ \text{SUBSEQ}(C)) \\ &= \text{SUBSEQ}(B \cup\!\!+ 0^{<s}).\end{aligned}$$

Further, during the simulation, the original run of N will change its mind only when the simulated N does. Thus the original run of N will output the index of a DFA recognizing SUBSEQ(A) cofinitely often, using α mind-changes. ☐

Theorem 3.19. *For all $\beta < \alpha < \omega_1^{\text{CK}}$, $\mathcal{F}_\alpha \notin$ SUBSEQ-EX$_\beta$. In fact, there is a computable function r such that, for each e and $\beta < \alpha < \omega_1^{\text{CK}}$, $M_{r(e,\alpha,\beta)}$ is total and decides a language $A_{e,\alpha,\beta} = L(M_{r(e,\alpha,\beta)}) \in \mathcal{F}_\alpha$ such that M_e does not learn* SUBSEQ($A_{e,\alpha,\beta}$) *with β mind-changes.*

Proof. This proof generalizes the proof of Theorem 3.11 to the transfinite case. We first define a computable function $v(e, c, t, b)$ such that for all $e, c, t, b \in \mathbb{N}$, the procrastinating learner $M_{v(e,c,t,b)}$ with language C on its input tape and $g \in \mathbb{N}$ on its ordinal tape[6] behaves as follows:

1. Without changing the ordinal tape or outputting anything, $M_{v(e,c,t,b)}$ simulates M_e for t steps with $(D_c \cap 0^*) \cup\!\!+ (C \cap 0^*)$ on M_e's input tape and b on M_e's ordinal tape.
2. $M_{v(e,c,t,b)}$ continues to simulate M_e as above beyond t steps, except that now:
   - Whenever M_e changes its ordinal tape to some value u, $M_{v(e,c,t,b)}$ changes *its* ordinal tape to the same value u (provided this is allowed).
   - Whenever M_e outputs a value k, $M_{v(e,c,t,b)}$ outputs the index of a DFA recognizing the language $\pi(m(D_c); L(F_k))$ (provided this is allowed).

The function v is defined so that if M_e learns SUBSEQ($D_c \cup\!\!+ C$) (for some $D_c, C \subseteq 0^*$) with β mind-changes *and* M_e manages to decrease its ordinal tape to some δ within the first t steps of its computation, then $M_{v(e,c,t,\beta)}$ learns SUBSEQ(C) with γ mind-changes, for any $\gamma \geq \delta$. (Observe that $\text{SUBSEQ}(C) = \pi(m(D_c); \text{SUBSEQ}(D_c \cup\!\!+ C))$.) We will use the contrapositive of this fact in the proof, below.

Given e and $\beta < \alpha < \omega_1^{\text{CK}}$ we construct the set $A_{e,\alpha,\beta} \subseteq 0^*$, which is decidable uniformly in e, α, β. The rough idea is that we build $A_{e,\alpha,\beta}$ to be of the form $B \cup\!\!+ C$, where $B, C \subseteq 0^*$ and $|B| \leq N(\alpha) + 1$ (assuming $\alpha \geq \omega$), while diagonalizing against M_e with β on its ordinal tape. We put strings into B to force mind-changes in M_e until either M_e runs out of mind-changes (and is wrong) or it decreases its ordinal tape to some ordinal $\delta < \Lambda(\alpha)$. If the latter happens, we then put one more string into B to code some γ such that $\delta < \gamma < \Lambda(\alpha)$, and then (recursively) make C equal to $A_{\hat{e},\gamma,\delta}$ for some appropriate $\hat{e}$ chosen using the function v, above. Here is the construction of $A_{e,\alpha,\beta}$:

[6] For the purposes of defining the function v, we must take b and g to be arbitrary numbers, although they will usually be notations for ordinals.

1. Let $\lambda = \Lambda(\alpha)$.
2. Initialize $B := \emptyset$ and $t := 0$.
3. Repeat the following as necessary to construct B:
   (a) Run M_e with $B \cup\!\!+ \emptyset$ on its tape and β initially on its ordinal tape until it outputs some k such that $L(F_k) = \mathrm{SUBSEQ}(B \cup\!\!+ \emptyset)$ after more than t steps. This may never happen, in which case we define $A_{e,\alpha,\beta} := B \cup\!\!+ \emptyset$ and we are done.
   (b) Let $t' > t$ be the number of steps it took M_e to output k, above. Let δ be the contents of M_e's ordinal tape when k was output. [Note that M_e did not have time to scan any strings of the form 0^s for $s > t'$.] Reset $t := t'$.
   (c) If $\delta < \lambda$, then go on to Step 4.
   (d) Set $B := B \cup \{0^{t+1}\}$ and continue the repeat-loop.
4. Now we have $\delta < \lambda$, and so λ is a limit ordinal with notation $3 \cdot 5^u$ for some u. Let p be least such that $p > t$ and $M_u(p+1)$ is the notation for some ordinal $\gamma > \delta$. [Note that $\gamma < \lambda \leq \alpha$.]
5. Set $B := B \cup \{0^p\}$. [This makes $m(B) = p + 1$.]
6. Let c be such that $B = D_c$. Set $\hat{e} := v(e, c, t, \beta)$, and (recursively) define $A_{e,\alpha,\beta} := B \cup\!\!+ A_{\hat{e},\gamma,\delta}$. [The ordinal in the second subscript decreases from α to γ, so the recursion is well-founded.]

For all e and all $\beta < \alpha < \omega_1^{\mathrm{CK}}$, we show by induction on α that $A_{e,\alpha,\beta} \in \mathcal{F}_\alpha$ and that M_e cannot learn $\mathrm{SUBSEQ}(A_{e,\alpha,\beta})$ with β initially on its ordinal tape. Let $\lambda = \Lambda(\alpha)$ (λ may be either 0 or a limit), and let $n = N(\alpha)$. Consider M_e running with $A_{e,\alpha,\beta}$ on its input tape and β initially on its ordinal tape. In the repeat-loop, t bounds the running time of M_e and strictly increases from one complete iteration to the next, and the only strings added to B have length greater than t. This implies two things: (1) that M_e behaves the same in Step 3a with $B \cup\!\!+ \emptyset$ on its tape as it would with $A_{e,\alpha,\beta}$ on its tape, and (2) the number of mind-changes M_e must make to be correct increases in each successive iteration of the loop.

We now consider two cases:

λ is the 0 ordinal. Then M_e can change its mind at most $n-1$ times (since $\beta < \alpha = n$). This means that the repeat-loop will run for at most n complete iterations, then hang in Step 3a on the next iteration, because by then M_e has run out of mind-changes and so cannot update its answer to be correct. In this case, $A_{e,\alpha,\beta} = B \cup\!\!+ \emptyset$, and we've added at most n strings to B. Thus $A_{e,\alpha,\beta} \in \mathcal{F}_\alpha$, and M_e does not learn $\mathrm{SUBSEQ}(A_{e,\alpha,\beta})$ with β mind-changes.

λ is a limit ordinal with notation $3 \cdot 5^u$ for some u. Then M_e can change its mind at most $n-1$ times before it must drop its ordinal to some $\delta < \lambda$ for its next mind-change. So again there can be at most n complete iterations of the repeat-loop—putting at most n strings into B—before we either hang in Step 3a (which is just fine) or go on to Step 4. In the latter case, we put one more string into B in Step 5, making $|B| \leq n+1$. By the inductive hypothesis and the choice of p and γ, we have $A_{\hat{e},\gamma,\delta} \in \mathcal{F}_\gamma = \mathcal{F}_{M_u(m(B))}$, and so $A_{e,\alpha,\beta} \in \mathcal{F}_\alpha$.
The index $\hat{e}$ is chosen precisely so that if M_e learns $\mathrm{SUBSEQ}(A_{e,\alpha,\beta})$ with β mind-changes then $M_{\hat{e}}$ learns $\mathrm{SUBSEQ}(A_{\hat{e},\gamma,\delta})$ with δ mind-changes. By the inductive hypothesis, $M_{\hat{e}}$ cannot do this. Thus in either case M_e does not learn $\mathrm{SUBSEQ}(A_{e,\alpha,\beta})$ with β mind-changes.

It remains to show that $A_{e,\alpha,\beta}$ is decidable uniformly in e, α, β. The only tricky part is Step 3a, which may run forever. It is not hard to see, however, that if M_e runs for at least ℓ steps for some ℓ, then either 0^ℓ is already in B by this point or it will never get into B. Hence we can decide whether or not $0^{2\ell+1}$ is in $A_{e,\alpha,\beta}$. Even-length strings in 0^* can be handled similarly, possibly via a recursive call to $A_{\hat{e},\gamma,\delta}$. □

We end with an easy observation.

Corollary 3.20.

$$\text{SUBSEQ-EX} \not\subseteq \bigcup_{\alpha<\omega_1^{\text{CK}}} \text{SUBSEQ-EX}_\alpha.$$

Proof. Let $\mathcal{F} \in$ SUBSEQ-EX be the class of Definition 1.8. For all $\alpha < \omega_1^{\text{CK}}$, we clearly have $\mathcal{F}_{\alpha+1} \subseteq \mathcal{F}$, and so $\mathcal{F} \notin \text{SUBSEQ-EX}_\alpha$ by Theorem 3.19. □

3.4 Teams

In this section, we show that $[a,b]$SUBSEQ-EX depends only on $\lfloor b/a \rfloor$. Recall that $b \le c$ implies $[a,b]$SUBSEQ-EX $\subseteq [a,c]$SUBSEQ-EX.

Lemma 3.21. *For all* $1 \le a \le b$,

$$[a,b]\text{SUBSEQ-EX} = [1, \lfloor b/a \rfloor]\text{SUBSEQ-EX}.$$

Proof. Let $q = \lfloor b/a \rfloor$. To show that

$$[1,q]\text{SUBSEQ-EX} \subseteq [a,b]\text{SUBSEQ-EX},$$

let $\mathcal{C} \in [1,q]$SUBSEQ-EX. Then there are learners $Q_1, \dots, Q_q$ such that for all $A \in \mathcal{C}$ there is some Q_i that learns SUBSEQ(A). For all $1 \le i \le q$ and $1 \le j \le a$, let $N_{i,j} = Q_i$. Then clearly, $\mathcal{C} \in [a, qa]$SUBSEQ-EX as witnessed by the $N_{i,j}$. Thus, $\mathcal{C} \in [a,b]$SUBSEQ-EX, since $b \ge qa$.

To show the reverse containment, suppose that $\mathcal{D} \in [a,b]$SUBSEQ-EX. Let $Q_1, \dots, Q_b$ be learners such that for each $A \in \mathcal{D}$, at least a of the Q_i's learn SUBSEQ(A). We define learners $N_1, \dots, N_q$ to behave as follows with A on their tapes.

Each N_j runs all of $Q_1, \dots, Q_b$. At any time t, let $k_1(t), \dots, k_b(t)$ be the most recent outputs of $Q_1, \dots, Q_b$, respectively, after running for t steps (if some machine Q_i has not yet output anything in t steps, let $k_i(t) = 0$).

Define a *consensus value at time* t to be a value that shows up at least a times in the list $k_1(t), \dots, k_b(t)$. There can be at most q many different consensus values at any given time, so we can make the machines N_j output these consensus values. If k_{correct} is the index of the DFA recognizing SUBSEQ(A), then k_{correct} will be a consensus value at all sufficiently large times, and so k_{correct} will eventually always be output by one or another of the N_j. The only trick is to ensure that k_{correct} is eventually output by the *same* N_j each time. To make sure of this, the N_j will output consensus values in order of seniority.

For $1 \le j \le q$ and $t = 1, 2, 3, \dots$, each machine N_j computes $k_1(t'), \dots, k_b(t')$ and all the consensus values at time t' for all $t' \le t$. For each $v \in \mathbb{N}$, we define the *start time* of v at time t to be either $t+1$, if v is not a consensus value at time t, or else the earliest time $s \le t$ such that v is a consensus value at all times t' with $s \le t' \le t$. As its t'th output, N_j outputs the value with the j'th earliest start time at time t. If there is a tie, then we consider the smaller value to have started earlier. This ends the description of the machines $N_1, \dots, N_q$.

Let Y be the set of all consensus values that occur cofinitely often. Clearly, $k_{\text{correct}} \in Y$, and there is a time t_0 such that all elements of Y are consensus values at all times $t \ge t_0$. Note that the start times of the values in Y do not change from t_0 onward, but the start time of any value not in Y increases monotonically without bound. Thus there is a time $t_1 \geq t_0$ beyond which any $v \notin Y$ has a start time later than that of any $v' \in Y$. It follows that from time t_1 onward, the start time of k_{correct} has a fixed rank amongst the start times of all the current consensus values, and so k_{correct} is output by the same machine N_j at all times $t \geq t_1$. □

To prove a separation, we cannot use unary languages as we have before; it is easy to see (exercise for the reader) that $\mathcal{P}(0^*) \in [1,2]$SUBSEQ-EX. To separate the team hierarchy beyond level 2, we use an alphabet Σ that contains 0 and 1 (at least) and show that $\mathcal{S}_{\leq n} \in [1, n+1]$SUBSEQ-EX $-$ $[1, n]$SUBSEQ-EX for all $n \geq 1$, where $\mathcal{S}_{\leq n}$ is given in Definition 3.1.

Lemma 3.22. *For all $n \geq 1$, $\mathcal{S}_{\leq n} \in [1, n+1]$SUBSEQ-EX and $\mathcal{S}_{\leq n} \cap \text{DEC} \notin [1, n]$SUBSEQ-EX. In fact, there is a computable function $d(s)$ such that for all $n \geq 1$ and all $e_1, \ldots, e_n$, the machine $M_{d([e_1,\ldots,e_n])}$ decides a language $A_{[e_1,\ldots,e_n]} \in \mathcal{S}_{\leq n}$ that is not learned by any of $M_{e_1}, \ldots, M_{e_n}$.*[7]

Proof. Fix $n \geq 1$. First, we have $\mathcal{S}_{\leq n} = \mathcal{S}_0 \cup \cdots \cup \mathcal{S}_n$, and $\mathcal{S}_i \in$ SUBSEQ-EX for each $i \leq n$ by Proposition 3.2. It follows that $\mathcal{S}_{\leq n} \in [1, n+1]$SUBSEQ-EX.

Next, we show that $\mathcal{S}_{\leq n} \notin [1, n]$SUBSEQ-EX. Fix any n learners $Q_1, \ldots, Q_n$. We build a set $A \subseteq \Sigma^*$ in stages $n, n+1, n+2, \ldots$, ensuring that $os(A) \leq n$ (hence $A \in \mathcal{S}_{\leq n}$) and that none of the Q_i learn SUBSEQ(A). At each stage $j \geq n$, we define n strings $y_1^j, \ldots, y_n^j \in \{0,1\}^*$ which are candidates for membership in $os(A)$. These strings satisfy

1. $|y_1^j| \leq \cdots \leq |y_n^j| \leq j+1$, and
2. $y_i^j \in 0^{n-i}1^*1$ for all $1 \leq i \leq n$.

Note that these two conditions imply that $y_1^j, \ldots, y_n^j$ are pairwise $\preceq$-incomparable. We then define A on all strings of length j.

Stage n: For all $1 \leq i \leq n$, set $y_i^n := 0^{n-i}1^{i+1}$. Set $A_n := \Sigma^{\leq n}$.
Stage $j > n$:
- Run each learner Q_i for j steps and let k_i be its most recent output (or let $k_i = 0$ if there is no output yet). Compute $s_i := |os(L(F_{k_i}))|$ for all $1 \leq i \leq n$.
- Let m_j be the least element of $\{0, \ldots, n\} - \{s_1, \ldots, s_n\}$.
- Set $y_i^j := y_i^{j-1}$ for all $1 \leq i \leq m_j$, and set $y_i^j := 0^{n-i}1^{j+1-n+i}$ for all $m_j < i \leq n$.
- Set $A_j := A_{j-1} \cup \{x \in \Sigma^{=j} : (\forall i \leq m_j) y_i^j \not\preceq x\}$.

Define $A := \bigcup_{j=n}^{\infty} A_j$. Also define $m := \liminf_{j\to\infty} m_j$, and let $j_0 > n$ be least such that $m_j \geq m$ for all $j \geq j_0$. For $1 \leq i \leq m$, we then have $y_i^{j_0} = y_i^{j_0+1} = y_i^{j_0+2} = \cdots$, and we define y_i to be this string. It remains to show that $os(A) = \{y_1 \ldots, y_m\}$, for if this is the case, then the obstruction set size $m = |os(A)| = |os(\text{SUBSEQ}(A))|$ is omitted infinitely often by all the learners, and so none of the learners can converge on a language with an obstruction set of size m, and hence none of the learners learn SUBSEQ(A).

To see that $os(A) = \{y_1, \ldots, y_m\}$, consider an arbitrary string $x \in \Sigma^*$. We need to show that $x \in$ SUBSEQ(A) iff $(\forall i) y_i \not\preceq x$. By the construction, no $z \succeq y_i$ ever enters A for any $i \leq m$, so if $x \preceq z$ and $z \in A$, then $(\forall i) y_i \not\preceq z$ and thus $(\forall i) y_i \not\preceq x$. Conversely, if $(\forall i) y_i \not\preceq x$, then $(\forall i) y_i \not\preceq x0^t$ for any $t \geq 0$, because each y_i ends with a 1. Fix the least $j_1 \geq \max(j_0, |x|)$ such that $m_{j_1} = m$, and let $t = j_1 - |x|$. Then $|x0^t| = j_1$, and $x0^t$ is added to A at Stage j_1. So we have $x \preceq x0^t \in A$, whence $x \in$ SUBSEQ(A).

Finally, the whole construction of A above is effective uniformly in n and indices for $Q_1, \ldots, Q_n$, and uniformly decides A. Thus the computable function d of the Lemma exists. □

[7] $[e_1, e_2, \ldots, e_n]$ is a natural number encoding the finite sequence $e_1, e_2, \ldots, e_n$.

Remark. The foregoing proof can be easily generalized to show that $\mathcal{S}_{j_1} \cup \mathcal{S}_{j_2} \cup \cdots \cup \mathcal{S}_{j_k} \in [1,k]\text{SUBSEQ-EX} - [1,k-1]\text{SUBSEQ-EX}$ for all $j_1 < j_2 < \cdots < j_k$.

Lemmas 3.21 and 3.22 combine to show the following general theorem, which completely characterizes the containment relationships between the various team learning classes $[a,b]\text{SUBSEQ-EX}$.

Theorem 3.23. *For every* $1 \le a \le b$ *and* $1 \le c \le d$, $[a,b]\text{SUBSEQ-EX} \subseteq [c,d]\text{SUBSEQ-EX}$ *if and only if* $\lfloor b/a \rfloor \le \lfloor d/c \rfloor$.

Proof. Let $p = \lfloor b/a \rfloor$ and let $q = \lfloor d/c \rfloor$.

By Lemma 3.21 we have

$$[a,b]\text{SUBSEQ-EX} = [1,p]\text{SUBSEQ-EX},$$

and

$$[c,d]\text{SUBSEQ-EX} = [1,q]\text{SUBSEQ-EX}.$$

By Lemma 3.22 we have $[1,p]\text{SUBSEQ-EX} \subseteq [1,q]\text{SUBSEQ-EX}$ if and only if $p \le q$. ☐

3.5 Anomalies and teams

In this and the next few subsections we will discuss the effect that combining the variants discussed previously have on the results of the previous subsections.

The next result shows that Theorem 3.6 is unaffected by teams. In fact, teams and anomalies are completely orthogonal.

Theorem 3.24. *The anomaly hierarchy collapses with teams. In other words, for all* a *and* b,

$$[a,b]\text{SUBSEQ-EX}^* = [a,b]\text{SUBSEQ-EX}.$$

Proof. Given a team $M_{e_1}, \ldots, M_{e_b}$ of b Turing machines, we use the collapse strategy from Theorem 3.6 on each of the machines. We replace each M_{e_i} with the machine $M_{h(e_i)}$, where h is the function of Theorem 3.6. If a of the b machines learn the subsequence language with finite anomalies each, then their replacements will learn it with no anomalies. ☐

3.6 Anomalies and mind-changes

Next, we consider machines which are allowed a finite number of anomalies, but have a bounded number of mind changes.

In our proof that the anomaly hierarchy collapses (Theorem 3.6), the simulating learner $M_{h(e)}$ may have to make many more mind-changes than the learner M_e being simulated. As the next result shows, we cannot do better than this.

Proposition 3.25. $\text{SUBSEQ-EX}_0^* \not\subseteq \text{SUBSEQ-EX}_c$ *for any* $c \in \mathbb{N}$ *(or even* SUBSEQ-EX_α *for any* $\alpha < \omega_1^{\text{CK}}$*).*

Proof. The class $\mathcal{F}$ of Definition 1.8 is in SUBSEQ-EX_0^* (the learner always outputs the DFA for $\emptyset$). But $\mathcal{F} \notin \text{SUBSEQ-EX}_c$ by Theorem 3.11 (and $\mathcal{F} \notin \text{SUBSEQ-EX}_\alpha$ by Corollary 3.20). ☐

In light of Proposition 3.25, it may come as a surprise that a *bounded* number of anomalies may be removed without *any* additional mind-changes.

Theorem 3.26. SUBSEQ-EX$_c^a$ = SUBSEQ-EX$_c$ *for all* $a, c \geq 0$. *In fact, there is a computable* h *such that, for all* e, a *and languages* A, $M_{h(e,a)}$ *on* A *makes no more mind-changes than* M_e *on* A, *and if* M_e *learns* SUBSEQ(A) *with at most* a *anomalies, then* $M_{h(e,a)}$ *learns* SUBSEQ(A) *(with zero anomalies).*

Proof. The $\supseteq$-containment is obvious. For the $\subseteq$-containment, we modify the learner in the proof of Theorem 3.6. Given e and a, we give the algorithm for the learner $M_{h(e,a)}$ below. We will use the word "default" as a verb to mean, "output the same DFA as we did last time, or, if there was no last time, don't output anything." The opposite of defaulting is "acting." Here's how $M_{h(e,a)}$ works:

When language A is on the tape:

1. Run M_e with A. Wait for M_e to output something.
2. Whenever M_e outputs some index k, do the following:
   (a) Let n be the number of times M_e has output something thus far. (k is the nth output.)
   (b) If there was some time in the past when we acted and M_e has not changed its mind since then, then default.
   (c) Else, if F_k has more than n states, then default.
   (d) Else, if $L(F_k) \cup \Sigma^{<n}$ is not $\preceq$-closed, then default.
   (e) Else, if there are strings $w \in os(L(F) \cup \Sigma^{<n})$ and $z \in A$ such that $w \preceq z$ and $|z| < |w| + a$, then default. [Note that w, if it exists, has length at least n.]
   (f) Else, find a DFA G recognizing the language SUBSEQ$((A \cap \Sigma^{<n}) \cup (L(F_k) \cap \Sigma^{\geq n}))$, and output the index of G. [This is where we *act*, i.e., not default.]

First, it is not too hard to see that $M_{h(e,a)}$ does not change its mind any more than M_e does: After M_e makes a new conjecture, $M_{h(e,a)}$ will act at most once before M_e makes a different conjecture. This is ensured by Step 2b. Note that $M_{h(e,a)}$ only makes a new conjecture when it acts.

Suppose M_e learns SUBSEQ(A) with at most a anomalies. Let F be the final DFA output by M_e with A on its tape. We have $|L(F) \triangle \text{SUBSEQ}(A)| \leq a$. Let n_0 be least such that M_e always outputs F starting with its n_0th output onwards. It remains to show that

1. $M_{h(e,a)}$ acts sometime after M_e starts perpetually outputting F, i.e., after its n_0th output, and
2. when this happens, the G output by $M_{h(e,a)}$ is correct, i.e., $L(G) = \text{SUBSEQ}(A)$. (Since $M_{h(e,a)}$ only defaults thereafter, it outputs G forever and thus learns SUBSEQ(A).)

For (1), we start by noting that there is a least $n \geq n_0$ such that

- F has at most n states, and
- all anomalies are of length less than n, i.e., $L(F) \triangle \text{SUBSEQ}(A) \subseteq \Sigma^{<n}$.

We claim that $M_{h(e,a)}$ acts sometime between M_e's n_0th output and its nth output, inclusive. Suppose we've reached M_e's nth output and we haven't acted since the n_0th output. Then we don't default in Step 2b. We don't default in Step 2c because F has at most n states. Since all anomalies are in $\Sigma^{<n}$, clearly, $L(F) \cup \Sigma^{<n} = \text{SUBSEQ}(A) \cup \Sigma^{<n}$, which is $\preceq$-closed, so we don't default in Step 2d. Finally, we won't default in Step 2e: if w and z existed, then w would be an anomaly of length $\geq n$, but all anomalies are of length $< n$. Thus we act on M_e's nth output, which proves (1).

For (2), we know from (1) that $M_{h(e,a)}$ acts on M_e's nth output, for some $n \geq n_0$, at which time $M_{h(e,a)}$ outputs some DFA G. We claim that $L(G) = \text{SUBSEQ}(A)$.

Since $M_{h(e,a)}$ acts on M_e's nth output, we know that

- F has at most n states,
- $L(F) \cup \Sigma^{<n}$ is $\preceq$-closed, and
- there are no strings $w \in os(L(F) \cup \Sigma^{<n})$ and $z \in \Sigma^{<|w|+a} \cap A$ such that $w \preceq z$.

It suffices to show that there are no anomalies of length $\geq n$, for then we have

$$\begin{aligned} L(G) &= \mathrm{SUBSEQ}((A \cap \Sigma^{<n}) \cup (L(F) \cap \Sigma^{\geq n})) \\ &= \mathrm{SUBSEQ}((A \cap \Sigma^{<n}) \cup (\mathrm{SUBSEQ}(A) \cap \Sigma^{\geq n})) = \mathrm{SUBSEQ}(A) \end{aligned}$$

as in the proof of Theorem 3.6.

There are two kinds of anomalies—false positives (elements of $L(F) - \mathrm{SUBSEQ}(A)$) and false negatives (elements of $\mathrm{SUBSEQ}(A) - L(F)$).

First, there can be no false positives of length $\geq n$: Suppose w is such a string. Then since w is at least as long as the number of states of F, by the Pumping Lemma for regular languages there are strings x, y, z with $|y| > 0$ such that the strings

$$w = xyz \prec xy^2z \prec xy^3z \prec \cdots$$

are all in $L(F)$. But since $w \notin \mathrm{SUBSEQ}(A)$, none of these other strings is in $\mathrm{SUBSEQ}(A)$ either. This means there are infinitely many anomalies, which is false by assumption. Thus no such w exists.

Finally, we prove that there are no false negatives in $\Sigma^{\geq n}$. Suppose u is such a string. We have $u \in \mathrm{SUBSEQ}(A)$, and so there is a string $z \in A$ such that $u \preceq z$. We also have $u \notin L(F) \cup \Sigma^{<n}$, and since $L(F) \cup \Sigma^{<n}$ is $\preceq$-closed, there is some string $w \in os(L(F) \cup \Sigma^{<n})$ such that $w \preceq u$. Now $w \preceq z$ as well, so it must be that $|z| \geq |w| + a$ by what we know above. Since $w \preceq z$, there is also an ascending chain of strings

$$w = w_0 \prec w_1 \prec \cdots \prec w_k = z,$$

where $|w_i| = |w| + i$ and so $k \geq a$. All the w_i are in $\mathrm{SUBSEQ}(A)$ because $z \in A$. Moreover, none of the w_i are in $L(F) \cup \Sigma^{<n}$ because $w \notin L(F) \cup \Sigma^{<n}$ and $L(F) \cup \Sigma^{<n}$ is $\preceq$-closed. Thus the w_i are all anomalies, and there are at least $a + 1$ of them, contradicting the fact that M_e learns $\mathrm{SUBSEQ}(A)$ with $\leq a$ anomalies. Thus no such u exists. □

Proposition 3.10 and Theorems 3.11 and 3.26 together imply that we cannot replace a single mind change by any fixed finite number of anomalies. A stronger statement is true.

Theorem 3.27. $\mathrm{SUBSEQ\text{-}EX}_c \not\subseteq \mathrm{SUBSEQ\text{-}EX}^*_{c-1}$ *for any* $c > 0$.

Proof. Let $R_i = (0^*1^*)^i$ as in Definition 2.13, and define

$$\mathcal{R}_c = \left\{ A \subseteq \{0,1\}^* : \begin{array}{c} A \subseteq R_c \wedge \\ (A \text{ is } \preceq\text{-closed}) \wedge \\ (\exists j)[0 \leq j \leq c \wedge R_j \subseteq A \subseteq^* R_j] \end{array} \right\}.$$

Recall (Notation 2.12) that $A \subseteq^* B$ means that $A - B$ is finite.

We claim that $\mathcal{R}_c \in \mathrm{SUBSEQ\text{-}EX}_c - \mathrm{SUBSEQ\text{-}EX}^*_{c-1}$ for all $c > 0$.

To see that $\mathcal{R}_c \in \mathrm{SUBSEQ\text{-}EX}_c$, with $A \in \mathcal{R}_c$ on the tape the learner M first sets $i := c$ and may decrement i as the learning proceeds. For each i, the machine M proceeds on the assumption that $R_i \subseteq A$. For $n = 1, 2, 3, \ldots,$ M waits until $n \geq 2i + 2$ and there are no strings in $A - R_i$ of length n. At this point, we know that $A - R_i \subseteq \Sigma^{<n}$. (It is possible that $(01)^{i+1} \in A - R_i$ but no string of length $2i+1$ is in $A - R_i$, which is why we insisted that $n \geq 2i+2$. This is the only exception.) M now starts outputting a DFA for $R_i \cup (A \cap \Sigma^{<n})$. If M ever discovers a string in $R_i - A$, then M resets $i := i-1$ and starts over. Thus M can make at most c mind-changes before finding the unique j such that $R_j \subseteq A \subseteq^* R_j$.

To show that $\mathcal{R}_c \notin \text{SUBSEQ-EX}^*_{c-1}$ we use a (by now) standard diagonalization. Given a learner M, we build A such that $A \cap \Sigma^{<n} = R_c \cap \Sigma^{<n}$ for increasing n until M outputs some DFA F such that $L(F) \triangle R_c$ is finite while only querying strings of length less than n. We then make A look like R_{c-1} on strings of length $\geq n$ until M outputs a DFA G such that $L(G) \triangle R_{c-1}$ is finite. We then make A look like R_{c-2} above the queries made by M so far, et cetera. In the end, M clearly must make at least c mind-changes to be right within a finite number of anomalies. We can make A decidable uniformly in c and a Turing machine index for M. ☐

Although we don't get any trade-offs between anomalies and mind-changes, we *do* get trade-offs between *anomaly revisions* and mind-changes. If a learner is allowed to revise its bound on allowed anomalies from time to time, then we can trade these revisions for mind-changes. The proper setting for considering anomaly revisions is that of transfinite anomalies, which we consider next.

Transfinite anomalies and mind-changes This section uses some of the concepts introduced in the section on transfinite mind-changes, above. If you skipped that section, then you may skip this one, too.

We get a trade-off between anomalies and mind-changes if we consider the notion of transfinite anomalies, which we now describe informally. Suppose we have a learner M with a language A on its tape and some constructive ordinal $\alpha < \omega_1^{\text{CK}}$ initially on its ordinal tape, and suppose that M can decrease its ordinal any time it wants to (it is not forced to by mind-changes). We say that *M learns* SUBSEQ(A) *with α anomalies* if M's final DFA F and final ordinal β are such that $|L(F) \triangle \text{SUBSEQ}(A)| \leq N(\beta)$. For example, if M starts out with $\omega + \omega$ on its ordinal tape, then at some point after examining A and making conjectures, M may tentatively decide that it can find SUBSEQ(A) with at most 17 anomalies. It then decreases its ordinal to $\omega + 17$ ($N(\omega + 17) = 17$). Later, M may find that it really needs 500 anomalies. It can then decrease its ordinal a second time from $\omega + 17$ to 500. M is now committed to at most 500 anomalies, because it cannot further increase its allowed anomalies by decreasing its ordinal.

More generally, if M starts with the ordinal $\omega \cdot n + k$ for some $n, k \in \mathbb{N}$, then M is allowed k anomalies to start, and M can increase the number of allowed anomalies up to n many times.

There was no reason to introduce transfinite anomalies before, because the anomaly hierarchy collapses completely. Transfinite anomalies are nontrivial, however, when combined with limited mind-changes.

The next theorem generalizes Theorem 3.26 to the transfinite. It shows that a finite number of extra anomalies makes no difference.

Theorem 3.28. *Let $k, c \in \mathbb{N}$ and let $\lambda < \omega_1^{\text{CK}}$ be any limit ordinal. Then* $\text{SUBSEQ-EX}_c^{\lambda+k} = \text{SUBSEQ-EX}_c^{\lambda}$.

Proof. We show the $c = 0$ case; the general case is similar. Suppose M learns SUBSEQ(A) with $\lambda + k$ anomalies and no mind-changes. To learn SUBSEQ(A) with λ anomalies and no mind-changes, we first run the algorithm of Theorem 3.26 with λ initially on our ordinal tape and assuming $\leq k$ anomalies (i.e., setting $M_e := M$ and $a := k$). If M never drops its ordinal below λ, then this works fine. Otherwise, at some point, M drops its ordinal to some $\gamma < \lambda$. If this happens before we act—i.e., before we output anything—then we abandon the algorithm, drop our own ordinal to γ, and from now on simulate M directly. If the drop happens after we act, then M has already outputted some final DFA F and we have outputted some G recognizing $L(G) = \text{SUBSEQ}((A \cap \Sigma^n) \cup (L(F) \cap \Sigma^{\geq n}))$ for some n. Since $L(F) \cup \Sigma^{<n}$ is $\preceq$-closed, it follows that $L(G) \triangle L(F) \subseteq \Sigma^{<n}$ and hence is finite. So we compute $d := |L(G) \triangle L(F)|$, drop our ordinal from λ to $\gamma + d$, and keep outputting G forever. Whenever M drops its ordinal further to some δ, then we drop ours to $\delta + d$, etc. If ℓ is the final number of anomalies allowed by M, then we have

$$|L(G) \triangle \text{SUBSEQ}(A)| \leq |L(G) \triangle L(F)| + |L(F) \triangle \text{SUBSEQ}(A)| \leq d + \ell,$$

and so we have given ourselves enough anomalies. □

We show next that ω anomalies can be traded for an extra mind-change.

Theorem 3.29. *For all $c \in \mathbb{N}$ and $\lambda < \omega_1^{\text{CK}}$, if λ is zero or a limit, then*

$$\text{SUBSEQ-EX}_c^{\lambda+\omega} \subseteq \text{SUBSEQ-EX}_{c+1}^{\lambda}.$$

Proof. Suppose M learns SUBSEQ(A) with $\lambda + \omega$ anomalies and c mind-changes. With ordinal λ on our ordinal tape, we start out by simulating M exactly—outputting the same conjectures—until M drops its ordinal to some γ. If $\gamma < \lambda$, then we drop our ordinal to γ and keep simulating M forever. If $\gamma = \lambda + k$ for some $k \in \mathbb{N}$, then we immediately adopt the strategy in the proof of Theorem 3.28, above. Our first action after this point may constitute an extra mind-change, but that's okay because we have $c + 1$ mind-changes available. □

Corollary 3.30. $\text{SUBSEQ-EX}_c^{\omega \cdot n+k} \subseteq \text{SUBSEQ-EX}_{c+n}^0$ *for all $c, n, k \in \mathbb{N}$.*

Proof. By Theorems 3.26, 3.28, and 3.29. □

Next we show that the trade-off in Corollary 3.30 is tight.

Theorem 3.31. $\text{SUBSEQ-EX}_c^{\omega \cdot n} \not\subseteq \text{SUBSEQ-EX}_{c+n-1}$ *for any c and $n > 0$.*

Proof. Consider the classes $\mathcal{C}_i$ of Definition 3.9. By Theorem 3.11, $\mathcal{C}_{c+n} \notin \text{SUBSEQ-EX}_{c+n-1}$. We check that $\mathcal{C}_{c+n} \in \text{SUBSEQ-EX}_c^{\omega \cdot n}$. Given $A \in \mathcal{C}_{c+n}$ on the tape and $\omega \cdot n$ initially on its ordinal tape, the learner M outputs a DFA for SUBSEQ($A \cap \Sigma^{\leq i}$) as its ith output (as in Proposition 1.9) until it runs out of mind-changes. M continues outputting the same DFA, but every time it finds a new element $0^j \in A$ it revises its anomaly count to j. It can do this n times. □

This can be generalized to $\text{SUBSEQ-EX}_c^{\omega \cdot n} \not\subseteq \text{SUBSEQ-EX}_{c+x-1}^{\omega \cdot (n-x)}$ for any $n \in \mathbb{N}$ and $0 \leq x \leq n$, witnessed by the same class $\mathcal{C}_{c+n}$.

3.7 Mind-changes and teams

In this section we will consider teams of machines which have a bounded number of mind changes. All of the machines have the same bound. Recall the definition of consensus value from Lemma 3.21 as a value that shows up at least a times in the list of outputs at time t.

We will start with generalizations of Lemma 3.21.

Lemma 3.32. $[1,q]\text{SUBSEQ-EX}_c \subseteq [a,aq]\text{SUBSEQ-EX}_c$ *for every* $q,a \geq 1$ *and* $c \geq 0$.

Proof. This follows exactly the first part of the proof of Lemma 3.21. The proof doesn't involve any additional mind changes. ☐

Lemma 3.33. $[a,b]\text{SUBSEQ-EX}_c \subseteq [1,\lfloor b/a \rfloor]\text{SUBSEQ-EX}_{b(c+1)-1}$ *for every* $1 \leq a \leq b$ *and* $c \geq 0$.

Proof. This follows from the second part of the proof of Lemma 3.21. Here it is easier to consider counting conjectures rather than mind changes. Each of the machines $N_1, \ldots, N_q$ might make a new conjecture any time any one of the Q_i does, but not at any other time. Since each Q_i can make $c+1$ conjectures, each N_j can make $b(c+1)$ conjectures. Therefore it can make $b(c+1)-1$ mind changes. ☐

Notice that the previous two results do *not* give us that

$$[a,b]\text{SUBSEQ-EX}_c = [1,\lfloor b/a \rfloor]\text{SUBSEQ-EX}_c$$

as in Lemma 3.21.

Corollary 3.34. *If* $\frac{a}{b} > \frac{1}{2}$ *then* $[a,b]\text{SUBSEQ-EX}_c \subseteq \text{SUBSEQ-EX}_{b(c+1)-1}$.

Theorem 3.35. $\text{SUBSEQ-EX}_{q(c+1)-1} \subseteq [a,aq]\text{SUBSEQ-EX}_c$ *for all* $a,q \geq 1$ *and* $c \geq 0$.

Proof. Divide the aq team learners into q groups $G_1, \ldots, G_q$ of a learners each. Suppose we are given some learner M with some A on the tape. The first time M outputs a conjecture k_1, the machines in G_1 (and no others) start outputting k_1. The next time M changes its mind and outputs a new conjecture $k_2 \neq k_1$, only the machines in G_2 start outputting k_2, et cetera. This continues through the groups cyclically. All the machines in some group will eventually output the final DFA output by M. There are q groups, and so each team machine makes a $1/q$ fraction of the conjectures made by M. If M makes at most $q(c+1)-1$ mind-changes, then it makes at most $(c+1)q$ conjectures, and so each team machine makes at most $c+1$ conjectures with at most c mind-changes. ☐

From here on out, we will work with teams of the form $[1,b]$. The next two results complement each other.

Corollary 3.36. $\text{SUBSEQ-EX}_{b(c+1)-1} \subseteq [1,b]\text{SUBSEQ-EX}_c$ *for all* $b \geq 1$ *and* $c \geq 0$.

Theorem 3.37. $\text{SUBSEQ-EX}_{b(c+1)} \not\subseteq [1,b]\text{SUBSEQ-EX}_c$ *for any* $b \geq 1$ *and* $c \geq 0$.

Proof. We prove that $\mathcal{C}_{b(c+1)} \notin [1,b]\text{SUBSEQ-EX}_c$ by building a language $A \in \mathcal{C}_{b(c+1)}$ to diagonalize against all b machines. We start by leaving A empty until one of the machines conjectures a DFA for $\emptyset$. Then we add a string to A long enough so as not to disturb this conjecture. Whenever a machine conjectures a DFA for a finite language, we add an appropriately long string to A that is not in the conjectured language. After breaking the $b(c+1)$ conjectures, we will have added at most $b(c+1)$ elements to A, so it is in $\mathcal{C}_{b(c+1)}$. ☐

Theorem 3.38. *For all* $b \geq 1$, $[1,b]\text{SUBSEQ-EX}_0 \subseteq \text{SUBSEQ-EX}_{2b-2}$ *and* $[1,b]\text{SUBSEQ-EX}_c \subseteq \text{SUBSEQ-EX}_{2b(c+1)-3}$ *for all* $c \geq 1$.

Proof. We are given b machines team-learning SUBSEQ(A) and outputting at most $c+1$ conjectures each. For $n = 1, 2, 3, \ldots$ we output the DFA (if there is one) that recognizes the $\subseteq$-minimum language among the machines' past outputs that are consistent with the data so far. That is, for each n we output F iff

1. F is an output of one of the b machines running within n steps (not necessarily the most recent output of that machine),
2. $\text{SUBSEQ}(A \cap \Sigma^{\leq n}) \subseteq L(F)$ (that is, F is consistent with the data), and
3. $L(F) \subseteq L(G)$ for any G satisfying items 1 and 2 above.

We'll call such an F *good* (at time n). If a good F exists, it is clearly unique. If no good F exists, then we default (in the same sense as in the proof of Theorem 3.26). We can assume for simplicity that at most one of the b machines makes a new conjecture at a time.

Clearly, for all large enough n, the correct DFA will be good, and so we will eventually output it forever. To count the number of mind-changes we make, suppose that at some point our current conjecture is some good DFA F. We may change our mind away from F for one of two reasons:

finding an inconsistency: we've discovered that F is inconsistent with the data (violating item 2 above) and another good G exists, or

finding something better: F is still consistent, but a good G appears such that $L(G) \subset L(F)$.

Let $V = \{G_1, \ldots, G_m\}$ be the set of all DFAs that we output. We only make conjectures that the team machines make, so $m \leq b(c+1)$. Whenever we change our mind from some G_i to some G_j, we draw a directed edge $G_i \to G_j$ from G_i to G_j. We color this edge *red* if the mind-change results from finding G_i to be inconsistent, and we color it *blue* if the mind-change occurs because G_j is better than G_i. Note that $L(G_j) \subset L(G_i)$ if the edge is blue and $L(G_j) \not\subseteq L(G_i)$ if the edge is red. Let R be the set of red edges and B the set of blue edges. We'll say that the *red degree* of a vertex G_i is the *out*degree of G_i in the graph (V, R), and the *blue degree* of G_i is the *in*degree of G_i in the graph (V, B). Our total number of mind-changes is clearly $|R| + |B|$.

If we find an inconsistency with some G_i, then we never output G_i again. Thus each vertex in V has red degree at most 1. We never find an inconsistency with the correct team learner's final (correct) output, and so our last conjecture has red degree 0. We therefore have $|R| \leq m - 1$.

Suppose that we conjecture some G_i, change our mind at least once, then conjecture G_i again later. We claim that any conjecture G_j we make in the interim must satisfy $L(G_j) \subseteq L(G_i)$. This is because G_i is known and consistent with the data all during this time, so any good G_j must satisfy $L(G_j) \subseteq L(G_i)$ by the $\subseteq$-minimality of $L(G_j)$. It follows immediately from the claim that the return to G_i can only come from following a red edge, i.e., finding an inconsistency, for otherwise we would have $L(G_i) \subset L(G_j)$ (and thus $L(G_j) \not\subseteq L(G_i)$) for the last G_j conjectured before the return to G_i. From this it follows that each vertex in V has blue degree at most 1, and our very first conjecture has blue degree 0. Thus $|B| \leq m - 1$. Combining this with the bound on $|R|$ gives us at most $2m - 2 \leq 2b(c+1) - 2$ mind-changes. This is enough for the $c = 0$ case of the theorem.

Now assuming $c \geq 1$, we will shave off another mind-change. We are done if $|R| < m - 1$, so suppose $|R| = m - 1$. This can happen only if there is a unique vertex G_{fin}—our final conjecture—with red degree 0. Let G_{init} be our initial conjecture. If $G_{\text{init}} \neq G_{\text{fin}}$, then G_{init} has red degree 1, and so at some point we follow a red edge from G_{init} to some other H. Since $L(H) \not\subseteq L(G_{\text{init}})$, the claim implies that we have not conjectured H before, and so, also by the claim, H has blue degree 0 (because we first encounter H through a red edge). So we have two vertices (G_{init} and H) with blue degree 0, and thus $|B| \leq m - 2$, and we have at most $2m - 3 \leq 2b(c+1) - 3$ mind-changes.

Now suppose $G_{\text{init}} = G_{\text{fin}}$. Then it is possible that $|R| + |B| = 2m - 2$, but we will see that in this case, $m < b(c+1)$, and thus our algorithm still uses at most $2b(c+1) - 3$ mind-changes. Let M be one of the b team machines that eventually outputs the correct DFA, i.e., G_{fin}. If one of the b machines other than M outputs G_{fin}, or if M outputs G_{fin} at some point before changing its mind, then the b machines collectively make strictly fewer than $b(c+1)$ distinct conjectures, and so $m < b(c+1)$. So we can assume that G_{fin} appears only as the final conjecture made by M. We claim that V does not contain any other conjecture made by M except G_{fin}, which shows that $m < b(c+1)$. If M makes a conjecture $H \neq G_{\text{fin}}$, it does so before it outputs G_{fin}, and so we know about H when we first output $G_{\text{init}} = G_{\text{fin}}$. Assume that H is consistent at this time (otherwise we never output H, hence $H \notin V$). Since G_{fin} is good, we must have $L(G_{\text{fin}}) \subseteq L(H)$ by the $\subseteq$-minimality of G_{fin}. But if we ever output H later on, then we do so between outputting G_{fin} initially and G_{fin} finally, and so it follows from the previous claim that $L(H) \subseteq L(G_{\text{fin}})$. Then we have $L(H) = L(G_{\text{fin}})$, and so $H = G_{\text{fin}}$, a contradiction. Thus we never output H, which proves the claim and the theorem. ◻

Theorem 3.38 is tight.

Theorem 3.39. *For all $b > 1$, $[1,b]$SUBSEQ-EX$_0 \not\subseteq$ SUBSEQ-EX$_{2b-3}$ and $[1,b]$SUBSEQ-EX$_c \not\subseteq$ SUBSEQ-EX$_{2b(c+1)-4}$ for all $c \geq 1$.*

Proof. We'll only prove the case where $c \geq 1$. The $c = 0$ case is easier and only slightly different.

Let $f : \mathbb{N}^+ \to \mathbb{N}$ be any map. For any $j \in \mathbb{N}$, define a *j-bump* of f to be any nonempty, finite, maximal interval $[x, y] \subseteq \mathbb{N}^+$ such that $f(t) > j$ for all $x \leq t \leq y$. Define the language

$$A_f := \{(0^t 1^t)^{f(t)} : t \in \mathbb{N}^+\}.$$

Observe that, if $\limsup_{t\to\infty} f(t) = \ell < \infty$, then f has finitely many ℓ-bumps and $R_\ell \subseteq \text{SUBSEQ}(A_f) \subseteq^* R_\ell$, where $R_\ell = (0^*1^*)^\ell$ as in Definition 2.13.

Now fix $b > 1$ and $c \geq 1$. We say that f is *good* if

- $f(1) = b$ and $0 \leq f(t) \leq b$ for all $t \geq 1$,
- f has at most c many 0-bumps,
- f has at most $c+1$ many ℓ-bumps, where $\ell = \limsup_t f(t)$, and
- if $(\exists t)[f(t) = 0]$ then $\limsup_t f(t) \leq b - 1$.

We define the class

$$\mathcal{T}_{b,c} := \{A_f : f \text{ is good}\},$$

and show that $\mathcal{T}_{b,c} \in [1,b]\text{SUBSEQ-EX}_c - \text{SUBSEQ-EX}_{2b(c+1)-4}$.

To see that $\mathcal{T}_{b,c} \in [1,b]\text{SUBSEQ-EX}_c$, we define learners $Q_1, \ldots, Q_b$ acting as follows with A_f on their tapes for some good f: Each learner examines its tape enough to determine $f(1), f(2), \ldots$. For $1 \leq j \leq b-1$, learner Q_j goes on the assumption that $\limsup_t f(t) = j$. Each time it notices a new j-bump $[x, y]$ of f, it assumes that $[x, y]$ is the last j-bump it will see and so starts outputting a DFA for

$$R_j \cup \text{SUBSEQ}(A_f \cap \{(0^t 1^t)^k : t \leq y \wedge k \leq b\}).$$

which captures all the elements of $A_f - R_j$ seen so far. Let $\ell = \limsup_t f(t)$. If $1 \leq \ell \leq b-1$, then Q_ℓ will see at most $c+1$ many ℓ-bumps of f and so make at most $c+1$ conjectures, the last one being correct.

The learner Q_b behaves a bit differently: It immediately starts outputting the DFA for R_b, and does this until it (ever) finds a t with $f(t) = 0$. It then proceeds on the assumption that $\limsup_t f(t) = 0$ and acts similarly to the other learners. Again, let $\ell = \limsup_t f(t)$. Since f is good, if there is a t such that $f(t) = 0$, then $\ell \leq b - 1$ and so all possible values of ℓ are covered by the learners. If $\ell = 0$, then since there are only c many 0-bumps, Q_b will be correct after at most $c + 1$ conjectures. If $\ell = b$, then $\text{SUBSEQ}(A_f) = R_b$, and since f is good, Q_b will never revise its initial conjecture of R_b. This establishes that $\mathcal{T}_{b,c} \in [1, b]\text{SUBSEQ-EX}_c$.

To show that $\mathcal{T}_{b,c} \notin \text{SUBSEQ-EX}_{2b(c+1)-4}$, let M be a learner that correctly learns $\text{SUBSEQ}(A_f)$ for every good f. We now describe a particular good f that forces M to make at least $2b(c+1) - 3$ mind-changes.

For $t = 1, 2, 3, \ldots$, we first let $f(t) = b$ until M outputs a DFA for R_b. Then we make $f(t) = b-1$ until M outputs a DFA F such that $R_{b-1} \subseteq L(F) \subseteq^* R_{b-1}$, at which point we start making $f(t) = b$ again, et cetera. The value of $f(t)$ alternates between b and $b - 1$, forcing a mind-change each time, until $f(t) = b - 1$ and there are $c + 1$ many $(b - 1)$-bumps of f. Then f starts alternating between $b - 1$ and $b - 2$ in a similar fashion until there are $c + 1$ many $(b - 2)$-bumps, et cetera. These alternations continue until $f(t) = 0$ and there are c many 0-bumps of f included in the interval $[1, t]$. Thus far, M has needed to make $2c + 1$ many conjectures for each of the first $b - 1$ many alternations, plus $2c$ conjectures for the $1, 0$ alternation, for a total of $(b-1)(2c+1)+2c = 2bc+b-1$ many conjectures.

Now we let $f(t)$ slowly increase from 0 through to $b-1$, forcing a new conjecture with each step, until we settle on $b-1$. This adds $b-1$ more conjectures for a grand total of $2bc+2(b-1) = 2b(c+1)-2$ conjectures, or $2b(c + 1) - 3$ mind-changes. □

3.8 All three modifications

Finally, we consider teams of machines which are allowed to have anomalies, but have a bounded number of mind changes.

Theorem 3.40. $[a, b]\text{SUBSEQ-EX}^k_c \subseteq [a, b]\text{SUBSEQ-EX}_c$ *for all* $c, k \geq 0$ *and* $1 \leq a \leq b$.

Proof. This follows from the proof of Theorem 3.26. We apply the algorithm there to each of the b machines. □

4 Rich classes

Are there classes in SUBSEQ-EX containing languages of arbitrary complexity? Yes, trivially.

Proposition 4.1. *There is a* $\mathcal{C} \in \text{SUBSEQ-EX}_0$ *such that for all* $A \subseteq \mathbb{N}$, *there is a* $B \in \mathcal{C}$ *with* $B \equiv_{\text{T}} A$.

Proof. Let

$$\mathcal{C} = \{A \subseteq \Sigma^* : |A| = \infty \wedge (\forall x, y \in \Sigma^*)[x \in A \wedge |x| = |y| \to y \in A]\}.$$

That is, $\mathcal{C}$ is the class of all infinite languages, membership in whom depends only on a string's length.

For any $A \subseteq \mathbb{N}$, define

$$L_A = \begin{cases} \Sigma^* & \text{if } A \text{ is finite,} \\ \bigcup_{n \in A} \Sigma^{=n} & \text{otherwise.} \end{cases}$$

Clearly, $L_A \in \mathcal{C}$ and $A \equiv_{\mathrm{T}} L_A$. Furthermore, $\mathrm{SUBSEQ}(L_A) = \Sigma^*$, and so $\mathcal{C} \in \mathrm{SUBSEQ\text{-}EX}_0$ witnessed by a learner that always outputs a DFA for Σ^*. □

In Proposition 1.10 we showed that REG $\in$ SUBSEQ-EX. Note that the $A \in$ REG are trivial in terms of computability, but the languages in SUBSEQ(REG) can be rather complex (large obstruction sets, arbitrary $\preceq$-closed sets). By contrast, in Proposition 4.1, we show that there can be $\mathcal{A} \in$ SUBSEQ-EX of arbitrarily high Turing degree but SUBSEQ($\mathcal{A}$) is trivial. Can we obtain classes $\mathcal{A} \in$ SUBSEQ-EX where $A \in \mathcal{A}$ has arbitrary Turing degree and SUBSEQ($\mathcal{A}$) has arbitrary $\preceq$-closed sets independently? Of course, if SUBSEQ(A) is finite, then A must be finite and hence computable. Aside from this obvious restriction, the answer to the above question is yes.

Definition 4.2. A class $\mathcal{C}$ of languages is *rich* if for every $A \subseteq \mathbb{N}$ and $\preceq$-closed $S \subseteq \Sigma^*$, there is a $B \in \mathcal{C}$ such that $\mathrm{SUBSEQ}(B) = S$ and, provided A is computable or S is infinite, $B \equiv_{\mathrm{T}} A$.

Definition 4.3. Let $\mathcal{G}$ be the class of all languages $A \subseteq \Sigma^*$ for which there exists a length $c = c(A) \in \mathbb{N}$ (necessarily unique) such that

1. $A \cap \Sigma^{=c} = \emptyset$,
2. $A \cap \Sigma^{=n} \neq \emptyset$ for all $n < c$, and
3. $os(A) = os(A \cap \Sigma^{\leq c+1}) \cap \Sigma^{\leq c}$.

We'll show that $\mathcal{G} \in \mathrm{SUBSEQ\text{-}EX}_0$ and that $\mathcal{G}$ is rich.

Proposition 4.4. $\mathcal{G} \in \mathrm{SUBSEQ\text{-}EX}_0$.

Proof. Consider a learner M acting as follows with a language A on its tape:

1. Let c be least such that $A \cap \Sigma^{=c} = \emptyset$ (assuming c exists).
2. Compute $O = os(A \cap \Sigma^{\leq c+1}) \cap \Sigma^{\leq c}$. (If $A \in \mathcal{G}$, then $O = os(A)$ by definition.)
3. Use O to compute the least index k such that $L(F_k)$ is $\preceq$-closed and $os(L(F_k)) = O$. (If $A \in \mathcal{G}$, then we have $L(F_k) = \mathrm{SUBSEQ}(A)$, because $O = os(A) = os(\mathrm{SUBSEQ}(A))$.)
4. Output k repeatedly forever.

It is evident that M learns every language in $\mathcal{G}$ with no mind-changes. □

The next few propositions show that $\mathcal{G}$ is big enough.

Definition 4.5. Let $S \subseteq \Sigma^*$ be any $\preceq$-closed set.

1. Say that a string x is *S-special* if $x \in S$ and $S \cap \{y \in \Sigma^* : x \preceq y\}$ is finite.
2. Say that a number $n \in \mathbb{N}$ is an *S-coding length* if $n > |y|$ for all S-special y and $n \geq |z|$ for all $z \in os(S)$.

The next proposition implies that S-coding lengths exist for any S.

Proposition 4.6. *Any $\preceq$-closed S contains only finitely many S-special strings.*

Proof. This follows from the fact, first proved by Higman [17], that $(\Sigma^*, \preceq)$ is a well-quasi-order (wqo). That is, for any infinite sequence $x_1, x_2, \ldots$ of strings, there is some $i < j$ such that $x_i \preceq x_j$.

A standard result of well-quasi-order theory, proved using techniques from Ramsey theory, gives a stronger fact: Every infinite sequence $x_1, x_2, \ldots$ of strings contains an infinite monotone subsequence

$$x_{i_1} \preceq x_{i_2} \preceq \cdots,$$

where $i_1 < i_2 < \cdots$.

Suppose that some S has infinitely many S-special strings $s_1, s_2, \ldots$ with all the s_i distinct. Then S includes an infinite monotone subsequence $s_{i_1} \prec s_{i_2} \prec \cdots$ of S-special strings, but then s_{i_1} clearly cannot be S-special. Contradiction. □

Corollary 4.7. *S-coding lengths exist for any $\preceq$-closed S.*

Definition 4.8. Let $\mathcal{G}'$ be the class of all $A \subseteq \Sigma^*$ that have the following properties (setting $S = \mathrm{SUBSEQ}(A)$):

1. A contains all S-special strings, and
2. there exists a (necessarily unique) S-coding length c for which the following hold:
   (a) $A \cap \Sigma^{=c} = \emptyset$,
   (b) $A \cap \Sigma^{=n} \neq \emptyset$ for all $n < c$, and
   (c) $A \cap \Sigma^{=c+1} = S \cap \Sigma^{=c+1}$.

Proposition 4.9. $\{S \subseteq \Sigma^* : S$ *is $\preceq$-closed and finite*$\} \subseteq \mathcal{G}' \subseteq \mathcal{G}$.

Proof. For the first inclusion, it is easy to check that the criteria of Definition 4.8 hold for any finite $\preceq$-closed S if we let c be least such that $S \subseteq \Sigma^{<c}$.

For the second inclusion, suppose $A \in \mathcal{G}'$, and let c satisfy the conditions of Definition 4.8 for A. It remains to show that

$$os(A) = os(A \cap \Sigma^{\leq c+1}) \cap \Sigma^{\leq c}. \tag{4}$$

Set $S = \mathrm{SUBSEQ}(A)$. Since c is an S-coding length, we have $os(A) = os(S) \subseteq \Sigma^{\leq c}$.

Let x be some string in $os(A)$. Then $x \notin S$, but $y \in S$ for every $y \prec x$. Consider any $y \prec x$.

- If y is S-special, then $y \in A$ (since A contains all S-special strings), and since $|y| < |x| \leq c$, we have $y \in A \cap \Sigma^{\leq c+1}$.
- If y is not S-special, then there are arbitrarily long $z \in S$ with $y \preceq z$. In particular there is a $z \in S \cap \Sigma^{=c+1}$ such that $y \preceq z$. But then $z \in A \cap \Sigma^{=c+1}$ (because $A \in \mathcal{G}'$), which implies $y \in \mathrm{SUBSEQ}(A \cap \Sigma^{\leq c+1})$.

In either case, we have shown that $x \notin \mathrm{SUBSEQ}(A \cap \Sigma^{\leq c+1})$, but $y \in \mathrm{SUBSEQ}(A \cap \Sigma^{\leq c+1})$ for every $y \prec x$. This means exactly that $x \in os(A \cap \Sigma^{\leq c+1})$, and since $|x| \leq c$, we have the forward containment in (4).

Conversely, suppose that $|x| \leq c$ and $x \in os(A \cap \Sigma^{\leq c+1})$. Then $x \notin A \cap \Sigma^{\leq c+1}$ but $(\forall y \prec x)(\exists z \in A \cap \Sigma^{\leq c+1})[y \preceq z]$. Thus, $x \notin A$ but $(\forall y \prec x)(\exists z \in A)[y \preceq z]$. That is, $x \in os(A)$. □

Theorem 4.10. *$\mathcal{G}'$ is rich. In fact, there is a learner M such that M learns every language in $\mathcal{G}'$ without mind-changes, and for every A and infinite S, M learns some $B \in \mathcal{G}'$ satisfying Definition 4.2 while also writing the characteristic function of A on a separate one-way write-only output tape.*

Proof. Given A and S as in Definition 4.2, we define

$$L(A, S) := S \cap \left(\Sigma^{<c} \cup \bigcup_{n \in \mathbb{N}} \Sigma^{=c+2n+1} \cup \bigcup_{n \in A} \Sigma^{=c+2n+2} \right), \tag{5}$$

where c is the least S-coding length.

Set $B = L(A, S)$, and let c be the least S-coding length.

We must first show that $S = \text{SUBSEQ}(B)$, from which it will follow easily that $B \in \mathcal{G}'$. We have two cases: S is finite or S is infinite. First suppose that S is finite. Then every string in S is S-special, and so by the definition of S-coding length, we have $S \subseteq \Sigma^{<c}$. Thus we clearly have $B = S = \text{SUBSEQ}(B) \in G'$ by Proposition 4.9. Now suppose S is infinite. Since $B \subseteq S$ and S is $\preceq$-closed, it suffices to show that $S \subseteq \text{SUBSEQ}(B)$. Let x be any string in S.

- If x is S-special, then $x \in \Sigma^{<c}$, by the definition of S-coding length. It follows that $x \in B$, and so $x \in \text{SUBSEQ}(B)$.
- If x is not S-special, then there is a string $z \in S$ such that $x \preceq z$ and $|z| \geq c + 2|x| + 1$. By removing letters one at a time from z to obtain x, we see that at some point there must be a string y such that $x \preceq y \preceq z$ and $|y| = c + 2|x| + 1$. Thus $y \in S$, and, owing to its length, $y \in B$ as well. Therefore we have $x \in \text{SUBSEQ}(B)$.

Now that we know that $S = \text{SUBSEQ}(B)$, it is straightforward to verify that $B \in \mathcal{G}'$. We've already shown this when S is finite. Suppose S is infinite. We showed above that B contains all S-special strings. The value c clearly satisfies the rest of Definition 4.8. For example, because S has strings of every length, we have $B \cap \Sigma^{=n} = S \cap \Sigma^{=n} \neq \emptyset$ for all $n < c$.

It is immediate by the definition that $B \leq_{\text{T}} A$, because S is regular. We now describe the learner M, which will witness that $A \leq_{\text{T}} B$ as well, provided S is infinite. M behaves exactly as in the proof of Proposition 4.4, except that for $n = 0, 1, 2, \ldots$ in order, M appends a 1 to the string on its special output tape if $B \cap \Sigma^{=c+2n+2} \neq \emptyset$, and it appends a 0 otherwise. If S is infinite, then S contains strings of every length, and so M will append a 1 for n if and only if $n \in A$. (If S is finite, then M will write all zeros.) ☐

Corollary 4.11. *$\mathcal{G}$ is rich.*

5 Open questions

We have far from fully explored the different ways we can combine teams, mind-changes, and anomalies. For example, for which a, b, c, d, e, f, g is $[a, b]\text{SUBSEQ-EX}_c^d \subseteq [e, f]\text{SUBSEQ-EX}_g^h$? This problem has been difficult in the standard case of EX, though there have been some very interesting results [10, 5]. The setting of SUBSEQ-EX may be easier since all the machines that are output are total and their languages have easily discernible properties.

One could also combine the two notions of queries with SUBSEQ-EX and its variants. The two notions are allowing queries *about the set* [15, 13, 11] and allowing queries *to an undecidable set* [8, 18].

6 Acknowledgments

The authors would like to thank Walid Gomma and Semmy Purewal for proofreading and helpful discussions.

References

1. G. Baliga and J. Case. Learning with higher order additional information. In *Proc. 5th Int. Workshop on Algorithmic Learning Theory*, pages 64–75. Springer-Verlag, 1994.
2. L. Blum and M. Blum. Towards a mathematical theory of inductive inference. *Information and Control*, 28:125–155, 1975.
3. J. Case, S. Jain, and S. N. Manguelle. Refinements of inductive inference by Popperian and reliable machines. *Kybernetika*, 30–1:23–52, 1994.
4. J. Case and C. H. Smith. Comparison of identification criteria for machine inductive inference. *Theoretical Computer Science*, 25:193–220, 1983.
5. R. Daley, B. Kalyanasundaram, and M. Velauthapillai. Breaking the probability 1/2 barrier in FIN-type learning. *Journal of Computer and System Sciences*, 50:574–599, 1995.
6. S. Fenner and W. Gasarch. The complexity of learning SUBSEQ(A). In *Proc. 17th Int. Conference on Algorithmic Learning Theory*, pages 109–123. LNAI 4264, Springer-Verlag, 2006.
7. S. Fenner, W. Gasarch, and B. Postow. The complexity of finding SUBSEQ(A), 2006. Accepted by *Theory of Computing Systems*.
8. L. Fortnow, S. Jain, W. Gasarch, E. Kinber, M. Kummer, S. Kurtz, M. Pleszkoch, T. Slaman, F. Stephan, and R. Solovay. Extremes in the degrees of inferability. *Annals of Pure and Applied Logic*, 66:231–276, 1994.
9. R. Freivalds and C. H. Smith. On the role of procrastination for machine learning. *Information and Computation*, 107(2):237–271, 1993.
10. R. Freivalds, C. H. Smith, and M. Velauthapillai. Trade-off among parameters affecting inductive inference. *Information and Computation*, 82(3):323–349, Sept. 1989.
11. W. Gasarch, E. Kinber, M. Pleszkoch, C. H. Smith, and T. Zeugmann. Learning via queries, teams, and anomalies. *Fundamenta Informaticae*, 23:67–89, 1995. Prior version in *Computational Learning Theory (COLT)*, 1990.
12. W. Gasarch and A. Lee. Inferring answers to queries. To appear in *Journal of Computer and System Sciences*. Proir version in *Proceedings of* 10^{th} *Annual ACM Conference on Computational Learning Theory*, pages 275–284, 1997.
13. W. Gasarch, M. Pleszkoch, and R. Solovay. Learning via queries to $[+, <]$. *Journal of Symbolic Logic*, 57(1):53–81, Mar. 1992.
14. W. Gasarch, M. Pleszkoch, F. Stephan, and M. Velauthapillai. Classification using information. *Annals of Math and AI*, pages 147–168, 1998. Earlier version in *Proc. 5th Int. Workshop on Algorithmic Learning Theory*, 1994, 290–300.
15. W. Gasarch and C. H. Smith. Learning via queries. *Journal of the ACM*, 39(3):649–675, July 1992. Prior version in *IEEE Sym. on Found. of Comp. Sci. (FOCS)*, 1988.
16. E. M. Gold. Language identification in the limit. *Information and Control*, 10(10):447–474, 1967.
17. A. G. Higman. Ordering by divisibility in abstract algebras. *Proc. of the London Math Society*, s3–2(1):326–336, 1952.
18. M. Kummer and F. Stephan. On the structure of the degrees of inferability. *Journal of Computer and System Sciences*, 52(2):214–238, 1996. Prior version in *Sixth Annual Conference on Computational Learning Theory (COLT)*, 1993.
19. H. Rogers. *Theory of Recursive Functions and Effective Computability.* McGraw-Hill, 1967. Reprinted by MIT Press, 1987.
20. G. E. Sacks. *Higher Recursion Theory.* Perspectives in Mathematical Logic. Springer-Verlag, Berlin, 1990.
21. Carl H. Smith. The power of pluralism for automatic program synthesis. *Journal of the ACM*, 29:1144–1165, 1982. Prior version in IEEE Sym on Found. of Comp. Sci. 1981 (FOCS)
22. R. I. Soare. *Recursively Enumerable Sets and Degrees.* Perspectives in Mathematical Logic. Springer-Verlag, Berlin, 1987.