Conditional Teaching Size

Manuel Garcia-Piqueras (✉)$^{1[0000-0001-8088-8393]}$ and
José Hernández-Orallo$^{2[0000-0001-9746-7632]}$

^{1} Math. Dept., Universidad de Castilla-La Mancha, Albacete, Spain
manuel.gpiqueras@uclm.es
^{2} VRAIN, Universitat Politècnica de València, València, Spain
jorallo@upv.es

Abstract. Recent research in machine teaching has explored the instruction of any concept expressed in a universal language. In this compositional context, new experimental results have shown that there exist data teaching sets surprisingly shorter than the concept description itself. However, there exists a bound for those remarkable experimental findings through *teaching size* and concept complexity that we further explore here. As concepts are rarely taught in isolation we investigate the *best* configuration of concepts to teach a given set of concepts, where those that have been acquired first can be reused for the description of new ones. This new notion of conditional teaching size uncovers new insights, such as the *interposition* phenomenon: certain prior knowledge generates simpler compatible concepts that increase the teaching size of the concept that we want to teach. This does not happen for conditional Kolmogorov complexity. Furthermore, we provide an algorithm that constructs *optimal curricula* based on interposition avoidance. This paper presents a series of theoretical results, including their proofs, and some directions for future work. New research possibilities in curriculum teaching in compositional scenarios are now wide open to exploration.

Keywords: Machine teaching · Kolmogorov complexity · Interposition · Curriculum

1 Introduction

Let us consider a *teacher* who instructs a given set of concepts by examples to a *learner*. Ideally, the teacher would design a curriculum such that the whole teaching session is shortest. For one concept, the field of *machine teaching* has analysed the efficiency of the teacher, the learner or both, for different representation languages and teaching settings [?,?,?,?,?].

For more than one concept, however, we need to consider different sequences of examples, or *curricula*, to make learning more effective. While there has been extensive experimental work in curriculum learning [?], the theoretical analysis is not abundant and limited to continuous models [?,?,?]. It is not well understood how curriculum learning can be optimised when concepts are *compositional*, with the underlying representation mechanisms being rich languages, even Turing-complete. Also, in a curriculum learning situation where a *teacher* chooses the examples sequentially, it is surprising that the connection with machine teaching has not been made explicit at a general conceptual level, with only a specific minimax approach for gradient-based representations [**?**,**?**,**?**,**?**]. In other words, to our knowledge, a theoretical framework has not yet been articulated for curriculum learning in machine teaching, or *curriculum teaching*, when dealing with universal languages, as a counterpart to incremental inductive inference based on simplicity [**?**,**?**].

While the teaching dimension has been the traditional metric for determining how easy it is to teach a concept [**?**], the *teaching size* [**?**] is a new metric that is more reasonably related to how easy it is to teach an infinite compositional concept class. It is also more appropriate to understand 'prompting' of language models as a kind of teaching, where users need to think of the shortest prompts that make a language model such as BERT, GPT-2 or GPT-3 achieve a task by few-shot learning [**?**,**?**,**?**]. However, as far as we know, the following issues are not clear yet: (1) *What is the relationship between the Kolmogorov complexity of a concept and how difficult it is to be taught under the teaching size paradigm?* and (2) *Is there a way to extend machine teaching, and teaching size in particular, to consider the notion of* optimal teaching curricula*?*

The first question prompts us to our first group of contributions: important theoretical results related to teaching size bounds. Theorem 1 shows that concepts with *high* complexity are *difficult to teach*, putting a limit to the surprising experimental finding recently reported in [**?**], where teaching a concept by examples was usually more economical (in total number of bits) than showing the shortest program for the concept. This connection between teaching size and complexity suggests that the second question may rely on a strong relation between incremental learning using simplicity priors and curriculum teaching.

For instance, consider the concepts c_+ for addition, $c_\times$ for multiplication, $c_\wedge$ for exponentiation and c_θ for the removal of zeros (Fig. 1). If the concept of c_+ is useful to allow for a shorter description of $c_\times$, is it also reasonable to expect that c_+ would also be useful to *teach* $c_\times$ from examples? Or even $c_\wedge$? In general, is the conditional algorithmic complexity $K(c_2|c_1)$ related to the minimal size of the examples needed to teach c_2 after having acquired c_1?

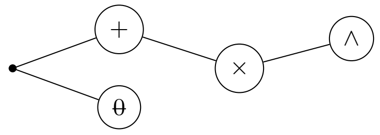

Fig. 1. Curriculum teaching for a set of concepts.

Our perspective studies the sequence of learning a set of concepts, instead of learning a sequence of instances under the same concept. In the general case, we define a teaching curriculum as a set of partial alternative sequences, such as the top and bottom branches in Fig. 1. The order between branches is irrelevant, but the order of concepts inside each branch is crucial. This tree structure is proposed as future work in [?]. Given a set of concepts, is there a curriculum that minimises the overall teaching size?

Our second group of contributions turns around this new concept of teaching curriculum. We provide a definition of *conditional teaching size*, given some other concepts already taught, $TS(c|c_1, \ldots, c_n)$. We show that, in general, $K(c_1|c_2) < K(c_2|c_1)$, for conditional Kolmogorov complexities, does not imply $TS(c_1|c_2) < TS(c_2|c_1)$, and vice versa. Furthermore, given a concept c, it is not true that $TS(c|B) \leq TS(c)$, $\forall B$. We find a new *interposition* phenomenon: acquired concepts may increase the teaching size of new concepts. We give conditions to avoid or provoke interposition. Theorems 3 and 4 are key results in this direction, providing an explicit range where interposition might happen. Finally, we present an effective procedure, $\mathbb{I}$-*search*, to design *optimal* curricula, minimising overall teaching size, for a given set of concepts.

We will see these results in the following sections, and interpret their relevance in the context of other machine teaching paradigms. This opens up new avenues for understanding curriculum learning, and new strategies for defining incremental teaching protocols for several applications, especially in compositional scenarios and language model 'prompting'.

2 Notation and background

Let us consider a machine M and a universal (i.e., Turing complete) language L. We assume that L is formed by a finite set of instructions in an alphabet Υ, each of them been coded with the same number of bits. Hence, each program p in language L can simply be represented as a string in $\Sigma = \{0,1\}^*$, whose length is denoted by $\dot{\ell}(p)$ (in number of instructions) and denoted by $\ell(p)$ (in bits). There is a total order, $\prec$, over programs in language L defined by two criteria: (i) length and (ii) lexicographic order over Υ only applied when two programs have equal size. Programs map binary strings in Σ to $\Sigma \cup \bot$, denoted by $p(\mathtt{i}) = \mathtt{o}$, with $p(\mathtt{i}) =\bot$ representing that p does not halt for $\mathtt{i}$. Two programs are equivalent if they compute the same function.

We say that c is an L-concept if it is a total or partial function $c : \Sigma \to \Sigma \cup \bot$ computed by at least a program in language L. The class of concepts defined by all programs in L is denoted by C_L; $[p]_L$ denotes the equivalence class of program p. Given $c \in C_L$, we denote $[c]_L$ as the equivalence class of programs in L that compute the function defined by c. Examples are just pairs of strings, and their space is the infinite set $X = \{\langle \mathtt{i}, \mathtt{o} \rangle : \langle \mathtt{i}, \mathtt{o} \rangle \in \Sigma \times (\Sigma \cup \bot)\}$. A *witness* can be any finite example subset of X, of the form $S = \{\langle \mathtt{i}_1, \mathtt{o}_1 \rangle, \ldots, \langle \mathtt{i}_k, \mathtt{o}_k \rangle\}$. In order to calculate the *size* of these sets, we consider self-delimiting codes. Let δ be the number of bits needed to encode S, using certain prefix code. For instance, if we consider Elias coding [?], the string 01010010001001000101 (size = 20) expresses the example set $\{\langle 1, 010 \rangle, \langle 0, 1 \rangle\}$ unambiguously. The size of an example set is the size of its encoding (e.g., $\delta(\{\langle 1, 010 \rangle, \langle 0, 1 \rangle\} = 20$ in Elias coding). For output strings, the natural number to be encoded is increased by 1, to accommodate for $\bot$. We also define a total order $\lessdot$ on X, i.e., $\forall S, S'$ such that $S \lessdot S'$ then $\delta(S) \leq \delta(S')$ with any preference (e.g., lexicographic) for equal size.

A concept c defines a unique subset of the example space X and we call any element in that subset a *positive* example. A concept c satisfies example set S, denoted by $c \vDash S$, if S is a subset of the positive examples of c. For instance, a witness set for the concept $c_{\mathtt{0}}$ (*remove zeros*) is $\{\langle 10011, 111\rangle, \langle 001, 1\rangle\}$. Example sets cannot have different outputs for equal inputs: $\{\langle 1, 00\rangle, \langle 1, 01\rangle\}$ is not valid.

A program p is compatible with $S = \{\langle \mathtt{i}_j, \mathtt{o}_j\rangle\}_{j=1}^{k} \subset X$, denoted by $p \vDash S$, if $p_S(\mathtt{i}_j) = \mathtt{o}_j$ for every $j \in \{1, \ldots, k\}$. For a finite example set S, there is always a program, denoted by $\ddot{p_S}$, that implements a conditional hard-coded structure of if-then-elses (trie) specifically designed for S. If we know the number of bits of input $\mathtt{i}$ and the set of examples in S, the number of comparisons using a trie-data structure is linearly time-bounded. Namely, for any $\ddot{p_S}$, there exists a constant, ρ, such that $\rho \cdot \min\{\ell(\mathtt{i}), \ell(\mathtt{i}_{max})\} + \ell(\mathtt{o}_{max})$ is an upper bound of time steps for each input $\mathtt{i}$, where $\ell(\mathtt{i}_{max})$, $\ell(\mathtt{o}_{max})$ are the lengths of the longest input string and output string in S, respectively. In general, for any program that employs a trie-data structure for S, there exists a time-bound linear function, denoted by $\lambda_L(\mathtt{i}, S)$, that represents an upper bound in time steps on every input $\mathtt{i}$.

Complexity functions $\mathsf{f} : \mathbb{N} \to \mathbb{N}$ act as time bounds. We say that a program p is f-compatible with the example set $S = \{\langle \mathtt{i}_j, \mathtt{o}_j\rangle\}_{j=1}^{k} \subset X$, denoted by $p \vDash_{\mathsf{f}} S$, if $p(\mathtt{i}_j) = \mathtt{o}_j$ within $\max\{\mathsf{f}(\ell(\mathtt{i}_j)), \lambda_L(\mathtt{i}_j, S)\}$ time steps (time-bound) for each $j \in \{1, \ldots, k\}$. In other words, within time bound, for each pair $\langle \mathtt{i}, \mathtt{o}\rangle \in S$ the program p on input $\mathtt{i}$: (1) outputs $\mathtt{o}$ when $\mathtt{o} \neq \bot$ or (2) does not halt when $\mathtt{o} = \bot$. Note that: (i) For any complexity function f and any example set S, there is always[3], a program f-compatible with S, (ii) there may be programs p such that $p \nvDash_{\mathsf{f}} S \wedge p \vDash S$, if f and S do not guarantee enough time bound and (iii) larger complexity functions distinguish more programs.

3 Absolute teaching size and complexity

Now we can study how a non-incremental teacher-learner setting works and the relationship between teaching size and Kolmogorov complexity.

Following the teaching settings based on the K-dimension [**?**,**?**], seen as preference-based teaching using simplicity priors [**?**,**?**], we assume that the learner is determined to find the shortest program (according to the prior $\prec$). Namely, the learner Φ returns the first program, in order $\prec$, for an example set S and a complexity function f as follows:

$$\Phi_{\ell}^{\mathsf{f}}(S) = \arg\min_{p}^{\prec} \{\ell(p) : p \vDash_{\mathsf{f}} S\}$$

Note that the f-bounded Kolmogorov complexity of an example set S, $K^{\mathsf{f}}(S)$, is the length of the program returned by the learner $K^{\mathsf{f}}(S) = \ell(\Phi_{\ell}^{\mathsf{f}}(S))$. We say that S is a *witness set* of concept c for learner Φ if S is a finite example set such that $p = \Phi_{\ell}^{\mathsf{f}}(S)$ and $p \in [c]_L$.

[3] Note that this $\ddot{p}_S$ is ensured by the max with time costs.

The teacher selects the *simplest* witness set that allows the learner to identify the concept, according to set size (δ) and associated total order $\lessdot$, as follows:

$$\Omega_\ell^{\mathsf{f}}(c) = \arg\min_{S}^{\lessdot} \left\{\delta(S) : \Phi_\ell^{\mathsf{f}}(S) \in [c]_L)\right\}$$

The K^{f}-teaching size of a concept c is $TS_\ell^{\mathsf{f}}(c) = \delta(\Omega_\ell^{\mathsf{f}}(c))$.

Every program the teacher picks defines a concept c. The teacher-learner protocol is computable for any complexity function f and able to create pairs (p_c, w_c), where p_c defines a concept c and w_c is a witness set of c. We can think of these pairs as if they were inserted sequentially in the so-called f-*Teaching Book* ordered by w_c, with no repeated programs or witness sets. For example, if we consider the concept $a \in C_L$ for swapping ones and zeros in a binary string, there will be a pair (p_a, w_a) in the f-Teaching Book, e.g., containing a witness set like $w_a = \{\langle 10, 01\rangle, \langle 110, 001\rangle\}$ that the teacher would provide with which the learner would output p_a, a program that swaps 1 and 0. Theorem 1 in [?] shows that *for any concept* $c \in C_L$*, there exists a complexity function* f *such that there is a pair* (p_c, w_c) *in the* f*-Teaching Book*. The teaching size makes more sense than the traditional teaching dimension (the smallest cardinality of a witness set for the concept) because some concepts could be taught by very few examples, but some of them could be extremely large. Also, the use of size instead of cardinality allows us to connect teaching size and Kolmogorov complexity, as we do next.

Our first result shows an equipoise between teaching size and data compression, an extra support for machine teaching; the compressing performance of the learner and the minimisation of the teaching size go in parallel.

Proposition 1. Let f be a complexity function and Φ_ℓ^{f} the learner. There exist two constants $k_1, k_2 \in \mathbb{N}$, such that for any given pair $(w, p) \in \mathsf{f}$-Teaching Book we have that:[4]

$$K(p) \leq \delta(w) + k_1 \textit{ and } K(w) \leq \ell(p) + k_2 \tag{1}$$

Proof. To begin with, we address the first part of the statement 1. Recall that Φ_ℓ^{f} is a one-to-one computable function:

$$\begin{aligned} \Phi_\ell^{\mathsf{f}} \colon \Sigma &\to \Sigma \\ w &\mapsto p. \end{aligned}$$

It is important to state that the domain of this function is the set of sets of examples w included in the f-Teaching Book. In addition, this function is computable since every pair $(w, p) \in \mathsf{f}$-Teaching Book is obtained through the $\mathsf{f}-$bounded *learner* [?].

[4] We use the standard definition of K using a monotone universal machine U [?] (we will drop U when the result is valid for any U), applied to binary strings (where programs and example sets are encoded as explained in the previous section). With K^f we refer to a non-universal version where the descriptional machine is the learner.

Let us recall the invariance theorem, for any two machines (i.e., description modes) U, V, with U universal, there is a constant $k \in \mathbb{N}$ such that for every s we have:

$$K_U(s) \leq K_V(s) + k \tag{2}$$

Interestingly, we can use Φ_ℓ^{f} as a description mode (a machine V), since it is a one-to-one function that goes from binary strings to binary strings. We can then define the Kolmogorov complexity of p, with respect to the description mode Φ_ℓ^{f} (which will be referred from now on simply as Φ):

$$K_\Phi(p) = min\{\delta(w) : \Phi(w) = p\} \tag{3}$$

It follows, from inequality 2 and definition 3, that for every U:

$$K_U(p) \leq K_\Phi(p) + k_1 \tag{4}$$

Since $K_\Phi(p) = \delta(w)$ as (w, p) is a pair of the f-Teaching Book. Then, the inequality 4 can be expressed as:

$$K(p) \leq \delta(w) + k_1$$

Note that U has been dropped as this is valid for any universal machine U.

Now we address the second part of statement 1. We recall that every pair $(p, w) \in \mathsf{f}$-Teaching Book is unique, i.e., there is just one and only one $p \in \Sigma$ such that $\Phi(w) = p$. In other words, the function $\Phi\colon \Sigma \to \Sigma$ is one-to-one. Recall also that this function is defined over the set of examples w of the f-Teaching Book. So that, there is an inverse (partial)[5] function $\Phi^{-1}\colon \Sigma \to \Sigma$.

Clearly, Φ^{-1} is also a computable function just by looking into the book. Now, we are able to consider Φ^{-1} as a description mode (a machine V) and we can obtain the Kolmogorov complexity of w with respect to Φ^{-1} as:

$$K_{\Phi^{-1}}(w) = min\{\ell(p) : \Phi^{-1}(p) = w\} \tag{5}$$

Again, since (p, w) is a pair of the f-teaching book then $K_{\Phi^{-1}}(w) = \ell(p)$. Similarly to the reasoning already used to obtain the first part of statement 1, through inequality 2 and definition 5, there exists a constant k_2, which does not depend on p, such that

$$K(w) \leq \ell(p) + k_2$$

$\square$

Proposition 1 is a key result ensuring that the size difference between programs and witness sets is bounded: a short witness set would not correspond with an arbitrarily complex concept and vice versa. This puts a limit to the surprising empirical observation in [**?**], where the size of the witness sets in bits was usually smaller than the size of the shortest program for that set, i.e., in

[5] A partial function, as the original Φ was.

terms of information it was usually cheaper to teach by example than sending the shortest description for a concept.

There is another close relationship between the Kolmogorov complexity of a concept and its teaching size. First we need to define the complexity of a concept through the *first program* of a concept in language L.

$$p_c^* = \arg\min_p^{\prec} \{\ell(p) : p \in [c]_L\}$$

For every concept $c \in C_L$, we will simply refer to the Kolmogorov complexity of a concept c with respect to the universal language L as $K_L(c) = \ell(p_c^*)$. Now,

Theorem 1. Let L be a universal language, M be a universal machine and k_M be a constant that denotes the length of a program for Φ in M.[6] For any concept $c \in C_L$, there exists a complexity function f, such that $K_L(c) \leq TS_\ell^{\mathsf{f}}(c) + k_M$.

Proof. Theorem 1 [**?**] guarantees the existence of a complexity function f and a witness set, w_c, such that the learner, Φ_ℓ^{f}, outputs p_c, on input w_c, satisfying:

$$p_c \in [c]_L$$

Then, p_c computes the same partial function that is defined by c. In other words, p_c is a description of c procured through the witness set w_c, the learner, Φ_ℓ^{f}, and some *glue* program of size ϵ that executes Φ_ℓ^{f} on input w_c. As a result, $K_L(c)$ cannot be greater than the addition of $\delta(w_c)$, $K_M(\Phi_\ell^{\mathsf{f}})$ and ϵ, i.e.,

$$K_L(c) \leq TS_\ell^{\mathsf{f}}(c) + K_M(\Phi_\ell^{\mathsf{f}}) + \epsilon$$

□

This gives an upper bound (the teaching size) for the Kolmogorov complexity of a concept. On the other hand, this theorem implies that concepts with *high* complexity are *difficult to teach* in this setting. The surprising observation found in [**?**] of some concepts having shorter TS than K has a limit.

4 Conditional teaching size

In this section we introduce the notion of conditional teaching size and the *curriculum teaching problem*. We now assume that the learner can reuse any already learnt concept to *compose* other concepts. The curriculum teaching problem is to determine the optimal *sequential* way of teaching a set of concepts $Q = \{c_1, c_2, \ldots, c_n\}$, in terms of minimum total teaching size. Let $TS(c_i|c_j, c_k \ldots)$ be the conditional teaching size of concept c_i, given the set of concepts $\{c_j, c_k \ldots\}$

[6] For any universal Turing machine M, a finite program can be built coding an interpreter for Φ in M and taking w_c as input. The length of this 'glued' program does not depend on the concept c but on the machine M to glue things together and how many bits of the program instructions are required to code Φ, i.e., $K_M(\Phi)$.

previously distinguished by the learner. The challenge is to minimise $TS(c_1) + TS(c_2|c_1) + TS(c_3|c_1, c_2) + \ldots$.

In this new setting we need a definition of $TS(c_i|c_j)$ that considers that (1) a concept c has infinitely many programs that generate it, so which one the learner has identified may be important, and (2) the learner must have some *memory*, where that program is stored. Interestingly, if we assume that memory is implemented by storing the identified programs in a library, where the learner can only make calls to —but not reuse its parts—, then it is irrelevant which program has been used to capture concept c, since the learner only reuses the functional *behaviour* of the program[7].

4.1 Conditional teaching size and minimal curriculum

We define a library $B = \{p_1, \ldots, p_k\}$, as a set of programs in the universal language used by the learner. Let $|B| = k$ the number of *primitives*. We assume that Υ always includes an instruction @ for making static[8] library calls. We use @i to denote the instruction that calls the primitive that is indexed as i in the library. If $|B| = 1$, then @ needs no index. Accordingly, the length of a call to the library is $\ell(@\mathsf{i}) = \ell(@) + \log_2(|B|) = \log_2(|\Upsilon|) + \log_2(|B|)$ bits.

Let p, p' be programs in the universal language L and B a library. We say that a program p *contains a call to* p' when @i is a substring of p and i is the index of $p' \in B$. L_B denotes a language L that implements static calls to a library B. Even with static calls, the flow of the program may never reach @ for an input. Interestingly, we can avoid this undecidable question when dealing with programs in the teaching book by considering @ as the last instruction regarding lexicographical order.

Lemma 1. Let f be a complexity function and B a library. For any $(w, p) \in$ f-Teaching Book, if p has a call to B then p effectively reaches @ and executes a primitive on at least one input of w.

Proof. If the learner identifies a program p that incorporates @ without executing it for any example in a given witness set w, there must be a previous program, smaller in size, that does not call the library. The only issue of this rationale is that there are some languages that *skip* some instructions, which are never executed. But, in this case, p will not include such instructions, because the learner would have identified a shorter program without them. □

Let us use $\dot{p}$ to denote program @i, where i is the index of p in the library.

7 Note that the learner may use a complexity function f. If that is the case, it can occur that a particular program p_1 identifies c_1 and c_1 is very useful for c_2, but p_1 is too slow to be used in any reasonably efficient program for c_2, so becoming useless incrementally. The computational time of the learner has also been considered in other machine teaching frameworks [**?**,**?**].

8 There is no loss of generality here, since every program that uses dynamic calls can be rewritten only using static calls [**?**].

Lemma 2. Let B be a library. The language L_B satisfies: $\dot{p} \prec p', \ \forall p'$ such that $p' \notin [p]_L \wedge p'$ has a call to p.

Proof. Consider a program p' such that: $p' \notin [p]_L$ and p' *calls* p. If p' has a *call* to p then @i is a substring of p', where i points to the primitive p. Either p' is $\dot{p}$ or is larger; in the latter case it is posterior in the order $\prec$. □

Now, we are able to redefine the learner, Φ_ℓ^{f}, and the time-bounded Kolmogorov complexity for a given library.

Definition 1. Let f be a complexity function, B a library and S an example set. The learner Φ calculates the *first program* for S in language L_B:

$$\Phi_\ell^{\mathsf{f}}(S|B) = \arg \min_{p \in L_B}{}^{\prec} \{\ell(p) : p \vDash_{\mathsf{f}} S\}$$

The f-bounded Kolmogorov complexity of S, denoted by $K^{\mathsf{f}}(S|B)$, is the length of the program returned by the learner: $K^{\mathsf{f}}(S|B) = \ell\left(\Phi_\ell^{\mathsf{f}}(S|B)\right)$. The extension of the teacher, denoted by $\Omega_\ell^{\mathsf{f}}(c|B)$, also selects the shortest witness set that makes the learner distinguish the concept:

$$\Omega_\ell^{\mathsf{f}}(c|B) = \arg \min_{S}{}^{\lessdot} \left\{\delta(S) : \Phi_\ell^{\mathsf{f}}(S|B) \in [c]_{L_B})\right\}$$

And the definition of the K^{f}-teaching size of a concept c is $TS_\ell^{\mathsf{f}}(c|B) = \delta(\Omega_\ell^{\mathsf{f}}(c|B))$.

We can also extend Theorem 1 in [?].

Corollary 1. Let L be a universal language and B a library. For any concept c in C_{L_B}, there is a complexity function f so that the f-Teaching Book will contain some (p_c, w_c) with $p_c \in [c]_{L_B}$ and $TS_\ell^{\mathsf{f}}(c|B) = \delta(w_c)$.

Proof. The result is direct since L_B is a universal language and we can apply Theorem 1 in [?] directly. □

Sometimes we will refer to the original L and sometimes to the augmented L_B depending on whether we see it conditional to B or not.

We are now in position to give a formal definition of the conditional teaching size given a set of concepts.

Definition 2. Let $a \in C_L$, $\{c_i\}_{i=1}^n \subset C_L$ and let $p_i = \Phi(\Omega_\ell^{\mathsf{f}}(c_i))$, for each $i = 1, \ldots, n$. Let $B = \{p_i\}_{i=1}^n$. We define the conditional teaching size of concept a given the concepts $\{c_i\}_{i=1}^n$, denoted by $TS_\ell^{\mathsf{f}}(a|c_1, \ldots, c_n)$, as

$$TS_\ell^{\mathsf{f}}(a|c_1, \ldots, c_n) = TS_\ell^{\mathsf{f}}(a|B)$$

The programs that identify the concepts are in the same f-Teaching Book.

We now give a definition of *curriculum*. Given a set of concepts, a curriculum is a set of disjoint sequences covering all the concepts. Our notion of curriculum is more general than just a simple sequence. If some branches are unrelated, a curriculum should not specify which branch comes first, and are considered independent 'lessons'. We will see how this flexibility is handled by the algorithm that finds the optimal curriculum in section 5. For instance, Fig. 2 shows how a set of concepts $\{a, b, c, d, e, f, g\}$ is partitioned into three branches: $\{a \to b \to c \to d, e \to f, g\}$, where $a \to b$ means that b must come after a in the curriculum. For each *branch*, there is no background knowledge or library at the beginning. The library grows as the teacher-learner protocol progresses in each branch.

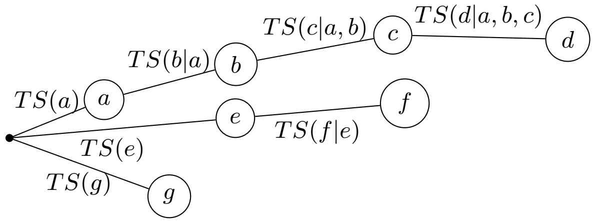

Fig. 2. Curriculum $\{a \to b \to c \to d, e \to f, g\}$ for a set of concepts $\{a, b, c, d, e, f, g\}$.

Definition 3. Let $Q = \{c_i\}_{i=1}^n$ a set of n labelled concepts. A curriculum $\pi = \{\sigma_1, \sigma_2, \cdots, \sigma_m\}$ is a full partition of Q where each of the m subsets $\sigma_j \subset Q$ has a total order, becoming a sequence. Also, we denote $\overline{Q}$ as the set of all the curricula in Q.

The order in which the subsets are chosen does not matter, but the order each subset is traversed does. For example, the curriculum $\pi = \{a \to b \to c \to d, e \to f, g\}$ can have many paths, such as $abcdefg$ or $gabcdef$. But note that π is different from $\pi' = \{b \to a \to c \to d, f \to e, g\}$. For any Q with n concepts, the number of different curricula is

$$|\overline{Q}| = n! \cdot \left(\sum_{k=0}^{n-1} \binom{n-1}{k} \cdot \frac{1}{(k+1)!} \right) \tag{6}$$

Let us explain how to get such number of distinct curricula (Eq. 6), since the rationale is useful to generate them. *For any set $Q = \{c_1, \ldots, c_n\}$ of n concepts, the total number of different curricula is:*

There are $n!$ permutations of n labelled elements. For each permutation, there are $n-1$ possibilities of starting a *branch*. So that, we can choose k positions out of $n-1$. It implies that there will be $k+1$ subsets which can change its order, i.e., $(k+1)!$ different permutations of the subsets express the same case. Therefore, there are $n! \cdot \binom{n-1}{k} \cdot \frac{1}{(k+1)!}$ cases. Since $k \in \{0, 1, \ldots, n-1\}$, the Eq. 6 gives the total number of distinct curricula.

In what follows we will consider that the concepts we work with are all in the original f-Teaching Book, so they can be taught independently. This is not an important constraint, given Theorem 1 in [**?**] and Corollary 1 here. With this we ensure the same f for all of them. Now we can properly define the teaching size of a curriculum:

Definition 4. *Let f be a complexity function and let Q be a set of concepts that appear in the original f-Teaching Book. Let $\pi = \{\sigma_1, \sigma_2, \cdots, \sigma_m\}$ a curriculum in Q. We define the teaching size of each sequence $\sigma = \{c_1, c_2, ..., c_k\}$ as*

$TS^{\mathsf{f}}_{\ell}(\sigma) = TS^{\mathsf{f}}_{\ell}(c_1) + \sum_{j=2}^{k} TS^{\mathsf{f}}_{\ell}(c_j|c_1,\ldots,c_{j-1})$. *The overall teaching size of* π *is just* $TS^{\mathsf{f}}_{\ell}(\pi) = \sum_{i=1}^{m} TS^{\mathsf{f}}_{\ell}(\sigma_i)$.

We say that a curriculum in Q is minimal, denoted by π^*, if no other has less overall teaching size. But *how can we identify minimal curricula?* That is what we analyse next.

4.2 Interposition and non-monotonicity

We now show a teaching phenomenon called *interposition*: new acquired concepts may lead to an increase in teaching size. The phenomenon might not even preserve the relationship established between two concepts, in terms of conditional Kolmogorov complexity, when considering conditional teaching size.

Definition 5. We say that B is an *interposed* library for concept c if $TS(c|B) > TS(c)$; if $B = \{p'\}$ we say that p' is an interposed program for c.

Proposition 2. For any $(w_c, p_c) \in \mathsf{f}$-Teaching Book, such that $@ \prec p_c$, there is an interposed library for concept c.

Proof. Let $S = \{\langle \mathtt{i}, \mathtt{o}\rangle \in w : \delta(w) \leq \delta(w_c) \wedge\ c \vDash w\}$, i.e., the union of all the witness sets of length less or equal than $\delta(w_c)$ that are compatible with concept c.

Let $\langle \mathtt{i}, \mathtt{o}\rangle$ be an input-output pair, which is not compatible with c and $\mathtt{i}$ is not an input in S.

We now define $S' = S \cup \{\langle \mathtt{i}, \mathtt{o}\rangle\}$ as a trie program implementing a lookup table for S'. Let $B = \{p_{\ddot{S'}}\}$ be a library. For every witness set w, such that $\delta(w) \leq TS(c)$, it occurs $\Phi(w|B) = @ \notin [c]_{L_B}$, because $p_{\ddot{S'}}$ covers $\{\langle \mathtt{i}, \mathtt{o}\rangle\}$, but c does not cover it. Therefore, any witness set w', such that $\Phi(w'|B) \in [c]_{L_B}$, satisfies $TS(c) < \delta(w')$. □

The above proposition means that virtually every concept (the only condition is that is represented in the teaching book by a program of more than one instruction) may be interposed by a primitive that makes the witness set lead to another concept. This is an important result, as it is not only the case that some concepts may be useless for the concepts yet to come in the curriculum, but that they may even be harmful. This will have important implications when we look for minimal curricula in the following section.

This contrasts with conditional Kolmogorov complexity, where for every a and b we have that $K(a|b) \leq K(a)$. Given this, we can study the monotonicity between concept complexity and teaching size. Namely, *is there any relationship between* $K(a|b) \leq K(b|a)$ *and* $TS(a|b) \leq TS(b|a)$*?* We now show that, for any universal language, the inequalities aforementioned have, in general, different directions. First, we give the following definition.

Definition 6. Let $c \in C_L$ and let B be a library. We define the Kolmogorov conditional complexity of a concept c given a library B as $K_{L_B}(c) = \ell(p^*_c)$ where p^*_c is calculated using L_B. We use the notation $K(c|B) = K_{L_B}(c)$.

We now extend the conditional Kolmogorov complexity to a set of concepts through programs that identify the concepts given in the same $\mathtt{f}$-Teaching Book and provide the result.

Definition 7. Let $a \in C_L$, the set $\{c_i\}_{i=1}^n \subset C_L$ and $p_i = \Phi(\Omega_\ell^{\mathtt{f}}(c_i))$, for each $i = 1, \ldots, n$. Let $B = \{p_i\}_{i=1}^n$. We define the Kolmogorov complexity of concept a given the concepts $\{c_i\}_{i=1}^n$, denoted by $K(a|c_1, \ldots, c_n)$, as

$$K(a|c_1, \ldots, c_n) = K(a|B)$$

In words, the conditional complexity of a concept given a set of concepts is equal to the conditional complexity of the concept given the canonical programs for those concepts as extracted from the original teaching book.

We now show the non-monotonicity between K and TS:

Theorem 2. There exist two concepts a, $b \in C_L$ and a complexity function, $\mathtt{f}$, such that $K(a|b) < K(b|a)$ and $TS_\ell^{\mathtt{f}}(a|b) > TS_\ell^{\mathtt{f}}(b|a)$.

Proof. In this proof we use a very unusual way of inputs which involve certain fixed 'structure', which we use to define the behaviour of the concepts a and b we want to find. The structure can be used, for instance, to link it with the teaching size of a concept and it also helps to define a partial concept, as we will see below.

Let us consider inputs in the form $\mathtt{i} = \mathtt{xyi}'$, which have three parts:

- $\mathtt{x}$ is a binary representation of a decimal number n using k-digit codification, with $k \geq 2$.
- $\mathtt{y}$ is a binary string that concatenates, k times, the binary string 01.
- $\mathtt{i}'$ is a binary string which employs k-digit codification.

The purpose of such convoluted disposition is to have some redundancy in the coding, which we will exploit for the purposes of this proof. This kind of structure will be used to define very particular concepts that will display the inequalities in the theorem.

For example, $\mathtt{i} = \overline{110011}0101\overline{\overline{1100110011}}$ expresses 10101:

- The substring where there is no repetition of bits is $\mathtt{y} = 0101$, then $k = 2$, because 01 appears twice.
- The first part is $\mathtt{x} = \overline{110011}_{(2}$. Undoing the k-digit codification, $\mathtt{x}$ represents the binary string 101, which gives the decimal number $n = 5_{(10}$.
- Since $n = 5$, we take the last $k \cdot n = 10$ binary digits, $\overline{\overline{1100110011}}$, as $\mathtt{i}' = \overline{\overline{1100110011}}$.
- Now, undoing the k-digit codification, we are *confident* that $\mathtt{i}$ expresses the binary string 10101 as input.

And the other way round, for instance, if we want to express the input 1100 through the structure $\mathtt{xyi}'$, then it is uniquely expressed, in $k = 2$ digit codification, as $\mathtt{i} = \overline{110000}0101\overline{\overline{11110000}}$.

We do not allow inputs starting with zeros. For instance, values of `x` such as 011 or 001 are not valid; we should write 11 and 1, respectively.

Note that we can rewrite every input, which do not start with zero, in the form `xyi'` and vice-versa.

Let b be a concept with very *high* complexity. It also needs every bit of the input, for instance, consider the class of programs that changes ones by zeros and vice-versa; every bit of the input is needed. If the input has another structure, different from `xyi'`, it outputs $\perp$.

Let $p_\perp$ be a program that outputs $\perp$ on every input. Concept b shall satisfy $K(b) > K([p_\perp])$, since the complexity of b must be very *high*.

Theorem 1 in [?], guarantees the existence of a complexity function, f, such that we can teach concept b. Let $(w_b, p_b) \in \mathsf{f}$-Teaching Book, with $TS^{\mathsf{f}}_{\ell}(b) = \delta(w_b)$.

We now define the program p_a as $p_a(\mathtt{xyi'}) = p_b(\mathtt{xyi'})$, when `x` represents a decimal number n, such that $n > \delta(w_b)$. Otherwise, $p_a(\mathtt{xyi'}) = \perp$.

If the complexity of b is high enough, then it is larger $K(b|a)$ than $K(a|b)$, because a is a partial concept of b. So that,

$$K(a|b) < K(b|a) \tag{7}$$

We address now the teaching size of concepts a and b.

Suppose that f does not allow to teach concept a. Then, Theorem 1 in [?] guarantees the existence of another complexity function to teach concept a. We set f as the greatest of both complexity functions.

We now take notice of $TS^{\mathsf{f}}_{\ell}(b|a)$. If the learner receives an input equal or smaller than $\delta(w_b)$, the library $B = \{p_a\}$ does not help to identify concept b. The only rationale against this is that a may help to identify other programs. However, we can always increase the k-digit codification. If k is high enough, then it is more difficult to produce an adequate input for p_a, so that, it will be shorter making p^*_b from scratch, than using the primitive p_a. Therefore,

$$TS^{\mathsf{f}}_{\ell}(b|a) \leq \delta(w_b) = n \tag{8}$$

We now consider $TS^{\mathsf{f}}_{\ell}(a|b)$. We should need example sets larger than n to identify concept a, but we could identify it considering witness sets having only $\perp$ as outputs. But this case is not possible. Since $K(a) > K(b) > K([p_\perp])$, it implies that $[p_\perp]$ precedes a as an output of the learner when it gets, as input, any w with $\delta(w) < n$. Therefore,

$$n < TS^{\mathsf{f}}_{\ell}(a|b) \tag{9}$$

If we consider the inequalities 8 and 9, then we get

$$TS^{\mathsf{f}}_{\ell}(b|a) \leq n < TS^{\mathsf{f}}_{\ell}(a|b) \tag{10}$$

The resolution follows from inequalities 7 and 10. $\square$

When considering conditional teaching size for curriculum learning, we need general conditions to avoid interposition. For instance, an important reduction of program size in language L_B usually minimises the risk of interposition.

Corollary 2. Let $(w_c, p_c) \in \mathsf{f}$-Teaching Book, with $p_c \in [c]_L$. If there exists a library B and a witness set w, verifying the following conditions (1) $\delta(w) < \delta(w_c)$ and (2) the first program $p'_c \in [c]_{L_B}$, using order $\prec$, such that $p'_c \vDash_{\mathsf{f}} w$, precedes any other program p in language L_B, satisfying $p \vDash_{\mathsf{f}} w$, then $TS^{\mathsf{f}}_{\ell}(c|B) < TS^{\mathsf{f}}_{\ell}(c)$.

Proof. If $\Phi(w'|B) \in [c]_{L_B}$, for certain w' such that $w' \lessdot w$, then we have the conclusion. Otherwise, the learner gets w as input; thus, (2) guarantees

$$\Phi(w|B) \in [c]_{L_B}$$

□

These conditions to avoid interposition are quite strong, since we shall elucidate, for instance, whether a program is the shortest one, using a time complexity bound f. The following section takes a different approach that enables curriculum teaching effectively.

5 Minimal curriculum: Interposition range and $\mathbb{I}$-search

One key reason why interposition is hard to avoid is the existence of programs (and concepts) with *parallel behaviour*, i.e., programs with equal inputs-outputs up to large sizes of the inputs, e.g., one implementing the even function, and the other doing the same except for the input 2^{300}. However, in practice, the concepts we use in the break-out for a curriculum do not have this problem. For instance, we can use addition to teach multiplication. They coincide in a few cases, $2+2=4$ and $2\times 2=4$, but they clearly differ in many other short inputs.

Thus, let a, b be distinct concepts such that $\exists (w_a, p_a), (w_b, p_b) \in \mathsf{f}-$Teaching Book, with p_a, p_b in L verifying $w_a \nvDash_{\mathsf{f}} p_b$ and $w_b \nvDash_{\mathsf{f}} p_a$. Assume that we use w_a first and the learner outputs p_a, and adds it to $B = \{p_a\}$. With this increased L_B, if we give w_b to the learner, it does not output p_a since $p_a \nvDash_{\mathsf{f}} w_b$. However, there might still be interposition. For instance, suppose that L has four instructions: x, y, z and t. Let $B = \{\text{xx}\}$ and suppose that $p_b = \text{zytxz}$ is f-compatible with w_b. Suppose that there exists $p = \text{xxytxx}$, expressed as $p = \text{@yt@}$ in L_B, such that $p \vDash_{\mathsf{f}} w_b$. Program p would interpose to p_b. It would be important to know about such programs p, i.e., the ones that precede p_b in L_B and are posterior in L.

5.1 Interposition range: $\mathbb{I}$-sets

Firstly, we define the set of *interposed programs.*

Definition 8. Let w be a witness set and B be a library. Let p be a program in language L_B such that $p \vDash_{\mathsf{f}} w$. We define the $\mathbb{I}$-set of *interposed programs* in language L_B for p and w as $\mathbb{I}^{\mathsf{f}}_{w}(p|B) = \{q \text{ in } L_B : q \vDash_{\mathsf{f}} w \text{ and } q \prec p\}$.

We now show how large the $\mathbb{I}$-sets can be. To do that, we use the size of a program when its library calls are *unfolded*, i.e., given a program p and a library B, we use $\circ(p)$ to denote the program that is equivalent to p (as it worked in L_B) , where each primitive call @ has been replaced by the instructions of the called primitive in B.

Given an $\mathbb{I}$-set, we call *size-range*, denoted as $[i_{min}, i_{max}]$, to the range of $i = \dot{\ell}(\circ(q))$, $\forall q \in \mathbb{I}$-set. The *call-range*, denoted as $[j_{min}, j_{max}]$, is the range of the number of library calls, j, $\forall q \in \mathbb{I}$-set. We call *s/c-ranges* to both ranges; interposition occurs within them. The following theorem gives the *s/c-ranges* explicitly and provides a bound for the cardinality of the $\mathbb{I}$-set.

Theorem 3. Let (w_a, p_a), $(w_b, p_b) \in \mathsf{f}$-Teaching Book, with p_a, p_b in L and $p_a \not\vDash_{\mathsf{f}} w_b$. Consider the library $B = \{p_a\}$. Let p'_b an equivalent program to p_b for L_B. Then, the cardinal of $\mathbb{I}^{\mathsf{f}}_{w_b}(p'_b|B)$ is bounded by $\sum_i \big(\sum_j \binom{i-\dot{\ell}(p_b)\cdot j+j}{j} \cdot (|\Upsilon|-1)^{(i-j\cdot\dot{\ell}(p_b))}\big)$ with i, $j \in \mathbb{N}$ ranging in the intervals: (1) $i_{min} = \dot{\ell}(p_b)$, $i_{max} = 1 + (\dot{\ell}(p'_b) - 1) \cdot \dot{\ell}(p_a)$, $j_{min} = \lceil \frac{i-\dot{\ell}(p'_b)}{\dot{\ell}(p_a)-1} \rceil$ and $j_{max} = \lfloor \frac{i}{\dot{\ell}(p_a)} \rfloor$, when $1 < \dot{\ell}(p_a) < \dot{\ell}(p_b)$; (2) $i_{min} = \dot{\ell}(p_a)+1$ and the rest is as (1), when $\dot{\ell}(p_a) \geq \dot{\ell}(p_b)$.

Proof. **1st case:** We consider that the library does not reduce p_b, i.e., $p'_b = p_b$. In this 1st case, we prove that the cardinal of $\mathbb{I}^{\mathsf{f}}_{w_b}(p_b|B)$ is bounded by

$$(|\Upsilon|-1) + \sum_i \left(\sum_j \binom{i - \dot{\ell}(p_b)\cdot j + j}{j} \cdot (|\Upsilon|-1)^{(i-j\cdot\dot{\ell}(p_b))} \right),$$

where $i \in \left[\dot{\ell}(p_b), (\dot{\ell}(p_b)-1)\cdot\dot{\ell}(p_a)\right]$ and $j \in \left[\left\lceil \frac{i-\dot{\ell}(p_b)}{\dot{\ell}(p_a)-1} \right\rceil, \left\lfloor \frac{i}{\dot{\ell}(p_b)} \right\rfloor\right]$.

We are interested in programs q in L_B, which do not precede p_b in L but they do in L_B, i.e.,

1. $\dot{\ell}(\circ(q)) \geq \dot{\ell}(p_b)$ (otherwise, $q \prec p_b$ in language L and $q \not\vDash_{\mathsf{f}} w_b$).
2. $\dot{\ell}(\circ(q)) \leq (\dot{\ell}(p_b)-1)\cdot\dot{\ell}(p_a) + 1$ (otherwise $p_b \prec q$ in L).

The last condition 2 is equivalent to *fill* all but one instruction of p_b with calls to the library. We do not consider the program @@ $\cdots$ @ of length $\dot{\ell}(p_b)$, since it is posterior or equal to p_b (@ is the last instruction in lexicographical order).

Thus, $\forall q \in \mathbb{I}^{\mathsf{f}}_{w_b}(p_b|B)$ then

$$\dot{\ell}(p_b) \leq \dot{\ell}(\circ(q)) \leq (\dot{\ell}(p_b)-1)\cdot\dot{\ell}(p_a) + 1$$

Now, for each $i = \dot{\ell}(\circ(q))$, we study the number of allowed library calls, j, i.e.:

3. $j \leq \left\lfloor \frac{i}{\dot{\ell}(p_a)} \right\rfloor$. At most, there can be $j = i/\dot{\ell}(p_a)$ library calls, which is the case $i = j \cdot \dot{\ell}(p_a)$

4. $j \geq \left\lceil \frac{i-\dot{\ell}(p_b)}{\dot{\ell}(p_a)-1} \right\rceil$

The last condition 4 occurs because any program in L_B, with higher priority than p_b, satisfies

$$\dot{\ell}(p_b) \geq (i - \dot{\ell}(p_a) \cdot j) + j$$

Since there are j library calls and $i - \dot{\ell}(p_a) \cdot j$ instructions that do use @. In other words, condition 4 guarantees

$$j \cdot (1 - \dot{\ell}(p_a)) \leq \dot{\ell}(p_b) - i,$$

which implies:

$$j \geq \frac{\dot{\ell}(p_b) - i}{1 - \dot{\ell}(p_a)}$$

That said, for each $q \in \mathbb{I}^{\mathsf{f}}_{w_b}(p_b|B)$, with $i = \dot{\ell}(\circ(q))$ and j library calls, we have different distributions of the calls. Thus, for instance, programs like @@xyz, x@y@z or xy@@z, employ the same number of instructions.

Once we fix $i = \dot{\ell}(\circ(q))$, we have to choose j positions out of $(i - j \cdot \dot{\ell}(p_a)) + j$, i.e.,

$$\binom{(i - j \cdot \dot{\ell}(p_a)) + j}{j} \text{ or } \binom{(i - j \cdot \dot{\ell}(p_a)) + j}{i - j \cdot \dot{\ell}(p_a)}$$

Furthermore, for each one of these cases, we have $|\Upsilon| - 1$ possible instructions for each one of the $(i - j \cdot \dot{\ell}(p_a))$ positions.

Therefore, given $i = \dot{\ell}(\circ(q))$, we get for each j that verifies

$$\left\lceil \frac{i - \dot{\ell}(p_b)}{\dot{\ell}(p_a) - 1} \right\rceil \leq j \leq \left\lfloor \frac{i}{\dot{\ell}(p_a)} \right\rfloor$$

a number of interposed programs less than

$$\sum_j \binom{(i - j \cdot \dot{\ell}(p_a)) + j}{j} \cdot (|\Upsilon| - 1)^{(i - j \cdot \dot{\ell}(p_a))}$$

Even, the case $i = (\dot{\ell}(p_b) - 1) \cdot \dot{\ell}(p_a) + 1$ can be isolated, since interposed programs q starting with @ in L_B are lexicographically posterior to p_b in L_B. In such a case, there are $|\Upsilon| - 1$ programs. Finally, we get the conclusion of the 1st case.

2nd case: We prove part (1) of the theorem with p_b' in L_B.

Now, the case $\dot{\ell}(\circ(q)) = (\dot{\ell}(p_b{}') - 1) \cdot \dot{\ell}(p_a) + 1$, can not be isolated. Because $p_b{}'$ might begin with library calls.

The rest is analogous to the 1st case of this proof.

3rd case: We prove part (2) of the theorem with p_b' in L_B.

The proof is also analogous to the 1st case, but the only difference being is $i_{min} = \dot{\ell}(p_a)+1$. This is because a unique call for the library, @, is not compatible with w_b and, with the translation to L_B, any program $q \in \mathbb{I}^{\mathsf{f}}_{w_b}(p_b{}'|B)$ must call the library, so that $\dot{\ell}(\circ(q)) \geq \dot{\ell}(p_a) + 1$. □

Could we identify an empty $\mathbb{I}$-set, based just on the sizes of the programs involved? It happens when the *s/c-ranges* define an *empty region*. In Theorem 3 (1), it occurs whenever $i_{max} < i_{min}$. Namely, we have $\mathbb{I}^{\mathsf{f}}_{w_b}(p_b{}'|B) = \emptyset$, when:

$$\dot{\ell}(p_b) > 1 + (\dot{\ell}(p_b') - 1) \cdot \dot{\ell}(p_a) \tag{11}$$

For instance, if $\dot{\ell}(p_a) = 4$, $\dot{\ell}(p_b) = 8$ and we know that $\dot{\ell}(p_b{}') = 2$, then $i_{min} = 8$ and $i_{max} = 1+(2-1)\cdot 4 = 5$. We see that this becomes more likely as p_b is much greater than p_a and the program for b using B, i.e., $p_b{}'$, is significantly reduced by the use of $B = \{p_a\}$.

Let p' be the first program in $[b]_{L_B}$ such that $p' \vDash_{\mathsf{f}} w_b$. With the conditions of Theorem 3 (1), p' must be equivalent to p_b and operating with Eq. 11 we get $\dot{\ell}(p') < \frac{\dot{\ell}(p_b)-1}{\dot{\ell}(p_a)} + 1$, which means there is no interposition for any program for b by including $B = \{a\}$ and $TS^{\mathsf{f}}_{\ell}(b|a) \le TS^{\mathsf{f}}_{\ell}(b)$. But, since $\ell(p_b') \ge K(b|a)$ we also have that Eq. 11 is impossible when $K(b|a) \ge (\frac{\dot{\ell}(p_b)-1}{\dot{\ell}(p_a)} + 1) \cdot \log_2 |\Upsilon|$.

We now consider a library with more than one primitive. We cannot extend Theorem 3 as a Corollary, since the relationships involved change completely, but we can connect both cases through the *s/c-ranges*.

Theorem 4. Let $\{(w_m, p_m)\}_{m=1}^n$, $(w_c, p_c) \in$ f-Teaching Book, with p_c, p_m in L, $\forall m$. Consider $B = \{p_m\}_{m=1}^n$ with $p_m \nvDash_{\mathsf{f}} w_c$, $\forall m$, and $1 < |B|$. Let $p_c{}'$ be an equivalent program to p_c for L_B. Let D, $r \in \mathbb{N}$ such that $\ell(p_c') = D\cdot\ell(@\mathsf{i})+r$, i.e., they are the *divisor* and the *remainder* of the division $\ell(p_c')/\ell(@\mathsf{i})$. Note that $\ell(@\mathsf{i}) = \log_2|\Upsilon| + \log_2|B|$. Let $p_{max} = \max^{\prec}\{p_m\}_1^n$ and $p_{min} = \min^{\prec}\{p_m\}_1^n$. Then, the cardinal of $\mathbb{I}^{\mathsf{f}}_{w_c}(p_c{}'|B)$ is bounded by $|B| \cdot \sum_{s=2}^{\dot{\ell}(p_c')} \left(\sum_{t=1}^{s}(|\Upsilon|-1)^{s-t} \cdot |B|^{t-1}\right)$ and the *s/c-intervals* are: (1) if $1 < \dot{\ell}(p_{min}) \le \dot{\ell}(p_c)$, then $i_{min} = \dot{\ell}(p_c)$, $i_{max} = D\cdot\dot{\ell}(p_{max})+\lfloor r/\log_2|\Upsilon|\rfloor$, $j_{min} = \lfloor \frac{\dot{\ell}(p_c')-\dot{\ell}(\circ(q))}{\dot{\ell}(@\mathsf{i})-\dot{\ell}(p_{max})} \rfloor$ and $j_{max} = \min\{D, \lfloor \frac{\dot{\ell}(\circ(q))}{\dot{\ell}(p_{min})} \rfloor\}$; (2) if $\dot{\ell}(p_c) < \dot{\ell}(p_{min})$, then $i_{min} = \dot{\ell}(p_{min}) + 1$ and the rest is as in (1).

Proof. Firstly, we want to find an upper bound for $\mathbb{I}^{\mathsf{f}}_{w_c}(p_c{}'|B)$. Programs without library calls are not interposed to $p_c{}'$, otherwise the non-incremental learner outputs a program different from p_c, which is a contradiction; we can only get interposition through library calls.

Since interposed programs must employ equal or less instructions in language L_B than $p_c{}'$, then $2 \le \dot{\ell}(q) \le \dot{\ell}(p_c{}')$. Therefore, $|\mathbb{I}^{\mathsf{f}}_{w_c}(p_c{}'|B)|$ is bounded by

$$\sum_{s=2}^{\dot{\ell}(p_c{}')} (|\Upsilon|-1)+|B|)^s - \sum_{s=2}^{\dot{\ell}(p_c{}')} (|\Upsilon|-1)^s = \sum_{s=2}^{\dot{\ell}(p_c{}')} ((|\Upsilon|-1)+|B|)^s - (|\Upsilon|-1)^s$$

If we apply the binomial theorem and use $|B|$ as common factor, then we get the conclusion

$$|\mathbb{I}^{\mathsf{f}}_{w_c}(p_c{}'|B)| \le |B| \cdot \sum_{s=2}^{\dot{\ell}(p_c{}')} \left(\sum_{t=1}^{s}(|\Upsilon|-1)^{s-t} \cdot |B|^{t-1}\right)$$

1st case: We now obtain the *s/c-ranges* for part (1) of the theorem.

Interposed programs must satisfy $\dot{\ell}(\circ(q)) \geq \dot{\ell}(p_c)$, for each $q \in \mathbb{I}^{\mathsf{f}}_{w_c}(p_b{}'|B)$.

Now, let us show the maximum value of $\dot{\ell}(\circ(q))$. Every q shall use the library and its priority must be higher than $p_c{}'$, i.e., $\ell(q) \leq \ell(p_c{}')$.

The higher the number of library calls is, j, the larger $\dot{\ell}(\circ(q))$ is. We have, at most, $\ell(p_c{}')$ bits, then the maximum number of library calls is D, where

$$\ell(p_c{}') = D \cdot \ell(@\mathsf{i}) + r$$

The remainder r can be used without library calls @, namely $\lfloor r/\log_2|\Upsilon|\rfloor$ instructions. Therefore:

$$\dot{\ell}(\circ(q)) \leq D \cdot \dot{\ell}(p_{max}) + \lfloor r/\log_2|\Upsilon|\rfloor$$

Now, once $\dot{\ell}(\circ(q))$ is fixed, we look for a lower bound of library calls, j. We know that, given $m \in \{1,\ldots,n\}$, then

$$\dot{\ell}(p_c{}') \geq j + (\dot{\ell}(\circ(q)) - \dot{\ell}(p_m)\cdot j) \tag{12}$$

So that, since $\dot{\ell}(p_m) > 1$, we get

$$\left\lfloor \frac{\dot{\ell}(p_c{}') - \dot{\ell}(\circ(q))}{\dot{\ell}(@\mathsf{i}) - \dot{\ell}(p_{max})} \right\rfloor \leq \left\lfloor \frac{\dot{\ell}(p_c{}') - \dot{\ell}(\circ(q))}{1 - \dot{\ell}(p_m)} \right\rfloor \leq j \tag{13}$$

Note that we take the higher denominator for all p_m, through p_{max}, to cover every possibility.

If we want to express inequality 12 in bits, it is

$$\ell(p_c{}') \geq j \cdot \ell(@\mathsf{i}) + (\dot{\ell}(\circ(q)) \cdot \log_2|\Upsilon| - j \cdot \dot{\ell}(p_m) \cdot \log_2|\Upsilon|)$$

So that

$$\ell(p_c{}') - \dot{\ell}(\circ(q)) \cdot \log_2|\Upsilon| \geq j \cdot (\ell(@\mathsf{i}) - \dot{\ell}(p_m) \cdot \log_2|\Upsilon|)$$

Suppose that

$$\log_2|B| < \log_2|\Upsilon| \cdot (\dot{\ell}(p_{max}) - 1) \tag{14}$$

Then $\ell(@\mathsf{i}) < \dot{\ell}(p_{max}) \cdot \log_2|\Upsilon|$, we get another lower bound for library calls

$$\left\lfloor \frac{\ell(p_c{}') - \dot{\ell}(\circ(q)) \cdot \log_2|\Upsilon|}{\ell(@\mathsf{i}) - \dot{\ell}(p_{max}) \cdot \log_2|\Upsilon|} \right\rfloor \leq j \tag{15}$$

We now consider the following expressions 16, 17 and 18:

$$\frac{\ell(\circ(q)) - \ell(p_c')}{\ell(p_{max}) - \ell(@\mathsf{i})} \leq \frac{j \cdot \log_2|B|}{\log_2|B|} = j \tag{16}$$

$$\frac{\ell(p_c{}') - \dot{\ell}(\circ(q)) \cdot \log_2|\Upsilon|}{\ell(@\mathsf{i}) - \dot{\ell}(p_{max}) \cdot \log_2|\Upsilon|} = \frac{\ell(\circ(q)) - \ell(p_c')}{\ell(p_{max}) - \ell(@\mathsf{i})} \tag{17}$$

$$\frac{\ell(\circ(q)) - \ell(p_c') + j \cdot \log_2 |B|}{\ell(p_{max}) - \ell(@\mathsf{i}) + \log_2 |B|} = \frac{\dot{\ell}(p_c{}') - \dot{\ell}(\circ(q))}{\dot{\ell}(@\mathsf{i}) - \dot{\ell}(p_{max})} \tag{18}$$

Also, we consider the following property:

$$\forall \frac{x}{y}, \frac{u}{v} \text{ such that } \frac{x}{y} \leq \frac{u}{v} \text{ then } \frac{x}{y} \leq \frac{x+u}{y+v} \tag{19}$$

If we consider the inequality 16 and property 19, then we can *connect* Eq. 17 and Eq. 18, through the following inequality:

$$\frac{\ell(\circ(q)) - \ell(p_c')}{\ell(p_{max}) - \ell(@\mathsf{i})} \leq \frac{\ell(\circ(q)) - \ell(p_c') + j \cdot \log_2 |B|}{\ell(p_{max}) - \ell(@\mathsf{i}) + \log_2 |B|} \tag{20}$$

Therefore, inequality 13 improves 15 as a lower bound:

$$\left\lfloor \frac{\ell(p_c{}') - \dot{\ell}(\circ(q)) \cdot \log_2 |\Upsilon|}{\ell(@\mathsf{i}) - \dot{\ell}(p_{max}) \cdot \log_2 |\Upsilon|} \right\rfloor \leq \left\lfloor \frac{\dot{\ell}(p_c{}') - \dot{\ell}(\circ(q))}{\dot{\ell}(@\mathsf{i}) - \dot{\ell}(p_{max})} \right\rfloor \leq j$$

If we use inequality 13 as a lower bound, there is no restriction for the number of primitives in the library. The assumptions (14) for the library are just to assure inequality 15.

Now we look for an upper bound of library calls j, given $\dot{\ell}(\circ(q))$. On one side, there is a limit for the number of instructions in L

$$j \leq \left\lfloor \frac{\dot{\ell}(\circ(q))}{\dot{\ell}(p_{min})} \right\rfloor$$

But, the equivalent limit expressed in bits is D, i.e.,

$$j \leq \left\lfloor \frac{\ell(p_c{}')}{\ell(@\mathsf{i}))} \right\rfloor = D$$

Since, every q shall meet both limits, then

$$j \leq min \left\{ D, \left\lfloor \frac{\dot{\ell}(\circ(q))}{\dot{\ell}(p_{min})} \right\rfloor \right\}$$

2nd case: We now obtain the *s/c-ranges* for part (2) of the theorem.

We know that, each $q \in \mathbb{I}^{\mathsf{f}}_{w_c}(p_c{}'|B)$ shall be of size greater than p_c in L. In other words, $\dot{\ell}(\circ(q)) \geq \dot{\ell}(p_c)$, otherwise the learner outputs q on input w_c, which is a contradiction.

The main difference with part (1) is that $\dot{\ell}(p_c{}') \leq \dot{\ell}(p_c)$, so that, any interposed program q in L_B, which does not employ the library, cannot cause interposition. Only programs with instructions @i can be interposed. Since $\dot{\ell}(p_{min}) > \dot{\ell}(p_c)$, then $\circ(@\mathsf{i}) > \circ(p_c)$. So that, just a call to the library would cause interposition, had it not been because $@\mathsf{i} \nvDash_{\mathsf{f}} w_c$. That is why an interposed program needs, at least, two instructions in language L_B. Therefore, $\dot{\ell}(p_{min}) + 1 \leq \dot{\ell}(\circ(q))$.

The rest of the proof is similar to the 1st case. □

We need $D \cdot \dot{\ell}(p_{max}) + \lfloor r/\log_2 |\Upsilon| \rfloor < \dot{\ell}(p_{min}) + 1$, to avoid interposition directly, in the same conditions as in Theorem 4 (1). It entails $\ell(p_c{}') < \ell(@\mathtt{i})$ when $\lfloor r/\log_2 |\Upsilon| \rfloor = 0$ in the extreme case. For Theorem 4 (2), an unfeasible *s-range* implies $D < \frac{\dot{\ell}(p_c) - \lfloor r/\log_2 |\Upsilon| \rfloor}{\dot{\ell}(p_{max})}$, which is restrictive.

5.2 Teaching size upper bounds: $\mathbb{I}$-safe

In practice, we deal with a program p that has the desired behaviour for a given witness set, but there may be interposition. If we know which the interposed programs are, then it is possible to get an upper bound of the teaching size of the concept that defines p, by *deflecting* interposition, refining the witness sets.

We employ $\mathbb{I}$-*safe* witnesses: example sets attached to input/output pairs. For instance, if we want to teach exponentiation, a set of examples might be $\{(3,1) \rightarrow 3, (2,2) \rightarrow 4\}$. This witness set is compatible with exponentiation, but also compatible with multiplication. To avoid multiplication being interposed, we can add another example to distinguish both concepts: $\{(3,1) \rightarrow 3, (2,2) \rightarrow 4, (2,3) \rightarrow 8\}$. We can always replace the original witness set by an $\mathbb{I}$-safe witness set, where, in general, we need to add examples to avoid interposition.

Proposition 3. Let f be a complexity function and (w,p), $\{(w_m,p_m)\}_{m=1}^{n} \in \mathsf{f}$-Teaching Book, with p, p_m in L, $\forall m$. Let $B = \{p_m\}_{m=1}^{n}$ be a library such that $p_m \nvDash_{\mathsf{f}} w$, $\forall m$. Let $c \in C_L$ such that $c \vDash w$. Let $p'_c \in [c]_{L_B}$ be the first program, using order $\prec$, such that $p'_c \vDash_{\mathsf{f}} w$. If $n = |\mathbb{I}^{\mathsf{f}}_{w}(p'_c|B)|$, there exist $\{\langle \mathtt{i}_k, \mathtt{o}_k \rangle\}_{k=1}^{n}$ such that $TS^{\mathsf{f}}_{\ell}(c|B) \leq \delta\big(w \bigcup_{k=1}^{n} \{\langle \mathtt{i}_k, \mathtt{o}_k \rangle\}\big)$.

Proof. Let us enumerate the interposed programs: $\mathbb{I}^{\mathsf{f}}_{w}(p'_c|B) = \{q_k\}_{k=1}^{n}$, such that $q_k \prec q_{k+1}, \forall k$.

For each k, we define what we call an $\mathbb{I}$-safe: the first input-output pair using $\lessdot$, $w_k = \{\langle \mathtt{i}_k, \mathtt{o}_k \rangle\}$, such that

$$p'_c \vDash_{\mathsf{f}} w_k \text{ and } q_j \nvDash_{\mathsf{f}} w_j$$

Note that, we define $\mathtt{o}_k = p'_c(\mathtt{i}_k)$ after generating $\mathtt{i}_k$ through $\lessdot$.

Finally, we aggregate every $\mathbb{I}$-safe pair to w. □

For a library B, if we find an example set w that can be converted into an $\mathbb{I}$-safe witness set $\overline{w} = w \bigcup_{k=1}^{n} \{\langle \mathtt{i}_k, \mathtt{o}_k \rangle\}$ with $\delta(\overline{w}) < TS^{\mathsf{f}}_{\ell}(c)$ using B, then we reduce the teaching size. This is a sufficient and necessary condition to avoid interposition and get $TS^{\mathsf{f}}_{\ell}(c|B) \leq TS^{\mathsf{f}}_{\ell}(c)$.

Finally, given these general bounds: *how can we find minimal curricula?* Let us consider, for example, the set of concepts $Q = \{a, b\}$, where (w_a, p_a) and (w_b, p_b) are in the f-Teaching Book. We also know that their behaviours are not parallel, i.e., $p_a \nvDash_{\mathsf{f}} w_b$ and $p_b \nvDash_{\mathsf{f}} w_a$. There are three different curricula $\{a, b\}$, $\{a \rightarrow b\}$ or $\{b \rightarrow a\}$. There is an $\mathbb{I}$-safe witness set $\overline{w}$, such that $\delta(\overline{w}) \leq TS^{\mathsf{f}}_{\ell}(b|a)$ (or $\delta(\overline{w}) \leq TS^{\mathsf{f}}_{\ell}(a|b)$). Thus, we can choose a curriculum, with less overall teaching size than the non-incremental version.

5.3 Minimal curriculum algorithm: $\mathbb{I}$-search

We now *search* minimal curricula. For example, let $Q = \{c_+, c_\times\}$ be a set of two concepts from Fig. 1, which appear in the non-incremental f-Teaching Book as (w_+, p_+) and $(w_\times, p_\times)$. The set of possible curricula, $\overline{Q}$, is $\pi_0 = \{c_+, c_\times\}$, $\pi_1 = \{c_+ \rightarrow c_\times\}$ and $\pi_2 = \{c_\times \rightarrow c_+\}$.

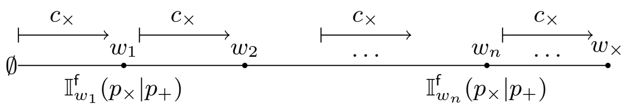

Fig. 3. Non-decreasing sequence of witness sets w_k, through $c_\times$ with $\delta(w_k) \leq \delta(w_\times)$.

The starting point for our algorithm will be π_0, the non-incremental curriculum, and its overall teaching size TS^{f}_ℓ. Then, we generate another curriculum: π_1. We know $TS^{\mathsf{f}}_\ell(c_+) = \delta(w_+)$ and we need to add $TS^{\mathsf{f}}_\ell(c_\times|c_+)$. We compare this total size to the best TS so far. We explore all the curricula in $\overline{Q}$ but, in order to save computational steps, we generate successive witness sets w_k, using order $\lessdot$, such that $c_\times \vDash w_k$ (Fig. 3). For each w_k, we get the first program p_k of $\mathbb{I}^{\mathsf{f}}_{w_k}(p_\times|p_+)$. We then investigate whether $p_k \in [p_\times]_{L_B}$ or not. If p_k acts like $p_\times$ to certain witness size limit, H, then we can identify p_k and $p_\times$. The following algorithm extends this strategy in general:

```
Algorithm: 𝕀-search
Input: $Q = \{a, b, \ldots\}$; f-Teaching Book $(w_a, p_a)$, $(w_b, p_b)$...; Witness size limit $H$
  1. For each distinct pair of concepts $\langle x, y\rangle \in Q \times Q$:
     (a) If $[TS^{\mathsf{f}}_\ell(y|x) \leq TS^{\mathsf{f}}_\ell(y) \wedge TS(x|y)^{\mathsf{f}}_\ell \geq TS^{\mathsf{f}}_\ell(x)]$
         then $\overline{Q} = \overline{Q} \setminus \{\pi : \exists$ a branch starting as $y \rightarrow x\}$
  2. $\pi^* = \{a, b, \ldots\}$, $TS^{\mathsf{f}}_\ell(\pi^*) = \sum_{x \in Q} TS^{\mathsf{f}}_\ell(x)$ and $\overline{Q} = \overline{Q} \setminus \{\pi^*\}$
  3. For each $\pi \in \overline{Q}$:
     (a) $TS^{\mathsf{f}}_\ell(\pi) = 0$
     (b) For each branch $\sigma \in \pi$:
           i. For each concept $x \in \sigma$ (ordered by $\sigma$):
                – $B = \{p_y : (y \in \sigma) \wedge (y$ precedes $x)\}$
                – Let $p'_x$ be the first program equivalent to $p_x$ in $L_B$, using order $\prec$
                – For each $w_k \in \{w \subset X : p'_x \vDash_{\mathsf{f}} w_k\}$, using order $\lessdot$:
                    • If $[TS^{\mathsf{f}}_\ell(\pi^*) \leq TS^{\mathsf{f}}_\ell(\pi) + \delta(w_k)]$ then break to 3
                    • $p = \min^{\prec}\{\mathbb{I}^{\mathsf{f}}_{w_k}(p'_x|B)\}$; use s/c ranges to refine the calculation
                    • If $[p \vDash_{\mathsf{f}} w \longleftrightarrow p_x \vDash_{\mathsf{f}} w, \forall w$ such that $\delta(w) < H]$
                      then [ $TS^{\mathsf{f}}_\ell(\pi) = TS^{\mathsf{f}}_\ell(\pi) + \delta(w_k)$ and break to 3(b)i ]
     (c) $\pi^* = \pi$ and $TS^{\mathsf{f}}_\ell(\pi^*) = TS^{\mathsf{f}}_\ell(\pi)$
Output: $\pi^*$ and $TS^{\mathsf{f}}_\ell(\pi^*)$
```

Note that the *s/c-ranges* reduce, *drastically*, the computational effort of executing the teacher-learner protocol (calculating teaching book and TS). In the previous example, e.g., if there is a w_n such that $TS^{\mathsf{f}}_\ell(c_\times|c_+) = \delta(w_n) < TS^{\mathsf{f}}_\ell(c_\times)$, then we set $\pi^* = \pi_1$ (and $TS^{\mathsf{f}}_\ell(\pi^*) = \delta(w_+) + \delta(w_n)$). Finally, we test π_2 and follow the same steps as with π_1. If, at some stage, there is a witness set w_m such that $TS^{\mathsf{f}}_{\ell}(c_\times) + \delta(w_m) \geq TS^{\mathsf{f}}_{\ell}(\pi^*)$, then π_1 is minimal and we stop.

The algorithm is complete but the search is not *exhaustive*, since we can *discard* curricula that contain a *branch* starting in a way that does not decrease the overall teaching size for sure. For example, if $TS^{\mathsf{f}}_{\ell}(c_\times|c_+) \leq TS^{\mathsf{f}}_{\ell}(c_\times)$ and $TS^{\mathsf{f}}_{\ell}(c_+|c_\times) \geq TS^{\mathsf{f}}_{\ell}(c_+)$, the branch $\sigma = \{c_+ \to c_\times \to c_\wedge\}$ has less or equal overall teaching size than $\sigma' = \{c_\times \to c_+ \to c_\wedge\}$. Consequently, we can remove all branches starting with $c_\times \to c_+$. We can test this for every pair of distinct concepts at the beginning of the branches.

The $\mathbb{I}$-*search algorithm* (5.3) satisfies the following theorem.

Theorem 5. Let H be certain witness size limit, f be a complexity function and Q be a set of concepts registered in the f-Teaching Book. We also assume, for each $c \in Q$, that $c \vDash w \to p_c \vDash_{\mathsf{f}} w$, $\forall w$ verifying $\delta(w) \leq \sum_{x\in Q} TS^{\mathsf{f}}_{\ell}(x)$. Then, the $\mathbb{I}$-search algorithm expressed in algorithm 5.3 returns a minimal curriculum and its overall teaching size.

Proof. Firstly, we want to discard some curricula.

Let x, $y \in Q$ such that $TS^{\mathsf{f}}_{\ell}(y|x) \leq TS^{\mathsf{f}}_{\ell}(y)$ and $TS^{\mathsf{f}}_{\ell}(x|y) \geq TS^{\mathsf{f}}_{\ell}(x)$ then:

$$TS^{\mathsf{f}}_{\ell}(x) + TS^{\mathsf{f}}_{\ell}(y|x) \leq TS^{\mathsf{f}}_{\ell}(x) + TS^{\mathsf{f}}_{\ell}(y) \tag{21}$$

and

$$TS^{\mathsf{f}}_{\ell}(y) + TS^{\mathsf{f}}_{\ell}(x|y) \geq TS^{\mathsf{f}}_{\ell}(x) + TS^{\mathsf{f}}_{\ell}(y) \tag{22}$$

We get $TS^{\mathsf{f}}_{\ell}(x) + TS^{\mathsf{f}}_{\ell}(y|x) \leq TS^{\mathsf{f}}_{\ell}(y) + TS^{\mathsf{f}}_{\ell}(x|y)$, using inequalities 21 and 22. Therefore, a branch starting with $y \to x$ cannot *improve* another branch starting as $x \to y$.

The order $\prec$ guarantees that the first index of the library points to its first program. As a result, the programs that the incremental learner builds, after teaching x and y, does not change whether the teaching order is $x \to y$ or $y \to x$.

We can repeat this procedure for every two concepts in Q and it might reduce the number of candidates to minimal curricula.

Secondly, we take $\pi_0 = \{a, b, \ldots\}$, the non-incremental curriculum, as a reference and we set $\pi^* = \pi_0$ and $TS^{\mathsf{f}}_{\ell}(\pi^*) = \sum_{x\in\pi_0} TS^{\mathsf{f}}_{\ell}(x)$. We now take a different curriculum, $\pi_1 \in \overline{Q}$ and we check whether

$$TS^{\mathsf{f}}_{\ell}(\pi_1) < TS^{\mathsf{f}}_{\ell}(\pi^*) \tag{23}$$

We do it by following the curriculum's branches and adding, successively, the teaching size of its concepts. If, at some stage of the process, inequality 23 is false, then π_1 cannot improve π^* and we take another curriculum π_2. Otherwise, if inequality 23 is true, then we set:

$$\pi^* = \pi_1 \text{ and } TS^{\mathsf{f}}_{\ell}(\pi^*) = TS^{\mathsf{f}}_{\ell}(\pi_1)$$

There are two important issues in this part of the algorithm:

– Let p'_x be the first program, using order $\prec$, which is equivalent to p_x in L, at some stage of the procedure. For each branch, $\sigma \in \pi$, each $x \in \sigma$ and each $w_k \in \{w \subset X : p'_x \vDash_{\mathsf{f}} w_k\}$, we calculate the first program of $\{\mathbb{I}^{\mathsf{f}}_{w_k}(p'_x|B)\}$, using order $\lessdot$.
  - If $\mathbb{I}^{\mathsf{f}}_{w_k}(p'_x|B) = \emptyset$, then we exit the loop we are inside and move to 3(b)i.
  - Otherwise, we get a program p with higher priority than p'_x.
  - If the behaviour of p and p_x is equal until certain witness size limit, H, then we can identify both programs and move to 3(b)i. Otherwise, we get the next witness set.

– We could have problems if there is a witness set w_k, such that $p'_x \nvDash_{\mathsf{f}} w_k$. We should look for another equivalent program, but it might be that f is not sufficient, i.e., we shall find another program f-compatible with w_k. All in all, it would be quite similar to the teacher-learner protocol. That is why, in order to avoid this issue, we assumed that $p_c \vDash_{\mathsf{f}} w$, $\forall w$ with $c \vDash w$. We need this assumption only for witness sets w such that $\delta(w) \leq \sum_{x \in Q} TS^{\mathsf{f}}_{\ell}(x)$, since we cannot get higher overall teaching size.

We proceed succesively with π_2, π_3 and so on. In the end, the algorithm returns a minimal curriculum and its overall teaching size. □

The $\mathbb{I}$-search algorithm shows that: (1) We should create curricula containing concepts that significantly reduce the complexity of another ones. For instance, if concepts $c_\times$ and c_+ (Fig. 1) satisfy $K(c_\times|c_+) < K(c_\times)$, then the chances to minimise the teaching size increase significantly. (2) Given a set of concepts, it may be useful to implement some kind of *isolation* (or even forgetting by separating concepts in different branches[9]). For instance, c_θ might be f-compatible with a considerable number of witness sets w_k and it may cause *interposition* with c_+, $c_\times$ or $c_\wedge$. This is why we should allocate c_θ in a different branch. (3) The branches (or lessons) could simply suggest ways in which we can arrange, *classify* and organise large sets of concepts. The tree-structure for curricula proposed here is a solution for the problem posed in [**?**].

6 Conclusions and future work

The teaching *size* —rather than teaching dimension— opened a new avenue for a more realistic and powerful analysis of machine teaching [**?**], its connections with information theory (both programs and examples can be measured in bits) and a proper handling of concept classes where examples and programs are compositional and possibly universal, such as natural language.

The intuitive concept of how much of the description of a concept is reused for the definition of another dates back to Leibniz's *règle pour passer de pensée en pensée* [**?**], and has been vindicated in cognitive science since Vigotsky's zone of proximal development [**?**,**?**], to more modern accounts of compositionality based on what has been learnt previously [**?**,**?**,**?**].

[9] Forgetting may simply refer to a lesson not using primitives that are considered out of the context of a "lesson".

In mathematical terms, a gradient-based or continuous account of this view of incremental teaching, and the reuse of concepts, is not well accommodated. Incremental teaching is usually characterised as a compositional process, which is a more appropriate view for the acquisition of high-level concepts. The learning counterpart is still very elegantly captured by conditional Kolmogorov complexity, and some incremental learning schemata have followed this inspiration [**?**,**?**,**?**,**?**,**?**]. However, even if the concept of teaching *size* suggests that a mapping was possible, we have had to face a series of phenomena in order to translate some of these intuitions to the machine teaching scenario, and a new setting for curriculum teaching.

The absence of monotonicity because of interposition presents some difficulties for implementing curriculum teaching for compositional languages. Theorems 3 and 4 and its consequences make possible such an implementation: either through sufficient conditions to avoid interposition, by implementing $\mathbb{I}$-safe witness sets or through the $\mathbb{I}$-search.

Given the theoretical bounds and the algorithms for the optimal curricula, we can now start exploring novel algorithms and strategies for curriculum teaching that are suboptimal, but more efficient, such as (1) greedy algorithms introducing the next concept as the one with maximum local TS reduction, (2) approximations based on Vigotsky's zone of proximal development principles [**?**,**?**] where each step is bounded by some teaching length Z, i.e., such that $TS(c_{i+1}|c_1,\ldots,c_i) \leq Z, \forall i$; or (3) variations of the *incremental combinatorial optimal path* algorithm [**?**]. All these new research possibilities in curriculum teaching, and even others, are now wide open to exploration.

Because of the fundamental (re-)connection we have done between K and TS in this paper, another novel possibility for curriculum teaching would be the combination of teaching by examples *and* descriptions of the concepts themselves. This is actually the way humans teach other humans, combining examples and descriptions, but it is nevertheless unprecedented in the application of machine teaching in natural language processing [**?**,**?**]. However, it is beginning to become common with language models, with prompts that combine examples and some indications of the task to perform [**?**,**?**].

Acknowledgements

This work was funded by the EU (FEDER) and Spanish MINECO under RTI2018-094403-B-C32, G. Valenciana under PROMETEO/2019/098 and EU's Horizon 2020 research and innovation programme under grant 952215 (TAILOR).