Some Theorems on Incremental Compression

Arthur Franz(✉)

Odessa, Ukraine
franz@fias.uni-frankfurt.de

Abstract. The ability to induce short descriptions of, i.e. compressing, a wide class of data is essential for any system exhibiting general intelligence. In all generality, it is proven that incremental compression – extracting features of data strings and continuing to compress the residual data variance – leads to a time complexity superior to universal search if the strings are incrementally compressible. It is further shown that such a procedure breaks up the shortest description into a set of pairwise orthogonal features in terms of algorithmic information.

Keywords: Incremental compression · Data compression · Algorithmic complexity · Universal induction · Universal search · Feature extraction

1 Introduction

The ability to induce short descriptions of, i.e. compressing, a wide class of data is essential for any system exhibiting general intelligence. In fact, it is fair to say that the problem of universal induction has been solved in theory [1]. However, the practical progress is impeded by the use of universal search which requires the execution of all lexicographically ordered programs until a solution is found. For the better or worse, Levin Search has the optimal order of computational complexity [2]. Nevertheless, the obvious slowness of this method, hidden in the big "O" notation, seems to be the price for its generality.

In practice, the problem of finding short descriptions is often solved by an incremental approach. For example, in the presently successful deep learning algorithms, each layer in a deep neural network usually detects features of its input x and computes the activation p of neurons in that layer, $p = f'(x)$, as opposed to essentially guessing descriptions in universal search. In the generative mode, typical inputs x can be computed from neural activations: $x = f(p)$. The next layer takes the feature values p and treats them as an input for the next compression step, which can be viewed as incrementally compressing the input since the number of neurons typically decreases at each layer. The hierarchical structure of the human visual cortex also seems to reflect an incremental, layered approach to the representation of real world perceptual data. Finally, the progress of science itself very much resembles incremental compression as

A. Franz—Independent researcher

B. Steunebrink et al. (Eds.): AGI 2016, LNAI 9782, pp. 74–83, 2016.
DOI: 10.1007/978-3-319-41649-6_8

evidenced by the strive for unified theories given a set of previously acquired theories in physics.

On the one hand, there are narrowly intelligent artificial systems and generally intelligent humans both using an efficient, incremental approach to the learning of concise representations of the world. On the other hand, generally intelligent artificial systems exist only on paper [3] and are impeded by the inefficient, non-incremental universal search. The present paper tries to bridge this gap and formulate a general incremental theory of compression. While there has been previous work on incremental search, it is often meant in the sense of reusing previously found solutions to problems (see [4] for a review). The meaning of incremental compression is however different and refers to the decomposition of a single problem into different parts and solving them one by one.

2 Preliminaries

Consider a universal, prefix Turing machine U. Strings are defined on a finite alphabet $\mathcal{A} = \{0, 1\}$ with ϵ denoting the empty string. Logarithms are taken on the basis 2. $\mathcal{A}^*$ denotes the set of finite strings made up of the elements of $\mathcal{A}$. Since there is a one-to-one map $\mathcal{A}^* \leftrightarrow N$ of finite strings on natural numbers, strings and natural numbers are used interchangeably. For example, the length $l(n)$ of an integer n denotes the number of symbols of the string that it corresponds to. The map $\langle \cdot, \cdot \rangle$ denotes a one-to-one map of two strings on natural numbers: $\mathcal{A}^* \times \mathcal{A}^* \leftrightarrow N$. The corresponding map for more than two variables is defined recursively: $\langle x, y, z \rangle \equiv \langle x, \langle y, z \rangle \rangle$. In particular, $\langle z, \epsilon \rangle = z$. Since all Turing machines can be enumerated, the universal machine U operates on a number/string $\langle n, p \rangle$ by executing p on the Turing machine T_n: $U(\langle n, p \rangle) = T_n(p)$. Similarly, a string y is applied to another string x by applying the yth Turing machine: $y(x) \equiv T_y(x) = U(\langle y, x \rangle)$. The prefix complexity $K(x|y)$ of x given y is defined by $K(x|y) \equiv \min\{l(z) : U(\langle z, y \rangle) = x\}$ and $K(x) \equiv K(x|\epsilon)$. The complexity of several variables is defined as $K(x, y) \equiv K(\langle x, y \rangle)$.

3 An Example

Consider the binary string $x = 10110111011110111110\ldots$. First, it can be discovered that the string consists of blocks of 1's. Let f_1 be the number of the Turing machine T_{f_1} in the standard enumeration of Turing machines that takes an integer m, prints m 1's and attaches a 0. f_1 will be called a *feature* of x and the set of parameters $p_1 = m_1 m_2 \ldots = 1, 2, 3, 4, 5, \ldots$ will be called *parameters* of the feature. Hence, the task of compressing x has been reduced to the task of compressing merely p_1 which is shorter than x, while x can be computed by the feature: $f_1(p_1) = x$. The next feature f_2 may represent the Turing machine taking a start value $p_2 = 1$ and increasing 1 each step, such that $f_2(p_2) = p_1$.

Note that universal search would try to find the the whole final description at once by blind search. In contrast to that, incremental compression finds intermediate descriptions (f_1, p_1), (f_2, p_2) and possibly many more layers one by one.

However, those intermediate descriptions are much longer than the final shortest program and will therefore be found much more slowly by universal search. In order to solve this problem, I introduce an inverse map, the so-called *descriptive map* f', that computes the parameters directly: $f'(x) = p$. In the above example, the descriptive map f'_1 may correspond to a Turing machine $T_{f'_1}$ that counts the number of 1's that are separated by 0's and thereby *computes* p_1 instead of trying to guess it as a universal search procedure would. The compression task will then consist of finding pairs (f, f') for each compression level, such that $f(f'(x)) = x$, which will turn out to be much faster than universal search.

4 Definitions

Definition 1 (Features, descriptive maps and parameters). *Let sf and x be finite strings and $D_f(x)$ the set of* ***descriptive maps*** *of x given f:*

$$D_f(x) \equiv \{f' : \ f(f'(x)) = x, \ l(f'(x)) < l(x) - l(f)\} \tag{4.1}$$

If $D_f(x) \neq \varnothing$ then f is called a ***feature*** *of x. The strings $p \equiv f'(x)$ are called* ***parameters*** *of the feature f. f^* is called* ***shortest feature*** *of x if it is one of the strings fulfilling*

$$l\,(f^*) = \min\{l(f) : \ D_f(x) \neq \varnothing\} \tag{4.2}$$

and f'^* is called ***shortest descriptive map*** *of x given f^* if*

$$l\,(f'^*) = \min\{l(g) : \ g \in D_{f^*}(x)\} \tag{4.3}$$

In the definition, any feature is required to do at least some compression, $l(f) + l(p) < l(x)$, since otherwise $f = f' = \mathrm{id}$ would always trivially satisfy the definition for any x. This procedure to search for description and its inverse at the same time has been proposed in [5], called SS'-Search, albeit not in the context of features and incremental compression.

Definition 2 (Incremental compression). *A string x is called* ***incrementally compressible****, if there exist features $f_1, \ldots, f_k$ such that $(f_1 \circ \cdots \circ f_k)(\epsilon) = f_1(f_2(\cdots f_k(\epsilon))) \equiv U(\langle f_1, \ldots, f_k\rangle) = x$.*[1]

5 Properties of a Single Compression Step

The central question for incremental compression is given a finite string x, how to find a pair of a feature f and descriptive map f', such that $f(f'(x)) = x$. In the following the consequences of choosing the shortest f^* and f'^* are explored. All proofs can be found in the appendix.

[1] Note that the $\langle\cdot,\cdot\rangle$-map is defined with $\langle z, \epsilon\rangle \equiv z$, hence $f_k(\epsilon) = U(\langle f_k, \epsilon\rangle) = U(f_k)$, so that f_k acts as a usual string in the universal machine.

Lemma 1. *Let f^* and f'^* be the shortest feature and descriptive map of a finite string x, respectively. Further, let $p \equiv f'^*(x)$. Then*

1. $l(f^*) = K(x|p)$ *and*
2. $l(f'^*) = K(p|x)$.

Theorem 1 (Feature incompressibility). *The shortest feature f^* of a finite string x is incompressible: $K(f^*) = l(f^*) + O(1)$.*

Theorem 2 (Independence of features and parameters). *Let f^* and f'^* be the shortest feature and descriptive map of a finite string x, respectively. Further, let $p \equiv f'^*(x)$. Then,*

1. $K(f^*|p) = K(f^*) + O(1)$,
2. $K(p|f^*) = K(p|f^*, K(f^*)) + O(1) = K(p) + O(1)$ *and*
3. $K(f^*, p) = K(f^*) + K(p) + O(1)$.

Interestingly, from the definition of the shortest feature and descriptive map, it follows that features and parameters do not share information about each other such that the description of the (f^*, p)-pair breaks down into the simpler task of describing f^* and p separately. Since Theorem 1 implies the incompressibility of f^* and $U(\langle f^*, p\rangle) = x$, the task of compressing x is reduced to the mere compression of p. However, f^* and p could store additional, residual information making the compression more difficult: $K(x) < K(f^*, p) + O(1)$. The following theorem shows that this is not the case.

Theorem 3 (Concise information transfer). *Let f^* and f'^* be the shortest feature and descriptive map of a finite string x, respectively. Further, let $p \equiv f'^*(x)$.*

1. *The description of the feature-parameter pair (f^*, p) breaks down into the description of x and a residual part:*

$$K(f^*, p) = K(x) + K(p|x, K(x)) + O(1) \tag{5.1}$$

2. *For a fixed f^*, minimizing the length of the descriptive map f' simultaneously minimizes the residual part:*

$$l(f'^*) \propto K(p|x, K(x)) + O(1)$$

3. *The parameters p do not contain information not present in x and $K(x)$:*

$$K(p|x, K(x)) = O(1) \tag{5.2}$$

4. *The shortest feature f^* does not contain information not present in x and $K(x)$:*

$$K(f^*|x, K(x)) = O(1) \tag{5.3}$$

This theorem guarantees that **all and only** the information in x is transferred to the (f^*, p) pair. Hence, there is no residual information contained in p; the information content in p is a genuine subset of the information in x with the rest being stored in f^*. f^* also does not contain residual information and genuinely represents an incompressible part of x. These conclusions are summarized in the following corollaries.

Corollary 1. *The shortest feature f^* and its parameters p contain no more and no less information than is in x:*

$$K(x) = K(f^*, p) + O(1) \tag{5.4}$$

Corollary 2. *After extracting the incompressible feature f^* all remaining information in x resides in p:*

$$K(x) = l(f^*) + K(p) + O(1) \tag{5.5}$$

This corollary expresses the important result that in order to compress x, it suffices to compress the shorter and simpler string p. Having found the shortest feature and descriptive map we can be certain to be on the right path to the compression of x and not to run into dead-ends.

6 Orthogonal Feature Bases

The following theorems show that compressing the parameters p further leads to an orthogonal feature basis that optimally represents the original string x.

Theorem 4 (Feature bases). *Let x be a string that is incrementally compressed by a sequence of shortest features $f_1^*, f_2^*, \ldots$ and their respective descriptive maps $f'^*_1, f'^*_2, \ldots$ with $p_i \equiv f'^*_i(p_{i-1})$ and $p_0 \equiv x$. Then there will be an integer k after which $p_k = \epsilon$, no further compression is possible and the shortest description of x breaks up into features:*

$$K(x) = \sum_{i=1}^{k} l(f_i^*) + O(1) \tag{6.1}$$

The case $k = 1$ degenerates into the usual, non-incremental compression, in which case the description of x does not break up into features.

Theorem 5 (Orthogonality of features). *Let x be a finite string that is incrementally compressed by a complete sequence of features $f_1^*, \ldots, f_k^*$. Then, the features are **orthogonal** in terms of the algorithmic information: $I(f_i^* : f_j^*) = K(f_j^*)\delta_{ij} + O(1)$, with δ_{ij} being the Kronecker symbol.*

7 Efficiency of Incremental Compression

In order to assess the time complexity of incremental compression we derive an upper bound on $l(f'^*)$.

Theorem 6 (Bound on the length of descriptive map). *Let f^* and f'^* be the shortest feature and descriptive map of a finite string x, respectively. Then the following bound holds on $l(f'^*)$:*

$$l(f'^*) \leq 2\log K(x) + 4\log\log K(x) + O(1) \leq 2\log l(x) + 4\log\log l(x) + O(1)$$

This bound allows to estimate the time complexity for a potential algorithm for incremental compression. If the algorithm uses universal search or similar to find the features and descriptive maps, the time complexity of a single compression step will be proportional to

$$O\left(2^{l(f^*)+l(f'^*)}\right) \leq O\left(l(x)^2\left(\log l(x)\right)^4 2^{l(f^*)}\right) \tag{7.1}$$

At each compression level i, $p_i = f_i'^*(p_{i-1})$ takes the role of x (with $p_0 \equiv x$). But since information is sliced off at each compression level (Corollary 2), we know that $K(p_i) < K(p_{i-1}) < \cdots < K(x) \leq l(x)$ up to a constant. Thus, the bound is valid for each $l(f_i'^*)$ and the time complexity of the whole incremental compression will be proportional to

$$O\left(l(x)^2\left(\log l(x)\right)^4 \sum_{i=1}^{k} 2^{l(f_i^*)}\right) \tag{7.2}$$

In standard universal search the final program $l(p) = Kt(x) \geq K(x)$ is searched for in a non-incremental way, where Kt denotes the resource-bounded Levin complexity. Universal search is therefore proportional to the huge factor $2^{l(p)}$. Since from Theorem 4, we get $2^{l(p)} \geq 2^{K(x)} = c \prod_{i=1}^{k} 2^{l(f_i^*)}$ and $\sum_{i=1}^{k} 2^{l(f_i^*)} \ll \prod_{i=1}^{k} 2^{l(f_i^*)}$ in almost all cases, we observe that incremental compression promises to be much faster than (non-incremental) universal search, if the string is incrementally compressible.[2] Incremental compression is slower only if the search for f'^* is slower than doing universal search from scratch, e.g. when $K(x) \leq 2\log l(x) + 4\log\log l(x)$ which is true only for very simple strings.

Unfortunately, since the Kolmogorov complexity is incomputable, the practical implementation of incremental compression will have to resort to some kind of universal search procedure for the features and descriptive maps which is not guaranteed to find the shortest ones. It remains to be seen whether the present theory can be formulated in terms of Levin complexity $Kt(x)$ instead of the prefix Kolmogorov complexity $K(x)$.

[2] It is not difficult to see that the "$\ll$" sign is justified for all but very few cases. After all, only for very few combinations of a set of fixed sum integers $\sum_i l_i = L$ the sum $\sum_i 2^{l_i}$ is close to 2^L.

8 Discussion

The present approach allows to represent the shortest description of a finite string by a complete set of pairwise orthogonal features in terms of vanishing mutual algorithmic information. The features can be searched for one by one without running into dead-ends, in the sense that for any incomplete set of orthogonal features the remaining ones always exist. At the same time, while the features are the carriers of the information about x, the descriptive maps have been proven to be simple, $l(f'^*) = O(\log K(x))$, allowing for a fast search for them. That makes intuitively sense, since the descriptive maps receive x as an input. It is due to these properties that make the present approach to incremental compression efficient.

The present work is a continuation of my general approach to artificial intelligence [6]. In fact, I have already demonstrated the practical feasibility and efficiency of incremental compression in a general setting. In [7] I have built an algorithm that incrementally finds close to shortest descriptions of all strings computable by 1- and 2-state and 80 % of the strings computable by 3-state Turing machines. Readers interested in a practical implementation of the present approach are referred to that paper.

The example in Sect. 3 demonstrates an actually incrementally compressible string that complies with the Definition 2. This proves that incrementally compressible strings exist. The question arises thus how many compressible strings actually are incrementally compressible. Are there any compressible strings at all that are not incrementally compressible? Another important question is how to find features in the first place. Universal search is still going to be slow, notwithstanding the present considerable improvement. There are ideas to address those questions and present exciting prospects for future research.

Acknowledgements. I would like to express my gratitude to Alexey Potapov and Alexander Priamikov for proof reading and helpful comments.

A Proofs

Proof (Lemma 1).

1. Suppose there is a shorter program g with $l(g) < l(f^*)$, that generates x with the help of p: $U(\langle g, p\rangle) = x$. Then there is also a descriptive map $g' \equiv f'^*$, that computes p from x and $l(g'(x)) = l(f'^*(x)) < l(x) - l(f^*) < l(x) - l(g)$. Therefore, g is a feature of x by definition, which conflicts with f^* already being the shortest feature.
2. Suppose there is a shorter program g' with $l(g') < l(f'^*)$, that generates p with the help of x: $U(\langle g', x\rangle) = g'(x) = p$. Then $g' \in D_{f^*}(x)$ since $f^*(g'(x)) = f^*(p) = x$ and $l(g'(x)) = l(p) < l(x) - l(f^*)$ by construction of f'^*. However, by Eq. (4.3) f'^* is already the shortest program able to do so, contradicting the assumption. □

Proof (Theorem 1). From Lemma 1 we know $l(f^*) = K(x|p)$, with $p = f'^*(x)$. In all generality, for the shortest program q computing x, $l(q) = K(x) = K(q) + O(1)$ holds, since it is incompressible (q would not be the shortest program otherwise). For shortest features, the conditional case is also true: $K(x|p) = K(f^*|p) + O(1)$. After all, if there was a shorter program g, $l(g) < l(f^*)$, that computed f^* with the help of p, it could also go on to compute x from f^* and p, leading to $K(x|p) \leq l(g) + O(1) < l(f^*) + O(1)$, which contradicts $l(f^*) = K(x|p)$.

Further, for any two strings $K(f^*|p) \leq K(f^*)$, since p can only help in compressing f^*. Putting it all together leads to $l(f^*) = K(x|p) = K(f^*|p) + O(1) \leq K(f^*) + O(1)$. On the other hand, since in general $K(f^*) \leq l(f^*) + O(1)$ is also true, the claim $K(f^*) = l(f^*) + O(1)$ follows. □

Proof (Theorem 2).

1. Follows immediately from $K(f^*) = l(f^*) + O(1) = K(x|p) + O(1) = K(f^*|p) + O(1)$.
2. The first equality follows from Theorem 1, since we only need to read off the length of f^* in order to know $K(f^*)$ up to a constant. For the second equality, consider the symmetry of the conditional prefix complexity relation $K(f^*, p) = K(f^*) + K(p|f^*, K(f^*)) + O(1) = K(p) + K(f^*|p, K(p)) + O(1)$ [8, Theorem 3.9.1, p. 247]. If p does not help computing a shorter f^*, then knowing $K(p)$ will not help either. Therefore, from (1), we obtain $K(f^*|p, K(p)) = K(f^*) + O(1)$ and therefore $K(p|f^*, K(f^*)) = K(p) + O(1)$.
3. In general, by [8, Theorem 3.9.1, p. 247] we can expand $K(f^*, p) = K(f^*) + K(p|f^*, K(f^*)) + O(1)$. After inserting (2) the claim follows. □

Proof (Theorem 3).

1. Expand $K(x, p)$ up to an additive constant:

$$K(p) + K(x|p, K(p)) = K(x, p) = K(x) + K(p|x, K(x)) \tag{A.1}$$

   From Lemma 1(1) and Theorem 1 we know $K(f^*) = K(x|p) + O(1)$. Conditioning this on $K(p)$ and using f^*'s independence of p and thereby of $K(p)$ (Theorem 2(1)) we get $K(x|p, K(p)) = K(f^*|K(p)) + O(1) = K(f^*) + O(1)$. Inserting this into Eq. (A.1) and using Theorem 2(3), yields

$$K(f^*, p) = K(p) + K(f^*) = K(x) + K(p|x, K(x)) + O(1) \tag{A.2}$$

2. Fix f^* and let $P_{f^*}(x) \equiv \{f'(x) : f' \in D_{f^*}(x)\}$ be the set of admissible parameters computing x from f^*. From Lemma 1(2), we know that minimizing $l(f')$, with $s = f'(x)$, is equivalent to minimizing $K(s|x)$, i.e. choosing a string $p = f'^*(x) \in P_{f^*}(x)$ such that $K(s|x) \geq K(p|x)$ for all $s \in P_{f^*}(x)$. Conditioning Eq. (A.2) on x leads to:

$$K(p|x) + K(f^*|x) = K(x|x) + K(p|x, K(x), x) = K(p|x, K(x)) \tag{A.3}$$

   up to additive constants. Since f^* and x are fixed, the claim $l(f'^*) = K(p|x) \propto K(p|x, K(x)) + O(1)$ follows.

3. It remains to show that there exists some $p \in P_{f^*}(x)$ such that $K(p|x, K(x)) = O(1)$. After all, if it does exist, it will be identified by minimizing $l(f')$, as implied by (2). Define $q \equiv \text{argmin}_s \{l(s) : U(\langle f^*, U(s)\rangle) = f^*(U(s)) = x\}$ and compute $p \equiv U(q)$. Since $f^*(p) = x$, $p \in P_{f^*}(x)$. Further, there is no shorter program able to compute p, since with p we can compute x given f^* and q is already the shortest one being able to do so, by definition. Therefore, $l(q) = K(p) + O(1)$ and $K(x|f^*) \leq K(p) + O(1)$. Can the complexity $K(x|f^*)$ be strictly smaller than $K(p)$ thereby surpassing the presumably residual part in p? Let p' be such a program: $l(p') = K(x|f^*) < K(p) + O(1)$. By definition of $K(x|f^*)$, $f^*(p') = x$. However, then we can find the shortest program q' that computes p' and we get: $f^*(U(q')) = x$. Since $l(q') \leq l(p') + O(1)$, we get $l(q') < K(p) + O(1) = l(q) + O(1)$. However, this contradicts the fact that q is already the shortest program able to compute $f^*(U(q)) = x$. Therefore,

$$l(q) = K(x|f^*) = K(p) + O(1) \tag{A.4}$$

In order to prove $K(p|x, K(x)) = O(1)$ consider the following general expansion

$$K(p, x|f^*) = K(x|f^*) + K(p|x, K(x), f^*) + O(1) \tag{A.5}$$

Since we can compute p from q and go on to compute x given f^*, $l(q) = K(p, x|f^*) + O(1)$. After all, note that with Theorem 2(2), we have $l(q) = K(p) = K(p|f^*) \leq K(p, x|f^*)$ up to additive constants, but since we can compute $\langle p, x\rangle$ given f^* from q, we know $K(p, x|f^*) \leq l(q) + O(1)$. Both inequalities can only be true if the equality $l(q) = K(p, x|f^*) + O(1)$ holds. At the same time, from Eq. (A.4), $l(q) = K(x|f^*)$ holds. Inserting this into Eq. (A.5) leads to $K(p|x, K(x), f^*) = O(1)$. Taking $K(p) = K(p|f^*) + O(1)$ (Theorem 2(2)), and inserting the conditionals x and $K(x)$ leads to: $K(p|x, K(x)) = K(p|x, K(x), f^*) + O(1) = O(1)$. Since this shows that a $p \in P_{f^*}(x)$ exists with the minimal value $K(p|x, K(x)) = O(1)$, (2) implies that it must be the same or equivalent to the one found by minimizing $l(f')$.
4. Conditioning Eq. (A.3) on $K(x)$ we get $K(p|x, K(x)) + K(f^*|x, K(x)) = K(p|x, K(x)) + O(1)$ from which the claim follows. □

Proof (Corollary 1). Inserting Eq. (5.2) into Eq. (5.1) proves the point. □

Proof (Corollary 2). Inserting Eq. (A.2) into Eq. (5.4) and using the incompressibility of f^* (Theorem 1) proves the point. □

Proof (Theorem 4). According to the definition of a feature, at a compression step the length of the parameters $l(p_i) < l(x) - l(f_i^*)$ and their complexity (Corollary 2) decreases. Since the f_i^* are incompressible themselves (Theorem 1), the parameters store the residual information about x. Therefore, at some point, only the possibility $p_k \equiv f'^*_k(p_{k-1}) = \epsilon$ with $l(f_k^*) = K(p_{k-1})$ remains and the compression has to stop. Expanding Corollary 2 proves the result: $K(x) = l(f_1^*) + K(p_1) + O(1) = l(f_1^*) + l(f_2^*) + K(p_2) + O(1) = \sum_{i=1}^{k} l(f_i^*) + O(1)$. □

Proof (Theorem 5). Algorithmic information is defined as $I(f_i^* : f_j^*) \equiv K(f_j^*) - K(f_j^*|f_i^*)$. The case $i = j$ is trivial, since $K(f_i^*|f_i^*) = 0$. If $i > j$, then $p_j = \left(f_{j+1}^* \circ \cdots \circ f_i^*\right)(p_i)$, which implies that all information about f_i is in p_j. But since according to Theorem 2(1), $K(f_j^*|p_j) = K(f_j^*) + O(1)$ we conclude that $K(f_j^*|f_i^*) = K(f_j^*) + O(1)$. If $i < j$, then we know that f_j^* in no way contributed to the construction of p_i further in the compression process. Hence $K(f_j^*|f_i^*) = K(f_j^*)$. □

Proof (Theorem 6). Let $p \equiv f'^*(x)$. Further, from Lemma 1 we know that $K(x|p) = l(f^*)$ and $K(p|x) = l(f'^*)$. Using Corollary 2, the difference in algorithmic information is $I(p : x) - I(x : p) = K(x) - K(x|p) - K(p) + K(p|x) = l(f'^*) + O(1)$. By [8, Lemma 3.9.2, p. 250], algorithmic information is symmetric up to logarithmic terms: $|I(x : p) - I(p : x)| \leq \log K(x) + 2 \log\log K(x) + \log K(p) + 2 \log\log K(p) + O(1)$. Since x is computed from f^* and p, we have $K(p) \leq K(x)$. Putting everything together leads to $l(f'^*) \leq 2 \log K(x) + 4 \log\log K(x) + O(1)$. The second inequality follows from $K(x) \leq l(x) + O(1)$. □

References

1. Hutter, M.: On universal prediction and Bayesian confirmation. Theor. Comput. Sci. **384**(1), 33–48 (2007)
2. Levin, L.A.: Universal sequential search problems. Problemy Peredachi Informatsii **9**(3), 115–116 (1973)
3. Hutter, M.: Universal Artificial Intelligence: Sequential Decisions based on Algorithmic Probability, 300p. Springer, Heidelberg (2005). http://www.hutter1.net/ai/uaibook.htm
4. Schmidhuber, J.: Optimal ordered problem solver. Mach. Learn. **54**(3), 211–254 (2004)
5. Potapov, A., Rodionov, S.: Making universal induction efficient by specialization. In: Goertzel, B., Orseau, L., Snaider, J. (eds.) AGI 2014. LNCS, vol. 8598, pp. 133–142. Springer, Heidelberg (2014)
6. Franz, A.: Artificial general intelligence through recursive data compression and grounded reasoning: a position paper. CoRR, abs/1506.04366 (2015). http://arXiv.org/abs/1506.04366
7. Franz, A.: Toward tractable universal induction through recursive program learning. In: Bieger, J., Goertzel, B., Potapov, A. (eds.) AGI 2015. LNCS, vol. 9205, pp. 251–260. Springer, Heidelberg (2015)
8. Li, M., Vitányi, P.M.: An Introduction to Kolmogorov Complexity and Its Applications. Texts in Computer Science. Springer, New York (2009)