# A Computational Theory of Meaning

Pieter Adriaans

SNE group, IvI
University of Amsterdam,
Science Park 107
1098 XG Amsterdam,
The Netherlands.

**Abstract.** In this paper we develop a possible world semantics for Kolmogorov complexity based on frames. We define a notion of computational meaning for strings: the facticity $\varphi(x)$ of a string is the amount of meaningful information it contains in a set of possible worlds. We prove that this notion is asymptotically invariant over frames, i.e. it defines a measure. We discuss various applications of the theory.

keywords: Meaningful information, Kolmogorov complexity, Facticity, Sophistication, agent theory, modal logic, semantics, invariance

## 1 Summary

We present a computational theory of meaning that draws together elements of Kolmogorov complexity, agent theory, modal logic and semantics. The computational theory of meaning developed in this paper can be seen as a generalization of the invariance framework that is the basis for Kolmogorov complexity. The main contributions of this paper are:

- *Meanings are inherenty complex.* Replacing nomenclature in earlier publications[1] we distinguish various flavours of complexity and meaning for a string:
  - *Kolmogorov complexity*: the length of the shortest program that generates a string running on an empty input string.
  - *Sophistication*: the length of the shortest program that optimally compresses a string on free input.

[1] This paper is the culmination of a ten year long quest for a sound, workable, definition of meaningful information. In [16] facticity was tentatvily defined as the product of the randomness deficiency and the Kolmogovorov complexity of a string. In [19] we introduced a concept of facticity that we now prefer to call *sophistication with self-delimiting indexes defined on partial functions*, which is the version of sophisticationthat is analyzed in this paper. Most authors define sophistication on total functions [23], but that does not have any fundamental consequences for the results in the paper. The notion of Facticity we introduce here is new and based on the complexity of sets of machines that compress string.

  - *Facticity*: the complexity of the set of programs that compress a string on free input.

  All these measures are in fact landscapes of possible definitions. For Kolmogorov complexity there is an invariance proof due to Solomonoff. We prove that sophistication is, what we call, *super instable* and there is no hope for an invariance proof. The facticity measure is more resilient and a proof of a weaker notion of invariance with reference to a finite set of universal machines is given below. Although the concept of sophistication is useful in certain contexts, we believe that facticity is closer to our intuitions of what meaning should be. In most cases facticity and sophistication measure roughly the same amount of meaning in a string, except for the cases in which sophistication shows unwanted erratic behaviour.
- *A computational meaning for a string is only defined in the context of a finite set of relevant universal Turing machines that must be specified a priori.* The current practice, motivated by Solomonoff's invariance proof, of defining Komogorov complexity loosely on the basis of some vague notion of small or 'natural' machines is misleading and insufficient for the development of a robust theory of computational meaning.
- We develop a theory that interprets *universal Turing machines as possible worlds* in the context of a *frame* with an *accessability relation.* We introduce the notion of the *variance of a frame.* This allows us to deal variations in Kolmogorov complexity assigned to a string in sets of universal machines.
- *We introduce a special class of Turing machines with naming functions.* A 'name' for a binary object is just an index by which it can be called. A naming function is always finite and it is part of the definition of the machine. By specifiying names for binary objects we can influence their Kolmogorov complexity and thus their probability. Turing machines with naming functions allow us to specify a very general class of agent that can 'learn' from experience.

## 2 Introduction

The discovery of the theory of algorithmic- or Kolmogorov complexity was partly motivated by Carnap's ambition [2] to assign a proper a priori probability to an individual statement given an infinite logical description of the world [4; 8]. Solomonoff formulated the idea that the set of prefix-free input strings for a universal Turing machine would provide such a probability distribution [4; 8; 15]. This concept of Carnap was further developed by Kripke [3] into what we now know as possibile world semantics. Solomonoff's proposals, although hugely influential in computer science, never had a big impact on philosophical research. This paper bridges part of the gap between modal logic and complexity theory. It describes a computational theory of meaning in which universal Turing machines are interpreted as descriptions of possible worlds and their associated universal distributions. This approach clarifies some of the inherent difficulties in the interpretation of Kolmogorov complexity, specifically the issue of the selection of a universal reference machine. We defend the view that the class of admissible reference machines should always be specified a priori since it plays an important role in the boundary estimates of model complexities.

## 3 Computational Meaning

The goal of this research is to develop a general and feasible theory that allows us to analyze the complexity of such diverse phenomena as the World Wide Web, Biodiversity, problem solving and artistic creation. The idea that an intension of an object could be a computation was originally formulated by Frege [1]. The expressions "1 + 4" and "2 + 3" have the same extension (Bedeutung) "5", but a different intension (Sinn). There are contexts in which such a distinction is necessary. Consider the sentence "John knows that $\log_2 2^2 = 2$". Clearly the fact that $\log_2 2^2$ represents a specific computation is relevant here. The sentence "John knows that $2 = 2$" seems to have a different meaning.

One mathematical object can have a variety of different meanings. The ambition to measure the *amount* of useful or meaningful information in a string may look natural from the perspective of complexity theory, but the underlying assumption seems to be that a meaning of an object should be a monolithic entity, which is clearly at variance with the philosophical notion of meaning. Take Frege's famous example of the unique descriptions of the planet Venus as 'The morning star' and 'The evening star'. There seem to be at least three dimensions in play:

1. The object: The planet Venus itself.
2. The name 'Venus' that refers to this object.
3. Unique descriptions of this object like 'The morning star'.

Intuitively the planet Venus is a more meaningful object then the average peble on the beach. One way to measure the notion of meaningfulness would be to analyze the complexity of the set of all possible unique decriptions of Venus, but in principle this set is infinite (e.g. consider the unique description "The star that was visible in London but not noticed by John on Friday the 13th at 8 o'clock."). In this paper we control this richness by taking only in consideration the finite set of unique descriptions that can be derived from the data set itself (i.e. the amount of self-descriptive information in the data set):

**Definition 1.** *The meaning of a digital object is the set of all unique descriptions that are informative in the sense that they compress the original object. The facticity of such an object is the complexity of the set of informative unique descriptions.*

Intuitively a bitmap of Picasso's Guernica contains more meaningful information than a bitmap of the same size of Malevich' Black square [16]. We can even approximate these difference by measuring the information in data sets gathered from these systems in terms of their Kolmogorov complexity. A different intuition tells us that a bitmap created by flipping a coin contains with high probability no or very little meaningful information. Here entropy based measures do not help us. They measure maximal information in random data sets. The situation becomes more clear when we realize that a random data set, although incompressible, has a very simple model: unbiased coin flipping. Data sets with low entropy and data sets with high entropy contain little meaningful information because they have small models.

### 3.1 One-part code compression

In Kolmogorov complexity the amount of information in an object is measured as the length of the smallest program that produces this object on a universal Turing machine. This is one-part code compression. The foundation of the theory is the invariance theorem that was first formulated by Solomonoff [4]: the shortest code for a certain routine in a universal programming language $A$ will only be a constant longer than the optimal code for the same routine in any other universal programming language $B$, since we can always construct a compiler of constant length for $A$ in $B$. Once we select an arbitrary universal programming language (i.e. a universal Turing machine) as our reference tool we enter a world of invariant measurement of the amount of information in individual data sets.

The three main weak points of this approach are 1) the fact that the measure is uncomputable and can only be approximated and 2) the initial choice of a universal Turing machine brings a subjective element in to the theory and 3) the theory is asymptotic and does not work for small data sets. Recent results show that the first point is actually not that harmful: the probability that one chooses a bad model decays exponentially with the lack of quality of that model [21]. The other two points are less well understood. Below we argue that they are related to a fourth issue: the instability of model selection.

### 3.2 Two-part code compression

In the past decennia there have been a number of efforts to define the amount of meaningful or useful information in a string in terms of two-part code compression. Here the optimal description of the data set is not a single program, but an index of a specific Turing machine (the meaningful/useful structural part) and an input string for this machine (the ad-hoc part) [5; 6; 10; 11; 13; 14] (See [19] for an overview). So far these attempts have not been very successful. The main reason is that the selection of models under this definition is not robust. Slight changes in either the data set or the reference Turing machine can generate uncontrollable changes in the complexity of the selected models [20; 21].

So far these observations have drawn little attention, but in fact they are quite harmful for the application of algorithmic complexity in practice, since they imply that the structural information that we can extract from a data set is fundamentally unstable. It seems impossible to formulate an invariance theorem for meaningful information along the lines of Solomonoff's original proof (cite 1964). We can specify the amount of information in an individual data set, but we are unable to say what the meaning of the set is: i.e. what type of generalizable knowledge the data set contains. This is counterintuitive. Learning is nothing but the extraction of general knowledge from individual data sets. A theory that does not explain what we can learn has little to substantiate the claim to be a "universal solution to the problem of induction" [8; 18].

### 3.3 Selecting the right universal Turing machine

The instability of models is directly related to the issue of selecting a reference Turing machine. There seems to be no consensus between the advocates of algorithmic complexity theory about the right choice. There are roughly two schools:

- Poor machine: choose a small universal Turing machine. If the machine is small it is also general and universal, since there is no room to encode any bias in to the machine. Moreover a restriction to small machines gives small overhead when emulating one machine on the other so the version of Kolmogorov complexity you get gives a measurement with a smaller asymptotic margin. Hütter explicitly defends the choice of 'natural' small machines [12; 18], but also [15] seem to suggest the use of small models.
- Rich machine: choose a big machine that explicitly reflects what you already know about the world. For Solomonoff, the inventor of algorithmic complexity, the choice of a universal Turing machine is the choice for a universal prior. He defends an evolutionary approach to learning in which an agent constantly adapts the prior to what he already has discovered. The selection of your reference Turing machine uniquely characterizes your a priori information [8].

It is our view that there is no 'right' choice here. For rigid mathematical proofs the poor machine approach is often best. For practical applications on finite data sets the rich model strategy often gets much better results, since a poor machine would have to re-invent the wheel any time it compresses a data set. This leads to the conclusion that when using Kolmogorov complexity the class of admissible universal models should be explicitly formulated and motivated.

## 4 Turing machines as possible worlds

In this paper universal Turing machines and their associated universal distributions are interpreted as (descriptions of) possible worlds. The following definitions set up an appropriate framework.

### 4.1 Notation

- The set $\{0,1\}^*$ contains all finite binary strings. $\mathbb{N}$ denotes the natural numbers and we identify $\mathbb{N}$ and $\{0,1\}^*$ according to the correspondence

$$(0,\varepsilon),(1,0),(2,1),(3,00),(4,01),\ldots$$

Here $\varepsilon$ denotes the *empty word.* This equivalence allows us to apply number theoretical predicates like '... is prime' to strings and complexity theoretic predicates like '... is compressible' to numbers. Any data set is a number any number is a data set.
- The *length* $l(x)$ of $x$ is the number of bits in the binary string $x$.
- Calligraphic capitals indicate sets: $\mathcal{T}$ is the set of Turing machines, $\mathcal{U}$ is the set of universal Turing machines, $\mathcal{M}$ is a meaning, a set of codes for Turing machines, $\mathcal{W}$ is a set of accessible universal machines, $\mathcal{P}$ is the set of informative predicates, $\mathcal{O}$ is the set of optimally informative predicates.
- Lowercase letters like $x$, $y$ and $z$ will be used as variable names indicating strings, $a$, $b$, $i$, $j$ are indexes of variables, sets, machines etc. Other lowercase letters will be used in denote functions $f$, $p$, $fp$, $v$, $s$, or when no confusion is possible as names for propositions $p$, $q$, $r$.
- Uppercase is reserved for special objects: $T$ is a Turing machine, $U$ is a universal Turing machine, $P$ is a logical predicate.
- Computational Meta-predicates are predicates that might be true of an object, but are uncomputable, such as $Random(x)$, $\delta(x)$, the randomness deficiency of $x$ and $K(x)$, its Kolmogorov complexity.

## 4.2 Turing machines

We will follow the standard reference for Turing machines [9], with additional new notation necessary to distinguish various ways of referring to machines.

- If $x$ is a string than $\overline{x}^U$ is the self delimiting code for this string in the format of the universal Turing machine $U$. What type of self delimiting code is used is not relevant here, apart from the fact that the length of $\overline{x}^U$ is limited for practical purposes by $n + 2\log n + 1$, where $n = l(x)$. When no confusion is possible we will write $\overline{x}$ for $\overline{x}^U$
- $T_i$ is an *abstract mathematical entity,* i.e. the Turing machine with index $i$.
- $\overline{T_i}^U$ is a *description* of $T_i$ for $U$, i.e. a prefix-free program that emulates $T_i$ on a universal Turing machine $U$. $\overline{T_i}^U$ can be processed directly on $U$.
- $\widehat{T_i}^U$ is a *name* for the Turing machine $T_i$ accessible from the universal Turing machine $U$, i.e. a prefix-free code that is associated with the program $\overline{T_i}^U$ for the universal machine $U$. $\widehat{T_i}^U$ can only be processed on $U$ if it has access to a naming function.
- A naming function $n$ : *names* $\rightarrow$ *descriptions*, for a universal Turing machine $U^n$ assigns a description to a name: $n(\widehat{T_i}^U) = \overline{T_i}^U$. Naming functions are part of the definition of the universal machine and thus always finite.
- $U_j^n$ is the universal Turing machine with index $j$ and naming function $n$. Turing functions with a naming function are a proper subset of the class of universal Turing machines. At the start of the computation the prefix-free code on the tape will be checked against the naming function. If it is defined, the computation will run on the associated description, if not, the prefix-free code will be interpreted as a proper program.

- $U_j^{[T_i,T_j,...T_k]}$ is the universal Turing machine with index $j$ and a specification of the naming function as an explicit list of Turing machines $[T_i, T_j, ...T_k]$. The sequence is meaningful: If $U_j^{[T_i,T_j,...T_k]}$ then $\widehat{T_i}^U < \widehat{T_j}^U < ... < \widehat{T_k}^U$ according to the association between strings and numbers defined above.
- $T(y) = x$ indicates an accepting computation: the Turing machine $T$ produces output $x$ given input $y$. The same holds for $U(y) = x$. The computation $T(x)$ does not necessarily have to end in an accepting state, it may be undecidable.
- $U(\widehat{T}^U y) = U(\overline{T}^U y) = z$ denotes the universal Turing machine $U$ emulating a Turing machine $T$ computing $z$ on input $y$. In $U(\widehat{T}^U y)$ the machine is called by its name. In $U(\overline{T}^U y)$ it is called by its description. We'll write $U(\widehat{T}y) = U(\overline{T}y) = T(y) = z$, leaving the superscript $U$ out if no confusion is possible.
- Universal Turing machines can emulate each other, i.e. the emulation relation is symmetrical:

  $$U_i(\overline{U_j}^{U_i}\overline{T}^{U_j}y) = U_j(\overline{U_i}^{U_j}\overline{T}^{U_i}y) = T(y) = z$$

  The sub- and superscripts are important in this notation. If $U_j$ is a Java interpreter then $\overline{T}^{U_j}$ is Java code for $T$, if $U_i$ is a Prolog interpreter then $\overline{T}^{U_i}$ is Prolog code for $T$. The computation $U_j(\overline{U_i}^{U_j}\overline{T}^{U_j}y)$ erroneously tries to run Java code on a Prolog interpreter. Emulations can be stacked indefinitly:

  $$U_1(\overline{U_2}^{U_1}\overline{U_3}^{U_2}...\overline{U_n}^{U_{n-1}}\overline{T}^{U_n}y) = T(y) = z$$

  Here $U_i$ reads the self-delimiting code for $U_{i+1}^{U_i}$ and starts to emulate it.

The exact details of the implementation of a naming function are unimportant but one could think of a finite *naming table* stored in the form of tuples $\langle name, description\rangle$ on a separate tape. One description may have various names. Note that 1) naming functions for universal machines cannot be reflexive or symmetrical, since that would lead to infinite regresses in the naming function: a name $a$ for a description $b$ containing a naming function $f$ associating a name $a$ to a description $b$ containing a naming function $f$ etc. 2) once the naming function is defined names can in principle be used by any other program on $U$ during its execution. This will, in the context of Kolmogorov complexity, happen automatically when needed, because it searches over all possible programs. As soon as a short name for a binary object is defined it will have consequences for the Kolmogorov complexity of a number of binary objects. The down side is that the naming function increases the complexity of our universal machine. Defining naming functions is finding a balance between computational power and descriptive complexity.

### 4.3 Turing frames

Universal Turing machines can emulate all other Turing machines, but it is not clear that they always have access to the code of other machines. We define an abstract accessibility relation that regulates this availability. This relation can be reflexive and symmetric and thus is more abstract than the naming function described above.

- A Turing frame $\mathcal{F} = \langle \mathcal{U}_i, R \rangle$ is a tuple where $\mathcal{U}_i$ is a set of universal Turing machines, and $R : \mathcal{U}_i \times \mathcal{U}_i$ an accessibility relation. The restrictions on $R$ are called the frame conditions.
- $R : \mathcal{U} \times \mathcal{U}$ is an accessibility relation between universal Turing machines. Some possible constraints on the accessibility relation are:
  - reflexive iff $R(U_w, U_w)$, for every $U_w$ in $\mathcal{U}$
  - symmetric iff $R(U_w, U_u)$ implies $R(U_u, U_w)$, for all $U_w$ and $U_u$ in $\mathcal{U}$
  - transitive iff $R(U_w, U_u)$and $R(U_u, U_q)$ together imply $R(U_w, U_q)$, for all $U_w$, $U_u$, $U_q$ in $\mathcal{U}$.
  - serial iff, for each $U_w$ in $\mathcal{U}$ there is some $U_u$ in $\mathcal{U}$ such that $R(U_w, U_u)$.
  - Euclidean iff, for every $U_u$, $U_t$, and $U_w$, $R(U_w, U_u)$ and $R(U_w, U_t)$ implies $R(U_u, U_t)$ (note that it also implies: $R(U_t, U_u)$.

  This definition allows us to formulate relations between universal machines in terms of various systems of modal logic [7]. If there are no constraints on $R$, each machine just has its own database with a haphazard collection of accessible systems. The associated modal logic is $K$. If each machine has access to a database with the same set of machines, then $R$ is an equivalence relation. The associated modal system is $S5$.
- A Transparent- or S5 Turing frame is associated with S5 Modal logic, i.e. the accessibility relation is reflexive, symmetric and transitive.
- $\mathcal{W}_{U,\mathcal{F}} = \{U_i | R(U, U_i)\}$ is the set of universal machines accessible from $U$ according to $R$ specified in $\mathcal{F}$.
- A model is a 3-tuple $M = \langle \mathcal{U}_i, R, v \rangle$.
- Here $v$ is a valuation function. We recursively define the truth of a formula relative to a set of universal Turing machines $\mathcal{U}_i$ and an accessibility relation $R$:
  - if $v(U_w, p)$ then $U_w \models p$
  - $U_w \models \neg p$ if and only if $U_w \not\models p$
  - $U_w \models (p \wedge q)$ if and only if $U_w \models p$ and $U_w \models q$
  - $U_w \models \Box p$ if and only if for every element $U_u$ of $\mathcal{U}$, if $R(U_w, U_u)$ then $U_u \models p$
  - $U_w \models \Diamond p$ if and only if for some element $U_u$ of $\mathcal{U}$, it holds that $R(U_w, U_u)$ and $U_u \models p$
  - $\models p$ if and only if $U \models p$, where $U$ is the reference universal machine.

### 4.4 Computational Predicates associated with Turing machines

Computational predicates allow us to reason about true statements associated with computations:

- $P(x)$ is a proposition that can be true or false, it denotes the predicate $P$ applied to variable $x$: $x$ is $P$.

– Any Turing machine $T_i$ has an associated predicate $P_i$ with the same index with the following semantics:

$$\forall(x \in \{0,1\}^*)\forall(U_y \in \mathcal{U})(v(U_y, P_i(x)) \Leftrightarrow \exists(k)T_i(k) = x)$$

Here $v$ is the valuation function, $P_i(x)$ is a proposition.

– The *extension* of $P_i$ is $\{x|\exists(k)T_i(k) = x\}$, where $k$ is an index of $x$ in this set.

In the following we will refer to Turing machines as *Turing predicates* if we are predominantly interested in the qualities of an object they compute. This construction allows us to formulate true propositions that are associated with computations, e.g. if $T_i$ is a machine that generates Fibonacci numbers, we can say that $P_i(x)$ is true i.e. $x$ is a Fibonacci number, if there is some index $k$ such that $T_i(k) = x$.

Note that 1) in general the extension of such a predicate $P_j$ is uncomputable, 2) this undecidability by itself has no consequences for proof theoretical results of the modal systems as long as we assume that the valuation function $v$ is defined (as is the case in the theory of Kolmogorov complexity introduced below) 3) this definition generalizes over the set of all possible universal Turing machines and 4) the restriction to one place predicates operating on strings does not reduce the expressiveness. Any $n$-place predicate can be implemented via a Turing machine that separates its input-string into $n$ parts via whatever technique we can think of (self-delimiting code, enumeration of sets, etc). Since we only discuss mathematical statements that are true in all worlds the valuation function $v$ will be the same for all models. We can abstract from individual models $M = \langle \mathcal{U}_i, R, v\rangle$ and concentrate on the associated Turing frame $\mathcal{F} = \langle \mathcal{U}_i, R\rangle$.

## 5 Kolmogorov complexity

We follow the standard reference for Kolmogorov complexity [15]. When we select a reference universal Turing machine $U$ from $\mathcal{U}$ we can define the prefix-free Kolmogorov complexity $K(x)$ of an element $x \in \{0,1\}^*$ the length $l(p)$ of the smallest program $p$ that produces $x$ on $U$. We first define the conditional complexity:

**Definition 2.** $K_U(x|y) = \min_i\{l(\bar{i}) : U(\bar{i}y) = x\}$

The actual Kolmogorov complexity of a string is defined as the one-part code:

**Definition 3.** $K(x) = K(x|\varepsilon)$

Here all the compressed information, model as well as noise, is forced on to the model part.

**Definition 4.** *The randomness deficiency of a string $x$ is $\delta(x) = l(x) - K(x)$*

For two universal Turing machines $U_i$ and $U_j$, satisfying the invariance theorem, the complexities assigned to a string $x$ will never differ more than a constant:

$$|K_{U_i}(x) - K_{U_j}(x)| \leq c_{U_iU_j} \tag{1}$$

The invariance theorem also holds for the randomness deficiency of a string:

**Lemma 1.** $|\delta_{U_i}(x) - \delta_{U_j}(x)| \leq c_{U_iU_j}$

Proof: $|\delta_{U_i}(x) - \delta_{U_j}(x)| = |(l(x) - K_{U_i}(x)) - (l(x) - K_{U_j}(x))| = |K_{U_i}(x)) - K_{U_j}(x))| \leq c_{U_iU_j}\Box$

A string is Kolmogorov random with reference to a universal Turing machine if there is no program that compresses it. Using randomness deficiency the definition is:

**Definition 5 (Kolmogorov randomness).**
*$Random_U(x)$ iff $\delta_U(x) \leq c_r$*

Usually the constant $c_r$ is taken to be 0 in this definition. Note that *Random*(...) is a meta-predicate.

A choice for a universal reference Turing machine $U$ generates a specific so-called 'universal distribution' $m_U$ over the set of strings:

$$\log m_U(x) = K_U(x) + O(1) \tag{2}$$

Note that we can reduce the Kolmogorov complexity of any string $x$ to one bit by means of the naming function: just define $T_i(\varepsilon) = x$ and $\widehat{T_i}^U = "1"$, then $U(1\varepsilon) = x$. This shows that the naming function influences the Kolmogorov complexity of objects and the corresponding universal distribution. In general we will have $l(\widehat{T_i}) << l(\overline{T_i})$. A learning agent can on the basis of his experience update his probability distribution over the world by specifying short names for long strings that occur frequently.

Because of the asymptotic nature of Kolmogorov complexity, small objects will almost always be random, simply because there is not enough room to compress them. A useful concept in this context is the notion of a randomness threshold:

**Definition 6.** *A Turing machine $T_i$ is monotone if $\forall(k \in \mathbb{N})T_i(k+1) \geq T_i(k)$. The* randomness threshold *of a monotonically increasing machine $T_i$ with reference to a universal Turing machine $U$ is the smallest value of $k$ such that $U(\overline{T_i}k) = x$ and $l(\overline{T_i}k) < l(x)$.*

## 5.1 Variance of Turing Frames

In the following paragraphs we will only consider S5 Turing frames. The variance of a Turing frame is the largest distance between the Kolmogorov complexity assigned to strings by different universal Turing machines in the frame:

**Definition 7 (Variance of a frame).** *The* variance of a frame $\mathcal{F} = \langle \mathcal{U}_i, R\rangle$ *is*

$$Var(\mathcal{F}) = \min c_{U_x U_y}(\forall (U_x, U_y \in \mathcal{U}_i)\forall (z \in \{0,1\}^*)|K_{U_x}(z) - K_{U_y}(z)| \leq c_{U_x U_y})$$

There are frames with unbounded variance. Any infinite S5 Turing frame has unbounded variance:

**Lemma 2.** *If $\mathcal{U}$ is a countably infinite set of Turing machines and $\mathcal{F} = \langle \mathcal{U}, R\rangle$ is an S5 Turing frame , i.e. a frame defined on R that is symmetric, reflexive and transitive then $Var(\mathcal{F})$ is unbounded.*

Proof: Suppose that $Var(\mathcal{F}) = c$. Select a fixed universal machine $U_i$. For any other universal machine $U_j$ we have:

$$U_i(\overline{U_j}^{U_i}\overline{T}^{U_j}\emptyset) = U_j(\overline{U_i}^{U_j}\overline{T}^{U_i}\emptyset) = T(\emptyset) = x$$

Since the length of the indexes for an infinite class of objects cannot be bounded and $R$ is $S5$ there will be an infinite number of universal machines $U_j$ with index $\overline{U_j}^{U_i}$ such that:

$$l(\overline{U_j}^{U_i}\overline{T}^{U_j}) > l(\overline{U_j}^{U_i}) > l(\overline{U_i}^{U_j}\overline{T}^{U_i}) + c$$

Which gives:

$$|l(\overline{U_j}^{U_i}\overline{T}^{U_j}) - l(\overline{U_i}^{U_j}\overline{T}^{U_i})| > c$$

and

$$|K_{U_i}(x|\emptyset) - K_{U_j}(x|\emptyset)| = |K_{U_i}(x) - K_{U_j}(x)| > c$$

So $Var(\mathcal{F}) > c$. $\Box$

In such a frame the notions of Kolmogorov complexity and randomness deficiency are de facto undefined. We can always select a machine that compresses a string, or makes a random string compressible. The smaller the variation, the more exact our measures will be. By Solmonoff's invariance theorem the variation is limited by the size of the Turing machines in the frames, so this is a more formal variant of the class of 'natural' machines proposed by Hütter [12; 18]. To show how the choice of a universal Turing machine affects the notions of randomness, compressibility and variance we give an elaborate example.

*Example 1.* Suppose we have a simple universal Turing machine $U^n$ with a naming function $n$. $U^{[]}$, with an empty list is the machine $U$ with an empty naming table, $U^{max}$ is $U$ with a large naming table that contains all known mathematical functions. $U^{[T_{fp}]}$ is the version of $U$ that only has a name for the machine $T_{fp}$ that generates Fibonacci primes. Consider the set of Fibonacci primes: Fibonacci numbers that are also prime. $FB(i)$ denotes the $i$-th Fibonacci prime. It is not known whether this set is finite or not [2]. When we want to study this set with algorithmic complexity theory the choice of a universal machine is not neutral:

[2] At september 2015 the largest known certain Fibonacci prime was $F_{81839}$, with 17103 digits. See: https://en.wikipedia.org/wiki/Fibonacci_prime, retrieved october 9 2015.

- In the world $U^{[]}$ a possibly infinite number of Fibonacci primes will be compressible, but there will be quite a large intitial segment of smaller Fibonacci primes that is still regarded as random (e.g. for sure the elements of the set: $\{2, 3, 5, 13, 89, 233, 1597\}$) . Since $U^{[]}$ is minimal the code for Fibonacci primes has to be stored explicitely in the program $T_{fp}$ that recognizes Fibonacci primes, so an example like $FP(11) = 2971215073$ will be regarded as incompressible since for $U(\overline{T_{fp}}11) = 2971215073$ we will have $l(\overline{T_{fp}}11) > l(2971215073)$.
- When we shift to $U^{[T_{fp}]}$ the concept of Fibonacci primes can be called as a name $\widehat{T_{fp}}$ and thus smaller examples up to an initial segment will be compressible, e.g $U^{[T_{fp}]}(\widehat{T_{fp}}11) = 2971215073$ and $l(\widehat{T_{fp}}11) < l(2971215073)$.
- When we study invariance between $U^{[]}$ and $U^{[T_{fp}]}$ this advantage will vanish because of the length of the extra codes in the expression $U^{[T_{fp}]}(\overline{U^{[]}}\ \overline{T_{fp}}11) = U^{[]}\ (\overline{U^{[T_{fp}]}}\ \widehat{T_{fp}}11) = 2971215073$. The Kolmogorov complexity measured with $U^{[T_{fp}]}$ in this frame gets a penalty of at least $l(\overline{U^{[]}})$ and 2971215073 again is a random number.
- If we select $U^{max}$ as our reference machine, not much will change with regarding to the set of Fibonacci primes that is compressible. However the variance between $U^{[T_{fp}]}$ and $U^{max}$ will be big because of the length of the code for $U^{max}$ in the expression $U^{max}(\overline{U^{[T_{fp}]}}\ \widehat{T_{fp}}11) = U^{[T_{fp}]}(\overline{U^{max}}\ \widehat{T_{fp}}11) = 2971215073$. The Kolmogorov complexity measured with $U^{[T_{fp}]}$ in this frame gets a penalty of $l(\overline{U^{max}})$. If the set of Fibonacci primes is indeed finite and $l(\overline{U^{max}})$ is big, it might be the case that the whole set is not 'visible' any more from the perspective of $U^{[T_{fp}]}$.

Since we do not know wether the set of Fibonacci primes is infinite there might be choices of universal Turing machines for which the set of numbers that can be adequately modelled as Fibonacci primes is empty. A rich world with more information is certainly not always better then a poor world. In a world in wich we are only interested in Fibonacci primes $U^{[T_{fp}]}$ is a better choice than $U^{max}$ or $U^{[]}$. in the frame $\mathcal{F} = \langle\{U^{[T_{fp}]}, U^{max}\}, R\rangle$ there might be no strings that are compressed by $T_{fb}$ due to the large variance.

## 6 Meaningful Information

If we combine Shannon's notion of an optimal code with Bayes' law, we get a rough theory about optimal model selection. Let $\mathcal{H}$ be a set of hypotheses and let $x$ be a data set. Using Bayes' law, the optimal computational model under this distribution would be:

$$M_{map}(x) = argmax_{M \in \mathcal{H}} \frac{P(M)P(x|M)}{P(x)} \tag{3}$$

This is equivalent to optimizing:

$$argmin_{M \in \mathcal{H}} - \log P(M) - \log P(x|M) \tag{4}$$

Here $-\log P(M)$ can be interpreted as the length of the optimal *model code* in Shannon's sense and $-\log P(x|M)$ as the length of the optimal *data-to-model code*; i.e. the data interpreted with help of the model.

This derivation is quite misleading in so far as it suggests that the best model is always one piece of code or one computational predicate. Take the example of Fibonacci primes. These are primes that are both Fibonacci numbers and prime. They are in some contexts better described by the complex description $Fibonacci(x) \wedge Prime(x)$ then by some predicate $FibonacciPrime(x)$. So in the following we will be more interested in sets of predicates describing objects than in defining a single descriptive predicate.

### 6.1 Informative Predicates

We investigate various sets of predicates relevant to the notion of meaning. The set of informative predicates relative to a world $U_p$ is and a threshold function $f : \mathcal{T} \times \mathcal{U} \rightarrow \mathbb{N}$ :

**Definition 8 (Turing predicates informative relative to a world).**

$$\mathcal{P}_{U_p}(x) = \{T_i | \exists(k)(U_p(\overline{T_i}k) = x \wedge l(\overline{T_i}k) < l(x)))\}$$

The amount of information the associated predicate $P_i$ of a Turing machine $T_i$ carries about an object $x$ relative to $U_p$ is:

**Definition 9 (Amount of information in a Predicate).**

$$I_{U_p}(T_i, x) = \begin{cases} (l(\overline{T_i}^{U_p}) & \textit{if } \exists(k)(U_p(\overline{T_i}k) = x \wedge l(\overline{T_i}k) < l(x)) \\ 0 & \textit{otherwise.} \end{cases}$$

*i.e. the length of the index for $T_i$ on $U_p$ if it compresses $x$ and $0$ otherwise.*

This notion of informativeness is instable. There are cases in which we can make a string $x$ compressible by a predicate $T_i$ simply by defining a short name $\widehat{T_i}$ for it:

**Lemma 3 (Instability of Informativeness).** *There are combinations of strings $x$, Turing machines $T_i$, and universal Turing machines $U_m$ and $U_n$ such that $I_{U_m}(T_i, x) = 0$ and $I_{U_n}(T_i, x) > 0$.*

Proof: Choose $T_i$ and $x$ so that $\exists(k)(U_m(\overline{T_i}k) = x) \wedge (k << \underline{l(x))}$ and $I_{U_m}(T_i, x) = 0$, i.e. $x$ is not compressible for $U_m$ due to the length of $\overline{T_i}$. Define $U_n = U_m^s$, a machine that is equal to $U_m$ except for the naming function $s(\widehat{T_i}) = \overline{T_i}$, where $\widehat{T_i} << \overline{T_i}$ so that $U_m^s(\widehat{T_i}k) = x$ and $l(k) + l(\widehat{T_i}) < l(x)$. $\square$

The optimally informative predicates are a subset of the informative sets.

**Definition 10 (Optimally informative Turing predicates relative to a world).**

$$\mathcal{O}_{U_p}(x) = \{T_i | \exists(k)(U_p(\overline{T_i}k) = x \wedge l(\overline{T_i}k) < l(x)) \wedge$$
$$\neg(\exists(T_j, m)\}(U_p(\overline{T_j}m) = x \wedge (\overline{T_i}k) > (\overline{T_j}m))\}$$

The first condition $\exists(k)(U_p(\overline{T_i}k) = x \wedge l(\overline{T_i}k) < l(x))$ ensures that $T_i \in \mathcal{P}_{U_p}(x)$. The optimally informative predicates are a subset of the informative sets. The second condition $\neg(\exists(T_j, m)(U_p(\overline{T_j}m) = x \wedge (\overline{T_i}k) > (\overline{T_j}m)))$ ensures that $T_i$ gives maximal compression of $x$. This makes $\mathcal{O}_{U_p}(x)$ super instable, on top of the instability of $\mathcal{P}_{U_p}(x)$:

**Lemma 4 (Super Instability of Optimal Informativeness).** *There are combinations of strings $x$, Turing machines $T_i$, and universal Turing machines $U_m$ and $U_n$ such that $\mathcal{O}_{U_n}(x) = \{T_i\}$, what ever the content of $\mathcal{O}_{U_m}(x)$.*

Sketch of Proof: Along the lines of the proof of lemma 3. As soon as we shift from $U_m$ to a new world $U_n = U_m^s$ that is similar except for the fact that we have a a naming function $s(\widehat{T_i}) = \overline{T_i}$, that compresses the string $x$ one bit better than the ones in the current set $\mathcal{O}_{U_m}$ of optimally informative predicates, the whole set is replaced by $\mathcal{O}_{U_n} = \{T_i\}$. $\Box$

Super instability in this case implies that a small change in the definition of our universal machine makes the whole set of optimally informative predicates collapse on to one new predicate. For the set of informative predicates only individual predicates jump out of the set so they are more resilient.

### 6.2 Sophistication

This explains why we can never formulate an invariance proof for the notion of *sophistication* over a general set of acessible worlds. There have been attempts to define a two-part variant of Kolmogorov complexity as $\min_{ij}\{l(ij) : U(\bar{i}y) = x\}$ or as:

**Definition 11.** $K^2_U(x) = \min_{i,j}\{l(\bar{i}) + l(j) : U(\bar{i}j) = x\}$

It is easy to prove that $K$ is an upper bound for $K^2$ which is more expressive:

**Lemma 5.** $\forall(x) K_U(x) \geq K^2_U(x)$

Proof: If there exists for $K$ a one part code $U(k) = x$ that is more efficient than the optimal two-part code $\bar{i}j$, then $k$ would be chosen for $K^2$ as two-part program with empty input: $U(k\varepsilon) = x$. $\Box$

The underlying idea of two-part code optimization is that $\bar{i}$ allows us to separate the concatenation $\bar{i}j$ into its constituent parts, $i$ and $j$. Here $i$ is the index of a Turing machine which can be seen as capturing the *regular* part of the string $x$, and $j$ describes the input for the machine, i.e. the *irregular* part, e.g. errors in the model, noise and other missing information. In its turn $j$ could be seen as an index of $x$ in the set of objects that is defined by accepting computations of $i$. This is the motivation behind the definition of the notion of the *sophistication* of a string.

**Definition 12 (Sophistication).** $S_{U_p}(x) = \min_i\{l(\bar{i}) : i \in \mathcal{O}_{U_p}(x)\}$

Where $\mathcal{O}_{U_p}(x)$ is the set of optimally informative predicates. According to lemma 4 this set is extremely sensitive to shifts of universal Turing machines, on top of the variance of the super set $\mathcal{P}_{U_p}(x) \supseteq \mathcal{O}_{U_p}(x)$ due to lemma 3. Change a couple of bits in your reference machine and the whole set vanishes. The notion of sophistication under two-part code compression is less useful when we want to te able to vary our reference models.

### 6.3 Computational Meaning and Facticity

There have been many attempts to define the meaning of a string in terms of the *most informative predicate*, the one which generates the highest compression, but according to lemma 4 it is impossible to proof any form of invariance for this notion. The analysis above also shows that meta-predicates like "... is random", "... is informative" and "... is compressible" are contingent. For any string given a set of universal machines there might be predicates that are neccessarily informative in all worlds, possibly informative only in some worlds or neccessarily uniformative in all worlds. Moreover lemma 2 tells us that we have to select an a prioiri finite set of reference machines in a frame, even if we want to define classical Kolmogorov complexity. So we might just as well make this assumption explixit and definine the meaning of a string as a *set* of predicates that are *necessarily informative* in a finite set of accessible worlds. A predicate $P_i$ is *intrinsic* for a string $x$ if it compresses $x$ in all accessible worlds:

**Definition 13 (Computational meaning of string relative to a world).**

$$\mathcal{M}_{U,\mathcal{F}}(x) = \{\overline{T_i}^{U} | \forall (U_y \in \mathcal{W}_{U,\mathcal{F}})(I_{U_y}(\overline{T_i}^{U_y}, x) > 0)\}$$

The meaning of a string is the set of codes for the Turing machines associated with the intrinsic predicates. Note that the indexes $\overline{T_i}^{U}$ of the machines in the meaning set are in the format of the reference machine $U$, while the indexes used in the measurement $\overline{T_i}^{U_y}$ use the format of the specific machine. The facticity of a string is the amount of meaningful information in $x$, relative to a specific world and a frame is simply the Kolmogorov complexity of its meaning:

**Definition 14 (facticity).**

$$\varphi_{U,\mathcal{F}}(x) = K_U(\mathcal{M}_{U,\mathcal{F}}(x))$$

We have:

**Theorem 1 (Invariance of Facticity).** *For any string $x$ we have*

$$\varphi_{U_a,\mathcal{F}_a}(x) = \varphi_{U_b,\mathcal{F}_b}(x) + O(1)$$

Proof: The set accessible worlds for $U_a$ and $U_b$ are: $\mathcal{W}_{U_a,\mathcal{F}_a}$ and $\mathcal{W}_{U_b,\mathcal{F}_b}$

By definition 13 we have for the meaning of a string $x$ in these worlds:

$$\mathcal{M}_{U_a,\mathcal{F}_a}(x) = \{\overline{T_i}^{U_a} | \forall (U_y \in \mathcal{W}_{U_a,\mathcal{F}_a})(I_{U_y}(\overline{T_i}^{U_y}, x) > 0)\}$$

$$\mathcal{M}_{U_b,\mathcal{F}_b}(x) = \{\overline{T_i}^{U_b} | \forall (U_y \in \mathcal{W}_{U_b,\mathcal{F}_b})(I_{U_y}(\overline{T_i}^{U_y}, x) > 0)\}$$

Note that these definitions only differ on two points: 1) $U_a$ versus $U_b$ and 2) $\mathcal{W}_{U_a,\mathcal{F}_a}$ versus $\mathcal{W}_{U_b,\mathcal{F}_b}$. These are mathematical objects for which the conditional complexity is defined. This gives: $K(\mathcal{M}_{U_a,\mathcal{F}_a}(x)) =$

$$K(\mathcal{M}_{U_b,\mathcal{F}_b}(x)) + O(K(U_b|U_a)) + O(K(\mathcal{F}_b|\mathcal{F}_a)) = K(\mathcal{M}_{U_b,\mathcal{F}_b}(x)) + O(1)$$

Here $O(K(U_b|U_a))$ is the conditional complexity of the world shift and $O(K(\mathcal{F}_b|\mathcal{F}_a))$ is the conditional complexity of the frame shift. They are both constant for different $x$. By definition 14 we have $\varphi_{U_a,\mathcal{F}_a}(x) = \varphi_{U_b,\mathcal{F}_b}(x) + O(1)$. $\square$

**Lemma 6.** *In an transparant $S5$ Turing frame $\mathcal{F} = \langle \mathcal{U}_i, R\rangle$ where $R$ is an equivalence relation (symmetric, reflexive, transitive) we have for all strings $x$ and for all $U_a, U_b \in \mathcal{U}_i$*

$$\varphi_{U_a,\mathcal{F}}(x) = \varphi_{U_b,\mathcal{F}}(x) + Var(\mathcal{F})$$

Proof: According to the result in theorem 1 we have:

$$\varphi_{U_a,\mathcal{F}_a}(x) = \varphi_{U_b,\mathcal{F}_b}(x) + O(K(U_b|U_a)) + O(K(\mathcal{F}_b|\mathcal{F}_a))$$

The frame $\mathcal{F}$ is fixed. We are only left with variance between accessible worlds $O(K(\mathcal{W}_{U_b,\mathcal{F}}|\mathcal{W}_{U_a,\mathcal{F}}))$:

$$\varphi_{U_b,\mathcal{F}}(x) + O(K(U_b|U_a)) + O(K(\mathcal{W}_{U_b,\mathcal{F}}|\mathcal{W}_{U_a,\mathcal{F}})$$

Since $\mathcal{F}$ is $S5$ the set of accessible worlds is the same for every world in this frame:

$$\varphi_{U_b,\mathcal{F}_b}(x) + O(K(U_b|U_a))$$

Which by definition 7 gives:

$$\varphi_{U_a,\mathcal{F}}(z) = \varphi_{U_b,\mathcal{F}}(x) + Var(\mathcal{F})$$

$\square$

What theorem 1 tells us is that the less you specify about your discourse, the more vague the notion of meaning becomes. If we specify nothing about the relation $R$, the corresponding frame is associated with modal system $K$. In such a frame the notion of meaning is very unstable, because any shift to a new world can generate a new set of accessible worlds. In principle one could investigate the notion of meaning in the whole landscape of modal logics in this way, but that is not the goal of the present study.

Lemma 6 tells us why it is misleading to concentrate on small machines when defining Kolmogorov complexity. In principle a frame with small machines can still have a variance of the size of the biggest machine, on the other hand we could define a frame rich with information about our domain with low variance: for most induction problems a rich frame with low variance seems a better option.

## 7 Discussion

Although the underlying proofs are quite involved we end up with a simple computational theory: The computational meaning of a string $x$ relative to a frame is the set of Turing machines associated with intrinsic computational predicates of that string. The amount of meaning measured in bits is its facticity $\varphi(x)$, the Kolmogorov complexity of this set. We have to check that this definition satisfies our intuitions. Some observations:

- For each finite dataset there is only a finite number of predicates that are informative. This is clear from the following observation: suppose that there are an infinite number of informative predicates. In that case there is an infinite number of associated Turing machines with an index longer than $x$, but to be informative a machine should have at least an index shorter than $x$. Note that many of the predicates in the meaning set will be redundant. If a Fibonacci number is prime it is necessarily also a Fibonacci prime. We can estimate the Kolmogorov complexity of the meaning of a string on the basis of a *partial description* of the meaning, leaving out redundant predicates.
- For each finite dataset there is an infinite number of predicates that are true, but not informative: for any natural number $n$ there is a Turing machine which checks whether $x$ is smaller than $n$. The associated predicates of these Turing machines represent true facts about $x$ but they cannot all be informative.
- Random strings have no meaning, i.e. the set of meaningful predicates is empty. There are no informative predicates. This coincides with our intuition that high entropy sets contain no useful information.
- No string contains more meaning than information: $\varphi_{U,\mathcal{F}}(x) \leq K_{U,\mathcal{F}}(x)$. This is clear from the fact that $K(x)$ is the amount of bits we need to produce $x$, which given $U$ and $\mathcal{F}$ defines $\varphi_{U,\mathcal{F}}(x)$. This coincides with our intuition that simple strings have little meaning.

All informative predicates In [21] other requirements for a theory of computational theory of meaning are mentioned. We check the theory of facticity against these requirements:

- Facticity should count the bits required for an effective description of the structural properties of a binary string: the notion of meaning as the Kolmogorov complexity of the set of intrinsic predicates satisfies this requirement.
- An analogue of invariance should hold: there must be strict limits on how much $\varphi(x)$ can be affected by a change in programming language. We have studied this problem in depth and we have defined an invariance proof.
- Facticity should not be bounded by a constant: this issue is technical [19; 23]. Theories that select the shortest model as meaning, will, for very large data sets, always select a small universal Turing machine as their basic model. This problem is evaded in our theory by working with sets of informative

predicates. Such small universal machines will possibly occur in our set of informative predicates together with all other informative predicates, but they will not limit the facticity of a string.

- Similarly, there should be no constant $c$ such that $K(x) - \varphi(x) \leq c$ for all $x$, because then facticity would be equivalent to Kolmogorov complexity. This is clear from the fact that random strings have low facticity.

### 7.1 Unique descriptions

The interaction between meanings can be as complex as anything that human cognition can concieve or anything that is expressible in finite mathematics. Yet the are ways to tame this complexity. As an example take the number 2971215073, that is rather special. Here are some unique decriptions:

1. The 143130479th prime number
2. The 47th Fibonacci number, where 47 is the 15th prime number.
3. The 11th Fibonacci prime.
4. The largest known Fibonacci prime without any pair of consecutive equal digits.
5. The number that John uses as code for his safe.

The first three predicates could be intrinsic in some appropriate world. The last two are ad hoc.

Let $T_p$, $T_f$ and $T_{fp}$ be Turing machines that respectively generate primes, Fibonacci numbers and Fibonacci primes.

$$2971215073 = T_{fp}(11) = T_f(47) = T_f(T_p(15)) = T_p(143130479)$$

Chains of compressing programs

$$T_1(T_2(...T_i(T_j(...T_{n-1}(T_n(k))...))...)) = x$$

$$T_1(T_2(...T_{n-1}(T_n(k))...)) = x$$

$$l(\overline{T_1}) + l(\overline{T_1}) + ... + l(\overline{T_n}) + l(k) < x$$

## 8 Further research and applications

The computational meaning of a string is a complex set of predicates, each of which compresses the string to some extent in all accessible worlds. The restriction to finite sets of accessible worlds may seem too severe, but according to lemma 2, it is a necessary restriction for a workable definition of Kolmogorov complexity itself. So there is no loss of generality. This definition is more robust than sophistication that selects one optimally compressing program as the meaning of a string. Since facticity is itself measured as the Kolmogorov complexity of the set of informative predicates, in the end it is compressed to one program generating the set. This will most cases be close to the meaning measured by factivity, modulo the erratic jumps that sophistication can make in borderline cases where one predicate subsumes another. In the case of facticity both predicates will be in the meaning set, so such a jump in sophisitcation will not affect facticity.

Another big advantage of the theory of facticity is the fact that it allows an agent to discover the meaning of a string incrementally, by identifying predicates that compress the set partially. This allows for incremental discovery of meaning and solves many of the problems described in [17]. Meaning is essentially composite: e.g. the number 2971215073 is meaningful, because it is prime *and* it is a Fibonacci number. The two different predicates are in most contexts more informative than a single one. Especially because we can discover the fact that it is prime *independent* of the fact that it is a Fibonacci number. Another example: The Mona Lisa is a rich work of art because we can see many different meanings in it, not because it represents some monolithic Platonic idea of 'Mona-Lisa-ness'.

We specify various *application paradigms* of the theory.

- A *learning agent*, that tries to detect the meaning of a string by searching for predicates that compress the string. As soon as it discovers such a predicate it has to do two things: 1) Check whether the predicate is intrinsic and, 2) check whether is in not subsumed by predicates it already knows. If the agent is confronted with a sequence of data sets it might hone its expectation by formulating names for string that occur frequently.
- A *creative agent* tries to maximize the meaning in a data set. Such a *factic process* is by nature unstable. A creative agent manipulates data sets in such a way that the facticity increases, i.e. it will be compressed by an increasingly rich set of predicates that are mutually independent. Such a process will be subject to sudden phase transitions where the set collapses in to randomness.
- *Mixed groups of these agents* can be defined to model various social, economical and other processes (Pupil-teacher, Artists-public). A mixed group of creative and learning agents can develop what one could call a 'culture': a set of expectations about (i.e. names for) the artefacts that agents produce. For agents that do not share this culture a lot of artefacts will seem random, because they do not have access to the naming functions that compress them. item In this context notions as ethics, esthetics, trust and theory of other minds can be studied. For a group of agents living in a certain environment the accessibility relation will not be reflexive, symmetrical or transitive: i,e, they can observe the behaviour of themselves and others but they do not have access to the code to emulate themselves or others.

The learning agent and the creative agent represent variantions of the facticity concept along two orthogonal axes: variation of the agent and variation of the data. By formulating invariance conditions over classes of machines we are left with a stable platform to study phase transitions over data sets. The concept of facticity does not allow a data set to shift to randomness as a result of selecting a new universal machine. Yet such abrupt phaseshifts to randomness can occur also in the context of facticity when varying the data. Two other concepts of meaningful information may be helpful in this context: selfdissimilarity [11] and Local Entropy Variation (LEV) [22]. Both approaches reflect the idea that adding more meaning to a data set increases the internal 'tension' between space needed to encode traces of predicates that compress the data set and the global variation of internal differences in the data set. There will be classes of data sets that reflect these sudden phase transitions. In a future publication we intend to study these phenomena in depth.

## 9 Acknowledgements

I thank Steven de Rooij, Peter Bloem, Frank van Harmelen, Amos Golan, Erik Schultes, Peter van Emde Boas and my fellows at the Info-metrics Institute for many inspiring discussions. I thank the University of Amsterdam and especially Cees de Laat for allowing me the leeway to pursue my research interests. This research was partly supported by the Info-Metrics Institute of the American University in Washington, the Commit project of the Dutch science foundation NWO, the Netherlands eScience center, the IvI of the University of Amsterdam and a Templeton Foundations Science and Significance of Complexity Grant supporting The Atlas of Complexity Project.

## Bibliography

[1] Gottlob Frege. Begriffsschrift: eine der arithmetischen nachgebildete Formelsprache des reinen Denkens. Halle, 1879.

[2] Rudolf Carnap. The two concepts of probability: The problem of probability. *Philosophy and Phenomenological Research*, 5 (4): 513–532, 1945.

[3] Saul Kripke, 1963. "Semantical Considerations on Modal Logic", Acta Philosophica Fennica 16:8394

[4] Ray. J. Solomonoff. A formal theory of inductive inference: Parts 1 and 2. Information and Control, 7:1–22 and 224–254, 1964.

[5] M. Koppel, (1987) Complexity, Depth, and Sophistication", in Complex Systems 1, pages = 1087-1091.

[6] M. Koppel (1995) Structure, The universal Turing machine (2nd ed.): a half-century survey. pages 403-419. Springer Verlag.

[7] Hughes, G. E., and Cresswell, M. J. A New Introduction to Modal Logic. Routledge, 1996.

[8] Ray R. Solomonoff, "The Discovery of Algorithmic Probability", Journal of Computer and System Sciences, Vol. 55, No. 1, pp. 73-88, August 1997.

[9] Hopcroft, J. E., Motwani, R., Ullman, J. D., Introduction to Automata Theory, Languages, and Computation Second Edition. Addison-Wesley, 2001.

[10] M. Gell-Mann and S. Lloyd (2003) Effective complexity. In Murray Gell-Mann and Constantino Tsallis, eds. *Nonextensive entropy–Interdisciplinary applications*, Oxford University Press, 387-398.

[11] J. W. McAllister, (2003) Effective Complexity as a Measure of Information Content, Philosophy of Science, Vol. 70, No. 2, pp. 302-307.

[12] Marcus Hütter, Universal Artificial Intelligence: Sequential decisions based on algorithmic probability. Berlin: Springer, 2005.

[13] P.M.B. Vitányi, (2006) Meaningful information, IEEE Trans. Inform. 52:10, 4617 - 4626.

[14] D.H. Wolpert and W. Macready (2007) Using self-dissimilarity to quantify complexity: Research Articles, Complexity, volume 12,number 3, pages 77–85.

[15] Li M., Vitányi P.M.B. (2008), An Introduction to Kolmogorov Complexity and Its Applications, 3rd ed., Springer-Verlag, New York.

[16] P.W. Adriaans , Between Order and Chaos: The Quest for Meaningful Information, Theory of Computing Systems, Volume 45 , Issue 4 (July 2009), Special Issue: Computation and Logic in the Real World; Guest Editors: S. Barry Cooper, Elvira Mayordomo and Andrea Sorbi Pages 650-674, 2009.

[17] P. W. Adriaans and P. M. B. Vitányi, (2009) Approximation of the Two-Part MDL Code, Comput. Sci. Dept., Univ. of Amsterdam, Amsterdam; Information Theory, IEEE Transactions on, Volume: 55, Issue: 1, On page(s): 444-457.

[18] Samuel Rathmanner and Marcus Hütter, A Philosophical Treatise of Universal Induction, Entropy 2011, 13(6), 1076-1136.

[19] P..W. Adriaans, Facticity as the amount of self-descriptive information in a data set, arXiv:1203.2245 [cs.IT], 2012.

[20] Luís Antunes, Andre Souto, and Andreia Teixeira. 2012. Robustness of logical depth. In Proceedings of the 8th Turing Centenary conference on Computability in Europe: how the world computes (CiE'12), S. Barry Cooper, Anuj Dawar, and Benedikt Lwe (Eds.). Springer-Verlag, Berlin, Heidelberg, 29-34.

[21] Peter Bloem, Francisco Mota, Steven de Rooij, Luis Antunes, Pieter Adriaans: A Safe Approximation for Kolmogorov Complexity. ALT 2014: 336-350.

[22] Personal communication by Erik Schultes.

[23] Peter Bloem, Steven de Rooij, Pieter Adriaans: Two Problems for Sophistication. ALT 2015: 379-394.