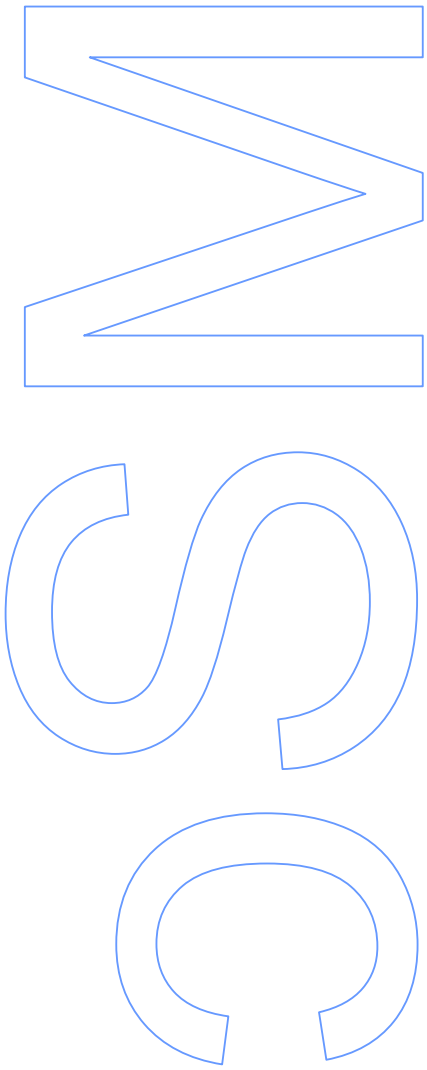
A Monotone Modal Logic for Algorithmic Statistics

Francisco Abreu Faro Mota
Mestrado em Ciência de Computadores
Departamento de Ciência de Computadores
2013

Orientador
Luís Filipe Coelho Antunes, Prof. Associado, Faculdade de Ciências da Universidade do Porto

Coorientador
André Nuno Carvalho Souto, Post. Doc., Instituto Superior Técnico



U.PORTO

FACULDADE DE CIÊNCIAS
UNIVERSIDADE DO PORTO

Todas as correções determinadas pelo júri, e só essas, foram efetuadas.

O Presidente do Júri,

Porto, _____/_____/_____

Abstract

According to Kolmogorov, a description of a string can be broken down into two parts: a model, and a data-to-model code. This is the motivation for Algorithmic Statistics, a field that studies the relation between the complexity of a model and the length of the data-to-model code. The field of Algorithmic Statistics is particularly concerned with sufficient models, a kind of model that contains no information beyond that which is needed to model the string. Unfortunately, sufficiency is not known to be stable: any operation on a sufficient model is unlikely to preserve its sufficiency.

We introduce *algorithmic models*, a broader notion of modeling in the context of Algorithmic Statistics, and show that algorithmic models are stable over all simple total recursive functions. We use algorithmic models to define Algorithmic Model Logic, a monotone modal logic for Algorithmic Statistics. We then discuss the possibility of generating multiple strings with the same model, and of mapping between models of different strings.

Keywords: Kolmogorov complexity, Algorithmic Information Theory, Algorithmic Statistics, Modal Logic, Intuitionistic Logic, Sophistication.

Resumo

A descrição de uma sequência tem duas partes: um modelo, e uma palavra de código. A Estatística Algorítmica é uma área que visa estudar a relação entre a complexidade do modelo e o comprimento da palavra de código. A Estatística Algorítmica tem um interesse especial por modelos suficientes, ou seja, modelos que não contêm informação para além da que é necessária para gerar a sequência. Infelizmente, este tipo de modelos não é necessariamente estável: qualquer operação sobre um modelo suficiente é capaz de fazer com que o modelo deixe de ser suficiente.

Para resolver este problem nós definimos *modelos algorítmicos*, uma reformulação mais abragente da modelação no contexto da Estatística Algorítmica. Demonstramos que os modelos algorítmicos são completamente estáveis relativos a funções totais recursivas simples. Usamos estes modelos para definir uma lógica modal monótona, a Lógica de Modelos Algorítmicos. Por fim, discutimos a possibilidade de modelar várias sequências com o mesmo modelo, e de mapear entre modelos de sequências diferentes.

Palavras-chave: Complexidade de Kolmogorov, Teoria da Informação Algorítmica, Estatística Algorítmica, Lógica Modal, Lógica Intuicionísta, Sofisticação.

Acknowledgements

I would like to thank my advisor Luís Antunes for the knowledge, wisdom, and patience he shared with me throughout this project. I would also like to thank Scott Aaronson and André Souto, without whom *Sophistication as Randomness Deficiency* [9], the predecessor to this dissertation, would not exist. Thank you to everyone who has helped me over the course of this degree and to everyone who has graciously given me much to think about and much to learn: Sandra Alves, Venkatesan Guruswami, Fernando Silva, Peter Blöem.

I could not be here without the love and support of my family. Thank you Rui, Ana, Filipe, Nuno, Judie, Pedro, MJ, Isabel, Carlos, Luisa, Óscar, Julia, and everyone else for giving me strength and courage and companionship.

And thank you, the reader, in advance.

Contents

Chapter 1

Introduction

Kolmogorov complexity measures the amount of information needed to reconstruct a string from scratch. For instance, a binary string that contains many long runs of the same bit will be highly compressible and therefore have a low Kolmogorov complexity, whereas a binary string that is generated using fair coin flips will have a high Kolmogorov complexity (with high probability).

However, just because a string has high Kolmogorov complexity does not mean that we cannot easily generate similar strings, or that we cannot figure out the properties of the original string without access to the original string. Consider the previous example. It is generally impossible to recreate an arbitrary string drawn from a series of fair coin flip if we don't know what those coin flips were, but it is very easy to generate a similar string by reflipping all the coins.

Let us put this principle in action. We flipped 50 coins and obtained the following string:

$$01001\,01010\,00001\,00110\,10000\,10101\,10111\,01111\,11011\,10101$$

We can see that this string has 24 zeros and its longest run is 6 bits long. It is highly unlikely that this string can be compressed, but we can easily reconstruct strings that are similar by flipping more coins. We do so 3 more times:

$$01011\,10100\,10101\,00100\,10110\,10001\,11000\,01010\,01111\,10000$$
$$10001\,01110\,00111\,10101\,00000\,11110\,00111\,00000\,00000\,10101$$
$$10100\,01010\,11001\,01010\,11100\,10010\,00110\,01001\,10100\,00100$$

The first string has 27 zeros, and its longest run is 5 bits long. The second string has 29 zeros, and its longest run is 10 bits long. The third string also has 29 zeros and its longest run is 3 bits long. We can see that, with some imprecision, the generated strings have a similar number of zeros as our original string. The length of the longest run of these strings is also within an acceptable range of our original string, but the connection isn't as clear.

This demonstrates a facet of this principle: we can approximately determine properties of the original string by looking at properties of similar strings, but as the complexity of the property increases, the accuracy of the approximation decreases. In this case, the property "roughly half of the bits are zero" is a simple property and is reproduced with accuracy, whereas "the longest run is around 6 bits long" is a more complex property and is reproduced with less accuracy.

This is what Algorithmic Statistics is about. Algorithmic Statistics studies the tolerances between the complexity of a model and the accuracy of its approximations. A model is a way of generating similar strings (in this case, by flipping coins). If we increase the complexity of the model we can obtain stronger approximations, until our model is as complex as the string itself and the "approximation" is the string itself (and therefore completely accurate).

A central concept in Algorithmic Statistics is the "algorithmic sufficient statistic", a kind of model that contains only as much information as is necessary to generate similar strings. This property of algorithmic sufficient statistics is the source of their power, and their greatest downfall: any operation performed on an algorithmic sufficient statistics is likely to knock a model out of this highly coveted status.

To cope with this shortcoming and develop a more user-friendly theory for Algorithmic Statistics, we introduce "algorithmic models". An algorithmic model can be used to generate an algorithmic sufficient statistic, but it is not itself an algorithmic sufficient statistic. This level of indirection allows us to show that algorithmic models are stable. From algorithmic models, we develop a monotone modal logic called Algorithmic Model Logic. This logic and its ramifications are the focus of this dissertation.

Contributions

We introduce algorithmic models (Chapter 3). We show that algorithmic models are stable over all simple total recursive functions, and satisfy the structural properties of intuitionistic sequent calculus (Section 3.1). We show that the Cut rule is invalid for algorithmic models, as is Pair Introduction (Section 3.2). We discuss variants of algorithmic models, and show that the original definition is the best possible definition with our current knowledge (Section 3.3). We discuss the transitivity of algorithmic models in depth (Section 4.6).

We introduce Algorithmic Model Logic (Sections 4.1, 4.2). We explore its logical properties and present a neighborhood model for Algorithmic Model Logic (Section 4.5).

We relate the complexity of algorithmic models to Koppel's sophistication and Antunes and Fortnow's coarse sophistication (Chapter 5). This allows us to define a different sophistication measure for every kind of algorithmic model (Section 5.1).

We consider the possibility and the ramifications of generating multiple strings with the same

model, presenting alternative ways of looking at this idea (Section 4.3). We also consider the interaction between maps and models (Section 4.4). We use these to define and study sophistication measures for multiple strings, Joint Sophistication and Shared Sophistication, and corresponding structure functions (Section 5.2). In so doing, we put into question Vereshchagin and Vitányi's equivalence between model classes [12].

This dissertation is also a continuation of the work in "Sophistication as Randomness Deficiency" [9], where we introduced *naive sophistication*, a sophistication measure based on naive statistics, and we drew new parallels between sophistication and computational depth, and the limits of lossy compression.

Chapter 2

Background

2.1 Notation

In this section we fix the notation used in the sequel.

Let $\mathbb{N}$ denote the set of natural numbers $\{0, 1, 2, \ldots\}$. Let Σ denote the binary alphabet $\{0, 1\}$. We will have Σ^n be the set of binary strings of length n, and Σ^* be the set of finite binary strings. Let λ denote the empty string. We denote string concatenation by juxtaposition: xy is x concatenated with y, where x and y are both strings. We define the prefix-of relation between strings: $x \sqsubseteq y$ if and only if there exists a $z \in \Sigma^*$ such that $xz = y$.

In standard notation, the modulo $|x|$ can mean either the absolute value of x, the string length of x, or the set cardinality of x. To decrease ambiguity, we will use this notation only to mean the absolute value of x. We will denote the length of a string x by $\operatorname{len} x$, and the cardinality of a set A by $\operatorname{size} A$.

If A is a set of strings, let $xA = \{\ xa\ :\ a \in A\ \}$ denote the prefixation of every element of A with x. Let $A + B = 0A \cup 1B$ denote the disjoint union of sets of strings A and B. Thus, $\operatorname{size}(A + B) = \operatorname{size} A + \operatorname{size} B$.

Due to its usefulness in information theory, we will always use the binary logarithm $\log_2$ to the exclusion of all other bases of logarithms. It will also be convenient for the logarithm function to always return a natural number. We define $\log$ as follows:

$$\log n = \min\{\ m \in \mathbb{N}\ :\ 2^m \geq n\ \}.$$

This means that $\log 0 = 0$ and $\log n = \lceil \log_2 n \rceil$ when $n > 0$.

Kolmogorov complexity is only defined up to a constant term, and algorithmic statistics is only meaningful up to a logarithmic significance level. So, to express approximate equalities

and inequalities, we will make judicious use of an error term. Let ε be the error term. This term functions similarly to $O(\log n)$, where multiple uses of ε in the same set of equations and inequalities do not necessarily have the same value. For example, we can say $\varepsilon + \varepsilon \leq \varepsilon$, which makes sense if you treat ε as syntactic sugar for $O(\log n)$. Let $x \approx y$ if and only if $|x - y| \leq \varepsilon$. Let $x \lesssim y$ if and only if $x \leq y + \varepsilon$.

We say that f is a partial recursive function if it is a partial function and there exists a Turing machine T such that when T is given input x and $f(x)$ is defined, it halts and outputs $f(x)$. When $f(x)$ is undefined, T loops forever. We identify a Turing machine with its partial function whenever convenient, so $T(x) = f(x)$. We say that f is a total recursive function if it is partial recursive and is also total.

2.2 The Frame Operator

In the sequel, it will be useful to encode arbitrary mathematical objects as strings using a prefix-free string encoding. To this end, we define the frame operator $[x]$, a canonical prefix-free string encoding for many kinds of mathematical objects.

For any natural number n, let $[n]$ be the Levenstein code of n:

$$[n] = \begin{cases} 0 & \text{if } n = 0 \\ 1\,[\log(n+1) - 1]\,\phi(n) & \text{if } n > 0 \end{cases}$$

where $\phi(n)$ is the standard binary representation of n without the first significant bit. In other words, $\phi(n)$ is the last $\log(n+1) - 1$ bits of the standard binary representation of n. The first several entries in this encoding are:

$$\begin{aligned} [0] &= 0, & [1] &= 10, \\ [2] &= 1100, & [3] &= 1101, \\ [4] &= 1110000, & [5] &= 1110001, \\ [6] &= 1110010, & [7] &= 1110011, \\ [8] &= 11101000, & [9] &= 11101001. \end{aligned}$$

This encoding has the property that $\operatorname{len}[n] = 1 + \sum_{k \geq 1} \log^{(k)}(n+1)$, where $\log^{(k)}$ denotes repeated application of the $\log$ function. This encoding satisfies Kraft's inequality exactly: $\sum_n 2^{-\operatorname{len}[n]} = 1$. This encoding is also universal [10, p.80].

Now that we have a prefix-free encoding of the natural numbers, we proceed to build a prefix-free

encoding of any finite binary string. Let x be a string of length n. Then, define

$$[x] = [n]\, x.$$

We can see that $\operatorname{len}[x] = \sum_{k\geq 0} \log^{(k)}(n+1)$. In the next section we will show that this encoding is optimal for most strings, up to a constant.

The frame operator generalizes easily to multiple values. Let $x_1, x_2, \ldots, x_k$ be a finite sequence. We define $[x_1, x_2, \ldots, x_k] = [k][x_1][x_2]\cdots[x_k]$. This will be the canonical way we encode tuples and sequences as strings. If A and B are sets, let $A \times B = \{\ [a,b]\ :\ a \in A, b \in B\ \}$ be the Cartesian product of A and B.

For any finite set $A = \{a_1, a_2, \ldots, a_k\}$ where the sequence $[a_1], [a_2], \ldots, [a_k]$ is lexicographically sorted, let

$$[A] = [k][a_1][a_2]\cdots[a_k].$$

This definition works for many kinds of sets. For example:

$$[\{1,2,3\}] = [3][1][2][3] = 11010111001101$$
$$[\{10,110\}] = [2][10][110] = 11001100101101110$$
$$[\{\emptyset,\{5\}\}] = [2][\emptyset][\{5\}] = 110001011100001$$

The first line is a prefix-free encoding of a finite set of natural numbers. The second line is a prefix-free encoding of a finite set of binary strings. The third line is a prefix-free encoding of a finite set of finite sets of natural numbers.

2.3 Kolmogorov Complexity

Kolmogorov complexity measures the amount of information contained in an individual string. Unlike Shannon's entropy, it is defined on a single string and not on a probability distribution. Nevertheless, it satisfies many of the same information theoretic properties, such as symmetry and stability.

A more thorough exploration of this topic is avaliable in Li and Vitányi [8].

Definition 1 (Kolmogorov Complexity)**.** *The Kolmogorov complexity of x given y is the length of the smallest program that outputs x when given y as input:*

$$\mathrm{K}(x|y) = \min\{\ \operatorname{len} p\ :\ U(p,y) = x\ \},$$

where U is an optimal universal prefix-free Turing machine. The prefix-freeness of U implies that if $U(p,y)$ is defined for some p and y, then $U(q,y')$ is undefined for all $q \sqsubset p$ and all

y'. *The optimality of* U *means that for any prefix-free Turing machine* T*, we have* $\mathrm{K}(x|y) \leq \min\{\ \mathrm{len}\, p\ :\ T(p,y) = x\ \} + c$ *where* c *is a constant that depends on* T *but not on* x *or* y*. Such a* U *exists according to the Invariance Theorem [8].*

Using the frame operator, we can easily extend this definition to different kinds of mathematical objects. For any mathematical objects x and y for which $[x]$ and $[y]$ are defined, let

$$\mathrm{K}(x|y) = \mathrm{K}([x]|[y]).$$

We can also omit the conditional y by setting it to the empty string λ. We therefore define the unconditional Kolmogorov complexity of x as $\mathrm{K}(x) = \mathrm{K}(x|\lambda)$.

Definition 2 (Total Kolmogorov Complexity)**.** *The total Kolmogorov complexity of* x *given* y *is the length of the smallest* total *program that outputs* x *when given* y *as input:*

$$\mathrm{KT}(x|y) = \min\{\ \mathrm{len}\, p\ :\ U(p,y) = x,\ p\ is\ total\ \}.$$

A program p *is total if and only if* $U(p,y)$ *is defined for any* y*.*

Note that $\mathrm{K}(x) = \mathrm{KT}(x|\epsilon)$, up to a constant error.

Definition 3 (Time-bounded Kolmogorov Complexity)**.** *Let* t *be a time-constructible function. The* t*-time-bounded Kolmogorov complexity of* x *given* y *is the length of the smallest program that outputs* x *when given* y *as input in the given time bound:*

$$\mathrm{K}^t(x|y) = \min\{\ \mathrm{len}\, p\ :\ U(p,y) = x\ in\ time\ t(\mathrm{len}\, x)\ \}.$$

Lemma 1 (Stability)**.** *Let* f *be a partial recursive function. Then:*

$$\begin{aligned}\mathrm{K}(f(x)|y) &\leq \mathrm{K}(x|y) + c\\ \mathrm{K}(x|y) &\leq \mathrm{K}(x|f(y)) + c\end{aligned}$$

where c *is a constant that depends only on* f*, not on* x *and* y*. Likewise, let* f *be a total recursive function. Then:*

$$\begin{aligned}\mathrm{KT}(f(x)|y) &\leq \mathrm{KT}(x|y) + c\\ \mathrm{KT}(x|y) &\leq \mathrm{KT}(x|f(y)) + c\end{aligned}$$

This is the Kolmogorov complexity analogue of the data processing inequality. It is also equivalent to the Invariance Theorem [8].

Definition 4 (Simple and Random Strings)**.** *Let* x *be a string. Let* $\varepsilon = \mathrm{O}(\log \mathrm{len}\, x)$*. We say* x *is simple if its complexity is approximately minimal, i.e.* $\mathrm{K}(x) \approx 0$*. We say* x *is random if its complexity is approximately maximal, i.e.* $\mathrm{K}(x) \approx \mathrm{len}\, x$*.*

Lemma 2 (Incompressibility). *The vast majority of strings are random. The probability that a string x taken uniformly from Σ^n is not random is less than $2^{-\varepsilon} = 1/\operatorname{poly}(n)$.*

Lemma 3 (Symmetry of Algorithmic Information). *For any x and y, we have that:*

$$\mathrm{K}(x,y) \approx \mathrm{K}(x) + \mathrm{K}(y|x) \approx K(y) + K(x|y)$$

where $\varepsilon = \mathrm{O}(\log \mathrm{K}(x) + \log \mathrm{K}(y))$. This result is known as Symmetry of Algorithmic Information, and it is one of the central results of Kolmogorov complexity theory. A standard proof can be found in [4].

2.4 Algorithmic Statistics

Kolmogorov observed that the shortest description of a string x can be broken down into two parts: a model, and a data-to-model code. The model captures all of the structure and regularity of x, whereas the data-to-model code captures the noise of x. This is called a two-part description of x.

Some strings, such as random strings and simple strings, have short two-part descriptions with simple models. However, there are also strings for which every simple model results in a longer two-part description. For these strings, if we allow the complexity of the model to increase, we may be able to develop shorter two-part codes. To formalize this intuition, Kolmogorov defined the structure function $h_x(\alpha)$ as follows:

$$h_x(\alpha) = \min \{ \ \log \operatorname{size} A \ : \ x \in A, \mathrm{K}(A) \leq \alpha \ \}.$$

In this definition, finite sets are used as models, and the data-to-model code is the index of x in the set. This definition stems from the fact that for any set A containing x, we have $\mathrm{K}(x) \leq \mathrm{K}(A) + \log \operatorname{size} A$. Thus, as $h_x(\alpha)$ approaches the line $K(x) - \alpha$, the two-part description becomes shorter and shorter. We say that a model A is an algorithmic sufficient statistic of x if it minimizes the length of the two-part code.

Definition 5 (Statistic). *For the purposes of this dissertation, a set A is a statistic of x if and only if A is finite and it contains x.*

Definition 6 (Algorithmic Sufficient Statistic). *A statistic A of x is an algorithmic sufficient statistic if and only if $\mathrm{K}(A) + \log \operatorname{size} A \approx \mathrm{K}(x)$, with error $\varepsilon = \mathrm{O}(\log \operatorname{len} x)$.*

If A is an algorithmic sufficient statistic of x, then x is indistinguishable from other elements of A. We can formalize this using randomness deficiency.

Definition 7 (Randomness Deficiency). *The randomness deficiency of x in A:*

$$\delta(x|A) = \log \operatorname{size} A - \mathrm{K}(x|A).$$

By convention we say that $\delta(x|A) = \infty$ if A is not a statistic of x.

Definition 8 (Naive Statistic). *A set A is a naive statistic of x if and only if $\delta(x|A) \approx 0$, with error $\varepsilon = \mathrm{O}(\log \operatorname{len} x)$. Equivalently, we say that x is typical in A.*

By Symmetry of Algorithmic Information, we know that A is an algorithmic sufficient statistic of x if and only if A is a naive statistic of x and $\mathrm{K}(A|x) \approx 0$. Therefore every algorithmic sufficient statistic is a naive statistic. The converse is not necessarily true, but we can show that every naive statistic can be converted into an algorithmic sufficient statistic.

Lemma 4 (Naive $\approx$ Algorithmic Sufficient). *If A is a naive statistic of x, then there exists an algorithmic sufficient statistic A' of x with $\mathrm{K}(A') \leq \mathrm{K}(A)$ and $\log \operatorname{size} A' \leq \log \operatorname{size} A$.*

Proof. Lemma A.4 of Vereshchagin and Vitányi [12] showed that for every statistic A of x there exists a statistic A' of x such that

$$\begin{aligned} \mathrm{K}(A') &\lesssim \mathrm{K}(A) - \mathrm{K}(A|x), \\ \log \operatorname{size} A' &\leq \log \operatorname{size} A. \end{aligned}$$

Repeated application of their lemma yields an algorithmic sufficient statistic of x with $\mathrm{K}(A') \leq \mathrm{K}(A)$ and $\log \operatorname{size} A' \leq \log \operatorname{size} A$. ■

Different model classes can be used. Vereshchagin and Vitányi [12] showed that we can replace finite sets with total recursive functions, or computable probability distributions, with at most a logarithmic change in the resulting structure function. In other words, the logarithmic error term allows us to switch between model classes freely. In Section 5.2, we show that the choice of model class isn't quite irrelevant, so we introduce functional statistics.

Definition 9 (Functional Statistics). *A total recursive function f is a statistic of x if there exists a d such that $f(d) = x$. Let p be the shortest and lexicographically first program of U that computes f. That is, for all d, we have $U(p, d) = f(d)$. We define $[f] = p$. Then we define:*

$$\begin{aligned} \mathrm{K}_f(x) &= \min \{ \ \operatorname{len} d \ : \ f(d) = x \ \}, \\ \delta(x|f) &= \mathrm{K}_f(x) - \mathrm{K}(x|f), \\ &= \mathrm{K}_f(x) - \mathrm{K}(x|p). \end{aligned}$$

We say that f is a naive statistic of x if and only if $\delta(x|f) \approx 0$. We say that f is an algorithmic sufficient statistic of x if and only if $\mathrm{K}(f) + \mathrm{K}_f(x) \approx \mathrm{K}(x)$.

2.5 Sophistication

To study Kolmogorov's structure function, Moshe Koppel [7] introduced sophistication, which measures the minimal complexity of an algorithmic sufficient statistic. In so doing, sophistication measures the amount of structural information in a string.

Definition 10 (Sophistication)**.** *The sophistication of x is the minimal complexity of an algorithmic sufficient statistic of x:*

$$\mathrm{soph}(x) = \min_A \{\ \mathrm{K}(A)\ :\ \mathrm{K}(A) + \log \mathrm{size}\, A \approx \mathrm{K}(x)\ \}$$

Here, the error term ε specifies the significance level of soph. We can specify ε as a subscript when we wish to be precise: $\mathrm{soph}_\varepsilon(x) = \min_A \{\ \mathrm{K}(A)\ :\ \mathrm{K}(A) + \log \mathrm{size}\, A \leq \mathrm{K}(x) + \varepsilon\ \}$. Otherwise, we let ε be defined by the context, or we let $\varepsilon = \mathrm{O}(\log \mathrm{len}\, x)$.

Sophistication is brittle: small increases in the significance level can result in large decreases in sophistication. To combat this issue, Antunes and Fortnow [2] proposed an alternative measure of structural information, coarse sophistication, that does away with the significance level ε.

Definition 11 (Coarse Sophistication)**.** *The coarse sophistication of x is the minimal complexity of an algorithmic sufficient statistic of x, penalized by how inefficient the model is:*

$$\begin{aligned}\mathrm{csoph}(x) &= \min_\varepsilon \{\ \mathrm{soph}_\varepsilon(x) + \varepsilon\ \}, \\ &= \min_A \{\ \mathrm{K}(A) + (\mathrm{K}(A) + \log \mathrm{size}\, A - K(x))\ \}.\end{aligned}$$

One of the main advantages of coarse sophistication over Koppel's sophistication is its stability. If f is a simple total recursive function, then $\mathrm{csoph}(f(x)) \lesssim \mathrm{csoph}(x)$. We present a simple proof of this in Chapter 5. Unfortunately, $\mathrm{csoph}(x)$ is not necessarily related to any algorithmic sufficient statistic of x: there exist strings x such that $\mathrm{csoph}(x)$ is low, but the complexity of every algorithmic sufficient statistic of x is high.

There also exist strings x of any length n such that $\mathrm{csoph}(x) \approx n/2$. We will say that these strings are *computationally deep*. In general, we will say that some string x is computationally deep whenever $\mathrm{csoph}(x) = \Omega(n)$, and we will say that some string x is computationally shallow whenever $\mathrm{soph}(x) \approx 0$. Computationally deep strings have no simple algorithmic sufficient statistics, not even if the error term ε is high, whereas computationally shallow strings admit simple algorithmic sufficient statistics.

Chapter 3

Algorithmic Models

One of the difficulties in working with algorithmic sufficient statistics directly is that they can only contain exactly as much information as is needed to generate their modeled strings. This is reasonable: the algorithmic sufficient statistic is sufficient precisely because it contains no unnecessary information. But this requirement implies that if we add more information, or if we perform any kind of operation on the statistic, we may no longer have an algorithmic sufficient statistic. This is why sophistication is brittle. It is difficult to perform any kind of manipulation on the models because there is no "algorithmic sufficient statistic" analogue to the stability lemma: the closest thing we have is stability for coarse sophistication, but coarse sophistication does not generally produce algorithmic sufficient statistics.

To rectify this issue and develop a more user-friendly theory, we propose a new definition and a new notation for Algorithmic Statistics that de-emphasizes working with algorithmic sufficient statistics directly. We propose the broader notion of *algorithmic models*. An algorithmic model allows us to obtain a naive statistic, which in turn implies the existence of an algorithmic sufficient statistic, thanks to Lemma 4.

Definition 12 (Algorithmic Models)**.** *Let x, y be any two strings. Let $\varepsilon = \mathrm{O}(\log \operatorname{len} y)$. We say that x is an algorithmic model of y, or x models y, written $x \models y$, if and only if there exists a set A such that:*

$$\mathrm{KT}(A|x) \approx 0$$
$$\delta(y|A) \approx 0$$

In other words, $x \models y$ if and only if x generates a naive statistic of y easily.

Examples. *If A is a naive statistic of y, then $A \models y$, because $\mathrm{KT}(A|A) \approx 0$. If y is random, then $\lambda \models y$, because $\Sigma^{\operatorname{len} y}$ is a naive statistic of y and $\Sigma^{\operatorname{len} y}$ is simple. By the same token, if y is simple then $\lambda \models y$, because $\{y\}$ is a naive statistic of y and $\{y\}$ is also simple. If f is a simple total recursive function, we have $x \models f(x)$. This implies the Axiom Rule of sequent calculus:*

for any x, we have $x \models x$.

The rest of this chapter will explore the properties of algorithmic models and their variants.

3.1 Stability

To justify the above definition, we will show that this relation is stable over all simple total recursive functions. As a result, the models relation satisfies the structural rules of (intuitionistic) sequent calculus.

Lemma 5 (Left-stability)**.** *The $x \models y$ relation is left-stable over all simple total recursive functions. Let f be a total recursive function with $K(f) \approx 0$. Then, $f(x) \models y$ implies $x \models y$.*

Proof. This is a trivial application of stability for total Kolmogorov complexity. Let $f(x) \models y$ and let A be the naive statistic of y generated by $f(x)$. We can show that

$$KT(A|x) \lesssim KT(A|f(x)) \approx 0.$$

Therefore, we can easily obtain A via x, so $x \models y$. ■

Corollary (Left Structural Rules)**.** *The $x \models y$ notation satisfies the left structural rules of intuitionistic sequent calculus. For any x, x', y we can show:*

Weakening:	$x \models y$	*implies*	$x, x' \models y$
Contraction:	$x, x \models y$	*implies*	$x \models y$
Exchange:	$x, x' \models y$	*implies*	$x', x \models y$

And many variants thereof, as straightforward applications of left-stability.

Right-stability is trickier to show. We break it down into two lemmas.

Lemma 6 (Invertible Right-stability)**.** *Let f be a total recursive function with $K(f) \approx 0$, and let y be any string with $KT(y|f(y)) \approx 0$. Then, for any x, we can show that $x \models y$ implies $x \models f(y)$.*

Proof. Let A be the naive statistic of y generated by x. Then, $B = \{\ f(a)\ :\ a \in A\ \}$ is a naive statistic of y. We can show the following inequalities:

$$K(y|A) \lesssim K(f(y)|B),$$
$$\log \text{size}\, B \leq \log \text{size}\, A.$$

As a result, $\delta(f(y)|B) \lesssim \delta(y|A) \approx 0$. Thanks to the stability of total Kolmogorov complexity, we also know that $K(B|x) \leq 0$. Therefore, $x \models y$. ■

Corollary (Right Structural Rules). *The $x \models y$ notation satisfies the right structural rules. Let x, y, y' be any strings, and let f be any simple total recursive function. Then,*

$$\begin{array}{rccc} \textbf{Weakening*:} & x \models y & \textit{implies} & x \models y, f(y) \\ \textbf{Contraction:} & x \models y, y & \textit{implies} & x \models y \\ \textbf{Exchange:} & x \models y, y' & \textit{implies} & x \models y', y \end{array}$$

as a straightforward consequence of Invertible Right-stability for algorithmic models.

Unlike sequent calculus, $x \models y, f(y)$ doesn't mean "x models y or $f(y)$" but "x models the pair $[y, f(y)]$". This divergence indicates that right structural rules may not have the same interpretation in our notation. We placed an asterisk on right weakening to indicate this disparity. The statements $x \models y$ and $x \models y, f(y)$ are equivalent whenever f is a simple recursive function, so this isn't actually a weakening. The following lemma is actually a weakening.

Lemma 7 (Pair Elimination). *Let x, y, y' be any strings. Then, $x \models y, y'$ implies $x \models y$.*

Proof. Let A be the naive statistic of $[y, y']$ that is generated by x. Let $A_a = \{\ b\ :\ [a, b] \in A\ \}$. Let $k = \log \operatorname{size} A_y$. Let $B = \{\ a\ :\ \log \operatorname{size} A_a = k\ \}$. We can quite easily show that $y \in B$, that $\log \operatorname{size} B \approx \log \operatorname{size} A - k$, and that, $\log \operatorname{size} A \lesssim \mathrm{K}(y, y'|A) \lesssim \mathrm{K}(y|A) + k \lesssim \mathrm{K}(y|B) + k$. As a result $\delta(y|B) \approx 0$. So B is a naive statistic of y, and we can obtain B easily if we know $[x, k]$. Since $\operatorname{len} k \approx 0$, we have $x \models y$. ■

Together, Invertible Right-stability and Pair Elimination can be used to show right-stability over all simple recursive functions.

Lemma 8 (Right-stability). *Let f be a total recursive function with $K(f) \approx 0$. Then $x \models y$ implies $x \models f(y)$.*

Proof. Invertible Right-stability tells us that $x \models y$ implies $x \models f(y), y$ when f is a simple total recursive function. Pair Elimination then gives us $x \models f(y)$. ■

Theorem 1 (Stability for Algorithmic Models). *Let f be a total recursive function with $K(f) \approx 0$. Then, for any strings x and y:*

$$\begin{array}{rcl} f(x) \models y & \textit{implies} & x \models y, \\ x \models y & \textit{implies} & x \models f(y). \end{array}$$

That is, the algorithmic models relation is stable for all simple total recursive functions.

3.2 Cut, Transitivity, and Pair Introduction

The Cut rule is not valid for algorithmic models. If it were valid, Cut would show that we can derive $x \models z$ from $x \models y$ and $x, y \models z$ for any x, y, z. Cut is logical deduction, but it does

not hold for algorithmic models in general. As a result, the $x \models y$ notation does not follow one of the basic rules of logic, in spite of its structural properties. Thanks to its deep connection to algorithmic statistics, the resulting "sequent calculus" may still be of interest, but we will remedy this flaw in the next chapter. In this section we explain why the Cut rule is invalid, and its relation to Transitivity and Pair Introduction.

Conjecture (Transitivity)**.** *If $x \models y$ and $y \models z$, then $x \models z$, for any strings x, y, z.*

Justification. If x can be used to obtain a model of y, and y can be used to obtain a model of z, we should be able to obtain a model of z directly from x. We believe this conjecture to be true, but we have not been able to prove it. We discuss this conjecture in depth in Section 4.6. ■

Conjecture (Weak Pair Introduction)**.** *If $x \models y$ then $x \models x, y$ for any strings x, y.*

Justification. If x can be used to obtain a model of y, we could potentially use x twice to create a model of $[x, y]$. We believe this conjecture to be false, however, because the pair $[x, y]$ can be computationally deep even if x and y are computationally shallow. Our proofs below shows that Strong Pair Introduction ($x \models y$ and $x \models z$ implies $x \models y, z$) is invalid, and Weak Pair Introduction is invalid if Transitivity is valid. ■

Lemma 9 (Cut $\iff$ Transitivity $\wedge$ Weak Pair Introduction)**.** *Cut is valid if and only if Transitivity and Weak Pair Introduction are both valid.*

Proof. (Cut $\implies$ Transitivity) Let $x \models y$ and $y \models z$. By Left-stability, $x, y \models z$. By Cut, $x \models z$. (Cut $\implies$ Weak Pair Introduction) Let $x \models y$. Let $z = [x, y]$. By the Axiom rule, $x, y \models x, y$. By Cut, $x \models x, y$. ($\Longleftarrow$) Let $x \models y$ and $x, y \models z$. By Weak Pair Introduction, $x \models x, y$. By Transitivity, $x \models z$. ■

Theorem 2 (Cut is invalid)**.** *There are infinitely many strings x, y, z such that $x \models y$ and $x, y \models z$ but $x \not\models z$.*

Proof. Let z be a computationally deep string. Let x be a random string independent of z, with the same length as z. Let $y = x \oplus z$, that is, y is x XORed with z bitwise. By construction $\lambda \models x$ and $\lambda \not\models z$ and $x \models y$ and $x, y \models z$.

By way of contradiction, let us assume that Cut is valid. Cut would allow us to show that $x \models z$. Cut implies Transitivity, so we could then show $\lambda \models z$, which is a contradiction. Therefore Cut is invalid. ■

Using the trick from the previous theorem, we can show the general version of pair introduction is invalid in a strong way.

Theorem 3 (Strong Pair Introduction is invalid)**.** *For any string x, there are infinitely many strings y, z such that $x \models y$ and $x \models z$ but $x \not\models y, z$.*

Proof. Let w be a computationally deep string of length $m \geq 2^{\operatorname{len} x}$. Let y be an independently random string of the same length. Let $z = y \oplus w$. Once again, we have $\lambda \models y$ and $\lambda \models z$ and $\lambda \not\models w$. Since $\operatorname{len} x = O(\log m) \approx 0$, we have that $x \models y$ and $x \models z$ and $x \not\models w$. Due to right-stability, we know that $x \not\models y, z$. ■

3.3 Variants

In this section we present variations on the models relation, and different kinds of algorithmic models. There are two ways we can change the models relation $x \models y$: we can change how much power we give to generate statistics from x, and we can change how powerful these statistics have to be as statistics of y. We call the former the *generating power* of the relation, and we call the latter the *statistical power* of the relation. The algorithmic models relation uses simple total recursive functions as its default generating power, and it uses naive statistics as its default statistical power. We present alternatives to these defaults, and how they affect the properties of $x \models y$.

Partial Recursive Generating Power: We can add more generating power by allowing any partial recursive function to transform x into the statistic A. In other words, we require only that $K(A|x) \approx 0$, not that $KT(A|x) \approx 0$. Let $x \models_K y$ if and only if there exists a naive statistic A of y such that $K(A|x) \approx 0$. This relation is left-stable for all simple partial recursive functions, but it cannot be right-stable for all simple partial recursive functions. In particular, it cannot be right-stable on U, the universal Turing machine. If it were, there would be no computationally deep strings. As a result, $\models_K$ cannot be transitive.

Algorithmic Sufficient Statistical Power: We can add statistical power by requiring that A be an algorithmic sufficient statistic. Let $x \models_a y$ if and only if there is an algorithmic sufficient statistic of y that can be easily obtained from x. Recall that an algorithmic sufficient statistic A of y is a naive statistic of y where $\mathrm{K}(A|y) \approx 0$. We can easily see that $x \models_a y$ implies $x \models y$, since every algorithmic sufficient statistic is a naive statistic. It is also easy to show that $x \models_a y$ is left-stable for all total recursive functions, and right-stable for all total recursive injections, but we have not been able to show Pair Elimination (Lemma 7) for $x \models_a y$. It is also unknown whether $x \models y$ implies $x \models_a y$, but it seems unlikely. We do know that if $x \models y$, there exists a string x' such that $\mathrm{K}(x'|x) \approx 0$ and $x' \models_a y$. This x' is obtained using Lemma 4 and its corollary.

Algorithmic Strong Sufficient Statistical Power: We can add further statistical power by requiring that A be a algorithmic strong sufficient statistic. Let $x \models_s y$ if and only if there is a algorithmic strong sufficient statistic of y that is easily obtainable from x. A algorithmic strong sufficient statistic A of y is a naive statistic of y such that $\mathrm{KT}(A|y) \approx 0$. Vereshchagin [13] introduced algorithmic strong sufficient statistics and showed that they have some nicer properties

than algorithmic sufficient statistics, such as the uniqueness (in some sense) of minimal strong sufficient statistics. Vereshchagin [14] showed that there are strings for which every algorithmic strong sufficient statistic has a much larger complexity than some algorithmic sufficient statistic. Thus, $x \models_s y$ is strictly stronger than $x \models_a y$. We can show that $x \models_s y$ is left-stable over total recursive functions, and right-stable over total recursive injections, but once again we have not been able to show Pair Elimination.

Closing thoughts: The big open question about $\models_a$ and $\models_s$ is whether they are right-stable. Increasing the generating power for these variants will not lead to right-stability, because statistics cannot be stable over all partial recursive functions. Reducing the generating power will not make $\models_a$ and $\models_s$ right-stable, because we only need to show Pair Elimination, i.e. that $\models_a$ and $\models_s$ are right-stable for the projection function $\pi_1[x, y] = x$. This is a very computationally weak function. If we choose a generating power that does not admit π_1 in order to establish right-stability, we would lose left-structural weakening. We would be throwing out the baby with the bath water.

This justifies our original decision to use naive statistics for algorithmic models. We don't know if more statistical power reduces stability, but we do know that $\models$ is stable. We know that $\models$ has the greatest amount of generating power that supports stability and the greatest amount of statistical power that is known to support stability.

Chapter 4

Algorithmic Model Logic

Let us now broaden the discussion. In the previous chapter we introduced the notation $x \models y$ to mean that a naive statistic of y can be obtained from x. We saw that this definition satisfies some structural properties, but falls short of being a logic by not admitting Cut.

In this chapter, we develop a notation $\phi \vdash \psi$ based on logical entailment, where ϕ and ψ are logical formulae with strings as atoms and with a modality M that creates algorithmic models. In this notation, $\langle\phi\rangle$ and $\langle\psi\rangle$ are sets of strings that represent these formulae, and $\phi \vdash \psi$ means that we can convert any string in $\langle\phi\rangle$ into a string in $\langle\psi\rangle$ with a small amount of additional information. The $x \models y$ notation is then equivalent to $x \vdash \mathrm{M}\, y$.

In so doing we develop Algorithmic Model Logic, an intuitionistic modal logic that is monotone and non-normal. Monotone non-normal modal logics are the kind of modal logic that talks about strategies that can be taken by agents. For a concrete example, Alur et al's Alternating Time Temporal Logic [1] belongs to this class of modal logics, as does Kyburg and Teng's Logic of Risky Knowledge [6] does as well.

4.1 Logical Operations

Following intuitionistic notions of realizability proposed by Kleene and Kolmogorov, Shen and Vereshchagin [11] defined an algebra of logical operations with which to study the relation between Kolmogorov complexity and logic. In this section, we introduce a variant of logical operations that uses our specific notion of a *map*, and we introduce the $\phi \vdash \psi$ notation.

Definition 13 (Map)**.** *Let A, B be sets of strings. We define a* map *between A and B to be any string x that allows us to translate elements of A into elements of B with a negligible amount of additional information. Formally, x maps A to B if and only if*

$$\forall a \in A.\ \exists b \in B.\ \mathrm{KT}(b \,|\, x, a) \approx 0.$$

We will denote the set of maps from A to B as $A \to_m B$.

Definition 14 (Logical Operations)**.** *We define a mapping from logical formulae to sets of strings. Let ϕ and ψ be logical formulae and let ε be an error term. Then we define:*

$$\langle \bot \rangle = \emptyset \qquad \langle \top \rangle = \Sigma^*$$
$$\langle \phi \wedge \psi \rangle = \langle \phi \rangle \times \langle \psi \rangle \qquad \langle \phi \vee \psi \rangle = \langle \phi \rangle + \langle \psi \rangle$$
$$\langle \phi \to \psi \rangle = \langle \phi \rangle \to_m \langle \psi \rangle \qquad \langle \neg \phi \rangle = \langle \phi \to \bot \rangle$$

Essentially, $\langle \phi \rangle$ is the set of strings that allow us to show that ϕ is valid. We supplement this notation by allowing any string x to be an atom of the logical formulae, and introducing a corresponding logical operation: $\langle x \rangle = \{x\}$. This supplement is not strictly necessary (as we show below), but it makes it easier to work with the resulting logic.

Notice that $\langle \phi \to \psi \rangle$ has an implicit ε term that specifies how much advice is permitted. We sometimes wish to be explicit about how large the ε term is for each of these maps. To do so, we can specify ε as a subscript. For example, $\langle (\phi \to_\varepsilon \psi) \to_{\varepsilon'} \xi \rangle$. To avoid clutter, we generally avoid this level of precision and rely on the reader's intuition instead.

Examples. *The formula $1100 \wedge 0010$ generates the set $\langle 1100 \wedge 0010 \rangle = \{[1100, 0010]\}$. The formula $1100 \vee 0010$ generates the set $\langle 1100 \vee 0010 \rangle = \{01100, 10010\}$. The formula $1100 \to 0010$ generates the set $\{\ x\ :\ \mathrm{KT}(0010 \,|\, x, 1100) \approx 0\ \}$. The formula $\langle 1100 \to_\varepsilon 0010 \rangle$ generates the set $\{\ x\ :\ \mathrm{KT}(0010 \,|\, x, 1100) \leq \varepsilon\ \}$.*

Definition 15 (Formula Encoding)**.** *We extend the frame operator by defining a prefix-free string encoding for logical formulae:*

$$[\bot] = 00 \qquad [\top] = 01$$
$$[\phi \wedge \psi] = 100\,[\phi]\,[\psi] \qquad [\phi \vee \psi] = 101\,[\phi]\,[\psi]$$
$$[\phi \to \psi] = 110\,[\phi]\,[\psi] \qquad [\neg \phi] = 110\,[\phi]\,00$$

If ϕ is the string x, we encode the formula ϕ as $111\,[x]$, where $[x]$ is the normal prefix-free encoding of an arbitrary string (using the frame operator). This encoding reflects the fact that $\neg\phi$ is shorthand for $\phi \to \bot$.

Definition 16 (Formula Complexity)**.** *We can measure the complexity required to satisfy a formula. Let ϕ be any formula. We then define:*

$$\mathrm{K}\langle \phi \rangle = \min_{x \,\in\, \langle \phi \rangle} \mathrm{K}(x).$$

More traditional notations for Kolmogorov complexity can be reformulated using this flexible notation. For example, total Kolmogorov complexity can be expressed as logical implication: $\mathrm{KT}(x \,|\, y) \approx \mathrm{K}\langle y \to x \rangle$ for any strings x and y.

Definition 17 (Entailment)**.** *We define a notation for logical entailment, based on formula complexity. Let $\phi \vdash \psi$ if and only if $\langle \phi \to \psi \rangle = \langle \top \rangle = \Sigma^*$. This is equivalent to requiring that for all $x \in \langle \phi \rangle$, there be a $y \in \langle \psi \rangle$ such that $\mathrm{KT}(y|x) \approx 0$ where $\varepsilon = \mathrm{O}(\log \mathrm{len}\,[\psi])$. Equivalently, $\phi \vdash \psi$ if and only if there exists a simple map from $\langle \phi \rangle$ to $\langle \psi \rangle$. Let $\phi \dashv\vdash \psi$ be shorthand for $\phi \vdash \psi$ and $\psi \vdash \phi$. Let $\phi \vdash \psi \vdash \xi$ be shorthand for $\phi \vdash \psi$ and $\psi \vdash \xi$.*

Lemma 10 (Properties of Logical Entailment)**.** *The $\phi \vdash \psi$ notation satisfies all of the properties of intuitionistic sequent calculus, and does not satisfy the law of the excluded middle. The proof is both straightforward and lengthy, so it has been omitted. In particular, $\vdash$ is reflexive and transitive.*

Lemma 11 (String Desugaring)**.** *From the perspective of entailment, strings as atoms of logical formulae are syntactic sugar. That is, for every formula ϕ with strings as atoms, there exists an equivalent formula ϕ' without with no strings as atoms.*

Proof. By definition:

$$\langle 0x \rangle = \langle x \vee \bot \rangle,$$
$$\langle 1x \rangle = \langle \bot \vee x \rangle.$$

Therefore, any formula containing strings as atoms can be turned into an equivalent formula containing no strings as atom except the empty string λ. Since $\lambda \dashv\vdash \top$, we can replace λ with $\top$ in such formulae and end up with no strings as atoms. However, the length of the desugared formula can be up to five times longer. ■

4.2 A Model Modality

Let us enhance the language above by introducing a modality for algorithmic models. We call the result "Algorithmic Model Logic" (AML).

Definition 18 (Model Modality)**.** *We define a modality for logical operations that gives us access to naive statistics. Let*

$$\langle \mathrm{M}\,\phi \rangle = \{\ [A]\ :\ \exists x \in \langle \phi \rangle.\ \delta(x|A) \approx 0\ \}\,.$$

That is, $\langle \mathrm{M}\,\phi \rangle$ is the set of naive statistics of elements of $\langle \phi \rangle$. If we wish to use other kinds of statistics (the ones used in section 3.4) we use subscripts on the modality. So $\langle \mathrm{M}_a\,\phi \rangle$ is the set of algorithmic sufficient statistics of $\langle \phi \rangle$, and $\langle \mathrm{M}_s\,\phi \rangle$ is the set of algorithmic strong sufficient statistics of $\langle \phi \rangle$.

The resulting modal logic subsumes the $x \models y$ notation neatly. For any strings x and y, the relation $x \models y$ holds if and only if $x \vdash \mathrm{M}\,y$. Using this connection we can retroactively define

the models relation for any two formulae. Let $\phi \models \psi$ if and only if $\phi \vdash \mathrm{M}\,\psi$.

We can use the new notation to rewrite the results from Chapter 3. For any strings x, y, z:

	Left-stability:	$x \vdash y \models z$	implies	$x \models z$
	equivalently:	$x \vdash y \vdash \mathrm{M}\,z$	implies	$x \vdash \mathrm{M}\,z$
	Right-stability:	$x \models y \vdash z$	implies	$x \models z$
	equivalently:	$y \vdash z$	implies	$\mathrm{M}\,y \vdash \mathrm{M}\,z$
(Conjecture)	**Transitivity:**	$x \models y \models z$	implies	$x \models z$
	equivalently:	$\mathrm{M}\,\mathrm{M}\,z \vdash \mathrm{M}\,z$	and	right-stability
(Open)	**Weak Pair Introduction:**	$x \models y$	implies	$x \models x \wedge y$
	equivalently:	$x \vdash \mathrm{M}\,y$	implies	$x \vdash \mathrm{M}(x \wedge y)$
(Invalid)	**Strong Pair Introduction:**	$x \models y,\ x \models z$	implies	$x \models y \wedge z$
	equivalently:	$\mathrm{M}\,y \wedge \mathrm{M}\,z$	$\vdash$	$\mathrm{M}(y \wedge z)$
(Invalid)	**Cut:**	$x \models y,\ x \wedge y \models z$	implies	$x \models z$
	equivalently:	$\mathrm{M}\,y \wedge (y \to \mathrm{M}\,z)$	$\vdash$	$\mathrm{M}\,z$

We also have a variant hierarchy from Section 3.3:

$$\forall x.\ \mathrm{M}_a\,x \vdash \mathrm{M}\,x,$$
$$\forall x.\ \mathrm{M}_s\,x \vdash \mathrm{M}_a\,x,$$
$$\exists^{\infty} x.\ \mathrm{M}_a\,x \not\vdash \mathrm{M}_s\,x.$$

The only open question in this regard is whether $\mathrm{M}\,x \vdash \mathrm{M}_a\,x$.

4.3 Sharing Models

The model of a string represents the string's structure, without necessarily any of the string's randomness. If two strings share the same structure, then they must be related, in some sense. Koppel [7] briefly touched on this idea when he introduced structure classes.

There are three distinct ways that we can formalize the sharing of models.

Model Product. The model product of y and z is $\langle \mathrm{M}\,y \wedge \mathrm{M}\,z \rangle$. Any string x such that $x \vdash \mathrm{M}\,y \wedge \mathrm{M}\,z$ gives us a way to generate a statistic of y and a statistic of z from the same

algorithmic model x. This is equivalent to saying $x \models y$ and $x \models z$.

Product Model. The product model of y and z is $\langle \mathrm{M}(y \wedge z) \rangle$. Any string x such that $x \vdash \mathrm{M}(y \wedge z)$ gives us a way to generate a statistic of $[y, z]$. This is equivalent to saying that $x \models y \wedge z$.

Shared Statistical Model. We say that A is a shared statistic of y and z if and only if $A \in \langle \mathrm{M}\, y \rangle$ and $A \in \langle \mathrm{M}\, z \rangle$. If we enrich the logical language with an "intersection" operator, such that $\langle \phi \cap \psi \rangle = \langle \phi \rangle \cap \langle \psi \rangle$, we can rewrite this condition as $A \in \langle \mathrm{M}\, y \cap \mathrm{M}\, z \rangle$. We say that x is a shared statistical model of y and z if and only if $x \vdash \mathrm{M}\, y \wedge \mathrm{M}\, z$.

Lemma 12 (Product Model $\subsetneq$ Model Product)**.** *Every product model can be converted into a model product, but there are infinitely many model products with no corresponding product model. In other words,* $\mathrm{M}(y \wedge z) \vdash \mathrm{M}\, y \wedge \mathrm{M}\, z$ *is valid, whereas* $\mathrm{M}\, y \wedge \mathrm{M}\, z \vdash \mathrm{M}(y \wedge z)$ *is not.*

Proof. Pair Elimination tells us that $\mathrm{M}(y \wedge z) \vdash \mathrm{M}\, y \wedge \mathrm{M}\, z$, and the invalidity of Pair Introduction tells us that there are infinitely many strings y and z for which the converse does not hold. ■

Lemma 13 (Shared Statistical Model $\subsetneq$ Model Product)**.** *Every shared statistical model is a model product, but not vice-versa. That is,* $\mathrm{M}\, x \cap \mathrm{M}\, y \vdash \mathrm{M}\, x \wedge \mathrm{M}\, y$ *for all* x *and* y*, but* $\mathrm{M}\, x \wedge \mathrm{M}\, y \nvdash \mathrm{M}\, x \cap \mathrm{M}\, y$ *for infinitely many* x *and* y*.*

Proof. Let $A \in \langle \mathrm{M}\, x \cap \mathrm{M}\, y \rangle$. Then, $[A, A] \in \langle \mathrm{M}\, x \wedge \mathrm{M}\, y \rangle$. Thus $\mathrm{M}\, x \cap \mathrm{M}\, y \vdash \mathrm{M}\, x \wedge \mathrm{M}\, y$.

Let x be a simple string of length n, and let y be a random string of the same length. Then $\lambda \vdash \mathrm{M}\, x$ and $\lambda \vdash \mathrm{M}\, y$. However, $\lambda \nvdash \mathrm{M}\, x \cap \mathrm{M}\, y$. Consider that for A to be a naive statistic of x, it must be satisfy $\log \mathrm{size}\, A \approx 0$ and for A to contain y, it must satisfy $\mathrm{K}(A) + \log \mathrm{size}\, A \geq n$. Therefore $\mathrm{K}(A) \geq n$. So $\lambda \nvdash A$ for any $A \in \langle \mathrm{M}\, x \cap \mathrm{M}\, y \rangle$, so $\lambda \vdash \mathrm{M}\, x \wedge \mathrm{M}\, y$ but $\lambda \nvdash \mathrm{M}\, x \cap \mathrm{M}\, y$. ■

4.4 Mapping Models

There are also three distinct ways to consider the interaction between functions and models.

Model Map: A string x is a *model map* between y and z if and only if $x \vdash \mathrm{M}\, y \to \mathrm{M}\, z$. This is a straightforward notion of maps between models, but these maps need not have any particularly nice property. For instance, $x \vdash \mathrm{M}\, y \to \mathrm{M}\, z$ does not imply $x \vdash y \to z$. This is in stark contrast to structural homomorphisms below.

Map Model: A string x is a map model between y and z if and only if $x \vdash \mathrm{M}(y \to z)$. Notice that all maps from y to z are also map models from y to z. That is, $x \to y \vdash \mathrm{M}(x \to y)$. We can show that not all map models can be converted into model maps (see below). We do not know whether all model maps can be converted into map models.

Structural Homomorphism: A total recursive function f is a structural homomorphism between y and z if and only if for any $A \in \langle \mathrm{M}\, y \rangle$ we have $B \in \langle \mathrm{M}\, z \rangle$ where

$$\begin{aligned} A_b &= \{\ a \in A\ :\ b = f(a)\ \}\,, \\ k &= \log \operatorname{size} A_z, \\ B &= \{\ b\ :\ \log \operatorname{size} A_b = k\ \}\,. \end{aligned}$$

More generally, f is a structural homorphism from $\langle \phi \rangle$ to $\langle \psi \rangle$ if and only if for every $y \in \langle \phi \rangle$ there exists a $z \in \langle \psi \rangle$ such that f is a structural homomorphism from y to z. We will denote the set of structural homomorphisms from $\langle \phi \rangle$ to $\langle \psi \rangle$ as $\langle \phi \to_M \psi \rangle$. This is the kind of function that we used to show Right Stability. As a result, $\lambda \vdash \phi \to \psi$ implies $\lambda \vdash \phi \to_M \psi$. This is not necessarily true if we replace λ with any string, however. The following lemma clarifies the relation between structural homomorphisms and maps.

Lemma 14 (Structural Homomorphisms $\subsetneq$ Maps)**.** *Every structural homomorphisms from $\langle \phi \rangle$ to $\langle \psi \rangle$ is a map from $\langle \phi \rangle$ to $\langle \psi \rangle$, and there are infinitely many formulae ϕ and ψ such that $\langle \phi \to_M \psi \rangle$ is strictly contained than $\langle \phi \to \psi \rangle$.*

Proof. Let $f \in \langle \phi \to_M \psi \rangle$. Let $y \in \langle \phi \rangle$. As we know, for any string y, the set $\{y\}$ is a naive statistic of y. Therefore $\{f(y)\} \in \langle \mathrm{M}\, \psi \rangle$. This implies that $f(y) \in \langle \psi \rangle$. Therefore, $f \in \langle \phi \to \psi \rangle$.

Now let us construct a map $g \in \langle \phi \to \psi \rangle$ that isn't in $\langle \phi \to_M \psi \rangle$, for infinitely many ϕ and ψ. Let x be any random string of length $2m$, and let y be any random string of length m. Let $g(x) = y$ by construction, and let $g(z) = z$ for any $z \neq x$. Consider the result of applying g to Σ^{2m} as if it were a structural homomorphism. We obtain the set $B = (\Sigma^{2m} \setminus \{x\}) \cup \{y\}$. We know that Σ^{2m} is a statistic of x, but B is not a statistic of y, since it is too large. Therefore g cannot be a structural homomorphism from x to y, despite being a map from x to y. ■

Corollary (Structural Homomorphisms $\subsetneq$ Map Models)**.** *Every map is a map model, so the set of structural homomorphisms is strictly contained in the set of map models.*

Theorem 4 (Structural Homomorphisms $\subsetneq$ Model Maps)**.** *All structural homomorphisms can be converted into model maps, but not all model maps can be converted into structural homomorphisms. In other words, $x \to_M y \vdash \mathrm{M}\, x \to \mathrm{M}\, y$ for all x, y, but $\mathrm{M}\, x \to \mathrm{M}\, y \nvdash x \to_M y$ does not imply $x \vdash y \to_M z$ for infinitely many x, y.*

Proof. That $x \to_M y \vdash \mathrm{M}\, x \to \mathrm{M}\, y$ is immediate from the definition of a structural homomorphism. To show that the converse is not necessarily true, let $x = \lambda$, and let y be a large random string. Clearly $\mathrm{M}\, x \vdash \mathrm{M}\, y$. But $x \nvdash y$. Therefore $\mathrm{M}\, x \to \mathrm{M}\, y \nvdash x \to_M y$. ■

Theorem 5 (Model Maps $\not\subset$ Map Models)**.** *There are infinitely many strings x, y such that $\mathrm{M}(x \to y) \nvdash \mathrm{M}\, x \to \mathrm{M}\, y$. In other words, not every map model can be converted into a model map.*

Proof. We reduce Strong Pair Introduction to $\mathrm{M}(x \to y) \vdash \mathrm{M}\, x \to \mathrm{M}\, y$. Let a, b, c be any three strings such that $a \vdash \mathrm{M}\, b \wedge \mathrm{M}\, c$. Let $x = c$, and $y = [b, c]$. Note that $b \vdash x \to y$, so $\mathrm{M}\, b \vdash \mathrm{M}(x \to y)$ thanks to right-stability. Therefore $a \vdash \mathrm{M}(x \to y)$, since $a \vdash \mathrm{M}\, b$. If $\mathrm{M}(x \to y) \vdash \mathrm{M}\, x \to \mathrm{M}\, y$, we have $a \vdash \mathrm{M}(b \wedge c)$. Thus we have shown that Strong Pair Introduction follows from $\mathrm{M}(x \to y) \vdash \mathrm{M}\, x \to \mathrm{M}\, y$. Since Strong Pair Introduction is invalid, so must $\mathrm{M}(x \to y) \vdash \mathrm{M}\, x \to \mathrm{M}\, y$ be. ■

Conjecture (Map Models $\subsetneq$ Model Maps)**.** *For any x and y, we have $\mathrm{M}\, x \to \mathrm{M}\, y \vdash \mathrm{M}(x \to y)$.*

4.5 Logical Properties

We will now discuss the logical properties of Algorithmic Model Logic. We will assume some familiarity with monotone modal logic for this section only, as it is beyond the scope of this dissertation to introduce this subject. We direct the interested reader towards Hansen [5].

It is clear that AML is a non-normal monotone modal logic. The defining properties of monotone modal logics are Necessitation and Monotonicity:

Necessitation:	$\top \vdash \psi$	implies	$\top \vdash \mathrm{M}\, \psi$
Mononoticity:	$\phi \vdash \psi$	implies	$\mathrm{M}\, \phi \vdash \mathrm{M}\, \psi$

We have already shown that these rules are valid. We have also shown that Normality is invalid:

Normality: $\mathrm{M}\, \phi \wedge \mathrm{M}\, \psi \vdash \mathrm{M}(\phi \wedge \psi)$

The fact that AML is monotone gives us access to a wealth of free theorems about algorithmic statistics. For example, the following statements can all be derived through logical manipulation:

Classicity:	$\phi \dashv\vdash \psi$	implies	$\mathrm{M}\, \phi \dashv\vdash \mathrm{M}\, \psi$
Axiom M:	$\mathrm{M}(\phi \wedge \psi)$	$\vdash$	$\mathrm{M}\, \phi \wedge \mathrm{M}\, \psi$
Axiom M':	$\mathrm{M}\, \phi \vee \mathrm{M}\, \psi$	$\vdash$	$\mathrm{M}(\phi \vee \psi)$
Axiom 4':	$\mathrm{M}\, \phi$	$\vdash$	$\mathrm{M}\, \mathrm{M}\, \phi$

Of these, the least trivial is Axiom M'. To demonstrate the simplifying power of Algorithmic Model Logic, we shall provide a standard proof of M', and a logical proof of M'.

Standard proof: Let $A \in \langle \mathrm{M}\, \phi \rangle$. Then $\{\ 0a\ :\ a \in A\ \} \in \langle \mathrm{M}(\phi \vee \psi) \rangle$, so $\mathrm{M}\, \phi \vdash \mathrm{M}(\phi \vee \psi)$. We can do the same for $\mathrm{M}\, \psi$. So if we have model for ϕ or a model for ψ, we can build a naive statistic for $\langle \phi \vee \psi \rangle$. Therefore $\mathrm{M}\, \phi \vee \mathrm{M}\, \psi \vdash \mathrm{M}(\phi \vee \psi)$.

Logical proof: Due to right disjunction introduction we know that $\phi \vdash \phi \vee \psi$ and $\psi \vdash \phi \vee \psi$. Thanks to monotonicity, $\mathrm{M}\, \phi \vdash \mathrm{M}(\phi \vee \psi)$ and $\mathrm{M}\, \psi \vdash \mathrm{M}(\phi \vee \psi)$. Due to left disjunction

introduction, we have $\mathrm{M}\,\phi \vee \mathrm{M}\,\psi \vdash \mathrm{M}(\phi \vee \psi)$.

All monotone modal logics have a neighborhood semantics. AML's neighborhood semantics are given by the algorithmic models relation we defined in Chapter 3.

Definition 19 (Neighborhood for Algorithmic Model Logic)**.** *Let us define a neighborhood function* $v : \Sigma^* \to \mathcal{P}\,\mathcal{P}\,\Sigma^*$*:*

$$v(x) = \{\ Y\ :\ \exists y \in Y.\ x \models y\ \}$$

The neighborhood for x *is the set of all sets that contain a string modeled by* x*. This neighborhood function gives us the neighborhood frame* $[\Sigma^*, v]$*. Together with a valuation of formulae to sets of strings,* $V(\phi) = \langle\phi\rangle$*, we have the neighborhood model* $[\Sigma^*, v, V]$ *for Algorithmic Model Logic.*

4.6 Transitivity of Models

In the previous chapter, we conjectured that $x \models y \models z$ implies $x \models z$. We now reveal our progress towards proving this conjecture. We show that in the presence of monotonicity, transitivity of $\models$ is equivalent to $\mathrm{M}\,\mathrm{M}\,x \vdash \mathrm{M}\,x$ for all x. In the study of modal logic, $\mathrm{M}\,\mathrm{M}\,\phi \vdash \mathrm{M}\,\phi$ is known as "Axiom 4". Applied only to strings, we shall call this "Axiom 4S". We then show that Axiom 4S is valid for M_a and M_s, and a variant of Axiom 4S is valid for M.

Lemma 15 (Transitivity $\iff$ Monotonicity $\wedge$ Axiom 4S)**.** *For any variant of* M*, transitivity is valid if and only if right-stability and axiom 4S are valid.*

Proof. (Transitivity $\implies$ Right-stability) Let $x \models y$ and $y \vdash z$. Since $y \vdash \{z\}$ and $\{z\} \in \langle\mathrm{M}\,z\rangle$ we have $y \models z$. Therefore $x \models z$, so we have right-stability.

(Transitivity $\implies$ Axiom 4S) Let $A \in \langle\mathrm{M}\,\mathrm{M}\,x\rangle$. By definition, there must be at least one $B \in A$ such that $B \in \langle\mathrm{M}\,x\rangle$ and $A \in \langle\mathrm{M}\,B\rangle$. Trivially, we have that $A \models B \models y$. By transitivity, $A \models y$. Therefore $\mathrm{M}\,\mathrm{M}\,y \vdash \mathrm{M}\,y$.

(Monotonicity $\wedge$ Axiom 4S $\implies$ Transitivity) Let $x \models y$ and $y \models z$. That is, $x \vdash \mathrm{M}\,y$ and $y \vdash \mathrm{M}\,z$. Thanks to monotonicity, we know that $x \models \mathrm{M}\,\mathrm{M}\,z$. Thanks to Axiom 4S, we get $x \vdash \mathrm{M}\,z$. Therefore $x \models z$. ■

Theorem 6 (Nearly Axiom 4S)**.** *For any* x*, we can show:*

$$\mathrm{M}_a\,\mathrm{M}\,x \vdash \mathrm{M}\,x$$
$$\mathrm{M}_a\,\mathrm{M}_a\,x \vdash \mathrm{M}_a\,x$$
$$\mathrm{M}_s\,\mathrm{M}_s\,x \vdash \mathrm{M}_s\,x$$

Proof. Let $x \in B \in A$ where B is a naive statistic of x and A is an algorithmic sufficient

statistic of B. In other words:

$$\begin{aligned}\mathrm{K}(B|A) &\approx \log \mathrm{size}\, A\\ \mathrm{K}(x|B) &\approx \log \mathrm{size}\, B\\ \mathrm{K}(A|B) &\approx 0\end{aligned}$$

Let $A_a = \{\ S \in A\ :\ a \in S\ \}$. Let $k = \log \mathrm{size}\, A_x$. Because x is typical in B and $\mathrm{K}(A|B) \approx 0$, we know that a non-negligible portion of elements of B have the same k. That is, let $C = \{\ a\ :\ \log \mathrm{size}\, A_a = k\ \}$. Then $\mathrm{K}(C|B) \approx 0$, so $\log \mathrm{size}(B \cap C) \approx \log \mathrm{size}\, B$. This shows that $\mathrm{K}(x|A) \approx \mathrm{K}(x|C) \approx \log \mathrm{size}\, A + \log \mathrm{size}\, B - k$. Note that $\log \mathrm{size}\, C \approx \log \mathrm{size}\, A + \log \mathrm{size}\, B - k$, since B is typical in A. Therefore, C is a naive statistic of x. Note that $\mathrm{KT}(C|A) \leq \mathrm{len}[k] \approx 0$, so $A \vdash C$. Therefore, $\mathrm{M}_a\, \mathrm{M}\, x \vdash \mathrm{M}\, x$.

If B is also an algorithmic sufficient statistic of x, then $\mathrm{K}(B|x) \approx 0$. Therefore, $\mathrm{K}(C|x) \approx \mathrm{K}(C|A) + \mathrm{K}(A|B) + \mathrm{K}(B|x) \approx 0$, so C is also an algorithmic sufficient statistic of x.

If A and B are algorithmic strong sufficient statistic of x, we have $x \vdash B \vdash A \vdash C$, so $x \vdash C$. Therefore C is also a algorithmic strong sufficient statistic of x. ■

This is as close to proving transitivity as we have come. This result tells us that $x \models_a \mathrm{M}\, y$ implies $x \models y$, that $x \models_a \mathrm{M}_a\, y$ implies $x \models_a y$, and that $x \models_s \mathrm{M}_s\, y$ implies $x \models_s y$. This result does not tell us that $\models_a$ or $\models_s$ are transitive, nor does it tell us that $x \models_a y \models z$ implies $x \models z$, because we have not shown that $\models_a$ and $\models_s$ are right-stable.

Chapter 5

Model Complexity

We now turn our attention towards the complexity of algorithmic models. What is the minimal complexity of a model for a certain string? If we know $x \models y$ and we know y, then what can we infer about the complexity of x?

It turns out that the complexity of algorithmic models is approximately equal to Koppel's sophistication (introduced in Section 2.5):

$$\begin{aligned} \operatorname{soph}(y) &= \mathrm{K}\langle \mathrm{M}_a\, y\rangle && \text{by definition,} \\ &\approx \mathrm{K}\langle \mathrm{M}\, y\rangle && \text{by Lemma 4,} \\ &\approx \min_{x \models y} \mathrm{K}(x) && \text{by definition.} \end{aligned}$$

We can also specify the significance level ε, for $\operatorname{soph}_\varepsilon(y) = \mathrm{K}\langle \mathrm{M}_{a\varepsilon}\, y\rangle$. Thus, we have:

$$\operatorname{csoph}(y) = \min_{\varepsilon} \left(\mathrm{K}\langle \mathrm{M}_{a\varepsilon}\, y\rangle + \varepsilon \right).$$

In this chapter we exploit these connections to develop new sophistication measures and, in particular, to look at the complexity of shared models.

5.1 Sophistication Measures

Definition 20 (Sophistication Measures)**.** *Let $x \models_* y$ be a variant of the $x \models y$ notation, such as those introduced in Section 3.3. We define a corresponding sophistication measure:*

$$\operatorname{soph}_*(y) = \min_{x \models_* y} \mathrm{K}(x).$$

We can also specify a significance level ε that controls how good the generated statistic must be. Note that this particular ε term must only affect the statistical power, not the generating

power. This allows us to also define a corresponding coarse sophistication measure:

$$\mathrm{csoph}_*(y) = \min\{\ \mathrm{K}(x) + \varepsilon\ :\ x \models_* y\ \}.$$

These are generalizations of Koppel's sophistication and Antunes and Fortnow's coarse sophistication because $\mathrm{soph}(y) \approx \mathrm{soph}_a(y)$ *and* $\mathrm{csoph}(y) \approx \mathrm{csoph}_a(y)$.

For example, $\mathrm{soph}_s(y) = \min\{\ \mathrm{K}(x)\ :\ x \models_s y\ \}$ is the minimum complexity of an algorithmic strong sufficient statistic of y, and $\mathrm{csoph}_s(y)$ is the corresponding coarse sophistication measure. Of the variants defined in Section 3.3, it is easy to see that $\mathrm{soph}(y) \approx \mathrm{soph}_a(y) \approx \mathrm{soph}_K(y)$.

The minimum complexity of a naive statistic, called *naive sophistication* and denoted by $\mathrm{nsoph}(y)$, was discussed in depth in Mota et al [9]. The definition above doesn't give us a way to discuss the quantity $\mathrm{nsoph}(y) = \mathrm{K}\langle \mathrm{M}\, y\rangle$ directly, but as we've seen, this is approximately equal to $\mathrm{soph}(y)$, so any result on nsoph is a result on soph and vice-versa.

The connection between coarse sophistication and algorithmic models yields a trivial proof that csoph is stable over all simple total recursive functions:

Lemma 16 (Stability of Coarse Sophistication)**.** *Let* x, y *be any two strings such that* $x \vdash y$. *Then* $csoph(y) \lesssim csoph(x)$.

5.2 Shared Model Complexity

Consider the shared models of Section 4.3: Model Products, Product Models, and Shared Statistical Models. Our goal in looking at these models was to look at how different strings can share the same structure. In this section, we will look at the complexity of Model Products and Shared Statistical Models. Let us define sophistication measures for these kinds of shared models.

Definition 21 (Joint and Shared Sophistication)**.** *The* joint sophistication *of* x *and* y *is the complexity of the model product of* x *and* y*:*

$$\mathrm{jsoph}(x, y) = \mathrm{K}\langle \mathrm{M}\, x \wedge \mathrm{M}\, y\rangle.$$

The shared sophistication *of* x *and* y *is the minimum complexity of a shared statistical model of* x *and* y*:*

$$\mathrm{ssoph}(x, y) = \mathrm{K}\langle \mathrm{M}\, x \cap \mathrm{M}\, y\rangle.$$

Like the above definition of sophistication measures, we can specify a significance level ε *to*

control the power of our statistics. We can then define coarse versions of the above definitions:

$$\begin{aligned}\mathrm{jcsoph}(x,y) &= \min_{\varepsilon}\{\ \mathrm{jsoph}_{\varepsilon}(x,y)+\varepsilon\ \},\\ \mathrm{scsoph}(x,y) &= \min_{\varepsilon}\{\ \mathrm{ssoph}_{\varepsilon}(x,y)+\varepsilon\ \}.\end{aligned}$$

It is easy to see that both jsoph and ssoph must dominate soph:

$$\begin{aligned}\mathrm{jsoph}(x,y) &\gtrsim \max\{\mathrm{soph}(x),\mathrm{soph}(y)\}\\ \mathrm{ssoph}(x,y) &\gtrsim \max\{\mathrm{soph}(x),\mathrm{soph}(y)\}\end{aligned}$$

We can also show that jsoph and jcsoph are additive, whereas ssoph and scsoph aren't. That is, $\mathrm{jsoph}(x,y)\lesssim \mathrm{soph}(x)+\mathrm{soph}(y)$ always, whereas $\mathrm{ssoph}(x,y)\gg \mathrm{soph}(x)+\mathrm{soph}(y)$ sometimes. However, if we switch to functional statistics, we can show that ssoph and scsoph are somewhat additive. This shows that although the model class used has at most a logarithmic effect when a single statistic is used for a single string, it has a profound effect when a single statistic is used for multiple strings. This places an uncomfortable asterisk on Vereshchagin and Vitányi's result on the equivalence of model classes [12].

Theorem 7 (Joint Sophistication is Additive)**.** *For any strings x and y,*

$$\begin{aligned}\mathrm{jsoph}(x,y) &\lesssim \mathrm{soph}(x)+\mathrm{soph}(y),\\ \mathrm{jcsoph}(x,y) &\lesssim \mathrm{csoph}(x)+\mathrm{csoph}(y).\end{aligned}$$

Proof. This is a very straightforward result. Let $A\in\langle \mathrm{M}\, x\rangle$ and let $B\in\langle \mathrm{M}\, y\rangle$. Then $[A,B]\in \langle \mathrm{M}\, x\wedge \mathrm{M}\, y\rangle$, and $K(A,B)\lesssim K(A)+K(B)$. Therefore $\mathrm{jsoph}(x,y)\lesssim \mathrm{soph}(x)+\mathrm{soph}(y)$.

As for jcsoph, consider the significance levels ε_A of A and ε_B of B. Then $[A,B]\in\langle \mathrm{M}\, x\wedge \mathrm{M}\, y\rangle$ with significance level $\varepsilon=\max\{\varepsilon_A,\varepsilon_B\}\leq \varepsilon_A+\varepsilon_B$. Therefore $\mathrm{jcsoph}(x,y)\leq \mathrm{csoph}(x)+\mathrm{csoph}(y)$. ■

Theorem 8 (Shared Sophistication is Not Additive)**.** *There exist infinitely many strings x and y such that:*

$$\begin{aligned}\mathrm{ssoph}(x,y) &\gg \mathrm{soph}(x)+\mathrm{soph}(y)\\ \mathrm{scsoph}(x,y) &\gg \mathrm{csoph}(x)+\mathrm{csoph}(y).\end{aligned}$$

Proof. Let x be a simple string of length n, and let y be a random string of length n. Any naive statistic A of x must have $\log \mathrm{size}\, A\approx 0$, and any naive statistic A of y must have $\mathrm{K}(A)+\log \mathrm{size}\, A\gtrsim n$. Thus, if A is a shared statistic of x and y, we know that $K(A)\approx K(A)+\log \mathrm{size}\, A\gtrsim n$. An example of such an A is the set $\{x,y\}$. So $\mathrm{ssoph}(x,y)\approx n$ whereas $\mathrm{soph}(x)\approx \mathrm{soph}(y)\approx 0$. This same kind of argument also works for $\mathrm{scsoph}(x,y)$, although it only shows that $\mathrm{scsoph}(x,y)\gtrsim n/2$ in this case. ■

Theorem 9 (Shared Sophistication with Functional Statistics is Additive). *For all strings x, y. Let $m = \max\{\mathrm{soph}_\varepsilon(x), \mathrm{soph}_\varepsilon(y)\}$. Then,*

$$\begin{aligned}\mathrm{ssoph}_{\varepsilon+m}(x,y) &\lesssim \mathrm{soph}_\varepsilon(x) + \mathrm{soph}_\varepsilon(y),\\ \mathrm{scsoph}(x,y) &\lesssim \mathrm{csoph}(x) + \mathrm{csoph}(y) + \min\{\mathrm{csoph}(x), \mathrm{csoph}(y)\}.\end{aligned}$$

Proof. Let f_x be a functional statistic for x, and let f_y be a functional statistic for y, with $\varepsilon_x = \delta(x|f_x)$ and $\varepsilon_y = \delta(y|f_y)$. Let g be a total recursive function defined by:

$$g(0d) = f_x(d), \qquad g(1d) = f_y(d).$$

Note that:

$$\begin{aligned}\mathrm{K}(g) &\lesssim \mathrm{K}(f_x) + \mathrm{K}(f_y),\\ \delta(x|g) &\lesssim \varepsilon_x + \mathrm{K}(f_y),\\ \delta(y|g) &\lesssim \varepsilon_y + \mathrm{K}(f_x).\end{aligned}$$

The theorem follows. ■

We do not know if these theorems are strict upper bounds. We can generalize joint and shared sophistication to more than two elements.

Definition 22 (Joint and Shared Sophistication of Sets). *Let $A = \{a_1, a_2, \ldots, a_k\}$ be a finite set. Then, we define:*

$$\begin{aligned}\mathrm{jsoph}(A) &= \mathrm{K}\langle \mathrm{M}\, a_1 \wedge \mathrm{M}\, a_2 \wedge \cdots \wedge \mathrm{M}\, a_k\rangle,\\ \mathrm{ssoph}(A) &= \mathrm{K}\langle \mathrm{M}\, a_1 \cap \mathrm{M}\, a_2 \cap \cdots \cap \mathrm{M}\, a_k\rangle.\end{aligned}$$

In this case, it is essential that the significance level $\varepsilon = \Omega(\Sigma_{a\in A} \log \mathrm{len}\, a)$ in order to establish flexibility between various possible representations of elements in $\langle \mathrm{M}\, a_1 \wedge \mathrm{M}\, a_2 \wedge \cdots \wedge \mathrm{M}\, a_k\rangle$. If A is large, this error term grows accordingly. We generalize jcsoph *and* scsoph *accordingly:*

$$\begin{aligned}\mathrm{jcsoph}(A) &= \min_\varepsilon \{\ \mathrm{jsoph}_\varepsilon(A) + \varepsilon\ \}\\ \mathrm{scsoph}(A) &= \min_\varepsilon \{\ \mathrm{ssoph}_\varepsilon(A) + \varepsilon\ \}\end{aligned}$$

We define $\mathrm{jsoph}(\emptyset) = \mathrm{ssoph}(\emptyset) = 0$.

How are these related? By definition,

$$\begin{aligned}\mathrm{jsoph}(\{x,y\}) &= \mathrm{jsoph}(x,y),\\ \mathrm{ssoph}(\{x,y\}) &= \mathrm{ssoph}(x,y).\end{aligned}$$

Also, $\mathrm{jsoph}(\{x\}) = \mathrm{ssoph}(\{x\}) = \mathrm{soph}(x)$. It is clear that $\mathrm{jsoph}(A) \gtrsim \max_{a\in A} \mathrm{soph}(a)$, since any model product for A gives a model for each $a \in A$. It is also clear that $\mathrm{jsoph}(A) \lesssim \mathrm{ssoph}(A)$ and $\mathrm{jcsoph}(A) \lesssim \mathrm{scsoph}(A)$, because any statistic generated by $\mathrm{ssoph}(A)$ can be used by $\mathrm{jsoph}(A)$ repeatedly.

Using the additivity proofs above, we can show that:

$$\begin{aligned}
\mathrm{jsoph}(A \cup B) &\leq \mathrm{jsoph}(A) + \mathrm{jsoph}(B) \\
\mathrm{jcsoph}(A \cup B) &\leq \mathrm{jcsoph}(A) + \mathrm{jcsoph}(B) \\
\mathrm{ssoph}_{\varepsilon+m}(A \cup B) &\leq \mathrm{ssoph}_{\varepsilon}(A) + \mathrm{ssoph}_{\varepsilon}(B) \\
\mathrm{scsoph}(A \cup B) &\leq \mathrm{scsoph}(A) + \mathrm{scsoph}(B) \\
&\quad + \min\{\mathrm{scsoph}(A), \mathrm{scsoph}(B)\}
\end{aligned}$$

where $m = \max\{\mathrm{ssoph}_{\varepsilon}(A), \mathrm{ssoph}_{\varepsilon}(B)\}$ and both ssoph and scsoph use functional statistics.

To conclude this section, we draw a parallel between joint and shared sophistication and variants of the structure function $\beta_x(\alpha)$ discussed in Vereshchagin and Vitányi [12]. The original definition is as follows:

$$\beta_x(\alpha) = \min_S \{\ \delta(x|S)\ :\ \mathrm{K}(S) \leq \alpha\ \}$$

We define two variants of this structure function for any finite set $A = \{a_1, a_2, \ldots, a_k\}$:

$$\begin{aligned}
\beta_A(\alpha) &= \min_{\vec{S}} \left\{\ \max_{1\leq i\leq k} \delta(a_i|S_i)\ :\ \mathrm{K}(\vec{S}) \leq \alpha\ \right\}, \\
\beta'_A(\alpha) &= \min_{f} \left\{\ \max_{1\leq i\leq k} \delta(a_i|f)\ :\ \mathrm{K}(f) \leq \alpha\ \right\},
\end{aligned}$$

where $\vec{S} = [S_1, S_2, \ldots, S_k]$ ranges over sequences of finite sets, and f ranges over total recursive function. We now draw the parallel between all of these:

$$\begin{aligned}
\mathrm{soph}(x) &= \min\{\ \alpha\ :\ \beta_x(\alpha) \approx 0\ \}, \\
\mathrm{jsoph}(A) &= \min\{\ \alpha\ :\ \beta_A(\alpha) \approx 0\ \}, \\
\mathrm{ssoph}(A) &= \min\left\{\ \alpha\ :\ \beta'_A(\alpha) \approx 0\ \right\}.
\end{aligned}$$

To study these variants of the β structure function is to study the complexity of shared models.

Chapter 6

Conclusion and Future Work

We have introduced algorithmic models and Algorithmic Model Logic in an attempt to develop a more user-friendly theory for Algorithmic Statistics. This new theory allows us to use logical deduction to prove theorems about Algorithmic Statistics. It also allows us to generalize sophistication in a principled way, and to define an additive joint sophistication measure.

Since this is a new theory, we have left behind a slew of open problems that would be nice to resolve in the future. We list these below, as well as some possible directions for future work in this area.

Open Problems

Transitivity: Does $x \models y \models z$ imply $x \models z$? Are $\models_a$ and $\models_s$ right-stable? Does $\mathrm{M}\, x \vdash \mathrm{M}_a\, x$?

Weak Pair Introduction: Does $x \models y$ imply $x \models x, y$?

Shared Model Complexity: Are there nontrivial strings x and y such that $\mathrm{jsoph}(x, y) \approx \mathrm{soph}(x)+\mathrm{soph}(y)$? What about $\mathrm{scsoph}(x, y) \approx \mathrm{csoph}(x)+\mathrm{csoph}(y)+\min\{\mathrm{csoph}(x), \mathrm{csoph}(y)\}$?

Model Distance: Does $\mathrm{md}(x, y) \approx \mathrm{md}_1(x, y)$? Does $\mathrm{md}_1(x, y) \approx \mathrm{md}_2(x, y)$?

Avenues for Future Research:

Pseudostatistics: We have focused entirely on recursion theoretic properties in this dissertation. We have not discussed the possibility of limiting the generating power or the statistical power of $x \models y$ with resource bounds. This would lend a Complexity Theoretic tone to Algorithmic Statistics. What are the limits of polynomial time-bounded models, for instance? A pseudostatistic can be defined as a statistic that holds the properties of naive statistics for a particular set of

adversaries, rather than for all adversaries (see the discussion in [9] regarding lossy compression). This seems to be related to pseudorandom number generators and derandomization.

Compression and Algorithmic Model Logic: One way to approximate Kolmogorov complexity is to take an off-the-shelf compressor and measure the length of the compressed string. Can we perform such approximations in Algorithmic Model Logic? One possibility is to replace K with C in the definition of $\langle \phi \to \psi \rangle$, where $\mathrm{C}(x)$ is the compressed length of x. It is not so clear what should be done to replace $\langle \mathrm{M}\, x \rangle$, however. What properties does the resulting logic have? Can we build compressors with nice logical properties on purpose?

Monotone Modal Logics: How is Algorithmic Model Logic related to other monotone modal logics? We have limited our discussion of the (meta)logical properties of AML in order to focus on Algorithmic Statistics, but it would be exciting to show that AML is equivalent to an existing monotone modal logic, for instance.

Joint and shared structure: The $\beta_A(\alpha)$ and $\beta'_A(\alpha)$ structure functions from Section 5.2 are a small part of the picture when it comes to potential structure functions for shared models. Do these functions have corresponding $h_A(\alpha)$ and $h'_A(\alpha)$ structure functions? What properties do joint and shared structure functions have?

Model distance: Bennett et al [3] used joint Kolmogorov complexity and conditional Kolmogorov complexity to define a metric between strings based on their information content. Can joint sophistication and the complexity of model maps be used to define a metric on strings based on their structure?

Bibliography

[1] Rajeev Alur, Thomas A. Henzinger, and Orna Kupferman. Alternating-time temporal logic. *J. ACM*, 49(5):672–713, 2002.

[2] Luís Antunes and Lance Fortnow. Sophistication revisited. *Theory of Computing Systems*, 45(1):150–161, 2009.

[3] Charles H Bennett, Péter Gács, Ming Li, Paul MB Vitányi, and Wojciech H Zurek. Information distance. *Information Theory, IEEE Transactions on*, 44(4):1407–1423, 1998.

[4] Péter Gács. On the symmetry of algorithmic information. *Soviet Mathematics Doklady*, 15:1477–1480, 1974.

[5] Helle Hvid Hansen. *Monotonic modal logics*. Institute for Logic, Language and Computation (ILLC), University of Amsterdam, 2003.

[6] Henry E. Kyburg Jr. and Choh-Man Teng. The logic of risky knowledge. *Electr. Notes Theor. Comput. Sci.*, 67:254–262, 2002.

[7] Moshe Koppel. Structure. In Rolf Herken, editor, *The Universal Turing Machine: a Half-Century Survey*, pages 435–452. Oxford University Press, 1988.

[8] Ming Li and Paul Vitányi. *An introduction to Kolmogorov complexity and its applications*. Springer-Verlag, 3rd edition, 2008.

[9] Francisco Mota, Scott Aaronson, Luis Filipe Coelho Antunes, and Andre Souto. Sophistication as randomness deficiency. In Helmut Jürgensen and Rogério Reis, editors, *DCFS*, volume 8031 of *Lecture Notes in Computer Science*, pages 172–181. Springer, 2013.

[10] David Salomon. *Data compression: The Complete Reference*. Springer-Verlag, 4th edition, 2007.

[11] Alexander Shen and Nikolai K. Vereshchagin. Logical operations and kolmogorov complexity. *Theor. Comput. Sci.*, 271(1-2):125–129, 2002.

[12] Nikolay Vereshchagin and Paul Vitányi. Kolmogorov's structure functions and model selection. *IEEE Transactions on Information Theory*, 50(12):3265–3290, 2004.

[13] Nikolay K. Vereshchagin. Algorithmic minimal sufficient statistics: A new definition. *Electronic Colloquium on Computational Complexity (ECCC)*, 17:90, 2010.

[14] Nikolay K. Vereshchagin. On algorithmic strong sufficient statistics. In Paola Bonizzoni, Vasco Brattka, and Benedikt Löwe, editors, *CiE*, volume 7921 of *Lecture Notes in Computer Science*, pages 424–433. Springer, 2013.