ALGORITHMIC INFORMATION THEORY: REVIEW FOR PHYSICISTS AND NATURAL SCIENTISTS

S D DEVINE
Victoria Management School, Victoria University of Wellington, PO Box 600, Wellington, 6140,New Zealand,
sean.devine@vuw.ac.nz

May 30, 2014

Preface

I was originally a research physicist in a government laboratory, with a modest publication record. On the way to becoming a manager in the New Zealand science system I became an economist inquiring into wealth creation through technological change. In doing this, I saw serious difficulties with the neoclassical equilibrium view of economics, as the view failed to recognise that an economy is far from equilibrium. In an equilibrium situation the development path is irrelevant. However the different paths and outcomes that occurred when the Soviet Union and China moved towards market economics, show path is critical. Again, The 2007-2008 shock to the US and the world economy shows that the macroeconomic drivers have little to do with a neoclassical equilibrium view.

In looking for a way to approach non equilibrium issues, I stumbled across Chaitin's work on Algorithmic Information Theory (AIT) outlined in the book John Casti's book "Complexification". This introduced me to the works of Gregory Chaitin and ultimately Li and Vitányi's [76] comprehensive book. While this book is a treasure trove, it is difficult for an outsider to master as the context of much of the development is not clear to a non mathematician. Nevertheless the book pointed to Kolmogorov's work on algorithmic complexity, probability theory and randomness. So while the application to economic systems is on the back burner, I realised that AIT provided a useful tool for scientists to look at natural systems.

In short, the developments in AIT, probability and randomness seemed to me to provide a valuable tool to apply to the kind of organised complex systems studied in the natural world. Unfortunately, these insights are not readily

accessible to the scientific community, or in a form that is easy to apply. I found that the earlier work of Zurek and Bennett on algorithmic entropy provided rich understandings of non equilibrium systems- yet this work seems to have become lost in the mists of time. Hence this book. The hope is that readers will be able to absorb the key ideas behind AIT so that they are in a better position to access the mathematical developments and to apply the ideas to their own areas of interest.

Sean Devine

Contents

Contents **2**

1 Introduction **5**
- 1.1 Some approaches to complexity 6
- 1.2 Algorithmic information theory (AIT) 7
- Algorithmic Information Theory and Entropy 8
- Problems with AIT 9
- 1.3 Algorithmic Information Theory and mathematics 11
- 1.4 Real world systems 11

2 Computation and Algorithmic Information Theory **15**
- 2.1 The computational requirements for workable algorithms 15
- 2.2 The Turing machine 16
- The Universal Turing Machine 18
- Alternative Turing Machines 18
- Noncomputable Functions 19
- 2.3 Measure Theory 20
- Cylindrical sets 22

3 AIT and algorithmic complexity **25**
- 3.1 Machine dependence and the Invariance Theorem 27
- Issues with AIT 29
- 3.2 Self Delimiting Coding and the Kraft inequality 31
- 3.3 Optimum coding and Shannon's noiseless coding theorem . . . 34
- Delimiting coding of a natural number 35
- The relationship with entropy, conditional entropy etc. 36
- 3.4 Entropy relative to the common framework 39
- 3.5 Entropy and probability 40

3.6 The fountain of all knowledge: Chaitin's Omega 41
3.7 Gödel's theorem and Formal Axiomatic Systems 42
3.8 The algorithmic coding theorem, universal semimeasure, priors and inference 45
Inference and Bayesian statistics and the universal distribution 47

4 The algorithmic entropy of strings exhibiting order but with variation or noise 51
4.1 Provisional entropy 52
A simple example 55
4.2 Algorithmic Minimum Statistics Approach and the provisional entropy 56
4.3 The provisional entropy and the true algorithmic entropy . . . 58
4.4 The specification in terms of a probability distribution 60
How to specify noisy data 62
Model examples 62
Model selection 64

5 Order and entropy 67
5.1 The meaning of order 67
5.2 Algorithmic Entropy and conventional Entropy 69
Missing Information and the algorithmic description 74

6 Reversibility and computations 79
Implications of an irreversible computation that simulates a reversible real world computation 80
6.1 Implications for Algorithmic entropy and the second law 83
6.2 The cost of restoring a system to a previous state 86
6.3 The algorithmic equivalent of the second law of thermodynamics 88

7 The minimum description level approach 93

8 The non-typical string and randomness 101
8.1 Outline on perspectives on randomness 101
8.2 Martin-Löf test of randomness 105

9 How replication processes maintain a system far from equilibrium 113
9.1 Introduction to maintaining a system distant from equilibrium 113
Three thought experiments to clarify entropy flows in natural systems 116
System processes and system degradation 118
9.2 Replication processes to counter the second law of thermodynamics by generating order 119
9.3 Simple illustrative examples of replication 121
Replicating spins 122

Coherent photons as replicators 124
9.4 The entropy cost of replication with variations 126
9.5 Entropy balances in a reversible system 129
9.6 Homeostasis and second law evolution 131
Information requirements of homeostasis 131
Natural ordering through replication processes 135
Replicating algorithms 138
9.7 Selection processes to maintain a system in a viable configuration 141
Adaptation and interdependence of replicas in an open system 145
Nested systems and degree of organisation 146
9.8 Summary of system regulation and AIT 149

10 AIT and Philosophical issues **153**
10.1 Algorithmic Descriptions, Learning and Artificial Intelligence . 153
10.2 The mathematical implications of Algorithmic Information Theory 154
10.3 How can we understand the universe? 155

Appendix A Exact cost of disturbance **161**

Bibliography **163**

Abstract

Chapter 1

Introduction

In recent years there has been increased interest in complexity and organised complex systems whether these be physical, biological or societal systems (see for example Casti [23]). However there is not always a consistent understanding of the meaning of the word complexity. Those approaching complexity from a mathematical viewpoint identify complexity with randomness- i.e. the most complex structures are those embodying the least redundancy of information, or the greatest entropy. Others from a physical, biological, or social sciences perspective use the phrase "complex system" to identify systems that show order or structure, but cannot be simply described. These systems do not have a relatively high entropy and, because they are far-from-equilibrium, they are anything but random. As this book is written from the perspective of the natural sciences, the phrase "organised systems" or "organised complex system" will generally be used to apply to ordered systems that are non-random and are far-from-equilibrium; i.e. those that natural scientists might call complex. However, as at times these organised systems will also be described as "complex" or exhibiting "complexity", a distinction needs to be made with the situation where "complex" refers to randomness in the mathematical sense. Hopefully the context will make the meaning clear so that no ambiguity arises.

A critical question is whether one should inquire into a complex organised system from the "bottom up so to speak, where understandings at the most fundamental level arise. However, while a bottom up understanding may provide insights into how organised systems can be constructed from simpler systems, at some level of complexity the bottom up process of sense making becomes too involved- there is just too much detail. If deeper understandings are to emerge, high level approaches are needed to inquire into the system from the "top down". The bottom up approach uses the simplest pattern-making structures such as a cellular automaton (see Wolfram [107]) or a Turing machine to see if understandings at the fundamental level throw insights into high level complex or ordered systems. Moving up to a higher level of detail, a few steps from the bottom, these pattern making structures may be interpreted as adaptive agents responding to an external environment (e.g. Crutchfield [39]).

In practice, the observed complex behaviour at the higher level may be said to be emergent because it cannot be understood in simple terms. For example, one can discuss the adaptive responses of a life form in a biosystem in these higher level terms. Understandings at these levels are primarily phenomenological often seeing the whole as more than "the sum of the parts" as this is the best that can be done with the way humans make sense with limited information.

However the argument here is that Algorithmic Information Theory can suggest ways to "sum the parts" in order to provide insights into the principles behind the phenomenological approach. The approach of Algorithmic Information Theory (AIT) (see for example Li and Vitányi [76] and Chaitin [33]) provides a thread that can link fundamental descriptions of simple systems to understandings of the behaviour of emerging systems and, furthermore, is helpful in providing insights into underlying processes implicit in the more phenomenological descriptions. Where the high level description makes reference to underlying structures; e.g. where nesting occurs, as it does for Stafford Beer's Viable Systems Model [9, 10], Algorithmic Information Theory can provide a bridge between a top down phenomenological approach to structure, and a bottom up, atomistic or microsystem's, approach.

1.1 Some approaches to complexity

Crutchfield and his colleagues have developed computational mechanics ([41, 39]) using the concepts of 'causal states' and statistical complexity to characterise pattern, structure and organisation in discrete-valued, discrete-time stationary stochastic processes. The computational mechanics approach recognises that, where a noisy pattern in a string of characters is observed as a time or spatial sequence, there is the possibility that future members of the sequence can be predicted. The approach [40, 91] shows that one does not need to know the whole past to optimally predict the future of an observed stochastic sequence, but only what causal state the sequence comes from. All members of the set of pasts that generate the same stochastic future belong to the same causal state. Measures associated with the causal state approach include the statistical complexity, which is the Shannon entropy of the set of causal states, the excess entropy E and the entropy rate h - the entropy per symbol. The approach has been extended to spatial sequences using the example of one dimensional spin chains [52].

Wolfram [107] has shown that simple cellular automaton can produce rich patterns. He suggests that the universe itself is explainable in terms of simple computational systems. Bennett [13] has suggested that "logical depth"; a measure of how many steps are required for a computational system to produce the desired pattern or structure, is a measure of the system's complexity.

Each of these bottom up approaches offers insights into organised systems. However the purpose of this book is to focus on how Algorithmic Information Theory (AIT), the measure of the complexity of a structure or object by

its shortest description, can provide a comprehensive understanding of organised systems. The development of an AIT approach that allows for noise or variation in pattern (Devine [46] and Vereshchagin and Vitányi [103]) is not inconsistent with the causal state approach of Crutchfield and his colleagues. Wolfram's approach (Wolfram [107] last chapter) also can be interpreted in an AIT framework, while Bennett's logical depth is a measure of the time or number of steps required to complete a computational process. The following section outlines the key points of AIT.

1.2 Algorithmic information theory (AIT)

AIT has emerged to offer insights into several different mathematical and physical problems. Solomonoff [93] was interested in inductive reasoning and artificial intelligence. Solomonoff argued that to send the result of a series of random events (such as the outcome of a series of football result) every result would need to be transmitted as a string of numbers. On the other hand, the first 100 digits of π could be transmitted more efficiently by transmitting the algorithm that generates 100 digits of π. The Soviet mathematician Kolmogorov [69] developed the theory as part of his approach to probability, randomness and mutual information. Gregory Chaitin [26] independently developed the theory to describe pattern in strings and then moved on to inquire into incompleteness in formal systems and randomness in mathematics. Significant developments took place within the Kolmogorov school but these did not become readily available to those in the West until researchers from the school moved to Europe and the US.

While the next chapter will describe the theory in more depth, the core of AIT is that where a structure can be represented by a string of symbols, the shortest algorithm that is able to generate the string is a measure of what mathematicians call the "complexity" of the string and therefore the structure. (Here, to avoid ambiguity, the phrase "algorithmic complexity" will be used to denote this concept.) As the length of any algorithm will depend on the computational system used, such a measure would appear to be of little use. Fortunately, the length of the set of instructions that generates the string is not particularly computer dependent (see section 3.1). As a consequence, a structure represented by a patterned string can be described by an algorithm much shorter than the string itself. On the other hand, a string that is a random sequence of symbols represents a system that has no discernible organisation. Such a string can only be described by an algorithm longer than the length of the string. For example a string made up of a thousand repeats of '01' has two thousand characters and is of the form $s = 010101...0101$. This string can be described by the algorithm

$$PRINT\ 01,\ 1000\ times.$$

The algorithm is relatively short. Its size will be dominated by the code for '1000' which, for a binary algorithm, is close to $log_2 1000$. On the other hand a

random binary string, of the same length $s = 10011...0111$ can only be produced by specifying each symbol; i.e. $PRINT$ 10011...0111. The instruction set will be at least 2,000 characters long, plus some extra instructions to specify the print statement.

In physical situations, as is discussed later, it is usually convenient to select the code for each computer instruction from a prefix-free set of codes; i.e. no code is a prefix of any other. In this case, the computer does not need any marker to indicate where one instruction or code ends and another begins. Such coding is called "self-delimiting" coding (or for some strange reason "prefix" coding) as the code carries information about the code length. Throughout this work, code words from a "prefix-free set", or code words that are "self-delimiting" will be used to describe these codes. The importance of self-delimiting coding is that the set of instructions obey the Kraft inequality. As is shown later (section 3.2) the extension of the Kraft inequality to an infinite sets of codes can be used to ensure programmes themselves are self-delimiting. This then

- allows algorithms and subroutines to be concatenated (joined), with little increase in overall length;
- provides a connection between the length of an algorithm describing a string and the probability of that algorithm in the set of all algorithms – thus tying AIT more closely to information theory; and
- allows a universal distribution to be defined and by so doing provides insights into betting payoffs and inductive reasoning;

Algorithmic Information Theory and Entropy

The algorithmic entropy of a string is taken to be identical to the algorithmic complexity measure using self-delimiting coding. I.e. the algorithmic entropy of the string equals the length of the shortest programme using self-delimiting coding that generates the string. Once provision is made for the computational overheads, (see sections 4.1 and 5.2) a calculation of the algorithmic entropy for a string representing an equilibrium state of a thermodynamic system is virtually identical to the Shannon entropy of the equilibrium state [92] and, allowing for different units virtually equals the entropy of statistical thermodynamics [101]. However conceptually, the algorithmic entropy is different from the traditional entropies. The algorithmic entropy is a measure of the exact state of a system and therefore has meaning outside of equilibrium and would appear to be more fundamental. Zurek [108] has shown how Algorithmic Information Theory can resolve the graining problem in statistical thermodynamics. Zurek also shows how to specify the state of the N particles in a Boltzmann gas (i.e. a gas where the particles have no internal degrees of freedom) in a container. The position and momentum coordinates of all the particles can be specified in a $6N$ position and momentum phase space. The grain or cellular structure of the phase space is first defined algorithmically. Once this is done,

the particular microstate of the gas, corresponding to an actual instantaneous configuration, can be described by a sequence of 0's and 1's depending on whether a cell in phase space is empty or occupied. The algorithmic entropy of a such a microstate at equilibrium is virtually identical to the Shannon entropy of the system. Zurek has defined the concept of "Physical Entropy" to quantify a system where some information can be described algorithmically and the remaining uncertainty encapsulated in a Shannon entropy term. However it is now clear that a new type of entropy, such as Physical Entropy is unnecessary as an algorithm that is able to generate a given string with structure but also with variation of the structore or uncertainty can be found and its length will return the same entropy value as Zurek's Physical Entropy (see next section and section 4) . The limiting values of the physical entropy of Zurek, where all the patterned information is captured, can be identified as the true algorithmic entropy of a noisy or variable patterned string.

Problems with AIT

There are, as Crutchfield and his colleagues Feldman and Shalizi [91, 42] point out, a number of apparent difficulties with AIT. Some of these are to do with custom or notation. For example the use of the word "complexity" by mathematicians to measure degree of randomness does not fit in with the intuitive approach of the natural scientist where the most complex systems are neither simple nor random, but rather are those systems that show emergent or non simple behaviour. Provided everyone is consistent (which they are not) there should be little problem. As is mentioned earlier, the approach here is to use the phrase "organised systems" to refer to what a natural scientist might call a "complex system", and to use "algorithmic complexity" or "algorithmic entropy" to mean the measure of randomness used by mathematicians. The remaining two key problems that these authors identify with the AIT approach are:

- The algorithmic entropy is not computable because of the Turing Halting problem; and
- The algorithmic description of a string must be exact; there is no apparent room for noise, as the description of the noise is likely to swamp an algorithmic description of a noisy patterned string.

However the so called halting problem is a fact rather than a problem. Any measure of complexity that avoids the halting problem inevitably sacrifices exactness. Nevertheless, when pattern is discovered in a string by whatever means, it shows a shorter algorithm exists. The second bullet point above is more critical. The author found that AIT could not be applied to even the simplest natural system because of the assumed inability to deal with noise or variation. However it has since become clear that AIT is able describe noisy strings effectively (Devine [46] and Vereshchagin and Vitányi [103]). In essence, the algorithmic entropy of a string exhibiting a noisy pattern or variation of a

basic pattern, is defined by a two part algorithm. The first specifies the set of patterned strings and the second identifies a particular string within the set. The second stage contribution turns out to have a value virtually identical to the Shannon entropy of the patterned set; i.e. the uncertainty inherent in set of outcomes.

As the algorithmic description of the pattern using this approach may not be the shortest possible, the algorithmic complexity or the algorithmic entropy is not defined in an absolute sense. Rather it is a measure of the upper limit to the algorithmic entropy. Here this upper limit will be called the "provisional" entropy, with the understanding the true entropy may be less (see section 4). Previously Devine [46] used the word "revealed" entropy. However "provisional" may be better as it ties in with Zurek's "physical entropy' and the Algorithmic Minimum Sufficient Statistic measure of a particular string section outlined in 4.2. These different measures, and that of Gács, ([58, 59] which extends the idea to infinite strings are all essentially the same. They provide an entropy measure of a noisy string based on the available information or observed pattern or structure.

An approach based on the Minimum Description Length (MDL) (see section 7) also gives the same insights into describing a noisy or a variable patterned string. The ideal MDL approach models a system by finding the minimum description length; i.e. the shortest model that fits the data. This description contains a description of the hypothesis or model (equivalent to the description of the set) together with a code that identifies the observed data. Given the model, the particular string x_i is identified by a code based on the probability p_i of the occurrence of the string. If two vertical lines are used to denote the length of the string between the lines, the length of this code, i.e. $|code(x_i)| = -log_2(p_i)$. The expectation value of $|code(x_i)|$ is the Shannon entropy of the set of possible strings consistent with the model. However while the MDL approach compresses the data it does not actually fulfill the requirement of AIT that the algorithm must generate the string. The MDL approach only explains it. The full algorithmic description must take the MDL description and combine this with an O(1) decoding routine to actually generate a specific string. All these approaches reinforce the point that the algorithmic entropy includes the length of the description of the hypothesis or model, and a further contribution that is virtually the same as the Shannon entropy measure of the uncertainty.

The above arguments show that the algorithmic entropy of a noisy or variable string with a pattern can be effectively described by combining two algorithms. One specifies the structure or pattern characteristic of the set of all similar strings, while the other identifies the particular string within the set. In effect this second string corresponds to the Shannon entropy or the uncertainty in the set of similar strings. Where an individual string shows no particular structure the string is typical and the algorithmic description or provisional entropy returns a value the same as the Shannon entropy for the set of all such strings. All members of the set of similar patterned strings have the same provisional entropy. However a few non-typical or non-representatives strings

may exhibit further pattern or structure. The algorithmic entropy for these strings will be less that that of a typical member of the set of similar strings.

1.3 Algorithmic Information Theory and mathematics

Most of the developments of AIT have taken place within the discipline of mathematics to address problems related to randomness, formal systems and issues that would appear to have little connection with the natural sciences. However, because these theorems are cast in mathematical terms, they are not readily available to the typical scientist.

A key outcome of Algorithmic Information Theory is its application to formal axiomatic systems. The assumptions and rules of inference of a formal system can be described by a finite string. The theorems of the formal system specified by an appropriate string, can be recursively enumerated through a programme that implements the axioms and rules of inference. Chaitin [30] has shown that a theorem is compressible to the axioms, the rules of inference, and the set of instruction that prove the theorem. The algorithmic complexity of a theorem is the length of the algorithm that generates the theorem thereby showing it is provable.

Chaitin [30] has also shown that some possible theorems or propositions cannot be compressed, i.e. cannot be enunciated in terms of a simpler set of axioms and rules. As these propositions are represented by a string that cannot be compressed they are undecidable. They may be true or false. This is of course Gödel's theorem, but with the twist that unprovable theorems are everywhere. As there are only $2^n - 1$ strings shorter than a string of length n, most strings cannot be compressed and therefore the associated theorems are undecidable. While these can be made decidable by including them as new axioms in a more complex formal system (in the sense of having a longer description), the number of random strings corresponding to unprovable statements, grows faster than the provable ones. As Chaitin [32] put it: "if one has ten pounds of axioms and a twenty-pound theorem, then that theorem cannot be derived from those axioms". This leads Chaitin to speculate that mathematics should accept this uncertainty and become experimental [31, 34, 36]. Randomness, Gödel's theorem and Chaitin's view of mathematics are discussed later in section 3.7.

1.4 Real world systems

In the discussion about an algorithmic description of a natural system, a distinction needs to be made between a physical system which emerges due to an ongoing computation based on physical laws, and one that just describes the final state from the point of view of an external observer. Wolfram [107] gives examples of cellular automata that, operating under simple rules, produced highly patterned structures. The rules that generate the structure after a fixed number of steps are more likely to provide a shorter description that an algo-

rithm that just describes the result. Another example is a flock of geese flying in V formation (See New Scientist [2]). This formation can also be described by an algorithm developed by an external observer. However the pattern arises because each goose optimises its position with respect to other geese both to minimise energy use, to maintain an adequate separation from other geese and to allow for sufficient vision to see ahead. The formation that emerges is a consequence of the computational process of each goose. The algorithmic description as an entropy measure will be the shortest of these two computational descriptions. This will either be the computations undertaken by each goose as computing machines, or that observed externally. However, which of these is the minimal description may depend on what degrees of freedom the observer is interested in. It is likely that, where the entropy measure must include all degrees of freedom, the on-going computation, based on physical laws that gives rise to the system is likely to be more compressed than one just derived from observation. Indeed, the universe itself is often assumed to be a computational machine undertaking an on-going computation that steps through all the valid states of the universe. However if we are only interested in a small part of the universe, the algorithmic description can be compressed more succinctly than describing the whole universe.

Algorithmic Information Theory contributes to making sense of real world systems in a number of ways. While the main approach of this book is primarily about emergence, and linking bottom up with top down approaches to understanding systems, reference will be made to these other contributions. AIT has provided a set of tools to address some deep philosophical questions. These might be to do with the conservation of information and entropy, and issues to do with why the universe is as it is. For example Paul Davies in the Fifth Miracle [44] taps into Algorithmic Information Theory. Some of the questions raised and approaches taken will be outlined below. The current book also expounds a view as to why scientific reductionism is such a successful sense-making approach (see section 10.3). In short, it is because much of the observable universe can be described algorithmically in terms of subroutines nested within in subroutines; i.e. the universe appears to be algorithmically simple. The physical realm mapped by a subroutine can be studied at the subroutine level. For example a stone falling under gravity can be described by a simple algorithm because interaction with the rest of the universe, including the observer, can usually be ignored. The subroutine that describes this behaviour has low algorithmic entropy and, as a consequence, it can be analysed by our own very limited cognitive capability. If the univese were not algorithmically simple, the reductionist process would bear little fruit as the computing capability of the human brain, which itself is part of the universe, would be inadequate to make any sense of a physical system (See also Chaitin [35]). In general it would be expected that our brains would be insufficiently algorithmically complex (even allowing for the ability to extend our brain's capability by the linking of human brains, or utilising computational machines) to decide on the truth of falsity of propositions belonging to a more algorithmically complex universe. However, because it seems that much of the universe around us can

be described by algorithmically simple subroutines, reductionism allows us to make sense of this subset of the whole.

The length of the string that describes the total universe is mind boggling long. Clearly, as Calude and Meyerstein [21] point out, sections of the string might appear ordered on a large scale- e.g. the scale of our existence, yet the string could be completely random over the full time and space scale of the universe that embeds this ordered section. In other words the ordering of our universe may be a fluctuation in a region of the much larger random string that specifies a larger universe. While this may be so, it is argued in this work that, where large scale order exists, no matter how it arises, Algorithmic Information Theory can be used to show how the laws of the universe can generate new ordered structures from existing order. This is discussed in detail in section 9.

Nevertheless, there is a fundamental issue; the algorithmic measure of a description requires the algorithm to halt. For example, the following algorithm, which generates the integers one at a time, is very short because it does not halt.

$$\begin{aligned} & N = 0 \\ 1.\ & N = N + 1 \\ & PRINT\ N \\ & GO\ TO\ 1. \end{aligned} \tag{1.1}$$

However the algorithm that specifies a particular integer must halt when that integer is reached. It is much longer. Consider the following algorithm that generates N_1 which for exammple could be the 1000000th integer, is of the form

$$\begin{aligned} FOR\, N = 0\, to\, N_1 \\ N = N + 1 \\ Next\ N \\ PRINT\ N. \end{aligned} \tag{1.2}$$

About $log_2 N$ bits are required to specify N_1. For example if $N_1 = 1000000$ the algorithm is about 20 bits longer than the programme that generates integers in turn without stopping. A similar issue arises with the algorithm that describes the universe. The universe would appear to be more like a non terminating Turing Machine (at least so far). The patterns we observe are for ever changing. However, just as the algorithm that steps through all integers requires a halt instruction to specify one particular integer, so the algorithm that specifies a particular configuration of the universe, or part of it, at an instant of time requires a halt instruction. The specification of the number of steps, either implicitly or explicitly, that forces the algorithm to halt at an instant of time is extremely large compared with the algorithm that just cycles through all

the universe's states without stopping. As is discussed in section 6.3, the value of the integer specifying the halt state is the dominant contributor to the algorithmic entropy for a particular configuration of the universe that is distant from its initial state, but which has not reached equilibrium.

Chapter 2

Computation and Algorithmic Information Theory

2.1 The computational requirements for workable algorithms

Algorithmic Information Theory defines the order or structure of an object by the length of the algorithm that generates the string defining the structure. This length is known as the Kolmogorov complexity, the algorithmic complexity or the programme sized complexity of the string. However, there are two types of codes that can be used for the instructions and algorithm. The first is where end markers are required for each coded instruction to tell the computer when one instruction finishes and the next starts. As is shown later, this coding gives rise to the plain algorithmic complexity. Alternatively, when no code is allowed to be a prefix of another, the codes can be read instantaneously requiring no end makers. This requires the codes to be restricted to set of instructions that are self-delimiting or come from a prefix-free set (Levin, 1974, Chaitin, 1975). The length of the algorithmic description, using self-delimiting coding will be termed the algorithmic entropy as is it turns out to be an entropy measure and, for physical situations this is preferred over plain algorithmic complexity. However, clearly the kind of algorithm and its length depends on the type of computer used. Fortunately, for two reasons, it turns out that the specifics of the computer is usually not of great significance as a measure of complexity. The first is that an algorithm written for a Universal Turing Machine (i.e. one that is sophisticated enough to simulate any another computing machine), can be related to any other computing machine, once the simulation instructions are known. The length of the simulation algorithm just adds a string of fixed length to the algorithm on the given universal machine. The second is that in science, only the entropy difference is the relevant physical measure because entropy is a function of state. Where entropy differences are measured using the same computer, any simulation constant cancels. Entropy differences are therefore machine independent. Also, common algorithmic instructions such as

those referring to physical laws, or describing common physical situations, such as the resolution of the phase space also cancel. In which case, the algorithmic entropy becomes closely aligned to the Shannon entropy. The following sections deal with the theory of the computing devices known as a Turing Machine (TM). For those who want to skip the detail, the take home messages are:

- A Turing Machine (TM) is a standard procedure for generating any computable function (see 2.2). Where a set of instructions allows one to unambiguously compute a particular function from a given input, such instructions can be written for a TM or for any computing device that is equivalent to a TM. Each TM is equivalent to a function. For example, given an integer n there exists a set of instructions that can turn n as an input into the function n^2 as an output. I.e. the TM that implements this set of instructions corresponds to the function n^2.

- There are many different computational processes and devices that are equivalent to a Turing Machine.

- A Turing Machine may only halt on some inputs. If the machine does not halt, no output is generated. Because a Turing Machine can be devised to check whether a mathematical theorem is true or not, the failure to halt might mean the computation needs to run longer or that it might never halt. In which case the theorem cannot be proven to be true or false.

- A Universal Turing Machine (UTM) is one that can simulate any Turing Machine or any other UTM. The typical computer, familiar to scientists, is a UTM (see section 2.2) that is programmed. Often it is desirable to use a simpler UTM and longer programme where the prgramme instructions are transparent. The alternative is to use a sophisticated UTM where much of the computation is embodied in the hardware or the computer's instruction set. The classical universe driven by deterministic physical laws is a reversible UTM. This is why we can use computers and rational laws to map the behaviour of the universe.

Those who wish to can read the next sections, those who might want to come back to this later can jump to the section 3.

2.2 The Turing machine

A Turing Machine (TM) is a simple computational device or, more accurately, a computational procedure that is able to compute any function that is intuitively computable. This is captured in the Church-Turing thesis - any effective computational procedure can be represented by a Turing Machine. Generally a Turing machine is considered to be deterministic as there is a well-defined output for each step, otherwise the machine halts. The TM may be described as follows (see Casti [23] for a worked example):

- It has an arbitrary long tape (extending in one direction) divided into cells. Each cell contains a symbol from a finite alphabet. One of these symbols corresponds to a blank. The finite number of non-blank cells on the tape represent the input data.
- The TM has a read/write head that can read a symbol, move one cell left or right, and write a symbol. At the beginning of each computational step the read/write head is poised to read the current cell.
- The machine is in one of a finite set of states which determine the action of the machine at each step.
- Given the symbol read, and the state of the machine, a transition table shows what is to be written in the cell, what the new state of the machine is, and whether the head moves right or left. I.e. the transition from the old configuration to the new configuration can be written as a quin tuples of symbols.

(current state, current symbol read) $\rightarrow$ (new symbol written, move left/right, new state).

(Readers should be aware that some definitions express these moves as a quad tuple by restricting the machine's ability to move and write in the same step.) The transition table is in effect the programme that determines what calculation the machine undertakes. A physical machine is unnecessary, as one can undertake a Turing computation on an Excel spreadsheet, or by following instructions using a piece of paper. It is the instructions, not the structure that embodies the machine.

The so called machine starts in an initial configuration and during the computation process steps through the different configurations reading and writing on the tape at each step as determined by the transition table. If there is no effective instruction for the current configuration to implement, the machine halts and the symbols remaining on the tape become the computational output. A Turing Machine uses two or more symbols. Where a binary alphabet is used, symbols need to be coded in binary form. A simple, but inefficient, form of coding for the natural numbers is to represent the number n by a zero and $n + 1$ ones followed by another zero, i.e. $01^{n+1}0$. The number zero can then be represented a 010 and 5 by 01111110. In a sense each Turing Machine is equivalent to just a single programme and, in modern terms, the tape serves as the memory of the machine. There are equivalent variations of the simple Turing Machine. The two tape machine outlined by Chaitin [27] is more straight forward representation of the computational processes underpinning the measurement of algorithmic complexity. This machine has both a work tape and a programme tape. The work tape initially contains the input string y. The computation can be represented by $T(p, y) = x$ which implies, given input y, and programme p, the machine T halts to output x.

The two tape machine is more efficient in a time sense and, because most of the instructions are on the programme tape, the machine is more like a conventional computer as the programme controlling the output is external to the computational system.

The Universal Turing Machine

Each possible Turing Machine can be viewed either as machine that implements a specific algorithm, or as an algorithmic process to enumerate a function. A Universal Turing Machine (UTM) is one that can simulate any other Turing Machine by including the simulation instructions as a programme input. As has been mentioned, a TM is defined by its set of quin tuples (or quad tuples). These map the before and after states of the transition table. A UTM is a machine that, when given the quin tuples that represents the transition table, can simulate the Turing Machine. It becomes possible to define the string of a quin tuple that specifies a transition table for a particular Turing Machine by a unique number known as its Gödel number. This string representing the quin tuple can be coded using self-delimiting coding (see section 3.2). A simple routine can generate all possible quin tuples by stepping through all strings in lexicographical order, rejecting those that do not represent valid transitions. This is similar to a routine that steps through all the natural numbers selecting, say the even ones. The ith set of quin tuplets is given the Gödel number i. The set of all Turing machines is countable, i.e. there is a one-to-one correspondence with the integers. Once the Gödel number i for the TM denoted by T_i is given, its transition table can be generated by a decoding routine. Let e be this short decoding routine. The UTM denoted by U can simulate $T_i(p)$ by prefixing a simulation programme to p. I.e. $U(e1^i0p) = T_i(p) = x$. The routine e counts the 1's until the 0 is reached. This gives i for the UTM to generate the ith transition table from the compressed coded version of the table. As the transition table encapsulates the ith machine, the UTM can then simulate the ith machine.

While there are an infinite number of possible Universal Turing Machines, Marvin Minsky has described a 7-state 4-symbol UTM [81] and Stephen Wolfram has described a 2 state 3 colour Turing Machine that has recently been shown to be universal by Alex Smith [3].

Alternative Turing Machines

There are a number of alternative computing procedures or processes that are equivalent to a Turing Machine or are extensions of Turing Machines. These are briefly outlined below for completeness.

1. Register Machines. The register machine is Turing equivalent and gets its name from its one or more "registers" which replaces a Turing machine's tapes and head. Each register holds a single positive integer. The registers are manipulated by simple instructions like increment and

decrement. An instruction that, on a simple condition, jumps to another instruction provides programming flexibility

2. Tag System. A tag system may also be viewed as an abstract machine that is equivalent to a Turing Machine. It is sometimes called a Post tag machine as it was proposed by Emil Leon Post in 1943. It is a finite state machine with one tape of unbounded length. The tape symbols form a first in first out queue, such that at each step the machine reads the symbol at the head of the queue, deletes a fixed number of symbols and adds symbols. One can construct a tag machine to simulate any Turing Machine.

3. Lambda calculus, developed by Church and Kleene in the 1930s is the smallest universal programming language. It is a formal system specifically designed to investigate function definition, function application, and recursion. While it is equivalent to a TM, its focus is on the transformation rules rather than computational steps.

4. A reversible Turing machine. A Turing Machine can be made reversible where information about past states is either stored, or its instructions themselves are reversible. As the laws of physics are reversible they can only be simulated adequately by a reversible TM. The physical implications of this are discussed in section 6.

5. Probabilistic Turing Machine. Generally a Turing machine is deterministic as there is a well defined output for each step, otherwise the machine halts. But a probabilistic Turing Machine, can be envisaged where, at some points in the computation, the machine can randomly access a 0 or a 1. The output of such a machine can then be stochastic. However, as there is no valid random number generator, only pseudo probabilistic machines would seem to exist. Nevertheless, Casti and Calude [25] have proposed that a quantum number generator be used to provide a true random input. Their article in New Scientist suggests some intriguing applications of a truly probabilistic TM. However, at least at the present time, AIT restricts itself to deterministic computation.

It should also be noted that if quantum computers become possible (e.g. New Scientists [43]), the Strong Church-Turing Thesis would not apply and efficient quantum algorithms could perform tasks in polynomial time.

Noncomputable Functions

Each Turing Machine can be envisaged as a function that maps natural numbers as inputs on to natural numbers as outputs. There are only a countable number of such functions. However, the real numbers are uncountable and as all the functions mapping natural numbers as inputs, to natural numbers as outputs includes the reals these functions are uncountable. It follows that there are

insufficient Turing Machines to compute every function. As a consequence a Turing Machine cannot halt for all inputs. If a Turing Machine halts after a designated finite number of steps the function is computable. However if it does not halt in the designated number of steps it is uncertain whether it will halt if allowed to continue running. Turing showed [102] that, where a computation is formulated as a decision algorithm, the halting problem implies that there are some propositions that are undecidable. The proof of the halting problem relies on a self referential algorithm analogous to the Epimendides' paradox "this statement is false". As a consequence, the halting problem is equivalent to Gödel's incompleteness theorem, and leads to Rice's theorem which shows that any nontrivial question about the behaviour or output of a Turing machine is undecidable.

A partially recursive function, or a partially recursive computation is one that is computable. I.e. the Turing Machine expressing the function halts on some inputs but not all. A Turing computer can therefore be envisaged as a partially recursive function; a procedure that maps some members of an input set of natural numbers on to an output set of natural numbers. On the other hand, a recursive function (or more strictly total recursive function) is defined for all arguments (inputs) and the Turing machine that embodies the procedure halts on all inputs.

A recursively enumerable or computably enumerable set is one for which an algorithmic or decision procedure exists that, when given a natural number or group of numbers, will halt if the number or group is a member of the set. Equivalently if a procedure exists to list every member of the set, it can be established that any string is a member of the set. The set is in effect the domain of a partially recursive function. However, as there is no certainty that the computation will halt, a partially recursive function is semi -decidable; the result of any computation is yes/possibly yes. All finite sets are decidable and recursively enumerable, as are many countable infinite sets, such as the natural numbers.

These concepts should be distinguished from primitive recursive functions which form a subset of the recursive functions. These primitive recursive functions are those able to be generated by a recursive process applied to a few simple functions. Most functions used in science are in fact primitive recursive as they can be implemented by a computation involving a "Do" or "FOR loop that terminates after a specified number of cycles.

2.3 Measure Theory

Measure theory has been developed to provide a framework to establish different measures on sets that may be infinite, such as the set of real numbers $\Re$. Useful measures include, probability, the counting measure (the number of elements in subsets), and the distance measure called the Lebesgue measure. In a multidimensional space the Lebesgue measure becomes the volume measure in that space. The critical feature of measure theory is that it provides the tools

to determine how a measure on separate subsets is related to a the measure for larger set containing the subsets. For example, in the case of the counting measure, one expects the total number of elements in separate subsets that have no elements in common to be the sum of the elements in each subset.

A general measure of a property of a set and its member will be denoted by μ. A measure must have appropriate additive properties as was mentioned above for the counting measure. The measure for all possible events, and arrangements of events, must also be able to be defined. Key to providing a framework for measure theory is the understanding of the Borel sigma algebra or Borel field. This framework is just a series of definitions which provide the rules for manipulating members of a set (e.g. the set of reals), to allow the properties, such as probability, to be defined consistently. The Borel algebra is what is called a sigma algebra that is restricted to a set Ω which is a metric space; i.e. a space, such as the collection of open sets on the reals, where there is a distance function. Keeping this in mind, the collection $\mathcal{B}$ of subsets of a metric space, has the following properties:

- The empty set is in $\mathcal{B}$
- If subset B is in $\mathcal{B}$ then its complement is in $\mathcal{B}$
- For any collection of a countable number of sets in $\mathcal{B}$ their union must also be in $\mathcal{B}$. I.e. if B_1 and B_2 are in $\mathcal{B}$ then $B_1 \cup B_2$ are in $\mathcal{B}$, or for countably infinite members, $\bigcup_i^{\infty} B_i$.

The axioms also imply that the collection of sets is closed under countable intersection and the collection also includes the full set Ω. For infinite reals, the Borel sigma algebra defines the smallest sigma algebra family that contains all the intervals; but it does not include all the subsets Ω. (The set of all subsets is known as the power set.) This restriction avoids the Banach-Tarski paradox. With these definitions of the properties of the subsets of the metric space, the definition of a general measure μ, such as probability or number of members, on the Borel algebra is as follows.

- $\mu(\mathcal{B}) \geq 0$ and for the empty set $\mu(\emptyset) = 0$
- The measure is additive.
- For any collection B_1, B_2, B_3 with union $\mathcal{B}$ of countable disjoint sets, (I.e. $B_1 \cap B_2 = 0$) then $\mu(\mathcal{B}) = \sum_i (B_i)$; noting that $\mathcal{B}$ can be a countable infinite union of sets.

This last bullet point says that if there are no elements in common between the (sub)sets, the measure is just the sum of the measure for each subset.

In the particular case where the measure μ over the whole sample space is $\mu(\Omega) = 1$, the measure is termed a probability and is usually denoted by P. The measure definition of P corresponds to the Kolmogorov definition of

probability used in AIT. However for infinitely many events, the Kolmogorov probability requires the following extra axiom (see [76] p. 18) namely;

- For a decreasing sequence of events $B_1 \supset B_2 \supset B_n$ such that the set of intersection of all such events is empty; $\bigcap_n B_n = 0$, then the probability in the limits is given by $lim_{n\to\infty} P(B_n) = 0$.

The Kolmogorov approach to probability provides a fundamental definition of probability that is more useful than the traditional 'frequentist' approach. However it has some counter intuitive features. When the approach is applied to the number line of reals [0,1], the probability of the set of irrationals is 1 and the probability of one particular irrational is zero, as there are an infinite number of irrationals. It follows that the probability of the set of all rationals is zero. In other words, the Kolmogorov probability of drawing a rational at random from the set of all numbers on the number line is not just virtually zero, it is zero.

Cylindrical sets

Not all numbers are rational. Often in science a measurement , for example one that specifies the position and momentum coordinate of an atom, can be treated as a rational number when it is specified to a certain level of precision. However it is sometimes necessary to develop the mathematical tools to handle the irrational numbers. These are numbers whose decimal or binary expansion is infinite, as they cannot be expressed as a simple fraction in the form of a/b where a and b are integers and b is non zero. The reals or real numbers include both the rational and irrational numbers. Normally one considers the set of reals to be represented by the set of infinite numbers lying on the number line between 0 and 1 as the properties of any real greater than 1 depends on its properties between 0 and 1. Similarly, using binary notation, there are an infinite set of reals starting with the sequence x, i.e. the number on the number line represented by $0.x\ldots\ldots\ldots$. The concept of a cylindrical set provides a convenient notation for dealing with such sequences. The cylindrical set allows sets of reals belonging to the same region of the number line to be specified by their common prefix, which is x in the above example. The cylindrical set is denoted by Γ_x and is the set associated with prefix x just after the decimal point. For example the binary string $0.1011\ldots$, is the one way set of infinite sequences starting with 1011 and is represented by Γ_{1011}. The cylindrical set x represents the reals from $0.x0^\infty$ to $0.x1^\infty$. (i.e. $0.x$ to $0.x111\ldots$ where the ∞ sign refers to a repeated 0 or 1). Just as in decimal notation$0.00009^\infty = 0.0001$, so in binary notation 0.00001^∞ is equal to 0.0001. It can be seen that the infinite set of 1s in the sequence 0.00001^∞ can be replaced by $2^{-|0000|}$ $(= 2^{-4})$. In general $0.x1^\infty$ can be replaced by $0.x + 2^{-|x|}$. (Note here the vertical lines denote the number of digits enclosed.) This allows one to see that Γ_x represents the half open set $[0.x$ to $0.x + 2^{-|x|})$. For those not familiar with the notation,

the square bracket "[" indicates the start point is a member of the set while ")" indicates that the endpoint is not.

For a measure μ based on the set of reals, such as the probability measure, $\mu(\Gamma_x) = \sum_b \mu_b(\Gamma_{xb})$, or in simplified notation $\mu(x) = \sum \mu_b(xb)$. This means that a measure such as the probability on a cylindrical set starting with the prefix x is the sum of the measure of all the strings starting with this prefix. For example the probability measure denoted by P is given by $P(\Gamma_{10}) = P(\Gamma_{1010}) + P(\Gamma_{1011})$. Useful measures include the following.

- *The Lebesgue measure* λ is a measure of distance between reals, and is an invariant under translation. For the real line interval [a,b] this corresponds to our intuitive concept of distance and $\lambda = (b - a)$. The Lebesgue measure for a three dimensional space i.e. $\Re^3$ is formed from the Cartesian product of the one dimensinal space and is a volume measure. The $6N$ dimensional equivalent is the measure used to quantify volumes in phase space in statistical entropy calculations. The Lebesgue measure $\lambda(\Gamma_x)$, is a measure on the cylinder represented by $[0.x$ to $0.x + 2^{-|x|})$. An important point that is used later is that the measure corresponds to $2^{-|x|}$ which is the distance between the ends of the cylinder. For example, the cylinder starting with the empty set $\emptyset$ denoted by $\Gamma_\emptyset$ has the Lebesgue measure $\lambda(\emptyset) = 1$ as this cylinder embraces the whole number line from $.\emptyset0000000\ldots$ to $.\emptyset1111\ldots$, i.e. $.00000\ldots$ to 1.

- *The Kolmogorov probability measure* on the reals [0,1], with cylindrical sets as the subsets, has been mentioned earlier. It is a consistent measure.

- The *counting measure* is straight forward; it is the number of members in a set or subset $\mathcal{E}$, and is denoted by $|\mathcal{E}|$.

The semi measure. A semi measure does not have the simple additive properties of a true measure. It is a sort of defective probability measure as the definition is loosened. The semi measure μ for the set of strings Ω has

$$\mu(\Omega) \leq 1.$$

In the discrete case, this corresponds to $\Sigma_i \mu(s_i) \leq 1$. For a continuous semi measure specified on the real number line between 0 and 1 the measure has different additive properties than a true measure. I.e.

$$\mu(\Gamma_x) \geq \Sigma_b \mu(\Gamma_{xb}).$$

The semimeasure is particularly important in Algorithmic Information Theory as $2^{-H(x)}$, where $H(x)$ is the algorithmic entropy is a semi measure. As section 3.2 shows, $\sum 2^{-H(x)} \leq 1$ because of the Kraft-Chaitin inequality.

This connection means the measure is useful in dealing with the halting probability, and in developing the idea of a universal semi measure to deal with inference as is outlined in section 3.8. The semi measure is computable from below, i.e. one can approximate it from below. This is probably all a bit

heavy going for the non-mathematician. The take home message is that set theory, with the idea of cylindrical sets to represent all strings with the same initial sequence, allows a number of different measures to be defined in a way that can cope with infinite sets.

Chapter 3

AIT and algorithmic complexity

As was pointed out in section 1.2, Algorithmic Information Theory (AIT) was originally conceived by Ray Solomonoff [93]. The approach was formalised by Andrey Kolmogorov [69] and independently by Gregory Chaitin [26]. These two authors showed how the machine dependence of the algorithmic complexity measure could be mostly eliminated. Later the approach was extended to self-delimiting coding of computer instructions by Leonard Levin [74, 75] and also Gregory Chaitin [27]. Algorithmic information theory (AIT) is based on the idea that a highly ordered structure with recognisable pattern can be more simply described than one that has no recognisable features. For example the sequence representing π can be generated by a relatively short computer programme. This allows the AIT approach to provide a formal tool to identify the pattern or order of the structure represented by a string. In this approach the algorithmic complexity of a string "s" is defined as length of the shortest algorithm p^* of length $|p^*|$ that is able to generate that string. Here $||$ denotes the length in bits of the programme between the vertical lines. The length of the shortest algorithm or programme that generates a string is also known as the Kolmogorov complexity or the programme sized complexity and furthermore, it is also a measure of the information contained in the string. A short programme means the string s is ordered as it is simple to describe. This contrasts to a programme that is as long as the string itself and clearly shows no order. The importance of AIT for the natural sciences is that a physical or biological system can always be converted to a string of digits that represents an instantaneous configuration of the system in an appropriate state space. As an example, consider how one might represent the instantaneous configuration of N players on a sports field. If jumping is to be allowed, three position and three velocity dimensions are required to specify the coordinates for each player. With N players, the instantaneous configuration is a binary string in what would be called the 6N dimensional state space of the system. If the configuration is ordered, as would be the case if all the players were running in line, the description would be simple than the situation where all players were placed at random and running at different speeds and directions. An actual

configuration in the state space of a collection of particles will specify the position of the particle, its momentum or the equivalent, and the electronic and other states. If the string that describes a particular structure shows order, features or pattern, a short algorithmic description can be found to generate the string.

In practice strings and algorithms are usually represented in binary form. A simple example might be that of a magnetic material such as the naturally occurring loadstone.

The magnetic state of each iron atom can be specified by the direction of its magnetic moment or spin. When the spin is vertically aligned the direction can be specified by a 1, and a 0 when the spin is pointing in the opposite direction. The instantaneous configuration of the magnetic system can be specified by a string of 0's anbd 1's representing the alignment of each spin. Above, what is called the Curie transition temperature, all the spins will be randomly aligned and there will be no net magnetism. In which case, the configuration at instant of time can be specified by a random sequence of 0's and 1's. However where the spins become aligned below the Curie temperature, the resultant highly ordered configuration is represented by a sequence of 1's and, as shown below, can be described by a short algorithm.

Consider the following two sequences, which might represent the configuraton of physical system. The first is a string of N repeated 1's and the second is string N random 0's and 1's. These could be considered as the result of tossing a coin N times where heads is denoted by a 1 and tails by a 0. In the first case, the coin has obviously been tampered with, while in the second case the coin would most likely be a fair coin. Alternatively, these two outcomes could be considered to represent the orientation of spins in a magnetic system as outlined in the previous paragraph. The

1. The first case is where the outcome is ordered, and is represented by a string of N 1's in a row, i.e. $s =$ "11111 . . . ". As this is highly ordered it can be described by a simple algorithm p' of the form;

$$\begin{aligned} &FOR\, I = 1\, to\, N \\ &\quad PRINT\ \text{“1”} \\ &NEXT\, I. \end{aligned} \tag{3.1}$$

   In this case the algorithm only needs to code the number N, the character printed, together with a loop instruction that repeats the print command. In what follows, the notation $|\ldots\ldots|$ is used to denote the length of the binary string representing the characters or computational instructions between the vertical lines. In which case the length of the algorithm p' is:

$$|p'| = |N| + |1| + |PRINT| + |loop\, instruction|. \tag{3.2}$$

   The integer N is usually specified by the length of its lexicographic representation. This discussed in more detail in see section 3.3). In the

lexicographic representation, $|N| = \lfloor log_2(N+1) \rfloor$. The floor function around the log term represents the greatest integer $\leq log_2(N+1)$. In what follows $l\hat{o}gN$, without the subscript, will be used to denote this integer - the length of the lexicographic representation. If N is large, the algorithm will be dominated by the length of N which is close to $log_2 N$ bits. If p' is a binary algorithm that generates s on a computer C, then $|p^*|$, the size of the shortest algorithmic description p^* that generates s must be $\leq |p'|$.

2. In this case the string is a random sequence of 0's and 1's represented by N characters of the form "$110010 \ldots 1100$". Because the sequence is random it can only be generated by a binary algorithm that specifies each character individually. I.e.

$$p' = PRINT \text{ "}s\text{"}.$$

If p^* is the shortest algorithm to do this then, then its length is

$$|p^*| \approx |p'| = |s| + |PRINT|$$

As the length of this algorithm must include the length of the sequence, the length of the $PRINT$ instruction, it must be somewhat greater than the sequence length.

In the AIT framework, the most complex strings are those that show no pattern and cannot be described by a shorter algorithms. However, as is outlined below, to be consistent, standard procedures are required to code the algorithms that specify the structure, and to minimize the computer dependence of the algorithm. There are two types of codes that can be used for the computational instructions. The first is where end markers are required for each coded instruction to tell the computer when one instruction finishes and the next starts. This coding gives rise to the plain algorithmic complexity denoted by $C(s)$. Alternatively, when no code is a prefix of another, the codes can be read instantaneously requiring no end makers. This requires the codes to be restricted to set of instructions that are self-delimiting or come from a prefix-free set (Levin, 1974, Chaitin, 1975). The algorithmic complexity using this coding will be denoted by $H(s)$ and, because it is an entropy measure of string s and the configuration s represents, it will be termed the algorithmic entropy. To sum up, any natural structure that shows order can be described by a short algorithm compared with a structure that shows no order and which can only be described by specifying each character.

3.1 Machine dependence and the Invariance Theorem

Nothing has been said about the computational device that implements the programme p^*. Fortunately, as is shown below, this machine dependence can be minimised. Let p be a programme that generates s on a UTM denoted by

U, i.e. $U(p) = s$. The algorithmic complexity measure $C_U(s)$ on Machine U is given by the shortest programme p that does this.

$$C_U(s) = min_{U(p)=s} |p|. \tag{3.3}$$

When the UTM generates s with input string y, i.e. $U(p, y) = s$ the measure becomes,

$$C_U(s|y) = min_{U(p,y)=s} |p|. \tag{3.4}$$

As any UTM can simulate the computations made on any other, if another UTM W, implements a similar programme to generate s then

$$C_U(s) \leq C_W(s) + c, \tag{3.5}$$

where the constant c allows for the length of an extra term needed to instruct machine W to simulate machine U. The constant c is of the order of 1 as it is independent of the string gnerated. I.e. $c = O(1)$. In practice algorithmic complexity can be evaluated on a standard reference UTM. The $O(1)$ term can often be ignored as in most physical situations only differences are important. The measure of algorithmic complexity, the shortest programme to generate string s must therefore be

$$C(s) \leq C_U(s) + O(1). \tag{3.6}$$

When string y is given as an input to compute s, i.e. $U(p, y) = s$ the invariance theorem states:

$$C(s|y) \leq C_U(s|y) + O(1). \tag{3.7}$$

The theorem is intuitively reasonable as a UTM can be built on a few simple instructions and an interpreter can simulate one computer on another. The proof of the theorem can be made robust as outlined in section 2.2. The section shows that a UTM U can simulate the ith Turing Machine T_i. Let $T_i(p^*) = s$. Let machine U read a string e, the short programme that extracts i from the string 1^i0 and then steps through all the transition tables of each Turing Machine to find the transition table for the ith one. With this information U can simulate p^* on the ith machine to generate s. I.e. $U(e1^i0p^*) = T_i(p^*) = s$. Thus the complexity measure $C_{Ti}(s))$ of string s using machine T_i is $|p^*|$ while on the U the measure is $|e|+i+2+|p^*|$. Hence $C_U(s) = C_{Ti}(s))+sim(T_i)$, where $sim(T_i) = |e| + i + 2$. (The routine e is not usually included in the simulation string as it usually envisaged as part of the UTM definition.) It makes sense to use a simple UTM with a minimal instruction set to be the reference machine. This approach encapsulates most of the instructions in the programme as the complicated programme instructions can be compiled from the minimal set of instructions. For this reason Chaitin uses LISP as as a programme language. He has shown that for LISP programming the simulation or $O(1)$ term is of

the order of 300 bits. Li and Vitányi (page 210) using combinatory logic based on lambda calculus get

$$C(x) \leq |x| + 8 \, and \, C(\emptyset) = 2.$$

An alternative two part definition of $C(s)$ is give by Li and Vitányi (PAGE 107) as

$$C(s) = min_{i,p}\{|T_i| + |p| : T_i(p) = s\} + O(1) \tag{3.8}$$

where the minimisation is over all Turing Machines and programmes that compute s. This definition minimises the combination of the programme length and the description of the Turing Computer that implements the programme. One could put most of the description in the Turing Machine, in which case the programme will be short, or put most of the description in the programme having a simple Turing Machine. There clearly will be an optimum description. As is discussed in section 4.2, the optimum occurs when all the pattern is included in the machine description and all the random structure is in the programme p.

The algorithmic complexity defined as above will be termed the 'plain algorithmic complexity' to distinguish it from a later definition that restricts the codewords to those that are self-delimiting; i.e those where there is no ambiguity about when one codeword finishes and another starts (section 3.2).

Issues with AIT

Crutchfield and his co-workers (Shalizi [91] Feldman, [42]) have noted the following difficulties with the AIT approach to measuring the complexity of strings.

- There is no way to know in advance whether the string s has a shorter description. Even if a shorter description, s' is found, there is no certainty that it is the shortest description. This is equivalent to the Turing halting problem.
- The AIT approach must specify a single string exactly. It appeared that irrelevant information from an observer point of view, including noise, must be exactly specified making the description excessively long and possibly useless for most observed patterns.
- The computational resources (time and storage) to run an AIT algorithm can grow in an unbounded manner.
- The complexity is greatest for random sequences.
- The use of a UTM may restrict the ability to distinguish between systems that can be described by less powerful computational models and the UTM lacks uniqueness and minimality.
- The AIT approach does not capture algebraic symmetries.

The first two bullet points were discussed in the Introduction 1. The halting problem is a fundamental problem and using some other measure of complexity only fudges the problem. The ability of AIT to include noise and variation in strings is no longer a problem. As is discussed further in section 4, the concept of provisional entropy, or Algorithmic Minimum Sufficient Statistic, circumnavigates this perceived problem. The following bullet points discuss the remaining issues

- The third bullet depends on what measure of complexity is appropriate for a particular application. Certainly for many physical situations simple AIT provides insights. However [13] has developed the additional concept of logical depth to capture issues related to resource use, while several other authors have discussed the tradeoff between size and resources [17, 77, 104].

- The fourth bullet point is a matter of definition as the critics recognise.

- The fifth bullet point arises because Crutchfield [39] is able to show that stochastic processes can be explained by a hierarchy of machines. Simpler processes can be described by computing devices that are much simpler than a TM. That this is not a problem can be seen if one uses the definition of algorithmic complexity that optimises the sum of the computer device description and the programme as in equation (3.8). Whenever a simple Turing machine captures the pattern, the description of the simpler device is short. One always needs to be aware of the issues related to uniqueness and minimality. But for many physical situations, such as the measurement of the algorithmic entropy of a system, differences rather than absolute values are the critical measure and these concerns are of no significance.

- The author is not sure what to make of the last bullet point. Unless the point has been misunderstood. Where algebraic symmetries and scaling symmetries are recognised, any algorithmic description recognising the symmetry will be shorter than one that does not.

Crutchfield and his colleagues [41, 39] developed the causal state approach to provide a systematic process to highlight structure in a stationary stochastic series. However the purpose of AIT, in contrast to the causal state approach, is to describe the pattern, not to discover it. Hence any pattern recognition process, that is able to highlight hitherto unobserved structure, can be incorporated into the AIT description. These include the causal state approach, the practical Minimum Description Length perspective (see section 7), and other systematic processes, whether based on symmetry or physical understanding. Once pattern is identified by whatever means, further compression is seen to be possible.

3.2 Self Delimiting Coding and the Kraft inequality

A machine implementing an algorithm must recognise where one instruction ends and another begins. This can simply done be using an end marker to delimit the code or the instruction. However, as Levin [75], Gács [55] and Chaitin [27] have shown, there are many advantages if the coding of each instruction or number is taken from a prefix-free set. In such a set, no code can be the prefix of any other code, and no end markers are necessary. In this case these codes can be called instantaneous codes, because they can be decoded as they are read, or self-delimiting codes as the code indicates its own length. They are sometimes termed prefix codes which would seem confusing as, strictly, they come from a prefix-free set. As has been mentioned, the Kraft inequality holds for self-delimiting codes. Let the codeword length of the ith code from the n codes in the prefix-free set be $|x_i|$; then for binary coding the Kraft inequality states:

$$\sum_i 2^{-|xi|} \leq 1. \tag{3.9}$$

This can be generalised to other alphabets.

This inequality can be understood by generating the tree of all possible codewords (see figure 1). The process is rather like cutting a square in half, keeping one half writing a code of length 1 on one half and keeping the other half for longer codes. As no code is a prefix of any other, only one code of a given length can be assigned. For example, if 011 is assigned as a code of length three, any code beginning with 011 is ineligible as a code. More codes can only be generated by taking the unused code, 010 of length 3, and extending it to either 0101 and 0100. Only one of these extensions of length 4 can be used, and the other can be extended further to produce longer codes. As there is only one assigned code for each code length, the sum of the assigned codes $\sum_i 2^{-|xi|}$ is at most $\frac{1}{2} + \frac{1}{4} + \frac{1}{8}$ etc. and at most equals 1. Figure 1 illustrates the requirement for codewords to satisfy Kraft inequality. Looking at the Figure 1, whenever a codeword is assigned, to ensure no code is a prefix of another, the assigned code cannot have a direct connection with a prior code or a subsequent code in the code tree. Thus any branch in the binary tree that starts with an already assigned code must be blocked off, as the remainder of that branch cannot be used.

Whenever a code is assigned, e.g. 0111 and/or 0110 as in figure 1, at most the contribution to the Kraft sum is $2^{-|0111|} + 2^{-|0110|}$. This is at most 2^{-3}; the same contribution that the parent code '011' would have made to the sum if it were assigned instead. As the contributions are halved at each split. Inspection shows that the overall sum can never be greater than 1.

The converse of the Kraft-McMillan inequality also holds. This inequality states that whenever the codeword sizes satisfy the above inequality, each member in the set of codewords can be replaced by an instantaneous code of the same length that will also satisfy the inequality.

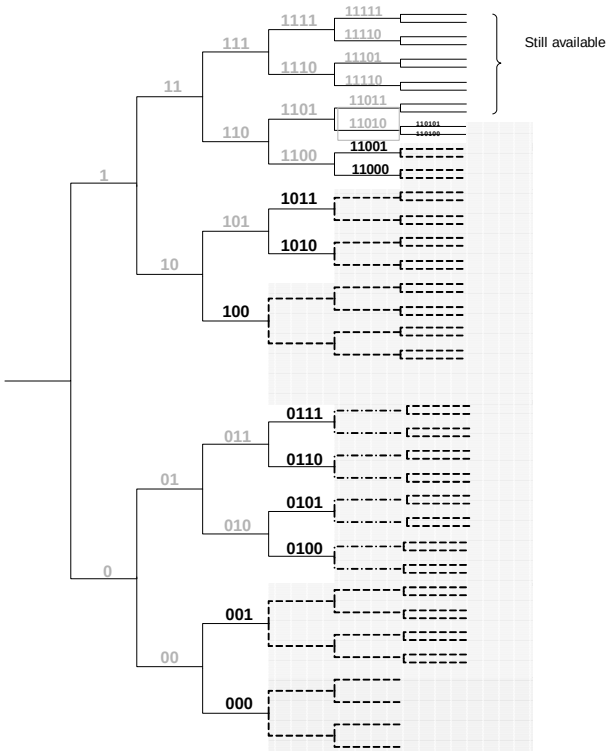

Figure 3.1: Selecting binary codes so that no selected code is a prefix of another

The approach can be extended to complete programmes, not just instructions. If algorithms do not have end markers, no halting algorithm (e.g. the string x representing an algorithm) can be a prefix of any other algorithm (e.g string xy) as, reading from the left, the machine would halt before undertaking the computation y. Thus if there are no end markers, the algorithms must also come from the prefix-free set. As the number of possible algorithms is infinite, the Kraft inequality needs to be extended to an infinite set of prefix-free algorithms. Chaitin [27] has proved the extension, known as the Kraft-Chaitin inequality to infinite recursively enumerable sets.

Chaitin's argument recognises that each assigned algorithm p of length $|p|$ forbids any other algorithm starting with p to be used. If this algorithm is envisaged as $0.pxxxldots\ldots$ it blocks out a region of $2^{-|p|}$ on the number line between 0 and 1. I.e. the region between $0.p0$ and $0.p1$. The sum of the length of all such algorithms can at most equal 1, the length on the number line.

More formally, consider computer $T(p) = x$ where programme p is a natural number in binary form. If p is a self-delimiting programme no other programme can start with p. If one considers all the real numbers between [0,1], the infinite set of strings starting with $0.p$ must be blocked out and cannot be used

for another programme. Section 2.3, introduces the idea of cylindrical sets, where the cylinder Γ_p representing the reals between $0.p0^\infty$ to $0.p1^\infty$ covers all these non-allowed programmes. As section 2.3, shows the Lebesgue or distance measure $\lambda(\Gamma_p)$ on this cylinder is $2^{-|p|}$. If all p come from a prefix-free set, none of the cylindrical sets can overlap (i.e. they are disjoint). The measure property means that the sum over all, possibly infinite p satisfies $\sum_p \lambda(\Gamma_p) = \sum_p 2^{-|p|}$. This sum can be no greater than 1, the value of the Lebesgue measure on all the reals [0,1]. I.e. the Kraft equality holds and $\sum_p 2^{-|p|} \leq 1$. Equality occurs when the union of all the cylindrical sets completely covers the reals.

There is another advantage in using codes from a prefix-free set or self-delimiting coding. In this case, the Kraft inequality ensures that two separate standalone algorithms that halt can be can be concatenated or joined with the insertion of an appropriate linking instruction. Bennett [13] outlines this in the following way (see 3.4). If $U(p, w) = x$ and halts, and $U(q, x) = y$ then $U(rpq, w) = y$, where r is an instruction that takes the output of the first routine to be an input of the second. At the most, the joining code adds O(1) to the length of the combined routine.

Instead of using the plain algorithmic complexity as outlined in the section 3.1, it is more useful for physical situations to use a definition that restricts the codewords to a prefix-free set. The corresponding algorithmic complexity denoted by $K(x)$ becomes;

$$K_U(x|y) = min_{U(p,y)=x} |p|$$

where U only accepts strings from a prefix-free set. Hence

$$K(x) \leq K_U(x) + O(1). \tag{3.10}$$

Also

$$K(x|y) \leq K_U(x|y) + O(1), \tag{3.11}$$

where, in the latter case the input y is fed into the computation process. In practice one evaluates $K_U(x)$ on a standard reference computer, sometimes called a Chaitin computer (see Calude [18]), knowing that the true algorithmic complexity will be within O(1) at worst. Provided that comparisons of algorithmic complexity are measured on the same reference computer, the O(1) term is of little significance.

The algorithmic entropy can then be defined to be the pre-fix free or self-delimiting algorithmic complexity i.e.;

$$H(x) \equiv K(x)$$

and the conditional entropy given an input string y is

$$H(x|y) \equiv K(x|y).$$

In the physical situation the entropy is the preferred concept as it avoids the ambiguity over the meaning of the word complexity and secondly, as is shown

later, (section 5.2) the algorithmic entropy is virtually the same as the Shannon entropy. Furthermore, concepts such as mutual information, conditional entropy etc. arise naturally. Because the algorithmic entropy is the length of an algorithm in the Kraft sum, the algorithmic entropy also must satisfy the Kraft inequality; namely $\sum_{all\ x} 2^{-H(x)} \leq 1$. In the following $H(x)$ or $H_{algo}(x)$ will denote the algorithmic entropy or algorithmic complexity and will usually be used in preference to $K(x)$. The Shannon entropy of a set will be denoted by H_{sh}.

3.3 Optimum coding and Shannon's noiseless coding theorem

If one wishes to code a message efficiently, it makes sense to use the shortest codes for the message symbols that occur most frequently. Shannon's noiseless coding theorem shows how this can be done. Consider a set of message symbols s_1, s_2, s_3, s_4, s_5,.... s_k that occur in the expected message with probabilities p_1, p_2, p_3, p_4, p_5, ... p_k. Let the self-delimiting or prefix-free binary code words for these be $code(s_1)$, $code(s_2)$, $code(s_3)$, $code(s_4)$, $code(s_5)$, ...$code(s_k)$. Shannon - Fano coding is an efficient coding methodology that satisfies the Kraft inequality with code words constrained by:

$$-log_2 p_k \leq |code(s_k)| \leq 1 - log_2 p_k. \tag{3.12}$$

This implies that $2^{-|code(s_k)|} \leq p_k \leq 2^{1-|code(s_k)|}$. Shannon-Fano coding can be implemented by ordering the message symbols from the most probable to the least probable. Divide the list into two halves so that the total probabilities in each half are as near equal as possible. Assign a 0 to the most probable group and a 1 to the other group. This is the first digit of the code. Repeat this for each subgroup in turn assigning a further 0 or a further 1 and so on until every probability is assigned. By splitting in this way, the process ensure that no code is a prefix of any other and the code lengths are no more than $-log_2 p_i + 1$.

An alternative, Huffman coding combines probabilities to form even more efficient coding. An arithmetic code [82] is a more efficient based entropy code and is in principle slightly better. Hence, given the probabilities of symbols in the message, the average length of coding can be made very close to the Shannon entropy of the source; i.e. the entropy based on the expected occurrence of symbols. The expected code length per symbol (i.e. $\sum p_k |code(s_k)|$) in a message coded in this way, is given by

$$H_{sh} \leq \sum p_k |code(s_k)| \leq H_{sh} + 1 \tag{3.13}$$

where $H_{sh} = -\sum_k p_k log_2 p_k$ is the Shannon entropy of the set of source symbols. As a consequence, given the probability distribution of the symbols, the expected length of a message consisting of N symbols can be made close NH_{sh}. This equation is known as Shannon's noiseless coding theorem. A code where

the expected code length per symbol equals the entropy, is optimal for the given probability distribution and cannot be bettered.

The following subsection shows how to optimally code the natural numbers when the probability distribution of the natural numbers in the set of message symbols is not known. Later in section 3.8 the algorithmic equivalent of Shannon's coding theorem will be discussed.

Delimiting coding of a natural number

As the algorithmic entropy requires the instructions to be self-delimiting, one needs to consider how to do this for a natural number. It is straightforward to assign self-delimiting codes to a finite set of instructions or numbers with a known probability distribution using Shannon's noiseless coding theorem outlined in the previous section. The programme can uniquely read each code and then decode it using a table or a decoding function. However, it is often necessary to code a natural number as an input to an algorithm, for example to provide information to define the number of steps in computational loop. There are two problems in doing this. The first is that the conventional binary representation of a natural number is ambiguous as it is unclear whether the string '010' is the same as the string '10'. The ambiguity of a binary representation can be avoided by coding the number using lexicographic (dictionary) ordering. E.g. the integers from 0 to 6 etc. are coded lexicographically as $\emptyset, 0, 1, 00, 01, 10, 11$, etc, where $\emptyset$ is the empty string. It can be seen that the lexicographic equivalent of a natural number n is the binary equivalent of $n+1$ with the leading 1 removed. This is equivalent to $|n| = \lfloor log_2(n+1) \rfloor$ where the floor function, in the equation is just the integer below the term in the L shaped brackets. Here this will be denoted by $l\hat{o}gn$, i.e. $|n| = l\hat{o}gn$, noting that this is, at worst, one bit less than $log_2 n$. The second problem is that where there is no probability distribution for the numbers to be coded, Shannon's theorem cannot be used and one must find another way of making the codes self-delimiting.

While lexicographic coding is not self-delimiting, an inefficient form of a self-delimiting instruction can be constructed by adding a prefix string of n 1s followed by a zero in front of the lexicographic representation to indicate the code size n. More formally, if n represents a natural number in lexicographic form, a possible self-delimiting universal coding of the natural number is $1^{|n|}0n$ or (using the $l\hat{o}g$ notation) $1^{l\hat{o}gn}0n$. The length of the code then becomes $2l\hat{o}gn + 1$. E.g. the number 5 is '10' lexicographically, and in this universal self-delimiting form becomes 11010 (i.e. two 1's to identify the length of '10, which is the code for 5, followed by a zero and finally 10). The machine reading the code counts the number of '1s' until the '0' is reached. The number of '1s' (i.e. 2 in the example) tells the decoder the length of the code that is to be read following the '0'.

While this code is universal, as it optimises the average code length irrespective of the unknown probability distributions, this a a lengthy representation of the number and is not asymptotically optimal (Li and Vitányi [77] p.78,79).

Calude [18] page 23 has a more compressed self-delimiting representation of n of length $1 + l\hat{o}gn + 2l\hat{o}gnl\hat{o}gn$ bits. Alternatively, a more compressed code can be generated by an iterative procedure. If $code_1(n) = 1^{|n|}0n$ the first iteration is given by $code_2(n) = code_1(n)n$. This, as in the Calude case, is of length $1 + l\hat{o}gn + 2l\hat{o}gnl\hat{o}gn$ bits. A further iteration where $code_3(n) = code_2(n)n$ has length $1 + l\hat{o}gn + l\hat{o}gl\hat{o}gn + 2l\hat{o}gl\hat{o}g1$. In principal there exists a code for n with length given by the infinite series,

$$1 + l\hat{o}gn + l\hat{o}gnl\hat{o}gn + l\hat{o}gl\hat{o}gl\hat{o}gn \ldots$$

In practice the size of the representation of n will be $\leq 1 + l\hat{o}gn + l\hat{o}gl\hat{o}gn + 2l\hat{o}gl\hat{o}g$ or a higher expansion. The additional information required for self delimiting coding can be interpreted as $H(|n|)$, the length of the algorithm that specifies the size of n.

An important question is how significant is the length information about the code for a natural number in determining the algorithmic entropy. Are the *loglog* and higher terms always needed? The answer is that for equilibrium (or a typical)l state in thermodynamic system, the length information is insignificant. For example, there are about $N = 10^{23} = 2^{69}$ particles in a mole of a Boltzmann gas. The algorithmic entropy and of an equilibrium state requires the position and momentum of each of the 2^69 particles to be specified to a required degree of precision. Assume both the position and momentum coordinates of each are defined to 1 part in 2^8. This requires 16 bits per particle. For 2^69 particles16×2^69 bits are required. The algorithmic entropy of an equilibrium configuration is therefore approximately 2^73 bits. The *loglog* term , which is the first in the sequence capturing the length information, is $log_2 2^73 = 73$. This is an insignificant contribution to the algorithmic entropy. The next term in the series is $log_2(73)$ which is smaller still. However where a particular string is highly ordered, the algorithmic description will be much shorter and the contribution the higher *loglog* term may be need to be included.

The discussion here is relevant to the evaluation of the provisional entropy of a string in a set of states discussed in section 4.1. The provisional entropy consists of a two part code, one part defines the set of strings exhibiting a particular structure or pattern, the other identifies the string within the set. In this case, it may be possible to embody the length information in the description of the set.

The relationship with entropy, conditional entropy etc.

An important question is when is it more efficient to compute two strings x and y together using a single algorithm rather than separately. In other words when is

$$H(x, y) < H(x) + H(y)?$$

If x and y are independent, there is clearly no gain in calculating them together in a single algorithm, but if y has information about x it would suggest that computing y first will lead to a shorter description of x given y. In the case

of the Shannon entropy, in general $H_{sh}(X,Y) = H_{sh}(Y) + H_{sh}(X|Y)$. If the random variable X does not depend on Y, knowing Y does not decrease the uncertainty in X, i.e. $H(X|Y) = H(X)$. In which case $H_{sh}(X,Y) = H_{sh}(Y) + H_{sh}(X)$.[1] A similar, but not identical, relationship occurs with the algorithmic entropy. If the strings x and y are to be described together, information about y may simplify the description of x. However, in contrast to the Shannon case Chaitin [31] has shown that, unless the compressed description y^* of y is used, algorithmic corrections are required. Let the reference computer which, when given the null string $\emptyset$ and minimal programme y^*, undertake the computation $U(y^*, \emptyset) = y$. If y^* is used instead of y as an input to the computer that generates x with the minimal programme x^*, then given y^*, $U(x^*, y^*) = x$. As both x and y can be computed knowing y^*,

$$H(x,y) = H(y) + H(x|y^*) + O(1)$$

Note that y^* contains information about both y, and the length of the shortest description of y. If y rather than y^* is to be used, $H(y)$, the length of the most compressed description of y needs to be passed to the computation of x, i.e.,

$$H(x,y) = H(y) + H(x|y, H(y)) + O(1).$$

The above equation recognises that, in many cases, string x can be computed more efficiently once the minimal description of string y is known. Where this is the case $H(x|y^*) \leq H(x)$ indicating that x and y can be calculated more efficiently together than separately. I.e.

$$H(x,y) \leq H(x) + H(y) + O(1).$$

As computing x independently may be achieved by a shorter programme than one that computes both x and y, $H(x) \leq H(x,y)$. In general,

$$H(x) \leq H(y) + H(x|y^*) + O(1). \tag{3.14}$$

When the full description of y^* is essential to compute x, the equals sign is necessary. This occurs when y must be computed as part of the shortest algorithm to compute x. For example the equals sign is needed when y^* represents the compressed version of the subroutines that express the physical laws needed to calculate x in a physical situation. In general, equation 3.14 shows programmes can be joined or concatenated in a subadditive manner to within O(1). Here, the $O(1)$ term is the few extra bits required to join two separate algorithms as was outlined in 3.2 and is further discussed in 3.4.

Shannon has defined the mutual information between two sets X and Y as $I(X:Y) = H(X) + H(Y) - H(X,Y)$ The mutual information is a measure of how many bits of uncertainty are removed if one of the variables is known. In the algorithmic case, the mutual information between the two strings is

[1] Note for the Shannon entropy, upper case labels are used to indicate the sets of random variables.

denoted by $I(x:y)$ (or $H(x:y)$). Given the minimal programme for x, the mutual information $I(x:y)$ can be defined as:

$$I(x:y) = H(x:y) \equiv H(x) - H(x|y^*)$$

or

$$I(x:y) \equiv H(x) + H(y) - H(x,y) + O(1)$$

Readers should be aware that definition of mutual information can differ slightly between authors. For example Chaitin in references [31, 28] has defined a symmetric version of the mutual information by using the second equation but without the O(1) term. In which case an O(1) term is transposed to the first equation. However, in the more fundamental paper, [27] Chaitin uses the definition above. Nevertheless, the difference in the definition is insignificant for measures on large strings. The remarkable similarity of the algorithmic entropy relationships with the Shannon entropy should not be overlooked. Neither should these be seen as being the same . Allowing for the use of y^* instead of y and the O(1) term they look identical. However the Shannon relationships show how the uncertainty in observing a random variable can be reduced by knowing about a dependent random variable. The algorithmic relationships show how computing one string simplifies the computation of a related string where there is common information. The algorithmic expression is about actual information shared between two strings. From the above algorithmic results other important results also follow.

$$H(x,y) = H(y,x) + O(1),$$

$$H(x) \leq |x| + H(|x|) + O(1)$$

with equality for a typical or close-to-random string.

$$H(x) = H(|x|) + H(x||x|) + O(1),$$

which for the case of a random string of length n gives

$$H(x) = H(n) + H(x|n) + O(1).$$

In algorithmic information theory, the information embodied in a string that specifies a state of a system or structure, is the minimum number of bits needed to specify the string and hence the system. This makes intuitive sense and it follows that information $I(x)$ is the same as the entropy i.e.,

$$I(x) = H(x).$$

Section 3.5 defines a halting probability Q. While Q is not a true probability, the algorithmic mutual information between two specific strings x and y can be defined in terms of Q. I.e.

$$I(x:y) = log_2(Q(x,y)/(Q(x)Q(y)) + O(1).$$

The expectation value of the mutual information takes the same form as the Shannon definition of mutual information using probabilities but with $Q(x,y)$ instead $P(X,Y)$ etc.

3.4 Entropy relative to the common framework

If y is essential as an input to generate x, and q^* is the routine that generates x given y, then $(U(q^*, y) = x$. The requirement that y is essential is the requirements that y must be generated during the computation that generates x. However the routines p and q cannot be combined directly as they are self-delimiting. In other words the string p^*q^* is not a valid algorithm. If we take the order of computation to start with the left most routine, the computation $U(p^*q^*)$ will terminate on running p^* and give just the output y. However a small linking routine, r^* of O(1),[13, 27] allows x to be generated by a combined programme of the form $U(r^*p^*q^*) = x$. Here the * notation is used to indicate that these are minimal programmes on the reference UTM. The algorithmic entropy is then the combined entropy given by $|p^*| + |q^*| + |r^*|$. It follows for this particular case where y is necessary to generate x , $H(x) = H(x, y)$. The algorithmic entropy of string x can then be expressed as $H(x) = H(y)+H(x|y, H(y))+O(1)$ as y must be generated on the way to generating x. Nevertheless, this simple additive expression only covers the case of optimally compressed essential routines. In the case where string y is not an essential input to x, $H(x) \leq H(x, y)$ and $\leq H(y)+H(x|y, H(y))+O(1)$.

As the entropy difference rather than the absolute entropy has physical significance, in any physical situation, common information strings, in effect the common instructions, can be taken as given. Effectively this information plays the role of the routine p^* in the previous paragraph. Common information can include the binary specification of common instructions equivalent to"PRINT" and "FOR/NEXT" etc. and the O(1) uncertainty implied by the reference UTM. In a physical situation, the common information can include the physical laws and for thermodynamic systems, the algorithmic description of the phase space grain size [108]. In what follows the common information routines will be represented by 'CI' and, given the common information, the physically significant entropy will be denoted by the conditional entropy $H_{algo}(x|CI)$. As the entropy difference between two configurations x and x' is $H_{algo}(x|CI) - H_{algo}(x'|CI)$, the entropy zero can to be $H(CI)$ and the common contribution ignored.

An ordered sequence is recognized because there is an implicit reference to a patterned set that will be recursively enumerable. Any sequence x of this set can be at least partially compressed (see section 4) and the primary entropy contribution will depend on parameters such as the length of the string and the number of possible states in the system. The conditional algorithmic entropy of a member of this patterned set will be termed the 'provisional' entropy. The provisional entropy is the entropy of a typical member of the patterned set; it is the upper measure of the conditional algorithm for member. However a very few non-typical members of the set may be further compressed. Whenever a member of a set can be compressed further, a more refined model is needed to capture the pattern in those particular strings. This is discussed further in the section 4.2 on Algorithmic Minimum Sufficient Statistic.

3.5 Entropy and probability

Let the algorithmic halting probability $Q_U(x)$ be the probability that a randomly generated programme string will halt on the reference UTM, denoted by U with x as its output [27]. The randomly generated programme that has a halting probability $Q_U(x)$ can be considered as a sequence of 0's and 1's generated by the toss of a coin. In which case, $Q_U(x)$ is defined by

$$Q_U(x) = \sum_{U(p)=x} 2^{-|p|} \tag{3.15}$$

The sum of all such probabilities, $\Omega_U = \sum_x Q_U(x)$. As each algorithm that generates x is comes from a prefix-free set $\Omega_U \leq 1$ because of the Kraft-Chaitin inequality. However, as $\Omega_U \neq 1$, $Q_U(x)$ cannot be a true probability but is instead a semimeasure. Furthermore, there is no gain in normalising $Q_U(x)$ by dividing by Ω_U as Ω_U is not computable. Even so, this concept allows a very important connection to be made between the algorithmic entropy of a string and the halting probability. The argument is as follows.

- $Q_U(x) \geq 2^{-H_U(x)}$ as $H_U(x)$ is the minimum $|p|$ to generate x while the sum defining $Q_U(x)$ includes not just one but all programmes that generate x. Basically the shortest programme generating x in the sum for $Q_U(x)$ is $H(x)$ and is a dominate contribution. This leads to the inequality,
$$H_U(x) \geq -log_2 Q_U(x) \tag{3.16}$$
- But, $H(x)$ cannot be much greater than $-log_2 Q_U(x)$ as there are only a few short programmes able to generate x. All the longer descriptions contribute little to the sum as is shown immediately below. Hence, it can also be shown that
$$H_U(x) \leq \lceil -log_2 Q(x) \rceil + c \tag{3.17}$$
Because of equation 3.16, it must follow that
$$H_U(x) = -log_2 Q(x) + c. \tag{3.18}$$

The upper limit on $H_U(x)$ (equation (3.17)) can be seen by noting (Downey private communication) that, as $H_U(x)$ is an integer, $H_U(x) \geq \lceil -log_2 Q_U(x) \rceil$. But $\lceil -log_2 Q_U(x) \rceil$ can also be used in an algorithm to generate string x. More specifically, a routine p that takes $\lceil -log_2 Q_U(x) \rceil$ and decodes it to give the string x, could be measure of algorithmic complexity of x. Basically the routine p takes the integral value of $-log_2 Q(x)$ and, starting with the lowest programme of this length in lexicographic order, steps through all allowable routines to see which, if any, generate the string x. If no routine generates x, the programme increments the length by one, and again steps through all allowable routines of this length. (This is a thought experiment as some routines may not halt. However, we are only interested in finding those that do.)

Eventually one, close to $-log_2Q(x)$ will generate s. The length of this routine cannot be less than than $H_U(x)$ by definition. I.e. the programme p with length $|p| = \lceil -log_2Q(x) \rceil + |stepping\, routine|$ is an upper measure of the algorithmic entropy or complexity. From the second bullet point above, it follows that $H_U(x) \leq |p|$ or

$$H_U(x) \leq \lceil -log_2Q(x) \rceil + c \tag{3.19}$$

where c represents the length of the decoding routine. This upper limit can be combined with equation (3.16) leading to the relationship;

$$-log_2Q(x) \leq H_U(x) \leq \lceil -log_2Q(x) \rceil + c.$$

Equation (3.18) follows by including the fraction that rounds up $-log_2Q(x)$ in the constant c. Note the O(1) constant c is not dependent on x. Also, as it will only vary slightly for different UTMs, one can drop the suffix U representing the specific UTM. It can be seen that the constant c indicates that there are 2^c programmes close to $H(x)$. If there are 2^c programmes of a size close to $H(x)$, and one ignores the significantly longer programmes it follows that $Q(x)$ would be $\leq 2^c 2^{-H(x)}$. This implies what was stated above that $H(x)$, $H_U(x) \leq -log_2Q_U(x) + c$. In effect, the constant 2^c becomes a measure of the small number of short programmes that make a significant contribution to $Q(x)$.

$Q(x)$ is a unnormalised measure of the probability that a string will produce x on the UTM. However it can be envisaged as a probability by assigning the unused probability space to a dummy outcome. This allows $Q(x)$ to be treated as a probability measure for a particular message x in an ensemble of messages. Relative to this probability, the expectation value of $H(x)$ over an ensemble of messages or strings then is close to $\langle H(x) \rangle = -\sum_x Q(x) log_2 Q(x)$. This shows that equation 3.19 in the previous paragraph is the algorithmic equivalent of Shannon's coding theorem. The algorithmic code for a string, its algorithmic entropy, is just a constant away from the halting probability of that string. This again reinforces what has been shown earlier, that, while the algorithmic entropy and the Shannon entropy.

A more general approach to prove the same relationship, that relies on the properties of a universal semimeasure will be outlined in section 3.8.

3.6 The fountain of all knowledge: Chaitin's Omega

Chaitin [27] asked the question; "What is the probability that a randomly selected programme from a prefix-free programme set, when run on a specific UTM will halt with a given output x?" Section 3.5 defined Q_U as this probability. Strictly it is a defective probability and should be termed a semimeasure (see section 3.5). The halting probability Ω_U is obtained by summing Q_U over all halting outcomes x generated by a random input.

$$\Omega_U = \sum_{U(p)halts} 2^{-|p|} = \sum_x Q_U(x).$$

Ω_U is between 0 and 1. As Ω contains all possible halting algorithms, if the universe operates under computable laws, Ω contains everything that can be known about the universe; it embodies all possible knowledge. While the value of Ω is computer dependent, it can be converted between UTMs. Unfortunately, all the contributions to Ω_U can only be established by running all possible programmes to see if they halt. Every k bit programme p that halts, (i.e. where $|p| = k$) contributes 2^{-k} bits to Ω_U. Ω is random in the Chaitin or Martin-Löf sense (section 8.2). If Ω could be generated by a simpler programme it would then be possible to predict whether a particular programme contributing to Ω will halt. This cannot be possible, as doing so would violate the Turing halting theorem. Chaitin (who named this number Ω) showed that, just as there is no computable function that can determine in advance whether a computer will halt, there is also no computable function able to determine the digits of Ω. Ω is uncomputable and can only be approximated by checking each possible programme from the smallest upwards to see which might halt in a given time.

3.7 Gödel's theorem and Formal Axiomatic Systems

Chaitin [30] has developed the algorithmic equivalent of Gödel's theorem. He has shown that some possible theorems or propositions cannot be derived from a simpler set of axioms and rules. As these propositions are represented by a string that cannot be compressed in terms of the axioms, they are undecidable. They may be true or false. This is of course Gödel's theorem, but with the twist that unprovable theorems are everywhere. As there are at most only $2^n - 1$ strings shorter than a string of length n, there are insufficient strings to compress most strings. Therefore the associated theorems are undecidable. Consider a formal logical axiomatic system that is as rich as arithmetic with multiplication, and which is required to be both consistent and sound. No statement in such a system can be both true and false, and only true statements can be proven to be true. In order to explore Chaitin's approach using AIT, let 'A' by a string of symbols that represents the axioms of the system. The length of the shortest algorithmic that describes the axioms is denoted by $|A|$. The set of all provable theorems is recursively enumerable. This means one can generate all theorems by a process that steps through all possible theorem proofs in ascending order of length and uses a proof checking algorithm to determine whether a particular statement generated is a valid theorem.

In principle, each provable theorem can be generated from the axioms plus the rules of inference. Those strings that might represent theorems, but which have no shorter description, do not belong to the recursively enumerable set of true theorems. As they cannot be expressed in terms of the axioms and rules of inference they cannot be shown to be true or false. They are undecidable. There are far statements that belong to the theorem set than valid possible theorems.

Worse, as the length of the possible theorem strings increases, the set strings associated with unprovable statements grows faster than the provable ones. One can never be sure that a potential theorem is decidable — only when a proof (or the proof of its converse) is found will this be known. I.e. it is only when a potential theorem can be expressed as a compressed description in terms of simpler theorems (or the axioms) and rules of inference, can the theorem be considered proven . This is of course Gödel's theorem but with and upsetting twist that un provable theorems are everywhere; the provable theorems are only a small selection of the space of all possible theorems. An alternative way of looking at this is to tie the argument into the halting problem. The undecidability of the halting problem underpins the undecidability of a rich formal system.

The above argument can be made more robust. Denote n as the length of string x where the statement "$H(x) \geq n$" is a statement to be proved in the formal system. This statment asserts that there exists a string x where the algorithmic complexity of the string is at least as great as its length. Such a string would be deemed random, as its algorithmic description is at least as long as its length. The computational algorithm that generates the random string needs to specify each digit plus some overheads depending on the computer used. It has already been argued that many such strings must exist, but the question is whether the statement, that a string with algorithmic complexity at least equal to n can found and proved to be genuine. I.e. given n, is it possible to find a particular x that is random or has no shorter description than it length? There are two ways to look at this. The first is to develop a procedure to step through all the valid proofs that can be derived from the axioms and the laws of inference until an algorithmic proof p finds the first x such that $H(x) \geq n$. As has been mentioned, this is akin to stepping through all the integers until one is found with a particular property. The proof p will contain the axioms, embody the algorithm stepping through and checking the proofs, as well as the specification of n to indicate when the string of interest has been found. As the size of n can be represented by $log_2 n$ bits in the algorithm, the length of this algorithm is:

$$|p| = |A| + log_2 n + |stepping\ algorithm\ and\ proof\ checker| + O(1). \quad (3.20)$$

This implies that a computational procedure T exists that finds p such that $T(p) = x$. But $|p|$ cannot be less than the shortest possible description defined by $H(x)$, otherwise p could be used to provide a more compressed definition of x. If c is taken to be the length of the description of the system and the stepping algorithm, and because $|p|$ cannot be less than the shortest description of x, it follows that

$$H(x) \leq |p| = log_2 n + c.$$

Here the finite constant c has absorbed the length of the fixed strings in equation 3.20. This implies that a proof has been found that outputs the first x where $H(x) \geq n$. But as $log_2 n + c > H(x)$, it follows that $log_2 n + c \geq n$. But this becomes impossible as for any given finite c, there will always exist an n

large enough so that $n - log_2 n \geq c$. For large n such a search for the proof can never halt as such a proof cannot be found.

A slightly different argument that is more in line with the original Gödel approach, codes a meta statement about the formal system within the system. Let us try and prove the meta statement that there exists an n for which the statement "$H(x) \geq n$ is not provable". This statement should either be true or false in the formal system. Assume the statement is false, this means we can find a contrary example - a value of x for which $H(x) \geq n$. Just as in the earlier argument If the system is sound, as there can be no contradiction, for n sufficiently large, where $n > c$, no such string p can be found. Again

$$log_2 n + c \geq n.$$

We cannot prove the statement is false, therefore the original statement must be undecideable.

While these rogue statements can be made decidable by including them as new axioms in a more complex formal system (in the sense of having a longer description), the number of random strings corresponding to unprovable statements, grows faster than the provable ones. As Chaitin [32] put it: "if one has ten pounds of axioms and a twenty-pound theorem, then that theorem cannot be derived from those axioms". This leads Chaitin to speculate that mathematics should accept this uncertainty and become experimental [31, 34, 36].

This can tell us something about Chaitin's somewhat mystical number Ω. Ω can be constructed from the provable theorems in a logical system - there is no contribution from undecidable theorems as the generating computation will not halt for these. If Ω represents the logical system that encapsulates all the physical laws, knowing Ω is equivalent to knowing everything. If Ω were computable, all the physical laws could be expressed simply. Interestingly Calude, Dinneen, and Shu [20] have calculated Ω to 64 places and get the binary value $\Omega = 0.0000001000000100000110001000011010001111110010111011101000010000\ldots\ldots$

While there are mathematical propositions in any formal system that are not decidable, it is impossible to determine which these are, even though such undecidable propositions grow faster than the decidable ones as the description length of the propositions increases. If it is assumed that a possible undecidable proposition is true, the formal system can be extended. However there is always the danger that the proposition was actually decidable and the assumed axiom could be inconsistent with the formal system. This has led Chaitin to suggest [34, 32] that mathematics should become experimental; useful mathematical hypotheses could be taken to be true unless later inconsistencies emerge. For example the Riemann hypothesis may be undecideable (although, for an update see the article by Kvaalen [70] in New Scientist that suggests it might be provable), yet experimentally it seems likely to be true. This hypothesis could then be taken as an extra axiom with the possibility that more new and interesting theorems could then be proved using the new axiom.

There is a further implication that many have wrestled with. Does the human mind operate as a finite Turing Machine when undertaking logical

thought? If so, it will be constrained by Gödel's theorem. Barrow chapter 8 [8] scopes the views of some of the key players in the debate, while more recently Feferman [51] discusses the historical development. The particular issues surrounding strong determinism and computability have been explored by Calude et al [19]. It seems to the present author that, while the human brain or mind need not act as a finite Turing Machine to hit upon new truths about the universe, formal logical processes would appear to be needed to prove that these are truths. In which case, the proof process, rather than the discovery, will be constrained by Gödel's theorem. While the human mind can add new theorems and use external devices to extend its computational capability, once the axioms and laws of inference become too great, the mind may not have sufficient capability in terms of its complexity to be assured of the consistency of the formal processes. Thus there may well be laws that are beyond human comprehension in that these laws will not be able to be deduced from simpler principles or be understood in simpler terms. Clearly not all agree.

3.8 The algorithmic coding theorem, universal semimeasure, priors and inference

It was shown in equation (3.12) that for an enumerable probability distribution P, where the source alphabet of symbols have probabilities as $p_1, p_2 ... p_k ..$ etc, a set of optimal codes denoted by $code(s_j)$ can be found for each symbol s_j. The expected code length of a message made up of these symbols, gives Shannon's noiseless coding theorem

$$H_{sh} \leq \sum p_k |code(s_k)| \leq H_{sh} + 1.$$

I.e. the expected length of the coded message is within one bit of the Shannon entropy of the message. As section 3.5 showed, Algorithmic Information Theory has a coding theorem, called the algorithmic coding theorem, which is the equivalent of Shannon's noiseless coding theorem. This section looks at an alternative approach to the algorithmic coding theorem based on establishing a universal semimeasure that is optimal. Just as one can have a Universal Turing Machine that, within fixed boundaries, can simulate any other Turing Machine, one can have a universal code that is asymptotically optimal in coding a message independently of the distribution of the source words. As is outlined below, and implied in section 3.5, this code for x is given by the semimeasures $H(x)$ or $log_2 Q(x)$. The two are equivalent to within an $O(1)$ constant.

The derivation is based on the fact that there is a universal enumerable semimeasure that is an optimal code for a discrete sample space. It is optimal in the sense that it must multiplicatively dominate all other constructive semimeasures. I.e. up to a multiplicative constant it assigns a higher probability to an outcome than any other computable distribution. In other words, if $m(x)$ is the universal enumerable semimeasure then $m(x) \geq c\mu(x)$ where $\mu(x)$ is any other semimeasure in the discrete sample space. The significance of this will become apparent, but firstly it should be noted that the universal property

only holds for enumerable semimeasures, not for the class of all semimeasures or the more restricted class of recursive semimeasures (see Li and Vitányi page 246 [76] and Hutter [64]). Gács [57] has shown that $m(x)$ is also constructive (i.e. computable from below).

While there are a countably infinite number of universal semimeasures, the universal halting probability $Q(x)$ can be taken to be a reference universal semimeasure. All semimeasures can in principle be enumerated by establishing the halting probability $Q_T(x) = \sum_{T_i(p)=x} 2^{-|p|}$ for every Turing Machine T_i. Let the universal halting probability $Q(x)$ be defined for a universal machine that can simulate any Turing machine. The set of all random inputs giving rise to $Q(x)$ will include the simulation of every Turing Machine and every programme p that runs on each Turing machine T_i. As $Q(x)$ includes every semimeasure it multiplicatively dominates them all. The universal probability is therefore a universal semimeasure. However, $2^{-H(x)}$ is also a constructive semimeasure, and because $H(x)$ is equal to $-log_2Q(x)$ to within an additive constant, $2^{-H(x)}$ is equal $Q(x)$ to within a multiplitive constant. $H(x)$ must also be a universal semimeasure. In other words, the semimeasures, $m(x)$, $Q(x)$ and $2^{-H(x)}$ are all universal, and are all equal to within a multiplicative constant as they, all multiplicatively dominate each other. As a consequence, their logarithms additively dominate each other.

This gives rise to the algorithmic equivalent of the coding theorem [75]. The following equality holds to within an additive constant c_p or an O(1) term.

$$H(x) = -log_2m(x) = -log_2Q(x)$$

This approach provides an alternative derivation of equation (3.18). For practical reasons it is convenient to take the universal semimeasure as $m(x) = 2^{-H(x)}$.

The Shannon coding theorem (3.12) shows that for a given distribution $P(x)$, a binary self-delimiting code can be constructed that, on average, generates a code for string x that is close to $log_2P(x)$ in length and this is optimal for that probability distribution. However such a code is not universal for all probability distributions. In contrast, the algorithmic coding theorem shows that there is a universal code for any enumerable distribution (i.e. one that can be specified to a given number of significant figures). This code word for x is the shortest algorithm that generates x. Its length is $H(x)$ or $log_2Q(x)$ to within an additive constant c_p independent of x.

The universal enumerable semimeasure $m(x)$ is a measure of maximum ignorance about a situation. It assigns maximal probability to all objects (as it dominates other distributions up to a multiplicative constant) and is in effect a universal distribution. As a universal distribution, the universal semimeasure provides insight into defining a typical member of a set and provides a basis for induction based on Bayesian principles as is outlined in the next section.

Inference and Bayesian statistics and the universal distribution

Laplace developed the concept known as the principle of indifference, or the principle of insufficient reason, from an idea first articulated by Jakob Bernoulli. This concept provides a tool to cope with situations where a particular outcome might have several possible causes. Because there is no reason to favour one cause over any other, the principle states that they all should be treated as equally probable. I.e. in the absence of any evidence that a tossed coin is biased, this principle would give equal probability to a head or a tail. [2]

The principle leads to Bayes' theorem which determines how likely a particular hypothesis $\mathcal{H}_i$ is, given information D. I.e. the probability of the hypothesis given the data is

$$P(\mathcal{H}_i|D) = P(D|\mathcal{H}_i)P(\mathcal{H}_i)/P(D). \tag{3.21}$$

This rule, which is little more than a rewrite of the definition of conditional probability, provides a basis for inference. If $\mathcal{H}_i$ represents a member of the set of hypotheses or models that might explain the data D, then the inferred probability or posterior probability is defined as $P(\mathcal{H}_i|D)$; the probability of that particular hypothesis given the data. If the probability $P(\mathcal{H}_i|D)$ of one particular hypothesis $\mathcal{H}_i$ is significantly greater than the alternatives, there is some confidence that $\mathcal{H}_i$ is to be preferred as an hypothesis. I.e. after seeing data D, $P(\mathcal{H}_i|D)$ gives a measure of how consistent the data is with the hypothesis. However, the Bayes' approach would seem to be of little value unless $P(\mathcal{H}_i)$, the probability that $\mathcal{H}_i$ is the correct hypothesis, is known for all possibilities. I.e. one would need an accurate measure of the $P(\mathcal{H}_i)$ for every possible hypothesis and ensure that $\sum_i P(\mathcal{H}_i) = 1$.

Nevertheless, if there exists some estimate of the probability of a hypothesis $P(\mathcal{H}_i)$ in the absence of any data, that estimate can give a measure of how consistent the particular hypothesis is, given the data, by using Bayes' theorem. In which case $P(\mathcal{H}_i)$ is known as the prior probability or the initial probability and the resultant probability, $P(\mathcal{H}_i|D)$, becomes the inferred probability given the data. Once an estimate of $P(\mathcal{H}_i|D)$ is established for the set of hypotheses, this information can be used as a new prior for a further estimate of $P(\mathcal{H}_i|D')$ using new data D'. If this iterative process is continued, $P(\mathcal{H}_i|D)$ will, depending on the evidence or data, generally favour one hypothesis over others. Thus this process of inference makes as much use of the available data, as is possible. However there is a major difficulty that statisticians from a frequentist perspective have with this approach. In the frequentist framework a large sample of outcomes is needed to infer a probability measure. There is no place for assigning probabilities for one-off events, whether these be based on the principle of indifference, or based on estimates of likely causes. Despite

[2] Of course this approach does not take into account the cost of a mistake in assuming no bias. While each cause can be assumed to be equally likely, one would need a betting or insurance strategy to cover hidden biases as is discussed later.

this, Bayesian statistics is now commonly used and Jaynes has encapsulated this principle in his idea of Maximum Entropy (section 7).

It emerges that the universal semimeasure $m(\mathcal{H}_i) = 2^{-H(\mathcal{H}_i)}$, provides the best estimate of the prior probability $P(\mathcal{H}_i)$. In simple terms, where $P(\mathcal{H}_i)$ is not known, $2^{-H(\mathcal{H}_i)}$ can be taken to be the prior probability as a best guess. Furthermore, Vitányi and Li [104] argue that that optimal compression is almost always the best strategy for identifying an hypothesis. The arguments above are taken further in the section 7 discussing the minimum description level approach to model fitting.

The universal semimeasure and inference

Originally, Solomonoff [93], in dealing with a similar inference problem, developed what in effect was the algorithmic approach. Solomonoff argued that a series of measurements, together with the experimental methodology, provided the set of data to feed into Bayes' rule in order to give the best estimate of the next data point. Solomonoff came up with the concept of the universal distribution to use as the Bayesian prior; the initial probability estimate. However Solomonoff's initial approach ran into problems as he did not use self-delimiting or codes from a prefix-free set. Levin [75], by defining the universal distribution in terms of a self-delimiting algorithm resolved that problem. As a consequence, the universal distribution of section 3.8 can be taken as the universal prior for a discrete sample space. Levin has argued, the universal distribution gives the largest probability amongst all distributions, and it can be approximated to from below. Effectively $m(x)$ maximises ignorance, or encapsulates the principle of indifference, and so is the least biased approach to assigning priors. Gács [57] using $2^{-H(x)}$ as the universal distribution (see section 8.2) has demonstrated the remarkable property that the randomness test for x given the universal semimeasure; namely $d(x|m)$ shows all outcomes random with respect to the universal distribution (see discussion in section 8.2). If the real distribution is $\mu(x)$, it follows that $m(x)$ and $\mu(x)$ are sufficiently close to each other for $m(x)$ to be used instead of $\mu(x)$. Hence $m(x)$ can be used in hypothesis testing where the true distribution is unknown. In other words, the universal distribution as a prior is almost as good as the exact distribution. This justifies the use of the universal distribution as the prior to plug into Bayes' rule, provided that the true distribution over all hypotheses is computable. Effectively, as Li and Vitányi [76] point out on p 323 that this process satisfies both Occam's razor and Epicurus' principle. Occam's razor states that entities should not be multiplied beyond necessity; effectively saying that simpler explanations are best. On the other hand, Epicurus had the view that if several theories are consistent with the data they should all be kept (Li and Vitányi [76] page 315). Because any hypothesis with probability greater than zero is included in the universal distribution as a prior, and the approach gives greater weight to the simplest explanation, the approach satisfies both Occam and Epicurus.

Solomonoff's prime interest was in predicting the next member of a possibly

infinite sequence (See Li and Vitányi [76] for a reference to the underlying history). However infinite sequences require a modified approach which will be schematically outlined here for completeness. Nevertheless, first time readers might find it convenient to ignore the following paragraphs.

In the infinite case, an output is not just x, but rather the output of all infinite sequences that start with x. This set of all such sequences starting with x is called the cylindrical set (see 2.3) and is denoted by Γ_x. In contrast to the discrete case, as a UTM that outputs the start sequence x with a particular programme may not eventually halt, possible non halting programmes must also be included in the summation defining the semimeasure. As a consequence, the UTM must be a monotone machine; i.e. defined as a UTM that has one way input and output tapes (as well as work tapes). The universal semimeasure for the continuous case is given by;

$$M_U(x) = \sum_{U(p)=x*} 2^{-|p|} \tag{3.22}$$

where M_U denotes that a monotone machine is to be used. Here $x*$ indicates that the output starts with x and the computation may not actually halt. This equation defines $M_U(x)$ as the probability that a sequence starting with x will be the output of a programme p generated by the toss of a coin. The continuous universal semimeasure M has similar but not identical properties to the discrete case. Furthermore,

$$-log_2 M(x) = H(x) - O(1) = Km(x) - O(1),$$

where $Km(x)$ is the monotone algorithmic complexity - i.e. the length of the shortest programme run on a monotone machine that outputs a string starting with x. For example, if the start of a sequence is the string x, the approach would imply that characters in the sequences are distributed according to a computable measure μ. In which case, the probability μ that the next element is a y is

$$\mu(y|x) = \mu(x, y)/\mu(x).$$

The estimate based on the continuous universal semimeasure $M(x)$ is of the same form namely;

$$M(y|x) = M(x, y)/M(x).$$

This converges rapidly to the equation immediately above, and Solomonoff has shown [94] that after n predictions, the expected error by using $M(x)$ instead of $\mu(x)$ falls faster than $1/n$.

Chapter 4

The algorithmic entropy of strings exhibiting order but with variation or noise

As most real world structures exhibit both structure and randomness, if algorithmic information theory is to provide insights into natural systems it must be able to deal with different strings that each capture the structure but also exhibit variations. These strings each exhibit the pattern of the structure and what might be termed noise representing the variations of a particular string (see discussion in section 3.1). For example a string representing the pixels that specify a noisy image will have some pixels that capture the image while other pixels are just noise. As the algorithmic entropy is derived from an algorithm that describes the string exactly, including both the structure and any noise or randomness, it was thought that the random components would dominate the length of the generating algorithm obscuring any pattern [90]. However this is not so. As the following outlines, one needs only to identify the string with its pattern from the set of all similar strings. In this case the contribution of the noise to the algorithmic entropy is equal to the Shannon entropy implied by the uncertainty in identifying a particular string. Three equivalent approaches not only resolve the problem of noisy strings, but also link the algorithmic approach more clearly into that of Shannon's Information Theory and statistical thermodynamics. The three different equivalent formulations are known as;

- the provisional entropy approach,
- the Algorithmic Minimum Sufficient Statistics approach, and
- the ideal form of the Minimum Description level.

The first two are discussed below while the third is discussed in more detail in section 7. Furthermore, Zurek's physical entropy [108], which is a mix of algorithmic entropy and Shannon entropy, returns the same value as the three

equivalent formulations. Although conceptually different, it can be taken to be the same for reasons that become obvious below.

4.1 Provisional entropy

Devine [46] has shown that where pattern in a noisy sequence is recognized there is an implicit reference to the set containing all similar strings exhibiting the pattern. As this set will be recursively enumerable, a noisy string that exhibits this pattern can be generated by an algorithm that consists of two routines. The first routine enumerates the set of strings S exhibiting the common pattern or structure and which contains the string of interest. The second routine identifies the string of interest given the set. The algorithmic measure of the particular string x_i in the set will be termed "the provisional algorithmic entropy" denoted by $H_{prov}(x_i)$. Devine [46] used the phrase "revealed entropy" instead of "provisional entropy" to indicate the value depended on the observed or revealed pattern. The term "provisional entropy" seems preferable, as the observed pattern may not comprehensively account for all the structure in the string. The measure is provisional only, as deeper, unobserved pattern might well exist. The two part description then becomes

$$H_{prov}(x_i) = H_{algo}(S) + H_{algo}(x_i|S) + O(1). \tag{4.1}$$

Here $H_{algo}(S)$ is the length of the shortest self-delimiting programme that describes the set S with its characteristic pattern, while $H_{algo}(x_i|S)$ is the length of the shortest self-delimiting programme that picks out x_i in the set. The contribution of the second term to the algorithmic entropy is in practice the same as the Shannon entropy of the set. If there are Ω_S elements in the set, and all members are equally likely, each string can be identified by a code of length $|\Omega|$. In which case, ignoring the extra contribution due to self-delimiting coding $H_{algo}(x_i|S) = log_2\Omega_s$. This is of course the Shannon entropy of the set. The provisional entropy then becomes;

$$H_{prov}(x_i) = H_{algo}(S) + \log_2\Omega_S + O(1). \tag{4.2}$$

As was discussed in section 3.3, if the information required for self-delimiting coding can be built into the algorithm that defines the members of the set, it can then be fed into the algorithm that identifies the actual string in the set. In effect;

$$\begin{aligned} H_{prov}(x_i) \quad &= H_{algo}(description\ of\ the\ set's\ structure) \\ &+H_{algo}(identifying\ the\ string\ in\ the\ set). \end{aligned} \tag{4.3}$$

The first contribution is the entropy of the algorithm that describes the set, wwhile the second contribution is equivalent to the Shannon entropy of the set. In which case, equation (4.2) becomes

$$H_{prov}(s_j) \simeq H(S) + H_{sh} \tag{4.4}$$

Just as the Shannon entropy is a measure of the uncertainty in identifying each member of the set, the equivalent provisional entropy term requires the same number of bits as the Shannon measure to identify the particular member x of the set given the set's characteristics. In other words, the provisional entropy is the entropy of a typical member of the patterned set.

As the upper measure of the algorithmic entropy, given the available information, is the provisional entropy, the provisional entropy is the best estimate of the actual entropy of a particular noisy patterned string representing for example, the microstate of a physical system. If all the structure is identified, the provisional algorithmic entropy will correspond to the true algorithmic entropy, resolving the problem raised in section 3.1. Nevertheless, as there is no certainty that all the structure has been identified, the minimum description $H(x_i) \leq H_{prov}(x_i)$. Also, even if all the structure of a typical string has been identified, there will be a few highly ordered non-typical strings that might show more structure. The algorithmic entropy is also less for these. The name "provisional" indicates that unrecognised structure may exist. Nevertheless, the discussion shows the concerns about the algorithmic approach, raised in section 1.2 are unwarranted. While the algorithm exactly specifies a particular string, the variation or noise does not swamp the algorithm. Zurek's physical entropy, which is discussed in section 5.2, includes the shortest algorithmic description of the state given limited information, plus a Shannon term to specify the remaining uncertainty. In contrast to the above discussion, the Shannon term of the physical entropy is not derived from an algorithm. While the physical entropy is not a complete algorithmic measure, its value is the same as the provisional entropy as the algorithmic measure that identifies the string in the set is virtually the same value as the Shannon entropy.

Coding the jth string in the set

The two part algorithm that defines the provisional entropy requires a routine that identifies the string of interest in a set of similar strings exhibiting the same pattern or structure. This section looks at the details of how to specify a particular string to make the procedure more concrete.

In order to identify the string within the set of similar strings, the algorithmic code for the particular string must be able to be decoded. For simplicity, assume that all the Ω_s strings in the set exhibiting the pattern have the same length. Each can be placed in lexicographic order and assigned a code depending on that order. If all strings occur with probability $1/(\Omega_s)$, from equation (3.12), the minimal code length of the jth string will be $|code(j)| = \lceil log_2 \Omega_s \rceil$. This length will be taken to be N. The practicalities of this are seen in the simple example where there are, say, 220 members of the set. In which case, $log_2 220 = 7.78$ showing that all the strings in the set can be identified by an 8 digit code ranging between 00000000 for the first member to 11011011 for the 220th member (i.e. the binary of 220-1). In general, the jth sequence in the ordered list is coded by $00000\ldots0 +$ *binary specification of* $(j-1)$. While none of these codes are a prefix of another, as was discussed earlier, information

about both N and the size of N is needed for the algorithm to determine when all the characters have been read. The algorithm that generates a particular string will need to include the following.

- The code $code(s_j)$ the code for the jth string. This will have length $N = |\Omega_s|$, which is close to $log_2\Omega_s$. This code is key to the subroutine that identifies the string j given the set characteristics.
- The algorithm also must have implicit or explicit information about the length, of $code(s_j)$ to know when the code has been read. This is the information given by $H(N)$ which specifies the size of N. Either the length information must be associated with $code(s_j)$ so that the self-delimiting code length is $H(N) + |code(j)|$, or be part of an instruction to indicate how much code is to be read. The latter is the case in the algorithm below. A code of length $H(N)$ provides sufficient information for the computer to know when $code(s_j)$ has been read.
- The programme outlined below needs to start with an initial string 0000 ... 00 of length N that is stored in the computer. In algorithmic terms $H(N)$ is needed to specify the $STORE$ routine at the start of the programme. I.e. $|STORE(\text{'0'}, N)| = |0| + |N| + O(1)$.
- A specification for the set of patterned strings which for convenience can be taken to the subroutine 'CRITERIA', with an output 'YES' if a string is a member of the set and 'NO' otherwise. In effect $H(S) = |CRITERIA|$

The programme that generates the jth string takes the following form..

$$
\begin{aligned}
&INPUT\ H(N)\ \{specifies\ the\ number\ of\ bits\ needed\ to\ read\ code(j)\}\\
&INPUT\ code(s_j)\ \{H(N)\ bits\ long\}\\
&\qquad s = STORE(0, N)\ \{creates\ a\ string\ of\ N\ zeros.\}\\
&\qquad FOR\ I = 0\ to\ code(s_j)\\
&A.\quad GOSUB\ CRITERIA\ \{Output\ =\ NO\ or\ YES\}\\
&1\quad NO \quad s = s + 1\\
&\qquad Go\ to\ A\\
&\qquad YES \quad s = s + 1\\
&\qquad NEXT\ I\\
&PRINT \quad s - 1\ \{Needed\ as\ the\ loop\ has\ stepped\ too\ far\}
\end{aligned}
\tag{4.5}
$$

Step A specifies the criteria that define the patterned set. This algorithm steps through all possible sequences, s; counts all those that fit the criteria. I.e. increments i until the jth patterned string is reached. As a consequence, the provisional algorithmic entropy of the string can be expressed as:

$$H_{prov}(s_j) = |code(j)| + H(N) + |criteria\ \ defining\ \ patterned\ \ set| + O(1).$$

Where the number of members of the set is large, this reduces to;

$$H_{prov} \simeq log_2\Omega_s + log_2 N + (log_2 log_2 N \ldots) + |0| + |criteria\ \ defining\ \ patterned\ \ set| + O(1).$$

Here the O(1) terms include standard instructions like STORE, PRINT, GOSUB etc. If necessary, the zero of entropy can be chosen to take these as given. The $log_2 log_2 N$ term is included as some information about the length of $H(N)$ probably is needed. In more formal terms, the algorithmic entropy of string s_j can be expressed as;

$$H_{prov}(s_j) = H(s_j|S) + H(S) \tag{4.6}$$

where the first term identifies s_j by its position in the set, while the second, term $H(S)$, is the length of the subroutine that characterises the set's pattern or structure. If the set represents strings generated by a model, this uses the model's parameters to specify the set members.

When $\Omega_s >> H(S)$, i.e. Ω is large relative to the definition of the set, $H_{prov} \simeq H_{sh}$. For the opposite and trivial case, when the set has only one or two members, the $log_2 N$ term becomes the main entropy term as is discussed in section 4.3.

A simple example

Devine [46] has used the above approach to determine the algorithmic entropy of a set of finite sequences where the 1's and 0's occur with different probabilities. Such a sequence can be envisaged as representing the microstate of a physical system at a given temperature. One example might be a system of N two level atoms where only some atoms are in the excited state or alternatively a system of N spins where, some spins are aligned with a small magnetic field and others are not. In the first case a '0' indicates that an atom in the system is in the ground state and a '1' that it is an excited state. If the system is isolated the number of 1's is a constant of the motion as energy is conserved. This example, while in some senses trivial, illustrates the key points. The microstate of the system can be specified by a string of length N with R ones and $(N - R)$ zeros. These form the hypergeometric sequence (see Jaynes p 69 [18]) generated by drawing a finite number of zeros and ones from an urn until the urn is empty. If the urn has N objects made up of R ones, the set of the sequences with different arrangements of R ones and $(N - R)$ zeros has $\Omega_s = N!/[(N - R)!R!]$ members. The provisional algorithmic entropy of the strings in the patterned set, which correspond to different microstates of the system, can be established by method outlined in the previous section. Let $COUNT(S)$ be the subroutine that counts the number of ones in the string S. If $COUNT(S) \neq R$, the string S is not a member of the patterned set. The subroutine "CRITERIA" (4.5) becomes

$$\begin{aligned} &CRITERIA \\ &IF\ COUNT(S) \neq R,\ (I.e. does\ not\ have\ the\ desired\ number\ of 1's) \\ &NO,\ \ ELSE\ \ YES. \end{aligned} \tag{4.7}$$

With this membership criterion, and ignoring the log_2log_2 and higher terms for $|R|$ and $|N|$, the relevant entropy for a microstate is: $H_{prov} \approx log_2\Omega_s + log_2N + log_2R$. Effectively log_2N bits are required to specify the length of the string and to set all cells to zero; log_2R bits are required to specify the number of 1's, and $log_2\Omega_s = log_2(N!/[(N-R)!R!])$ bits are required to specify the position of each of the R ones in the configuration. If all the atoms or spins were in the ground state, effectively at zero temperature, the algorithmic entropy would be $\approx log_2N$. The increase in algorithmic entropy due to the rise in temperature is $log_2\Omega_s + log_2R$. The increase is due to an increase in uncertainty and the need to specify the pattern.

However, for large R, given N, rather than specify R directly, it may be algorithmically shorter to calculate R from R/N with a short algorithm if R/N is a simple ratio such as $\frac{3}{4}$. In this simple case, as given N, $log_2R = log_2(\frac{3}{4}) + log_2N$, few bits are required to specify the algorithm that divides 3 by 4. On the other hand for large R a significant number of bits are required to specify R directly. When a short specification happens it can be seen as a form of accidental ordering, i.e. an ordered fluctuation away from a typical configuration. However, as energy or bits must be conserved in a natural system, this can only happen if energy is lost or transferred another part of the system. In other words if the simple system described, is a natural system, it cannot exist in complete isolation from the vibrational or momentum states which are essential to fully describe the system. Later in section 9 this issue is dealt with in more detail.

4.2 Algorithmic Minimum Statistics Approach and the provisional entropy

As has been discussed, the provisional entropy is the algorithmic entropy of a typical string in a set of similar strings. Earlier Kolmogorov introduced the algorithmic equivalent of Fisher's minimum sufficient statistic concept known as the Kolmogorov Minimum Sufficient Statistic, or the Algorithmic Minimum Sufficient Statistic (AMSS). This turns out to be identical to the provisional entropy. The approach, which is implicit in section 2.1.1 in Li and Vitányi [76], seeks the smallest set in which the string of interest is a typical member. If the smallest set captures all the regularity in the string, the shortest algorithmic description of the set together with the algorithm that identifies the string in the set should correspond to the algorithmic entropy. The set that satisfies this is also the set that gives the the provisional entropy. However one can never be sure that the minimal set has been found.

Prior to the work of Devine [46]on the entropy of patterned noisy sets, Vereshchagin and Vitányi [103] developed the AMSS approach to specify the shortest description of a string by optimizing both the length of the programme to generate the string and the description of the Turing Machine running the programme. The approach optimises the combination of the Turing Machine T_i over all machines i where $T_i(p) = x$ and the programme p. In which case

(see section 3.8);

$$H(x) = min_{i,p}\{H(T_i) + |p|\} + O(1).$$

There are many Turing Machines that can generate a string x. In order to describe a string with randomness and structure there is a trade-off between a Turing Machine that has a short description, but which requires a longer programme $|p|$, and a machine with a short programme and a long description. The optimum occurs when all the pattern or structure is included in the description of the Turing Machine and only the variation or random part, is captured by the programme p. Again the shortest description of a particular string involves finding the optimum set where the string is a typical member; i.e. relative to other members in the set it is random. As a Turing Machine is a function, it can be considered as a model that captures the essential structure or pattern in the observed string. Alternatively the characteristics of the model can be simulated by a programme on a UTM. In which case the programme captures the Turing Machine instructions that embody the model, while the remainder, the random part is captured by a code that identifies the string of interest. If the description of the model is too sophisticated, it will over fit the data, but the reduction in the programme length $|p|$ is insufficient to compensate for the increase the length of the model description. Alternatively if model is too simple, the size of the programme $|p|$ will need to increase to provide the detailed corrections. For a given x, the optimum is found by (4.8) below. This is identical to equation 4.1

In the language of sets, the model defines a set of string exhibiting a common structure or pattern. All the structural information in the string is captured by the algorithm that defines the set. Once this is done, the particular string can then be specified by the algorithm that generates the set of strings, coupled with an algorithm that identifies the particular string in the set. The algorithmic complexity of the string x_i is [103]

$$H(x_i) \leq H(x_i|S) + H(S) + O(1). \tag{4.8}$$

Similarly from a provisional entropy point of view, $H(x_i) \leq H_{prov}(x_i)$ where $H_{prov}(x_i)$ equals the right hand side of equation 4.8. When the equals sign applies, $H(x_i)$ is the Algorithmic Minimumm Sufficient Statistic. It is a minimum statistic, as for a typical member of the set all the information has been captured and no further pattern is detectable in the string. In which case, if this set S is given, and Ω_S denotes the number of members of the set, $H(x_i|S) \leq log_2\Omega_S + O(1)$. $H(x_i|S)$, the length of the description of x_i given S, is the length of the algorithm that identifies the particular x_i within the set. A typical random member of the set will be within a constant c of the $log_2\Omega_S$, i.e. $H(x_i|S) \geq log_2\Omega_S - c$. Just as for the provisional entropy (4.6) the string is defined by a two part algorithm. In the AMSS case, the first describes the structure of the optimal set - the simplest set in which the string is random with respect to other members. The second part identifies which string is the string of interest. This term captures the level of uncertainty arising from the

data. In general, the algorithmic sufficient statistic of a representative member of this optimal set S is.

$$H(x_i) = H(S) + log_2\Omega_S + O(1).$$

$H(S)$ represents the description of the set and $log_2\Omega_S$ is the length of the algorithmic description of a typical member of the set and returns virtually the same value as Shannon entropy measure of the uncertainty implied by the number of members of the set. While the main mathematical discussion in the literature uses the term AMSS to define the optimum set in order to define the shortest algorithm, here the word provisional entropy will be more commonly used as it links more easily into the entropy concepts of the natural sciences.

4.3 The provisional entropy and the true algorithmic entropy

There are a number of coding methodologies, such as entropy based coding that can compress a patterned string to provide an upper measure of its algorithmic entropy. The question is how are these methods related to the provisional entropy or the equivalent AMSS approaches. The relationship is perhaps best shown by the simple example outlined below. But to recapitulate, a recursively enumerable set is one where a computable process exists to generate all the elements of the set, such as generating all the odd integers, in order. Equivalently a computable process exists for such a set when one can step through all possible strings and can identify whether a particular string is a member of the set. The routine CRITERIA did this in the example discussed earlier. The example discussed shows how one gets the same result with either process. Furthermore, the results are compared with a process that enumerates the particular string of interest directly without going through the provisional entropy approach. The conclusion is that several algorithms, all of similar length, can generate a particular string. While it is not always clear which is the shortest it usually does not matter for a typical string.

The example to be discussed is the noisy period-2 binary strings consisting of the set of all sequences of length N of the form $s_i = 1y1y1y1 \ldots 1y1y \ldots 1y$ (where y is randomly 0 or 1). The $2^{N/2}$ members of the set give $log_2\Omega_S = N/2$ and the set can be specified by a recursively enumerable process. The provisional entropy or AMSS approach would argue that the entropy is based on the length of the algorithm that defines the set of similar strings plus a term that identifies the string within the set. The following are two different processes to specify the set of all such strings.

- One process to enumerate all the strings of the set in order is to start with $N/2$ zeros and increment the string by one in turn to form the $N/2$ binary strings from $000 \ldots\ldots 00$ to $111 \ldots\ldots 11$. At the start, and each increment thereafter, $N/2$ 1's can be dovetailed between each binary digit to generate the constant part of the noisy period-2 binary string.

This algorithm needs to specify $N/2$ zeros which takes $log_2(N/2)$ bits, the small stepping and dovetailing routine and finally requires $N/2$ bits to specify the string of interest, the ith string in the set. Ignoring the stepping routine the provisional entropy is,

$$H_{prov}(s_i) \simeq N/2 + log_2(N/2) + |0| + |1|.$$

- Alternatively if a checking algorithm is used to identify which strings are in the set and then specifies the ith string in the set, the algorithm that defines the set's members will be $\approx log_2(N/2)$ as it will need to read $N/2$ characters to determine whether the first member of each pair is a '1. This is the requirement that defines the set. A further $N/2$ bits are required to identify the ith string in the $2^{n/2}$ strings. The result is the same as before if the small details of the algorithm that specifies whether a string is in the set or not is ignored.

However, rather than using the provisional entropy approach, it is possible to have an algorithm that directly specifies the string. A particular string in this noisy, period-2 sequence, can be coded by replacing each '11' in the noisy period-2 string by a '1' and each '10' by a '0'. For example 11101011 ... becomes 1001 .... These codes are self-delimiting with respect to each other and the coded string is half the length of the original string. However this compressed coded string is, by itself, not an algorithmic measure, as the code does not generate the original string. A short decoding routine needs to be appended to the code to generate the original string. Additionally the requirement that the codes must be self-delimiting relative to the decoding instructions as well as with respect to each other will add another $\sim log_2 N$ In which case the relevant algorithmic entropy based on this coding is,

$$H(s_i) \simeq |code(s)| + |decoding\ routine| + O(1).$$

In this case, the decoding routine scans the coded string and replaces a '1' by a '11', and a '0' by a '10'. In order to be self-delimiting, the overall specification of the string s must contain information about the length of $code(s)$. This can be done by either including it in the code, giving $|code(s)| = N/2 + |N/2|$ etc., or including $\sim |N/2|$ in the decoding routine so that the routine knows when to finish reading the code. For large N, and ignoring the details of the O(1) decoding routine the result is virtually identical to the previous approach. I.e.

$$H_{prov}(s_i) \simeq N/2 + log_2(N/2) + |1| + |0|. \tag{4.9}$$

The point to note is that several algorithms can generate a particular string with the required characteristics. While it is not obvious which of these algorithms is the shortest, and therefore which gives the true algorithmic entropy, the dominant terms are the same, the difference lies in relatively short algorithms that specify stepping, checking or decoding routines. Chaitin has argued that there is a cluster of short description of similar lengths that dominate all the algorithms that can generate a particular string.

Nevertheless, the above equation assumes there is no hidden structure. If further structure is identified, then the description of the set is not optimal. For example in the set of period-2 sequence, there are strings such as the string $s'_i = 101010\ldots10$ which are non-typical as a shorter descriptions exists. In this case the true algorithmic entropy will be less than the AMSS measure or the provisional entropy. In particular $H_{algo}(s'_i) \simeq log_2 N$ corresponding to "$PRINT$ 10 $N/2$ times ". In other words, for a non typical member of the set, $H_{algo}(s'_i) \leq H_{prov}(s'_i)$. The first term in equation 4.9 becomes unnecessary indicating that, as this string is not typical, the set or the model needs to be refined. For this reason, the entropy measure is termed "provisional, as discussed in the previous section, more information may indicate the optimal set is smaller, leading to a lower algorithmic entropy. Nevertheless, in many physical situations, the overwhelming majority of members of a set will be typical.

Just to reiterate, as equation 4.3 indicates, the algorithmic entropy of most strings in a patterned set is given by the length of the algorithm that defines the set of all binary strings exhibiting the pattern, combined with the length of the algorithm that identifies the string of interest in the set.

The AMSS or provisional entropy approach provides the shortest algorithm for the typical member of the set of similar strings. The typical member is random relative to the set of similar strings. There are implications for this in establishing the entropy of a physical system. In a physical system there may be configurations that are accidentally ordered yet which still satisfy the basic criteria defining the system, such as the number of particles and the total energy and physical constraints. These accidental orderings are rare fluctuations from the typical configuration, identical to the accidental fluctuations away from equilibrium in a thermodynamic system. For example the instantaneous configuration of the particles in a Boltzmann gas might be uniformly distributed over space. This shows structure or order and is something equivalent to a string such $1010101\ldots10$. In these accidentally ordered cases, the true entropy of an actually observed configuration would be lower than the typical provisional entropy of the majority of states specifying the particle positions. However as is discussed later, for real thermodynamic systems, in contrast to mathematical strings with no physical significance, when such an ordered fluctuation occurs in one part of the system, such as the position states, in order to conserve energy, entropy must shift to the momentum states of the system.

4.4 The specification in terms of a probability distribution

The provisional entropy and AMSS approach outlined above specified the algorithmic complexity in terms of the algorithm that described the optimal set containing the string of interest, together with an algorithm that specified the string in that set. This is appropriate where all members of the set are equally likely. Such an approach is appropriate to specify a string representing a con-

figuration of a real world system. For example an isolated gaseous laser system might contain lasing atoms in an excited or ground state, incoherent and coherent photons and momentum states capturing the motion of the particles. All the possible configurations appropriate to a given energy establish the set of possible states. As each of the possible configurations in the macrostate is equally likely, the actual state requires the set to be defined first and given the set the Shannon entropy term which identifies the particular configuration in the set.

However for the situation where all strings are not equally likely and the set of strings can be defined by a probability distribution (such as that generated by a Bernoulli process) the characteristics of the set can be specified by the probability distribution of the members. However if a set is to be defined in terms of a probability distribution, the distribution must be computable to an agreed level of precision. In this case Shannon's coding theorem can specify each string in the distribution by a unique code based on the probability value for each data point in the distribution.

Let $H(P)$ represent the length of the shortest programme that computes a distribution on the reference UTM. The algorithmic entropy of a particular string x_i that belongs to the distribution can now be specified by a two part code. The first part is the description that captures the structure of the set of possible strings specified in terms of a probability P, while the second part that identifies the actual string given the distribution. Hence

$$H(x_i) \leq H(P) + H(x_i|P^*). \tag{4.10}$$

An equals sign in equation 4.10 holds for the overwhelming majority of strings in the distribution, but a few strings will have a shorter description. For example in a Bernoulli process where 1's are emitted with a probability of p and 0's emitted with a probability $1-p$, the majority of strings of length n, consisting of a mix of r 1's and $n-r$ 0's, will occur in the set with the probability distribution $p^r(1-p)^{n-r}$. In which case an equals sign is appropriate. However a few non-typical strings, such as those where 1's take up the first r places in the string, can be algorithmically compressed further than implied by equation 4.10 and an alternative distribution or algorithm is needed. The right hand side of this equation is also the provisional entropy or the Algorithmic Minimum Sufficient Statistic for the string.

According to Shannon's coding theorem (equation (3.12)), each member of the distribution can be optimally represented by a code such that $|code(x_j)| = -\lceil log_2 P(x_i) \rceil$, where the ceiling function indicates the value is to be rounded up. Rather than continually using the ceiling notation, as is the custom, this integer will be represented by $-log_2 P(x_i)$ recognising that the code length must be an integer. With this in mind

$$H(x_i) = -log_2 P(x_i) + H(P) + O(1), \tag{4.11}$$

for a typical member of the set. Also, the $O(1)$ includes any decoding routine if it is not explicit in the algorithm. The expected value of $H(x_i)$ (i.e.

$\langle H_{algo}(x_i)\rangle)$, given this distribution, is found by multiplying each term in equation (4.11) by the probability of x_i and summing over all states. As the overwhelming majority of contributions to the sum have the equals sign (s in (4.11)) and those that do not have very low probabilities, the expected code length is close the Shannon entropy, H_{sh} as $H(P)$ is small. This is discussed in more detail in section 5.2. In other words,

$$\langle H_{algo}(x_i)\rangle = H_{sh}(set\ of\ all\ states) + H(P) + O(1) \tag{4.12}$$

The $O(1)$ decoding routine is implicit in the algorithm that generates the string. Given the code for x_i, where $|code(x_i)| = -log_2 P(x_i)$ the decoding procedure steps through all the strings with the same probabilities in order, until the x_i associated with the correct code is reached. However in practice, codes and strings must be ordered consistently. For example, those strings with the same number of zeros and ones in a Binomial distribution will have the same probability. If codes and strings are ordered lexicographically for the same probability, the fifth string, in lexicographic order is specified by the fifth code, in lexicographic order associated with that probability. Strings can be unambiguously coded and decoded, as there are always sufficient codes of a given length to code all the strings with the same probability [1]

How to specify noisy data

The provisional entropy or the Algorithmic Minimum Sufficient Statistic approach showed how to find an algorithmically compressed specification of a particular string with recognisable structure but also with random components by a two part algorithm. The first part specified the set of all strings with the same pattern or structure and, the second part given the set, specified the string of interest.

Following this, section 4.4 extended the process to one where the first part specified the set in terms of an enumerable probability, and the second part specified the particular string by a probability based code. This is distinct from the first approach where the string itself embodied noise or random parts. The probability approach is suitable for the situation where the observed data is noisy. I.e. one needs to find an algorithm that specifies a string using a model or physical laws and then specifies the exact string and its deviations from the model by this enumerable probability distribution. in the probability formulation the noise is the deviation from the ideal, rather than being embodied in the string as in the first case.

Model examples

An example in two dimensional space might be the straight line model that optimally fits a noisy set of data described by the equation $y_i = ax_i + b + noise =$

[1] This works because the probability has been defined to a specific level of precision. If the level of precision is increased, the length of the $log(probability)$ increases and more codes become available.

$y(x_i) + noize$. Where the noise can specified by a probability distribution P, such as the normal distrbution, $y_i = y(x_i) + P(y_i - y(x_i))$. In order to specify a particular string in the set of strings using an algorithm one needs to recognise that the deviation from the prediction of the model is just $y_i - y(x_i)$, the probability distribution. From Shannon's Coding Theorem, the deviation of a particular string from the model can be optimally coded by $|code(y_i - y(x_i))| = -log_2 P(y_i - y(x_i))$. I.e. given this code the deviation can be generated from its code as outlined in section 4.4.

In the straight line example outlined, the provisional entropy for a particular string y_i given x_i, is the length of the algorithm that defines the string from its equation and the associated probability distribution, plus a code that specifies the deviation of that string from the equation. The result is

$$H_{prov}(y_i) = H[ax_i + b, P] - log_2 P[y_i - y(x_i)] + |decoding\ routine| + O(1),$$

or

$$H_{prov}(y_i) = H[ax_i + b, P] + |code(y_i - y(x_i))| + |decoding\ routine| + O(1).$$

Here the O(1) decoding routine is specifically identified. The first term captures both the shortest description of the model and its associated enumerable probability distribution, while the second term specifies the deviations from the exact model following the arguments of section 4.4. The expectation value for the second term is the Shannon entropy of the deviations.

This approach effectively is selecting the Algorithmic Minimum Sufficient Statistic (AMSS)for the string y_i or the best estimate of the provisional entropy. It is closely related to the Minimum Description Level approach discussed later in the section 7 as the ideal form of MDL gives gives the same model for noisy data. The algorithm that generates point y_i given point x_i contains the probability based code, a decoding routine, the probability distribution needed to calculate the deviation from the model, and finally the specification of the model. The algorithm undertakes the following computational steps;

- Takes the code and knowing that $|code(y_i - y(x_i))| = -log_2 P(y_i - y(x_i))$ generates the value of the probability for that string;
- Calculates $y_i - y(x_i)$ from the probability values using the routine for P;
- Generates $y(x_i)$ from $ax_i + b$;
- Adds $y_i - y(x_i)$ to $y(x_i)$ from the second bullet point above and generates the specific y_i as required.

A more general example

Instead of listing the earth's position for every second of a year, Newton's laws can capture the motion in a set of equations. If the equations exactly

described the earth's motion around the sun, the equations represent an extremely compressed version of the data. However the capture is not exact, as variations occur. The variations from the model might be due to measurement error, perturbations from the moon or a breakdown in the assumptions of simple spherical bodies etc. Where the deviations appear as noise they can be represented by a probability distribution. In this situation the observational data can still be compressed by describing the data algorithmically - initially in terms of Newton's laws (which specifies the particular string should Newton's laws exactly predict the data point) and then by coding how the observed data point deviates from the simple predictions of Newton's laws. If later a further cause or pattern in the variation is uncovered, as might occur by taking into account the moon's effect, the compression can be improved.

A good model is one that has maximally compressed the data. All the regularity is in the model while, the variation (which may be due to the limits of the model, noise, or experimental error) will appear as random deviations from the model. This again is the argument used to find the Algorithmic Minimum Sufficient Statistic for the data discussed in section 4.2 but where the model is captured in physical laws, and the randomness captured by the code that identifies the particular string given the laws.

One would expect this best fit to a series of data points to be determined by minimising the deviations from the model for each point y_i.

Model selection

The previous two sections show how to specify a string representing a data point given an understanding of the laws which generate the string and the deviation from the laws. Vereshchagin and Vitányi [103] apply the ideas to provide a comprehensive approach to selecting the best model that describes a particular string or strings based on the AMSS approach of Kolmogorov. As Vereshchagin and Vitányi [103] point out "the main task for statistical inference and learning theory is to distil the meaningful information in the data".

In a nutshell, given a set of data, the best model of that data is the one that captures all the useful information. In general, a model, based on natural laws, provides the best compression of the data at hand. It follows that one can try explaining the data with different models and select the one with the minimal description as the best explanation. Hence for each credible model of the data a two part algorithm can be used to define each string. The first part uses the model and the second part codes the deviation from the model. The model with the shortest overall description is the preferred model. The algorithm describing a data string using the optimum model for that string is the Algorithmic Minimum Sufficient Statistic as all the useful information is contained in the algorithm. As Vereshchagin and Vitányi [103] explain; "For every data item, and every complexity level, minimizing a two part code, one part model description and one part data-to- model, over the class of models of at most given complexity, with certainty selects models that are in a rigorous sense are the best explanations of the data". The two part code that provides

the shortest algorithmic description of the model, plus the deviations from the model is preferred. In other words it is a short step from finding the shortest description of a string to finding the best model that models a set of strings.

With this in mind, the algorithm that generates a typical string in a set of strings defined by a model and the deviations from the model, requires the following components.

- The description of the model, which in the straight line model described above requires $2d$ bits to specify a and b to a precision d.
- The description of the deviation from the model. I.e. given an input x_i, the deviation is $y_i - y(x_i)$, where y_i, is the predicted value for a perfect model. The deviation $y_i - y(x_i)$ is distributed according to a probability model. According to Shannon's Coding Theorem, the deviation can be optimally coded by a code of length $= -log_2 P[y_i - y(x_i)]$. For a Gaussian distribution the log probability term becomes $(y - y(x_i)]^2/(2\sigma)^2 - log_2(\sigma\sqrt{2\pi})$ noting the standard deviation, σ is determined from the data. If the outcome was not Gaussian or another symmetrical probability distribution, the implication is that hidden structure exists and the deviations would not be random. In which case, a more refined model would be needed.

It should be noted that the parameters a and b, and the characteristics of the distribution are not determined by the model selection process. They must arise from other processes that determine the best model. In the simple straight line case they might be determined by a fitting procedure; for example, one that minimises the sum of the squared deviations. Consider the situation where it is suspected that the straight line model is the best explanation of a set of data. As is discussed later in section 7, if a 2nd degree polynomial provides a better fit than the straight line, this would be because the increase in the model description to three parameters is offset by a reduction in the deviations from the model. In the 3 parameter case an extra d bits are required to describe the model ($3d$ bits in all) and this must be offset by a decrease in the variable part of the log probability term, i.e. $(y_i - y(x_i))^2/(2\sigma)^2$. The limitations of the algorithmic approach can now be seen. The best parameters for each model must first be determined before the models can be compared. However once the best model is found by whatever means, the provisional algorithmic entropy based on the model is the best estimate of the entropy of each string.

The model selection approach is identical to the ideal Minimum Description Level approach discussed later in section 7. It is only valid if the string of interest or the strings of interest are typical strings in the two part description. They are random in the set and have no distinguishing features. If this is not so, as was discussed in section 4.4, another probability distribution may be better.

One further point, the model obtained by minimising the two part description is the best explanation of the data, however it is not necessarily the correct explanation.

Chapter 5

Order and entropy

5.1 The meaning of order

In principle it is possible to extract work from two physical systems, or two identifiable parts of one physical system, provided one is more ordered than the other. In doing so, the total disorder of the universe increases. However in order to extract work, the observer must be able to recognise the order and its nature so that it can be usefully harnessed. A set of ordered states is a subset of all the possible ordered and disordered states of a system. In general it is only where members of a subset exhibit a common pattern that the order can be recognised and described. To an alien, one string of human DNA would show no obvious pattern as the chemical sequence would be seen as just one of the myriad possibilities. While and intelligent alien might have the capability to undertake a randomness test (section 8.2) to determine whether the discovered DNA string does show order, this is unlikely to work as the DNA string may be maximally compressed. The surest way for the alien to identify a pattern is by observing the set of DNA from a variety of humans and noting there is much in common.

In general, as section 8.2 argues, the degree of order, or the lack of randomness, is difficult to quantify. Nevertheless, an ordered sequence can be recognised as exhibiting a pattern because our minds, like that of the alien, perceive that the sequence belongs to a subset of similar patterned sequences. Our minds see this subset is distinct from the set of apparently ordinary or random strings. If the pattern is recognised, any ordered sequence s of the subset can be compressed i.e. be described by an algorithm much shorter than the sequence length. Thus any common pattern implies that an algorithm s^* exists which can specify the patterned string and for which $|s^*| << |s|$. Once such a pattern is determined empirically by observation or some trial process, the fact that no computational procedure exists to determine the pattern becomes irrelevant - it can be compressed (see the comments of [91] and [42].

A simple example illustrates the argument. Consider the set of all strings of length N of the form $1y1y1y \ldots 1y1y$ where y is randomly either '0' or '1'.

As the set has a recognisable pattern a sequence from the set is able to be compressed into a shorter description. This contrasts with a general sequence of length N, where no order is discernible and there is no certainty that any order exists. As the typical member s of the full set will not be able to be compressed, in general $|s| \approx$ the length of the string. In more formal terms, any patterned set is a subset of all possible strings.

Devine (2006) argues that the use of AIT framework to specify the algorithmic complexity of patterned approach has some advantages. Also, it would appear to be applicable, at least in principle, to systems that are not just simple patterns with noise. Indeed, if AIT concepts can be extended, they offer some interesting potential applications.

For example, a particular plant at an instant of time is specified by an algorithm based on its DNA, and the instruction set encoding the physical and biochemical laws (as is discussed in more detail in section 6.3). The inputs to the algorithm are the accessible nutrients and the impacts from the external environment that determine the plant's growth trajectory. The algorithmic entropy describing the plant's configuration at an instant of time is generated by an algorithm that halts at that instant. As time flows the same algorithm, halting at a later time, generates a more biologically complex pattern. The algorithm that describes the plant growth is copied in each cell. The inputs to each cellular algorithm are the chemical outputs of the neighbouring cells, neighbouring structures and the external environment mediated by neighbouring cells.

A biological cluster of cells would appear to be something like an attractor in a state space appropriate to an extremely complicated non-linear system as section 9.6 outlines. The components of this cluster maintain homeostasis as the whole is maintained in a stable region by balancing interactions between the cells. Where homeostasis cannot be maintained, the algorithm terminates, causing death; perhaps because of an external input shock; perhaps because the algorithm gets corrupted by repeated copying; or perhaps because resource or physical limits constrain the continued reproduction of the cellular algorithm.

The set of all cells in a plant or animal have an underlying pattern and a variation. For example, while a cell in an eye is different from one in a muscle, it is specified by the same DNA and differs only because of different environmental inputs. In principle, two algorithmic approaches to specify different cell types are possible. The first might specify different cell types by a string from the set containing the different variations of the basic cell pattern. The second might specify instead the algorithm that generates the cells from the seed DNA but which also includes information on the mechanism that determines the genes that are expressed for the different cell types. In a subsequent section it is argued that, once everything is taken into account, the shortest algorithm is the one that generates the plant from the seed. Nevertheless, which ever approach gives the shorter algorithm, AIT can be used to investigate any biologically complex system (i.e. an organised complex system) at different levels of scale and, at each nested level, only the relevant patterned set needs to be explained. Such an approach would be consistent with the top down approach of Beer's

Viable Systems Model ([10, 9]) which is a framework for probing organised systems in terms of nested subsystems. This approach is expanded in section 9 and tied into Chaitin's concept of d-diameter complexity [28].

Whether there is any possibility of applying this approach to such highly complicated problems is another matter, but for some extremely simple systems progress might be possible. This possibility was illustrated in the previous chapter where algorithmic information is used to specify a member in an ordered set. This shows that the perceived problem with variation, which appears as noise, is avoided (section 3.1).

5.2 Algorithmic Entropy and conventional Entropy

A microstate, which corresponds to a particular configuration in the state space of a thermodynamic ensemble, can be expressed as a string of binary characters. Here the state space refers includes what is called the phase space, the space that specifies the position and momentum of a particle and other states such as electronic, which, in contrast to the phase space coordiantes may be discrete discrete. Where the specification of say a position or the momentum of a particle is not discrete, it is necessary to specify the coordinates to an agreed degree of precision in binary notation. One way to do this is to start with particle 1 and specify its position coordinate along the x to the agreed degree of precision, followed by the same process for the y and the z axes The momentum coordinates of each particle are similarly expressed in turn. The discrete electronic state of the particle can also be expressed by a binary string. Particle 1's position and momentum coordinates are defined in a 6 dimensional phase space. This space which specifies the positions for all N particles is known as Γ space and is made up of 6N dimensions. The string specifying a configuration s_i looks like;

$$s_i = x_1, y_1, z_1, x_2, y_2, z_2 \ldots x_n, y_n, z_{,n}, p_{x1}, p_{y1}, p_{z1}, p_{x2}, p_{y2}, p_{z2}, \ldots\ldots p_{zn}$$

Here x_1 is the x coordinate of atom 1 etc., while p_{x1} is its momentum coordinate in the x direction. The agreed degree of precision for the specification of coordinates is in effect the resolution that imposes a grid or cellular structure on the space. Doubling the resolution by doubling the number of bits specifying each coordinate, increases the number of cells in the Γ space by 2^{6N}. Zurek uses an alternative approach to define the microstate as a string [108]. He divides the $6N$ dimensional Γ space into cells and then argues that the cell size is chosen to be fine enough so that a cell either contains one particle or no particles. The configuration of a particular microstate can be specified by systematically listing the cells in order and denoting an empty cell by a 0 and an occupied cell by a 1. It makes no real difference which way a microstate is specified provided a simple short routine exists to convert one specification to another. This can be done for the two approaches just outlined.

Where there is no shorter algorithmic description of the microstate, the algorithmic entropy is a little more than the length of the description. On the

other hand, the algorithmic description of an ordered state will be shorter as the description can be compressed. Effectively the cellular resolution or grid resolution can be chosen to be physically realistic within certain limits. Strictly the algorithmic description of the microstate in the classical system requires a description of the grid structure and the specification of the microstate within this grid structure. However this is part of the given information, like the physical laws, and cancels for entropy differences. For a quantum system, the resolution of the grid of cells is determined by the uncertainty principle. This can always be taken to be the limiting resolution requirement for the classical case.

The point to note is that the algorithmic approach, like the Boltzmann and Gibb's measures of entropy, also must assume an underlying grid structure. Zurek [108] shows that the ambiguity in the Gibbs entropy, arising from the choice of coarse graining of the phase space, can be absorbed into the background algorithmic description of the system. In other words, the way the phase space or other continuous coordinates are divided into cells is encapsulated in the message about the system. Zurek [108] points out that the overall description must be minimal given a particular UTM. Fanciful grid structures, which complicate the coarse graining concerns of the Gibbs' entropy, require long descriptions and become ineligible. Effectively, as Zurek points out, in the algorithmic approach the complications of the choice of the grid structure is absorbed into the choice of the UTM. In practical terms, because the critical physical parameter is the entropy difference between states, issues to do with the UTM used, or the message about the system (i.e. the common information and the resolution) only shift the algorithmic entropy zero and therefore can be ignored.

Once the specification of the microstate is defined, the length of the shortest self-delimiting algorithm that computes the microstate in the state space coordinates is the algorithmic entropy. It is argued below that the algorithmic entropy of a typical or equilibrium state has a formal similarity to the Shannon entropy, $H_{sh}(\mathcal{E}) = -\sum p_i log_2 p_i$ of the ensemble $\mathcal{E}$. Nevertheless, there is a conceptual difference between the algorithmic entropy and the Shannon entropy. The algorithmic entropy exactly defines a given microstate whereas the Shannon entropy is a measure of the uncertainty in an ensemble of microstates.

There are a number of arguments that clarify the relationship between the Shannon entropy and the algorithmic entropy. The first is more of a "hand waving" argument that provides insights. The second is more rigorous, whereas the third is a detailed algorithmic entropy calculation that shows the algorithmic entropy for a Boltzmann gas is the same as the entropy given by the Sackur-Tetrode equation while the fourth shows the dominant term of the provisional entropy is the Shannon entropy.

- Let the state space coordinate of the ith microstate of a thermodynamic system be specified as a string s_i of length N. Also let s_i^* be the minimal programme that generates string s_i, where s_i occurs with probability p_i in the ensemble. If the expected value of the algorithmic entropy string

s_i in the ensemble be denoted by $\langle|s_i^*|\rangle$, the first argument recognises that the expected value of the algorithmic entropy for all strings s_i $\langle|s_i^*|\rangle = \sum_i p_i|s_i^*|$. This sum can be rearranged in terms of lengths of algorithmic descriptions so that $\langle|s_i^*|\rangle = \sum_l p_l l$ where p_l is the probability of an algorithmic description of length l. In a thermodynamic ensemble, the overwhelming majority of possible states belong to an equilibrium set of configurations. These cannot be compressed significantly. If there are 2^N configurations in the ensemble, most will be expressed by an algorithm of length $log_2(2^N)$. In which case the overwhelming majority of strings in the sum will have a length l close to N. As $\sum_l p_l l$ becomes $l\sum_l p_l$ and $\sum_l p_l = 1$, the result for the expected algorithmic entropy of a string is approximately N. In other words, the algorithmic entropy of most string will be the same value as the Shannon entropy of the ensemble. The Shannon entropy itself is related to the entropy of statistical mechanics S by $S = k_B ln 2 H_{sh}(\mathcal{E})$, which in turn is related to the macroscopic thermodynamic entropy.

- The second argument following Zurek [108] and Bennett [12] (see also [76]) uses the approach of section 4.4 to highlight the relationship between the algorithmic entropy and the Shannon entropy. Let $P(x)$ be the probability distribution of all the states in a thermodynamic ensemble with the Shannon entropy taken as $H_{sh}(\mathcal{E})$. Equation (4.12) shows the algorithmic entropy is no more than $H(P)$ greater than the Shannon entropy. But because Shannon's noiseless coding theorem 3.3 shows that l_i, the code length for each string, can be made to be within one bit of $log_2 p_i$ the expected value of this length, i.e. $\langle H_{algo}(x_i)\rangle$ will at worst be slightly greater than $H_{sh}(\mathcal{E})$ hence $H_{sh}(\mathcal{E}) \leq \langle H_{algo}(x_i)\rangle$. As Bennett [12] argues, the length of the algorithm that calculates the probability distribution given by $H(P)$ is only a few thousand bits, whereas the Shannon entropy is typically extremely large. If the position and momentum coordinates are defined to 8 significant binary digits, this requires 16 bits for each of the 10^{23} particles in a mole of gas. Therefore as $H(P)$ is negligible in comparison, it can be seen that $\langle H_{algo}(x_i)\rangle$, the expected entropy of the microstate, is within O(1) of the the Shannon entropy of the ensemble. For a system in equilibrium, or an isolated system with many states, the typical algorithmic entropy is virtually the same as the Shannon entropy.

- The third argument, developed by Zurek, shows that the algorithmic entropy of particular equilibrium state in an ideal Boltzmann gas of N indistinguishable particles is identical to the entropy provided by the Sackur-Tetrode equation [108]. A simple argument to demonstrate this is to give the address of a particular configuration among the Ω possible configurations. A typical value of the address is $\sim \Omega$. This requires a description of length about $log_2\Omega$ bits. Any configuration is a sequence of 0's and 1's. If there are $\mathcal{C}$ cells in the phase space, there will be $\mathcal{C}^N/N!$ distinct configurations. As the number of cells is given

byt $\mathcal{C} = (V/\Delta V)(\sqrt{mk_BT})/\Delta_p)^3$, the algorithmic specification leads to the Sackur-Tetrode equation. Here m is the particle mass, ΔV is the 3 dimensional cell volume, and Δ_p is the cell volume in momentum space. This argument shows that the algorithmic entropy of a typical state is the same as the entropy of statistical mechanics.

- Section 4 and in particular (4.2) show that the provisional entorpy of a string is given by a two part code. The first defines the structure of the set containing the string and the second identifies the particular string within the set. The latter term is in effect the Shannon entropy of the set of strings. For a theromdynamic macrostate, Where the specification of the set is a short algorithm, the major contributer to the provisional entropy (the best guess of the algorithmic entropy) is the Shannon entropy of the set of strings in the macrostate.

The algorithmic approach can be used to argue that, over time, an isolated system will tend to an equilibrium configuration corresponding to one of the set of most probable states. As is discussed later in section 6.3, Zurek [108] shows that where an isolated system evolves under physical laws, after a large number of steps, but small relative to the Poincaré repeat period $\mathcal{P}$, the trajectory will settle for most of time in the equilibrium region when the states are algorithmically random and a typical description is $H_{algo}(s_k) = |s_k^*| \approx log_2\mathcal{P} = H_{sh}(\mathcal{E})$. This is discussed further in section 6.3.

Boltzmann, Gibbs, Shannon and Algorithmic entropy

The Boltzmann, the Gibbs and the Shannon entropies, are in effect measures of ignorance about the whole system as, in contrast to the algorithmic entropy, they say nothing about the particular state of the system. This section, further discusses the relationship between the Boltzmann, Gibbs and Shannon entropies and the algorithmic entropy. The difference between the algorithmic approach and the statistical mechanics approach is first discussed. Following this discussion, Zurek's concept of physical entropy is outlined in section 5.2. Zurek's entropy measure is a combination of the algorithmic and Shannon entropy that captures both what is known, and what is uncertain, about the microstate of a physical system. Zurek uses this approach to provide a robust understanding of reversibility and work extraction from such a system. However, it is now clear that there is no need for a special definition of Zurek's physical entropy, as it corresponds to the provisional entropy discussed in section 4, [46, 48], or equivalently Gacs' alternative entropy [59] or the Algorithmic Minimum Sufficient Statistic outlined by Vereshchagin and Vitányi [103]. These are all the same. Despite their different labels they are just the best guess algorithmic entropy based on the available information. As a consequence, Zurek's insight into the requirements to extract work from a physical system apply unchanged to the provisional algorithmic entropy (or its equivalents) without the need to define a physical entropy. Furthermore, this whole approach would suggest that the algorithmic entropy is a more fundamental

concept than either the Boltzmann and Gibbs entropies (or the Shannon entropy). Indeed, these entropies, allowing for units, can be seen as the limiting case of an algorithmic description when the there is no information about the exact state, but only about the set of possible states The remainder of this section discusses these points in more detail. However, just to reiterate, the algorithmic entropy measure is a measure of the microstate of the system; it is the shortest description of an actual state. Nevertheless, the expected value of the algorithmic entropy over the whole universe of states is virtually the same as the Shannon entropy and therefore, apart from the units used, is equivalent to the entropy of statistical mechanics. The alignment is not exact as, in contrast to the other entropies, the algorithmic approach must specify the conditions of the system under study. These conditions are understood in the other cases. Nevertheless, as these conditions are common to the specification of the system as a whole, they only shift the entropy zero and have no bearing ontropy differences.

The Boltzmann approach considers a gas of N particles in a container. Boltzmann specified the instantaneous macrostate of the gaseous system by the number of gaseous particles in finite cells in the 6 dimensional position-momentum space; i.e. the system's phase space. A given macrostate of the gas specified in terms of temperature, pressure and volume will encompass a myriad of microstates. The number of microstates is taken to be Ω. Boltzmann describes the statistical entropy using modern notation as $S_B = k_B log_2 \Omega ln 2$. Where the 6 dimensional phase space is divided into m cells with N_i particles in the mth cell, the number of microstates is given by $\Omega = N!/(N_1!N_2!\dots N_m!)$. Using Stirling's approximation this becomes the familiar $S_B = -k_B N \sum_j p_j ln p_j$. Here p_j represents the probability that a single particle is in cell j. For non interacting systems, and a fine grid of cells, this is equivalent to $S_B = -k_B ln \Delta\Gamma$, where $\Delta\Gamma$ is the volume occupied by the macrostate in the 6N dimensional Γ space.

On the other hand, the Gibbs approach envisages an ensemble of replicas of a system all in different microstates and able to exchange energy with each other. The approach identifies one microstate of a system in terms of the $6N$ position-momentum coordinates; the systems Γ space. I.e. the microstate $X = q_{1x}, q_{1y}, q_{1z} \dots q_{nx}, q_{ny}, q_{nz}, p_{1x} \dots p_{nz}$ where the p_{1x}, q_{1x} are the position and momentum vector of particle 1 etc. In which case the Gibbs entropy is

$$S_G = -k_B \int \rho(X) ln[\rho(X)] dX,$$

where ρ represents the probability density of the states X. In practice, the entropy measure is based on a coarse graining of the Γ space into cells of size Δ to avoid the consequences of Liouville's theorem. In this case, the equation becomes a sum over all the coarse grained cells $S_\Delta = -k_B \Sigma_i p_\Delta(i) ln p_\Delta(i)$ and $p_\Delta(i)$ is the average probability that the microstate is in cell i. This contrasts with the Boltzmann measure where the probability measure specifies the probability that a particle is in cell i. However, for a large system of non

interacting particles in equilibrium, the Gibbs measure of entropy is the same as the Boltzmann measure.

Missing Information and the algorithmic description

Where most states are algorithmically random, the system is most likely to be in an equilibrium state. In which case, as shown above, the value of the algorithmic entropy is virtually the same as the value of the Shannon entropy. As the equilibrium states dominate the expected algorithmic description, most algorithmic descriptions are of the order of $log_2\Omega$. Nevertheless, where states show some order, the algorithmic entropy is much lower than the Shannon entropy and it becomes possible to extract work from the system. It follows that if no information is known about the microstate of the system at an instant of time, the best guess, i.e. the most probable algorithmic measure, will be the Shannon value. However an observation may provide information that the system is in a non equilibrium or ordered state. In which case, the algorithmic description will be shorter than the description of an equilibrium state. For example if all the particles of the Boltzmann gas are observed to be on one side of the container, the algorithmic description for this ordered state will be lower than the Shannon entropy. As more information is known about an ordered state, it becomes possible to compress the algorithmic description. This contrasts with a system in an equilibrium state where a more accurate specification of the algorithmic entropy leads to no change. As has been mentioned, this insight prompted Zurek [108] to define the 'Physical Entropy' $\mathcal{S}_d$ of a microstate. This is the sum of the most concise but partial algorithmic description based on the available information, and a Shannon entropy term to allow for the remaining uncertainty. When given information d, the best algorithmic description gives the algorithmic entropy as $H_{algo}(s_i|d)$. This is equivalent to the information that defines the structure or pattern in the string, i.e. the provisional entropy or the AMSS. Zurek sees the remaining uncertainty as captured in the conditional Shannon entropy term $H_{sh} = -\sum_k p_{k|d} log_2 p_{k|d}$. The physical entropy, given the available information, is $\mathcal{S}_d = H_{algo}(s_i|d) + H_{sh}$. Zurek used this approach to argue that, when the physical entropy is low, the system is ordered and work can be extracted. However, $\mathcal{S}_d$ can be seen to be the provisional entropy by comparison with sections 4.1 and 4.2. Furthermore, as Equation (4.4) is of the same form as the equation immediately above, there is no need to introduce a new entropy concept. I.e.

$$\mathcal{S}_d = H_{prov} = H_{algo}(description\, of\, pattern) + H_{sh},$$

as the algorithm that identifies the particular string in the set has the same value as the Shannon entropy. The provisional algorithmic entropy is the best guess of the true algorithmic entropy based on the given information. As more information becomes available, the true algorithmic entropy will drop for ordered states, but remain constant for equilibrium states. Our Figure 2 illustrates these two cases (a) and (b) using Zurek's argument but with the provisional entropy replacing his physical entropy. As the Shannon entropy is

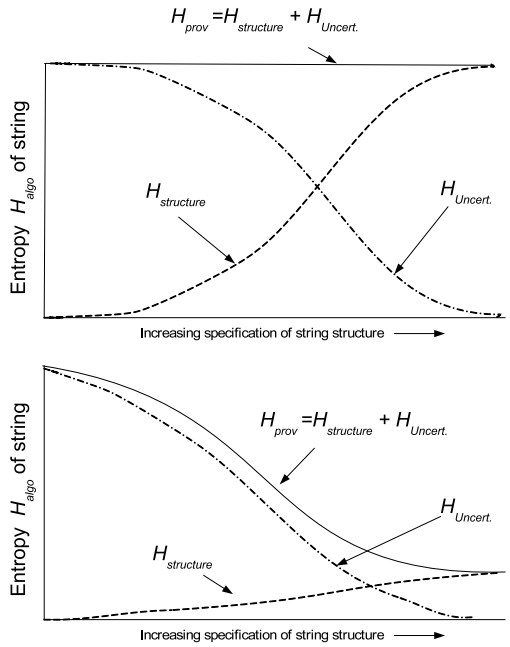

Figure 5.1: Provisional entropy (a) Where the string is a typical member of the set (b) Where the string comes from an ordered configuration.

changing depending on the available information, it is denoted by $H_{uncertain}$ in the figure.

a When the microstate is an equilibrium state (figure 2a), i.e. a typical or random state, the provisional entropy remains unchanged even if more information reduces the uncertainty in the Shannon entropy term (labelled H_{uncert} in the diagram) allowing the particular random state to be more clearly identified. The algorithmic term increases to compensate for the reduction in uncertainty. For such a state, the shortest description of the state returns the same value as the Shannon entropy, even when more information becomes available to better identify the exact state. In effect, the Algorithmic Minimum Sufficient Statistic (AMSS) for the state is the set of equilibrium states (see 4.2).

b When the microstate is ordered as in figure 2b, the extra information leads to a reduction in provisional entropy once the extent of the ordering is ascertained. The approach is general for any noisy description where pattern, structure or order, is revealed with more precise measurements. This is equivalent to refining a model with further information that narrows down the noise or uncertainty. Li and Vitányi [76] look at this another way. They describe the process of revealing order, as one which more accurately defines the microstate of a gas. At low resolution there is only sufficient information to identify the macrostate and the corresponding provisional entropy is high. Increasing the resolution of the system decreases the algorithmic entropy as the uncertainty is less. The process of doubling the resolution for each of the 6N axes, corresponds to doubling the number of significant (binary) figures to the specification of the coordinate of that axis. This process will ultimately identify that the configuration is an ordered one and the provisional algorithmic entropy falls with increasing resolution until it finally approaches the actual algorithmic entropy. This process of more accurately determining the exact configuration of the system is equivalent to finding the optimum set containing the configuration. I.e this is equivalent to determining the AMSS for the state as discussed in section 4.2.

Comments on the Algorithmic Entropy and the Shannon Entropy

The previous sections have argued that the algorithmic entropy of a typical string, representing a configuration in an equilibrium set of states, is virtually identical to the Shannon entropy. If there are Ω states in the equilibrium configuration only a minuscule number of these strings can be compressed. As a consequence, a typical state can only be represented by a string no shorter than $log_2\Omega$. But to reiterate, the Shannon entropy and the thermodynamic entropy, are conceptually different. The algorithmic entropy returns an entropy value for an actual state and has meaning for a non equilibrium configuration. On the other hand the traditional entropies return a value for a set of states, which, in the thermodynamic case, is usually the set of equilibrium configurations. Also, the Shannon entropy takes significant background information as given. It does not specify the system in detail, nor indicate which messages or members are to be included in the set of possible outcomes. On the other hand, the Algorithmic Information Theory approach must:

- specify the system including the resolution or the grain size where, for example, the system is a Boltzmann gas;
- specify the computing overheads and the relevant physical laws; i.e. the computational language that expresses statements equivalent to 'PRINT' and 'FOR/NEXT';
- specify the size of the strings defining elements in the system (hence the log_2log_2 term that often appears in the specification).

As is discussed in section 3.4, the given information that must be specified algorithmically can be taken to be the zero point in defining the algorithmic entropy. In which case, the Shannon entropy is seen to be the particular case of the algorithmic entropy for a typical state in an equilibrium configuration.

Despite the conceptual difference, the relationship between the algorithmic entropy and the traditional entropies is sufficiently robust that one can slip between one description and the other, provided one allows for rounding of the algorithmic measure and recognises the system specification requirement of the algorithmic approach. For example, in the algorithmic case the system specification, such as the phase space resolution, is either included in the description, or is taken as given [108, 48]. This means the ambiguity of the Boltzmann approach is avoided

A simple thought experiment shows the connection between the algorithmic entropy and the thermodynamic entropy. Effectively the difference in algorithmic entropy between two states is about the transfer of bits of information, whereas the thermodynamic entropy is about the transfer of energy. These are related. When ΔH bit are transferred between two states in a computational system at a temperature T the energy transfer is $k_B T ln2 \Delta H$. This follows Landauer's principle that $KTln2$ joules are required to change one bit of information. Consider states i and o that belong to different macrostates. While the system is isolated each is a typical configuration in the macrostate. The minimal possible number of bits that must be transferred into or out of the system to change state i to state o is $|o^*| - |i^*|$ [15]. As this is $H_{prov}(i) - H_{prov}(o)$, the energy inflow required to set the bits in the new configuration in a real world computational system at temperature T is $k_B T ln2[H_{prov}(o) - H_{prov}(i)]$ (see section 6) [1]. From a thermodynamic view, the heat flow δQ needed to shift the real world system at temperature T from the set of microstates containing i to the new equilibrium macrostate containing the state o, will be identical to the energy flow needed change the bit settings in shifting from i to o. As a consequence, the thermodynamic entropy difference between the states is also $\delta Q/T = k_B ln2[H_{prov}(o) - H_{prov}(i)]$ (cf [109]). An example that illustrates the point is seen for the difference between a block of ice at 0 K, and melted ice at the same temperature constrained to the same shape. Given the original ice configuration, the bits that need to enter the system to specify the melted configuration is $\delta Q/(k_B T ln2)$. Thus the thermodynamic entropy difference between the two equilibrium states is identical to the algorithmic entropy difference in bits multiplied by $k_B ln2$. Not only are the two entropy differences identical for equilibrium states, allowing for units, the provisional entropy has meaning for off-equilibrium or non-typical configurations.

The concept of provisional entropy conclusively links the two frameworks. Where a string specifying a configuration shows the configuration is partly ordered and partly random the string can only be partially compressed as many

[1] Note we can measure the number of bits with a computer operating at any temperature, but as the reference computer is mimicking a real world UTM, one can envisage the reference UTM as operating at the same temperature as the real world system.

other strings may have the same basic structure. In summary, the provisional entropy is the shortest known description of a string representing an instantaneous configuration of a thermodynamic system in its state space, whether an equilibrium state or not. The provisional entropy is the key to understanding the entropy embodied in a system exhibiting pattern or structure.

Chapter 6

Reversibility and computations

A historical difficulty with the statistical approach to thermodynamics is illustrated by the thought experiment involving an informed agent, commonly known as Maxwell's demon, who is able to extract work at no cost by judiciously manipulating the particles of a gas in a container. One version of the thought experiment makes the argument that a container of gas can be divided into two halves, both at the same temperature. Maxwell's demon opens a trap door in the dividing wall to allow faster moving particles to collect on one side of the container and closes the door to stop the faster particles returning, thus keeping the slower moving particles on the other side. Over time, the side with the faster moving particles will be at a higher temperature than the side with the slower moving particles. It appears that work can then be extracted violating the second law of thermodynamics.

Initially it was understood (see Szilard [99, 98] and Brillouin [16]) that, as the demon needed to measure the position-momentum coordinates of a particle to know when to open the trap door, it was necessary for the demon to interact with the system under study. It was then argued that the measurement process would require an entropy increase corresponding of $k_B ln2$ for each bit of information obtained.

Landauer [71] showed that this explanation was not completely satisfactory as a thermodynamically reversible process cannot lead to an overall increase in entropy. Landauer pointed out that it is the **irreversibility** of the process that is critical. This irreversibility is reflected in the logically irreversibility of the computation undertaken by the physical system. This happens when information about the prior state is not available, either because information has been discarded, or because there is more than one prior state. As a consequence, when the equivalent computational process becomes logically irreversible the system is no longer thermodynamically reversible. Landauer's approach resolves the paradox of Maxwell's demon - the intelligent interventionist who otherwise would be able to extract work for no cost. Bennett [12, 13, 11, 14] and Zurek [109] show that the demon cannot violate the second law of thermodynamics as the demon must erase the information that was gained in making a

measurement, in order to make room for the next measurement. Unless it does this by returning energy to the environment, the demon will need a continuous supply of energy to flip new bits resulting in an increase in its temperature. The demon is in effect part of the total system and constrained by physical laws. Bennett (see also [109]) illustrates the argument with a gas of one particle and uses the demon to trap the particle on one side of a partition in order to extract work. The entropy loss occurs when the demon memory is reset to allow the process to cycle. However, there are subtleties as Bennett [14] points out. Where information initially is random, the entropy of the system drops if the random information is erased, but the entropy of the environment still increases. One can envisage this erasure as setting all bits to zero, effectively ordering the system and transferring energy elsewhere. The key papers expressing the different interpretations of the Maxwell demon problem, and a discussion on the "new resolution" involving erasure and logical irreversibility, are to be found in Leff and Rex [73]

Implications of an irreversible computation that simulates a reversible real world computation

This principle of Landauer has important implications not only for reversibility, but also for AIT, the second law of thermodynamics and regulation. However there is an important difference between a real world computation and that on a typical reference computer. The generalised gates in the real world computer (i.e. the atoms and molecules and computational elements that change their states under the computation while passing information through the system) are reversible. Reversibility is hardwired into the computational process through the physical laws that determine the computational behaviour of the components such as atoms and molecules. The input string to the computation is stored in the initial states of the atoms or molecules at the start of the process while, given the initial state, the physical laws determine the trajectory of the system through its state space. Similarly, as Landauer [72] points out, any conventional UTM is itself constrained by physical laws and is also a real world device. In this case, the input and the output are not the digits written on an input/output device such as a tape, but the initial configuration and final configuration of bits stored within the computer at the start and finish of the computation. The computation that specifies how the system steps through its bit space is determined by the initial bit settings which includes the programme that manipulates the physical laws embedded in the gates [71, 72, 100]. However, the AND and OR gates of the typical reference UTM are irreversible. While it is possible to construct reversible gates, in practice, when an irreversible reference UTM is required to simulate a real world UTM, the history of the computation that contains, the information needed for reversibility must be kept and tracked. As the next section outlines, Bennett [11] lists the steps required to generate the initial state from this computational history. That this is necessary can be seen with the example of oxygen and hydrogen igniting to form water and in so doing increases the systems temperature. If nothing

escapes, this is a reversible process. Given the initial state of the system, the programme on the reference UTM that defines the algorithmic entropy of the final state of the system must also specify the final momentum states, not just the atomic and molecular species. These momentum states contain the history needed for reversibility. If heat is irreversibly lost, lowering the temperature and leaving water behind, the conventional computer must lose bits associated with these momentum states in order to simulate the final configuration of the real world system. This is such an important point that it will be revisited throughout the book. In a nutshell, the state space configurations of the real world computer is encapsulated in what Landauer [72] and Bennett [14] call the Information bearing Degrees of Freedom (IBDF). The bit space configurations of the reference UTM capture the same information as the real world computer that it simulates provided the history information in the reference UTM is tracked properly.

The Landauer principle, outlined above, clarifies the essentials of a computing process undertaken by a physical system. The input string to the computation is stored in the states of the atoms and molecules at the start of the physical process. In an isolated computer system, bits are conserved as energy is conserved. Because the Hamiltonian dynamics of the system conserves information, whenever a process becomes irreversible, information is not conserved; i.e. when information is lost, $k_B ln2$ entropy units per degree of freedom must be transferred from the system with at least a corresponding entropy increase in the environment. As Landauer showed, it is not the cost of measurement of the information that is critical, but it is the energy cost that arises when information leaves the system by being transferred to the external environment. It is this that causes irreversibility. Bennett [12, 13, 11, 14] and Zurek [109] take the argument further. Discarding 1 bit of information from the IBDF at a temperature T, contributes about $k_B T ln2$ joules to the environment. Both logical and thermodynamic irreversibility follow as information is removed from the computation. This leads to a reduction of entropy of the system matched by an increase of entropy to the universe. Irreversibility can occur when heat is transferred out of the system or when material escapes carrying the information in its "information-bearing" degrees of freedom. A transfer of information into the system also destroys reversibility unless the transfer process retains reversibility. Chapter 8 of Li and Vitányi [76] review relevant issues related to reversibility and the thermodynamics of real world computing. Figure 6.1 is a schematic diagram of different regions of the bit space in a conventional computer. These regions are more or less distinct in a conventional computer, while in real world computations the programme region, which expresses physical laws, is not as clearly defined. In a conventional computer, the bit settings in the programme region under the programmer's control determine the trajectory of the system in bit space as the input moves to the output. Although, once an input and the programme are encoded as "on" and "off" bits in a computer, the distinction between the programme and the input becomes arbitrary. A more rigorous argument that demonstrates this point for a simple UTM is shown in Chaitin [27].

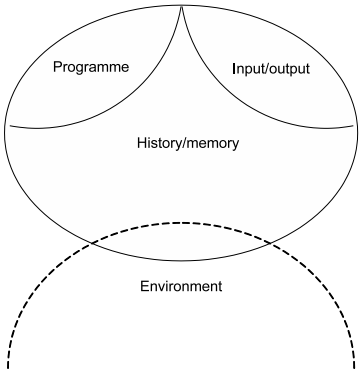

Figure 6.1: Schematic of a computational system

For similar physical situations (as was outlined in section 1.2 bullet point 2), the contribution of the common routines to the algorithmic entropy can be ignored. As only entropy differences are significant the common routines such the relevant physical laws and the specification of the system cancel. This is extremely fortunate, as the specification of all the natural laws would be an impossible task. In effect one can set an entropy zero that takes most laws as given and only considers those laws that are specifically needed to describe the changes of the states.

The simple ballistic computer of Fredkin [54] illustrates how a physical process can act as a computation. However a more relevant example that

illustrates the relationship between a physical or chemical process and a computational process is the Brownian computer of Bennett [12, 11]. This is the computation embodied in the process by which RNA polymerase copies a complement of a DNA string. Reversibility is only attained at zero speeds as the computation randomly walks through possible computational paths. Indeed, because the process is driven forward by irreversible error correction routines that underpin natural DNA copying, the process is no longer strictly reversible.

6.1 Implications for Algorithmic entropy and the second law

Physical processes are in principle reversible and can map on to a logically reversible Turing machine that operates on the input string (representing the initial state) to produce an output string (representing the final state). The computation takes place through the operation of the physical laws determining the trajectory of the system through its states. However in practice most real world systems are open systems and information and material may enter or leave the system. As is mentioned in the previous sections, when information is removed either because energy is removed or molecules leave, the computational possibilities of the system are altered.

However, reversible and irreversible processes cannot be compared on the same reference computer unless, for the irreversible case, the information required to maintain reversibility is is tracked. This follows from the Landauer principle, where any computational process, whether on a conventional UTM or real world computer, cannot be reversible if information about prior states in the computational process is discarded.

However reversibility can be retained in an irreversible computer if the information needed for reversibility is stored in the computer memory. The information kept is the computational history. Reversibility then becomes possible if this information can be inserted at critical steps in the reverse computing process. The real world equivalent is where the information about to be passed to the environment is kept maintaining reversibility. An example is the heat produced when ice crystallises from water. If this heat is kept the process is reversible, but if it is passed to the environment reversibility is lost.

The usual non reversible UTM describes a non reversible physical process when the history of the computation is lost. Nevertheless, a reversible process can be mapped on to a non reversible UTM by using an algorithm that works in both the forward and the reverse direction, inserting the information that otherwise would be lost at the irreversible computational steps [77, 11, 108]. As Bennett [11] shows in detail, such a reversible program broadly speaking has the capability to:

- Generate the output from the input
- Copy the output

- Reverse the computation to remove cumulated history and regenerate the input.
- Swap the regenerated input and the copied output to allow the reverse computation.

The question arises as to how the algorithmic entropy $H(o|i)$, which is usually derived from the shortest irreversible computation that describes the output o given i, is related to the reversible computation. Let $KR(o|i)$, or its equivalent $KR(i|o)$, be the length of the shortest reversible computation that maps the same physical process. Because it is likely that extra information is needed to make the process reversible,

$$H(o|i) \leq KR(o|i) \; or \; KR(i|o).$$

The algorithmic entropy is given by the minimal reversible programme p^* that generates the output from the input on the reference UTM. Its size can be no longer than the reversible program that maps the reversible physical process.

However the program implicit in the Brownian computer described by Bennett [12, 11] may be able to be shortened if a path involving, say, a catalyst is replaced by a more direct computing path. The speed of the process without the catalyst might be less, but the outcome would be the same. Several authors [17, 77, 105] have considered the trade off between computer storage and number of computing steps to reproduce a given output. The shorter the computation the more information storage is required to achieve the reversible computation. For on-going computations, the information stored as the history must be erased. This can only happen if entropy is lost to the environment, making the process more difficult to reverse.

The difference in algorithmic entropy between and input state i and and output state o measures the number of bits that must enter or leave the system when i shifts to o. This entropy difference is independent of the UTM used. If the system is to be restored form o to i, the flow of bits must be reversed. The thermodynamic cost of maintaining order or restoring a system, as bits are added or discarded, is $k_B log_2$ per bit from Landauer's principle [71]. This is discussed in more detail in the next section.

Zurek [109] and Bennett et al [15] have discussed this in detail and have shown that the minimum entropy passed from the universe to the physical system in an irreversible process is $H(i) - H(o)$, noting that $H(i)$ represents the information content of the initial state of the system and $H(o)$ that of the final state. If the system is originally ordered and the final state more disordered, $H(i) < H(o)$ and entropy flows in. In which case, the system can do work, as happens when a heat input forces a gas to do work against a piston. The change in entropy represents the difference between the minimal bits added to specify the original input string, and the minimal bits discarded in the final description of the output. Zurek [109, 108] argues that if one knew the exact description of these states, a cyclic process based on this information would be maximally efficient for extracting work. The consistency of the argument can be seen by considering the following examples.

- An adiabatic expansion of an ideal gas against a frictionless piston, increases the spatial disorder. The movement of the piston in effect is like a programme, changing the computational trajectory of the components of the gas. The algorithmic entropy is derived from the shortest description of the instantaneous configuration of the gas in terms of the position and momentum coordinates of each gas particle. In algorithmic terms, while the contribution to the algorithmic entropy of the position coordinates increases through disordering, the algorithmic entropy of the momentum coordinates decreases to compensate. This decrease corresponds to a drop in the temperature. Adiabatic compression is the converse process. Adiabatic demagnetisation is analogous to adiabatic expansion. Adiabatic demagnetization randomises a set of aligned spins as an external magnetic field is reduced. The disordering soaks up the entropy embodied in the thermal degrees of freedom lowering the system's temperature. While a system remains isolated no erasure of information, or total change in entropy occurs.

- On the other hand, an isothermal expansion of a gas against a piston, can disorder or randomize a system as heat enters to do work against the piston. In contrast to the adiabatic case, the algorithmic description of the system increases as heat flows in. The thermodynamic entropy increases by $k_B ln2$ per bit change in the algorithmic description. Conversely, isothermal compression orders the gas. For an isothermal magnetic system, the equivalent of isothermal compression is where external magnetic field is applied to aligns spins and in doing so passes $k_B T ln2$ joules per spin to the environment.

- In an open, real world system, the computational processes that reset the states of the computational elements (i.e. the atoms, molecules, spins etc.) require energy to be expended or released. This may occur through electrical energy; through magnetic field gradients that align spins; through photons and molecular structures that enter or leave the system; through heat flows into or out of the system; and finally through work being done on or by the system. Energy sources within the system can also be redistributed to reset the computational states as, for example, when hydrogen and oxygen are converted to water by these real world computational processes. In the hydrogen oxygen case, the energy released is passed to the kinetic energy states of the system. Provided the system is isolated there is no entropy change. The process increases the length of the algorithmic description of the momentum states, while reducing the length of the algorithmic description of the position coordinates. As one water molecule replaces a hydrogen and an oxygen atom the description of the species also changes. The physical (or computational) processes within the system disperse the entropy. Regions that are disordered relative to the environment such as those embodied in the kinetic energy degrees of freedom can then pass excess entropy to the environment leaving the more ordered regions behind.

Ordering only occurs when entropy is ejected overall - for example when heat is passed to a low entropy sink in the environment or high entropy molecular fragments escape the system. The programme on the reference UTM that specifies the states must capture the same information as the real world UTM. Whereas real world states are set by the operation of physical laws, in the reference computation, the programme captures the information embodied in the laws. When material enters or leaves a real world system, expanding or contracting the state space, the corresponding process in the reference computer involves expanding or contracting the bit space. This could correspond to bits passing to the computer memory area corresponding to the environment. Alternatively it could occur through the connection or removal a memory device such as a memory stick. For example when species enter a real world system altering the trajectory through its state space, the equivalent for the reference computer can be envisaged as a programme change brought about by adding programme bits via a memory stick.

6.2 The cost of restoring a system to a previous state

As has been mentioned, the degradation processes of the second law of thermodynamics can be visualized as a computing process. The input string i specifies the components of the system, the nutrients and any source of concentrated energy. A programme string p specifies the algorithm embodying the physical laws, which have not been made explicit. The output string o is generated by the computation on computer U where $U(p^*, i) = o$. If all the information is kept in the computation; i.e. nothing escapes the system, the process is thermodynamically and computationally reversible.

However, the process becomes irreversible when the system loses information through heat transfer, or through material exiting the system. This information must be replaced if the system is to be restored to its original macrostate. This means that a living system, must maintain itself in a far-from-equilibrium configuration by offsetting the degradation implied by the second law of thermodynamics. Section 1.2 introduced Chaitin's degree of organisation D_{org} as a measure of how far a system is from equilibrium. If there are Ω_{set} states in the equilibrium configuration, the algorithmic entropy of a typical string will be close to $log_2\Omega_{set}$. Denoting the algorithmic entropy of the ordered string by s by $H(s)$, $D_{org} = log_2\Omega_{set} - H(s)$. This distance from equilibrium of a system can only be maintained by the injection of appropriate order and the ejection of disorder.

Zurek [109] has applied the argument of section 6 to quantify the cost of restoring a system to its input state i when information has been discarded to produce the output state o. In order to restore the system at least $|i^*| - |o^*| = H(i) - H(o) = H(i|o^*)$ must be returned to the system [109]. While Zurek [108, 109] has also shown that a slightly stronger form of the restoration process involves returning $H(i|o)$ bits to the system, the weaker version, generating i

from o^* provides a lower bound to the cost of restoration. The weaker version is more useful, as it is anti symmetric and can therefore track the entropy gains and losses around a thermodynamic cycle, allowing for the directions of entropy flow (Appendix A, [15]). Furthermore, the restoration path does not need to be the exact reverse computation of the forward path.

Consider a system that was originally in a microstate ϵ_i belonging to a stable set of states ϵ_i, ϵ_j etc., all of which have the same provisional entropy e.g. $H_{prov}(\epsilon_i)$. The disturbance δ switches the trajectory of the system to a new configuration η_j outside the stable set. This is analogous to plugging a memory stick containing a computer virus into a computer. The corresponding computation p^*_δ effected by the disturbance or virus is $U(p^*_\delta, \epsilon^*_i) = \eta_j$. The number of bits that enter the system due to the disturbance is

$$|p^*_\delta| = H(\eta_j) - H_{prov}(\epsilon_i).$$

Here a positive sign represents an entropy inflow. The measure is an entropy difference and is the same whether on a real world computer or a reference UTM. Applying Zurek's argument over a cycle, at least $H_{prov}(\epsilon_i|\eta^*_j)) = H_{prov}(\epsilon_i)) - H(\eta_j)$ bits must be returned to the system to restore it to one of statees in the original stable set. This leads to the thermodynamic requirement from Equation A.1 that at least

$$k_B ln2 H(\epsilon_i|\eta^*_j) = k_B ln2[H_{prov}(\epsilon_i) - H(\eta_j)] \tag{6.1}$$

Joules per degree must be introduced into the system to restore i. The real cost of course may be higher as systems are seldom thermodynamically efficient.

Ashby [5] has argued that an open system must have the regulatory capability to adapt to a changing environment to maintain homeostasis. From a cybernetic point of view, an external regulator is required to maintain the system's variables within a viable region. This approach leads to Ashby's law of requisite variety, which states the requirement for a system to regulate is that the Shannon entropy of the set of regulation states must at least equal the Shannon entropy of the set of disturbances [5] [22]. From a computational point of view, Zurek's argument above shows that where an external regulator passes information in and out of the system, a law similar to Ashby's law of requisite variety emerges. Here the law becomes; "The algorithmic entropy introduced by the correction process must equal the algorithmic entropy introduced by the disturbance.

This point of view considers the disturbance and the regulator to be external system and both act on it. However when a system is distant from equilibrium, the second law processes that drive the system towards equilibrium act as the disturbance but in this case they are part of the system. Similarly, where a system self-regulates, the regulation process acts within the system. For example a forest can maintain itself by utilising photons from the sun. If the light is blocked the forest will degrade to a local equilibrium. Section 9.6 generalises the argument discussed above for the situation of self-regulating systems operating under second law degradation processes.

6.3 The algorithmic equivalent of the second law of thermodynamics

The second law of thermodynamics requires that the entropy of an isolated system increases to a maximum as the system's trajectory moves from a non equilibrium configuration towards an equilibrium one. The Boltzmann approach explains the entropy gain by the increase in the system's available states as the energy becomes uniformly dispersed. Ultimately at equilibrium, when the number of available states is Ω, the Boltzmann entropy becomes $k_B ln\Omega$. On the other hand, the algorithmic approach identifies the entropy gain as the increase in the length of the algorithmic description of the microstate of the system as its state trajectory moves from an ordered configuration to a typical or equilibrium one. This increase is, in effect, due to the number of programme bits that must be added to the system at the start of the computation for the computation to terminate at the appropriate halt configuration. As is discussed later, at equilibrium the history of the trajectory is lost, and the overwhelming majority of states require an algorithmic description of length $log_2\Omega + O(log_2\Omega)$. The corresponding algorithmic entropy is close to $k_B log_2\Omega$, a value that aligns with the Boltzmann entropy of $k_B ln\Omega$ for the equilibrium configuration.

When the system is both reversible and isolated, no information can be lost, and each state has a unique precursor. In essence, the system is at least quasi ergodic and the system's trajectory will access all Ω states in Ω steps. The evolution of such a system can then be tracked. The algorithmic details of the second law process by which an isolated real world classical system moves from an initially ordered state s_0 to an equilibrium state has been developed in Appendix C in an article by Zurek [108]. Assume a finite but large discrete set of accessible states, or alternatively define a discrete set of states by choosing an appropriate degree of resolution. Consider an isolated reversible natural system where the initial state s_0 is the most highly ordered state of the system and its algorithmic entropy is $H(s_0)$. Let t_h be the number of computational steps to output the state s_h from the state s_0 and then halt. The real world computer that shifts s_0 to s_h has the physical laws defining the trajectory embodied in generalised gates, the natural equivalent of computational gates. The reversible reference UTM that simulates this trajectory must also simulate the physical laws in the programme that drives the evolution of the system. As each step in the evolution of the system is reversible, the reversible algorithm that maps the physical laws on the reference UTM to produce the final or halt state s_h can be expressed in the following schematic form (e.g. see Zurek, [108]):

$$\begin{aligned} STATE &= s_0 \\ FOR\ STEP &= 0\ to\ t_h \\ &Compute\ next\ STATE. \\ NEXT\ STEP. & \end{aligned} \tag{6.2}$$

Only if this algorithm is the shortest possible will its length give the algorithmic entropy of s_h. As this computation maps reversible physical laws, reversibility is allowed for on the irreversible reference UTM when the history of the computational process is kept. In this reversible situation, the total entropy before and after a second law process is constant. The apparent entropy increase enters the system as the bits or state settings embodied in the programme including the bits specifying the halt instruction t_h. The bit settings of the initial programme and initial state can then be reconstituted from the end state given its history [11, 13]. Furthermore, as science is predicated on the assumption that the natural laws are simple and few, the subroutines expressing the physical laws (i.e. *compute next* $STATE$) are expected to be highly compressed relative to the size of the overall programme as t_h grows. The dominant contribution to the length of this algorithm (i.e. Equation 6.2) is $log_2 t_h$, where t_h is the number of steps to the halt state. The programme bits are just as much part of the initial setting of the computer as the start state s_0 and in a sense represent the pre-history of the computation. The algorithmic entropy is expected to consist of the entropy of the initial state, the length of the routine $H(t_h)$ that specifies t_h, the halt requirement, and the entropy in the routines that compute the next state. I.e. if the binary algorithm that generates the trajectory is the shortest possible, the algorithmic entropy is given by,

$$H(s_h) = H(s_0) + H(t_h) + H(Compute\ next\ State) + O(1). \tag{6.3}$$

The above equations shows that the final algorithmic entropy consists of the entropy of the initial state, $H(t_h)$, the length of the routine that specifies the halt configuration and the length of the routines that compute the next state. Strictly the above equation should have a $\leq$, but sign for large t_h, as is shown below, shorter algorithmic descriptions will occur extremely rarely. While each step of this process is reversible, any slight forward driver will ensure that reversibility is impracticable once t_h becomes large. However in general, as discussed in section 6, when the history is kept, no entropy is lost to the environment and the process is logically and thermodynamically reversible [12, 11, 109]. At this point a conceptual issue needs to be addressed. As discussed later in section 9.5, the above Equation 6.3 implies the entropy is increasing as the reversible system moves from s_0 to s_h. Yet there is a paradox. There cannot be an entropy increase as the entropy of a natural process must be the same along a reversible path if no entropy is passed to the environment [12, 11, 109]. This paradox is resolved once it is realised that the system is isolated and the entropy embodied in the programme bits that drive

the computation, must also be identified as part of the initial setting of the system. If p_h represents the programme bit settings that generate the halt state given s_0, the shortest description of the initial configuration, which consists of the programme and the so called initial state, is strictly $(p_h s_0)^*$. Note that p_h is not constant and as the halt configuration increases by one step, the number of bits in p_h will need to increase to cover this. As the trajectory through bit space reversibly generates the final state, entropy is conserved [108] and

$$H(p_h s_0) = H(s_h) = H(s_0) + H(p_h).$$

As there is no alternative path to s_h (other than the reverse computation from the successor state s_{h+1}), this (or the reverse computation from s_{h+1}) must constitute the shortest reversible algorithm to generate the halt state and

$$H(s_h) = H(s_0) + H(s_h|s_0^*) = H(p_h s_0) \tag{6.4}$$

As the system is isolated, the programme bits are transferred into the string s_h and become part of the output. Because of this, the requirements for reversibility are embodied in s_h. If an irreversible UTM is used to map the reversible process, reversibility can be maintained by keeping the history of the computation, or by keeping the initial state in memory together with the final state ([108]). As the reversible system steps through the allowable states as in equation 6.3, the value of $H(t_h)$ on average hugs the $log_2 t_h$ curve (ignoring the higher logarithm terms required for self delimiting coding). While Equation 6.2 grows on average as $H(t_h)$, there are many fluctuations where $H(t_h)$ drops below this as figure 3.2 in Li and Vitanyi [76] indicates. This is because shorter descriptions of the integer t_h exist for some values. As $log t_h$ approaches $log_2\Omega$, the size of the specification of t_h, given by $|t_h|$ changes very slowly and the systems settles in the equilibrium set of configurations. The entropy of a typical, or equilibrium state is

$$H(s_{equil}) \approx log_2\Omega$$

Once the history as waste is passed to the external environment (for example as heat from the vibrational degrees of freedom) the string that is left may be described by an irreversible algorithm that is shorter than the trajectory. I.e. s_h is given by $s_h = \hat{s}_h w_h$ where $\hat{s}_h$ emerges once the memory of the computational process as the waste string w_h is ejected. The state $\hat{s}_h$ will be more orded if it can be generated by an irreversible algorithm shorter than that specifying s_h.

The question arises as to whether the above stepping algorithm represents the shortest description of the halt state of a system, or can a shorter algorithm be found that just describes the final state $\hat{s}_h$. It is shown below that, for an evolving system far from its equilibrium, the algorithmic entropy of the overwhelming majority of configurations is derived from the stepping or evolving algorithm of Equation 6.3 with the equals sign.

The physical laws and the initial state underpinning the evolutionary algorithm are highly compressed, hence the algorithm that describes an evolving

system is dominated by the specification of t_h. I.e.

$$log_2 t_h >> |Compute\ next\ STATE|.$$

As the system evolves, each successive halt state (given by t_h) steps through all the self-delimiting strings representing the integers between 0 and t_h. because there are no more than 2^m programmes shorter than length m, there can be at most 1 in 2^{10} programmes shorter than $m - 10$. This is about 1 in 1000. The number of actual shorter programmes is far less than this, as the requirement for self-delimiting coding eliminates many possible programmes. For the overwhelming majority of outcomes it follows that the stepping or evolving algorithm of Equation 6.3 represents the most compressed description of the final state as very few shorter programmes exist. Therefore one can assume the equals sign in Equation 6.3 for the overwhelming majority of cases.

However, when t_h specifies an equilibrium state the above argument breaks down. As t grows the algorithmic entropy will be dominated by $H(t_h)$ embodying the halt requirement. Except for the rare accidental ordering just mentioned, the algorithmic entropy initially grows on average as $log_2 t_h$ and hugs the $log_2 t_h$ curve [76], indicating that the system's configuration is becoming less ordered as the number of steps to reach the halt state increases. Eventually, when t_h becomes sufficiently large the system will be in an equilibrium configuration. If $t_h = \alpha\Omega$ where α is less than 1 but approaches 1, the algorithm is dominated by $log_2\Omega + log_2\alpha$. Hence the set of equilibrium configurations will have an algorithmic entropy given by

$$H(s_{equil}) \approx log_2\Omega + log_2\alpha.$$

The equation reduces to $log_2\Omega$, whenever $log_2\alpha$ is negligible relative to $log_2\Omega$. The provisional alogrithmic entropy is equivalent to the Boltzmann value but with different units. Just as happened during the system's evolution, there might be spontaneous fluctuations away from equilibrium, even if there are no entropy exchanges with the environment.

Similarly, where a set of states with a common structure exists far-from-equilibrium, the algorithm that specifies the trajectory may be longer than one that specifies the common pattern or structure for these set of states. In this case the provisional entropy of this set of states provides the shortest algorithmic description rather than the trajectory algorithm.

In the particular case where the evolving algorithm itself involves the replication of a basic unit it is not clear whether the shortest algorithm will be one that describes the structure directly, or one that uses an evolving algorithm to map the structure's growth. I.e. is the algorithm that maps the process that generates the tree from the seed, shorter than the algorithm that defines the tree directly by specifying the final state of each cell of the tree in terms of the cell type and how the cells are combined together? Specifying the placement of each cell of the tree and the cell type on a case by case basis is likely to require a longer algorithm than the one that generates the tree from the genetic code. The reason is that the algorithm based on generating the tree from the

genetic code, specifies that certain genes will be expressed depending on the environment of neighbouring cells and external resource inputs. The cell type is determined by the gene expression. Perhaps a simpler argument is to see the natural world as a computer. The programme that generates the tree on the real world computer involves the algorithm embodied in the seed, the external resources needed to grow the seed and the final halt state that determines how much the tree should grow. It was argued above, that for the overwhelming majority of outcomes, there exists no shorter programme. Hence deriving the algorithmic entropy from the start condition (the initial state of the seed, the resource inputs and the programme) will in almost all cases be shorter than attempting to derive the measure from the algorithm that specifies the final state of the tree.

Tracking changes to the entropy in these situations will be discussed in section 9.7.

Chapter 7

The minimum description level approach

The Minimum Description Length approach (MDL) was developed by Rissanen [83, 87] (see also Barron et al [7]) as a practical method to find the best model to describe a sequence of data points. There are basically two forms of MDL.

The first, a true algorithmic approach, is known as ideal MDL [104]. It is the theoretical approach that seeks the shortest description of a data set by a two part algorithm. The first algorithm captures the regularity of the data with a hypothesis or model. The second part is the algorithm that captures the deviation of the actual data from the regularity given the model. If the model is too simple it may have a short description, but this will be more than off-set by the longer description of the deviations from the regularity implied by the model. An optimum clearly exists. The algorithmic or ideal version is an extension of the AMSS or provisional entropy approaches described in section 4.3. However while the ideal MDL approach finds the algorithm that best fits the data, it does not generate the data as required by algorithmic information theory. Nevertheless, once a short decoding routine is appended the data can be generated from the optimum coding algorithm. In which case, the length of the MDL algorithm with this decoding routine added is virtually the same as the provisional entropy. In the ideal form, the model captures the regularity of the set and the string of interest is identified given the set. Like the provisional entropy, the ideal approach also suffers from the fact that the algorithmic description is non-computable and there is no systematic procedure to decide whether the understanding embodied in the hypothesis or model is the best possible. Different models can be investigated, but the process is *ad hoc*.

Practical MDL, the second approach, addresses the non-computability problem by replacing the idea of algorithmic complexity with stochastic complexity. I.e. rather than finding the length of the shortest actual algorithm, the stochastic complexity seeks the minimum code length based on Shannon's Coding Theorem. As is outlined later, practical MDL provides a tool for selecting

the best model from a class of models by a systematic process. If the class is sufficiently broad, such as the class of polynomials, or the class of Markov models, the process identifies which model in the class describes the data with the fewest number of bits. Inevitably, the optimality is relative to the class or classes chosen. If this were not so, the approach could be used to determine all the laws of physics.

In what follows the ideal or algorithmic MDL and the general principles of MDL will be discussed. Following this, the practical MDL approach will be briefly outlined as a model selection process. Once a model is established, it can be used define the appropriate set, just as is done for the provisional entropy approach. However, as the practical approach to MDL is a separate discipline in its own right, it is not the focus of this article. Rather here the major focus will be on the algorithmic version and its identification with the provisional entropy. But first the discussion on optimal coding (3.3) needs to be revisited.

Much of what is to be discussed in this section is the same as that discussed in section 4.4 which generalised the algorithmic approach to model development. This generalised approach is virtually identical to the Minimum Description Level. Take a string $y^{(n)}$ representing the sequence $(y_1, y_2, y_3 \dots y_n)$ that is the output data of a computational process given the input data string $x^{(n)}$. The ideal MDL approach provides a more formalised method for capturing data that links Algorithmic Information Theory with model selection. As was mentioned in section 4.4 a model, in effect, is a way of compressing data. A good model is one that has a maximally compressed description of the data. Consider as an example planetary motion under Newton's laws. The regularity is in the algorithm that implements the laws. This is the model. The variation (which may be due to the limits of the model, noise, or experimental error) of each data point y_i will appear as random deviations from the model. The optimum compression of a particular data point is given by the provisional entropy or the Algorithmic Minimum Sufficient Statistic as was discussed in section 4.2. The model specifies the characteristic of the set of valid outcomes, and the variation or deviation from the ideal is captured by the code that identifies the particular string in the set. The deviation will appear random, because if it were not, the model could be altered to give a better fit. The ideal MDL approach selects the Algorithmic Minimum Sufficient Statistic (AMSS) or the best estimate of the provisional entropy for each data point. As is discussed below all typical strings in the set defined by the model will have the same description length. A few strings which are not typical will have shorter descriptions. The following paragraphs make this clear using an alternative derivation of MDL based on Bayes' rule. This is helpful because of the insights the derivation provides, and because of the links into the practical version of MDL. Li and Vitányi [76] outline the principles.

Returning to Bayes' rule outlined in section 3.8, the probability of the particular hypothesis or model, $\mathcal{H}_i$ from a set of alternatives given the data D satisfies:

$$P(\mathcal{H}_i|D) = P(D|\mathcal{H}_i)P(\mathcal{H}_i)/P(D). \tag{7.1}$$

The hypothesis or, in our case the model, may be articulated as a physical law; a simple model (such as a straight line fit to the data); or be from a class of models such as the polynomial, Bernoulli, or Markov models. The approach outlined below does not establish the parameters of the possible models; rather, given the best fit obtained from a number of possible models, the approach establishes which of these best-fits is preferred. For example a trial fit of the data could use a straight line based on a standard fitting process such as least squares. Then a second degree polynomial could be trialled and so on. This process establishes the optimum parameters for each potential model. Given these models with their parameters, the MDL approach then establishes which of the possible fits provides the shortest algorithmic description of the data. Because higher degree polynomials require more parameters, higher degree polynomials require longer descriptions. Such a polynomial will only be preferred if the deviations from the model are sufficiently reduced to compensate for the longer description. The ideal MDL approach is derived by taking the negative logarithm of Bayes' equation (7.1) above, to get

$$-log_2 P(\mathcal{H}_i|D) = -log_2 P(D|\mathcal{H}_i) - log_2 P(\mathcal{H}_i) + log_2 P(D). \qquad (7.2)$$

The Bayes' approach selects the hypothesis that maximises $P(\mathcal{H}_i|D)$. This is equivalent to choosing the hypothesis that minimises the right hand side of equation (7.2) and in so doing, provides the shortest description of the data. However, as $log_2 P(D)$ is independent of the hypothesis or model, it is irrelevant to the minimisation process and can be ignored. Where, as is usually case, the prior probability in the above equation is not known, the universal distribution $m(\mathcal{H}_i)$can be used as the prior (see section 3.8) to replace $P(\mathcal{H}_i)$. The universal distribution is the best estimate of the unknown probability. Furthermore, provided that the restrictions outlined below are satisfied, $m(D|\mathcal{H}_i)$ can also replace $P(D|\mathcal{H}_i)$ and the equation to be minimised becomes:

$$-log_2 m(D|\mathcal{H}_i) - log_2 m(\mathcal{H}_i).$$

Because of the algorithmic version of Shannons coding theorem (section 3.8), the universal distribution $m(.)$ is equal to $2^{-H(.)}$. Hence $H(\mathcal{H}_I$, the shortest algorithmic description of the model hypothesis $\mathcal{H}_i$, can replace $log_2 m(\mathcal{H}_i)$ in the above equation. This leads to the equivalent expression of the above equation in algorithmic entropy terms. I.e. to within the $O(1)$ constant, the optimum mode minimises:

$$H_{algo}(D|\mathcal{H}_i) + H_{algo}(\mathcal{H}_i).$$

This is equivalent to finding the provisional entropy of the hypothesis or model where

$$H_{prov}(D) = H_{algo}(D|\mathcal{H}_i) + H_{algo}(\mathcal{H}_i).$$

I.e. the hypothesis that minimises the provisional entropy, or selects the Algorithmic Minimum Sufficient Statistic, as given by equations 4.6 and 4.8, is preferred. If there is more than one possibility with the same minimisation, the

hypothesis $\mathcal{H}_i$ chosen is the one with the shortest $H(\mathcal{H}_i)$. Effectively the MDL argument is that the shortest algorithmic description, given the information available, is the preferred hypothesis.

For a finite set of hypotheses, the Bayesian approach coincides with the ideal MDL using $m(\mathcal{H}_i)$ as the prior probability provided that the substitution of $H(D|\mathcal{H}_i)$ for $m(D|\mathcal{H}_i)$ is valid. This substitution is only valid where the data string, represented by D is a typical member of the data set implied by the hypothesis. To be typical, the string must be random with respect to the set. This can be illustrated by considering a model that describes a particular string as the outcome of Bernoulli process where the probability of a '1' is p and that of a '0' is $1 - p$. If the hypothesis is that $p = 0.75$ the approach will only work if the outcome is a typical string in the set of strings satisfying the hypothesis. A string of 75 1's followed by 25 0's is not a typical string in the possible strings. The particular string mentioned above would only be typical (or random) for a different model. In general, replacing $m(D|\mathcal{H}_i)$ by $H(D|\mathcal{H}_i)$ is only valid when the data string is random with respect to the hypothesis. If the data string is not typical, just as in the AMZZ case a new hypothesis or model is needed. When the right hypothesis is found, and each member of the set is equally likely with respect to the new hypothesis then $log_2[P(D|\mathcal{H})] = -log_2(1/\Omega)$ where Ω is the number of members in the new set. Similarly in algorithmic terms, $H(D|\mathcal{H}) = log_2\Omega$. Also, like the AMSS and the provisional entropy approaches, the minimum description length codes a string in a two part code; one part that defines the model and the other part that specifies the particular string, given the model. Alternatively, this model can be thought in terms of the first part describing the theory, and the second describing how, given the theory, the data with deviations can be exactly fitted. Strictly, while MDL identifies the optimal hypothesis or model, it only becomes an algorithmic measure when a short decoding routine is included to generate the string from the model's parameters.

The MDL approach highlights again the fact that the most compressed description of the set of observations $y^{(n)} = (y_1, y_2, y_3, \ldots y_n)$ involves finding the model that best fits the set of observations in a way that minimises the deviations from the model of each data point y_i. The ideal MDL approach is identical to the argument outlined in more detail in section 4.4. For example, a straight line model used for an MDL approach will attempt to explain a noisy set of data by the equation $y_i = ax_i + b + noise$. Here the hypothesis is the straight line model but which also includes a probability distribution to cover the noise or variation. The minimum description approach minimises the lengths of the algorithmic description of the model plus any deviation due to the noise. Such an approach would require $2d$ bits to specify the constants a and b to a precision d, parameters c and σ to specify the distribution to the same precision. The deviations from the model are specified using a code based on Shannon's Coding Theorem as outlined in section 4.4. In the linear case, where $y(x)$ is a straight line, the best minimum description for a particular

string, given the input x_i, becomes the provisional entropy:

$$H_{prov}(y_i) = H[(ax_i^{(n)} + b), P] + |code[y_i - y(x_i)]| + |decoding\ routine|$$

Here $|code(y_i - y(x_i))| = -log_2 P(y_i - y(x_i)$ allowing for the need to round up the probability to give an integer code. Once MDL has selected the optimum model, or the equivalent set, and the typicality requirements have been met, the model gives the provisional entropy for a particular string. However, there may always be a shorter description waiting and the shorter description could point to a better model. A more practical process that overcomes the non-computability issues in ideal MDL in order to determine **which** model is optimum given the data is discussed in the next section.

Practical MDL

The practical MDL approach developed by Rissanen, bypasses the computability problem of ideal MDL approach to model selection by defining the stochastic complexity to replace the algorithmic complexity [83, 86, 87]. The stochastic complexity is computable. It corresponds to the minimum description of the observed data string using the optimal codes based on probabilities, rather than on an algorithm (although at times the most efficient code might well be virtually the same as the shortest algorithm). Thus the stochastic complexity is defined as the shortest description of the data, relative to a class of probabilistic models. Stochastic complexity was inspired by the AIT approach and Solomonoff's approach to induction (see section 3.8). Reviews can be found in Grünwald [60] and Hansen and Yu [61].

Shannon's Coding Theorem, which was discussed in section 3.3, shows that an optimal code can be found for a countable set of objects denoted by y_i, distributed according to a probability $P(y_i)$. These objects can be strings and, in this section these strings are considered to be derived from particular models. As section 3.3 points out, Shannon's noiseless coding theorem (Equation (3.12), demonstrates that optimum codes from a prefix free set can be chosen to virtually equal to the rounded probability namely that:

$$|code(y_i)| = -\lceil log_2(P(y_i))\rceil.$$

As the probability is usually not an integer, the code length will be the next greatest integer. The convention is to denote the code length by $-log_2 P$ with the understanding it is to be an integer. One further point is that where the probability model is expressed as a probability density, the variables y_i are taken to a chosen level of precision such as four significant figures. In this case a probability $-log_2 P(y_i)$ for each y_i is replaced by $-log_2[P(y_i)\delta]$, where δ represents the grid size determined by the resolution. As the δ is constant throughout the MDL optimisation procedures, it can be ignored.

Again the principal behind the approach is that the better the model of the observed data, the more compressed the description will be. This suggests that the best model or models will be those that provide the shortest description.

However the question arises as to whether the process has advantages over other statistical methodologies that select hypothesis or models. In general, the practical MDL approach avoids over fitting, where there are too many parameters in the model relative to the data, as well as under fitting where the model is too simplistic. When there are too many parameters, the error term is small, but the overall description including the model description is longer. Under fitting the data gives rise to excessive variation in the data from the simplistic model's values, which requires a longer description of the deviation. For example, a data string of 20 points could be fitted by a simple straight line but the deviations might require a large contribution from the $-log(probability)$ term. However a model with 20 parameters could give an exact fit but the model description might be too long. The best fit lies between these. In a practical sense, MDL satisfies Occam's razor.

The approach has some important differences from traditional data fitting as there is no assumption that the process produces the true model or true parameters underlying the data. Rissanen points out [86] that the optimal model is not an approximation to anything; it just fits the data. The practical MDL approach can be derived from Bayes' theorem as outlined previously for ideal MDL. Alternatively, it can be argued that when the data is maximally compressed, all the useful information has been extracted. Minimising the stochastic complexity becomes, in effect, ensuring that the data is random relative to the class of models and that all pattern has been captured in the model in a manner similar to the AMSS approach.

Given a data set, the practical MDL approach seeks to find which member of a class of models provides the shortest description of the data. The MDL model class under consideration is usually expressed in terms of the parameters that define the class. The class could be a set of linear regressions where the data y is a linear function of several variables say θ_1 to θ_n. In the non-linear regression case a similar set of parameters θ_1 to θ_{k+1} representing the coefficients of a polynomial of degree k can be used. In this case, the model can be represented in terms of the parameter set θ and the model label k. A general Markov model is described by its order k, representing the length of the history required to define the future together, while in this class of models the parameter set θ becomes a set of probabilities $p^{(k)}$. A Bernoulli process is the simplest Markov model with $k = 1$ and a probability p. (Note here that $P(y)$ represents the probability of string y in the set, while p is the probability of the next character in string y.)

The non-predictive version [83, 84] shows that in certain circumstances the minimum description of a string of length n reduces to:

$$-log_2 P(y|p^{(k)}, k) + (k/2)log_2 n + O(log_2 n).$$

The second term parameterises the model class, while the first term is a probability based code that codes for the data given the model specified by parameters p^k and k. Consider a data string generated by a Bernoulli process, with r $1's$ in the string of length n, and with $k = 1$. The shortest description

for large n depends on $nh(p) + (log_2 n)/2$, where h is the entropy rate - the entropy per symbol [85]. The maximum likelihood estimate of the probability gives $p = r/n$. However if a higher order Markov model is to be considered as a better explanation of the data, k will increase adding $(k/2-1)log_2(n/2)$ bits. If this increase is more than off-set by a decrease in the code length derived from the probability term, the latter model will be preferred. The optimum description is the one that minimises the total length over the whole class of models. The practical MDL approach has evolved significantly since the early work of Rissanen [83, 86, 87]. Later versions use a normalised maximum likelihood model to provide better descriptions of the data.

Maximum Entropy formulation

The "maximum entropy" approach developed by Jaynes is an attempt to make the maximum use of information when modelling a system [66, 67, 68]. The approach, which has at its heart Laplace's principle of indifference, seems to work despite the scepticism of the traditional frequentist understanding of probability. Interestingly, as the following outlines, the Minimum Description Length approach, can be used to justify the maximum entropy approach of Jaynes [84, 76]. In other words, minimising the description length is equivalent to maximising the Shannon entropy.

In an actual experimental situation, the true probability p_i of a particular outcome i may not be known. If there are n independent trials and k outcomes and the outcome i occurs n_i times, the probability of outcome i given by p_i can be approximated by n_i/n. However while n_i may not actually be known, any information observed determines the constraints on the possible spread of outcomes. For example one constraint should be that the total approximations to the probability should be 1. I.e. $\Sigma n_i/n = 1$. Similarly for an energy distribution over a number of outcomes each with energy E_i and total energy E, the average observed energy per state is constrained by $\Sigma E_i n_i/n = E$. These constraints limit the possible values of the probabilities for each outcome. In which case, the Maximum Entropy principle can identify the model that best fits the constraints by determining the probability distribution $\hat{p_i}$ that maximises the entropy. If the Shannon entropy

$$H(p_1 \ldots, p_k) = -\Sigma_i p_i log_2 p_i. \tag{7.3}$$

has a maximum when $p_i = \hat{p_i}$, the set of probabilities $\hat{p_i}$ is the preferred distribution.

In the Maximum Entropy formulation, the Lagrangian multiplier is used to capture the information embodied in the constraints to find the value of $\hat{p_i}$ that maximises the entropy. This is effectively the approach used in the derivation of Maxwell-Boltzmann statistics. Li and Vitányi page [76] give the example of the so-called Brandeis dice problem (see also Jaynes [67]), where the average outcome of the throw of a dice indicates bias. The observed bias constrains the allowable probabilities. The set of probabilities $\hat{p_i}$ that maximises the entropy,

given the constraints, is then found. The following outlines why minimising $-log_2 P(x|p)$ is equivalent to maximising the Shannon entropy.

In general, the linearly independent constraints can be represented by a series of equations of the form: $\Sigma_j a_{i,j} n_j/n = x_i$, Where x_i represents the observed value of a physical quantity (Rissanen [84]). There will be a set of possible distributions $p = (p_1, p_2 \ldots p_k)$ each of which satisfies the above constraints, and each will have an associated probability as a hypothesis for explaining the constraints. (Note that here the probability of the hypothesis P representing a particular model, is to be distinguished from a particular set of probabilities p that satisfy the constraints x.) Any allowable probability distribution fits the data hence $P(x|p) = 1$. In which case, the ideal code length associated with the model ought to be $-log_2 P(x|p) = 0$. Thus the best set of probabilities is the set that minimises $-log_2 P(x|p)$. For large sequences of data, this is equivalent to maximising $n!/(n_1! \ldots n_k!)$. This, given Stirling's approximation, is equivalent to maximising the entropy of equation (7.3) demonstrating the validity of the Jaynes' approach.

Interestingly too, Jaynes [68], Appendix A argues that Kolmogorov's more general probability approach is consistent with his own.

Chapter 8

The non-typical string and randomness

8.1 Outline on perspectives on randomness

Mathematicians have had great difficulty in defining randomness in a robust manner. For example, Von Mises [106] attempted to define a randomness test for an infinite sequence by requiring that relative frequencies converge for all sub sequences. While intuitively this would seem to be reasonable, mathematicians have identified weaknesses with the approach. However Algorithmic Information Theory provides a number of different perspectives on randomness that ultimately lead to a more robust approach. From an AIT point of view, most strings are random in the sense that they are not compressible. However, while the sequence '111 ... 11' of length 100 is as just likely to occur by chance as any other string of the same length, we do not see it as random. The reason is that '111 ... 11' is compressible and is therefore not a typical member of the set of all strings of length 100.

Typicality is related to the equilibrium concept in statistical thermodynamics. The different possible configurations of a gas in phase space (position-momentum space) can be represented by a series of 1's and 0's depending on whether each cell in phase space is occupied or not (see section 5.2). The overwhelming majority of strings in an equilibrium configuration are typical. However some rare strings that indicate order and can be compressed are not typical. Such an example is the configuration '111 ... 11100000 ... 000', represented by a string of repeated ones and repeated zeros. While such a highly ordered string can occur by chance it is very unlikely for this to happen.

Martin-Löf, a colleague of Kolmogorov, recognised that randomness is to do with typicality and the idea of incompressibility. He developed the randomness test known by his name. Chaitin developed the equivalent concept of m-randomness which is probably simpler to grasp. The following outlines the key AIT perspectives on randomness. However, as the different perspectives can be a bit overwhelming they are summarised first and will be discussed in

more detail in the following sections.

- Most strings are algorithmically random as is discussed below, as few can be algorithmically compressed by more than say, 10 %.
- Martin-Löf [80] defined a universal P-test for randomness that is as good as every conceivable valid randomness test. If a string is deemed random at a particular level by a universal test, no valid test of randomness can do any better.
- Kolmogorov developed a measure he called the "deficiency in randomness" of a string; i.e. the more ordered a string is, the more it is deficient in randomness. Randomness deficiency defines a test for randomness that is a Martin-Löf P-test, and is therefore universal.
- Chaitin's measure of m-randomness, where m is the number of characters the string can be compressed algorithmically, is equivalent to a Martin-Löf universal test of randomness allowing for the fact that in the Chaitin case self-delimiting codes (i.e. codes from a prefix-free set) are used.
- Another alternative test is the sum P-test of Gács [56]. The idea behind this is that a judicious betting process can turn a profit if the string is claimed to be random but is actually ordered. The outcome of the sum P-test is virtually identical to the Martin-Löf P-test based on deficiency in randomness, except that again the sum P-test requires the computation to be restricted to a prefix-free set of routines.
- Algorithmic Information Theory makes considerable use of universal concepts; i.e. those that dominate all others members in the class. In this sense the concept of a universal test for randomness is analogous to the concept of a universal computer (additively dominates) and the concept of a universal semi measure (multiplicatively dominates).

Before looking at the different perspectives in detail, the concept of randomness deficiency will first be outlined. AIT recognises that most real numbers of a given length are algorithmically random. This can be seen as there are $2^n - 1$ binary strings with lengths less than a given n. I.e. there are just under 2^n strings shorter than n. Consequently, there are less than 2^{n-m} strings shorter than $n - m$. For example, if $n = 100$ and $m = 10$, there are 1.27×10^{30} strings of length 100 but only 1.24×10^{27} strings of length less than 90. Hence at most, only 1 in 1024 strings of length 100 can be compressed by more than 10 % as there are too few shorter strings available to provide the compressed description. In practice there will be even fewer strings available as many shorter possibilities will already have been used. For example some of these short string will not be available as they are a compressed representation of strings longer than n; others will not be available as they themselves can be compressed further. As most strings cannot be significantly compressed the algorithmic entropy of a string x is most likely to be close to $|x| + ||x||$; i.e.

the strings actual length, and the size of the code for its length. Those strings that cannot be compressed to less than $n - m$ are called "m-random" (see for example Chaitin [28]).

Given the uniform distribution, the randomness deficiency, a measure of how non-random a string is, was defined originally by Kolmogorov for a string of length n as

$$d(x|n) = n - C(x|n).$$

The uniform distribution corresponds to the situation where each outcome is equally likely. This definition uses $C(x|n)$, the plain algorithmic complexity of string x given its length $|x| = n$. For highly ordered strings, such as a string of repeated ones, knowing the length n allows for a much lower value of $C(x|n)$ as $C(1^n|n) = O(1)$. While the definition could be in terms of $K(x|n)$ using self-delimiting coding, the examples discussed below will use simple coding as the dicussion using $C(x|n)$ as the complexity measure is more straightforward. However, as is discussed later, this means that the outcome of a randomness test such as the sum P-test that uses self delimiting coding, or coding from a prefix-free set, will differ slightly from the P-test that uses plain coding.

Returning to the concept of deficiency in randomness, where $d(x|n)$ is large, and close to $|x|$ the string is highly non-random. Such a string is non-typical as its algorithmic complexity is much less than its length. Hence the label 'deficiency in randomness' as the low value of $C(x|n)$ indicates it can be significantly compressed. For example, the string '$11\ldots1$' of one hundred 1's can be compressed to about $log_2 100$, i.e. $C(x) = log_2 100 + O(1)$ (equation (3.1)). As $C(x|100) = O(1)$, its randomness deficiency given the string is $100 - O(1)$; i.e. close to 100. Clearly $111\ldots1$ is a non-typical string in the set of strings is of length $n = 100$. This approach resolves the intuitive problem that, while tossing a coin and obtaining one hundred heads in a row is just as likely as any other outcome it is somehow completely unexpected.

Another way of viewing this is to divide a set of outcomes into two sets where the typical set contains those strings that can be compressed very little and the non-typical set those elements that can be significantly compressed. Most outcomes will fall into the typical set containing the more random outcomes. A significantly compressible or a non-typical string will only occur rarely in a coin toss (unless the coin is biased). As has been mentioned, this is analogous to the argument used in statistical thermodynamics that equilibrium (i.e. random) configurations are overwhelmingly the most likely. While an ordered configuration from the equilibrium set can be observed in principle, this will happen only as a rare fluctuation. It is seen that randomness deficiency is a measure of how far a configuration is from random. Similarly, Chaitin's [28] degree of organisation, is a similar measure but uses $H(x)$ not conditioned on the set of outcomes. Chaitin's measure is the difference in algorithmic entropy of an equilibrium configuration and the configuration of interest. In other word it is a measure of how far the configuration of interest is from equilibrium.

A further point to be noted now, but which will later be seen to be significant, is to look at the fraction of strings that can be compressed a given

amount. Consider the string x of length n generated by an algorithm of length less than or equal to $n-m$. It can be compressed to at least $n-m$ bits. In which case $C(x|n)) \leq n-m$ and the randomness deficiency, $d(x|n) \geq m$. The fraction 2^{-m+1}, of strings with $d(x|n) \geq m$ is small. E.g. for $n = 100$ and $m = 10$ only 1 string in 512 can have $d(x|n) \geq 10$. It can be seen that m can be used as a measure of randomness.

The previous discussion involved defining randomness for strings of a given length. This implies the uniform distribution. However, the measure of randomness deficiency can be extended to a string x in a set of strings S. The notation used here will be different from that commonly used for the number of elements in a set S. This is often denoted by $|S|$. However in this book, the vertical lines are taken to refer to the length of the algorithmic specification. Hence to avoid confusion, Ω_S will be used to denote the number of members of the set. For large sets, most strings will be typical and will be represented by a code of length close to $log_2\Omega_S$, i.e.. the algorithmic entropy of a typical string is close to the Shannon entropy of the set.

The randomness deficiency for the string x in a set S is given by $d(x|S) = log_2\Omega_S - C(x|S)$. If the string is ordered and can be compressed, its randomness deficiency will be high as it is an non-typical member of the set. Consider the set S of all strings of length n with r 1's and $(n-r)$ 0's. An example is the set of sequences of length 100 with 80 1's and 20 zeros. In this case the set has $\Omega_S = 100!/(80!20!)$ members and the typical string can be compressed to $log_2\Omega_S = 69$. Thus any member of the 80:20 set can be coded in a way that compresses the representation to no more than 69 characters in contrast to a truly random string which would require 100 characters. So while the strings in the 80:20 set are ordered relative to all strings, the typical string in the set has $d(x|S) \approx 69$ [1]. However these are non typical strings in this set and can be can be compressed further. They can be deemed "deficient in randomness" relative to the typical string in the 80:20 set where the typical string is compressed to 69. A non-typical member is the set consisting of 80 1's followed by 20 0's, i.e. $111\ldots11000\ldots00$. The algorithm that generates this string would of the form '$PRINT\ 1, (100-20)\ times\ and\ PRINT\ 0, 20\ times$. As $n = 100$ is given as part of the definition of the set, only the value of 20 is needed to generate this string as $80 = 100-20$ can be calculated. Hence $C(x|S) \approx log_2 20 + O(1)$. Here the $O(1)$ covers the specification of 0, and 1, and the calculation of 100-20. The randomness deficiency of the ordered string relative to the typical 80:20 strings will be is $d(x|S) \approx 69 - 4 - 0(1) = 65 - O(1)$. It can be seen that $d(x|S)$ for the set S of all strings of length n will range from around 0 for random strings of length 69 to $log_2\Omega_S - O(1)$ for the ordered string of 80 ones and 20 zeros.

An extension is where the deficiency in randomness is defined relative to

[1] For large strings Stirling's approximation can be used to show that $log_2\Omega_s = nh$ where h is what is called the entropy rate. The entropy rate is the entropy per symbol for a string produced by a sequence 0s and where, for example, the probability of a 0 being generated is p and that of a 1 is $1-p$. In this case the probability p would be 0.2 and $h = -plog_2 p + (1-p)log_2(1-p) = 0.72$, rather than 69. For Stirling's approximation to be valid the string needs to be longer

a recursively enumerable probability distribution P. In which case a typical member of the set will be coded by its probability in the set. According to Shannon's Noiseless Coding theorem, this will produce a code with length the integral value of $-log_2 P$. This is the optimum code length for such a string. In this case a typical member is one that fits the probability distribution. Most strings can be compressed to a code of length $-log_2 P$. However, an ordered string or one that is non-typical relative to the probability, can be compressed by more than $log_2 P$, and the deficiency in randomness becomes $d(x|P) = -log_2 P - C(x|P)$. For the simple case of the uniform distribution of all strings of length n each of the 2^n string is equally likely and $P = 2^{-n}$. and $d(x|P) = n - C(x|n)$. Clearly the deficiency in randomness measures how much an ordered string can be compressed relative to a random string. As the following outlines, the deficiency in randomness provides basis for a robust randomness test.

8.2 Martin-Löf test of randomness

Martin-Löf, who worked with Kolmogorov for a short period, developed a test for randomness now known as the Martin-Löf test [80]. However, before discussing the test in detail the idea will be explored. A string of length say 100, consisting of a mix of 50 ones and 50's zeros would be probably be considered random unless there was an obvious pattern such as the 50 ones forming the initial sequence. But what about a mix of 49 ones and 51 zeros, or 40 ones and 60 zeros? In other words how much variation from the 50:50 ratio would correspond to a non random string.

Martin- Löf's recognised that, while each string in the set might be equally likely, it is possible to divide the set into regions based on some quantifiable feature of the strings. If this can be done, and one of these region has fewer strings than the other, the probability of a string falling in the region with fewer string will be lower than the typical string which corresponds to strings falling into the larger region. Randomness is always relative to an implicit or explicit set of string. For example a string with a sequence of 100 ones in the set of all strings with length 100, would intuitively seem less random than any string where there is a roughly even mix of ones and zeros. The trick is to define the feature that puts the string of 100 ones in a region which has few strings, and therefore there is a low probability of any string falling in this region. Such a region would contain strings that are deemed non typical. While all strings are equally likely to be observed, the probability of observing a string that falls into a region with appropriate features or characteristics can differ markedly from one falling in the typical region. Even though there is no absolute cut off between typical and non-typical, the boundary between the typical and non-typical regions can bet set at a chosen significance level defined by a positive integer m.

The quantifiable feature that divides the strings into regions is the test function $Test(x)$. This test function is designed so that the probability that a

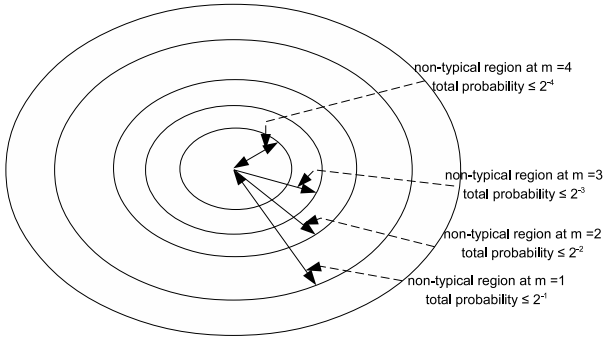

Figure 8.1: $Test(x) \geq m$ defines string x as non-typical or non random in the region m where the total probability $\leq 2^{-m}$

string falls into the non-typical region is given by $\epsilon = 2^{-m}$ and the probability for the typical region of the majority is $1 - 2^{-m}$. A succession of typical/non-typical regions can be characterised by halving ϵ at each increment of m. As m increases, the typical region expands and fewer strings will be deemed non-typical or ordered, and more strings will be typical or random. At $m = 0$, i.e. $\epsilon = 1$, no strings are typical. At $m = 1$ half the strings will fall in the typical region as the value of m implies the cut off is $\epsilon = 2^{-1}$. At $m = 2$, 75% of strings will fall in the typical region and be deemed random at this level, while 25% will be deemed non-typical hence will be seen to be more ordered. As can be seen in figure 3, the typical region of the majority at level $m + 1$ includes the typical region at level m, $m - 1$ etc. As m increases, more strings will be deemed random as the typical region grows. In other word the requirement to categorise a string as non typical is more restrictive as m increases.

The Martin-Löf test of randomness for a string effectively uses a test function to ask whether string x lie outside the majority or typical region at the significance level m and therefore can it be rejected as being random at that level m? Such a test must be enumerable and lower semicomputable to be able to be implemented. The test does not show whether a strings is random, rather it determines the basis on which a string can be rejected as being random. If there are 2^n strings, the key constraint on the test is that it must assign no more than 2^{n-m} strings to the non-typical region defined by a particular m. I.e. if $m = 2$, no more than 25 % of the strings can fall in the non-typical region, otherwise the test would be inconsistent as there are insufficient binary strings available. There are therefore two aspects to the test. The first is to ensure that, under the test, the fraction of strings belonging to the reject or non-typical region is no more than 2^{-m}, and the second is to have a test procedure to determine whether a particular string belongs to the reject region. This test procedure defined by $Test(x)$ determines where $Test(x) \geq m$ for string x. If the fraction is less that 2^{-m} for $Test(x) \geq m$, then x falls on the non-typical side of the typical/non-typical boundary at significance level m and therefore

x can be rejected as random The mathematical formulation of this, when given the set of relevant strings characterised by a recursive (computable for all x) probability measure $P(x)$, becomes [76]

$$\{\Sigma P(x) : Test(x) \geq m\} \leq 2^{-m}.$$

In words, the above test states that that one identifies all the strings where the test has a value $\geq m$. The string can be rejected as random and deemed ordered if the sum of the probability of all these strings is $\leq 2^{-m}$. In other words if the cumulated probability of all strings in the region $\leq 2^{-m}$, the string can be considered ordered. If this requirement is satisfied a string can be consistently assigned to the reject or non-typical region for the given value of m. If there were more than 2^{-m} strings in the region, the test would not be sufficiently discriminating as there cannot be that many non random strings. This assignment is a Martin-Löf P-test of randomness. If there are 2^n strings in the set, an equivalent definition, can be based on the number of strings (denoted by $\#(x)$) that satisfy the criterion that $Test(x) \geq m$. I.e. in this case the definition becomes:

$$\{\#(x)\, where\, Test(x) \geq m\} \leq 2^{n-m}.$$

Martin-Löf test examples

The Martin-Löf process can test whether a sequence produced by tossing a possibly biased coin is random or not; more accurately at what level would the observed string be deemed random. Let $f(x)$ be the number of 1's in the string representing the outcome of n tosses of a coin that might be biased. One might expect $f(x)$ to be about $n/2$ if the outcome is random but how far from $n/2$ is reasonable for a random string? The procedure is to devise a test of the above form that involves $f(x)$. For example Cover, Gács and Gray [38] use Chebyshev's inequality to develop a test function $Test(x) = log_2\{2n^{-1/2}|f(x)-n/2|\}$. As there are 2^{n-2m} strings that satisfy $Test(x) \geq m$, this satisfies the test requirement as $2^{n-2m} < 2^{n-m}$.

An alternative test is to use the entropy rate mentioned in the previous section 8.1. Again let the number of 1's in a string of length n be $f(x)$ and define $p = f(x)/n$ so that $(1-p) = [(n-f(x)]/n$. For large n, the Shannon entropy per symbol is termed the entropy rate; i.e. $h(x) = -plog_2p-(1-p)log_2(1-p)$. With appropriate coding, and for large n, a string of length n can be compressed to at least $nh(x)$, i.e. the string can in principle be shortened by an integer m, where $n(1-h(x)) \geq m$. For example a string with 75% of 1's can be compressed to $0.811n$. Such a string of length 100 can be compressed by at least $m = 18$ characters. Let a Martin-Löf test be $Test(x) = n(1-h(x))$. $Test(x)$ effectively measures the amount the string x can be compressed on this basis and $Test(x)$ is a valid test, as the cumulated probability of all strings where $Test(x) \geq m$ is $\leq 2^m$.

The entropy rate test can be demonstrated by considering a string of length 170. (This length is chosen as it is sufficiently long that the entropy rate is

meaningful, but not too long that 170!, which is needed to calculate cummulative probabilities, cannot be evaluated on an Excel spreadsheet.) Consider the ratio 75:95 ones to zeros. At what level of ϵ would this string be rejected as random? In this case $Test(x) = 1.701 > 1$; it can only be compressed to a length of 169. The cumulated probability of all strings in the region with $m > 1$ is 0.2499 which is less than 2^{-1}. Thus the 75:95 string would only be rejected as random at the $m = 1$ cut off level i.e. 50 % level. It is still reasonably random. On the other hand, a 70:100 string has $Test(x) = 3.8 > 3$. Taking $m = 3$, the cumulative probability of a string being in the region is $< 2^{-3} = 0.038$. This is the region where $\epsilon < 0.125$. The 70:100 string can be rejected as random or considered ordered at the level of $m = 3$ or $\epsilon = 0.125$. On the other hand, the 70:100 string would not be considered ordered at $m = 4$.

Randomness deficiency as a Martin-L öf test

The randomness deficiency $d(x|P)$ in section 8 can be made into a test. I.e. for a set of strings defined be an enumerable probability distribution P, the test [2] is

$$\delta_0(x|P) = d(x|P) - 1.$$

If the uniform distribution is considered where $P = 1/n$, the randomness deficiency test assigns strings of length n to a region using $\delta_0(x|P) = n - C(x|n) - 1$. As the number of strings with plain Kolmogorov complexity $C(x|n) \leq n - m - 1$ is $\leq 2^{n-m}$, the number of strings with $\delta_0(x|P) \geq m$ must be $\leq 2^{n-m}$. Putting this in probability terms by dividing by the number of strings the result is:

$$\{\Sigma P(x) : \delta_0(x|P) \geq m\} \leq 2^{-m}.$$

This clearly satisfies the criteria of the test but more importantly as the next section shows this is a universal test for randomness.

The Martin-Löf universal P-test of randomness

It can be shown that there are universal test of randomness that dominate all other tests of randomness. The argument, which is given in detail in Li and Vitányi page 131 [76], shows that all effective tests of randomness can be listed by a systematic procedure. If $\delta_y(x)$ is the yth test, then the universal test with respect to a set with probability distribution P can be chosen to be

$$\delta_0(x|P) = max\{\delta_y(x) - y\}$$

Hence

$$\delta_0(x|P) \geq \delta(x|P) - c.$$

The universal test is as good as (allowing for the constant) any other conceivable or even inconceivable enumerable test. However some understanding of "as

[2] Readers will note that as $C(x|n) < n - m$, the '-1' is needed to ensure the relationship is $\leq$ rather than just $<$.

good as" is called for in this context. If the ordinary Martin-Löf Test implies that $\delta(x|P) \geq m$. While the string x is in the reject region at the m level, the universal test has $\delta_0(x|P) \geq m - c$. Thus when $\delta(x|P)$ rejects a string as random at $m = 4$, the test $\delta_0(x|P)$ rejects the string as random and considers it ordered at the level, i.e. $m - c$. It is more sensitive in its rejecting randomness and assigning order. This is saying that, in general, $\delta(x|P)$ requires a higher m value than the universal test $\delta_0(x|P)$ to recognise order. The universal test will pick out fewer random strings for a given m. However, there is no one unique universal P-test. Rather there is a number such that each of the universal tests can additively dominate any other. In which case all return the same measure of randomness allowing for the constant. While the universal test may use a different value of m to categorise a level of randomness, it still ranks the level of randomness for all strings in the same order.

The masterstroke in the argument is that the test based on randomness deficiency; i.e. the test $\delta_0(x|P) = d(x|P) - 1$, is a universal test. What is more it is consistent with the definition of m-randomness outlined in section 8.1. I.e. Chaitin [27] has defined randomness as a string x for which $H(x)$ is approximately equal to $|x| + H(|x|)$ or $H(x|n)$ is close to n. To be specific a string of length n is Chaitin m-random if it can be compressed to a string with length $n - m$. I..e. $H(x|n) > n - m$. This also applies to an infinite string α where the test is applied to sequences of the infinite string, i.e. provided that $\exists m$ for all n s.t. $H(\alpha_n) > n - m$. It is seen (Schnorr [89]) that this definition is equivalent to the Martin-Löf definition of randomness at significance level m allowing for the fact the Chaitin definition uses prefix-free complexity rather than plain algorithmic complexity.

Evaluating $\delta_0(x|P)$ and concluding that x is random using a universal test, is equivalent to establishing that the string satisfies all computable randomness tests including any not yet devised. However unfortunately $\delta_0(x|P)$ is not computable.

The sum P-test

Another way of looking at randomness developed by Gács [56, 57] uses the level of payoff for a fair bet as a procedure to recognise a non-typical member of a set of outcomes (see outline in Li and Vitányi [76]. Highly non-random outcomes are a surprise and ought to generate a high payoff if a bet is placed on such an outcome occurring. For example if a coin is tossed 100 times, the hypothesis is that all outcomes are equally likely. However, if the supposed fair coin happens to be biased 3:1 in favour of heads, in all likelihood there will be three times as many heads as tails. If each outcome is supposed to be equally likely, the occurrence of such a surprise outcome should be able to generate a return with a suitably cunning bet. The critical point is that surprise outcomes will be more algorithmically compressed than typical outcomes and therefore can be identified. In order to illustrate the idea, let $H(x)$ be the algorithmic entropy for each outcome x.. Make a \$1 bet with the agreement that the payoff for the outcome of a (possibly biased) coin tossed 100 times is to be

$2^{100-H(x)}$. If the outcome is random, the payoff would be small as $H(x)$ would be close to 100, the algorithmic description length of a typical string. But if $H(x)$ is small for the actual outcome, the measure of surprise is the amount of payoff the outcome produces. This payoff in effect takes $2^{-H(x)}$ as the universal probability distribution $m(x)$ (3.8)

If all outcomes are equally likely the probability for each is $p(x) = 2^{-100}$. The expected payoff for an unbiased outcome x then becomes

$$\sum_{|x|=100} 2^{-100} 2^{100-H(x)} \leq 1.$$

As the expected return for the bet based on tossing a coin 100 times is no more than the stake, the bet is fair. This is because the Kraft inequality ensures $\sum_{|x|=100} 2^{-H(x)} \leq 1$. However if the coin is biased or cheating is taking place, the returns become significant. E.g. for the particular, case of complete bias, where the outcome is '$hhh\ldots$' i.e. 100 heads, and noting that $H(hhhh\ldots hh) \sim log_2 100 = 6$. The return of $2^{100-H(x)}$ for a 1 dollar bet is close to $\$2 \times 10^{28}$; a significant sum.

For the less unexpected case, where the coin is biased 3:1 in favour of heads, the string can still be compressed. The algorithmic entropy of a typical string is close to the $log_2 \Omega_s$ where Ω_s is the number of members in the set of all strings exhibiting this ratio (see section 4). There are 2×10^{23} members in the set giving $H(x) = 77$. In the 3:1 case, the expected return for is $\$2^{100-77} = \$8.4$ million. This return is also not insignificant.

While it is unlikely that the person providing the coin would accept a bet where the payoff depended on how algorithmic compressible the outcome was, the idea can be formalised to define a randomness test. This payoff test is called the sum P-test [56, 57]. Conceptually the approach is to see whether an apparently random string, is actually random or not by whether a judicious betting procedure will generate a positive return. If it is possible to define a fair betting regime that generates a positive return for certain outcomes, there must be hidden structure or pattern in the outcome string. On the other hand for a truly random outcome, no fair betting regime will return an expected payoff higher than the stake. Not only does the payoff understanding of randomness intuitively make sense, within the sum P-test framework it becomes mathematically robust.

In order to generalise the approach, define $\delta(x)$ as a sum P-test where P again is a recursive or computable probability. In the illustration above, $\delta(x) = 100 - H(x)$. This is Chaitin's degree of organisation (section) and is a variant of the deficiency in randomness. In general, for the sum P-test, $\delta(x)$ must satisfy the payoff requirement (mentioned above) that the expected return is less than one. I.e..

$$\sum P(x) 2^{\delta(x)} \leq 1.$$

It can be shown that if $\delta(x)$ is a sum P-test it is also a Martin-Löf P test (Li and Vitányi page 257 [76]), but the sum P-test is slightly stronger as the above

equation implies $\delta(x)$ uses self delimiting coding and therefore satisfies the Kraft-Chaitin inequality (3.2). As a consequence, an ordinary P-test, denoted by $d(x)$ will differ from the sum P-test by a term less than $2log_2d(x)$. This is due to the difference between simple coding and self-delimiting coding.

Similarly a universal sum P-test exists for any recursive probability distribution P, not just the simple distribution in the illustration. Let $\kappa_0(x|P) = log_2[m(x)/P(x)] = log_2[2^{-H(x)}/P(x)]$ as the universal distribution $m(x)$ is equivalent to $2^{-H(x)}$ (section 3.8). The expected return for a bet with payofff $2^{\kappa_0(x|P)}$ is $\sum P(x)2^{\kappa_0(x|P)}$. This is ≤ 1 as $\sum 2^{-H(x)} \leq 1$. If the return for such a bet is signficantly > 1 then the string x cannot be random. Just as $log_2m(x)$ additively dominates all other semi measures, $\kappa_0(x|P)$ additively dominates all other sum P-tests. For example the uniform distribution L_n of a string of length n has $\kappa_0(x|L_n) = n - H(x|n)$. (Note that this differs slightly from the P-test based on the randomness deficiency because (1) the sum P-test uses self-delimiting coding to provide the entropy measure $H(x|n)$ and (2) the extra -1 in the deficiency in randomness is not included. I.e. if $\kappa_0(x|L_n)$ is defined using a prefix-free UTM, it differs by no more than $2log_2C(x|n)$ from the randomness deficiency test $\delta_0(x|L_n) = n - C(x|n) - 1$. This shows that the distance from equilibrium is a robust Martin-Löf measure of order or the lack of randomness of a specific configuration in the natural world. This has implications in the discussion as to whether the order in the natural world indicates design or not.

Gács' sum P-test is intuitively appealing as it implies that if one can win money by a judicious betting procedure on a set of apparently random outcomes, the outcomes cannot be random. In this case, the degree of order is related to the return on a fair bet.

The use of universal concepts

AIT makes use of universal concepts. These are measures that either additively or multiplicatively dominate other measures. In the case of additive dominance, a universal measure gives the same result as any another universal measure but the zero level is shifted by a constant. For example, a universal computer can simulate any other computer and the length of an algorithm on a universal computer additively dominates all other computers. Other than the setting of the zero point, all universal computers are as good as each other (although in practice the measures are more transparent if a simple UTM with a few core instructions is used). Again, the universal semi measure $m(x)$ was defined to multiplicatively dominate all other semi measures - it is as good as any alternative. This is equivalent to the logarithm of the measure additively dominating all other semi measures. As $-log_2m(x)$ additively dominates all measures of algorithmic complexity, this allows one to substitute $H(x)$ for $-log_2m(x)$ and vice versa.

Similarly for randomness tests, there is a test that additively dominates all others randomness tests – it is as good as the best. Different universal randomness tests generate the same order when strings are listed according to

their level of randomness. However the randomness labels may be shifted by a constant between different universal tests. As a consequence, the simple deficiency in randomness provides a randomness measure that is consistent with all other universal randomness tests. In the same way, the self-delimiting equivalent of the deficiency in randomness is $-log_2[m(x)/P(x)]$ which is equivalent to $-log_2 P - H(x|P)$ This can also be taken to be a universal randomness test. These show that high probability outcomes have low randomness deficiency, whereas low probability outcomes are non-typical.

As has been mentioned, physicists will recognise that the randomness arguments apply to an equilibrium configuration in statistical thermodynamics and a typical string from an equilibrium configuration would be expected to pass all randomness tests. However there are always non typical configurations in the set of possible equilibrium states. For example all the atoms in a Boltzmann gas could by chance find themselves on one side of the container. As this would be an extremely unlikely to occur it would be deemed to be an ordered fluctuation from equilibrium. While such a fluctuation could be considered as an equilibrium configuration it has a neglible contribution to the thermodynamic entropy. Because an ordered configuration is non-typical, once such a configuration has been identified an entropy gradient exists between parts of the system. Such an entropy gradient can be harnessed to extract work from the system.

Chapter 9

How replication processes maintain a system far from equilibrium

9.1 Introduction to maintaining a system distant from equilibrium

Section 1.2 points out, the length of the shortest, appropriately coded algorithm that generates a particular string on a UTM defines H, the algorithmic entropy of this string in bits. The algorithmic entropy can provide insights into the natural world, once it is recognised that the physical laws driving a system from one state to another are in effect computations on a real world Universal Turing Machine (see section 5.2). As algorithmic entropy differences are independent of the UTM used, the algorithmic entropy derived from a programme that maps a real world computation by manipulating bits in a reference UTM in the laboratory, is the same as the equivalent number of bits derived from the real world UTM. The flow of bits and information through the two systems is the same.

The algorithmic entropy is the entropy of a particular state, whereas the traditional entropies return a value for a macrostate corresponding to a set of microstates. Nevertheless, the provisional algorithmic entropy outlined in section 4, is the algorithmic entropy measure for a typical microstate in an equilibrium macrostate and returns the same value, allowing for units as the conventional entropies. The provisional entropy is the length of the shortest programme that specifies a particular configuration. Because of this relationship, one can easily move between the algorithmic measure and the conventional entropy description. This allows the provisional entropy to be used to track entropy changes and flows as a system evolves over time.

A living system is deemed to operate far from a local equilibrium, facing real time degradation as the entropy increases under the relentless operation of the second law of thermodynamics (see section 6.3). If there is no compen-

sation for the increase in entropy, the system will revert to a local or global equilibrium. Just to clarify notation, generally in this section an equilibrium will usually be a local equilibrium which is stable for a long period of time. Although, given sufficient time it will degrade further to a global equilibrium. For example, if light from the sun is blocked from reaching a forest, given sufficient time, the forest and its ecology will be reduced to mainly carbon dioxide, water, and minerals. In thermodynamic terms, the second law ensures that the entropy in such a system increases as the concentrated energy embodied in ordered structures becomes more dispersed and becomes unavailable to do work.[1] A living system must have in-built mechanisms to maintain itself in a far-from equilibrium configuration by offsetting the degradation implied by the second law of thermodynamics. One can perhaps see this as a regulation process where the cost of maintenance is the cost of restoring the system to an initial macrostate. The principle argument in this chapter is that replication processes, which generate algorithmically simple and therefore low entropy structures, is a major mechanism in generating and maintaining a living system distant from equilibrium

Landauer's principle [71] formalises the understanding of the computational processes embodied in real world computations and provides a tool to inquire into the emergence and maintenance of order in these real world systems (see [11, 12, 13] and [109]). His principle states that where one bit of information in a computational machine, operating at a temperature T, is erased, at least k_BTln2 Joules must be dissipated. In a conventional computer this dissipation appears as heat passing to the non computational parts of the system. However, where the length of the algorithmic description of the state of a system changes when ΔH bits are removed, the corresponding thermodynamic entropy change in the real world system is $k_Bln2\Delta H$. Similarly, ΔH bits must be returned to the system to restore it to the initial state.

The entropy changes of both living and non-living natural systems are constrained by the same laws. This allows one to apply the entropy and energy requirements of the process of replication to very simple systems and carry the initial insights over to biologically more complex living systems. However in order to take this argument further, the next section explores replication processes, while the section following outlines the principles of AIT. Later, section 9.5 deals with some conceptual issues around the reversibility of natural laws.

The provisional algorithmic entropy can not only quantify the order embodied in a system to capture its information content, it also provides the thermodynamic constraints that govern the emergence of local order. It is when these natural computation processes operating under physical laws eject disorder that more ordered or low entropy structures are left behind.

The algorithmic approach provides a measure of the distance a natural system is from equilibrium in terms of the number of information bits that

[1] Fortunately for life, the rate at which second law processes drive the system towards equilibrium is constrained by rates involved in natural processes and barriers such as the chemical activation energy.

must be added to the system to shift it from an ordered to an equilibrium state. Chaitin's degree of organisation D_{org} is in effect a measure of how far a system is from equilibrium. If there are Ω states in the equilibrium configuration, the algorithmic entropy of a typical string will be $log_2\Omega$. Denoting the algorithmic entropy of the ordered string by s by $H(s)$, $D_{org} = log_2\Omega - H(s)$. The distance from equilibrium of a system can only be maintained by the ejection of disorder and the injection of order (or alternatively high grade energy) into the system to create ordered structures.

The following insights emerge from the perspective provided by the AIT approach to the maintenance of a system far from equilibrium

- As replication processes are core to many natural systems, understanding replication is important to understanding these systems. The critical point is that the algorithmic entropy of a system of replicated structure is low. Relatively few bits are required to specify such a system as the algorithm only needs to specify the structure once and then copy, or generate, repeats of the structure by a short routine. This contrast with the number of bits needed to specify a system where each structure must be independently specified. It follows that replication is a naturally ordering process that reduces the algorithmic entropy of a system in a quantifiable way. Provided resources are available, and high entropy waste can be ejected, replicated structures are more likely to emerge than similar structures produced by other natural process.

- Replication processes can create and maintain an ordered system away from equilibrium in an attractor-like region of the system's dynamical state space where all the states have the same provisional algorithmic entropy. When order is being destroyed through degradation processes driven by the second law of thermodynamics, replication processes that access high quality energy and eject disorder, are able to restore the system to the original ordered set of configurations. In essence, replication processes use natural laws to self-regulate to maintain the system far-from-equilibrium.

- Variation in a system of replicated structures leads to an increase in the algorithmic entropy. However, variation also provides a natural mechanism to stabilise the system against change by allowing the system to evolve to a more restricted region of its state space. In other words, variation can maintain the system in a stable configuration through adaptive evolutionary-like processes. AIT provides a reasonably convincing argument that, in many situations, those variants that persist in a changing environment are those that use resources more efficiently and have lower entropy throughputs.

- Coupled and nested replicator systems create greater stability against change by co-evolving as, from an entropy perspective, the replicated

> structures use resources more efficiently. This efficiency, called here 'entropic efficiency' would seem to be an important constraint on evolutionary selection processes. Nevertheless, while each replicated structure is efficient in this sense, the number of interdependent or nested replicated structures increases to ensure that overall, the high quality energy degrades more effectively than would otherwise be the case [88].

Many living systems also need to do work, increasing the thermodynamic cost of maintaining the system away from the global equilibrium. Systems further from equilibrium have a greater capacity to do work. However, as they are further from equilibrium, these systems need to be more efficient in their use of the resources, i.e. in computing terms more efficient in the use of the computational inputs. Also, those systems further from equilibrium need to be more efficient in ejecting disorder embodied in the output string that exits the system. The cost of maintaining the system in an ordered configuration was discussed in section 6.2 while a more general approach will be outlined in section 9.6.

Three thought experiments to clarify entropy flows in natural systems

There are some conceptual issues that need to be resolved to avoid confusion over what order is, and where order comes from. The following three simple representative thought experiments help to clarify the computational issues and the difficulty in maintaining a system off-equilibrium under the influence of the Second Law of Thermodynamics. The three processes described from an algorithmic point of view are analogous to the second law expansion outlined in section 6.3. The disorder that emerges is not contained in the original description, but disorder arises because of bits introduced into the system to drive the system to its final halt state. The first and simplest example is where heat is removed from the system making the system more ordered. In a sense order is imported from the external environment. The second looks at maintaining a photodiode in an excited state, while the third looks at maintaining a hydrogen oxygen mixture away from equilibrium. The last two examples describe systems which start in a configuration where high grade energy is stored and which redistribute the energy throughout the system over time.

- Consider a gas in a container where the off-equilibrium configuration requires a region of the gas to be hotter than the bulk of the gas. The difference in temperature between the two regions creates an energy gradient and the second law will drive the system towards a uniform temperature. The system at the start is not more ordered than at the finish because nothing has left the system - the entropy is unchanged. In a sense, the bit settings in the momentum degrees of freedom of the warmer part of the gas are the programme that drives the system to its equilibrium and in so doing diffuse the energy through the system. The entropy only

drops when the excess heat is removed passing to the environment. If such a system were to be maintained off-equilibrium, heat would need to be removed and compensating heat would need to be introduced to the high temperature region of the gas. The point of this illustration is to emphasise that the maintenance requirement for an off-equilibrium system is that the entropy gradient between the system and the environment must be maintained. Heat needs to be introduced to one part of the system and extracted from another. In this example the order is not introduced by the injection of heat, but is introduced by the removal of heat. This is equivalent to the injection of cold, as happens when more ordered phonons from the environment enter the system.

- The second example that captures degradation issues is that of a photodiode where initially the electrons have been excited from the valence band to the conduction band. The macrostate of the system is defined by the set of microstates consisting of the excited electronic states, the vibrational states and other states of the system. The excitation produces a potential difference between the conduction and valence band that can, in principle, be used to heat the system either directly (through collision process as the electrons return to the valence band), or indirectly by heating a resistor connected to the photodiode. The photodiode system can also do work if the potential difference generated by the excitation is used to drive an electric motor. Whether by doing mechanical work that dissipates heat, or producing heat directly without work, if left to itself the system will undergo a second law degradation process and settle in a local equilibrium where most electrons return to the valence band increasing the temperature. This increase corresponds to disorder passing to the momentum states of the system.

  In principle, the photodiode can be maintained off-equilibrium in a flux of photons that continuously repopulate the conduction band, provided that the excess heat generated by the process is removed by passing heat to the environment. However, as argued, below this would be a fortuitous event.

- The third representative case is where a stoichiometric mixture of hydrogen and oxygen burns to form either liquid water or steam, depending on the temperature and pressure. This system is initially ordered (see discussion below) but after ignition, the system moves from the initial configuration and will settle in chemical equilibrium where

$$O_2 + 2H_2 \rightleftharpoons 2H_2O + heat.$$

  If the system is isolated, most of the hydrogen and oxygen will have been converted to H_2O, and the temperature of the local equilibrium will increase. What has happened is that stored energy has been released to form water and heat. However, this process is reversible and in itself does not generate order. It is only when heat is lost does the remaining

system become more ordered. Once heat is lost, and if the pressure is such that sufficient liquid water forms, the system can be restored to the original configuration by losing heat and using external energy to regenerate hydrogen and oxygen by, for example, electrolysing the water.

These examples highlight an important point. In the first of the above bullet points, raw energy was originally introduced as heat to create the entropy differential between the system and the environment. While the entropy differential between parts of the system can be harnessed to do work, ordering only occurs (i.e. the entropy is lowered) when disorder as waste or heat is ejected. The entropy of an isolated system does not reduce, even if a chemical reaction takes place, as can be seen in the hydrogen oxygen system. When the components ignite to become water +heat the entropy is the same. The critical factor is that diffusing energy through the system creates a situation where excess entropy can pass to the environment. Ordering occurs because this disorder can be ejected, often as waste species or as heat, leaving a more ordered structure behind. In a sense one can envisage the environment as injecting order into the system.

Maintaining a system off-equilibrium corresponds to maintaining an entropy gradient between the system and the external environment. As has been mentioned, the thermodynamic cost of maintaining a non-equilibrium system is captured by Landauer's principle [71]. The trajectory of a real world system can be envisaged as the result of a computational process operating under algorithms embodied in the laws of nature. As the thermodynamic cost of replacing the lost information becomes the cost of maintaining the system off-equilibrium, AIT provides a tool to inquire into the emergence and maintenance of order in these real world systems.

System processes and system degradation

A general system, such as a living system, must maintain itself in a far-from-equilibrium configuration by offsetting the degradation implied by the second law of thermodynamics. A system can recreate the original order by using stored energy, and by ejecting disorder. As has been mentioned, Landauer's principle [71] showed that the thermodynamic cost of maintaining a non-equilibrium system requires the lost information in bits to be replaced. However most systems, like the inert systems described, can only maintain themselves in the initial set of configurations under special circumstances. They cannot self-regulate, as to do so requires an autonomous process to replace the lost information. The photodiode system can only be sustained in a stable set of configurations by accident, as for example, happens when the system exists in a fortuitous flux of photons that excite electrons into the conduction band, and the system can eject the sufficient excess heat to maintain the energy balance. Similarly, the hydrogen oxygen system would require an unlikely set of circumstances to be sustained off equilibrium.

An inert earth (without life) will reach a steady state where the surface temperature is such that the radiation from the sun exactly balances the heat radiation from the earth. The inert earth as a system is broadly analogous to the photodiode system. The steady state situation that emerges when energy inputs and outputs balance is not true regulation. Contrast this with a living cell. A living cell can actively maintain itself by accessing nutrients and ejecting waste. Much of the self-organising capability of life on earth, or of systems of cells, would seem to be a consequence of evolutionary processes that lead to the capability of structures to replicate. The argument developed below shows that the ability of the earth to self-organise is a consequence of the replication capability of the living structures on earth. While inert systems can maintain stable configurations, replication processes provide a natural mecahnism for true regulation.

9.2 Replication processes to counter the second law of thermodynamics by generating order

As has already been argued, Algorithmic Information Theory indicates that replication processes are key to generating much of the observed local order around us. In contrast to alternative structures, like those discussed earlier, a system of replicated structures is more ordered as the algorithm that generates the system is more compressed. As fewer bits of information are required to describe the structure and, as replication occurs through natural laws, replication processes can both create and maintain order. Indeed, it seems that many natural systems rely on replication processes as a means of self-organisation and self-regulation. Given one replicator, others will form and a system of replicated structure can be maintained in a way that compensates for the second law of thermodynamics.

This can be seen by considering a set of N repeated replicated structures that can be generated by a short algorithm of the form "Repeat replicated structure N times". The algorithmic entropy of this algorithm corresponds to

$$H_{algo}(system) = |N| + |replicator\ description| + |CI|,$$

where CI refers to the common information. As the algorithmic entropy is dominated by a the relatively short string representing $|N|$. The length of this description is close to $log_2 N$ bits. Such a short algorithm implies an ordered structure. Contrast this with a similar set of non-replicated structures where the algorithm must specify each of the N structures separately, taking at least N bits [2].

The process of replication is like a chain reaction that creates repeats of a particular structure using structural or stored energy. While replicated struc-

[2]Note that here the word "replica" will be used to denote those physical or biological structure that use external energy and resources to reproduce themselves. This means the biological cell is the replicating unit, not the DNA information string that is copied as part of the replicating process.

tures cumulate in the system, other physical processes, such as collisions or chemical reactions, will destroy these structures. If the system is isolated it will eventually settle in an equilibrium configuration for long periods. The increase in order due to replication only occurs when the disorder appearing in the states that capture the history of the computational process is ejected allowing the replicating structures to dominate the system. There can be no overall entropy change when the system is isolated.

In general, replication involves physical or biological structures that reproduce by utilizing available energy and resources and, at the same time, ejecting excess entropy as heat and waste to the environment. As the structure left behind has lower entropy it is ordered. Furthermore, it is argued that the drive for replication in such a system becomes a self-regulating process that maintains the system off equilibrium even though the replicated structures degrade under the second law of thermodynamics. Simple physical replication processes can be used to illustrate the key characteristics of natural ordering through replication, even though biological replication and system degradation involves much more complex pathways.

Physical examples of such replicating systems are a crystal that forms from a melt, a set of spins that align in a magnetic material, and coherent photons that emerge through stimulated emission. For example, molecules are more likely to solidify on a seed crystal in a melt.

Biological examples include an autocatalytic set, bacteria that grow in an environment of nutrients and the growth of a biological structure such as a plant. In the last example, as different genes are expressed in different parts of the plant, variations of the basic structure emerge at these points.

The characteristic of a replicator is that, in a resource rich environment, the probability that a structure will replicate increases with the number N of existing structures. Consider a classical physical system, initially isolated from the rest of the universe where the instantaneous microstate of the system is represented by a point in the multidimensional state space. Over time the state point of the system will move through state space under the operation of physical laws. As the probability of replicated structures appearing increases with their occurrence, replication will drive the state point to a region of state space dominated by a large number of repeated replicated structures. In this case replicated structures are more likely to be observed than alternatives. However for the replicated structures to dominate the system, because bits are conserved for an isolated system, compensating disorder must shift to other states. For example when spins become aligned in a magnetic material, disorder is shifted to the vibrational states (i.e. the temperature rises). The vibrational states, which become more disordered, capture the history or the memory of the computational process. Unless the corresponding heat can be removed, the system remains reversible and the spins will in time revert back to a more random configuration.

Consider the situation where the probability of the appearance of a replicated structure is proportional to N^x (where $x > 0$), then $\mathrm{d}N/\mathrm{d}t \propto N^x$ [97]. For this to happen it is assumed that entropy as waste products or as heat can

be ejected so that entropy and waste does not build up within the system. For example, given a double helix of DNA in the right environment, the probability of a second double helix appearing is comparatively high.

The focus here is on true replication processes where the presence of the replicated structure triggers more replicas.[3] Where resources are limited, the number of replicated structures grows over time until a state of homeostasis is reached; a state where the set of replicators and the environment reach a long-term stable relationship.

A schematic representation of this process is the logistic growth equation

$$\mathrm{d}N/\mathrm{d}t = \hat{r}N(1 - N/K). \tag{9.1}$$

This equation, while trivially simple, captures the essence of replication. Here $\hat{r}$ represents the usual exponential growth rate and K represents the carrying capacity of the system. The number of replicas in a replicating system grows over time until the system reaches homeostasis; a state where the set of replicated structures and the environment reach a long-term stable relationship. In the homeostatic state replicas may die and be born. $\mathrm{d}N/\mathrm{d}t = 0$) and the system is maintained in a homeostatic state by the flow through of nutrients and energy and the ejection of waste. The rate at which the system can access resources and eject waste determines where this homeostatic balance occurs. For some simple physical processes this waste can be ejected as latent heat through the thermal degrees of freedom, as happens when the vibrational states of the system lose heat to the environment. Alternatively the waste might be relatively less ordered photons, or degraded chemical products that exit the system. Provided that the high entropy waste can be ejected from the system, the algorithmic entropy, and therefore the thermodynamic entropy, of the remaining structure decreases.

Many living systems also need to do work, increasing the thermodynamic cost of maintaining the system away from the global equilibrium. In order to fly, birds do work at a greater rate than mammals or reptiles. As a consequence, their body temperature is higher. While systems further from equilibrium have a greater capacity to do work, they need to be more efficient in their use of the resources that are inputs to the computational process. They also need to be more efficient in ejecting disorder that, from a computational point of view, is the output string that exits the system. As is discussed later, evolutionary type processes acting on a system of replicated structures select variations that are more efficient in this sense.

9.3 Simple illustrative examples of replication

In contrast to the photodiode system or the hydrogen and oxygen system discussed above, replication process are naturally ordering processes. While many

[3]True replication requires that the presence of one unit creates a chain reaction to create more units. This contrasts with processes such as the combustion of oxygen and hydrogen, where the heat of combustion, not the presence of water, triggers the process.

replicating processes occur in living systems, much can be learned about the process by considering simple physical examples. Two illustrative but simple examples are the alignment of magnetic spins (i.e. magnetic moments) below the Curie temperature and the replication of photons by simulated emission. In the second example, stimulated emission provides ordered repeats of a structure by passing disorder to the electronic states of the atoms and then to thermal degrees of freedom. These examples are discussed in some detail below to clarify the entropy flows in an ordering process.

Replicating spins

For those unfamiliar with magnetic systems, in what follows the word "spin" refers to the magnetic moment of a spinning electron. One can envisage this as a micro magnetic associated with a magnetic atom such as an iron atom. These micro magnets might all be aligned in which case one says " the spins are aligned" etc. Hence above the Curie temperature an aligned system of spins will become more disordered as thermal energy passes into the spin system. If the disordered system is isolated, it will settle in an equilibrium configuration where the spins are partially aligned. It is only when the temperature falls below the Curie temperature, with ejection of latent heat, will full spin alignment occur. In this case ferromagnetic coupling between spins will dominate the thermal energy. The system has become more ordered by extracting order from an external colder environment by the process of discarding heat. The ideas can be illustrated by a simple model - the ordering of a 1-D chain of spins. This model not only illustrates replication, but also is informative about what happens to the information during the ordering process. Let the configuration of the coupled 1-D chain be specified by a string of N disordered spins denoted by x where x is either 1, representing a spin up, or 0 representing a spin down. Let the vibrational states be below the Curie temperature. As the vibrational states are at a low temperature there is an entropy gradient between the spin subsystem and the vibrational subsystem. The ordered vibrational configuration can be represented schematically by an ordered string of N zeros. The initial configuration i consisting of a random set of spins and the ordered vibrational states is then represented by,

$$i = 1xxx \ldots xx000000 \ldots .$$

The first character is a '1 and represents the particular spin that will be replicated in the sense that, under the right conditions, the spin directions will align because of coupling between the spins. Spin alignment is a spontaneous symmetry breaking process. The computational equivalent of the replication process that aligns all the spins with the first spin involves a computation that scans the string and, because of coupling due to natural laws, finds the replicated structure '1', and alters the next characters to '1'. As this continues through the randomly aligned string, information (i.e. bits) must be transferred to the $00 \ldots 00$ section of the string, disordering the vibrational degrees of freedom.

Thermal energy must be transferred from the spin states to the more ordered vibrational states disordering them. This effectively raises the temperature of the vibrational degrees of freedom. While the system is isolated no entropy is lost and the vibrational states retain the memory of the physical process maintaining reversibility. If we now consider that the system is in contact with a heat sink, the vibrational states can revert to their original more ordered configuration by passing the disorder as heat to the external sink, leaving the more ordered system behind. At this stage, the output string of the process that aligns the spins by replication is

$$o = 11111111 \ldots 11xxxxx \ldots xxxx.$$

As was mentioned, the randomness has been transferred to what previously was the ordered section of the string. This indicates that the temperature of the vibrational states has increased, but without the ejection of waste heat the entropy has been conserved. It is only when the disorder is ejected, because of the coupling between the vibrational states and the low temperature external sink, will the system become more ordered at the expense of the rest of the external universe.

For large N, ignoring computing overheads, the information content of the input string is given by the algorithmic entropy $H(i)$.

$$\begin{aligned} H(i) \quad &\approx (N + log_2 N) \;\; (specifying\ the\ random\ 1xxxx..x) \\ &\quad + log_2 N + |0| \quad (Specify\ 0,\ N\ times). \end{aligned} \tag{9.2}$$

Let U_1 be the computation that maps the real world computation aligning the next spin with the current spin. The overall ordering process can be envisaged as a repeat of this alignment process down the spin chain. In which case, the output after N steps is:

$$o = (U_1)^N i = 11111 \ldots 11xxxxxx \ldots xx. \tag{9.3}$$

The output string can be specified by a similar algorithim but where this time the spins are ordered and the vibrational states disordered. I.e. the algorithmic entropy becomes algorithm

$$\begin{aligned} H_{algo}(o) \quad &\approx log_2 N + |1| \; (aligned\ spins) \\ &\quad + N + log_2 N + |0| \; (disordered\ vibrational\ states) \\ . \end{aligned} \tag{9.4}$$

At this stage, as nothing is lost from the system reversibility is implicit in the history of the computation embodied in the final state. Hence $H_{algo}(o) = H_{algo}(i)$. Once the vibrational states lose heat to the external environment the algorithmic entropy is lowered and the output becomes;

$$o = 11111 \ldots 1100 \ldots 00.$$

The algorithm that shifts i to o is implicit in the bit settings of the state i as these carry the algorithm in the physical laws that are embodied in the

generalised gates of the real world system. I.e. the instructions U_1^N are implicit in the original state.

Coherent photons as replicators

Another simple example is the replication of coherent photons through simulated emission. Consider a physical system of N identical two-state atoms where a photon is emitted when the excited state returns to the ground state [47]. The emitted photons can either be in a coherent state, or a set of incoherent states. The state space of the system initially consists of;

- the position and momentum states of the atoms;
- the momentum states of the photons;
- the excited and ground electronic states of each atom.

Assume the system of photons and particles are contained within end mirrors and walls and no energy loss occurs through collisions with the container. The initial ordered configuration will be taken to be the N atoms existing in their excited states with the photon states empty. If the lifetime of the excited state of the atom is short, only incoherent photons will be emitted, and the system will settle in an equilibrium consisting of excited and ground state atomic states. These incoherent photons with momentum vectors scattered over all directions, will not be significantly ordered. In addition, under second law processes, some excited atoms will return to the ground state via collisions or other processes before emitting a photon and in so doing increasing the thermal energy of the system. Ultimately the system will settle in a local equilibrium at a higher temperature with most atoms in the ground state.

On the other hand if the lifetime of the excited states is long stimulated photons becomes possible. This can be envisaged as replicas of a single photon state. Once a stray photon replicates, photons will continue to replicate until there is a balance between the number of excited atoms, the number of coherent photons and the very small number of incoherent photon states. A symmetry breaking process takes place as the stimulated photons have aligned momentum vectors, rather than being randomly oriented over all directions. While the system is isolated, each photon replication is reversible and a coherent photon may be absorbed, returning an atom to its excited state. Photon replicas die and are born as the system settles in an attractor-like region of its state space. The physical computation process for an isolated system is both thermodynamically and logically reversible.

The algorithmic description of an instantaneous configuration of this isolated system after a period of time will consist of the description of the atomic states; a description of the state of each incoherent photon; the description of the coherent photon state which consists of replicas of the original stimulating photon; and the momentum states of the atom which are deemed to be random. Let the electronic states of the system be represented as a string where a

'1' represents an excited atom and a '0' represents the ground state. The input string, i representing the initial highly ordered system of excited atoms with no photons takes the schematic form:

$$\begin{array}{rll} & \text{[Atom electronic states]} & \text{[Photon states]} \\ i= & \text{[1111111111111111111]} & \text{[empty].} \end{array}$$

In the discussion the incoherent photon states will be ignored as these will be few. The momentum states and position states of the atoms can also be ignored. During the replication process they will become more disordered as the temperature rises and reverts back to the previous description if the final temperature is unchanged from the initial temperature. Initially all the atoms are in their excited state and there are no photons. If stimulated emission is possible and if the system is isolated, after a period of time, a photon will be emitted and this will stimulate the emission of other coherent photons. Similarly incoherent photons may also be emitted. Over time the system will consist of a mixture of atomic states and n photons in a coherent state. The algorithmic description of an instantaneous configuration of the total system will consist of the description of the atomic states in terms of whether they are excited or in the ground state and the description of the coherent photon states which are copies of the original stimulating photon. I.e.

$$\begin{array}{rll} & \text{[Atom electronic states]} & \text{[Coherent photon state]} \\ o= & \text{[0011111011011010110]} & \text{[n]} \quad . \end{array}$$

The n coherent photon state is more ordered than the incoherent states as the momentum vector is no longer randomly oriented. The memory of the photon creation process needed to maintain reversibility passes to the momentum states of the atoms. In practice this memory disappears when heat is exchanged with the environment. If photons do not replicate, the system will tend to an equilibrium as stored energy is passed to the momentum states of the atoms analogous to a free expansion. However, photon replication generates order by produce the coherent photon state once entropy is passed to the environment.

However, if photons can escape, the system will need to access an external energy source to re-excite some of the atomic states to compensate for the photon loss. The external source that excites the atoms could be an electrical discharge or a flash lamp.

These examples, while seeming trivial capture some of the key conceptual issues in understanding ordering from an AIT perspective. If the physical laws are embodied in programme p that runs on UTM U, the process can be envisaged as $U(p, i) = o$. As the above examples illustrate, in general a replication process creates order by passing the disorder elsewhere; effectively partially randomizing other degrees of freedom. If the system is isolated for long periods, the system will settle in a region of state space where replicated structures die and are born as the total system moves through possible configurations. Nevertheless, the system is still in principle reversible. However, when entropy is transferred to the momentum states (the thermal degrees of freedom) of the

system, increasing its temperature, the difference in temperature between the systems momentum states and the environment allows entropy to be transferred to the thermal degrees of freedom in an external sink. Because the computation process is now irreversible, the system will settle in a restricted region of its state space. The length of the algorithmic description is reduced and the entropy cost of this is $k_B ln2$ per bit transferred to the sink.

The process of maintaining the system the system in such a configuration is discussed in section 6.2 and 9.6.

9.4 The entropy cost of replication with variations

The previous sections considered the situation where replicated structures were identical. However, many real world systems consist of near identical, but not completely identical replicated structures. For example the expression of a cell in the eye is not identical to one in the liver yet all contain the same DNA coding. This variation in the replicated structures has an entropy cost that can be quantified by the Algorithmic Information Theory approach. In the case of near identical replicated structures, the provisional algorithmic entropy identifies the entropy increase with the uncertainty arising from the variations. As a consequence, the algorithmic description is longer.

A simple example, which can be generalised, illustrates the principle. Consider the variation introduced into the simple string 101010...10 by allowing both '10' and 'also 11' to be considered as valid variants of the replicated structure. This string then becomes the noisy period-2 string discussed in section 4.3, where every second digit is random. Thus a valid system of replicated structures would now be any output string of length N having the form $o = 1y1y1y1y..1y$, and where y is 0 or 1. There are $\Omega_s = 2^{N/2}$, members of the set of all possible variants of the string. In general, the provisional algorithmic entropy, which is given in section 4.1, includes the length of the code that identifies the string within the set of all variants, together with the specification of the pattern, or the model, that identifies the set itself. I.e.

$$H_{prov}(o) \simeq log_2\Omega_s + |description\ of\ pattern\ of\ replicated\ units|. \qquad (9.5)$$

As there are several similar algorithms that might be used to specify the provisional entropy, here '$\simeq$' is used to indicate that small inefficiencies in a specific algorithm can be ignored. The code that identifies a particular member of the set requires $|\Omega_s|$ bits.

In the noisy period-2 string, the size of the specification of Ω, i.e. $|\Omega_s|$, is close to the $N/2$ bits that are needed to identify the particular member of the set of variations. In order to define the set of allowable strings, a further $log_2 N/2$ bits are required to specify the length of the string in the set needed for self-delimiting coding and the specification of the variants '10' and '11'as was discussed in section 4.3 equation 4.9. The provisional entropy becomes;

$$H_{prov}(o) \simeq N/2 + log_2(N/2) + |1| + |0|, \qquad (9.6)$$

ignoring higher "*loglog*" terms. By comparison, the provisional algorithmic entropy of the string $o = 101010...10$, a string which represents no variations in the replicated unit (i.e. 10 not $1y$), has the algorithmic entropy given by

$$H_{prov}(o) = log_2(N/2) + |10|. \tag{9.7}$$

As the length of the code representing '10 and that representing '1 and '0 separately are virtually the same, the provisional algorithmic entropy of the string with variation has increased by $N/2$ over the simple string of equation 9.7. As was mentioned earlier, this is the increase in the uncertainty (i.e. the Shannon entropy) that arises by allowing variants of the original structure.

The general case

More generally let $r_1, r_2, r_3, \ldots r_{\mathcal{M}}$ be the $\mathcal{M}$ variations of the replicated unit r. If the logarithm base is taken to be the number of replicated structural types $\mathcal{M}$, the code that identifies a particular mix, would then be of length L in this base, as there are $\mathcal{M}$ choices at each position in the base-$\mathcal{M}$ number system. The length of the same specification in binary notation would be $L(log_2\mathcal{M})$. As the codes are self-delimiting, information on the length of L and M must be embodied in the codes, or else in the definition of the set. The contribution to the algorithmic entropy of these needs to be recognised and tracked.

The first step in quantifying the provisional entropy is to establish the provisional entropy for each replicator. Let the *ith* replicator be identified in the set of $\mathcal{M}$ replicated structures by a code. As the code needs to be self-delimiting, the length of the code to identify a particular replicator will be the logarithm of the number of members in the set, i.e. $log_2\mathcal{M}$, plus information on the length of the code. I.e. the *ith* replicator can be identified within the set by a code of length $log_2\mathcal{M} + |log_2\mathcal{M}|$ plus higher "*loglog*" terms. The algorithm $Algo(r_i)$ reads the code for the *ith* replicator and, given the criteria defining the set of replicators, outputs r_i. As each replica has the same algorithmic entropy $H_{prov}(r_i)$ for any i becomes;

$$H_{prov}(r_i) = |Algo(r_i)| \simeq log_2\mathcal{M} + |log_2\mathcal{M}| + |criteria\ defining\ replica\ set|.$$

Once the specification of each non -dentical replica is established, the description of the set of non-identical replicas can be determined. Let S_L be a mix of L non-identical replicated structures of the form $S_L = r_i r_j r_j r_i \ldots$. The next step is to output S_L, knowing the codes $Algo(r_i)$ for each replicator in the code string for S_L The set of all possible replica variations of length L has $\mathcal{M}^L$ members, as there are $\mathcal{M}$ choices for each replica position. The algorithm that generates S_L, denoted by $Algo(S_L)$, needs to contain the following.

- the specification of L and, in addition, in order to be self-delimiting, the length of the code that specifies L needs to be included. I.e. $||L||$.
- The algorithm needs to access the value of $\mathcal{M}$, the number of variations of the replica, from the subroutine that generates a particular replicator variation.

- As there are $\Omega_s = \mathcal{M}^L$ members in the set of possible variations of S_L, the particular variation is given by a code of length $L(log_2\mathcal{M})$
- Once the particular mix of replicas is determined using the process mentioned previously, given $\mathcal{M}$, the specifying algorithm reads $Algo(r_i)$ in S_L specifying the first replica, implements the algorithm and outputs r_i.
- The algorithm reads the next instruction for the next replica $Algo(r_j)$ and outputs the string r_j and joins this to r_i. This process continues until all the L algorithms for each replica in the string is processed and the algorithm finally halts after outputting the L replicas in sequence.

The provisional algorithmic entropy of a particular replica mix becomes;

$$H(S_L) \simeq L(log_2\mathcal{M}) + |L| + |log_2\mathcal{M}| + |criteria\ defining\ replica\ set| + O(1). \quad (9.8)$$

Here the $O(1)$ term includes the contribution of minor computations such as the instructions that calls routines within an algorithm.

An alternative less detailed argument recognises that there are M^L members of the set of strings of the same form as S_L. In order to specify a particular string, the set itself needs to be defined together with the Shannon entropy contribution term $L(log_2\mathcal{M})$ that is a measure of the length of the code that identifies a particular member of the set. One can step through all the strings S_L that are members of the set until a particular code is reached to define a particular string as outlined in section. The result is as before

$$H(S_L) \sim L(log_2\mathcal{M}) + |L| + |log_2\mathcal{M}| + |criteria\ of\ replica\ set| + O(1).$$

For the particular case where all replicas r are identical, further pattern exists, and in this case as $\mathcal{M} = 1$ the $L(log_2\mathcal{M})$ and the $\log_2\mathcal{M}|$ terms are eliminated. Equation 9.8 then becomes, as one would expect;

$$H_{prov}(identical\ replicate\ string) \simeq log_2 L + |r|. \quad (9.9)$$

This is equivalent to the period-2 string in equation 4.9 with $|L| = log_2 L = log_2 N/2$, while $|r| = |10|$, specifies the characteristic of the particular structure to be replicated. If N is large relative to the pattern description, equation 9.8, which allows for the variation, leads to an increase in entropy of $L(log_2\mathcal{M}) + |log_2\mathcal{M}|$ over the situation where all replicas are identical. However this is an upper limit as it is based on the assumption that the arrangement of variations is random. This is seldom so in living species where, for example, all the variations corresponding to an eye cell, occur in clusters. For this reason, the actual increase in provisional entropy is likely to be much smaller. I.e. when internal triggers determine which genes are to be expressed, the information capturing this information is already embedded in the growth algorithm. On the other hand, external switches such a food inputs, light etc. may determine the gene expression and will add to the algorithmic entropy of a simple replicating algorithm. In essence the provisional entropy of living beings built from variations of a core replicated structure may not be significantly different from a structure with identical replicated units.

9.5 Entropy balances in a reversible system

Equation 3.14 shows that in general the algorithmic entropy of the output o of a computational process may be less than the sum of the algorithmic entropies of the initial state i and the conditional entropy $H(o|i^*)$. This happens when information is discarded in going from i to o. In which case, o is said to be redundant with respect to i (see Zurek [109]). If however all the information is accounted for in a reversible process, an equals sign is appropriate in Equation 3.14 as there can be no entropy change. It is important to note that the programme that defines the algorithmic entropy must halt by definition, as the programme must specify the configuration exactly. For example the algorithm that generates a tree from the DNA instructions in the seed must halt, either because the observer defines the halt requirement, or because a natural event is fed into the computation to terminate the growth algorithm. In an isolated system there can be no entropy increase as the system moves reversibly from an apparently ordered configuration to the halt state. This would imply a paradox as the system is supposed to be moving from an ordered and therefore a low entropy configuration to a less ordered one and yet reversibility implies no entropy change.

The paradox is resolved once it is realised any apparent increase in entropy of the output is reversibly introduced into the system from the initial bit settings in the programme with its halt instruction (see section 6.3). The separation between an initial state s_0 and the programme driving the system, is an arbitrary one. Once the bit setting of the programme, p_h that drive the system to the halt state are considered as part of the initial configuration, it is seen that $H(p_h s_0) = H(s_h, \zeta)$, where $H(s_h, \zeta)$ is the entropy of the halt configuration including the history of the computation ζ that is needed to maintain reversibility. This understanding is implicit in the discussion by Zurek [108] and Bennett [11, 13].

Nevertheless, because this is an important point, the following theorem shows why algorithmic entropy is truly conserved in reversible processes.

Theorem: In an isolated, reversible, discrete system, the algorithmic entropy that enters the system as the intitial configuration and the programme with its halt instruction must be conserved, appearing as the entropy of the computed final state and the entropy embodied in the computational history.

Proof: Consider the shortest reversible trajectory through a system's bit space (or state space) in the form of Equation 6.2. Let the halting configuration be s_h after t_h steps and let the history of the computation needed to maintain reversibility be represented by the string ζ. Thus having selected the shortest reversible path to generate s_h with the history ζ, after t_h steps, it needs to be shown that

$$H(p_h s_0) = H(s_h \zeta) = H(s_o) + H(trajectory\, after\, t_h\, steps). \tag{9.10}$$

Case 1. For a specific t_h where the state s_h has no shorter irreversible trajectory, s_h and ζ are generated together by the same programme implying that $H(\zeta|s_h^*) = 0$ from Equation 3.14. No shorter programme can generate $s_h \zeta$ as,

being reversible, there can be only one path to the configuration $s_h\zeta$ [4] In this situation, the equals sign in Equation 9.10 is valid.

Case 2. Where a specification of s_h exists that is shorter than that implied by the above trajectory with the equals sign in Equation 9.10, a specific argument is needed to show the equals sign is still valid. In this case, in contrast to Case 1, $H(s_h) < H(s_h\zeta)$. This corresponds to one of those rare situations where an ordered state appears in the trajectory as a fluctuation. Nevertheless, while a shorter programme may generate s_h, no shorter programme can generate both s_h and ζ as, again being reversible, there can be only one path to the combined string $s_h\zeta$. It follows that the shortest programme must be irreversible and the record of the history of the computation must be different from the history generated by the stepping programme that produced the trajectory. The entropy difference between the reversible algorithm and the shortest algorithm is $H(s_h\zeta) - H(s_h)$. This implies $H(\zeta|s_h^*)$ is now greater than zero. I.e. information must be discarded to define $H(s_h)$ without reference to the computational history. As a consequence, a shorter description of s_h implies a specific and longer description of ζ. If the information in bits of this longer description is taken into account it is seen that the overall entropy has not decreased, and if the system is to settle in s_h a greater number of bits need to be ejected to remove reversibility.

Where the number of steps increases and the trajectory takes an ergodic system through its equilibrium to a more ordered configuration the other side of the equilibrium set, $H(s_h\zeta)$ will continue to grow as $H(\zeta)$ grows. However past the equilibrium, s_h will start to become more ordered (i.e. a shorter description exists) as extra bits are transferred from description of s_h to the history as was described in Case 2 above. Indeed past the equilibrium, as the system is ergodic, the shortest computation path may be the one that determines s_h by running backwards in time from the original state.

A somewhat analogous situation occurs when the system passes through a set of states s_h with the same provisional entropy. Again, $H(s_h\zeta)$ grows as the trajectory steps through each state in the set of states but the provisional entropy does not change except when s_h specifies a highly ordered fluctuation from the typical state. In this case $H(s_h\zeta)$ can be separated into $H(s_h)$ and $H(\zeta)$ as specifying s_h provides no information that can be used to specify ζ more simply. This point is used to describe the entropy requirements of a homeostasis in the next section.

A further point needs to be noted. Where a catalyst provides an alternative reversible trajectory, as more programme steps are required to include the catalyst, the history of the process will be greater. In this case the appropriate $H(s_h\zeta)$ will be greater. While the computation with the catalyst will still be reversible, and an equals sign is appropriate for Equation 9.10. However, the algorithm that generates both s_h and the new waste string ζ will be longer than

[4] Here, the possibility that an algorithm stepping going backwards in time might be shorter is ignored. This would only be possible for a state that is generated when the system had passed through equilibrium.

the original algorithm that generate s_h with a shorter history or waste string. The algorithmic entropy is dervived from the shortest (in this case reversible) path to generate s_h, not the most efficient path in a time sense.

Provided one keeps a track of all the entropy inputs and outputs, the system can be deemed isolated and reversibility is maintained while entropy is conserved. The equivalent of Equation 9.10 for the reversible situation where all entropy changes are tracked, then becomes;

$$H(s_h\zeta) = H(s_0) + H(s_h\zeta|s_0^*) + O(1). \tag{9.11}$$

For example, as is discussed in more detail earlier, a stoichiometric mix of hydrogen and oxygen is ordered, as the energy is concentrated in the initial states of the system. Under the right conditions, hydrogen and oxygen will ignite and, if the system is isolated it will settle in an equilibrium set of states involving hydrogen, oxygen and water. The equilibrium temperature is higher than the initial temperature, as the stored energy has diffused through the system, disordering the momentum states. If no information is lost or added (i.e. the system is reversible), there is no entropy change to the total system and, in principle, the system could revert back to the original state. Reversibility is only lost when information corresponding to the history ζ is lost, as happens when excess heat is ejected from the system. In which case, the system's entropy decreases as the disorder in the momentum states is reduced.

9.6 Homeostasis and second law evolution

Section 6.2, provides the requirements for externally driven regulation where the regulation processes act from outside the system. In contrast, this section looks at the requirements for autonomous regulation to counter second law effects. I.e. the requirements for the regulation processes to operate from within the system itself, rather than be imposed externally. Autonomous regulation is essential if a natural living system is to be maintained in a viable regime far-from-equilibrium. The argument here is that autonomous regulation is not a fortuitous outcome of complex interactions of natural laws but, at least in simple cases, and perhaps in many cases, is accomplished by replication processes. Before looking at replication processes as such, the thermodynamic information requirements for a system to maintain homeostasis are outlined.

Information requirements of homeostasis

In order to see how an autonomous process can constrain a system to a stable or homeostatic set of far-from-equilibrium configurations under the influence of the second law of thermodynamics, we first define the viable region of the system. Let this viable region of the system E be a macrostate that contains a set of far-from-equilibrium configurations denoted by ϵ_i. Each of these microstates specifies a configuration in the system's state space all with the same thermodynamic entropy (given the particular resolution of the state space grid).

Provided an agreed zero of entropy is established, allowing for units, the thermodynamic entropy of the macrosate returns the same value as the provisional algorithmic entropy of a typical microstate ϵ_i in the macrostate 5.2 (I.e. the algorithmic entropy of the overwhelming majority of microstates). Section 4, showed the provisional entropy of a microstate consists of the equivalent of the Shannon entropy of the set of all states, and a term that defines the structure common to all the microstates in the set. As a consequence, changes in the provisional entropy can be used to track the thermodynamic entropy flows in and out of the system. In its viable region, the system moves through an "attractor-like" region of the state space where each microstate ϵ_i has the same provisional entropy [5]. This attractor-like region maintains its shape as it drifts through a hypercube where the labels on the variables, but not the type of variables, are changing. If a system is originally in a microstate ϵ_i belonging to the far-from-equilibrium macrostate, the second law processes, acting within the system, shift the computational trajectory away from the viable macrostate E. The system can only be maintained in the viable region if the second law degradation processes can be countered by accessing the stored energy embodied in species (such as photons, atoms, molecules or chemicals) represented by the string σ and, at the same time, ejecting disorder or waste represented by the string ζ. Alternatively, work can create an entropy gradient, as happens for example when a heat pump produces a low entropy sink to absorb excess heat[6].

In a homeostatic set of states, the system moves from one state in the viable region to another as entropy and replacement species enter the system while heat and degraded waste leave. A snapshot of a particular state in the homeostatic stable regime will show a mix of structures; some highly ordered, some that are partially disordered, and some that can degrade no further. For the algorithmic entropy of a typical state in the homeostatic set of configurations to remain constant, the history of the computation that increases at each step must continuously be ejected as waste.

The second law process that degrades the species, and the restoration computation that re-creates order, are best seen as two separate processes. Consider the first stage of the process as a disturbance that drives the system away from the viable region towards a non-viable configuration η_j that also includes the history of the computation. This disturbance degrades the system's structures. From a computational point of view this disturbance can be represented by the algorithm p^*_{2ndlaw}, the shortest algorithm that implements second law processes to shift the system to η_j. In which case, the natural processes are equivalent to the following computation on the reference UTM.

$$U(p^*_{2ndlaw}, \epsilon_i) = \eta_j. \tag{9.12}$$

[5] I dont think this is a true attractor as the labels on the variables change. I.e. over time atom i is replaced with another identical atom j.

[6] As any machine requires an entropy gradient to do work, work is just another way of transferring order from one system to another via this entropy gradient.

As nothing leaves the system at this stage, the computation is reversible, the algorithmic entropy of η_j is given by,

$$H(\eta_j) = H_{prov}(\epsilon_i) + |p^*_{2ndlaw}| + O(1). \tag{9.13}$$

Here the equals sign means that we have represented the second law process by p^*_{2ndlaw} the shortest algorithm that shifts ϵ_i to η_j. I.e. a change of $H(\eta_j) - H_{prov}(\epsilon_i)$ bits.

Without the ability to eject disorder, and without access to concentrated energy or low entropy sources when needed, the system would degrade and over time reach a local or global equilibrium under the second law of thermodynamics. What might be termed a regulation or restoration process counters the effect of the second law disturbance, by the regulation algorithm p^*_{reg}. This process corresponds to a computation on the input string σ_j, that describes the species containing the stored energy, together with the string η_j that specifies the non viable configuration. The net output for the regulation process is the final microstate ϵ_k with the same provisional entropy as ϵ_i and with an extra waste string ζ_l. In general a computation like p_{reg} generates ϵ_k and ζ_l together and therefore there is a high level of mutual information between the two components of the outcome. From Equation 9.12, This regulation process is represented by the computation;

$$U(p^*_{reg}, \eta_j, \sigma_j) = \epsilon_k, \zeta_l. \tag{9.14}$$

From Equation 9.14,

$$H_{prov}(\epsilon_k, \zeta_l) = H(\sigma_j) + H(\eta_j) + |p^*_{reg}| + O(1). \tag{9.15}$$

The $O(1)$ term that represents the bits required to link independent halting subroutines will be ignored. Equations 9.13 and 9.15 can be added, and after rearranging, the number of waste bits, $H(waste)$, can be determined. This is the number of bits that need to be ejected from the system for it to be restored to the original macrostate. This basic requirement for a replicating system to self-regulate becomes,

$$H(waste) = H_{prov}(\epsilon_k, \zeta_l) - H_{prov}(\epsilon_i) = |p^*_{2ndlaw}| + |p^*_{reg}| + H(\sigma_j). \tag{9.16}$$

As one would expect, the entropy of the waste string must include the bits embodied in the second law disturbance algorithm, the regulation algorithm and the bits that have entered the system to provide the concentrated source of energy. All this additional entropy must be ejected if the system is to be returned to the initial macrostate. This can be related Zurek's approach (discussed in section 6.2), which argues that, as $H(\eta_j) - H(\epsilon_i)$ $(= |p^*_{2ndlaw}|)$ bits shift the system away from the stable configuration, the same number of bits needs to be returned to the system if it is to return to the original macrostate. However in the present situation of autonomous regulation, the bits involved in the regulation process are part of the system and need to be accounted for

whereas in the Zurek case, they are envisaged as being external to the system. Equation 9.16 implies that, for autonomous regulation, it is the net decrease in bits due to the regulation process that must equal the increase in bits that enter through the disturbance. I.e.

$$H(waste) - |p^*_{reg}| - H(\sigma_j) = |p^*_{2ndlaw}|.$$

In contrast to the situation where the regulation process is external to the system, autonomous regulation, requires these bits to be tracked as part of a wider system.

If the entropy that flows through the system, and the bits associated with the high grade energy σ_j, were not ejected, bits would build up in the system. A simple example is where an external low temperature sink extracts heat from the system. The low temperature sink is more highly ordered than the system. The absorption of heat or waste from the system by the action of physical laws, is in effect a transfer of order into the system from the environment.

Equation 9.16 shows that the ability of the system to maintain homeostasis does not directly depend on $H(\epsilon_k)$, and therefore the system's distance from equilibrium. Rather, as is discussed later in section 9.6 and implied by the requirements of equations 9.19, the rate at which the second order degradation processes can be countered by the restoration process depends on:

1. the rate at which the concentrated source of energy can enter the system, and

2. the rate at which excess entropy can be ejected from the system.

These two requirements determine the entropy balance for the far-from-equilibrium set of configurations available for the system. The greater the rate that that the system degrades, the further from equilibrium the system is likely to be in order to eject sufficient waste such as heat. Also, the further the system is from equilibrium, the greater its capability to do work, as this depends on the entropy difference between the system and its surroundings.

As the second law programme embodied in the natural laws shifts the trajectory out of the homeostatic set of configurations increasing the entropy by $|p_{2ndlaw}|$, any regulatory programme, embodied in natural laws, such as a replication process, must turn the excess entropy into waste that can now be discarded similar to that described in section 6.2. As Equation 9.16 implies, bits move in and out of the system during the regulation process. This contrasts with Equation 6.1 where the regulation bits do not become part of the system [5]. The end result is still the same.

Many natural systems, such as the photodiode mentioned above, cannot self-regulate unless there are fortuitous circumstances or outside interventions. The hydrogen-oxygen-water mix can be maintained if hydrogen and oxygen resources are available to feed into the system, but a heat input is needed to trigger the combustion process. In general however, in order to regulate, the system must be able to sense or predict the deviation from the homeostatic

configuration and implement a correction process [5]. Excluding human or animal cognitive responses, natural regulation processes must be wired into the system in some way. Here it is shown that replication processes behave like a primitive form of regulation that does not require intelligent external interventions.

True replication processes are able to naturally regulate as, given adequate resources, the drive to create or maintain order is inevitable. Because replication drives a system towards a more ordered configuration, when degradation occurs, natural replication processes, seek to access external resources to create or maintain the order. This is why the production of coherent photons in a laser, the alignment of spins, or the replication of cells in a broth of bacteria, continue far-from-equilibrium. Such replicating systems adjust the numbers of replicated units. In so doing they counter the second law degradation degradation provided resource inputs can be accessed and waste can be ejected, as shown in Equation 9.16. Such replicating systems settle in an attractor-like region of the state space with the same provisional entropy. As resources flow through the system replicas are created and destroyed. Outside the viable attractor region, the trajectory is highly sensitive to its initial state and the system behaviour is chaotic. The capability of replicating systems to regulate suggests that it should be no surprise that replication processes underpin the maintenance of living systems far-from-equilibrium.

The next sections consider how the algorithmic structure of a replication processes and how these processes can restore a system to a homeostatic configuration.

Natural ordering through replication processes

In contrast to an inert system that cannot maintain itself, equation 9.1 shows that replication processes can naturally generate and re-generate ordered structures with low algorithmic entropy, by ejecting disorder and, if necessary by accessing high quality concentrated energy, such as photons etc.. The high quality energy repackages the system, separating the heat and degraded species from the ordered regions allowing natural processes to eject the waste. A simple illustration is the growth of bacteria in a broth of nutrients which eventually settles in a stable situation provided that waste products can be eliminated and nutrients become available to replace the bacteria that degrade. The carrying capacity of the system depends on the rate the nutrients are replenished and the rate at which bacteria die producing waste products that need to be ejected. In order to understand the computational processes that reflect this replication process equation 6.2 can be modified to allow a replication algorithm to embed many micro steps into a larger replicating subroutine. Here the parameter N will define the number of times the replication routine is called to append a further replica to the string "$STATE$" before finally halting. The schematic form of the algorithm is;

$$\begin{aligned} STATE &= r_0 \\ FOR\ I &= 0\ to\ N \\ REPLICATE\ r_I & \\ STATE &= STATE, r_I \\ NEXT\ I. & \end{aligned} \tag{9.17}$$

The final output is $STATE = r_0, r_1, r_2, \ldots r_N$. Again, the replication situation is reversible if nothing escapes the system. The following examples are simple models to help envisage computational processes involved in replication.

- The alignment of magnetic spins or the formation of a block of ice from water are simple example of a replication process. In the magnetic case (see section 9.3), above the Curie temperature, the spins in a magnetic material are disordered, as the thermal energy overrides the magnetic coupling between spins. The crystallisation of a block of ice is similar. The particular configuration of the magnetised spin system (or a block of ice) could be defined by specifying a unit cell of the structure and use a copy algorithm to translate a copy of the unit cell to each position in the lattice to specify the whole. Because the vibrational states are the means by which heat is transferred to the environment, the specification of the instantaneous configuration of the vibrational states needs to be included in the description to properly track entropy flows.

  On the other hand, rather than using a copying algorithm, one can specify the system by a replicating algorithm that implements physical laws to grow the ordered system step by step. At each step, energy is passed from the structure to the vibrational states and then to the external environment. In this case, given the initial vibrational states, the final vibrational states of the ferromagnetic system (or similarly the ice system) arise when heat is passed to the environment during the replication process. In the replicating algorithm approach, these states emerge through the replication process and do not need to be specified separately. If one is interested in comparing the algorithmic entropies of two different physical states, e.g. water and ice, where the laws of nature can be taken as given, the copying algorithm is only likely to be shorter than the replicating algorithm if one chooses to ignore the vibrational and other relevant states that are strictly part of the system.

  Take as the particular example a crystal of ice. If each water molecule in the ice is denoted by r, the ice structure of N molecules is represented by a string $r_1, r_2 \ldots r_N$ where the label denotes the position of each molecule in the structure. However the vibrational states that are coupled with these also need to be specified. As a result, the full specification becomes $r_1, r_2 \ldots r_N, \phi_i(T)$. Here $\phi_i(T)$ represents an instantaneous specification

of the vibrational and electronic states and other degrees of freedom for the of whole system at temperature T.

- Another example that models typical behaviour is a system of identical two-state atoms in a gas where a photon can be emitted from excited atoms (see section 9.3). In contrast to the crystallisation case, stimulated emission requires high quality energy embodied in the excited atomic states to feed the replication process. A symmetry breaking process takes place as the stimulated photons have aligned momentum vectors, rather than being randomly oriented. Stimulated emission can be seen as a photon replication process. Once heat is lost, the macrostate can only be sustained by ejecting heat and accessing energy to populate the atomic states. The macrostate consists of the coherent photon state, a very small number of incoherent photons and most atoms in their ground state. In this situation the photon replicas die and are born as the system settles in an attractor.

  The string that represents the microstate of the system of coherent photons, needs to specify the coherent photon state, the incoherent photons states and the atomic energy levels which interchange energy with the photons as the system moves from one microstate to another all within the same macrostate. Other states of the system such as the momentum states, can be represented by $\phi_i(T)$, noting that, as coherent photons are replenished, the energy that enters the system is passed to the momentum states and then to the environment maintaining a constant temperature.

An important question is whether a living system, such as a tree is best described by the algorithm embodied in its DNA, or a copying algorithm that copies each cell, with its variations, to a different position in the tree. However, in addressing which is the shortest algorithmic description, a consistent entropy zero needs to be defined. This can be achieved by taking the common routines that the laws of nature and specifications of the basic structure of the system, as given. In which case, the replicating algorithm that generates the tree from the genetic code in the seed, must access external resources as inputs, as the tree develops along the lines of equation 9.17. The variation in the cells as the tree grows depends on which genes are expressed. While the DNA is the same, the replicating algorithm calls different sub routines depending on the environment of neighbouring cells and external inputs (such as the availability of water and light). The environment of neighbouring cells in effect shows how the history of the process determines the next state. Consequently a cell in the leaf is expressed differently from a cell in say the root system.

However a routine that copies each cell, with its variations, will usually only be shorter than the one that describes the trajectory if the states in $\phi_i(T)$, which include the instantaneous states of each cell's constituents, are ignored. Specifying the variation fo a cell by calling a routine, based on the history to date, is almost certainly more efficient that specifying each variation as a separate structure. For example, if there are $\mathcal{M}$ variations of N replicating cells,

the particular configuration that needs to be identified from $\mathcal{M}^N$ variations for the copying routine, requires at least an extra $N(log_2\mathcal{M})$ bits. Furthermore, each replicated cell will need to be positioned in space and will also carry kinetic energy that in principle needs to be specified to exactly define the state. The above argument suggests that, for the great majority of cases describing real physical situations, the replicating algorithm, while appearing longer in the first instance, is likely to be the shorter than the copying algorithm given the need to properly specify a biologically complex (i.e. non simple) structure. Nevertheless, if the full number of replicated structures is to emerge, entropy as heat or waste must ultimately be passed to the environment. In principle a growing tree is a reversible system but, throughout the growth process, waste exits the system. As the waste drifts away reversibility becomes highly unlikely in the life of the universe. With these understandings it becomes possible to track algorithmic entropy flows and therefore thermodynamic entropy flows in real systems.

Replicating algorithms

Simple replication processes, such as the crystallisation of ice, or the alignment of spins, are formed by ejecting excess entropy, whereas most replicating and ordering processes require the input of high quality or concentrated energy in addition to ejecting waste. In the general case, the replication process can be seen as a computation that acts on an input string σ, that represents the resource string σ $(= \sigma_1, \sigma_2 \ldots \sigma_N)$ embodying concentrated energy as wll as the structure r_1 which is to be replicated. The input can be represented by $i = r_1\sigma$, while the output is the system of replicated structures together with a waste string. If the replication cycle loops N times, N replicated structures will emerge.

$$U(p_N, r_1, \sigma) = r_1 r_2 r_3 \ldots r_N, w(T'). \tag{9.18}$$

In this process the number of replicated structures r_1, r_2, etc increases where the subscripts identify successive structures. The string $w(T')$ is the history or waste string that ultimately must be ejected from the system to inhibit reversibility, and to allow the ordered structure to emerge. Other states that contribute to the algorithmic entropy of the replicated system, originally at temperature T, are essentially unchanged at the end of the process of replication. Originally, these states, which can be represented by $\phi_i(T)$, become $\phi_j(T')$ where $T' > T$. This happens when, for example, the temperature rises changing the vibrational states. As the waste string $w(T')$ is ejected, $\phi_j(T')$ separates into $\phi_k(T) + w(T')$ leaving $\phi_k(T)$ behind. Because the before and after contribution to the provisional entropy of the states represented by $\phi_i(T)$ is unchanged, i.e. $|\phi_k(T)| = |\phi_j(T)|$, these states can be ignored. Overall, with the injection of waste, the provisional entropy is lower and the structure is more ordered.

The following argument tracks the degradation process and the restoration process from both a computational view and a phenomenological point of view. We assume the system is initially in a stable viable configuration denoted by ϵ_i that specifies the N repeated structures and also the other degrees of freedom associated with the system. The set of stable configurations all have the same provisional entropy. This implies the number of replicated structures is fixed, nevertheless, there are many different ways the thermal energy can be distributed amongst the other degrees of freedom such as the momentum or vibrational states of the system. The overwhelming majority of possible configurations will have the same number of replicated structures.

The second law degradation process is given by equation 9.12 i.e.

$$U(p^*_{2ndlaw}, \epsilon_i) = \eta_j$$

. This shifts the system to the non viable configuration η_j. From the phenomenological perspective, the replicated structures are destroyed at a rate given by αN that depends on the number of replicated structures present. The regulation computation given by equation 9.14 accesses the resource string σ to regenerate the structure. The rate at which this can happen is usually limited by the rate at which the resource structures enter the system. This is taken to be β structures per unit time. From the computational point of view the non-viable configuration $\eta_j = r_1, r_2, \ldots r_{N-M}, w(T')$ where M replicated structures have been destroyed, or decay, creating the waste string $w(T')$ that captures the history of the process. The corresponding restoration programme $p_{reg,M}$ that loops M times to regenerate the M structures and ejects the waste is

$$U(p_{reg,M}, \eta_j, \sigma_1, \sigma_2 \ldots \sigma_M) = r_1, r_2, \ldots r_N, w(T''),$$

The overall waste ejected includes the bits causing the replicas to decay; the bits in the regulation algorithm $p_{reg,M}$ that restores the number of replicated structures from $N - M$ to N; together with the bits specifying the extra resources that have been added. I.e. all extra inputs need to be ejected to stop the build up of the waste in the system.

The growth characteristic of the replication algorithm is similar to the discrete version of the logistic growth equation. However, the logistics growth equation fails to adequately track the resource flows for the replication process. A more realistic approach that identifies the replication dependence on the resource supply is the Holling type-II functional response which is similar to the Monod function or the Michaelis-Menten equation. This equation form can be used to describe the interaction of a predator and its prey (which in this case is replicator-resource interaction). As mentioned above, the equations below include a term αN to capture the decay or destruction of replicated structures due to second law effects in the growth equation. The assumption is that the removal of structures (i.e. their death) at any time is proportional to their number. Also the constant flow through of the resources β is needed to drive the replication process. While a further assumption is that all the computational waste material can be ejected, as is mentioned later, the inability to

eject waste efficiently can limit the replication process. With this understanding, let N represent the number of replicated structures and let σ represent the number of resource units that are required to feed the replication process. In which case,

$$dN/dt = \frac{\hat{r}\sigma N}{(b+\sigma)} - \alpha N$$
$$d\sigma/dt = -\frac{\hat{r}\sigma N}{\gamma(b+\sigma)} + \beta. \tag{9.19}$$

The parameter γ represents the number of replicated structures produced from the consumption of one resource unit, i.e. $-\gamma\Delta\sigma \approx \Delta N$. The maximum growth rate is the initial growth rate $\hat{r}$ when $\gamma\sigma >> N$ whereas b is the resource level at which the growth rate is halved when $\sigma = b$. The conditions for a stable long term homeostatic state occur when the two derivatives approach zero. In which case the number of replicated structures increases, tending to a long term value $N_\infty = \gamma\beta/\alpha = K$ where K can be identified as the carrying capacity of the system.

Equations 9.12 and 9.18 are the computational equivalent of the replication process captured by Equation 9.19 . In each unit of time β resources flow into the system, maintaining the length of the resource string to σ. In the stable configuration, the resources are just sufficient to replace the αN replicated structures that decay over that period. From a computational point of view, the long term description of the replicated structures becomes $r_1 r_2 r_3 \ldots r_K$ with $K = N$. The length of the algorithmic description of the string of replicated structures produced by a copying routine would be close to $|r_1 r_2 r_3 \ldots r_K, | = |r_1^*| + log_2 K$. I.e. one replicated structure needs to be specified by r_1^* and the specification repeated K times. Here the asterisk indicates this is the shortest description of the replica. On the other hand, if the length is derived from the replicating algorithm rather than the copying algorithm, the dominant contribution would come from $|r_1^*| + log_2 N$. The optimum number of replicated structures will emerge when $N = K$ and $log_2 N = log_2 K$.

Replicated structures can be structurally slightly different if the inputs to the replicating algorithm vary. For example an eye cell is generated by the same algorithm as a heart cell but different genes are expressed because of the specific environment surrounding the eye cell provides different inputs to the generating routine. Flexibility in function is achieved with little increase in algorithmic entropy as the variation occurs when external triggers call different routines.

Replicating systems act as pumps to eject entropy to form the ordered state. Entropy ejection is efficient in reducing entropy gradients and degrading high quality energy. When, for example, water crystallises through a replication process, rather than cooling slowly, a spike of latent heat is generated which passes to the environment more rapidly because of the increased temperature differential between the system and the environment. Similarly when high

quality energy is converted to waste species in the replication process, the ejected waste is more efficiently transferred to the environment compared with other processes. This is consistent with Schneider and Kay [88] who argue that the emergence of biologically complex structures is a move from equilibrium. As natural systems resist any such movement, living systems are more effective than inert structures in dissipating energy, by degrading high quality energy and in countering entropy gradients. Schneider and Kay show that evaporation and transpiration of the constituents of an ecology are the major forms of energy and entropy dissipation. Here one would see that replication is the driver of more efficient dissipation at the system level.

While the real world replication is a complex parallel computing process, the approach described above captures its essentials. Provided the creation rate of replicas exceeds their decay rate, the number of replicas will grow until the system settles in a stable set of configurations. In this homeostatic situation, the carrying capacity of the replicated structures is constrained either by the rate at which the waste can exit the system, or by the rate the resource inputs are available to replace decayed replicas. Many living systems also need to do work, increasing the thermodynamic cost of maintaining the system away from the global equilibrium. While systems further from equilibrium have a greater capacity to do work, they need to be more efficient in their use of the resources that are inputs to the computational process. They also need to be more efficient in ejecting disorder that, from a computational point of view, is the output string that exits the system. As is discussed below, evolutionary type processes acting on a system of replicated structures select variations that are more efficient in this sense.

9.7 Selection processes to maintain a system in a viable configuration

Selection processes are key to the emergence of structures better adapted to the environment. This section looks at selection processes from a computational point of view, showing that diversity in the replicated structures, while better maintaining the system distant from equilibrium in an adverse environment, also drive the emergence of different structural types. The section argues that not only can a system of replicated structures counter second law degradation processes by replacing lost species, if there is sufficient variety among near identical replicas, adaptation processes can counter more virulent disturbances impacting on the system. For this to happen, variations in the replicated structure must emerge at a time fast relative to the external environmental changes. This adaptation can be viewed as a form of regulation as, in general, organised systems consisting of variations of a simple replica are more stable against change. They have an increased ability to maintain themselves in a far-from-equilibrium situation as the set of viable configurations is larger. However it is a viable set that is expanding its horizons as diversity increases overtime. Variation primarily arises through changes to the genetic code by

copying errors, gene transfer, and more general mutations. New resources flowing into an open system of replicas are in effect additions to the input string expanding its state space. Similarly, resources flowing out of the system lead to a loss of information and a contraction of the state space. Consider the replicator-resource dynamics where a disturbance threatens the viability of the system of replicated structures. The disturbance, which could involve an inflow or outflow of species or energy, will vary the computational paths undertaken by the system. Some variations of the replica may become less likely, while others become more likely in a changing environment. Those best adapted will dominate the replica mix. Examples might be where variant of one tree species is better able to cope with drought or the emergence of drug resistant bacteria. The phrase "best adapted" in these cases generally means that, from an entropy flow point of view, the system is more efficient in using resources in the sense the system is easier to maintain far-from-equilibrium. This resource efficiency will be termed "entropic efficiency". In essence, a system of replicated structures that has sufficient diversity increases the ability of the system to survive in an adverse environment. Nevertheless, it needs to be recognised that as selection processes reduce diversity in the short term, the algorithmic entropy lowers. But as diversity inevitably reappears over time, one can argue that these adaptive processes determine a long term trajectory leading to the evolution of the system.

The insights of Eigen and his co-workers [49, 50] on the emergence and growth of self-replicating polynucleotides can be applied to more general replicating systems. In the polynucleotide case, copying errors provide variations in the genetic code, some of which through selection processes, dominate the molecular soup. Interestingly, Eigen shows that a distribution of closely interrelated polynucleotides, which he terms a "quasi-species", dominate the mix rather than one particular variant. New structures can emerge if there is sufficient diversity among near identical replicas, as adaptation processes can counter more virulent disturbances impacting on the system.

A simple model, based on predator-prey dynamics (or in this case the equivalent replicator-resource dynamics), provides useful insights on how variation and selection processes enable a system to adapt when two variations of a replicating structure compete for limited resources. As perhaps is no surprise, just as for the single replica case, a simple model where two variants compete for limited resources shows that the variant with the highest carrying capacity will dominate the mix. As is shown below, the carrying capacity depends on the efficiency of the replication process in the sense of efficient use of resources, and also the variant's capability to withstand the onslaught of second law processes. However in this case, the carrying capacity is not an external parameter and coupled versions of the Lotka-Volterra equations are inappropriate. Eigen's approach [49] is more realistic. This establishes the requirements for a stable equilibrium, where there is a constant flux of monomers corresponding to the resources that feed the process. His result is essentially the same as the following simple discussion, which is based on the simple predator-prey model addressed in section 9.6. In the approach here there are two variants of the

basic replica with numbers N_1 and N_2 competing for the resources σ that feed the replication process. The first two of the following equations capture the growth rate of the two variants, given their respective decay rates α and the additional parameters c_1 and c_2 that represent the impact of one variant on the other. The last equation captures the rate the resources change, given the resource consumption by the variants and the constant inflow β of new resources.

$$\begin{aligned} dN_1/dt &= \frac{\hat{r}_1 \sigma N_1}{(b_1 + \sigma)} - \alpha_1 N_1 - c_1 N_1 N_2 \\ dN_2/dt &= \frac{\hat{r}_2 \sigma N_2}{(b_2 + \sigma)} - \alpha_2 N_2 - c_2 N_1 N_2 \\ d\sigma/dt &= -\frac{\hat{r}_1 \sigma N_1}{\gamma_1 (b_1 + \sigma)} - \frac{r_2 \sigma N_2}{\gamma_2 (b_2 + \sigma)} + \beta. \end{aligned} \tag{9.20}$$

While these equations are similar to the equations for predator-prey relationships (see [4]), the last equation has a term β to indicate resources are supplied to the system at a constant rate (see [49]). Another assumption is that the waste generated by this process is able to be ejected by the system.

The requirement for a non-periodic stable solution is that all the derivatives become zero. For nearly identical replica variants, dN_1/dt and dN_2/dt are positive and approach zero together. In this case, the requirements are similar to the single replica case outlined in 9.6 and again the system settles in a region determined by the rate the resources are consumed by the replication processes. However where say $dN_1/dt > dN_2/dt$, because $\gamma_1 > \gamma_2$ and/or $\alpha_2 > \alpha_1$, both subspecies grow initially but eventually N_2 ceases to grow and ultimately decreases as the term $c_2 N_1$ drives dN_2/dt through zero. Once this happens, the population of subsystem N_2 goes to zero while N_1 trends towards its maximum value. Assuming variant 2 becomes zero, the limit of N_1, i.e. $N_{1\infty}$ is $N_{1\infty} = \gamma_1 \beta / \alpha_1$. This is K, the carrying capacity of the variant dominating the system. The dominating variant is the one with the highest replication efficiency and the lowest decay rate due to second law effects (i.e. the highest γ/α). As this is the variant needing the lowest entropy throughput, resource fitness corresponds to what will be termed "entropic efficiency". The result can be generalised to more than two variants as shown by equation II-47 of Eigen [49]. The above equation gives essentially the same result as Eigen for the stable long term configuration. However Eigen's approach shows that a closely interrelated quasi-species dominates the mix rather than just one variant. In the Eigen approach ϕ_M is set to equal the average growth rate and death rate over all the variants. Here ϕ_M corresponds to $\gamma\beta$.

One can argue that this is a like a regulation process which selects the most efficient replicating variant to return the system to a state marginally different from the original. This allows the system to survive the threatening disturbance. However, there is a trade-off. While increasing the variety in the replicated structures increases the entropy of the system, making it closer to equilibrium, variety stabilises the system against environmental change. For a

system of N identical structures, such as a tree grown from a seed, the algorithmic representation of N in the generating algorithm will be $\approx log_2 N$ and the provisional entropy would be something like $H(tree) + log_2 N$. However if there are $\mathcal{M}$ variants of the seed representing the variants in the genetic code, a system of N structures now requires an algorithm that identifies the particular variant for each structure. As there $\mathcal{M}^N$ possible distributions of variants, $Nlog_2\mathcal{M}$ replaces $log_2 N$ in the provisional algorithmic entropy measure and the provisional entropy is something like $H(tree) + Nlog_2\mathcal{M}$. While diversity increases the provisional algorithmic entropy, if there are few variants, this increase will be small relative to the overall generating algorithm. If the variation is coded at the genetic level rather than at the structural level, as has been argued, in the competitive situation for most living systems, the selection processes will operate at the genetic level. Just as genetic algorithms mimic real world selection processes, the algorithmic approach sees these real world computations as genetic algorithms. While chance rather than physical laws, may be the determining factor in many evolutionary selection processes, entropic efficiency may still be important in semi- stable environments. The question is whether such entropic efficient selection processes, considered here, are widespread.

Furthermore, the situation where a process that selects for entropic efficiency might be perceived as a process that selects for behaviour. For example, separated structures that have some movement might be selected to increase movement enabling the structures to move together or apart to control the overall temperature. Another example might be those bees that have developed the ability to warm their hive by beating their wings when the temperature drops. Presumably, this behaviour driver arises as regulation has been built into the genetic code's software through selection processes that operate on individual bees. The effect is that entropic efficiency of the whole system of bees has increased. What might be seen as adaptation, or "survival of the fittest" at the biological level can be interpreted as a drive for entropic efficiency at the algorithmic level. This is a drive that uses replication processes, based on natural laws, to regulate the system. In general terms, variation in the core replicated structure can be selected by the environment to provide an autonomous response to external factors and so can be seen as quasi-regulation process.

However, while the system selects the variant that optimises the entropic efficiency of each replica, the overall entropy exchange with the environment does not decrease as the number of structures in the system increases to counter any gain at the level of an individual structure. As is discussed later, this is consistent with Schneider and Kay [88] who show that as biological complexity increases, the overall system becomes more efficient in degrading high quality energy.

While variation increases the provisional algorithmic entropy, if there are few variants, this increase will be small relative to the overall generating algorithm. Selection processes that favour one particular variant of the replica reduce the provisional entropy of the system over the short term. Over the

long term further variations may well increase the provisional entropy for future selection processes to act on the system.

What does this mean in terms of the regulation equation 9.16? The input string σ includes the nutrients to maintain the replicas and the waste needs to be ejected perhaps to the lower temperature sink in the environment. The principles are captured in a set of bacteria in a nutrient mix that exists in a set of viable configurations by accessing the nutrient and ejecting waste heat and materials to the external, cooler environment. Clearly if the nutrients become unavailable, or the temperature of the environment rises or an antibiotic enters the bacteria food chain, most bacteria will die and those better adapted will survive and dominate the population. From a computational point of view certain computations terminate whereas others do not.

Adaptation and interdependence of replicas in an open system

Open systems of replicated structures can become interconnected and interdependent leading to more efficient resource use. Such interconnections can emerge when the waste string ζ of one system becomes the resource input σ in another. Fewer resources are needed to maintain the combined systems. The predator prey is one such relationship. The stored energy in one system (the prey) is used as the resource input to the other. The predator is just a highly efficient method of degrading the high quality energy embodied in the prey. Alternatively, mutualism can occur when two different species enhance each other's survival.

Interdependent systems tend to stabilize each other and, as a consequence, the far-from-equilibrium coupled systems become more viable as the overall entropy cost of maintaining the system off equilibrium is less. Replicas may also share resources, as happens when, for example, total heat loss is reduced as replicas cluster. The situation where one set of constituents enhances the survival of another can be captured by Equation 9.20 by making c_1 negative and c_2 zero. With appropriate values of the parameters the coupled systems can settle in a stable population regime. Furthermore, where the overall temperature becomes higher, the system has more capability to do work.

A simple example is where photons exit one laser system to create a population inversion in another laser system with the result that less information is lost to the environment. Where overall resources are constrained, and there is sufficient variety in the second system (e.g. sufficient line width overlap between the lasers) this system can adapt by settling in more restricted areas of its state space. In contrast to two independent systems, as the algorithmic specification of the combined system involves the sharing of information between routines, fewer bits need to enter and therefore to be ejected from the coupled structures. As the coupled systems co evolve by using resources more efficiently, in a resource constrained environment the coupled systems are more stable against input variations. Mutualism, or interdependence between systems of replicas increases the order and therefore shifts the organised system

further from equilibrium. Where this interdependence increases the entropic efficiency of the systems under resource constraints, interdependence is more likely to eventuate.Their mutual attractor region will not drift through state space at the same rate as similar, but uncoupled, systems. However, as the population of the coupled structures increases, the overall degradation of the input resources is both more rapid and more complete than would otherwise be the case. Nevertheless, where such structures are possible, they would seem to be likely.

Nested systems and degree of organisation

Structures that are nested within other structures exhibit a particular form of interdependence that increases the overall order, as the nested structures can be mapped algorithmically to a nested set of subroutines that pass information to each other. As a consequence, less waste or information is ejected and the nested structures are more algorithmically efficient. Many living systems consist of nested substructures where the substructures are often variations of a basic unit. For example cells themselves can be nested in an organ such as an eye, while the organ is nested in a higher structure. Variations in the substructures, such as cells, can occur when genetic switches express certain genes in preference to others. The algorithmic equivalent of these switches corresponds to information passed from a higher routine to the generating subroutine. This may come from a neighbouring cell, as is the case when a particular set of cells produce an organ such as an eye. Alternatively the information may be inputs from the external environment. Nesting also allows specialisation of function, with little increase in algorithmic entropy, particularly if the output of a nested algorithm depends only on the information inputs. This allows the pattern of variations to emerge as the structure evolves by a generating algorithm similar to Equation 6.2. As has been mentioned in section 6.3, the algorithm that describes the trajectory that generates the structure, such as the algorithm embodied in the seed of a tree, is likely to be shorter than one that describes the structure in its final form unless significant information can be ignored.

Chaitin's [28] concept of 'd-diameter complexity', which quantifies order at different levels of scale, is helpful in understanding nested structure. Imagine that we could only observe a tree at the smallest level of scale, unable to recognise any order at a larger scale. When large scale order is not perceived, the tree can only be described by a lengthy algorithm that specifies the structure at the molecular level and how the molecular structures combine to form the tree.

Figure 9.1 which illustrates a nested system of replicated structures. Here the smallest scale is denoted by denoted by d_n where n is large and d_n is small. At this scale, as no pattern is discernible, the system can only described by an algorithm that specifies what appears to be a random arrangement of fundamental structures. The provisional algorithm specifying the arrangement would be extremely long and the algorithmic entropy would be indistinguishable from that of a local equilibrium macrostate.

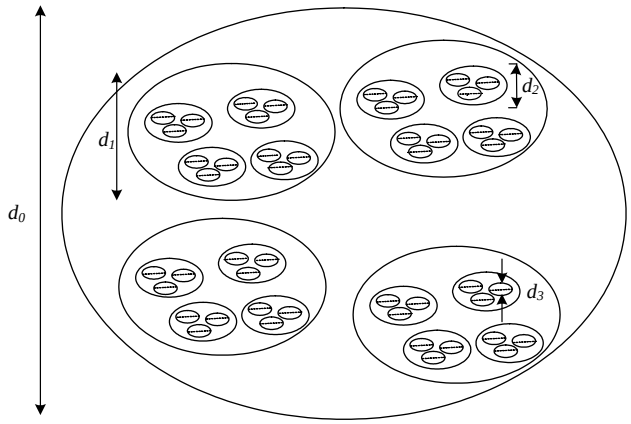

Figure 9.1: Nested structures at scales d_0, d_1, d_2 and d_3.

However, observing the system at a higher level of scale allows some structure to be discerned and the algorithm specifying the system becomes shorter. At an even higher level of scale, cells might be recognised. In which case, once the cell is defined, the overall algorithm becomes much shorter, decreasing the algorithmic entropy. This is because the structure can be more simply built by repeating the cellular building blocks.

Using this approach, Chaitin [28] quantifies the degree of organization (D_{org}) of structure X at scale n as the distance the system is from a local equilibrium which corresponds to a description at the smallest level of scale where there is no recognisable pattern. The degree of organization, is a variant of Kolmogorov's "deficiency in randomness' using self-delimiting coding. The degree of organisation D_{org} at scale n with self-delimiting coding is given by:

$$D_{org} = H_{max}(X) - H_{dn}(X).$$

H_{max} corresponds to the logarithm of the number of possible states where no structure can be discerned. Structure is identified with increasing scale and the generating algorithm shortens reaching the maximum at the largest scale which identifies the maximum value of D_{org} .

Figure 9.2 illustrates how the algorithmic entropy varies with scale parameter d. In the figure, d_2, d_1 and d_0 denote the scale dimensions from figure 9.1 with $d_n << d_2 < d_1 < d_0$. Let H_{d0} represent the minimum value of the algorithmic entropy of the system, based on its shortest description at the largest scale d_0 [47]. The nested structures can be described by interconnected subroutines. However when the large scale structure is suppressed the system is perceived as a collection of sub-structures, such as a collection of cells or, at smaller scale, a collection of molecules. The corresponding algorithmic description must now specify each substructure at this scale and how the substructures are to be assembled. As the large pattern is lost, the entropy increases. At the greatest scale, $H_{d0} << H_{d\infty}$, where d_∞ is the smallest scale.

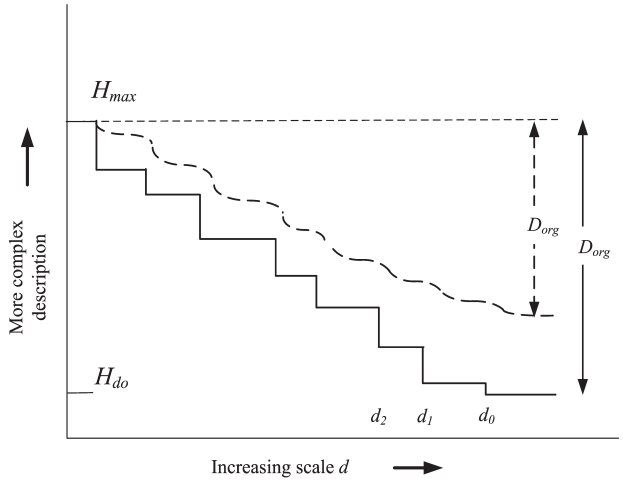

Figure 9.2: Variation of d-diameter complexity with scale; — nested replicators with no variation; – – – – – nested replicators with variation; ---- No organization at any scale.

In figure 9.2. The stepped bold line shows an ideal system where all the nested systems at each level of scale are identical replicated structures. The entropy remains about the same until a lower level of scale suppresses structure. As the scale parameter d decreases from d_0 to d_∞ the entropy increases in well-defined steps. However where variations in the replicated structures occur, as happens with different cell types at the same scale, the steps are smoothed in the dashed line. Diversity increases the provisional entropy at each level of scale. When no organization at all exists, the scale is irrelevant, and the algorithmic entropy is the same at all levels of scale as shown by the dotted horizontal line, H_{max} in figure 9.1.

The implication is that where an algorithm, such as that embodied in DNA contains subroutines that can be switched depending on a particular input, the diversity increases with only a small increase in the algorithmic entropy. As software variation occurs at lower levels of scale, it would appear to be algorithmically more efficient to generate the variation through the software (e.g. variation in DNA) rather than independently specifying similar structures. It is plausible to conjecture that the nesting of structures, decreases the entropy in a way that more than compensates for the entropy cost of building in diversity in the replicated structures [48]. Indeed, this may be an inevitable consequence of selection processes acting on structures. Identical replicated structures with high D_{org}, probably have less ability to adapt to change than systems with variation or greater diversity. The cost of allowing variation is a lower D_{org}.

Observationally it appears that, in comparison to simpler structures, a system of nested subsystems (such as a forest ecology nesting species, with species nesting organs, and organs nesting cells) is more viable far-from-equilibrium, as

the waste of one part of the system is the resource input for another. Overall, there is less need for waste to be ejected per nested unit because of the mutual dependencies and the efficiency in ejecting disorder (such as heat) is higher. As selection processes favour entropically efficient replicating structures, the reduced dependence on external resources means the carrying capacity of interdependent nested systems is higher than systems where interdependence does not exist. The interdependent system is more likely to emerge when resources are limited. These factors suggest that such a system searches for entropically efficient configurations to maintain survival of replicated structures. This tends to build interdependence and, in so doing, drives the system further from equilibrium.

Perhaps life on earth is more entropically efficient at the structural level than no life, as life, consisting of interdependent and nested replicating systems, creates its own autonomous capability to survive and adapt. The caveat is that a concentrated source of energy such as photons from the sun must be accessible, and waste must be able to be ejected. However there is a cost as life drives the whole system further from equilibrium at the expense of being a more efficient dissipator of energy. While resource use is more efficient in nested systems, more resources are consumed as the population of different replicated structures grows to more efficiently process high quality energy. This is consistent with Schneider and Kay [88], who argue that a system of hierarchical structures, such as an ecology, is more efficient in degrading high quality energy than simpler or inert structures. At each step in an interconnected hierarchy, waste ejected by one system can be degraded by a lower system in the hierarchy. This happens when insects, fungi or bacteria degrade plant material far more rapidly than would happen if driven by non-living processes. In a nutshell, individual replicated structures seek entropic efficiency, but at the system level, the population rises to increase overall dissipation. Hierarchical systems make better use of the resources, at the expense of driving their universe more quickly to equilibrium.

9.8 Summary of system regulation and AIT

Algorithmic Information Theory identifies the algorithmic entropy or the information content of a system by the minimum number of bits required to describe the system on a UTM, while Landauer's principle [71] holds that the erasure of one bit of information corresponds to an entropy transfer of $k_B ln2$ to the environment. Together these understandings provide a consistent way of looking at the thermodynamic cost of maintaining a system in stable set of configurations distant from equilibrium [109]. As only entropy differences have physical significance, the number of bits needed to shift a system from one configuration to another, multiplied by $k_B ln2$, corresponds to the equivalent thermodynamic entropy change of the system. In thermodynamic terms, where a disturbance or a second law degradation process, threatens the viability of a system by increasing the algorithmic entropy by H bits or equivalently the

thermodynamic entropy by $k_B Hln2$ Joules per degree Kelvin, the thermodynamic cost of restoring the system is to compensate for the change by removing $k_B Hln2$ Joules per degree Kelvin.

Furthermore, it is argued that, as a system of replicated structures can be described by few bits, natural replication processes use all the available resources to generate simple ordered structures with low algorithmic entropy. This perspective allows one to recognise that a natural replication process can be envisaged as computation on a real world UTM that is able to generate order far-from-equilibrium. Not only can replicating processes generate order, these processes can self-regulate or self-organise as they can naturally compensate for second law degradation by accessing external resources as inputs to regenerate degraded replicated units while ejecting entropy as waste, fulfilling the requirements of Landauer's principle.

When there is a changing environment, and there is diversity among replicating units, selection processes act as an autonomous quasi-regulation process. The replica variant that persists is likely to be the one that is more entropically efficient in terms of its use of resources and its capability to eject waste. In these cases, environmental fitness carries with it a requirement to optimise the entropic efficiency of the structures in the system. Entropic efficiency means the entropy cost of maintaining the system is minimised, and the system adapts by settling in a far-from-equilibrium configuration that is as close to reversible as possible. However while selection processes do allow a system to return to something close to the original configuration, overtime the system drifts or evolves to new configurations as new variations appear.

Similarly, where interdependence between different replicated structures is possible, simple calculations suggest that selection processes will favour the variants that exhibit interdependence. As interdependence is more entropically efficient at the structural level, their numbers increase. Hierarchies of replicated structures would seem likely to emerge as selection processes favour those that waste fewer resources. But as the numbers of such structures increase the overall energy dissipation is higher and, while the hierarchical system persists, it hastens the degradation of the high quality concentrated energy that feeds, the living process. As Schneider and Kay show [88], an ecology consists of myriads of self-replicating efficient units but, because species lower in the food chain consume the waste of higher species, the overall degradation of the system is greater than would be the case for a non-living system, or even a simpler living system. Living systems hasten the death of the universe.

Where a system needs to do work, it must be maintained further from equilibrium and again replicating process can maintain the system in a suitable stable configuration.

The results are general and can in principle apply to any system. However in many real world situations there will be insufficient understanding of the details of the system to quantify the cost of maintaining the system far-from-equilibrium. Nevertheless, the approach may well be useful in understanding incremental changes to real systems and provide broad descriptions of some system behaviour. While there is more to a living system than replication

or entropy flows, just as energy constraints determine system behaviour, so entropy requirements constrain structural possibilities.

Chapter 10

AIT and Philosophical issues

10.1 Algorithmic Descriptions, Learning and Artificial Intelligence

Algorithmic Information Theory provides some practical insights into learning and artificial intelligence. One can take an intuitive view of learning as a process by which an agent, when given data associated with some phenomenon, can make sufficient sense of the data to partially predict future outcomes. Learning is impossible where the observations show no pattern or structure. As these observations cannot be compressed; each future outcome is a surprise. Leibnitz ([36] has argued that the best theory compresses data more effectively than alternatives. As a consequence, the better the theory, the better the learning that takes place and the better it can predict future outcomes. As the fields of learning and artificial intelligence are specialist disciplines, only the flavour of the algorithmic insights can be outlined here. Hopefully those who are interested can delve deeper into the topic.

Two lectures by Solomonoff [95, 96] review the connection between compression, learning and intelligence. Learning can be interpreted within an AIT framework. In particular, the outcome of a learning process is as effective as the Minimum Description Length procedure. As has been outlined in section 7, the Minimum Description Length (MDL) approach argues that the best fit to observed data is found by the physical model that provides the most compressed version of the data. It was pointed out the approach satisfies both the simplicity embodied in Occam's razor, and the breadth of Epicurus' principle, as the aproach seeks the simplest explanation without rejecting possible causes. However there are some deeper questions surrounding MDL. Vitányi and Li [104] have pointed out that data compression using MDL is almost always the best strategy for both model selection and prediction, provided that the deviation from the model is random with respect to the envisaged probability hypothesis. What this means is that to be effective, there can be no identifiable order in the deviations. As was mentioned in section 7, the ideal approach to MDL can be derived from a Baysian approach using the algorithmic universal distribution

of section 3.8 as the prior distribution. With this universal probability, the MDL approach minimises the sum of two terms. The first term characterises the model by the universal distribution, which is effectively the algorithmic description of the model, and the second codes the deviations from the model by a code with length equal to the logarithm of probability of that deviation; i.e. a code that satisfies Shannon's noiseless coding theorem 3.3. Barron and Cover [6] apply the approach to selecting the most appropriate probability distribution to explain a set of random variables drawn from an unknown probability distribution. They show that the two part MDL approach will eventually find an optimum data distribution from the class of candidate distributions. However the approach is one based on having an effective strategy to find potential models, rather than having any belief about the simplicity of the true state of the world. In contrast to the original Bayesian approach, MDL makes no assumptions about the underlying distribution, or any subjective assessments of the probabilities. As Grünwald points out [60], MDL is asymptotically consistent without any metaphysical assumptions about the true state of things. Furthermore, the practical MDL approach is an effective method of inductive inference with few exceptions.

Vereshchagin and Vitányi [103] show that the structure function determines all stochastic properties of the data: for every constrained model class it determines the individual best fitting model in the class, irrespective of whether the "true"model is in the model class. In this setting, this happens with certainty, rather than with high probability as is in the classical case. It becomes possible to precisely quantify the goodness-of-fit of an individual model with respect to individual sets of data.

Hutter [62, 63, 65] combines the idea of universal sequence prediction with the decision theoretic agent by a generalised universal semimeasure. Induction is optimised when the agent does not influence the outcome, whereas combining this with decision theory allows the agent to interact with the world. The model he proposes is not computable, but computable modifications of it are more intelligent than any time or space bound algorithm.

10.2 The mathematical implications of Algorithmic Information Theory

Chaitin [37] argues that, as there are mathematical statements that are true but not provable, mathematics should be taken to be quasi empirical, rather than the usual *a priori* approach. This would allow new axioms to be added to the formal mathematical structure where there is some evidence to support either the truth or falsifiability of the unknowable statements. Chaitin [35] argues that Einstein saw mathematics as empirical, even suggesting that the integers are a product of the human mind. Chaitin also argues that, while Gödel, with his Platonic world view, saw mathematics as an *a priori* activity, Gödel does state that: "If mathematics describes an objective world, there is no reason why inductive methods should not be applied as the same as in physics".

This leads Gödel to argue that a probable decision about the truth of a new axiom is possible inductively. Where such a possible axiom has an abundance of verifiable consequences and powerful impacts, it could be considered as an axiom in the same sense that a hypothesis is accepted physical theory. The implications are that mathematics, by judiciously selecting axioms that appear to be true for good empirical reasons, but which are not yet provable can decide "truth" empiically.

10.3 How can we understand the universe?

Those researchers who see the evolving path of the universe as a computational process, raise the possibility that the universe is itself digital computer. The first to do so was Leibniz according to Chaitin [36], as Leibnitz believed in digital physics. More recently such notables as John Wheeler (quoted in [29]) coined the phrase 'It from bit', while Ed Fredkin [54, 53] sees the fundamental particles of the universe as processors of bits of information. At the Planck scale of 1.6×10^{-35} m, quantum effects dominate, and the universe would seem to behave like a quantum computer almost by definition.

Seth Lloyd [79] (see also [78]) explores the idea of the universe as a computer as it processes information in a systematic fashion. He suggests the universe may perform quantum computations where every degree of freedom registers information and the dynamical interactions between degrees of freedom process information. Lloyd puts bounds on the processing capability of the universe. If gravitation is taken into account, the total number of bits is $\approx 10^{120}$ and operations $\approx 10^{120}$ are simple polynomial functions of the constants of nature, h, c, G and the age of the universe. The number of operations within time horizon t is the Planck time, where gravitational effects are of the same order as quantum effects. Lloyd looks at a matter dominated universe and a radiation dominated universe, and has found that the formulae for describing the operations and the number of bits are about the same in both cases. It is not the purpose of this work to delve deeply into the points Lloyd makes, but to point readers to the possible ramifications of seeing the universe in such a way.

Even if the universe is a quantum computer, the strong Church Turing thesis implies that under the circumstances, where there are no infinities in the universe, every finitely realisable system can be simulated by a finite Turing Machine [45]. However there are arguments against a digital physics based on continuous symmetries and the implication of hidden variables [1].

Even so, it would seem reasonable to assume that above the Planck scale a digital computations may effectively describe physical processes. The internal states of the computer can be considered discrete to within quantum limits. Similarly there is little difficulty in envisaging the universe as a computer that steps through a trajectory in its reasonably well defined state space. For example, the behaviour of say a gas in a container can be seen as a computing process when collisions move the system from one microstate to another.

Paul Davis in the 5th Miracle (page 95 [44]) states that the universe has the

same logical structure as a computer. He argues from Algorithmic Information Theory that physical laws show pattern, and therefore are information poor in the sense that the laws are algorithmically simple. The data embodied in laws is assumed to be compressible. On the other hand, the genome is information rich and is close to random. (See however the later work of xxx which indicates the genome is compressible.) Nevertheless, the genome is not just close to random, but highly specific. Davis then argues that this is possible because the genome arose from evolutionary processes through random mutations and natural selection. This leads to a compressed, but specific, information structure. Personally I think that replication processes, as I argue in section 9, are a natural way to generate ordered structures.

Wolfram [107] puts forward a stronger view and argues that the order we see in the universe arises through simple computational processes analogous to those undertaken by cellular automata. If so, the universe is not random but pseudo-random, as the complex structures that emerge do so through relatively simple computational processes. Wolfram then surveys simple computational worlds hoping to find insights into the computational processes of the universe itself and claims a Theory of Everything (TOE) is in principle possible by trialling different simple computational machines. He believes that interpreting the universe in digital form is likely to provide better insights than the more conventional mathematics based on continuum dynamics. Interestingly, in chapter 12 of his book, Wolfram [107] provides a detailed account of Algorithmic Information Theory.

Another approach is that of Chaitin [29] who discusses the intelligibility of the universe in terms of AIT. He quotes Weyl who drew attention to Leibniz' claim that physical laws must be simple. Chaitin supports Weyl's belief that simplicity is central to the epistemology of science and gives examples from a number of emminent scientists about simplicity and complexity. These scientists support the view that a belief that the universe is rational and lawful is of no value if the laws are beyond our comprehension. Is then a TOE possible? Such a TOE would maximially compress the observable universe. Chaitin quotes Barrow [29] in arguing that, even if a TOE was uncovered, there would be no certainty that a deeper explanation might exist. This possibility appears to contradict Wolfram's views outlined above that a simple generator of physical reality might be possible. However there is no contradiction as Chaitin points out. The TOE he and Barrow are describing is an algorithmic description, effectively limited by Gödels's unprovability, whereas Wolfram suggests systematically searching for simple generators of the universe starting with the most simple.

Calude and Meyerstein [21] raise the question as to whether the universe is lawful or not. Their conclusion is that the universe is lawless in the sense that there is no overall structure implied by the word "law" Their argument in effect is saying that the human mind is not capable of grasping the behaviour of the universe. While these authors refer first to the historical perspectives from Plato to Galileo, their own argument focuses on the strong articulation of Gödel's theorem that shows the set of unprovable but true statements is

large, unprovability is everywhere. The arguments are interesting and capture Chaitin's statement that: "that god not only plays with dice in quantum mechanics, but even with the whole numbers'. Calude and Meyerstein [21] suggest that the universe may best be described by what they call a lexicon produced by say the toss of a coin. Within this infinite lexicon, every possible sequence will occur. Sections of this string representing our universe may be partially ordered, and we can hope that this ordered part of the lexicon may be partially explained by science. But elsewhere the string may be random. However where we do make sense it will only be for that part of the lexicon that is ordered. Calude and Meyerstein argue that as the human mind can never fully understand mathematics, the human mind can never fully understand the physical world. We are part of the system and we do not have the energy resources needed to probe the relativistic universe. On the other hand, Casti and Karlqvist [24] have the hope that, while the mathematical universe might be too algorithmically complex for the human mind, in principle the physical universe might not be.

Why can we make any sense of the universe?

We do seem to be able to make sense of the universe, but our capability to do so would seem to be limited for three reasons.

- The storage capability of the human mind is limited. Even if this is extended through external storage and processing of information (such as was done in the proof of the four colour theorem), there will still be finite limits to the amount of information humans can process. If computers do the processing for us, will we have the same confidence in the result if our minds are unable to process the information directly.

- Even if human access adequate storage, they might have insufficient time to undertake sufficient computing steps to come up with an adequate understanding of say a law of the universe. This is analagous to the fact that it may take longer to calculate the weather than the universe takes to generate the observed weather.

- Finally the Gödel limit may well constrain human sensemaking. Even if the other problems did not limit human rationality, the above arguments, implicit in Gödel's theorems, indicate that most of the statements that characterise the behaviour of the universe might well be unprovable to us, either as individuals or as a collection of humans. In effect, if scientific understandings are based on formal logical processes, most theorems will be beyond us.

The question then arises as to why we are able to make any sense of anything. As Einstein, quoted by Chaitin [29] said: "The most incomprehensible thing about the universe is that it is comprehensible". It could be that sense making through physical laws is not really rigorous in the formal system sense

and therefore is not limited by Gödel's theorem. We may have explanations that work, but which might at some level not require consistency but rather a lesser form of certainty perhaps embodied in a probability. It could be that our brains are quantum computers and are not constrained by Gödel's theorem. However the following argument suggests that reductionism, which is core to much of our sensemaking, works because the universe is relatively simple.

While the scientific reductionist approach seems to be extraordinarily successful in making sense of the universe, despite the attaacks of some philosophers and postmodernists, this is a surprise.

Yet we do seem to be able to make sense of the world by reducing the whole, which appears inpenetrable, to smaller manageable components. Surely this is because the world is structured that way. In other words the surprise is not that the universe is complex, but that it is algorithmically simple. This is a simplicity that allows the observed universe to a large extent to be reduced to simple laws. These laws are so simple that they can be processed by a finite brain that is insignificant relative to the rest of the universe. Such a simplicity seems to arise because small parts of the universe can be described simply. I can describe algorithmically the behaviour of a falling stone; a stone which is part of a much greater whole. My algorithmic description of a falling stone can be seen as a subroutine in a much more complex algorithm. But to a large extent, the inputs of the rest of the universe make little difference to the result. The rest of the universe impacts on the falling stone mostly as small perturbations. It would seem that it is because the universe we observe consists of structures nested within structures, or order nested within order, that we can make sense of it. Each level of nesting can be mapped on to a subroutine that describes the structure at that level. At our level of inquiry, many of the subroutines describe highly ordered systems with low algorithmic complexity. This suggests that we can make sense at a particular level of nesting, because the algorithmic complexity of the system is less than the algorithmic complexity of our cognitive processes. While we may not have the computational capacity to understand the system in its entirety, we appear to have the capacity to understand the much less algorithmically complex subroutines that are nested within the system, and can model the manner in which these subroutines are integrated into the whole. Such an approach may not always work, but in practice it seems to work surprisingly often. However in reconstituting the subroutines we often need to allow for apparently random contributions; i.e. contributions that are outside our capability to comprehend. External impacts may often appear as randomness. Whether hidden laws, noise, or quantum uncertainty, what seems to us like chance events in the macro world often only have a minor impact on the part of the universe we can make sense of.

Perhaps we rely on phenomenological ways of sense making that avoid the Gödel dilemma. The implicit or explicit axioms of phenomenological laws may have empirically selected one of two possible truth statements or even selected a path through a binary tree of unprovable truth statements. This may allow us to describe highly algorithmically organised systems at the macro level to a moderate degree of satisfaction. Nevertheless, it may be impossible to relate

these higher level understandings to more fundamental understandings because of our cognitive limitations. When the whole cannot be explained by combining the parts, it might be because our brains cannot operate outside strict formal logical processes.

It could even be that the uncertainty principle arises through such a process. While, the great physicists and mathematicians, including Gödel, have been sceptical about any relationship between his theorem and the uncertainty principle, and while the formal framework of quantum physics does not allow for hidden variables, who knows?

Appendix A

Exact cost of disturbance

Zurek [109] has shown that $H(i|o)$ bits are required to restore a system from the output state o to the input state i when information has been lost in going from i to o. However in general, as $H(i|o) \neq -H(o|i)$, this does not match the information lost in going from i to o unless the computation is reversible. As a consequences there does not appear to be simple relationship between the entropy impact of the disturbance and the entropy of the restoration process. Zurek has argued that a slightly weaker requirement that looks at restoring the system from o^*, the algorithmically compressed description of

$$H(i|o^*) = |p^*_{reg}| = H(i|o, H(o)) = H(i) - H(o), \tag{A.1}$$

where the restoration programme is given by p^*_{reg}. As knowing o^* is equivalent to knowing both o and $H(o)$, the weaker version requires fewer bits as necessary information is captured in $H(o)$.

But

$$U(p^*_{reg}, oH(o)) = U(p^*_{reg}H(o), o) = i,$$

as $H(o)$ can be part of the programme string rather than the input string. This implies that $H(i|o) = H(i|o^*) + |H(o)|$ or $H(i|o) = H(i|o^*) + log_2 H(o)$ (see also Zurek [109].

The weaker version will be used to characterize the disturbance and any regulatory response as

- As the stronger version of the regulatory response must replace the lost history of $H(o)$, the regulatory response in the stronger version is not as simply related to the algorithmic entropy introduced by the disturbance.
- The weaker version is antisymmetric in that $H(o|i^*) = -H(i|o^*)$, and the entropy cost of restoring the system after a disturbance is the same as the entropy change to the system, but the direction of flow is opposite.
- The entropy difference $H(i) - H(o)$ is independent of the UTM and is path independent.

Bibliography

[1] Digitial physics. http://en.wikipedia.org/wiki/Digital_physics.

[2] Flying in v-formation gives best view for least effort. *New Scientist*, (April), 2007.

[3] Wolfram's 2,3 turing machine is universal, 2007.

[4] J. Alebraheem and Y. Abu-Hasan. Persistence of predators in a two predators- one prey model with non-periodic solution. *Applied Mathematical Sciences*, 6(19):943–956, 2012.

[5] W. R. Ashby. *Introduction to Cybernetics*. University Paperbacks, London, 1964.

[6] A. Barron and T. Cover. Minimum complexity density estimation. *IEEE Trans. on Information Theory*, 37(4):1034–1054, 1991.

[7] A.R. Barron, J. Rissanen, and B. Yu. The mdl principle in modeling and coding. *IEEE Trans. on Information Theory*, IT-44(6):2743–2760, 1998.

[8] J. D. Barrow. *Impossibility: The limits of science and the science of limits*. Random House, London, ISBN:0-09-0772116, 1999.

[9] S. Beer. *The Heart Of Enterprise*. Wiley, Chichester, 1979.

[10] S. Beer. *Brain of the Firm, 2nd ed.,*. John Wiley & Sons, Chichester, 1981.

[11] C. H. Bennett. Logical reversibility of computation. *IBM Journal of Research and Development*, 17:525–532, 1973.

[12] C. H. Bennett. Thermodynamics of computation- a review. *International Journal of Theoretical Physics*, 21(12):905–940, 1982.

[13] C. H. Bennett. Logical depth and physical complexity. In R Herken, editor, *The Universal Turing Machine- a Half-Century Survey*, pages 227–257. Oxford University Press, Oxford, 1988.

[14] C. H. Bennett. Notes on landauer's principle, reversible computation, and maxwell's demon. http://xxx.lanl.gov/PS_cache/physics/pdf/0210/0210005.pdf, 2003.

[15] C. H. Bennett, P. Gács, M. Li, P. M. B. Vitanyi, and W. H. Zurek. Information distance. *IEEE Transactions on Information Theory*, 44(4):1407–1423, 1998.

[16] L. Brillouin. *Science and Information Theory.* Academic press, New York, 2nd edition, 1962.

[17] H Buhrman, J. Tromp, and P. Vitanyi. Time and space bounds for reversible simulation. *Journal of Physics A: Mathematical and General*, 34:35:6821–6830, 2001.

[18] C. Calude. *Information and Randomness:An Algorithmic perspective.* Springer-Verlag, 2nd edition, 2002.

[19] C. Calude, D. I. Campbell, K. Svozil, and D. Stefanescu. Strong determinism vs. computability. In W. Depauli-Schimanovich, E. Koehler, and F. Stadler, editors, *The Foundational Debate, Complexity and Constructivity in Mathematics and Physics*, pages 115–131. Kluwer,, Dordrecht, Boston, London, 1995.

[20] C. Calude, M. J. Dinneen, and C. K. Shu. Computing a glimpse of randomness. *Experimental Mathematics*, 11(3):361–370, 2002.

[21] C. S. Calude and F. W. Meyerstein. Is the universe lawful ? *Chaos, Solitons & Fractals*, 106:1075–1084, 1999.

[22] J. Casti. The great ashby:complexity, variety, and information. *Complexity*, 2(1):7–9, 1996.

[23] J. L. Casti. *Complexification:Explaining a Paradoxical World Through the Science of Surprise.* Harpercollins, 1995.

[24] J. L. Casti and A. Karlqvist, editors. *Boundaries and Barriers.* AddisonWiley, New York, 1996.

[25] John L. Casti and C. Calude. Randomness: the jumble cruncher. *New Scientist*, 183(2466):36–37, 2004.

[26] G. Chaitin. On the length of programs for computing finite binary sequences. *J. ACM*, 13:547–569, 1966.

[27] G. Chaitin. A theory of program size formally identical to information theory. *Journal of the ACM*, 22:329–340, 1975.

[28] G. Chaitin. Toward a mathematical definition of "life". In R. D. Levine and M. Tribus, editors, *The Maximum Entropy formalism*, pages 477–498. MIT Press, Massachusetts, 1979.

[29] G. Chaitin. On the intelligibility of the universe and the notions of simplicity, complexity and irreducibility. In W. Hogrebe and J. Bromand, editors, *Grenzen und Grenzberschreitungen, XIX. Deutscher Kongress fr Philosophie, Bonn, September 2002*, pages 517–534, Berlin, 2004. Akademie Verlag.

[30] G. J. Chaitin. Information-theoretic limitations of formal systems. *J. ACM*, 21(3), 1974.

[31] G. J. Chaitin. Algorithmic information theory. In *Encyclopedia of Statistical Sciences*, volume 1, pages 38–41. Wiley, New York, 1982.

[32] G. J. Chaitin. Gödel's theorem and information. *International Journal of Theoretical Physics*, 22:941–954, 1982.

[33] G. J. Chaitin. *Information, randomness & incompleteness : papers on algorithmic information theory*. World Scientific, New Jersey, 1987.

[34] G. J. Chaitin. Randomness and complexity in pure mathematics. *International Journal of Bifurcation and Chaos*, 4:3–15, 1994.

[35] G. J. Chaitin. *the Limits of Mathematics*. Springer-Verlag, London, ISBN 1-85233-668-4, 2003.

[36] G. J. Chaitin. *Meta math! The quest for Omega*. Pantheon ISBN: 0375423133, 2005.

[37] G. J. Chaitin. The limits of reason. *Scientific American*, 294:74–81, 2006.

[38] T. M. Cover, P. Gcs, and R. M. Gray. Kolmogorovs contributions to information theory and algorithmic complexity. *Ann. Probab*, 17:840–865, 1989.

[39] D. Crutchfield. The calculi of emergence: Computational, dynamics, and induction. *Physica D*, 75:11–54, 1994.

[40] J. P. Crutchfield and C. R. Shalizi. Thermodynamic depth of causal states: When paddling around in occam's pool shallowness is a virtue. Working paper 98-06-047, Santa Fe Insitute, 1999.

[41] J. P. Crutchfield and K. Young. Inferring statistical complexity. *Physical Review Letters*, 63:105–108, 1989.

[42] Feldman D. *Computational Mechanics of Classical Spin Systems*. Ph.d., University of California, Davis, 1998.

[43] Saswato Das. Quantum threat to our secret data. *New Scientist*, 13 September(2621), 2007.

[44] P. C. W. Davies. *The Fifth Miracle: the Search for the Origin of Life*. Penguin Books Ltd, 2003.

[45] D. Deutsch. Quantum theory, the church-turing principle and the universal quantum computer. *Proceedings of the Royal Society, Series A*, 400:97–117, 1985.

[46] S. D. Devine. The application of algorithmic information theory to noisy patterned strings. *Complexity*, 12(2):52–58, 2006.

[47] S. D. Devine. An algorithmic information theory approach to the emergence of order using simple replication models. In *First International Conference on the Evolution and Development of the Universe*. 2008.

[48] S. D. Devine. The insights of algorithmic entropy. *Entropy*, 11(1):85–110, 2009.

[49] M. Eigen. Selforganization of matter and the evolution of biological macromolecules. *Naturwissenshaften*, 58:465–523, 1971.

[50] M. Eigen, J. McCaskill, and P. Schuster. Molecular quasi-species. *The Journal of Physical Chemistry*, 92(24):6881–6891, 1988.

[51] S. Feferman. Gdel, nagel, minds, and machines: Ernest nagel lecture, columbia university, sept. 27, 2007. *The Journal of Philosophy*, 106:201–219, 2009.

[52] D. P. Feldman and J. P. Crutchfield. Discovering noncritical organization: Statistical mechanical, information theoretic, and computational views of patterns in one-dimensional spin systems. (SFI Working Paper 98-04-026), 1998.

[53] E. Fredkin. Digital mechanics. *Physica D*, pages 254–270, 1990.

[54] E. Fredkin and T. Toffoli. Conservative logic. *International Journal of Theoretical Physics*, 21:219–253, 1982.

[55] P. Gács. On the symmetry of algorithmic information. *Soviet Mathematics Doklady*, 15:1477–1780, 1974.

[56] P. Gács. Exact expressions for some randomness tests. *Zeitschrift fur mathematische Logik und Grundlagen der Mathematik*, 26:385–394, 1980.

[57] P. Gács. Lecture notes on descriptional complexity and randomness. Lecture notes, Boston University Computer Science Department, 1988. Up to date version at `http://www.cs.bu.edu/~gacs/papers/ait-notes.pdf`.

[58] P. Gács. The Boltzmann entropy and randomness tests- extended abstract. In *Proceedings of the Workshop on Physics and Computation*, pages 209–216. IEEE Computer Society Press, 1994.

[59] P. Gács. The Boltzmann entropy and random tests. Technical report, Boston University Computer Science Department, 2004. `http://www.cs.bu.edu/~gacs/papers/ent-paper.pdf`.

[60] P.D. Grünwald. Tutorial on minimum description length. In P.D. Grünwald, I.J. Myung, and M.A. Pitt, editors, *Advances in Minimum Description Lengh:Theory and Applications*. MIT Press, 2005.

[61] M. H. Hansen and B. Yu. Model selection and the principle of minimum description length. *Journal of the American Statistical Association*, 96:746 774, 2001.

[62] M. Hutter. A theory of universal artificial intelligence based on algorithmic complexity. `http://arxiv.org/abs/cs.AI/0004001`, 2000.

[63] M. Hutter. *Universal Artificial Intelligence*. Springer, ISBN: 3-540-22139-5, London, 2005.

[64] M. Hutter. On generalized computable universal priors and their convergence. `http://arxiv.org/PS_cache/cs/pdf/0503/0503026v1.pdf`, 2005b.

[65] M. Hutter. Universal algorithmic intelligence: A mathematical top→down approach. In B. Goertzel and C. A. Pennachin, editors, *Artificial General Intelligence*, pages 227–290. Springer, Berlin, 2007.

[66] E. T. Jaynes. Information theory and statistical mechanics. *Physical Review*, 106:620–630, 1957.

[67] E. T. Jaynes. Where do we stand on maximum entropy. In R.D. Levine and M. Tribus, editors, *The Maximum Entropy formalism*, pages 15–118. MIT Press, Cambridge, Massachusetts, 1979.

[68] E. T. Jaynes and G. L. Bretthorst. *Probability theory: the logic of science*. Cambridge University Press, New York, 2003.

[69] K. Kolmogorov. Three approaches to the quantitative definition of information. *Prob. Info. Trans.*, 1:1–7, 1965.

[70] E. Kvaalen. Has the riemann hypothesis finally been proven? *New Scientist*, (2648):40–41, 2008.

[71] R. Landauer. Irreversibility and heat generation in the computing process. *IBM Journal of Research and Development*, 5:183–191, 1961.

[72] R. Landauer. Information is physical. In *Proceedings of PhysComp 1992*, pages 1–4. Los Alamitos: IEEE Computer Society Press, Oxford, 1992.

[73] H. S. Leff and A. F. Rex. *Maxwell's Demon: Entropy, Information, computing*. Princeton University Press, Princeton, 1990.

[74] L.A. Levin. On the notion of a random sequence. *Soviet Mathematics Doklady*, 14(5):1413–1416, 1973.

[75] L.A. Levin. Laws of information (nongrowth) and aspects of the foundation of probability theory. *Problems of Information Transmission*, 10(3):206–210, 1974.

[76] M. Li and P. M. B. Vitanyi. *An introduction to Kolmogorov Complexity and its Applications*. Springer-Verlag, New York, second edition, 1997.

[77] M. Li and P.M.B. Vitanyi. Reversibility and adiabatic computation: trading time and space for energy. *Proceedings of the Royal Society of London, Series A*, 452:769–789, 1996.

[78] S. Lloyd. *Programming the Universe: A Quantum Computer Scientist Takes On the Cosmos*. Knopf, ISBN 1-4000-4092-2, New York, 2006.

[79] Seth Lloyd. Computational capacity of the universe. *Phys. Rev. Lett.*, 88:237901, May 2002.

[80] P. Martin-Löf. The definition of random sequences. *Information and Control*, 9:602–619, 1966.

[81] M. Minsky. Size and structure of a universal turing machine using tag systems. In *In Recursive Function Theory: Proceedings, Symposium in Pure Mathematics*, volume 5, pages 229–238, Provelance, 1962. AMS.

[82] J. Rissanen. Generalized kraft inequality and arithmetic coding. *IBM Journal of Research and Development*, 20(3):198–203, 1976.

[83] J. Rissanen. Modeling by the shortest data description. *Automatica*, 14:465–471, 1978.

[84] J. Rissanen. A universal prior for integers and estimation by minimum description length. *Annals of Statistics*, 11(2):416–431, 1983.

[85] J. Rissanen. Stochastic complexity and modeling. *Annals of Statistics*, 14:1080–1100, 1986.

[86] J. Rissanen. Stochastic complexity. *J. Royal Statistical Society*, 49B(3):223–265 and 252–265, 1987.

[87] J. Rissanen. *Stochastic Complexity in Statistical Inquiry*. World Scientific, New Jersey, 1989.

[88] E. D. Schneider and J. J. Kay. Life as a manifestation of the second law of thermodynamcis. *Mathl. Comput. Modelling*, 16(6-8):25–48, 1994.

[89] C. P. Schnorr. Process complexity and effective random tests. *Journal of Computer and System Sciences*, 7:376–388, 1973.

[90] C. R. Shalizi and J. P. Crutchfield. Pattern discovery and computational mechanics. *arxiv.org/abs/cs/0001027v1*, 2000.

[91] C. R. Shalizi and P. Crutchfield. Computational mechanics pattern and prediction, structure and simplicity. *Journal of Statistical Physics*, 104:819–881, 2001.

[92] C. E. Shannon. A mathematical theory of communication. *Bell System Tech. J.*, 27:379–423, 1948.

[93] R. J. Solomonoff. A formal theory of inductive inference; part 1 and part 2. *Information and Control*, 7:1–22,224–254, 1964.

[94] R. J. Solomonoff. Complexity-based induction systems: Comparisons and convergence theorems. *IEEE Trans. Information Theory*, IT-24(4):422–432, 1978.

[95] R. J. Solomonoff. Lecture 1: Algorithmic probability. `http://world.std.com/~rjs/iaplect1.pdf`, 2005.

[96] R. J. Solomonoff. Lecture 2. applications of algorithmic probability. `http://world.std.com/~rjs/iaplect1.pdf`, 2005.

[97] E Szathmary and J Maynard Smith. From replicators to reproducers: the first major transitions leading to life. *Journal of Theoretical Biology*, 187:555–571, 1997.

[98] L. Szilard. On the decrease of entropy in a thermodynamic system by the intervention of intelligent beings, 2003.

[99] S. Szilard. Uber die entropieverminderung in einnem thermodynamischen system bei eingriffen intelligenter wesen. *Zeitschrift fr Physik*, 53:840–856, 1929.

[100] T. Toffoli. Physics and computation. *International Journal of Theoretical Physics*, 21:165–175, 1982.

[101] R. C. Tolman. *Statistical Mechanics*. Oxford University Press, London, 1950.

[102] A. Turing. On computable numbers, with an application to the entscheidungsproblem. *Proceedings of the London Mathematical Society, Series 2*, 42, 1936.

[103] N. K. Vereshchagin and P. M. B. Vitányi. Kolmogorovs structure functions and model selection. *IEEE Transactions on Information Theory*, 50(12):3265–3290, 2004.

[104] P. M. B. Vitányi and M. Li. Minimum description length induction, bayesianism, and kolmogorov complexity. *IEEE Trans. Inform. Theory*, 46:446–464, 2000.

[105] P.M.B. Vitányi. Time space and energy in reversible computing. In *Proceedings of the 2005 ACM International Conference on Computing Frontiers*, pages 435–444, Ischia, Italy, 2005.

[106] R. von Mises. Grundlagen der wahrscheinlichkeitsrechnung. *Mathematische Zeitschrift*, 5:52, 1919.

[107] S. Wolfram. *A New Kind of Science*. Wolfram Media, Champaign, IL, 2002.

[108] W. H. Zurek. Algorithmic randomness and physical entropy. *Physical Review A*, 40(8):4731–4751, 1989.

[109] W. H. Zurek. Thermodynamics of of computation, algorithmic complexity and the information metric. *Nature*, 341:119–124, 1989.