Minimum Message Length and Kolmogorov Complexity

C.S. Wallace and D.L. Dowe

Computer Science, Monash University
Clayton Vic 3168 Australia.
Email: (csw@cs.monash.edu.au, dld@cs.monash.edu.au)

The notion of algorithmic complexity was developed by Kolmogorov (1965) and Chaitin (1966) independently of one another and of Solomonoff's notion (1964) of algorithmic probability. Given a Turing Machine, T, the (prefix) algorithmic complexity of a string, S, is the length of the shortest input to T which would cause T to output S and stop. The Solomonoff probability of S given T is the probability that a random binary string of 0s and 1s will result in T producing an output having S as a prefix. We attempt to establish a parallel between a restricted (two-part) version of the Kolmogorov model and the Minimum Message Length approach to statistical inference and machine learning of Wallace and Boulton (1968), in which an "explanation" of a data string is modelled as a two-part message, the first part stating a general hypothesis about the data and the second encoding details of the data not implied by the hypothesis. Solomonoff's model is tailored to prediction rather than inference in that it considers not just the most likely explanation, but it also gives weights to all explanations depending upon their posterior probability. However, as the amount of data increases, we typically expect the most likely explanation to have a dominant weighting in the prediction.

Keywords: Algorithmic complexity, Kolmogorov complexity, Solomonoff probability, Minimum Message Length, MML, prediction, learning, inductive inference, statistical inference.

Received ?; revised ?

1. INTRODUCTION

The study and applications of the complexity of a body of information have followed at least three streams, springing from at least three independent conceptions of the concept. The motivations, and hence the direction of development, of these three streams have been different, yet there are obvious similarities among them, and each stream has something to offer, and something to learn from, the others. The intent of this paper is to describe some of the relationships among the different streams, and to try to clarify some of the important differences in their assumptions and development. Other studies mentioning the relationships appear in [1, section IV, pp 1038–1039], [2, sections 5.2, 5.5] and [3, p 465].

The first stream to be described was initiated by Kolmogorov [4] with important later developments by Martin-Löf and Chaitin [5]. In this stream (as in the others) the body of information is usually assumed to be a finite string of binary digits S. We will write $\#S$ for the length of S. The prefix Kolmogorov or Algorithmic complexity of S with respect to some specified Universal Turing Machine (UTM) T may be defined as the length $\#I$ of the shortest binary string I which, when supplied to T, causes T to output S and stop. The set of such inputs for all finite S forms a prefix set, as no member of the set can be a prefix of another. This prefix set may be regarded as a prefix code for the set of finite binary strings. The code is decoded by T. The universality of T ensures that for all strings S, the difference between the complexities of S with respect to the UTMs $T1$ and $T2$ is bounded above by a constant independent of S, namely the length of the longer of the programs required to make $T1$ imitate $T2$ and to make $T2$ imitate $T1$.

We will write $K_T(S)$ for the Kolmogorov complexity of S with respect to UTM T.

The second stream (chronologically the first) springs from the work of Solomonoff [6] and again involves the idea of the length $\#I$ of an input string which can cause a UTM T to output the given string S. However, S is now treated not as just an abstract binary string, but as a string which represents in some binary code data about the real world. The intent is not to measure the complexity of S, but rather to develop a probability distribution over the set of finite binary strings as a model of the probability that the data represented by a string S is or will be true in the real world. Rather than emphasising the shortest I which will produce S, this stream considers all strings I which will cause T to produce output having S as a prefix, and such that no proper prefix of I will produce S. It then defines an (unnormalized) probability for S as

$$P_T(S) = \sum_I (2^{-\#I})$$

The above sum may often be dominated by the term for the shortest input, in which case $P_T(S)$ may be well approximated by

$$2^{-K_T(S)}$$

Note, however, that the definition of $K_T(S)$ requires the output of T to be precisely S, whereas the definition of $P_T(S)$ requires only that the output begins with S, so $K_T(S)$ will in general exceed the length of the shortest stream-two input.

The primary motivation for the development of this probability distribution is inductive prediction. Given known data represented by S, and strings $N1$, $N2$ representing two possible future events or observations $E1$ and $E2$ in the same code as used in S, the relative probability of $E1$ and $E2$ is estimated as

$$\mathrm{Prob(E1)/Prob(E2) = P_T(S:N1)/P_T(S:N2)}$$

where ":" represents concatenation. As with Kolmogorov complexity, Solomonoff probabilities are defined with respect to some particular UTM, but universality guarantees that different choices of UTM will affect the odds between future events only by a factor with bounds independent of S, $E1$ and $E2$.

The third stream was introduced by Wallace and Boulton [7, 8, 9, 10, 11, 12], with a similar but independent development by Rissanen [3]. Unlike the other streams, its basis is Shannon's theory of information rather than the theory of Turing machines. Like the second stream, it regards the given string S as being a representation, in some code, of data about the real world. We now seek a string $I = H : A$ where the first part H specifies an hypothesis about the source of S (normally selected from a limited set of possible hypotheses), and the second part A is an encoding of the data using a code which would be optimally efficient were the stated hypothesis true. By "optimally efficient" we mean a code such as a Huffman or Arithmetic code which minimizes the expected length of the coded string.

Shannon's theory shows that the length of the string coding an event E in an optimally efficient code is given by $-\log_2(Prob(E))$ (we will neglect rounding to integer values), so the length of A is given by

$$\#A = -\log_2(f(S|H))$$

where $f(S|H)$ is the conditional probability of data S given the hypothesis H. (We use H to indicate both the stated hypothesis and the string which states it.)

The length $\#H$ of the specification of the hypothesis is based on the assumption of a prior probability distribution $h(H)$ over the set of possible hypotheses, and again using an optimal code for the specification, is given by $-\log_2(h(H))$. The resulting total length $\#I$ is thus

$$\begin{aligned}\#I = \#H + \#A &= -\log_2(h(H)) - \log_2(f(S|H)) \\ &= -\log_2(h(H) \times f(S|H))\end{aligned}$$

The aim in this stream is to find the hypothesis H which leads to the shortest such string I, which may be regarded as the shortest message encoding the data given in S. For this reason, the technique is termed Minimum Message Length (MML) or Minimum Description Length (MDL) inference. The motivation for this aim is that the MML hypothesis can be shown to capture all of the information in S which is relevant to the selection of an hypothesis, and this method of inference automatically embraces selection of an hypothesis of appropriate complexity as well as leading to good estimates of any free parameters of the hypothesis. If no I exists with $\#I < \#S$, then there is no acceptable hypothesis for the data, at least within the set considered possible, and the data is concluded to be random noise.

The minimization of $\#I$ is, as shown by the equation above, equivalent to maximization of $h(H) \times f(S|H) = Prob(H, S)$, i.e. the joint probability of hypothesis and data. It is thus formally equivalent to choosing the hypothesis of highest Bayesian posterior probability given S. Note in passing that this is a finite posterior *probability* that we maximise and *not* a density, so the chosen H is not in general the posterior mode. Continuing on, however, as shown in [13, 8], the shortest message is achieved in general by devising a coding for the hypotheses in which there are code strings defined only for a countable subset of the possible hypotheses. We may call this subset the set of "usable" hypotheses, since only these may be stated by the string H. Roughly speaking, the selection of the subset is done in such a way that no two hypotheses in the subset are so similar that the available volume of data cannot be expected to distinguish reliably between them. A useful side-effect of this subsetting is that real-valued free parameters of the original hypothesis space may have prior density distributions defined, but these cannot lead to finite prior probabilities for all parameter values. The restriction to a countable set of usable hypotheses allows a non-zero prior probability to be assigned to each usable vector of parameter values, and hence allows the construction of an optimal finite-length binary code for the hypothesis strings.

As is evident from the above, the primary motivation for the MML/MDL stream is the inductive inference of a general hypothesis about the data, rather than prediction of future data. Of course, a good model of the data source should lead to good predictions, but not necessarily as good as those of Solomonoff, which in a sense weight all possible hypotheses about the data.

Sections 2 to 4 of this paper are an attempt to relate stream three to the Algorithmic Complexity (AC) frameworks of streams one and two. Specifically, we will attempt to show that if the set of hypotheses considered in stream three is expanded to include all computable probability functions over the set of possible data, then the MML selection of an hypothesis from given data is equivalent to the selection of a shortest input which will cause a certain UTM to output the data. The correspondence with stream-one and stream-two frameworks clearly cannot be exact, because there is no requirement in these frameworks that the inputs producing a given string S permit interpretation as stating any hypothesis about the data, or have the two-part structure of an MML message. We therefore have to introduce certain constraints on the acceptable input strings. The discussion which follows should be read as a first attempt to define a set of constraints on the AC framework which may bring it into correspondence with MML/MDL. The constraints we suggest may well be neither sufficient nor wholly necessary, and undoubtedly further work will be needed to establish the correspondence on a rigorous basis, if indeed this proves to be possible.

2. SOME TECHNICAL DETAILS

2.1. The Turing machine

The Turing machine used in the definitions of Kolmogorov complexity and Solomonoff probability is most conveniently taken to have a couple of special features. Rather than finding its input, and leaving its output, on a single work tape as assumed in the classical Turing machine model, our machine has a one-way input tape which may be read and advanced, but not altered or backspaced. It has a one-way output tape which may be written and advanced, but not backspaced or overwritten. It has the usual one or more work tapes. The input and output tapes use a binary alphabet with no blank or delimiter symbol in addition to 0 and 1.

In all cases where we speak of the machine reading an input I and producing a string S, we mean that the machine is started in a distinguished start state, with unspecified work tape content, and with the input tape containing I followed by unspecified digits; that the machine reads all of I; that it writes S on its output tape; and that no proper prefix of I satisfies these conditions. In stream one, the machine is then required to stop. In stream two, its further action is unspecified: it may or may not write further output and/or read further input.

We will argue that for comparison with stream three, it is convenient to require that the machine, having output the last digit of S, should enter a read state. Indeed, in stream three, we will assume that the reference UTM T is modified by replacing all stop states by read states. Note that if, for stream one, input I causes the (unmodified) T to produce S, then for stream three, the same input I causes the modified T to produce S. However, there may be strings J which by the stream-three definition cause the modified T to produce S, but which would not cause the unmodified T to produce S.

For all streams, if I causes T to produce S according to the appropriate definition, we will write

$$T(I) \;=\; S$$

with T being assumed to be modified in stream three.

2.2. Probability from complexity

Solomonoff explicitly relates complexity and probability. The Solomonoff probability $P_T(S)$ w.r.t. T of the data represented by a string S is just the probability that T, when presented with an infinite string of random digits as input, will produce an output having S as prefix. (The event that T produces no finite string may be considered as contributing to the probability of the null string.) The undecidability of the halting problem prevents the exact computation or normalization of this probability distribution over finite strings, but it is a well-defined concept with a clear probabilistic interpretation.

The Algorithmic complexity $K_T(S)$ may also be used to define an unnormalized probability

$$\mathrm{Prob(S) = 2^{-K_T(S)} = P^K_T(S)}, \text{ say}.$$

This "probability" does not correspond to the probability of any easily-defined event, and is always less than the Solomonoff probability $P_T(S)$. However, it may reasonably be considered as a conservative estimate of $P_T(S)$, since it will often be only a little smaller.

In Shannon's theory, if a uniquely-decodeable code is devised for a set of mutually-exclusive and exhaustive events, where the probability of event E_i is P_i, the expected length of a message announcing an event is minimized when the length of the code word for E_i is $-\log_2(P_i)$, and a code so constructed is called optimal or efficient. One of the properties of an efficient code is that it provides exactly one code word or string for each possible event: it is non-redundant. A UTM T may be regarded as a decoder for a code over finite strings, where I is a code word for S if input I produces output S. The code is clearly not efficient, since many code words exist for each S, so the code is highly redundant. However, if we only ever use the shortest code for each string, the code may be fairly efficient for the unnormalized probability distributions over strings defined by $K_T()$. In particular, it shares with efficient codes the property that its words are random, in senses now to be described.

2.3. Randomness

Informally, a random source of binary digits provides an infinite stream of digits such that knowledge of the first N digits provides no information about the next digit, and which produces ones and zeros with equal frequency. A property of an efficient (Shannon) code is that if a sequence of events is randomly drawn from the probability distribution for which the code is efficient, and the code words for these events concatenated, then the resulting stream of digits comes from a random source. Note that this statement is about the randomness of a source or process for producing binary digits, not about the randomness of any finite string. The randomness of a finite string is usually assessed by asking whether any standard statistical test would cause us to reject the hypothesis that the string came from a random source. If the string indeed came from a random source such as an efficient binary coding of an event sequence, the test(s) will of course sometimes result in our falsely rejecting the randomness hypothesis in just those cases where the source happens by chance to give a string exhibiting some pattern.

Complexity theory provides a different definition of randomness directly applicable to finite strings. A string S is random w.r.t. a UTM T if $K_T(S) \geq \#S - c$, where c is a small constant chosen to impose a "significance" requirement on the concept of non-randomness, larger values of c requiring a lower complexity before a string is accepted as non-random. It has been shown that this definition is not in conflict with the ordinary notion of randomness [5]. Another way of stating this definition is that S is random w.r.t. T if the code defined by T provides no coding of S which is significantly shorter than S. That is, S is not compressible by T.

Note that the notion of randomness as incompressibility makes sense only in the context of a *redundant* coding system. In the code defined by T, there are many representations for S. One of these has the form of a "copy" program for T followed by a specification of the length $\#S$ followed by S itself. The length of this representation is about $(\#S + \log_2(\#S))$. If this is the shortest representation of S, then S will be declared random. Only if the code permits a substantially shorter representation will S be seen as non-random. But alternative representations exist only in redundant codes.

It is easily shown that, if S is drawn from a random source (in the earlier sense), the probability that $K_T(S) < (\#S - d)$ is less than 2^{-d}, as there are 2^L strings of length L, all equiprobable from a random source, but at most $2^{(L-d-1)}$ prefix-free representations available of length less than $(L - d)$. Conversely, if S comes from a non-random source, its early digits give some hint as to the values of later digits. In this case, T may be given an input comprising a fixed-length program to compute the probability (in the usual sense) of the $N\,th$ digit's being 1 as a function of the earlier digits, and then to decode a (Shannon) optimal code for the $N\,th$ digit based on this probability. Whenever the probability differs from 1/2, the optimal code will be able to code the next digit with less than one bit on average. Provided S is long enough, this input will be shorter than S, and so S will be declared non-random w.r.t. T.

Since T is universal, it follows that the shortest input I to T which produces S is random w.r.t. T. Were it not, and had a significantly shorter representation J, a shorter representation of S could be constructed comprising a short program followed by J. The program would instruct T to decode J, remember the resulting string (which is I) and then act as if this were its input. Hence, when only the shortest code word for every string is used, the UTM-based code words, like those of a Shannon code, are random, although in a somewhat different sense.

Similar remarks apply to the two-part MML code. It too is redundant, since the data S may be encoded using the assertion of any usable hypothesis H for which $f(S|H) > 0$. Hence compressibility is a relevant concept, and MML chooses H to maximize compression. If S is incompressible, it is regarded as random, i.e. having no pattern expressible within the set of possible hypotheses.

2.4. Universal distributions

The interest in Algorithmic Complexity as a means for prediction arises from a special property of $P_T(S)$ and $P_T^K(S)$: they are universal if T is universal. In particular, given any effectively computable distribution $Q(S)$ over all finite strings, the ratio $Q(S)/P_T^K(S) < C_{Q,T}$, where $C_{Q,T}$ depends on Q and T but not on S.

This result follows from the fact that for any such $Q()$, there exists a finite-length program q for T which enables T to compute $Q(S)$ for any finite S, and hence to decode a Shannon-optimal code for $Q()$. Thus for any S, $K_T(S)$ cannot exceed $(\#q - \log_2(Q(S))$. Indeed, $C_{Q,T}$ may be taken as $2^{\#q}$. This argument applies to $P_T()$ as well as to $P_T^K()$.

Now suppose that within the context of some observational program, the real world yields data strings randomly drawn from a distribution closely approximated by some $Q()$. Suppose a string S has been observed, and we are interested in the relative probabilities of two alternative future events, representable respectively by S_1 and S_2. The ratio of their probabilities is approximately

$$Q(S : S_1)\ /\ Q(S : S_2).$$

As each of the probabilities in this ratio is bounded above by $P_T(S : S_i) \times C_{Q,T}$ (for $i = 1$ or 2), the ratio may be approximated by

$$P_T(S : S_1)\ /\ P_T(S : S_2).$$

If $P_T^K()$ is used, the "approximation" will be crude at best, since for all S, $P_T^K(S)$ is by definition a negative power of two. For either AC-based probability, the possibility exists that $P(S)$ greatly exceeds $Q(S)$ for some strings which happen to have a computationally simple form (e.g. 01010101....), so no useful hard bounds can be placed on the approximation error. However, for a wide class of computable distributions over strings, there is a high probability approaching one for very long strings that the two-part input form described above will be among the shortest inputs producing S, and hence a high probability that the approximation is quite close [1].

It is this universal ability of the complexity-based probabilities to approximate the probability ratios of effectively computable distributions which justifies their use in prediction.

3. DATA AND HYPOTHESES

We are trying to work towards a version of Algorithmic Complexity which can be assimilated to a version of Minimum Message Length inference. In the latter, the concepts of Data and Hypotheses play central roles, but these concepts do not appear explicitly in stream one, and Hypotheses have no formal role in stream two. To allow any correspondence between these streams and MML, we appear to need to impose a specific interpretation on the *meaning* of the binary strings involved. In this section we attempt to specify this interpretation and to outline the consequent formal constraints on the structure of the binary strings.

3.1. The data string

We assume that the given binary string S is a representation of data obtained by observation of a real-world phenomenon which we wish to understand or whose future we wish to predict. If S is to be such a representation, it will be a representation of a number of facts, and to some extent we must be concerned with the mapping between the facts which might be observed and the binary strings. We propose to assume that S consists of the concatenation of one or more *sentences* in a data language L. Each sentence records one or more observed facts relating to a particular instance of the studied phenomenon. The sentences in L form a prefix set, so a string S can be uniquely parsed into its several sentences. It will be helpful to assume that every sufficiently long string has a sentence as prefix, and that distinct sentences represent distinct sets of facts. That is, we assume L is non-redundant. This assumption could probably be weakened at the cost of some elaboration of our argument.

We make the important assumption that each sentence in S records data from an *independent* instance of the phenomenon. By independence we here mean that we believe that the facts recorded in one sentence are not causally or statistically related to the facts recorded in another sentence in any way except that both sentences record instances of the same phenomenon. That is, we believe that the sentences would be conditionally independent given full knowledge of the nature of the phenomenon. It may be objected here that the data is what the world gives us, and our beliefs about it are irrelevant to what it means. While this objection is valid in principle, real data used in real science has invariably been already captured, interpreted and transformed by our human choice of what to look at, what instruments to use, and what we believe our senses and instruments observe. Our similar beliefs about independence of instances are of the same inescapable nature, and may be as legitimately part of our understanding of L as any of these other choices and beliefs.

A consequence of the independence assumption is that if S is the concatenation of several sentences, the order of the sentences may be permuted without affecting the meaning of the data string, so a permuted string may be substituted for S.

3.2. Hypotheses

An hypothesis about the phenomenon studied is a computable probability distribution over the sentences of L. Hence we consider hypotheses only in the context of a defined data language, and do not consider what meaning if any an hypothesis might have outside this context. If s is a sentence of L, the probability of s under hypothesis Q is written $Q(s)$. The independence of sentences implies that the probability of a data string $S = s_1 : s_2 : ... : s_n$ under Q is the product

$$Q(S) = Q(s_1) \times Q(s_2) \times ... \times Q(s_n).$$

Note that we are not excluding hypotheses for sequences of data such as might be produced by a high-order Markov process, where we expect the probability of the next observation in the sequence to depend on the values of previous observations. We regard the record of such a sequence to be a single sentence in L. If the data string comprises several such sentences, these are read as the sequences produced by several independent runs of the same Markov process, or perhaps as widely-separated subsequences from the one run.

Conversely, we do not require that the facts recorded in a single sentence of L necessarily be interdependent. Suppose, for example, that each "fact" is the weight of a randomly-selected cow, and that the sentences of L begin with a code for the sentence's length followed by one or more weights. If s_1 and s_2 are respectively the sentences $3, w_1, w_2, w_3$ and $2, w_4, w_5$ then the sentence

$$s_3 = 5, w_1, w_2, w_3, w_4, w_5$$

may also be a sentence of L. If, as is probably the case, we believe each weight to be an independent instance of "cow-weight", then we believe the meaning of s_3 to be the same as the meaning of $s_1 : s_2$ and of $s_2 : s_1$. However, some hypothesis Q may hypothesize a correlation among the weights of a sentence, and so treat s_3 quite differently from the two-sentence forms. Of course, even if Q hypothesizes the independence of all weights in a sentence, the presence of length codes in the sentences will in general mean that

$$Q(s_3) \neq Q(s_1) \times Q(s_2).$$

4. TWO-PART ENCODING

We now consider how the notion of an hypothesis-based encoding of a data string may be embedded in an Algorithmic Complexity context.

One approach, which has been described by Li and Vitányi [2], is via the stream-one concept of conditional prefix complexity. The conditional complexity $K_T(X|Y)$ of a string X given a string Y is defined as the length of the shortest input string which will cause the reference UTM T to produce X, when T already has access to the string Y, e.g. on its work tape. Here, Y is considered as a representation of the "hypothesis". The difficulty with this approach is that, while $K_T(X|Y)$ may be regarded as defining a conditional probability $P_T^K(X|Y) = 2^{-K_T(X|Y)}$, this probability does not correspond closely to the ordinary conditional probability of obtaining data X given that proposition Y is true of its source. In particular, whatever X and Y may be, $P_T^K(X|Y)$ is never zero, and indeed cannot be much less than $P_T^K(X)$. Hence, no data X can ever falsify the hypothesis Y in the Popperian sense. The string Y is not treated as asserting the truth of some hypothesis, it is merely information which may or may not be used in the compression of the data.

We propose instead that the string q representing the hypothesis be a prefix of the input to T. In what follows, we use stream-three conventions. Recall that we assume T is a UTM with no stop state, and that we define input I to produce output S, written $T(I) = S$, iff T reads all of I, writes S, then enters a read state, and that no proper prefix of I satisfies these conditions.

An input I will be regarded as an acceptable MML message encoding a data string S, where S is one or more sentences of the data language L, if certain conditions on I are met. These conditions and the reasons for them are now described.

C1 : $T(I) = S$ This requires I to encode S.

C2 : $\#I < \#S$. This condition requires that some compression of the data is achieved, so some regularity in the data has been discovered.

C3 : I has the form $I = q : j$ where

C4 : $T(q) = \Lambda$, the empty string. That is, the hypothesis itself does not determine the data.

Define Tq as the Turing machine equivalent to the state of T after reading q. That is, for all strings X, $Tq(X) = T(q : X)$. It follows from C1, C4 and C3 that $Tq(j) = S$. Then:

C5 : For all strings X,
$Tq(j : X) = S : Tq(X)$, or equivalently,
$T(q : j : X) = S : T(q : X)$.

This condition means that reading j (and so producing S) does not permanently alter the state of T: it remains equivalent to Tq. The intent is to ensure that whatever regularity in S is exploited by I to compress the data, it is expressed in q and not in j. That is, the machine makes no further inference from j. This condition excludes second-part "incremental" codes in which successive independent data items (sentences) are used to refine the distribution used to code the next item. Such codes, while efficient, do not lead to first parts capturing all the regularity in the data, and hence are not a good basis for estimation.

C6 : $\#j < K_T(S)$. This condition requires q to say something significant about the data. The condition is not strictly testable, since $K_T(S)$ is not computable, and in practice the best known upper bound on $K_T(S)$ would have to be used, or $\#S$ if that is smaller.

Suppose S comprises several sentences
$s_1 : s_2 : ... : s_i : ... : s_n$ of L. Then we require

C7 : $j = j_1 : j_2 : ... : j_i : ... : j_n$ where for all $i = 1...n$, $Tq(j_i) = s_i$. This condition forces separate sentences to be treated as independent propositions, given the hypothesis.

C8 : Again supposing S to comprise several sentences, there must exist an equivalent data string $S_p = S_1 : S_2$ where S_1 comprises some of the sentences of S and S_2 comprises the remainder such that the correspondingly-permuted input $I_p = I_1 : I_2$ satisfies
$Tq(I_1) = S_1$ with $\#I_1 < K_T(S_1)$ and
$Tq(I_2) = S_2$ with $\#I_2 < K_T(S_2)$.

C8 is an attempt to require a certain generality in the hypothesis. It means that the hypothesis says something useful about at least two independent subsets of the available data. It would be too strong to require (using the notation of C7) that for all sentences of S, $\#j_i < K_T(s_i)$.

C9 : q has no proper prefix satisfying all the above conditions on q. This merely means that all of q is needed for the specification of the hypothesis.

The above rather untidy collection of conditions is an attempt to specify conditions on the input which make it plausible to identify the first part of the input with the specification of an hypothesis, without requiring an attempt to understand the course of the computation by detailed inspection of I.

In this vein, we now show from C1,C3,C5 and C9 that the division of I into the two parts q and j is unique. Suppose the contrary. Then I can be written as either
$I = q_1 : j_1;\ \ q_1 = a;\ j_1 = b : c;$ or
$I = q_2 : j_2;\ \ q_2 = a : b;\ j_2 = c.$
For any string X:
$T(q_1 : j_1 : X) = T(q_2 : j_2 : X)$
Using C5, $S : T(q_1 : X) = S : T(q_2 : X)$,
and so $T(q_1 : X) = T(q_2 : X)$.
Putting $X = c = j_2$:
$T(q_1 : j_2) = T(q_2 : j_2) = T(I) = S$.
Hence, contrary to C9, $q_2 = a : b$ has a proper prefix $q_1 = a$ satisfying all conditions, so the supposition is false.

4.1. Interpretation of the first part

We have suggested conditions on the input I which are intended to ensure that it can bear the interpretation as a two-part message of which the first part states a general hypothesis. As described in 3.2, we regard an hypothesis as a probability distribution over the sentences of the data language L. If $I = q : j$ is an acceptable MML message causing T to produce a data string S, we need to ask what is the hypothesis represented by the first part q.

The string q has the effect of converting T into an equivalent of another Turing machine Tq. This machine can decode certain strings, e.g. j, into sentences or sequences of sentences in L. Indeed, if S is a concatenation of sentences as in condition C7, we can be sure that Tq can decode a set of input strings into each of these sentences. More generally, Tq can be seen as a decoder for a code over some subset of the sentences of L. The subset contains every sentence s such that there exists a string w with $Tq(w) = s$. The code words of the code are those strings which, when input to Tq, cause it to output a sentence of L. Since the sentences of L are a prefix set, the code so defined is a prefix code, but it is not necessarily non-redundant. There may exist infinite strings which do not begin with a word of the code, and there may be several code words producing the same sentence.

We propose that q be interpreted as a specification, relative to T, of a probability distribution over the sentences of L given by:
$P_{Tq}(s) = 2^{-\#w}$ where w is a shortest code word for sentence s, or
$P_{Tq}(s) = 0$ if there is no code word for s. The prefix property of the code ensures that

$$\sum_s P_{Tq}(s) \leq 1.$$

That is, we propose that with respect to T, q be interpreted as specifying an hypothesis for which Tq decodes a pseudo-optimal code. By this we mean that, if the hypothesis is true, then the code decoded by Tq minimizes the expected length of the code word needed to encode a data sentence. Of course, the code is not truly optimal for this (or any other) hypothesis unless it is non-redundant, which is not necessarily the case.

Note that if the code is non-redundant, equality is reached in the above inequality, and also Tq is clearly not universal. Its only capability is to decode a non-redundant prefix code for sentences. Further, the probability distribution defined by the hypothesis is clearly computable. On the other hand, if the code is redundant, it may be that there exist input strings (not commencing with a code word) which can cause Tq to emulate any UTM. Whether this is so is probably undecideable, in which case the code words cannot be enumerated, and may not correspond to a computable probability distribution over L. This would not necessarily mean the (incomplete) hypothesis defined by the known code word lengths is without meaning and useless. It would simply mean that the hypothesis assigns computable probabilities to some sentences of L, including the sentences in the given data, but not to all possible sentences. It can still be useful in predicting the relative probabilities of potential future sentences which lie within its domain.

Any hypothesis (complete or not) specified as we propose by the first part of an acceptable MML input string is necessarily falsifiable, as we now show. Recall from 3.1 that the data language L is assumed to be non-redundant. That is, its sentences form a complete prefix code for data facts. Hence L defines a "default" probability distribution over its sentences:
$P_L(s) = 2^{-\#s}$ with
$\sum_{s \in L} P_L(s) = 1$
If $I = q : j$ is an acceptable MML input for sentence s, then by C2, $\#I < \#s$ and hence $\#j < \#s$. Hence the Tq code provides a shorter coding of s than does the (complete) L code. It follows that there must be sentences s of L for which either the Tq code has no code word, or the Tq word is longer than s. If future data add enough such sentences to the given data, no input beginning with q will be able to encode the augmented data more briefly than L, and so q will no longer be an acceptable hypothesis.

4.2. Two-part vs. One-part codes

This section has developed a model of MML with an hypothesis set comprising all computable distributions, as a restricted form of Algorithmic Complexity. The restricting conditions C1...C9 are not trivial, so the question arises as to whether our restricted form deviates so far from the AC framework that no interesting correspondence exists. In essence, the question is whether, and under what conditions, we can expect the shortest MML encoding of a data string to have a length close to the AC complexity of the string. Defining $M_T(S)$ as the length of the shortest acceptable MML input to UTM T producing S, it is clear that the difference
$M_T(S) - K_T(S)$ is at least zero.
Li and Vitányi [2, sections 5.2, 5.5] discuss the same question, but assuming a two-string form based on conditional AC complexity. To gloss over the details of their treatment, the conclusion reached is that if the data string is very long, is a "typical" realization of some recursive distribution, and this distribution is "typical" of the distribution over recursive functions implied by the lengths of the shortest programs required to compute them, then the difference is small with high probability, and the "hypothesis" string will probably approximate the distribution from which the data was drawn. Their treatment appears to apply to our two-part form with the further proviso that the independence assumptions made in the data language and enforced by C7 are true of the data source. Indeed, most theoretical results concerning the use of complexity in inductive inference, such as the powerful convergence results of Barron and Cover [1], have assumed a two-part input form similar to ours, rather than the conditional-complexity form (using $K_T(X|Y)$ as in Section 4).

The difference $M_T(S) - K_T(S)$ has an interesting interpretation. $M_T(S) = \#q_S + \#j_S$, where q_S and j_S are the two parts of the shortest MML message. The length $\#q_S$ is the AC complexity with respect to T of the distribution defined by T_{q_S} representing the MML hypothesis Q, so $\#q_S$ may be taken as $K_T(Q)$. By definition, $\#j_S = \log_2(P(S|Q))$. If we treat the AC "probabilities" $P^K_T(.) = 2^{-K_T(.)}$ as ordinary probabilities, then it follows that
$M_T(S) - K_T(S) = -\log_2(P^K_T(Q) \times P(S|Q)/P^K_T(S))$
By Bayes's theorem, the difference is the negative log of the *posterior* probability of the hypothesis Q given the data S, where $P^K_T(Q)$ plays the role of the *prior* probability of Q. Hence, choosing Q to minimize $M_T(S)$ is formally equivalent to choosing the hypothesis of maximum posterior probability given the data.

Further, it follows that for any two hypotheses Q_a and Q_b about data S, the posterior odds ratio in favour of Q_a is
$Prob(Q_a|S)/Prob(Q_b|S) = 2^{(\#I_b - \#I_a)}$
where I_a and I_b are the two shortest inputs which use the respective hypotheses.

5. EXPECTED AND ACTUAL LENGTHS

Some experts in Algorithmic Complexity (AC) see a fundamental difference between algorithmic and MML approaches to the inference of hypotheses, in that the former is concerned with the shortest encoding of the actual given data (with respect to some Turing machine), whereas the latter considers Shannon-optimal codes giving the shortest *expected* length of encodings. This view is mistaken.

The mistake is understandable given the different histories of the streams. In MML, developed within a more-or-less standard statistical inference framework, the probability models were the givens: the standard Gaussian, Binomial, etc. distributions traditionally used to model data. Hence the line of thought went from the probabilities of data strings to the code word lengths needed to encode them. The natural code to use in transmitting data to a receiver who had been informed of the inferred model was a Shannon-optimal code. But in AC theory, the Turing machine was the given, and the line of thought naturally runs from the "code word" strings which encode a data string to the probability of that data string.

In AC theory, the probability associated with an output string is (at least in stream two) the probability that T will emit the output given random input. But suppose that T is such that the "code words" it accepts form a non-redundant code for some set of output strings. (T is then clearly not universal.) Using AC definitions, the T-relative probability of an output S is just the probability that T will emit S given random input, and this is $2^{-\#I}$ where I is the (unique) input producing S, or zero if T cannot produce S. But this probability distribution over outputs is exactly the distribution for which the code decoded by T is Shannon-optimal. That is, T decodes a code for its possible outputs which has minimal expected code word length. Even for universal T, it is reasonable to regard $P^K_T(.)$ as the distribution for which T is a near-optimal decoder.

All streams agree to identify input string length with negative log of probability, so all are dealing with input codes for output strings which are close to optimal in terms of *expected* length. And both streams one (AC) and three (MML/MDL) use the shortest word within this code which will produce the given output S.

It is worth noting that, in the stream-three development of MML-type codes in section 4, the *only* mention of expected lengths occurs in 4.1, where the *interpretation* of hypotheses was discussed. Expected length played no role in the definition or selection of the MML input string. The only criterion was the actual length of the input required for the actual data.

Expected code word lengths do play a role in streams two and three, being an important consideration in the choice of the decoder T. This choice determines the complexities of the distributions which may be employed in coding the data. In two-part codes, it determines the lengths of the shortest first-part q string needed to specify the hypothesis distribution Q, and if T is not universal, as is typically the case in practical applications of MML/MDL, it determines the set G_T of possible hypotheses. In fact, for the given data S, the choice of T limits the usable hypotheses even if T is universal: the only useable hypotheses are those whose complexity $\#q = K_T(Q)$ is less than $\#S$.

In any attempt at inference from a finite body of data, there is always some relevant prior knowledge. In fact, without some prior knowledge of the phenomenon studied, it would scarcely be possible to obtain and record information about it. In practice, even informal everyday inductions are guided by the huge body of knowledge about the world that we all possess. Any serious attempt at inductive inference should make some use of this knowledge, at least to the extent of not biassing the inference towards hypotheses which are ridiculous. In the context of algorithmic complexity, this means choosing T so as to provide short representations of regularities thought most likely to be found in the data, and longer representations for the long shots.

It is most important to note that the choice of T must be made *without knowledge of* S, and using only *genuinely prior knowledge* about the source of the data and the way it will be represented in L. Encoding is a concept which arises in the transmission of information from one agent to another, or in the storage of information for later recall by an agent who does not know, or has forgotten it. In either case, it may reasonably be assumed that the agent who is to decode the information (the "receiver") has some general prior information about the context of the information, and perhaps some prior beliefs about its original source. The encoding can then reasonably be chosen with this prior knowledge in mind. However, an encoding chosen as a function of the information itself will necessarily by unintelligible to the receiver, who by definition does not have the information when he attempts to decode its encoded form. Any coding choices made in the light of the information itself must be stated in the encoded form in a code decodeable by a receiver who as yet does not have the information. They cannot be embodied in T itself, which in our context *is* the receiver. In choosing T, we are or should be attempting to model a receiver with just such knowledge as is genuinely available prior to receipt of the data S.

It should now be clear that whether we are using the MML framework or the stream-one approach, the actual encoding used for a string S is the shortest input which will cause T to produce S. The only difference in approach lies in the choice of T. Whereas the traditional stream-one approach has tended to dismiss the choice of T as unimportant, provided it is universal, in MML and stream two there is an attempt to specify T as a model of a receiver agent informed with relevant prior information. If the inference problem is such that our prior knowledge suffices to exclude all but a limited set of hypotheses, T need not be universal. If, however, we are willing to entertain any hypothesis representable by a computable function, MML will use a universal T, but still one chosen as far as practicable to reflect our relevant prior knowledge. It is in making this choice of T, which is not much addressed in stream one, that *expected* string lengths are of interest. Once the choice is made, and data S obtained, MML chooses its encoding exactly as in the stream-one school, namely, it chooses the shortest which T can decode.

5.1. A related confusion

A related confusion re a "fundamental difference" between streams one and three is best illustrated by an example.

Suppose the data string S is (unbeknown to us) generated by a random process which emits each digit independently, and gives 1s with probability 0.3. If our set of possible hypotheses contains the family of Bernoulli processes with probability parameter p, it is highly likely that a member of G_T decoding a code which encodes digit 1 with length $-\log_2(0.3)$ and digit 0 with length $-\log_2(0.7)$, or values close to these, will provide the shortest two-part encoding of S. But it is possible, if unlikely, that the second part j of the encoding happens to have 90% of "1" digits. Suppose it does. Then the AC school is inclined to suggest that a still shorter encoding will be possible, because j is clearly non-random and can be compressed, whereas the published MML treatments of Bernoulli processes admit no such further compression. It is true that, provided T is universal and S is long enough, there will indeed be a shorter encoding of S in the AC framework. But there will also be one in the MML framework. Even if limited to non-universal machines, the hypothesis set envisaged in the MML analysis can still contain the hypothesis family that the source of S is such that, when data is encoded using an optimal Bernoulli code, the coded string itself will be generated by a Bernoulli process with probability other than 0.5. The corresponding family of hypotheses in the MML G_T is a two parameter family of hypotheses, each being composed of two Bernoulli decoders one feeding the other. If such an hypothesis family is in the MML G_T, then it will provide a shorter encoding than any single-Bernoulli hypothesis, and be the MML inference. If no such family is available in G_T, then the plain Bernoulli hypothesis with p near 0.3 will be inferred, despite the form of j. But this is not because the MML method deals in expected rather than actual lengths, it is because *this* MML analysis did not allow hypotheses capable of representing the particular grotesque regularity which appeared in the data.

Note also that if T is non-universal, then it follows [13, p241][14] that the second-part strings j will appear as if they have been developed by a purely random process, a 0th order Markov model. It necessarily follows that some, if few, of those code-words might appear compressible to some Turing machine (cf. Section 2.3). However, in an efficient code, it is redundant not to use all available code-words.

6. CHOOSING A DECODER

We now describe how the reference Turing machine T is designed in MML inference. There are two creeks contributing to stream three, MML and MDL. Although in practice similar in their results, there are differences in the way the decoding machine is selected, and hence in the way they encode data. We will describe the MML construction, but offer some comments on MDL.

6.1. The MML code

As described in the Introduction, MML inference is normally concerned with selection of an hypothesis or model of the data source from among a limited set of possible hypotheses. Recasting the notation of the Introduction into a form making more reference to decoding machines, we assume that there are given:

A set G of possible hypotheses. This set may contain continua of various dimensions, each spanned by some set of real-valued parameters.

A prior probability function (a density if G is a continuum) $h(H)$ defined on G.

A countable set X_S of possible data strings.

A conditional probability or likelihood function $f(S|H)$ defined for all S and H.

For all H, let Q_H be a Turing machine which decodes an optimal code for the distribution $f(.|H)$. If $f(.|H)$ is computable, as will usually be the case, this code can be a Shannon-optimal code requiring for each S an input $J_{H,S}$ of length $-\log_2(f(S|H))$. If $f(.|H)$ is not computable, Q_H may be chosen to minimize as far as practicable the expected length of the shortest input $J_{H,S}$ which will cause Q_H to produce S, the expectation being taken over the distribution $f(.|H)$ for all S for which $f(S|H) > 0$. We thus select for each hypothesis a machine best able to compress data expected to occur when that hypothesis is true.

For every S, define

$$r_S = \sum_H h(H) \times f(S|H)$$

The sum is replaced by an integral if G is a continuum. r_S is the marginal probability of data S. The construction of T can now proceed in two different ways.

6.1.1. *Strict MML*

Let V be a partition of X_S into non-empty subsets $X_1, X_2, ..., X_j, ...$. For each subset, define

$$r_j = \sum_{S \in X_j} r_S$$

and let Q_j be the machine corresponding to H_j where H_j minimizes

$$L_{V,j} = \sum_{S \in X_j} (r_S / r_j) \times \# J_{H_j,S}$$

This quantity is the weighted average of the lengths of the inputs $J_{H_j,S}$ needed to make machine Q_j produce S, averaged over the strings in X_j. The set of machines $\{Q_j; j = 1, 2, ...\}$ is the set of "useful" Turing machines, and the set of the corresponding hypotheses $\{H_j; j = 1, 2, ...\}$ is the set of "useful" hypotheses which may be used in a two-part encoding of the data.

Let T_V be a Turing machine capable of imitating all the useful machines (so G_{T_V} includes the set of useful hypotheses) and such that the length of the input I_{V,Q_j} needed to make T_V imitate Q_j is minimized in expectation over the distribution $\{r_j; j = 1, 2, ...\}$. In the usual computable case, T_V need decode a Shannon-optimal code for this distribution, and $\# I_{V,Q_j} = -\log_2(r_j)$.

Finally, the partition V is chosen (with consequent choice of the useful machines, their encoding, and T_V) to minimize

$$L_V = \sum_j r_j \times (I_{V,Q_j} + L_{V,j})$$

L_V is the expected length of a two-part encoding of a string drawn from the marginal distribution $\{r_S; S \in X_S\}$ using the "useful" machine Q_j for the subset X_j of X_S containing S.

Discovering the minimizing partition V is an extremely difficult calculation even for very simple and regular functions $h(.)$ and $f(.|.)$. Properties of the resulting inference method have been described elsewhere [13, 14, 8]. Briefly, it has been shown that the method is very general, invariant under arbitrary measure-preserving transformations of G, statistically consistent, and efficient in the sense that the hypothesis giving the shortest two-part encoding of S captures (almost) all the information in S relevant to the choice of hypothesis [13, 1, 14]. Some other widely-used Bayesian point estimators, such as Maximum A Posteriori (MAP) which selects the mode of the posterior density, and the mean of the posterior, are not invariant in this sense. The estimate obtained by such methods depends on how the model space is parameterized.

6.1.2. *Practical MML*

The SMML construction is infeasible for all but the simplest problems [15]. Fortunately, for purposes of inferring an hypothesis from data, it is not actually necessary to construct its encoding. We need only to be able to calculate with sufficient accuracy the length of the encoding for any hypothesis, and then to find the hypothesis which minimizes this approximate length. The practical applications of MML use such approximations. One result useful in many problems is now stated [13]. For a region of G which is a k-dimensional continuum with vector parameter H, prior density $h(H)$ and probability function $f(S|H)$ a member of the exponential family, the length of the shortest encoding of S is approximately

$$\begin{aligned} - \quad & \log(h(H)) - \log(f(S|H)) \\ + \quad & 0.5(\log(F(H)) - k\log(2\pi) + \log(2\pi(k+1))) \end{aligned}$$

where all logs are natural, the length is given in units of $\log_2(e)$ bits, $F(H)$ is the Fisher Information of the probability function, and H is chosen to minimize the expression. The error in this expression is less than 2 for $k \geq 10$, and corrections for smaller k are known. The minimizing H is chosen from the full set G of possibilities, not restricted to the SMML "useful" set, which need not be constructed. The minimizing H is taken as the MML inference.

We emphasize again that the construction outlined here in no way introduces a fundamental distinction between MML and minimizing two-part complexity for hypothesis inference. All that MML does differently is that it gives serious consideration to the choice of reference machine most appropriate for the data at hand, whereas stream one largely does not (see section 7).

For computable $h(.)$ and $f(.|.)$, exactly the same SMML construction can be derived from the aim of producing Bayesian point estimates of high posterior probability, without any reference to Information or AC theory [8].

6.2. Minimum Description Length

The MDL development [3, 16] differs in some respects from the MML approach, although in practical applications, similar results are usually obtained.

First, in MDL the stated aim is most usually to select not a single fully-specified hypothesis, but rather to select a parameterized family (called a "model class"). For instance, in encoding a set of (x, y) values using as G the set of all polynomial functions of the form $y = p(x)$ plus a Gaussian noise term, the MDL emphasis would be on selecting the best *order* of polynomial, with estimation of its coefficients to be considered later, perhaps by different means. Consistent with this aim, MDL defines the "stochastic complexity" of the data with respect to a model class as the length of an encoding of the data using a code optimized for the distribution obtained by integrating over all parameters of the model class with respect to some measure considered appropriate. In practice, the stochastic complexity is often approximated by the code length of a two-part encoding [16], where the first part names one of a discrete set of parameter vectors, and the second codes the data using a code based on the distribution given the stated vector, very much as in MML. Indeed, the discrete set employed is very similar to the set of "useful" models constructed in MML. We have reservations about MDL's emphasis on model classes rather than fully-specified models. It loses the ability of MML to make parameter estimates often superior to the Maximum Likelihood estimates typically used in conjunction with MDL [17, 18, 19, 20]. Also, in some problems e.g. mixture modelling, two competing hypotheses of the same formal structure but differing parameter values may really represent models with markedly different conceptual structure, and lumping these together in the same "class" seems little different from conflating different model classes.

Second, MDL eschews the assumption or use of prior probabilities. The codes used to encode a parameter vector attempt to avoid appeal to priors, instead relying on a "natural" enumeration of the vectors. Similarly, when integration is performed over parameters to obtain an exact stochastic complexity, appeal is made to some "natural" measure on the parameter space. Of course, the resulting code may always be interpreted as the code Shannon-optimal for some prior, and if so interpreted, the "prior" implied by the codes used in practice in MDL would usually not alarm a Bayesian, but the MDL school resists such interpretations. The implied "prior" often resembles the "Jeffreys prior" proportional to the square root of the Fisher Information, which is inadmissible as a genuine Bayesian prior, as it depends on the properties of the observational protocol used to measure the parameters rather than genuinely prior beliefs about their values [21, 22].

Third, a recent development in MDL has been an attempt to shorten the second part of two-part encodings [23]. It is based on the fact that if the hypothesis giving the shortest encoding is always used in encoding data, many data strings which are possible under the probability distribution $f(S|H)$ for some useful H will never in fact lead to the inference of H: a different hypothesis will be preferred. Putting it another way, for a particular H which may be used in a two-part coding, the strings which will ever be coded using the code optimal for $f(.|H)$ include only those strings for which H is the best hypothesis, and will not include many strings which could be coded in this optimal code. It is therefore proposed that the code used when H is stated in the first part be based, not on the distribution $f(.|H)$, but on a truncated form of this distribution,

$$f_c(S|H) = f(S|H) \,/ \sum_{s \in X_H} f(s|H) \quad (S \in X_H)$$

$$f_c(S|H) = 0 \quad (otherwise)$$

Seen as an attempt to approximate the stochastic complexity better, this technique, known as "complete coding", may have virtue. However, in the light of two-part coding it destroys the basis used in MDL for selecting "useful" parameter vectors, as there is now no advantage in code length terms in not using as many vectors as possible. In fact, the shortest length is then obtained by allowing every data string its own hypothesis, thus reducing the length of the second part to zero. When this is done, the minimal code length achieved does not even depend on what hypothesis each string is mapped to, provided only that the string has non-zero probability in that hypothesis! The construction thus provides no basis for selecting one such mapping over another, and hence no useful estimator.

The code length achieved by this "complete" version of MDL for data S is just $-\log_2(r_S)$ where r_S is the marginal data probability defined in 6.1.

Finally, the potential saving in code length is usually very small. It has been shown [13] that the expected saving for regular model classes with k free parameters is about $(1/2)\log(2\pi(k+1))$ nits for large k, and[2] about 0.18 nit for $k = 1$. For models in the exponential family, the greatest saving for any string exceeds the expected saving by less than two bits. For less regular model classes, the best-case saving may be much greater, but for the model classes we have studied, the strings giving substantial savings are rare in the marginal data distribution. These are not just large-sample results: they hold even for small samples.

7. TERMS "OF ORDER ONE"

The stream one literature pays little attention to the choice of the reference UTM T in analysing the behaviour of complexity. The reason usually given is that the difference in the complexities of a string relative to two UTMs can be no more than the length of an interpreter program, which is independent of the string and hence "of order one". We consider this dismissal of order-one terms as misguided and potentially dangerous in any attempt to apply stream one theory

[2] 1 nit $= \log_2(e)$ bits.

to real problems of prediction, estimation and induction. Solomonoff's approach (stream two) does consider terms of order one, and hence the choice of T, to be important. Moreover, since it deals with the relative probabilities of small extensions to the same long data string, using the same reference machine, its results may be relatively insensitive to the choice of T. However, when AC theory is advanced as a means of discovering the pattern (or randomness) of a real, finite data string, the choice of T may be crucial.

First, note that the length of an interpreter making $T1$ imitate $T2$, while fixed independently of the data, may not be small. The length of an actual interpreter to make one general-purpose computer imitate another quite similar computer is typically many thousands of digits. While some of this length is due to speed considerations, much is essential.

Second, many of the results suggesting the inductive prowess of AC rest on derivations which use the two-part input paradigm, e.g. [1]. In obtaining these results indicating consistency, convergence and efficiency, the length of the first part of the input plays a crucial role. It is only the balance between the lengths of the first and second parts which prevents the data string being completely encoded in the first part, which determines whether a string is random or not, and which allows us to assert that the first part captures the regularities in the data leaving the noise in the second part. But what is the crucial length of this first part? It is precisely the length of an interpreter which makes T imitate another Turing machine. That is, it is a length of precisely the same order as that which is dismissed as unimportant in the selection (or rather, non-selection) of T.

Finally, recall that the difference in the lengths of two two-part encodings of the same data S using the same reference machine T can be interpreted as the log of the posterior odds ratio between the two hypotheses used. A difference of 10 bits represents a ratio of approximately 1000:1. Such a likelihood or posterior ratio is usually considered highly significant in statistical inference, and a length difference of 30, giving posterior odds of billions, should be overwhelming. But arbitrary or unwise selection of T can easily make much larger changes in the first-part lengths needed to encode the hypotheses, and so can completely vitiate or even reverse what should be seen as convincing evidence. If complexity-based methods are to be accepted as practical and reliable methods for induction and prediction, they must be less cavalier with "terms of order one".

Some work in the MDL school is also open to a weaker version of this criticism. Different but equally "natural" choices of parameterization and enumeration of model classes can lead to variations in calculated complexity large compared with what should be convincing length differences. Also, some MDL work has replaced the calculation of the log Fisher information, an important component of the first-part length, with the gross approximation $k \log(N)$, where k is the number of parameters of the model class and N the data sample size. The error introduced can easily exceed 1000 in the posterior odds, e.g. in poorly conditioned multiple linear regression problems – regardless of sample size. Similarly, most MDL work [16] has neglected geometric factors arising in the spacing of "useful" parameter vectors, thus introducing a smaller but still significant error in length calculations.

In MML work, considerable care has been given to minimizing such errors [24, 19, 25, 26, 27, 28, 29, 30, 31]. The largest uncertainty in most cases comes from the use of priors chosen for their mathematical convenience as much as their accurate representation of reasonable prior beliefs.

8. PREDICTION AND INDUCTION

In a sense, all *practical* uses of previous data are predictive. We use our past experience to guide our future behaviour according to our predictions of future events. It is therefore reasonable to ask whether inductive inference, in the limited dictionary sense of forming general theories from specific data, need play any role in intelligent reasoning. If we can predict from data without formulating theories about its source, why bother with theories which are inevitably uncertain and usually flawed? Solomonoff's stream-two program seems to offer just such theory-free prediction. Starting with a reference UTM T, and given data S, it allows the inference of the relative probabilities of future events represented by strings N_1, N_2, without ever requiring commitment to any "theory" as represented by the first part of the two-part encodings used in streams one and three.

Part of the answer is simple curiosity: we like to understand the world, and inductively-derived theories help us do so. The more pragmatic desire for theories is that Solomonoff's program is (leaving aside questions of computability) beyond our capacity to execute. We cannot as individuals, or even as a society, record in full all the data which we would like to use for guidance, and we certainly cannot afford the computational effort of computing $P_T(S : N_1)$ / $P_T(S : N_2)$ where S is all previous data, just to predict how high a ball will bounce or how much alkali is needed to neutralize an acid spill.

General theories serve us as useful if imperfect summaries of the *relevant* aspects of previous data, being compact enough to remember or record, and simple enough to give rapid calculation of approximate probabilities for future events. In our discussion, such theories have been represented by the first parts of two-part inputs, or rather, by the Turing machines encoded by these strings. These "theory" machines may be used for prediction. If Tq is the equivalent of reference machine T after T has read the first part q of an MML input for data S, then the Tq estimate of the relative probability of future data strings N_1, N_2 is $P_{Tq}(N_1)/P_{Tq}(N_2)$.

Solomonoff has shown that given T and S, a Turing machine T_S can be constructed such the Solomonoff probability ratio $P_T(S:N_1)$ / $P_T(S:N_2)$ is given by the ratio $P_{T_S}(N_1)$ / $P_{T_S}(N_2)$, so in a sense T_S summarizes the known data S [32]. However, the general construction for T_S requires it effectively to contain in its design a complete copy or representation of S, and in the general case its computation of predictive probability ratios is no faster than that of T. Like the Bourbons, T_S forgets nothing and learns nothing, and its construction was intended as an existence proof rather than as a practical means of prediction.

The pragmatic need for compact theories with readily-computed implications drives scientific enquiry and, on its modest scale, the work on MML. We concede that an MML- or stream-one-derived theory will not predict as well as Solomonoff probability ratios, nor as well as their equivalent in computable domains, the ratios of conventional Bayesian posterior densities as used in Bayesian Decision Theory. However, commitment to a single theory brings great benefits, as it allows deductions to proceed easily. The more accurate alternatives in effect require every possible theory consistent with the data to be given some weight in prediction, and in deductive reasoning from the data.

A compromise is possible. If an analysis of S fails to find a single theory of overwhelming superiority, the few best theories may be retained and predictions based on a weighted average of their various predictions. Examples of this compromise have been developed [33].

9. CONCLUSION

We have argued that, as applied to the inference of models or theories from data, there is no essential difference between Kolmogorov Complexity and Minimum Message Length approaches. They differ only in the choice of reference Turing machines, and in the attention given to this choice. Whereas in stream one, any Universal machine is regarded as acceptable, MML usually (but not necessarily [34]) restricts the reference machine to a non-universal form in the interest of computational feasibility. Further, MML attempts as far as possible to choose the machine so that the complexities of different theories relative to the reference machine reflect the prior probabilities which would be afforded these theories by an intelligent agent knowing the circumstances under which the data was obtained. This attention to the choice of machine allows MML, in domains where the possible theories are computable, to estimate complexities with errors of only a few digits. By contrast, significant terms "of order one" are usually neglected in stream one theory. As a result, MML can be, and has routinely been, applied with some confidence to many problems of machine learning, inductive and statistical inference from finite bodies of real data, whereas a comparison of competing theories using stream one theory would seem to require overwhelming evidence in the data before the approximations in the theory could be safely ignored.

Minimum Description Length is close to MML in practice, and has many successful applications. However, its avoidance of priors and (in some work) less accurate length approximations may make it less reliable than MML.

None of the above techniques is ideally suited to predictive inference. Solomonoff's Algorithmic Probability theory can ideally offer better predictive performance, by taking into account all possible explanations of the known data. Practical application of the ideal Solomonoff model appears very difficult, but Solomonoff has proposed an applicable model based on resource-limited computations of T. For practical prediction from real data, the best approach where feasible may be to form a weighted combination of the predictions from the "best" few theories found by the two-part stream one or MML methods, or (following a suggestion of Solomonoff's), by selecting the prediction yielded by one of the shortest known stream-two encodings of the data, the selection favouring the shorter inputs.

Although not demonstrated in this paper, experience of the authors with complexity-based inference, particularly MML, over the past 30 years has led to some confidence in its properties. In problems where the number of parameters to be estimated grows in proportion to the data, e.g. mixture modelling, factor analysis and the Neyman-Scott problem, MML is known in theory and can be seen in practice to give consistent results, where Maximum Likelihood, Akaike's Information Criterion (AIC) and related classical techniques are known to fail [31, 17, 18, 20].

We have found MML to give statistical estimators with good performance on some difficult distributions [19], and to handle easily complex model selection problems such as the inference of causal nets [35].

Mindful of the Bayesianism inherent in streams one and two (see Sections 5 & 7), this experience has lead the second author to conjecture [36] that only Minimum Message Length, minimum Expected Kullback-Leibler distance and closely related Bayesian techniques can in general infer fully-specified models with both statistical consistency and invariance under 1-to-1 re-parameterisation.

10. ACKNOWLEDGMENTS SECTION

This paper has gained from discussions with Ray Solomonoff, Vladimir Vovk, Paul Vitányi and Jon Oliver, and was supported by Australian Research Council (ARC) Large Grants A49330656, A49703162 and A49602504.

REFERENCES

[1] A. R. Barron and T. M. Cover, Minimum complexity density estimation, *IEEE Trans. Inform. Theory* **37** (1991) 1034–1054.

[2] M. Li and P. M. B. Vitányi, *An Introduction to Kolmogorov Complexity and its Applications* (2nd edition), Springer Verlag, New York, 1997.

[3] J. J. Rissanen, Modelling by shortest data description, *Automatica* **14** (1978) 465–471.

[4] A. N. Kolmogorov, Three approaches to the quantitative definition of information, *Problems of Information Transmission* **1** (1965) 4–7.

[5] G. J. Chaitin, On the lengths of programs for computing binary sequences, *J. Assoc. Comput. Mach.* **13** (1966) 547–569.

[6] R. J. Solomonoff, A formal theory of inductive inference I, II, *Inform. and Control* **7** (1964) 1–22, 224–254.

[7] C. S. Wallace and D. M. Boulton, An information measure for classification, *Computer J.* **11** (2) (1968) 185–194.

[8] C. S. Wallace and D. M. Boulton, An invariant Bayes method for point estimation, *Classification Society Bulletin* **3** (1975) 11–34.

[9] D. M. Boulton and C. S. Wallace, A program for numerical classification, *Computer J.* **13** (1970) 63–69.

[10] D. M. Boulton and C. S. Wallace, An information measure for hierarchic classification, *Computer J.* **16** (1973) 254–261.

[11] D. M. Boulton and C. S. Wallace, An information measure for single-link classification, *Computer J.* **18** (3) (August 1975) 236–238.

[12] D. M. Boulton and C. S. Wallace, The information content of a multistate distribution, *J. Theoret. Biol.* **23** (1969) 269–278.

[13] C. S. Wallace and P. R. Freeman, Estimation and inference by compact coding, *J. Royal Statist. Soc. B* **49** (3) (1987) 223–265.

[14] C. S. Wallace, False oracles and Strict MML estimators, in: D. L. Dowe, K. B. Korb and J. J. Oliver (eds.), *Information, Statistics and Induction in Science: Proc. ISIS '96* (Melbourne, 20–23 August 1996), World Scientific, Singapore, 1996, pp. 304–316. Earlier version: *Technical Report 89/128* (June 1989), Department of Computer Science, Monash University, Australia.

[15] G. E. Farr and C. S. Wallace, The Complexity of Strict Minimum Message Length Inference, *Technical Report 97/321* (August 1997), Department of Computer Science, Monash University, Australia.

[16] J. J. Rissanen, *Stochastic Complexity in Statistical Inquiry*, World Scientific, Singapore, 1989.

[17] C. S. Wallace and P. R. Freeman, Single factor analysis by MML estimation, *J. Royal Statist. Soc. B* **54** (1) (1992) 195–209.

[18] D. L. Dowe and C. S. Wallace, Resolving the Neyman-Scott Problem by Minimum Message Length, *Computing Science and Statistics Proc. 28th Symposium on the interface* (Sydney, Australia, 1997), pp. 614–618; was *Technical Report No. 97/307*, Dept. of Computer Science, Monash University, 1997.

[19] C. S. Wallace and D. L. Dowe, MML estimation of the von Mises concentration parameter, *Technical Report 93/193*, Dept. of Computer Science, Monash University, 1993.

[20] C. S. Wallace, Multiple factor analysis by MML estimation, *Technical Report No. 95/218*, Dept. of Computer Science, Monash University, March 1995.

[21] D.V. Lindley, Bayesian Statistics, A Review, SIAM, Pa. 1972 (p.71).

[22] J. M. Bernardo and A. F. M. Smith, *Bayesian Theory*, Chichester, 1994.

[23] B. E. Dom, MDL estimation for small sample sizes and its application to linear regression, *Technical Report RJ 10030 (90526)* , IBM Almaden Research Division, CA, U.S.A., 1996.

[24] L. Allison, C. S. Wallace and C. N. Yee, When is a String Like a String?, *Proc. Int. Symposium on Artificial Intelligence and Mathematics* (Ft. Lauderdale, Florida, U.S.A., 1990) (1990) 14pp.

[25] C. S. Wallace and D. L. Dowe, MML mixture modelling of multi-state, Poisson, von Mises circular and Gaussian distributions, *Proc. 6th International Workshop on Artificial Intelligence and Statistics* (Ft. Lauderdale, Florida, 4–7 Jan. 1997), pp. 529–536.

[26] J. J. Oliver and C. S. Wallace, Inferring decision graphs, *IJCAI '91 Workshop 8* (Sydney, 1991).

[27] J. J. Oliver, D. L. Dowe and C. S. Wallace, Inferring decision graphs using the minimum message length principle, *Proc. 5th Joint Conf. on Artificial Intelligence* (Hobart), 1992, 361–367.

[28] J. J. Oliver, Decision graphs — an extension of decision trees, *Proc. 4th Internat. Conf. on Artificial Intelligence and Statistics* (Fort Lauderdale, Miami, USA), 1993, pp. 343–350.

[29] R. A. Baxter and D. L. Dowe, Model selection in linear regression using the MML criterion, *Technical Report No. 96/276*, Dept. of Computer Science, Monash University, 1996. Earlier summary in: J. A. Storer and M. Cohn (eds.), *Proc. 4th IEEE Data Compression Conf.* (Snowbird, Utah), IEEE Computer Society Press, Los Alamitos, Ca., 1994, p. 498.

[30] J. D. Patrick and C. S. Wallace, Stone circle geometries: an information theory approach, in: D. Heggie (ed.), *Archaeoastronomy in the Old World*, Cambridge University Press, Cambridge, 1982, pp. 231–264.

[31] C. S. Wallace, An improved program for classification, Proc. 9th Austr. Comput. Sci. Conf. (ACSC-9), *Australian Computer Science Communications* **8** (1) (Feb. 1986) 357–366.

[32] R. J. Solomonoff, Two kinds of probabilistic induction, *Computer J.* *** THIS ISSUE ***

[33] J. J. Oliver and D. J. Hand, Averaging on decision trees, *Journal of Classification* **13** (1996) 281–297.

[34] C. S. Wallace and M. P. Georgeff, A general objective for inductive inference, *Technical Report No. 32*, Dept. of Computer Science, Monash University, March 1983.

[35] C. S. Wallace and K. B. Korb, Learning linear causal models by MML sampling, in: A Gammerman (ed.), *Causal Models and Intelligent Data Management* Springer-Verlag, 1998.

[36] D. L. Dowe, R. A. Baxter, J. J. Oliver and C. S. Wallace, Point estimation using the Kullback-Leibler Loss Function and MML, in *2nd Pacific-Asia Conference on Knowledge Discovery and Data Mining: Proc. PAKDD98* (Melbourne, 15–17 April 1998), Singapore, 1998.