Complexity

When attempting to characterize the complexity of an object, a useful question to ask is: How much information does it contain? In other words, what is the shortest description we can give the object such that no information about that object is lost, that is, it can be accurately reproduced? For our purposes, a description is a program which outputs the object. Kolmogorov complexity is a measure of the information contained in such a description. Specifically, the Kolmogorov complexity of an object is the length (literally the number of 1s and 0s) of the shortest binary string that is sufficient to replicate it. Hence, we have only countably many describable objects.

The first objection one would likely raise at this point is that program length is dependent on language. For instance, some objects are more simply described using C++ than say FORTRAN. It turns out the difference in description length for an object programmed in different languages is bounded by an additive constant. We will use the language of Turing machines.

Definitions and Notation

Definition 1. By associating inputs and outputs, a Turing machine defines a partial function from n-tuples of integers onto the integers, with n ≥ 1. We call such a function partial recursive or computable. If the Turing machine halts for all inputs, then the function computed is defined for all arguments and is called total recursive, or simply recursive.

For some countable set of objects, S, we can assume some standard enumeration where x is associated with a natural number n(x). We want to know if there exists another specification for x above with (x,y). We say additively optimal methods f,g of specifying objects in S are equivalent in the following way:

C(x) = C*(x) ≤ C_f(x) 

for all x, where C_f is a constant depending only on f and g.

There is no universal partial function f for all programs p. However, there does exist a universal element in the class of partial recursive functions. This is a modest and rather natural restriction of our descriptions, as there would be little use in attempting to define the information content of the non-existent output of programs which do not halt. We thus consider the class of description methods \( \phi \) is a partial recursive function. We use \( \phi \) to denote the universal description method, which gives us the following definition (2, 95-97).

Definition 3. Let x,y,p be natural numbers. Any partial recursive function \( \phi \) together with program y and input p, such that \( \phi(y,p) = x \), is a description of x. The complexity \( C_\phi \) of x conditional to y is defined by:

\( C_\phi(x|y) = \min \{ |p| : \phi(y,p) = x \} \)

and \( C_\phi(x) = \infty \) if there is no such p.

By selecting a fixed \( \phi \) as our reference function for C, we can drop the subscript to denote the conditional Kolmogorov complexity where \( C(x|y) = C_\phi(x|y) \). Note the unconditional Kolmogorov complexity \( C(x) = C_\phi(x|0) \).

Two Theorems

The Invariance Theorem and the Incompleteness Theorem form the basis for the whole study of Kolmogorov Complexity, and are sufficient for many important proofs.

Lemma 1. There is a universal partial recursive function.

This result from computability theory generalizes to the Invariance Theorem, which considers the complexity of an object x facilitated by an already specified object y. Recall that Kolmogorov complexity for arbitrarily many conditionals can be defined by recursive use of the bijective pairing function.

The Invariance Theorem 1. There is a universal partial recursive function \( \phi \) for the class of partial recursive functions to compute x given y. Formally this says that \( C(x|y) \leq C(x|z) + c \), for all partial recursive functions \( \phi \) and all x and y, where c is a constant depending on \( \phi \) but not on x or y.

Notice that the universal description method may not give the shortest description for all x, but no other method gives a shorter description for more than finitely many cases. We also note a trivial upper bound given by the following lemmas (but omit the proofs for brevity).

Lemma 2. There is a constant c such that for all x and y \( C(x) \leq C(x|y) + c \) and \( C(x|y) \leq C(x) + c \).

In the case of objects conditionally belonging to finite sets, we can offer an improved upper bound with the following lemma.

Lemma 3. \( C(x|A) \leq C(x) + \log|A| + c \), where \( C(x) \leq n \).

The Incompleteness Theorem.

Definition 4. For each \( \alpha \), there is an \( \epsilon \)-incompletable if \( C(x) \geq \alpha \).

The Incompleteness Theorem 2. Let \( x \in N^* \). For each fixed \( \epsilon \), any finite set A of cardinality m has at least \( (1 - \epsilon)^{-m} \) elements x with \( C(x) \geq \alpha \).

What we see from this theorem is the fairly surprising result that of all binary strings of length n, at least half of them can only be compressed by no more than one digit. Another quarter or more of the strings can only be compressed by at most 2 digits, and so on. This itself has some rather counter intuitive results.

For instance, if x is an incompletable string, all substrings in x are also incompletable. Intuitively, the ability to compress a substring would seem to give us a means to compress x. We can place a lower bound on substring x given by \( C(x) \leq C(y) + O(\log n) \) but cannot prove \( C(x) \leq C(y) - O(1) \). If the latter were true, x could contain no long regular subsequences since, for example, a sequence of k zeros has complexity O(log k). But for strings of length n, only a small subset have no regular substrings, which gives us an easy way to describe them. Thus, for x to be incompletable, it must have compressible substrings (2, 112).

Graphs

Canonically, a graph \( G \) = \( (V,E) \) with \( n \) vertices labeled \( 1, \ldots, n \) is encoded as an \( n(n-1)/2 \) length string \( E(G) \) where each bit corresponds lexicographically to a vertex pair. Thus \( E(G) = 1 \) if \( (1,2) \in E \), \( (1,3) \in E \), \( (1,4) \in E \), \( (1,5) \in E \), \( (1,6) \in E \), \( (1,7) \in E \), \( (1,8) \in E \), \( (1,9) \in E \), \( (1,10) \in E \), \( (2,3) \in E \), \( (2,4) \in E \), \( (2,5) \in E \), \( (2,6) \in E \), \( (2,7) \in E \), \( (2,8) \in E \), \( (2,9) \in E \), \( (2,10) \in E \), \( (3,4) \in E \), \( (3,5) \in E \), \( (3,6) \in E \), \( (3,7) \in E \), \( (3,8) \in E \), \( (3,9) \in E \), \( (3,10) \in E \), \( (4,5) \in E \), \( (4,6) \in E \), \( (4,7) \in E \), \( (4,8) \in E \), \( (4,9) \in E \), \( (4,10) \in E \), \( (5,6) \in E \), \( (5,7) \in E \), \( (5,8) \in E \), \( (5,9) \in E \), \( (5,10) \in E \), \( (6,7) \in E \), \( (6,8) \in E \), \( (6,9) \in E \), \( (6,10) \in E \), \( (7,8) \in E \), \( (7,9) \in E \), \( (7,10) \in E \), \( (8,9) \in E \), \( (8,10) \in E \), \( (9,10) \in E \).

Acknowledgments

I would like to thank my advisor, Prof. Ran Libeskind-Hadas for his support, assistance, and direction throughout my research. I would also like to thank Prof. Michael Orrison, Prof. Art Benjamin, and Prof. Henry Krieger for their input and assistance at various sticking points during the year, and Prof. Ming Li at the University of Waterloo for allowing me to take his course on Kolmogorov Complexity.

References