Kolmogorov Complexity

1 What Is Randomness?

Some sequences are random, some are not. Let us flip a coin many times and every time, mark 1 if it falls heads and 0 if it falls tail. Which sequences of 0s and 1s will we get?

- If we get a sequence $11\ldots1$ of one thousand 1 symbols, something is clearly wrong, this cannot be a fair coin.
- Can we have a sequence $01010101\ldots01$ in which a string 01 is repeated 1000 times? This is also very suspicious, this sequence is too regular to become the result of a random process.

Imagine that in a lottery, for several years in a row:

- every even year, the winning ticket ends on an even digit, and
- every odd year, it end on an odd digit.

This could happen 1 year, it could happen for 10 years in a row. However, if this happens for 20 (or even 30) years in a row, the police will start investigating – clearly the results are not random.

Why are we sure that a sequence 0101... is not random. We are sure because this sequence can be generated by a simple – thus short – program:

```
for(i = 1; i <= 1000; i++)
  {System.out.print("01");}
```

On the other hand, if we have the actual sequence $01\ldots$ of 0s and 1s obtained from coin flipping, we do not expect any patterns there. For such sequences, the only way to print is to actually write

```
System.out.print("01..");
```

The length of this program – i.e., the number of symbols in it – is about the same as the length $\text{len}(x)$ of the sequence x of 0s and 1s: we have symbols "S", "y", "s", etc., and then symbols "0", "1", etc., corresponding to the bits that form this sequence.

So, we arrive at the following conclusion:

- if a sequence x can be generated by a program p whose length is much shorter than its length $\text{len}(x)$, then the sequence x is *not* random;
- on the other hand, if the shortest program for generating x has length approximately equal to $\text{len}(x)$, this means that the sequence x *is* random.

In other words, whether the sequence is random or not depend on the relation between:

- the smallest length of the program that generates x, and
- the length of the sequence x itself.

2 What Is Kolmogorov Complexity, Why It Is Not Computable, and What Can We Say about It

The notion of Kolmogorov complexity. The shortest length of the program p that generates a sequence x is called the *Kolmorogov complexity* $K(x)$ of this sequence.

Comment. In these terms, a sequence x is random if $K(x) \approx \text{len}(x)$.

To make this precise, we can select some small integer C, and say that a sequence is random if $K(x) \geq \text{len}(x) - C$.

Kolmogorov complexity is not computable. Interestingly, $K(x)$ is not computable. This can be proved by contradiction.

First, note that there exist sequences with arbitrary large $K(x)$. Indeed:

- if there was a bound B such that every binary sequence x has $K(x) \leq B$,
- this would mean that every binary sequence can be generated by a program of length $\leq B$.

But:

- there are only *finitely many* such programs, no more than S^B, where S is the overall number of ASCII symbols, while
- there are *infinitely many* possible sequence.

So, for every B, there exists a sequence x for which $K(x) > B$.

Suppose now that the Kolmogorov complexity $K(x)$ *is* computable by some program. Let us denote the length of this program by L.

- As we did in our previous proofs, we can assign, to each binary sequence x, a natural number by appending 1 in front of this sequence.
- Vice versa, for each positive natural number n, we can strip off the first 1 from its binary expansion and get a binary sequence that we will denote by $x(n)$.

Then – similarly to what we did for mu-recursive functions – we can write the following program for computing the smallest number n_0 for which $K(x(n_0)) > 3L + 100$:

```
int n = 0;
while(K(x(n)) <= 3 * L + 100)
  {n++;}
```

This program has fewer than 100 symbols: symbol "i", symbol "n", symbol "t", etc. Thus, if we add a program of length L to computing $K(x)$, we will get a program for computing $x(n_0)$ whose length is smaller than $L + 100$. So, the shortest length $K(x(n_0))$ of the program for computing $x(n_0)$ cannot exceed the length of the above program, so

$$K(x(n_0)) < L + 100.$$

On the other hand, by definition of n_0, we have

$$K(x(n_0)) > 3L + 100 > L + 100.$$

The two inequalities cannot be both true, so we get a contradiction. This contradiction shows that Kolmogorov complexity is *not* computable.

So what can we say about $K(x)$? We cannot compute $K(x)$, but we can provide an upper bound on $K(x)$. Indeed:

- $K(x)$ is the length of the shortest program for computing x;
- so, if we know any program p for computing x, we can count the number of symbols $\text{len}(p)$ in this program and conclude that $K(x) \le \text{len}(p)$.

Example. The above program p for computing the sequence $x = 0101\ldots01$ consists of $\text{len}(p) = 50$ symbols, so we can conclude that $K(x) \le 50$.

3 Barry's Paradox

Paradox. The above proof – that $K(x)$ is not computable – was obtained by modifying the following paradox.

- Some texts describe natural numbers. For example, "twenty three" or "nineteen plus four" both describe the number 23.
- There are finitely many words – half a million in English.
- So, if we use a fixed number of words, we will only be able to describe *finitely many* numbers.
- However, there are *infinitely many* natural numbers.

- Thus, for any given number of words N, there exist numbers that cannot be described by fewer than N words.
- Let us consider, for $N = 20$, the smallest of such words, i.e., "the smallest natural number that cannot be described by fewer than twenty words".
- This description contains only 13 words, which is smaller than 20 – while this should be a number that cannot be described in fewer than 20 words – a contradiction.

How is this paradox resolved. The usual solution is that natural language is often ambiguous.

If we try to formalize it – e.g., by replacing "defined" by "computed" – we get exactly the above proof that Kolmogorov complexity if not computable.