Kolmogorov Complexity and Incompleteness

Ma 236 - Introduction to Mathematical Reasoning

Spring 2017

1 The Richard-Berry Paradox

Define a natural number as "the least number which cannot be described in fewer than 100 characters". (A character is a letter, numeral, space, or punctuation mark.) If such a number exists, we have just described it in 71 characters. If it does not exist, then every natural number can be described in less than 100 characters. But there are infinitely many natural numbers and only finitely many descriptions less than 100 characters long.

2 Descriptions

One way out of the Richard-Berry Paradox is to define descriptions formally. First we fix a set of characters. We choose the 95 printable ASCII characters listed in Table 1, but the particular choice is not important.

Definition 1. A *word* is any sequence of characters.

For example $x\$yZA;b$ is a word of length 8 (there is a space between Z and A). Notice that 76543210 is also a word of length 8; words include numbers.

Here is our formal definition of descriptions.

Definition 2. A description of a word is a program which prints out that word together with any data needed by the program.

Most any programming language can be used, but we assume that programs and their data are written in the characters of Table 1. A program and its data are a special kind of word.

00	SP	19	3	38	F	57	Y	76	l
01	!	20	4	39	G	58	Z	77	m
02	"	21	5	40	H	59	[	78	n
03	#	22	6	41	I	60	\	79	o
04	$	23	7	42	J	61	]	80	p
05	%	24	8	43	K	62	^	81	q
06	&	25	9	44	L	63	_	82	r
07	'	26	:	45	M	64	`	83	s
08	(	27	;	46	N	65	a	84	t
09	)	28	<	47	O	66	b	85	u
10	*	29	=	48	P	67	c	86	v
11	+	30	>	49	Q	68	d	87	w
12	,	31	?	50	R	69	e	88	x
13	-	32	@	51	S	70	f	89	y
14	.	33	A	52	T	71	g	90	z
15	/	34	B	53	U	72	h	91	{
16	0	35	C	54	V	73	i	92	|
17	1	36	D	55	W	74	j	93	}
18	2	37	E	56	X	75	k	94	~

Table 1: 95 ASCII characters

According to Definition 2 a program which just prints out its data is a description of that data. Depending on our programming language such a program might look like

w=ReadData()
output (w)

Strictly speaking since we are using the characters from Table 1, this program with input, say, 218376, would be written as the word

$$w = ReadData(); output(w);;218376$$

where we are using ";" to separate lines and ";;" to mark the end of the program. However, in the interest of legibility we will feel free to display programs as above.

3 Complexity

Definition 3. The Kolmogorov complexity, $C(w)$, of a word w is the length of the shortest description of w.

It is not easy to find the shortest description for a word w, but we can always use the program from the preceding section with w as data. That program

has length 26, so for any word w we have $C(w) \leq 26 + ||w||$ where $||w||$ is the length of w. Of course the constant 26 will vary depending on the programming language we use.

Theorem 4. *There is a constant c such that for every word w, $C(w) \leq c + ||w||$.*

We will never need to know the exact length of the programs we use.

4 Incompressibility of Words

Definition 5. A word w with $C(w) \geq ||w||$ is called *incompressible*. Every description of an incompressible word is at least a long as the word itself.

Theorem 6. *Incompressible words of every length exist.*

The proof is by counting. There are a lot more words of length n than descriptions of length less than n, so most words of length n do not have descriptions of length less than n.

Consider words of length 3. There are 95 choices for the first character. For each such choice there are again 95 choices for the second character etc. The total number of words of length 3 is

$$95 \times 95 \times 95 = 857375.$$

Now count the number descriptions of length less than 3. We use a crude estimate. Each description is a word, so the number of descriptions of length less than 3 is at most equal to the number of words of length 0, 1 or 2 which is

$$1 + 95 + 95 \times 95 = 9121.$$

At most $\frac{9121}{857375} \simeq 1.06\%$ of words of length 3 are compressible.

The same crude estimate works for all n. The number of words of length n is

$$95^n$$

and the number of words of length at most $n-1$ is

$$1 + 95 + 95^2 + \cdots + 95^{n-1} = \frac{95^n - 1}{95 - 1} < \frac{95^n}{95 - 1} = \frac{1}{94} 95^n$$

by the formula for a geometric series.

We see that most $\frac{1}{94} \simeq 1.06\%$ of all words of length n are incompressible. The vast majority of words are incompressible.

5 Compressibility is Undecidable

Suppose there were a program which could tell whether or not a word was compressible. Then there would be a program which on input n tested each word of length n to see if it was compressible and printed out the first one which was not. (By Theorem 6 there is at least one.) Thus for that number x we would have

$$n \leq C(x) \leq c + \text{ the number of digits of } n$$

Suppose n has 6 digits. Since the smallest number with 6 digits is $100000 = 10^5$, we would have

$$10^5 \leq n \leq C(x) \leq c + 6 \quad \text{and} \quad 10^6 \leq c + 7 \quad \text{and so on}$$

which is impossible.

Theorem 7. *Compressibility is not decidable. Another way of saying the same thing is that the set of compressible words is not computable.*

6 Computably Enumerable Subsets

Although compressibility is undecidable, there is a program $\mathcal{P}$ which prints out a list of all compressible words and only those. Of course this list is infinite and $\mathcal{P}$ runs forever.

Why can't we tell whether of not a word is compressible by looking to see if it is on the list? If it is, we will eventually see it; but if it is not, we will never know.

$\mathcal{P}$ works in the following way: it makes a list of the first 100 descriptions and runs each of them for 100 steps. If a description halts within 100 steps and is shorter than its output, $\mathcal{P}$ adds that output to the list of compressible words. Otherwise $\mathcal{P}$ goes on to the next description.

When $\mathcal{P}$ is through with the first 100 descriptions, it does the same thing for the first 1000, and then for the first 10000, etc. Every word added to the list is compressible, and every compressible word is eventually added.

Sets of words whose elements can be listed by a program are called computably enumerable. It does not matter whether an element occurs once or many times, but each element which is listed must be on in the set.

Theorem 8. *The set of compressible words is computably enumerable but not computable.*

7 A Computably Enumerable Subset of N

For our purposes we need a computably enumerable but not computable subset of N. $\mathcal{P}$ almost works except it lists words instead of numbers.

Change $\mathcal{P}$ slightly so that instead outputting a word w it outputs first a 1 and then the numbers 00 - 94 from Table 1 corresponding to the ASCII characters in w. For example instead of

$$\$\$ABCD$$

$\mathcal{P}$ would output

$$1040433343536$$

Given a word w it is easy to compute the corresponding number and vice-versa. Thus if the set of numbers were computable the set of word wold be too.

Theorem 9. *There is a subset of N which is computably enumerable but not computable.*

8 Formulas in Arithmetic

We need a little more preparation for the next section.

Take a sentence with quantifiers and remove one of the quantifiers. The result is a formula. For example the sentence

$$\forall x \, \exists y \, y + y = x$$

yields two formulas.

$$\exists y \, y + y = x \text{ and } \forall x \, y + y = x.$$

Each of the formulas above has a single free (unquantified) variable and is true or false depending on the value of that variable in N. The set of numbers which make the formula true is called the subset of N defined by the formula.

We see that $\exists y \, y+y = x$ defines the even numbers while $\forall x \, y+y = x$ defines the empty set.

A subset of $S \subset N$ defined by a formula is called *arithmetic*.

We will make us of the following important and well known theorem proved by Martin Davis in his doctoral dissertation written under the supervision of Alonzo Church.

Theorem 10 (Martin Davis (1949))**.** *Every computably enumerable subset of N is arithmetic.*

This theorem together with Theorem 9 gives us what we need.

Corollary 11. *There is a subset of N which is arithmetic but not computable.*

9 The Incompleteness Theorem

.

Recall the seven axioms of Robinson arithmetic ($Q1, \ldots, Q7$) plus the rule of inference for induction. We will show that there are is a sentence which is true in N but cannot be proved from $Q1, \ldots, Q7$ plus induction (or in fact from any reasonable set of axioms).

By Corollary 11 there is a formula $\phi(x)$ which is true if x is in a certain arithmetic but uncomputable subset $S \subset N$, and $\neg\phi(x)$ is true otherwise.

To check if a number n is in S apply the tree test to $\phi(s^n 0)$ and $\neg\phi(s^n 0)$ in parallel. If our axioms are strong enough to prove every true statement, the tree test will eventually show that one of these sentences is true. But then we have shown that membership in S is computable, which is not the case. Thus there is some sentence $\phi(s^n 0)$ or $\neg\phi(s^n 0)$ which is true but not provable.