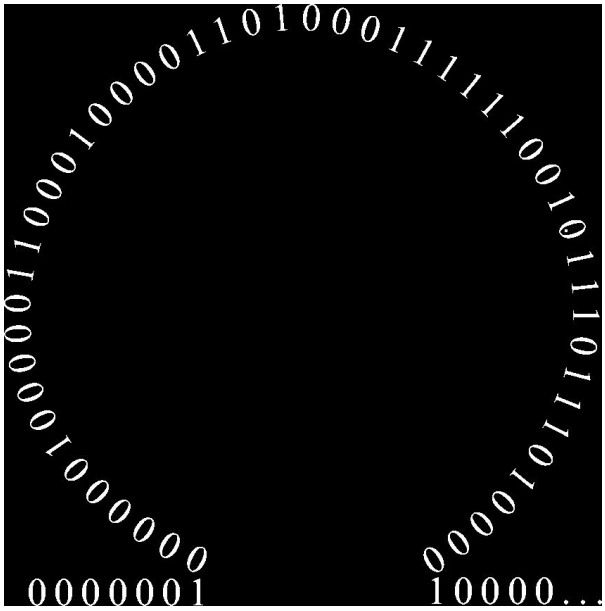
COMPUTING A GLIMPSE OF RANDOMNESS

Cristian S. Calude
(joint work with M.J. Dinneen and C.-K. Shu)
University of Auckland

Chaitin Omega Numbers are halting probabilities of universal self-delimiting Turing machines. They have been described as numbers which "can be known of, but not known, through human reason." The talk will review the main properties of Omega Numbers and will describe the main steps (mathematical and programming) leading to the computation of 64 exact bits of a (natural) Omega Number:



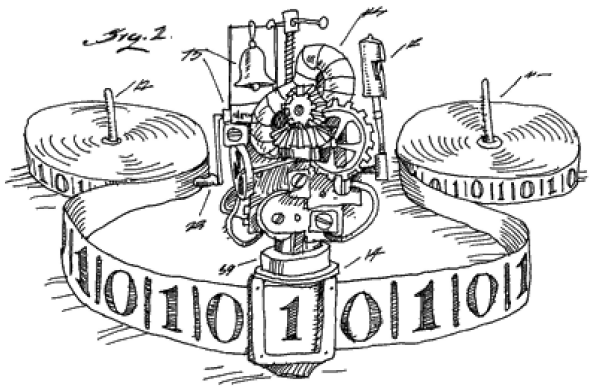
Reals like $\alpha = \pi, e, \sqrt{2}, \log_2 3, \sqrt{5}$ (in fact all "natural" constants) are computable because there is a computable function $f : \mathbf{N} \to \{0, 1\}$ such that

$$\alpha = \sum_{i=1}^{\infty} f(i) 2^{-i}.$$

This is equivalent to say that the left Dedekind cut $\{q \in \mathbf{Q} \mid q < \alpha\}$ is computable.

Most numbers are not computable. Specker (1949) gave the first example of a real α which is not computable, a real for which the set $\{q \in \mathbf{Q} \mid q < \alpha\}$ is computably enumerable but not computable. Such a number is called *computable enumerable (c.e.)*.

We fix with a universal self-delimiting Turing machine (TM) U working on $\Sigma = \{0, 1\}$, i.e. a universal TM whose domain $\{x \in \Sigma^* \mid U(x) \text{ halts}\}$ is prefix-free.



Denote by $string(i)$ the ith binary string. The real

$$c_U = \sum_{U(string(i)) \text{ halts}} 2^{-i}$$

is c.e. and not computable because the Halting problem is undecidable.

To know which programs of less than N bits halt we need to know $2^{N+1} - 1$ bits of c_U. Can we do it better?

Knowing the first N bits of the Chaitin's Omega real

$$\Omega_U = \sum_{U(x) \text{ halts}} 2^{-|x|}$$

we can decide which program of less than N bits halts. Ω_U is c.e. and not only non-computable, but algorithmically *random*.

Randomness means that Ω_U passes all c.e. statistical tests of randomness. For example, every binary N bit string appears in the binary expansion of Ω_U with the probability 2^{-N}.

Chaitin (1975), Calude, Hertling, Khoussainov, Wang (1998) and Slaman (1998) *The set of Ω_U coincides with the set of c.e. random reals.*

Omega "can be known of, but not known, through human reason"

(Chaitin, 1975) *Assume that ZFC is arithmetically sound. Then ZFC can determine the value of only* finitely *many bits of* Ω_U.

(Calude, 2000) *For very c.e. random real we can effectively construct a self-delimiting machine U such that ZFC, if arithmetically sound, cannot determine more than the* initial block of 1's *of the binary expansion of* Ω_U.

(Solovay, 1999) *We can effectively construct a self-delimiting machine U so that ZFC, if arithmetically sound, cannot determine* any *bit of the* Ω_U..

Is it important to "know" Ω_U?

[a natural Omega] embodies an enormous amount of wisdom in a very small space ... inasmuch as its first few thousands digits, which could be written on a small piece of paper, contain the answers to more mathematical questions than could be written down in the entire universe. (C. Bennett)

This number is so random that no program can ever be written to print out even one of its digits. (T. Hoare)

Yet...

Computing a few exact bits of a "natural" Ω_U

We have worked with the universal machine constructed in 1987 by Chaitin using register machine programs. A register machine has a finite number of registers, each of which may contain an arbitrarily large non-negative integer. The list of instructions is given below:

```
L:  GOTO  L2
L:  JUMP  R  L2
L:  GOBACK  R
L:  EQ  R1  R2  L2
L:  NEQ  R1  R2  L2
L:  RIGHT  R
L:  LEFT  R1  R2
L:  SET  R1  R2
L:  HALT
L:  OUT  R
L:  DUMP
```

A *register machine program* consists of a finite list of labeled instructions from the above list, with the restriction that the `HALT` instruction appears only once, as the last instruction of the list. The data (a binary string) follows immediately the `HALT` instruction. The use of undefined variables is a run-time error. A program not reading the whole data or attempting to read past the last data-bit results in a run-time error. Because of the position of the `HALT` instruction and the specific way data is read, register machine programs are self-delimiting.

A Java version interpreter for register machine programs has implemented the universal machine. This interpreter has been used to test the Halting Problem for all register machine programs of at most 84 bits long.

To minimize the number of programs of a given length that need to be simulated, we have used "canonical programs" which are register machine programs in which
(1) labels appear in increasing numerical order starting with 0,
(2) new register names appear in increasing lexicographical order starting from 'a',
(3) there are no leading or trailing spaces,
(4) operands are separated by a single space,
(5) there is no space after labels or operators,
(6) instructions are separated by a single space.

For every register machine program there is a unique canonical program which is equivalent to it. If x is a program and y is its canonical program, then $|y| \leq |x|$.

Here is a canonical program:

```
0:!a 1:^b 4 2:!c 3:?11 4:=a 0 8 5:&c 110 6
:(c 101 7:@b 8:&c 101 9:(c 113 10:@b 11:%10
```

or in a more "understandable" form:

```
0:! a          // read the first data bit into
               // register a
1:^ b 4        // jump to a subroutine at line 4
2:! c          // on return from the subroutine
               //call c is written out
3:? 11         // go to the halting instruction
4:= a 0 8      // the right most 7 bits are
               // compared with 127; if they
               // are equal, then go to label 8
```

```
5:& c ‘n’      // else, continue here and
6:( c ‘e’      // store the character string ‘ne’
               // in register c
7:@ b          // go back to the instruction with
               //label 2 stored in register b
8:& c ‘e’      // store the character string ‘eq’
               // in register c
9:( c ‘q’
10:@ b
11:%           // the halting instruction
10             // the input data
```

1. Start by generating all programs of 7 bits and test which of them stops. All strings of length 7 which can be extended to programs are considered prefixes for possible halting programs of length 14 or longer; they will simply be called *prefixes*. In general, all strings of length n which can be extended to programs are *prefixes* for possible halting programs of length $n + 7$ or longer. *Compressed prefixes* are prefixes of compressed (canonical) programs.

2. Testing the Halting Problem for programs of length $n \in \{7, 14, 21, \ldots, 84\}$ was done by running all candidates (that is, programs of length n which are extensions of prefixes of length $n - 7$) for up to 100 instructions, and proving that any generated program which does not halt after running 100 instructions never halts. For example, (uncompressed) programs that match the regular expression `"0:\^ a 5.* 5:\? 0"` never halt on any input.

Statistics of halting programs

P + D L = program plus data length,
 # HP = number of halting programs

P + D L	**# HP**	**P + D L**	**# HP**
7	1	49	1012
14	1	56	4382
21	3	63	19164
28	8	70	99785
35	50	77	515279
42	311	84	2559837

Computing all halting programs of up to 84 bits for U seems to give the exact values of the first 84 bits of Ω_U. False! Indeed, given the first N bits of Ω_U one can solve the Halting Problem for programs of length at most N, but the converse implication is *false* because longer halting programs can conspire to modify values of "very early" bits of Ω_U.

So, to be able to compute the exact values of the first N bits of Ω_U we need to be able to *prove* that longer programs do not affect "too much" the first N bits of Ω_U.

Due to the special form of programs, the "tail" contribution to the value of

$$\Omega_U = \sum_{n=0}^{\infty} \sum_{\{|w|=n,\, U(w)\ \text{halts}\}} 2^{-|w|}$$

is bounded from above by the sum of the following two convergent series:

$$\sum_{m=0}^{\infty} \sum_{n=k}^{\infty} \underbrace{\#\{x \mid \text{prefix } x \text{ not containing HALT}, |x| = k\}}_{x} \cdot \underbrace{48^{n-k}}_{y} \cdot \underbrace{1}_{\text{HALT}} \cdot \underbrace{2^m}_{u} \cdot 128^{-(n+m+1)},$$

and

$$\sum_{m=1}^{\infty} \underbrace{\#\{x \mid \text{prefix } x \text{ containing HALT}, |x| = k\}}_{x} \cdot \underbrace{2^m}_{u} \cdot 128^{-(m+k)}.$$

Plugging in the data obtained from simulating the programs we get:

$$\begin{aligned}
&\sum_{m=0}^{\infty}\sum_{n=13}^{\infty} 402906842 \cdot 48^{n-13} \cdot 2^m \cdot 128^{-(n+m+1)} \\
&\quad + \sum_{m=1}^{\infty} 1748380 \cdot 2^m \cdot 128^{-(m+13)} \\
&\quad = 402906842 \cdot \frac{64}{128 \cdot 48^{13}} \cdot \sum_{n=13}^{\infty} \left(\frac{48}{128}\right)^n \\
&\quad + 1748380 \cdot \frac{1}{63 \cdot 128^{13}} \\
&\quad < 2^{-68},
\end{aligned}$$

that is, the first 68 bits of Ω_U^{84} "may be" correct by our method.

Actually we do not have 68 correct bits, but *only* 64 because adding a 1 to the 68th bit may cause an overflow up to the 65th bit:

$$\Omega_U^{84} = 0.\underline{000000100000010000011000100001101000111100101110111010000100000}11110$$

In conclusion,

Calude, Dinneen, Shu (December 2001) *The first 64 bits of* Ω_U *are*:

0000001000000100000110001000011010001111100101110111010000100000

Conclusions

This is the first attempt to compute some initial exacts bits of a random real. The method, which combines programming with mathematical proofs, can be improved in many respects.

In contrast with the computation of π, for which you can compute as many digits as you want (provided you have enough time and money) this method is essentially non-scalable: only finitely digits can be computed.

Knowing the N digits of Ω_U allows you to solve the Halting Problem for programs of up to N bits but the converse is not true. However, with an oracle for the Halting Problem you *can* get as many digits as you want from Ω_U.

We have solved the Halting Problem for programs of at most 84 bits, but we have obtained only 64 exact initial bits. This (and any other) method will fail when the first program corresponding to an unsolved problem is generated (e.g. a program for Goldbach's conjecture)

Our Ω_U starts with 0, still there is no contradiction between Solovay's result and our computation. Each computably enumerable real $\alpha = \Omega_U$ for infinitely many U. Among them, we have Solovay "bad" machines for which ZFC cannot determine any digit. Fortunately, each computably enumerable real α can be defined as the halting probability of a universal machine which is not a Solovay machine.

The web site

`ftp://ftp.cs.auckland.ac.nz/pub/CDMTCS/Omega/`

contains all programs used for the computation as well as all intermediate and final data files (3 giga-bytes in gzip format).