THE LIMITS OF MATHEMATICS
(Extended Abstract)

G. J. Chaitin
IBM Research Division
chaitin@watson.ibm.com

In June 1994 I gave a five-day course on the limits of mathematics at the University of Maine in Orono. This course featured a new approach to algorithmic information theory (AIT). Four versions [1]–[4] of the course notes for this course, each using a somewhat different approach, are available. To automatically obtain any one of them in \TeX, for example \texttt{chao-dyn/9407003}, send e-mail to “chao-dyn \texttt{@ xyz.lanl.gov}” with “Subject: get 9407003”.

AIT deals with program-size complexity. I define the complexity \( H(X) \) of an object \( X \) to be the size in bits of the smallest program that can calculate \( X \). Up to now, to get elegant mathematical properties for this complexity measure \( H(X) \), I had to measure the size of programs for an abstract universal Turing machine. This gave the right mathematical properties, but it was not a programming language that anyone could actually use. Now I have found a way to obtain the correct program-size complexity measure of AIT by measuring the
size of programs in a series of powerful and easy to use programming languages. These programming languages are versions of LISP that I have invented expressly for this purpose. Which of these programming languages one considers most natural is to a certain extent a matter of personal taste.

What does AIT have to say concerning the limits of mathematics? My theory yields two fundamental information-theoretic incompleteness theorems. First of all, my theorem, originally going back to 1970, that an \( N \)-bit formal axiomatic system cannot enable one to exhibit any specific object \( X \) with program-size complexity \( H(X) \) greater than \( N + c \). Secondly, my theorem, originally going back to 1986, that an \( N \)-bit formal axiomatic system cannot enable one to determine more than \( N + c' \) scattered bits of the halting probability \( \Omega \). In chao-dyn/9407003, \( c = 2359 \) bits and \( c' = 7581 \) bits. In chao-dyn/9407005, \( c = 1127 \) bits and \( c' = 3689 \) bits. In chao-dyn/9407006, \( c = 994 \) bits and \( c' = 3192 \) bits. And in chao-dyn/9407009, \( c = 735 \) bits and \( c' = 2933 \) bits.

I think I prefer the “aggressive” formulation in chao-dyn/9407009. I can also make a case for the “conservative” formulation in chao-dyn/9407003. chao-dyn/9407005 and chao-dyn/9407006 are the intermediate steps between chao-dyn/9407003 and chao-dyn/9407009.

After the references we summarize chao-dyn/9407003 in a four-page appendix. The first page is a table summarizing the version of LISP that is used. The second page is an example of a program written in this LISP. The third page summarizes the definitions, and the fourth page summarizes the results.

References

[1] G. J. Chaitin, “The Limits of Mathematics,” IBM Research Report RC 19646, e-print chao-dyn/9407003, July 1994, 270 pp.

[2] G. J. Chaitin, “The Limits of Mathematics II,” IBM Research Report RC 19660, e-print chao-dyn/9407003, July 1994, 255 pp.

[3] G. J. Chaitin, “The Limits of Mathematics III,” IBM Research Report RC 19663, e-print chao-dyn/9407006, July 1994, 239 pp.
[4] G. J. Chaitin, “The Limits of Mathematics IV,” IBM Research Report RC 19671, e-print chao-dyn/9407009, July 1994, 231 pp.
| Symbol | Description | Arguments | Result |
|--------|-------------|-----------|--------|
| `'`    | quote       | 1 arg     | `(abc) → (abc)` |
| `+`    | head        | 1 arg     | `+(abc) → a`  |
| `-`    | tail        | 1 arg     | `-a → a`      |
| `*`    | join        | 2 args    | `*a(bc) → (abc)` |
| `. `   | atom        | 1 arg     | `.a → 1`      |
| `=`    | equal       | 2 args    | `=aa → 1`     |
| `/`    | if          | 3 args    | `/0ab → b`    |
| `&`    | function    | 2 args    | `(&xy)y ab → b` |
| `,`    | display     | 1 arg     | `,x → x and displays x` |
| `!`    | eval        | 1 arg     | `!e → evaluate e` |
| `?`    | try         | 3 args    | `?teb → evaluate e time t with bits b` |
| `@`    | read bit    | 0 args    | `@ → 0 or 1` |
| `%`    | read exp    | 0 args    | `% → any s-expression` |
| `#`    | bits for    | 1 arg     | `#x → bit string for x` |
| `^`    | append      | 2 args    | `^‘(ab)(cd) → (abcd)` |
| `~`    | show        | 1 arg     | `~x → x and may show x` |
| `:`    | let         | 3 args    | `:xv e → (^&(x)e v)` |
| `^&`   | define      | 2 args    | `^&xv → x is v` |
| `” `   | literally   | 1 arg     | `”+ → +`      |
| `{ } ` | unary       | 1 arg     | `{3} → (111)` |
| `[ ] ` | comment     | 0         | [ignored]     |
| `()`   | empty       | 0         |         |
| `0 `   | false       | 0         |         |
| `1 `   | true        | 1         |         |
lisp.c

LISP Interpreter Run

[[[(Fx) = flatten x by removing all interior parentheses]]]
[Define F of x as follows: if x is empty then return empty, if x is an atom then join x to the empty list, otherwise split x into its head and tail, flatten each, and append the results.]
& (Fx) /=x() / .x*x() ^(F+x)(F-x)

F: (&(x)(/=x())()/.x*x() (^F+x)(F-x))

(F,F) [use F to flatten itself]

expression (F,F)
display (&(x)(/=x())()/.x*x() (^F+x)(F-x))
value (&x/=x/.x*x^F+x-F-x)

[[[(Gx) = size of x in unary]]]
[Let G of x be [if x is empty, then unary two, if x is an atom, then unary one, otherwise split x into its head and tail, size each, and add the results] in ...]
: (Gx) /=x()’{2} / .x’{1} ~ (G+x)(G-x)
[Let G of x be [...] in:]
(G,G) [apply G to itself]

expression (((’(&G)(G,G))))’(’(x)(/=x())’(11)’/.x’(1) ’)(~(G+x)(G-x)))))

display (’(x)(/=x())’(11)’/.x’(1)’(G+x)(G-x)))

value (111111111111111111111111111111111111111111111111111111111)

End of LISP Run

Elapsed time is 0 seconds.
DEFINITIONS

• An S-expression $x$ is elegant if no smaller S-expression has the same output. (Here “output” may be either its value or what it displays.)

• Let $x$ be an S-expression. The LISP complexity $H_L(x)$ of $x$ is the size in characters $|p|$ of the smallest S-expression $p$ whose value is $x$.

• Let $X$ be an infinite set of S-expressions. The LISP complexity $H_L(X)$ of the infinite set $X$ is the size in characters $|p|$ of the smallest S-expression $p$ that displays the elements of $X$.

• $[\text{U}(p) = \text{output of universal machine U}]
\quad \left[\text{given binary program p.} \right]
\& (\text{Up}) ++?0'!%p$

• Let $x$ be an S-expression. The complexity $H(x)$ of $x$ is the smallest possible value of $7 \times$ (the size in characters $|p|$ of an S-expression $p$ whose value is $x$ if it is given the binary data $d$) plus (the size in bits $|d|$ of the binary data $d$ given to $p$).

• Equivalently $H(x) \equiv H_U(x)$ is the size in bits $|p|$ of the smallest bit string $p$ such that $U(p) = x$.

• The halting probability $\Omega$ of $U$ is the limit as $t \to \infty$ of (the number of $t$-bit programs $p$ such that $U(p)$ halts within time $t$) divided by $2^t$.

• Let $X$ be an infinite set of S-expressions. The complexity $H(X)$ of the infinite set $X$ is the smallest possible value of $7 \times$ (the size in characters $|p|$ of an S-expression $p$ that displays the elements of $X$ if it is given the binary data $d$) plus (the size in bits $|d|$ of the binary data $d$ given to $p$).

• Equivalently $H(X) \equiv H_U(X)$ is the size in bits $|p|$ of the smallest bit string $p$ such that $X = \text{lim}_{t \to \infty} -?t'!%p$. 
RESULTS

- Lowcase variables $x, y, n$ are individual S-expressions.

  Uppercase variables $X, Y, T$ are infinite sets of S-expressions.

- $H_L(x, y) \leq H_L(x) + H_L(y) + 8$.

- If $x \in T \implies x$ is elegant, then
  \[ x \in T \implies |x| \leq H_L(T) + 378. \]

- If $(x, n) \in T \implies H_L(x) \geq n$, then
  \[ (x, n) \in T \implies n \leq H_L(T) + 381. \]

- $H(x, y) \leq H(x) + H(y) + 140$.

- Let $x$ be a string of $|x|$ bits.
  \[ H(x) \leq 2|x| + 469, \text{ and } H(x) \leq |x| + H(|x|) + 1148. \]

- Let $\Omega_n$ be the first $n$ bits of $\Omega$.
  \[ H(\Omega_n) > n - 4431. \]

- $H(X \cap Y) \leq H(X) + H(Y) + 4193$.

- $H(X \cup Y) \leq H(X) + H(Y) + 4193$.

- If $(x, n) \in T \implies H(x) \geq n$, then
  \[ (x, n) \in T \implies n \leq H(T) + 2359. \]

- $T$ cannot determine more than $H(T) + 7581$ bits of $\Omega$. 
