Large Numbers, Busy Beavers, Noncomputability and Incompleteness

Food For Thought
November 1, 2007

Sam Buss
Department of Mathematics
U.C. San Diego

PART I

Large Numbers, Busy Beavers, and Undecidability

The "name the larger integer" game.

In this two player game, each player has a blank 3×5 card and writes out a description of an integer. They reveal their descriptions, and the player with the larger integer wins.

Rules:

- There are restrictions on fonts, font size, etc. that limit how much can be written on a card.
- The description must unambiguously describe a unique integer.
- The player with the larger integer wins.

Joe and Anna try the game:

Joe's card:

$10^{10^{10^{10^{10}}}}$

Anna's card:

$1000^{1000^{1000^{1000^{1000}}}}$

WHO WINS?

Joe and Anna try the game:

Joe's card:

$10^{10^{10^{10^{10}}}}$

Anna's card:

$1000^{1000^{1000^{1000^{1000}}}}$

WHO WINS?

Joe wins. ✓

Joe and Anna try a second round:

Joe's card:

Let $f_0(x) = 10^x$.
Let $f_{k+1}(x) = f_k(f(x))$.

Then: $f_{100000}(100000)$.

Anna's card:

Joe and Anna try a second round:

Joe's card:

Let $f_0(x) = 10^x$.
Let $f_{k+1}(x) = f_k(f(x))$.

Then: $f_{100000}(100000)$.

Anna's card:

The value on Joe's card, plus one.

Joe and Anna try a second round:

Joe's card:

Let $f_0(x) = 10^x$.
Let $f_{k+1}(x) = f_k(f(x))$.

Then: $f_{100000}(100000)$.

Joe wins. ✓

Anna's card:

The value on Joe's card, plus one.

Judges rule that Anna cheated since her answer takes advantage of the space available on Joe's card, and thus she is not using only her own 3×5 card.

Joe and Anna try a third round:

Joe's card:

For all h, let $h^{(0)}(x) = x$
and $h^{k+1}(x) = h(h^{(k)}(x))$.
Let $f_0(x) = 10^x$.
Let $f_{k+1}(x) = f_k^{(x)}(x)$.

Then: $f_{100000}(100000)$.

Anna's card:

Joe and Anna try a third round:

Joe's card:

For all h, let $h^{(0)}(x) = x$ and $h^{k+1}(x) = h(h^{(k)}(x))$.
Let $f_0(x) = 10^x$.
Let $f_{k+1}(x) = f_k^{(x)}(x)$.

Then: $f_{100000}(100000)$.

Anna's card:

(The largest value that can be specified on a 3×5 card according to the rules of the game) $+ 1$.

Joe and Anna try a third round:

Joe's card:

For all h, let $h^{(0)}(x) = x$ and $h^{k+1}(x) = h(h^{(k)}(x))$.
Let $f_0(x) = 10^x$.
Let $f_{k+1}(x) = f_k^{(x)}(x)$.

Then: $f_{100000}(100000)$.

Anna's card:

(The largest value that can be specified on a 3×5 card according to the rules of the game) $+ 1$.

Anna argues that the font restrictions mean that there are only finitely many possible valid ways to fill out a 3×5 card, and thus the quantity in parentheses is well-specified.

Problem: Anna's card has revealed a paradoxical situation similar to Berry's paradox.

Berry's Paradox:
"The smallest postive integer not definable with under eleven words."

Clearly, Anna's description cannot be a valid entry under the rules of the game (since it is larger than any valid entry), but *why* is it not valid?

Problem: Anna's card has revealed a paradoxical situation similar to Berry's paradox.

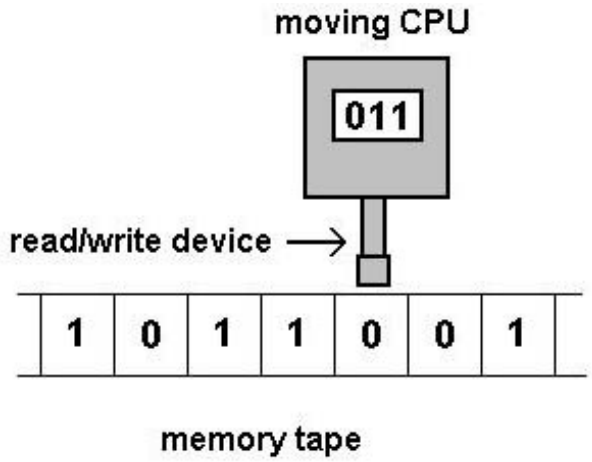
Berry's Paradox:
"The least integer not nameable in fewer than nineteen syllables."

Clearly, Anna's description cannot be a valid entry under the rules of the game (since it is larger than any valid entry), but *why* is it not valid?

The judges decide the problem is that the rules about what counts as a description are too open ended in allowing arbitrary natural language descriptions of integers. (Side remark: Tarksi's theorem that truth is undefinable.). So they decide to change the rules by using a Turing machine model for specifying integers.

[B. Russell, 1908 – extending a suggestion of G. Berry.]

Turing machines



A Turing machine is a simple computational model based on a finite state control working with an infinite tape memory. The Turing machine has a single tape head that can read and write only one tape square at a time.

A Turing machine is specified by a finite set of rules (5-tuples) that specify a function

$$f : Q \times \{0, 1\} \to Q \times \{0, 1\} \times \{R, L\}.$$

Here, $f(q, a) = (q', b, X)$ means "If in state q, reading symbol a, then overwrite a with b, move one square in the direction X (Left/Right), and go to state q'."

Church-Turing Thesis

Church-Turing Thesis: Any computable function can be computed by a Turing machine.

Church's thesis was originally stated by Church for the lambda calculus. Turing independently made the same point much more effectively for "Turing machines". Turing machines are as powerful as ordinary computers, except without finite limits on memory.

[A. Church, 1936; A. Turing, 1936.]

New rules for the "large integer" game

The judges decide that the new rules for the game are that the card must contain a Turing machine specified explicitly as a set of 5-tuples, over the alphabet {0,1}, with any number of states. (As before, the number of 5-tuples is limited by the size of the card, but fairly sophisticated programs can fit on a single card.)

The integer value specified by a Turing machine M is as follows: M is started with a tape containing all 0's.

- If M eventually halts, then the number of 1's left on the tape is the specified integer.

- If M never halts, then M does not specify any integer.

Anna's new card

(An encoding of) a Turing machine M that implements the following algorithm:

1. Enumerate all sets S of 5-tuples which fit on a 3×5 card and which code some Turing machine N_S.
2. For each such N_S, determine if it halts.
   If not, proceed to the next N_S.
3. Otherwise simulate N_S until it halts and count the number of 1's left on the tape when N_S halts.
   Remember the maximum number of 1's obtained so far.
4. Once all such N_S have been considered, write a string of 1's that is 1 longer than the maximum found, and halt.

Anna's new card

(An encoding of) a Turing machine M that implements the following algorithm:

1. Enumerate all sets S of 5-tuples which fit on a 3×5 card and which code some Turing machine N_S.
2. For each such N_S, **determine if it halts**.
   If not, proceed to the next N_S.
3. Otherwise simulate N_S until it halts and count the number of 1's left on the tape when N_S halts.
   Remember the maximum number of 1's obtained so far.
4. Once all such N_S have been considered, write a string of 1's that is 1 longer than the maximum found, and halt.

We have now shown:

Theorem (Turing, 1936). The halting problem is undecidable.

Small Turing machines can be very powerful

Define the Busy Beaver function by: [T. Radó, 1962]

$BB(n) =$ the largest integer output by halting n-state Turing machine blank, 2-symbol tape.

Then $BB(1) = 1$, $BB(2) = 4$, $BB(3) = 6$, $BB(4) = 13$, $BB(5) >= 498$, $BB(6) >= 1.2 \cdot 10^{865}$.

A $25,000 Wolfram prize was recently won by A. Smith [2007] who showed there is a 2-state, 3-symbol universal Turing machine (a machine which has the power to simulate *any* Turing machine when stared with appropriate initial tape contents).

PART II

Computability, Number Theory, and Incompleteness

Terminology

Def'n A function which can be computed by an algorithm, is called "recursive", or "computable".

Def'n A decision problem that can has an algorithmic solution is called "decidable" or "recursive".

Def'n A decision problem (such as the Halting problem) that can be expressed in the form

$$A(x) \Leftrightarrow \exists y \in \mathbb{N}(B(x,y)),$$

where $B(x,y)$ is decidable, is called "recursively enumerable (r.e)" or "computably enumerable (c.e.)".

Defn A *Diophantine equation* is an equation involving polynomials with integer coeffients, in which the variables are allowed to take on (non-negative) integer values only.

Example Pell's equation for fixed $n >= 0$ is $x^2 - ny^2 = 1$.

Thm [Matijasevich-Robinson-Davis-Putnam, 1959,1960,1970]. There is a many-one reduction from the Halting problem to Diophantine solvability. That is to say, there is a computable f such that for any Turing machine M, $f(M)$ is (the encoding of) a Diophantine equation $p_M(\vec{x}) = q_M(\vec{x})$, and such that M halts after starting with a blank tape iff $p_M(\vec{x}) = q_M(\vec{x})$ has a solution.

Corollary There is no algorithm the decides whether Diophantive equations have solutions.

This resolved Hilbert's 10th problem, by proving there is no algorithm for solving Diophantine equations.

Formal theory

Consider a formal theory, with a formal axiomatization and fully specified rules of inference. Examples include

1. Peano arithmetic (PA): A formal proof system for the non-negative integers, with axioms for $+$, $\cdot$ and induction.

2. Set theory (ZFC): A formal proof systems for sets, that is powerful enough to formalize *all*(!) of mathematics.

For the purposes of this talk: The more powerful theory you think of, the better.

At the least, such a theory T

- Is expressive enough to make statements about integers and (particular) polynomials and the solvability of any particular Diophantine equation.
- Is powerful enough to prove many basic properties of integers and polynomials.
- Has a mathematically defined notion of "valid proof", and there are algorithmic methods for recognizing, parsing, and manipulating valid proofs.
- The proofs in T can be encoded by strings of symbols over a finite alphabet.

For a given solvable Diophantine equation $p = q$, the theory T can prove

$$\exists \vec{x}(p(\vec{x}) = q(\vec{x})).$$

The proof consists of exhibiting one solution.

Defn T is said to be *complete* if for every sentence (statement) ϕ in the language of T, T proves either ϕ or $\neg\phi$.

If T is complete, then for any *unsolvable* Diophantine equation $p(\vec{x}) = q(\vec{x})$, T proves

$$\neg\exists \vec{x}(p(\vec{x}) = q(\vec{x})).$$

Gödel's 1st Incompleteness Theorem. [1931] There is no consistent complete, axiomatizable theory for the integers (over $+$, $\cdot$, ...).

To sketch the proof: Suppose T is complete. Then the following would be an algorithm for deciding solvability of Diophantine equations, contradicting the undecidability of the set of solvable Diophantine equations.

Input: A Diophantine equation $p = q$.

Algorithm:

- Enumerate all valid T-proofs, by enumerating all strings of symbols that may encode a T-proof.
- As the proofs are enumerated, check each to see if it a valid proof of $\exists\vec{x}(p(\vec{x}) = q(\vec{x}))$ or of $\neg\exists\vec{x}(p(\vec{x}) = q(\vec{x}))$.
- If so, halt and output "solvable" or "unsolvable", respectively.
- If not, continue to the next T-proof.

Since T is complete, this algorithm is guaranteed to halt.

Gödel's 2nd Incompleteness Theorem

One might object to the philosophical importance of the 1st Incompleteness Theorem by pointing out that its proof showed only that some rather contrived statements about unsolvability of Diophantine equations are not provable in T.

However, the 2nd Incompleteness Theorem resolves this issue by stating:

Gödel's 2nd Incompleteness Theorem. [1931] Let T be consistent and axiomatizable (and strong enough to prove sufficiently many properties of $+$ and $\cdot$ on $\mathbb{N}$). Then T does not prove the (encoding of the) statement

"T is consistent".

the end