Penrose's Philosophical Error

L.J.Landau

Mathematics Department
King's College London
in Concepts for Neural Networks, L.J.Landau and J.G.Taylor eds. (Springer 1997) ISBN 3-540-76163-2
This is a somewhat modified HTML version.

1. Can Computers Think?

The United States Department of Energy's Sandia National Laboratory and Intel Corporation have built a supercomputer that on December 17, 1996 reached the one trillion-operations-per-second mark.[1] It weighs 44 tons, has 573 gigabytes of memory and 2.25 trillion bytes of disk space. In the time it takes you to blink an eye, the computer will complete 40 billion calculations. It is only 25 years since Intel introduced the first microprocessor, which could carry out 60,000 operations per second. Who can say what breakthroughs will occur in the next 25 years or 250 years? Computers with tremendous power and speed will surely be developed. Will these computers be able to think? Will they be conscious? Is there today anyone who would say that no matter what breakthroughs occur in microelectronics, no matter what developments occur in computer programming, such as learning and random elements as in neural networks, no matter how powerful computers become, a computer will never think? One such person is Roger Penrose, who has written in his book *Shadows of the Mind* [14] that:

there must be more to human thinking than can ever be achieved by a computer. ... Consciousness, in its particular manifestation in the human quality of understanding, is doing something that mere computation cannot. I make clear that the term `computation' includes both `top-down' systems, which act according to specific well-understood algorithmic procedures, and `bottom-up' systems, which are more loosely programmed in ways that allow them to learn by experience.

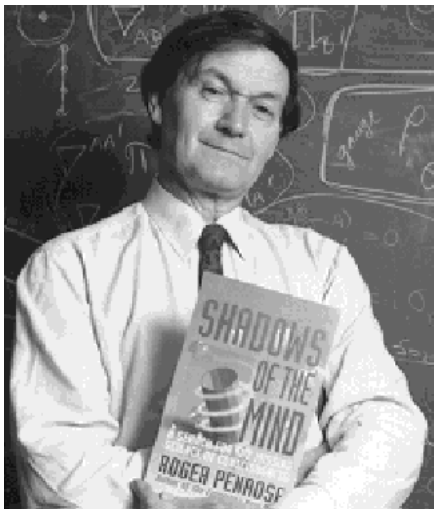

Figure1: Roger Penrose, 1996. © Steve Green. Reproduced with permission.

The question at issue is not a dualistic distinction between the conscious mind and the physical brain. Penrose accepts that the conscious mind arises as a functioning of the physical brain, but he does not believe this functioning could be simulated by a large number of silicon chips operating as they do in a modern computer.

John Lucas

In the 1930's Alan Turing (1912-1954) proved a theorem showing that certain types of idealized computers, operating according to fixed rules, do have limitations. Turing's theorem is of a general type first proved by Kurt Gödel (1906-1978) concerning limitations to formal mathematical systems. Penrose bases his argument that the mind is not computational on these limitations uncovered by Turing and Gödel. His use of Gödel's theorem is similar to that of J.R.Lucas [13]. Lucas's article begins like this:
Gödel's theorem seems to me to prove that Mechanism is false, that is, that minds cannot be explained as machines.
It is interesting, however, that Lucas doesn't rule out the construction of intelligent computers! At the end of his article, Lucas says:
When we increase the complexity of our machines there may, perhaps, be surprises in store for us. [Turing] draws a parallel with a fission pile. Below a certain ``critical'' size, nothing much happens: but above the critical size, the sparks begin to fly. So too, perhaps, with brains and machines.... This may be so. Complexity often does introduce qualitative differences. Although it sounds implausible, it might turn out that above a certain level of complexity, a machine ceased to be predictable, even in principle, and started doing things on its own account, or, to use a very revealing phrase, it might begin to have a mind of its own. It would begin to have a mind of its own when it was no longer entirely predictable and entirely docile, but was capable of

doing things which we recognized as intelligent.... But then it would cease to be a machine, within the meaning of the act.
So for Lucas, a machine cannot think *by definition of what a machine is*. Penrose certainly does not agree with Lucas's view here. On the contrary, Penrose claims that no matter how complex they become, computers will not have the capacity of human understanding.

Hilary Putnam

The results of Turing and Gödel belong to an area of mathematics known as mathematical logic. How do mathematical logicians react to Penrose's argument?[2] In his book review [16] of *Shadows of the Mind*, Hilary Putnam[3] states:
In 1961 John Lucas - an Oxford philosopher well known for espousing controversial views - published a paper in which he purported to show that a famous theorem of mathematical logic known as Gödel's Second Incompleteness Theorem implies that human intelligence cannot be simulated by a computer. Roger Penrose is perhaps the only well-known present-day thinker to be convinced by Lucas's argument.... The right moral to draw from Lucas's argument, according to Penrose, is that noncomputational processes do somehow go on in the brain, and we need a new kind of physics to account for them.

Shadows of the mind will be hailed as a ``controversial'' book, and it will no doubt sell very well even though it includes explanations of difficult concepts from quantum mechanics and computational science. And yet this reviewer regards its appearance as a sad episode in our current intellectual life. Roger Penrose ... is convinced by - and has produced this book as well as the earlier *The emperor's new mind* to defend - an argument that all experts in mathematical logic have long rejected as fallacious. The fact that the experts all reject Lucas's infamous argument counts for nothing in Penrose's eyes. He mistakenly believes that he has a philosophical disagreement with the logical community, when in fact this is a straightforward case of a mathematical fallacy.

Kevin Warwick

How do the robotics experts react to Penrose's argument? The leader of the cybernetics group at Reading University is Kevin Warwick, who has just published a book [28] subtitled *Why the New Race of Robots will Rule the World*, in which he states:
I believe that in the next ten to twenty years some machines will become more intelligent than humans.

We need to go to an appropriate level of modelling the functioning of a human brain in order to obtain a very good approximation.... Is it down, much further than the neuron level, to the atomic level? Should we look at quantum physics... as Penrose thinks, generating an atomic model of the brain's functioning that we do not yet have?... In reality I feel that we already do have sufficient basic modelling blocks.... What are those blocks? They are the neurons, the fundamental cells in the brain which cause it to operate as it does.

One major problem with consciousness is that it is a state which we ourselves feel that we are in, and through this we are aware of ourselves. It can, therefore, be extremely difficult, when thinking about ourselves, to realise that this is simply a result of a specific state of the neurons in our brain. These `private' thoughts and feelings we have, how can they possibly be simply a state of mind, realised by a state of the neurons? Well, that is exactly what they are.

Alan Turing

Figure2: Alan Turing, 1951. By courtesy of the National Portrait Gallery, London.

Since part of Penrose's argument is based on a theorem of Turing concerning a limitation on what can be achieved by computers, it is interesting to know Turing's thoughts on these issues. Turing wrote an article [24][4] in which he gives his opinion on whether or not a computer will ever be able to think. In the article Turing states his

belief that his own fundamental theorem concerning the limitations of computing machines is not an obstacle to the creation of intelligent computers.

Turing begins his article with the sentence:

I propose to consider the question, `Can machines think?'
The question in this form is, he believes, too vague and he restricts the machine to being a computer. He then replaces the question whether a computer can think like a human with the question whether a computer can *behave* like a human, and more specifically whether a computer can answer questions posed by an interrogator so that the interrogator cannot distinguish the replies of the computer from those of a human. (He calls this the imitation game. It is now known as the Turing test[5].)

Turing states clearly that he believes the computer will do well at imitating human behavior and gives 50 years as the time scale when computers will have become sufficiently powerful to begin doing this. He was writing in 1950 and so we are just coming to his predicted time period. (Remember the new Sandia-Intel supercomputer!) About his beliefs he says:

I believe further that no useful purpose is served by concealing these beliefs. The popular view that scientists proceed inexorably from well-established fact to well-established fact, never being influenced by any unproved conjecture, is quite mistaken. Provided it is made clear which are proved facts and which are conjectures, no harm can result. Conjectures are of great importance since they suggest useful lines of research.
To construct intelligent computers, Turing proposes to use the technique of computer learning:
Instead of trying to produce a programme to simulate the adult mind, why not rather try to produce one which simulates the child's? If this were then subjected to an appropriate course of education one would obtain the adult brain.... Our hope is that there is so little mechanism in the child-brain that something like it can be easily programmed. The amount of work in the education we can assume, as a first approximation, to be much the same as for the human child.
Turing considers, and rejects, contrary views to his own. He considers in particular the mathematical objection based on Turing's theorem, the issue we will be considering in this chapter. We give his discussion of this point in its entirety:
There are a number of results of mathematical logic which can be used to show that there are limitations to the powers of discrete-state machines. The best known of these results is known as Gödel's theorem, and shows that in any sufficiently powerful logical system statements can be formulated which can neither be proved nor disproved within the system, unless possibly the system itself is inconsistent. There are other, in some respects similar, results due to Church, Kleene, Rosser, and Turing.

The latter result is the most convenient to consider, since it refers directly to machines, whereas the others can only be used in a comparatively indirect argument: for instance if Gödel's theorem is to be used we need in addition to have some means of describing logical systems in terms of machines, and machines in terms of logical systems. The result in question refers to a type of machine which is essentially a digital computer with an infinite capacity. It states that there are certain things that such a machine cannot do. If it is rigged up to give answers to questions as in the imitation game, there will be some questions to which it will either give a wrong answer, or fail to give an answer at all however much time is allowed for a reply. There may, of course, be many such questions, and questions which cannot be answered by one machine may be satisfactorily answered by another. We are of course supposing for the present that the questions are of the kind to which an answer `Yes' or `No' is appropriate, rather than questions such as `What do you think of Picasso?' The questions that we know the machines must fail on are of this type, ``Consider the machine specified as follows... Will this machine ever answer `Yes' to any question?" The dots are to be replaced by a description of some machine in a standard form, which could be something like that used in 5. When the machine described bears a certain comparatively simple relation to the machine which is under interrogation, it can be shown that the answer is either wrong or not forthcoming. This is the mathematical result: it is argued that it proves a disability of machines to which the human intellect is not subject.

The short answer to this argument is that although it is established that there are limitations to the powers of any particular machine, it has only been stated, without any sort of proof, that no such limitations apply to the human intellect. But I do not think this view can be dismissed quite so lightly. Whenever one of these machines is asked the appropriate critical question, and gives a definite answer, we know that this answer must be wrong, and this gives us a certain feeling of superiority. Is this feeling illusory? It is no doubt quite genuine, but I do not think too much importance should be attached to it. We too often give wrong answers to questions ourselves to be justified in being very pleased at such evidence of the fallibility on the part of the machines. Further, our superiority can only be felt on such an occasion in relation to the one machine over which we have scored our petty triumph. There would be no question of triumphing simultaneously over *all* machines. In short, then, there might be men cleverer than any given machine, but then again there might be other machines cleverer again, and so on.

Turing ends his article looking towards the future:
We may hope that machines will eventually compete with men in all purely intellectual fields. But which are the best ones to start with? Even this is a difficult decision. Many people think that a very abstract activity, like the playing of chess[6],

would be best. It can also be maintained that it is best to provide the machine with the best sense organs that money can buy, and then teach it to understand and speak English. This process could follow the normal teaching of a child. Things would be pointed out and named, etc. Again I do not know what the right answer is, but I think both approaches should be tried.

We can only see a short distance ahead, but we can see plenty there that needs to be done.

Penrose's Error

We have seen that Penrose's argument, its basis and implications, is rejected by experts in the fields which it touches. So why expend effort studying it? Firstly, a discussion of Penrose's argument gives us the opportunity to consider the mathematical theorems on computation and formal systems which are interesting in their own right. Second, the conscious mind remains shrouded in mystery. If a precise mathematical theorem (and the bedrock of scientific precision is mathematics) can shed even a glimmer of understanding on the nature of consciousness, then it is worth careful study indeed. Perhaps you don't have the opportunity to invest the time and effort required to understand fully the point of Penrose's argument. As Alwyn Scott says [21] in answer to the question why he has such a stark difference of opinion with Penrose:

One answer to this question might be that I am unable - because of my intellectual limitations, interests in other matters, general laziness, or some combination of all three - to follow Penrose's arguments through their many details. Perhaps he is right, and I just don't get it.

To understand Penrose's mathematical argument against a computational mind, it will be necessary to disentangle all the various ingredients: Turing's theorem on limitations to computations, Gödel's theorem on limitations to formal systems, the formal system used to analyze a computation, computational models, mathematical truths, and mathematical beliefs. We shall uncover in section 6 a category-mistake in Penrose's reasoning, a confusion of the deduction of **H believes X** with the deduction of **X** itself. By supposing that the beliefs such as **X** are theorems, Penrose has made the consistency and correctness of his deductive system dependent on the consistency and correctness of H's beliefs, and this cannot be right. The beliefs and other thoughts of a mind cannot be theorems of the deductive system composing our theory of mind. The basic structure of the theory must be consistent whether or not the beliefs of a particular mind are consistent. Indeed, even in our everyday reasoning about other people's beliefs, those beliefs do not become part of our reasoning. And if those beliefs are contradictory, it does not follow that our reasoning about those beliefs is contradictory. We should be able, for example, to consistently and correctly deduce

that someone will perform a particular action as a result of the incorrect beliefs he holds.

But let us begin by discussing the basic mathematical results, unencumbered by possible applications to a theory of mind.

2. Solving Problems in Arithmetic

Suppose I give you a piece of paper with this written on it:
`27+96=?`
You might say, `That's easy; 123.' Or you might say `What do you want me to do with this piece of paper?', or `Que voulez-vous que je fasse de ce papier?' if you're French. The point is that before you can respond to something I do you have to understand what I'm doing. We would have had to establish a means of communication so you will understand that I'm asking a question and understand what the question is before you can answer it. All sorts of conventions have been established as you have grown up and studied at school so that the initial communication hurdle is not an obstacle. Suppose you understand that I'm asking a question but you're not sure what some of the symbols on the paper represent. You might ask me `What is that symbol `27`?' Perhaps I might show you a bowl with 27 marbles in it. But you might not know what aspect of what I showed you corresponded to the 27. So I might then show you a bowl with 27 oranges. In case you still do not understand I might show you a bowl with 27 mice. You might then get the idea that 27 is the common characteristic of all the things I showed you: It must be a bowl! No, I'm sure you would understand that 27 represents the quantity of marbles, or oranges. In fact to keep everything as concrete as possible and avoid unnecessary abstractions, no harm will result if we agree that we are always going to talk about marbles. So 27 means 27 marbles. Going back to what was written on that piece of paper, we will now suppose that you understand that I'm asking you a question about quantities of marbles. There are two quantities mentioned: 27 and 96. You might guess that the symbol + represents some *operation* you are to perform on the two quantities of marbles, but which operation? I'm sure you would say `That's easy, it means to add the two quantities of marbles together.' It's only easy because you were taught how to do addition and the symbol which represents it in school. We must start somewhere, with some things understood[7] (for example the meaning of 27 marbles and certain basic operations which could be carried out on quantities of marbles). Then the meaning of more and more complicated operations on marbles could be defined. For example, I might hand you a piece of paper with this written on it:

$$(27+96)\times(5+8^2) = ?$$

You might say, `That's easy; 8487.'

In order to solve problems in arithmetic (which for concreteness we will take to mean questions about quantities of marbles) we must first agree on a certain set of basic instructions, or operations on the marbles, from which a rule may be given to perform more complicated operations. The rule is a program to be executed by performing the basic instructions one after the other. We might ask questions about the result of executing a program starting from some initial collection of bowls with marbles in them. If the program represents addition then we are asking a question about addition. If the program represents multiplication then we are asking a multiplication question. And so on.

Why do we choose to formulate arithmetic in terms of bowls of marbles? Because in this way arithmetic corresponds to concrete actions on concrete objects (marbles, or coins, etc.). This is how arithmetic originally developed, as a concrete activity rather than an abstract mental construct. At this stage there is no logic, no deduction or proof, just concrete actions carried out according to instructions which define addition, multiplication, and other more complicated arithmetical procedures. At a later stage we may reason about these arithmetical procedures, using rules of deduction which seem to us to be sound.

The Basic Instructions

There are a number of ways of selecting basic instructions out of which all arithmetical operations can be performed. Turing took one approach and arrived at Turing machines. We'll use a different approach here. The basic instructions are simple operations on bowls of marbles (for concreteness). We start with a sufficient supply of bowls, each of which is labeled with a symbol 0,1,2,□. Each operation which we will define will require a particular number of bowls, but we won't specify an upper limit to the number of bowls so that arbitrary arithmetical procedures can be carried out. So we have bowls $R_0,R_1,R_2,$□ which are referred to as *registers*. The number r_k of marbles in the bowl (register) R_k is referred to as the *content* of the register. An arithmetical procedure or program consists of a finite sequence of instructions $I_0,I_1,I_2,$□,I_b. Only four types of instructions are used:

Zero instructions.

For each k = 0,1,2,□ there is a zero instruction Z(k). The meaning of the instruction Z(k) is to change the content r_k of the register R_k to 0, leaving all other registers unaltered. Thus, to carry out the instruction Z(k) simply empty the k^{th} bowl.

Successor instructions.

For each k = 0,1,2,□ there is a successor instruction S(k). The meaning of the instruction S(k) is to change the content r_k of the register R_k to r_k+1, leaving all

other registers unaltered. Thus to carry out the instruction S(k) simply add a marble to the k^{th} bowl.

Transfer instructions.

For each j = 0,1,2,□ and k = 0,1,2,□ there is a transfer instruction T(j,k). The meaning of the instruction T(j,k) is to replace the content r_k of the register R_k with the number r_j contained in R_j, leaving all other registers unaltered. Thus to carry out the instruction T(j,k) take a spare bowl and put into it a quantity of marbles equal to the quantity of marbles in the j^{th} bowl. Then replace the marbles in the k^{th} bowl with the marbles in the spare bowl.

Jump instructions.

The preceding three instructions are called arithmetical instructions, and they manipulate the quantity of marbles contained in the bowls. The final type of instruction is of a different sort and determines how you process the program. For each j = 0,1,2,□, k = 0,1,2,□, and *l* = 0,1,2,□ there is a jump instruction J(j,k,*l*). The meaning of the instruction J(j,k,*l*) is as follows[8]:

1. if $r_j = r_k$ proceed to the l^{th} instruction in the program
2. if $r_j \neq r_k$ proceed to the next instruction in the program

None of the contents of the registers is altered by a jump instruction.

That's all there is to it. The program $P = I_0,I_1,I_2,\square,I_b$ is executed by carrying out each instruction, one after the other, except in response to a jump instruction which may cause you to jump to another instruction in the program. The execution of the program P ends when the required next instruction does not exist. For example, having executed the last instruction I_b you proceed to the next instruction I_{b+1}, but there is no such instruction. Alternatively, a jump instruction J(j,k,b+1) may indicate a jump to the instruction I_{b+1}, which does not exist.

Note: A program, containing only finitely many instructions, can refer to only finitely many registers[9], called the *working registers*. The contents of the remaining registers will not be changed during the execution of the program, and the values in these registers will not affect the values in the working registers.

The setup we have been considering, consisting of registers (bowls) R_k containing numbers (of marbles) r_k and programs of instructions of the previously discussed type, is called an *Unlimited Register Machine* or URM for short.[10] Using this approach you can compute any function which can be computed in any other way. In particular, you can compute the same functions as Turing machines compute.

A program P has a finite specification by the instructions $I_0,\square,I_b$. You can however execute the program with infinitely many possible initializations of the registers. The

program thus represents a *uniform* way for you to deal with the infinitely many cases you may be presented with.

Comment

Let's make the remark that a program P has no `purpose'and does not `ascertain' anything. It's just a list of instructions which you carry out if you decide to execute that program. It is we who give the program a purpose according to the use we put the result we get when we complete the program.

A Sample Program

Let's show how to write a program P to add any two numbers (quantities of marbles).

$$P = J(2,1,4),S(0),S(2),J(0,0,0),Z(1),Z(2) \qquad (1)$$

The registers are initially all set to 0 except R_0 which contains m (marbles) and R_1 which contains n (marbles). The numbers r_0 and r_2 will be increased by 1 until $r_1 = r_2$, whereupon registers R_1 and R_2 will be emptied (set to 0) and the program completed. (Note that the jump instruction J(0,0,0) has the effect of causing you to jump to the first (0^{th}) instruction, since r_0 is always equal to r_0.) You will wind up with m+n (marbles) in the register R_0, and 0 (marbles) in all other registers. The way the computation goes adding 3 and 2 is shown in Figure 3. At each step of the program, the state of the computation is described by the configuration of the registers[11] and the next instruction to be carried out. The computation halts when the next instruction is I_6, since there is no such instruction.

R_0	R_1	R_2	R_3	next instruction
3	2	0	0	0
3	2	0	0	1
4	2	0	0	2
4	2	1	0	3
4	2	1	0	0
4	2	1	0	1
5	2	1	0	2
5	2	2	0	3
5	2	2	0	0
5	2	2	0	4
5	0	2	0	5
5	0	0	0	6

Figure3: The computation 3+2.

Another Sample Program

The only *comparison* of two numbers n and m which we use as part of our basic instructions is equality[12], n = m: whether or not n is the *same* as m. Another useful comparison is n ≤ m: whether or not n is *less than or equal to* m. Here is a program to determine if n ≤ m. Initialize the registers to r_0 = n, r_1 = m, and all other registers initialized to 0. When the program is completed, the register R_0 will contain 1 if n is less than or equal to m, or 0 if n is greater than m. All other registers will contain 0:

T(0,2),T(1,3),J(1,2,7),J(0,3,10),S(2),S(3),J(0,0,2), (2)

Z(0),S(0),J(0,0,11),Z(0),Z(1),Z(2),Z(3) (3)

The program works by first transferring a copy of n and m into registers R_2 and R_3. Then the contents of those registers are successively increased until either n is equal to some successor of m, in which case n > m, or m is equal to some successor of n, in which case n < m.[13] The register R_0 is then set appropriately to 0 or 1, and finally all other registers are set to 0.

Numbering All Programs

Programs are specified by giving their lists of instructions. When reasoning about programs and what can be achieved by using them, it is useful to specify the programs in another way, by giving a rule to *encode* all the details of the program into a single number. Thus, if I give you a list of instructions making up a program, you will be able to compute a single number which will represent that program. Conversely, if I give you a single number which represents a program, you will be able to *decode* the number and find the program to which it refers. There are many ways this could be done, but whatever way I choose I must *explain* it to you, so you will understand which program I am referring to if I say `Execute program number 7.' Here is one way to number the programs: in order to assign numbers to programs, we first assign numbers to instructions. The first step is to assign numbers to the instructions of each of the four sorts:

Zero instructions.

Assign numbers to the zero instructions by associating the instruction Z(k) with the number k.

Successor instructions.

Assign numbers to the successor instructions by associating the instruction S(k) with the number k.

Transfer instructions.

In order to assign numbers to the transfer instructions we must be able to encode *pairs* of numbers (j,k) by a single number. We associate[14] the pair of numbers (j,k) with the number $n = f(j,k) = 2^j(2k+1)-1$. Conversely, we can find j and k from n by $j = f_1(n)$, $k = f_2(n)$ where $f_1(n)$ is the largest power j such that 2^j divides n+1, and:

$$f_2(n) = \frac{1}{2}\left\lfloor \frac{n+1}{2^{f_1(n)}} - 1 \right\rfloor$$

Then the transfer instruction T(j,k) is associated with the number n = f(j,k).

Jump instructions.

In order to assign numbers to the jump instructions we must be able to encode *triples* of numbers (j,k,*l*) by a single number. In fact, once we have encoded pairs of numbers, we can encode triples, quadruples, etc. We associate to the triple of numbers (j,k,*l*) the number m = g(j,k,*l*) = f(f(j,k),*l*). From m we can find j,k,*l* by *l* = $f_2(m)$, j = $f_1(f_1(m))$, and k = $f_2(f_1(m))$.

Then the jump instruction J(j,k,*l*) is associated with the number m = g(j,k,*l*).

The next step is to combine the above encoding into a single encoding of all instructions:

Zero instructions.

The zero instruction Z(k) is associated with the number 4k.

Successor instructions.

The successor instruction S(k) is associated with the number 4k+1.

Transfer instructions.

The transfer instruction T(j,k) is associated with the number 4f(j,k)+2.

Jump instructions.

The jump instruction J(j,k,*l*) is associated with the number 4g(j,k,*l*)+3.

In this way given any instruction I we know how to find the corresponding number n, and conversely, given any number n we know how to find the corresponding instruction I.

In the final step we assign numbers to the programs P = $I_0,I_1,\ldots,I_b$. Since we have assigned numbers to the individual instructions, we can associate with P the sequence of numbers $m_0,m_1,\ldots,m_b$, where m_j is the number assigned to the instruction I_j. Thus we must encode all such sequences of numbers into a single number. One way to do this is to associate to the numbers $m_0,m_1,\ldots,m_b$ the number:

$$h(m_0,\ldots,m_b) = {}_2^{2^{m_0}+2^{m_0+m_1+1}+2^{m_0+m_1+m_2+}} + \ldots+2^{m_0+m_1+m_2+\ldots+m_b+b}-1 \qquad (4)$$

This works because every number ≥ 1 may be uniquely expressed as a sum of powers of 2 (which just gives the binary expression for the number). Hence if m_j is the number associated with the instruction I_j in the program P, then P is associated with the number $h(m_0,\ldots,m_b)$.

We may explain formula (4) this way. Given the program P and the corresponding Gödel numbers $m_0,m_1,\square,m_b$ we compute h using the following prescription:

$$h(m_0,m_1,\square,m_b)_{binary} = 1\ \underbrace{0\square0}_{m_b}\ 1\square1\ \underbrace{0\square0}_{m_1}\ 1\ \underbrace{0\square0}_{m_0} - 1 \qquad (5)$$

Here h is expressed in binary form. Conversely, given $h(m_0,m_1,\square,m_b)$ we express $h(m_0,m_1,\square,m_b)+1$ in binary form and read off $m_0,m_1,\square,m_b$ from the number of zeros between successive 1's.

Computing Gödel Numbers

The numbers associated with instructions and programs are called Gödel numbers, since Gödel first used the coding of non-numerical quantities by numbers. Today such coding is used all the time. For example, a chess playing computer is running a program which codes the chess pieces, their positions on the board, and their allowed moves into numerical values and relations which can then be manipulated as the game progresses.

Let's find the programs corresponding to some Gödel numbers:

1. The program with Gödel number 0: Since $0 = 2^0-1$ we see from (2) that $b = 0$ and $m_0 = 0$. Thus the program has only one instruction and that instruction has number 0. The zero instruction Z(k) has the number 4k, hence the instruction Z(0) has number 0. In conclusion, the program with Gödel number 0 is P_0 = Z(0). What would you accomplish by executing the program P_0? Simply empty the bowl R_0.
2. The program with Gödel number 1: Since $1 = 2^1-1$ we see from (2) that $b = 0$ and $m_0 = 1$. The successor instruction S(k) has the number 4k+1, hence the instruction S(0) has number 1. In conclusion, the program with Gödel number 1 is P_1 = S(0). What would you accomplish by executing the program P_1? Simply add a marble to bowl R_0.
3. The program with Gödel number 2: Since $2 = 2^0+2^1-1$ we see from (2) that $b = 1$ and $m_0 = 0,m_1 = 0$. Then P_2 = Z(0),Z(0). You will obtain precisely the same result by executing P_2 or P_0 (but you empty bowl R_0 twice!).
4. The program with Gödel number 7: Since $7 = 2^3-1$ we see that $b = 0$ and $m_0 = 3$. Thus the program consists of only one instruction and that instruction has number 3. This is the jump instruction J(0,0,0). So P_7 = J(0,0,0). If you execute this program you will never halt. You will keep repeating the first instruction

(which is to perform the first instruction) but you will make no change to the contents of the registers!

Let's find the Gödel number of the program P (1) which adds any two numbers. First write the Gödel number of each instruction in the program:

Zero instructions.

Since the Gödel number of Z(k) is 4k, the Gödel number of Z(1) is 4 and the Gödel number of Z(2) is 8.

Successor instructions.

Since the Gödel number of S(k) is 4k+1, the Gödel number of S(0) is 1 and the Gödel number of S(2) is 9.

Jump instructions.

Since the Gödel number of J(j,k,*l*) is

$$4g(j,k,l)+3 = 4f(f(j,k),l)+3 \ ,$$

we compute the Gödel number of J(0,0,0) as 4f(f(0,0),0)+3. Since f(0,0) = $2^0(2\cdot 0+1)-1 = 0$, we get 4f(0,0)+3 = 4·0+3 = 3. So the Gödel number of J(0,0,0) is 3. Similarly, the Gödel number of J(2,1,4) is 4g(2,1,4)+3 = 4f(f(2,1),4)+3. Now f(2,1) = 11, so g(2,1,4) = f(f(2,1),4) = $2^{11}\cdot 9-1 = 2048\cdot 9-1 = 18431$. So the Gödel number of J(2,1,4) is 73727.

The sequence of numbers corresponding to the instructions in P is thus:

$$73727,1,9,3,4,8$$

Hence the Gödel number of the program P is:

$$h(73727,1,9,3) = 2^{73727}+2^{73729}+2^{73739}+2^{73743}+2^{73748}+2^{73757}-1$$

A very large number! Gödel numbers can be extremely large, but we use them as a theoretical device rather than a practical one.

The Marrakech Game

Let's pause for a moment and consider an analog of this process of Gödel numbering, where non-numerical quantities are coded into numbers. Here is an amusing way to code the game tic-tac-toe (noughts and crosses) into numbers. First assign the numbers from 1 to 9 to squares of the board as shown in Figure 4.

8 1 6
3 5 7
4 9 2

Figure4: Assigning numbers to the squares of a tic-tac-toe board.

This is a magic square, where the numbers add to 15 along any row, column, or diagonal.

The first player begins the game by putting an X in one of the squares on the board. This is the same as choosing a number from 1 to 9. The second player now puts an O in one of the remaining squares. This is the same as choosing one of the remaining numbers. Play proceeds like this until one player has marked all the squares in a row, column, or diagonal. This is the same as one player having three numbers which add to 15.

We have thus reformulated the game of tic-tac-toe as an equivalent game involving a purely numerical procedure of choosing numbers from 1 to 9, the game being won when a player gets three numbers which add to 15. It is described in Karl Fulves' *Self-Working Number Magic* [9] where it is called the *marrakech game*.

Of course, as numbers do not carry any meaning beyond their place in the succession of numbers, the marrakech game no longer has the same `meaning' as the original game. (The marrakech game is harder to play than tic-tac-toe because the geometrical symmetries of the tic-tac-toe board and the geometry of the game play are not present in the marrakech game.)

3. The Barber Paradox

Imagine a small town with one barber who shaves all those and only those who do not shave themselves. Can there be such a barber? No, there can't. To see this consider whether or not the barber shaves himself. If he doesn't shave himself then he must shave himself, and if he shaves himself then he must not shave himself. This paradox can be translated into mathematics and leads to a version of Gödel's incompleteness theorem. A nice discussion of the application of this and other paradoxes to obtain limitations on what can be achieved using programs is given by Gregory Chaitin [5].

We have seen that all programs can be numbered: $P_0,P_1,P_2,\square$. Let's say that a number m is produced by a program P_n if there is an initial value r_0 of register R_0 such that when all other registers are initialized to 0 ($r_k = 0$ for $k \neq 0$) and the program executed, the program will be completed with m in register R_0 (and any numbers in the other registers). We will be interested in the family of statements S_n of the form `n is not produced by P_n'. Which statements S_n can be shown to be true? For some values of n, by reasoning about the sequence of instructions in P_n you might find a way of demonstrating that S_n is true. For other values of n you might be able to demonstrate that S_n is false. For still other values of n you might have no idea whether or not S_n is true (or even whether or not it is meaningful to ask if it is true). For example:

1. The program $P_0 = Z(0)$. Hence no matter what the value r_0 initially in register R_0 this program will be completed after one step and the content of register

R_0 will be 0. Thus 0 is the only number produced by P_0. Since 0 is produced by P_0 the statement S_0 is false.

2. The program $P_1 = S(0)$. Hence if r_0 is initially in register R_0 this program will be completed after one step and the content of register R_0 will be r_0+1. Thus all the numbers 1,2,3,□ will be produced by P_1. Since 1 is produced by P_1 the statement S_1 is false.
3. The program $P_2 = Z(0),Z(0)$. This program produces only the number 0. Since 2 is not produced by P_2, the statement S_2 is true.
4. Let a be the number $2^{73727}+2^{73729}+2^{73739}+2^{73743}+2^{73748}+2^{73757}-1$. Then P_a is the program (1) which adds any two numbers. Hence if we set r_0 to any initial value and all other registers to 0 (in particular $r_1 = 0$) then the result of executing P_a will be $r_0+r_1 = r_0$ in register R_0. Thus P_a produces all numbers. Since a is produced by P_a, the statement S_a is false.

Perhaps if we rummage around among our programs $P_0,P_1,$□, we might come across one which just happens to produce precisely the numbers n such that S_n is true! Is this possible? Let's call a program *sound* if all the numbers n that it produces are numbers of true statements S_n. So might there be a sound program P_l which actually produces the numbers of *all* the true statements? No, *there is no such program*, and the reason could be thought of as coming out of the barber paradox.

To see that no such P_l exists, suppose P_l produces the number l. Then, given that P_l is sound, it follows that S_l is true, i.e. P_l does not produce l. This contradiction means that *a sound program cannot produce its own Gödel number*. Thus, given that P_l is sound, it follows that P_l does not produce l. Therefore S_l is *true*. Hence the sound program P_l cannot produce the number l of the true statement S_l. Hence we have our first main result:

> There is no sound program which can produce the numbers n of *all* true statements S_n. (6)

Our reasoning showed more, giving a particular true statement S_l whose number is not produced by P_l. This is our second main result:

> If the program P_l is sound,then the statement S_l is true and l is not produced by P_l.In this way, **we can go beyond any program which is known to be sound,and obtain a true statement not produced by the program**. (7)

But it is not just the one statement S_l which is missed by P_l, but infinitely many statements. This can be seen by observing that given the sound program P_l, we can find a program P_l' which produces precisely the numbers produced by P_l together with the number l. We illustrate how to do this in the flow diagram, Figure 5:

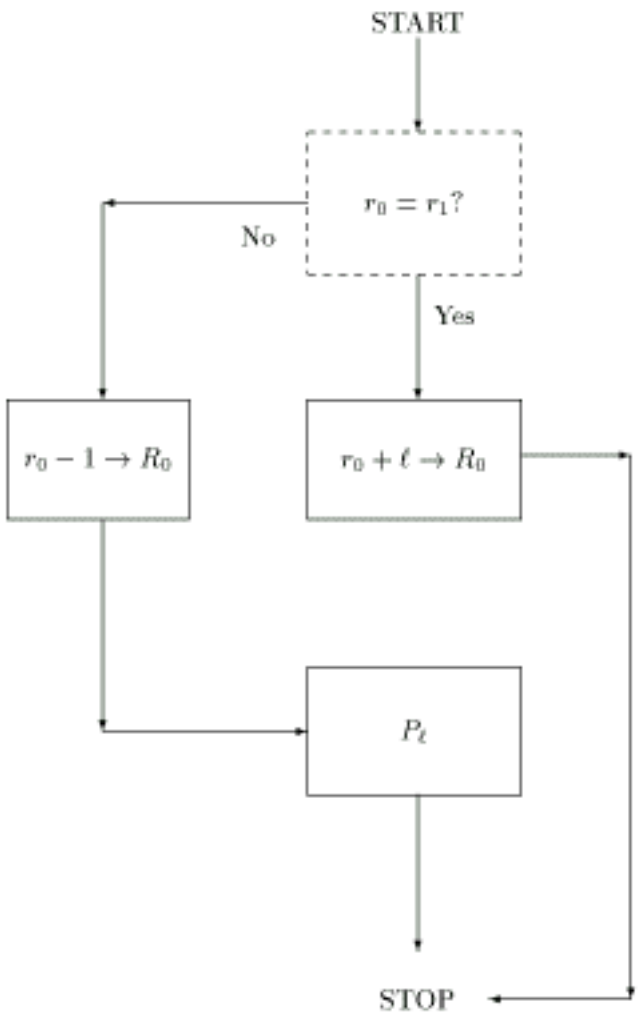

Figure5: Flow diagram for the program P_l' .

The registers are initialized to 0 except R_0 which is set to $r_0 = m$. First test if $m = 0$ by comparing r_0 and r_1 (since $r_1 = 0$). If $m = 0$, add[15] l to r_0 and halt. If m □ 0, subtract[16] 1 from r_0 and then execute the program P_l.

Consequently the program P_l' is also sound and furthermore produces the number l, a number not produced by P_l. But the previous argument applied to P_l' shows that S_l' is true but not produced by P_l' (and hence also not by P_l). This is our third main result:

> For each sound program P_l, there is anotherprogram P_l' which is also sound. The statements produced astrue by P_l' are the statements produced as true by $P_l$together with S_l,which was not produced by P_l. The true statement S_l' is not produced by P_l' (nor by P_l). (8)

We may now apply the preceding argument to $P_{l'}$ to obtain another program $P_{l''}$ which is again sound and which gives all the true statements given by $P_{l'}$ together with $S_{l'}$. We may repeat this process as often as desired, leading to our fourth main result:

For each sound program P_l, there isa sequence P_{l1},P_{l2},□ of programs, each of which is soundand each P_{lj+1} outputs the same as the precedingprogram P_{lj} together with l_j. For each P_{lj} there remaininfinitely many true statements not obtained by P_{lj}. (9)

In all of this discussion it is absolutely essential that the initial program P_l is sound. Is there a method of obtaining *all* the sound programs P_l? That is, is there a program P_k which produces precisely the numbers of all sound programs? No, there isn't. Indeed, if P_k is such a program then if P_k produces l then it follows from the preceding discussion that S_l is true. Thus P_k is sound. It follows that P_k doesn't produce k and hence P_k cannot produce the numbers of all sound programs. Hence no such program exists. So we have obtained our fifth and final main result:

There is no program P_k which produces precisely the numbers of all sound programs. (10)

Comment

Given a program P, we can execute the instructions and determine one by one the numbers produced by P. Note that the numbers produced by P depend only on the instructions listed by P. If we choose some Gödel numbering of the programs then the program P will be assigned some number, say 7, so P is P_7. Now if P doesn't produce the number 7 then S_7 is true. But suppose we choose some other Gödel numbering. Perhaps now some other program P′ is assigned the number 7, so P′ is now P_7. If P′ produces the number 7 then S_7 is false. So whether or not the statement S_7 is true depends on our choice of Gödel numbering. Consequently, whether or not a program is sound also depends on our choice of Gödel numbering. This shows again that programs do not have an inherent `purpose'. They just produce numbers. It is we who interpret and utilize the numbers as we wish.

4. Gödel's Theorem

Figure6: Kurt Gödel, 1939. Reproduced by permission of the Institute for Advanced Study, Princeton, USA

In section 2 we defined the basic instructions and the programs composed of these instructions. I can explain to you very simply what you are to do when following each instruction and you may then execute the program. It is all very concrete and involves moving marbles in a specific way. A chimpanzee could be trained to execute some programs. A mechanical contraption could be built which would execute the programs, depending on how various levers were set. A present-day computer could be set up to execute any program if you type the instructions on the keyboard. There is no `logic' or reasoning which forms part of the process of executing a program. On the other hand, in section 3 we have been *reasoning about programs*. This involves something different from the programs themselves. What constitutes valid reasoning? Can we be absolutely precise as to what shall consistute an acceptable proof? In what language shall the reasoning take place?[17] It was with the objective in mind of setting down a precise, consistent framework in which to conduct mathematical reasoning that formal mathematical systems were developed. A formal mathematical system is formulated with a specified language, which consists of an alphabet of symbols, together with rules for writing down formulas. In addition, there are rules[18] which determine when a finite sequence of formulas constitutes a proof, the final formula of the proof being the theorem which is proved. The most characteristic property of such systems is the mechanical nature of proof-checking, a property emphasized by David Hilbert (1862-1943). Let's decide on a method[19]2 to encode formulas and finite sequences of formulas into (Gödel) numbers. Then the mechanical nature of proof-checking means that there is a program P_l which will produce precisely the Gödel numbers of proofs, and a program P_m which will produce the Gödel numbers of the

last formula in each proof. Every proof will eventually be produced by P_l and hence every theorem will eventually be produced by P_m.

When we use a formal mathematical system we have a particular interpretation in mind for the symbols and formulas. However, someone else may use the same formal system with a different interpretation.[20] The formal system gives rules for writing symbols on paper. It has no intrinsic meaning and does not come with an interpretation. For this reason, Bertrand Russell once said (quoted in chapter XI of [12]):

Mathematics may be defined as the subject in which we never know what we are talking about nor whether what we are saying is true.
and Henri Poincarè said (quoted in chapter XII of [12]):
Mathematics is the art of giving the same name to different things.
Once an interpretation is settled upon we can inquire as to whether or not a particular formula is *true*. You may use any interpretation you wish provided all the axioms of the formal system are true in your interpretation. The formal system is said to be *sound* if its theorems are true *no matter which interpretation you are using*.[21]

So suppose we have settled on a particular formal system to make deductions about the programs we have been considering. We interpret[22] the formulas as statements about programs. The statement S_n, discussed in the preceding section, will be expressed by a formula ϕ_n. Then if S_n is true we will say that the formula ϕ_n is true, and if S_n is false we will say that the formula ϕ_n is false.

From the program P_m which produces the Gödel numbers of all the theorems of the formal system, we can write down a program P_j which produces the number n if and only if ϕ_n is a theorem. Here's a prescription to do this. For given input, execute P_m. If P_m halts and produces the number k, then we know k is the Gödel number of a theorem. To find out if k corresponds to one of the formulas ϕ_n, we need to use the program P_M which, given input n, produces the Gödel number of ϕ_n. We execute P_M repeatedly with input 0, then 1, then 2, etc. comparing the number produced with k. If k is not the Gödel number of some ϕ_n the program will not halt. If k is the Gödel number of ϕ_n then the program produces n as output. This prescription in words must be translated into a program P_j. This is easy to do once we have the programs P_m and P_M. A flow diagram for P_j is shown[23]7 in Figure 7. Summarizing, from the program P_m which produces the Gödel numbers of theorems of our formal system, and the program P_M which from input n produces the Gödel number of ϕ_n, we can write down a program P_j which produces the numbers n such that ϕ_n is a theorem.

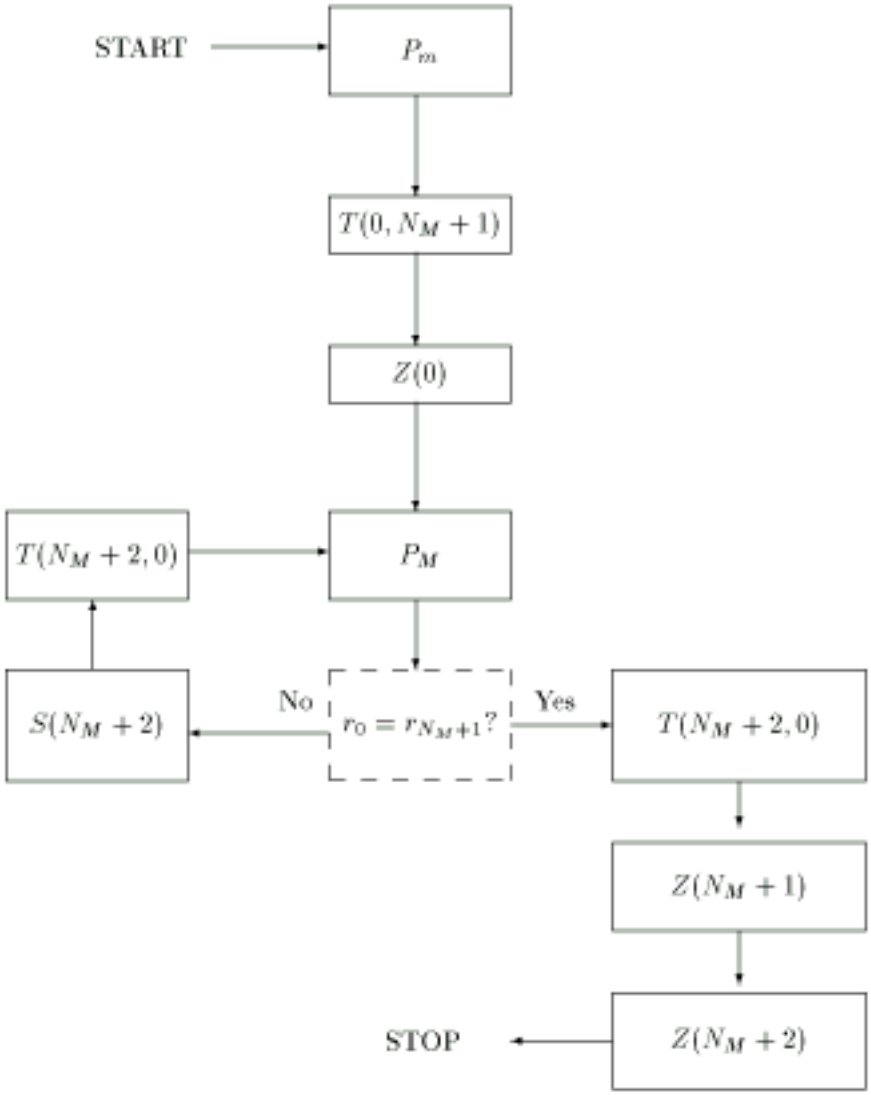

Figure7: Flow diagram for the program P_j.

Let us now suppose that our formal system is sound, i.e. the theorems are true. Thus P_m will produce only Gödel numbers of true formulas, and hence P_j will produce only numbers n such that S_n is true. Thus P_j is sound. It follows by (5) that the number j is not produced by P_j and S_j is true. Thus ϕ_j is true and not produced by P_m, i.e. ϕ_j is true but is not a theorem of the formal system. This conclusion holds for any sound formal mathematical system with a language sufficiently broad to express the statements S_n. We thus have the following version of Gödel's theorem:

For any sufficiently broad sound formal mathematical system, we canexplicitly exhibit a true formula ϕ which cannot be proved by the (11)

formalsystem. Furthermore, the negation of ϕ is falseand so also cannot be proved.

Recall that the formula which we have exhibited is ϕ_j, which expresses the statement S_j: `j is not produced by $P_j$'. But by the way $P_j$ was constructed, this is equivalent to the statement `ϕ_j is not a theorem of F'. Thus the formula ϕ_j says of itself that it is not a theorem of F. And this is true!

How to Understand Gödel's Theorem

The version of Gödel's theorem (9) which we have discussed above says that the sort of formal system F (supposed sound) which mathematicians use to derive truths (say, about arithmetic) will not be able to deduce *all* the true statements. Furthermore we can explicitly exhibit a true formula ϕ which cannot be proved in the formal system F. How are we to understand this limitation to formal systems? There are a number of viewpoints which can lead to insights into the significance of Gödel's theorem and even to new theorems concerning formal systems. Here are some ideas to think about concerning Gödel's theorem:

1. We can explicitly exhibit a formula ϕ_j which is not a theorem of the formal system but is nevertheless true *in our interpretation*. However *in another interpretation* ϕ_j will be false.[24]

   An interpretation gives *meaning* to the formal system and this meaning leads us to truths not derivable from the formal system itself. This must be expected if a formal system can be given *different* interpretations, and so different meanings and truths.

2. Gödel's theorem says that we cannot deduce as theorems *all* true formulas, from the limited set of true formulas which constitute the axioms. The axioms constitute a set of truths of limited complexity and we cannot expect to derive truths of unlimited complexity from them. Chaitin (see [5]) puts it this way:

   > The complexity of the formal system □ is a measure of the amount of information the system contains, and hence the amount of information that can be derived from it. □ Gödel's theorem does not appear to give cause for depression. Instead it seems simply to suggest that in order to progress, mathematicians, like investigators in other sciences, must search for new axioms.

   > If one has ten pounds of axioms and a twenty pound theorem, then that theorem cannot be derived from those axioms.

3. Using a formal system to deduce theorems (which are truths if the formal system is sound), you are acting as an *oracle* producing statements. Gödel's theorem describes a limitation which applies not only to computational oracles (programs or formal systems) but to any oracle, regardless of its nature. *An oracle which is known to be sound can never tell us everything we can ascertain. Thus the Gödel phenomenon cannot be overcome by searching for a non-computational oracle.* We will show how this limitation arises by analogy with our previous discussion.

   An oracle in the form of a `black box', the interior of which we do not know, produces statements. The oracle is sound if all the statements are true. Now define[25] a device D which does the following: if the oracle produces the statement `D does not produce anything`, then D produces the number 0 in response. If the oracle produces any other statement, D does not produce anything in response.

   We may easily deduce that *if the oracle is sound*, then it will not produce the statement `D does not produce anything`, and furthermore it is true that D does not produce anything. Thus if we know the oracle is sound, then we know a truth not produced by the oracle.

5. Penrose

We will rephrase Penrose's argument using the particular mathematical results we have developed in sections 3 and 4. We will try to adhere to the meaning of Penrose's argument, and thus follow the moral of the following story related in section 2.10 of *Shadows of the Mind*:

I am reminded of a story concerning the great American physicist Richard Feynman. Apparently Feynman was explaining some idea to a student, but mis-stated it. When the student expressed puzzlement, Feynman replied: `Don't listen to what I say; listen to what I *mean*!'

In Chapter 2, *The Gödelian case*, and Chapter 3, *The case for non-computability in mathematical thought*, Penrose argues that the human mind cannot be simulated by a computer:

I shall shortly be giving (in Chapters 2 and 3) some very strong reasons for believing that effects of (certain kinds of) understanding cannot be properly simulated in any kind of computational terms... Thus, the human faculty of being able to `understand' is something that must be achieved by some non-computational activity of the brain or mind... The term `non-computational' here refers to something beyond any kind of effective simulation by means of any computer based on the logical principles that underlie all the electronic or mechanical calculating devices of today.

Penrose puts forward his main argument in section 2.5 of *Shadows of the Mind*, and then deals with various counter-arguments in Chapters 2 and 3:

The argument I shall present in the next chapter (section 2.5) provides what I believe to be a very clear-cut argument for a non-computational ingredient in our conscious thinking... In due course (in Chapters 2 and 3), I shall be addressing, in detail, *all* the different counter-arguments that have come to my attention.

The Gödelian Case

Consider again the problem of determining the true statements S_n, that is, the numbers n such that the n^{th} program does not produce n. According to the discussion in section 3 there is no sound program which will produce the numbers n of all the true statements S_n. *Can the human intellect succeed in producing all the true statements S_n?* There is no evidence of this, so how good is the human intellect at finding the true statements S_n? Say that a program P_l *encapsulates human understanding* if every number n of true statements S_n that human mathematicians can produce is also produced by P_l.

Question: Can human mathematical understanding concerning the statements S_n be encapsulated in a sound program?

According to (5) if the l^{th} program is sound, then S_l is true and l is not produced by P_l. Does this mean that the answer to the question is no: human mathematical understanding concerning the statements S_n cannot be encapsulated in a program? Not quite! In order to go beyond the program P_l we must *know* that P_l is sound. If we know a program is sound call it *knowably sound*. Then we can give an answer to the question as follows:

Answer: Human mathematical understanding concerning the statementsS_n cannot be encapsulated in a knowably sound program. (12)

This answer is given[26] by Penrose in his statement G in section 2.5 of *Shadows of the Mind*. This is basically all there is in section 2.5. Penrose would probably have preferred the answer:

Penrose's Preferred Answer: Human mathematical understanding concerning the statements S_n cannot be encapsulated in a sound program.

but this has not been shown. Here is what Putnam says [16] about Penrose's argument so far:

What Penrose has shown is quite compatible with the claim that a computer program could in principle successfully simulate our mathematical capacities. The possibility exists that each of the rules that a human mathematician explicitly relies on, or can be rationally persuaded to rely on, can be known to be sound and that the program generates all and only these rules but that the program itself cannot be rendered sufficiently ``perspicuous" for us to know that that is what it does. Actual programs sometimes consist of thousands of lines of code, and it can happen that by the time a

program has been tinkered with and debugged no one is really able to explain exactly how it works. A program which simulated the brain of an idealized mathematician might well consist of hundreds of thousands (or millions or billions) of lines of code. Imagine it given in the form of a volume the size of the New York telephone book. Then we might not be able to appreciate it in a perfectly conscious way, in the sense of understanding it or of being able to say whether it is plausible or implausible that it should output correct mathematical proofs and only correct mathematical proofs.

In short, there may be a program P_I which is sound and which produces precisely the same numbers n of true statements S_n that human mathematicians can produce. But for this it is necessary that human mathematicians are unable to ascertain that that is what P_I does.

Do Mathematicians Use Unsound Reasoning?

Are the statements S_I produced by human mathematicians as true really true? Might mathematicians claim that P_n does not produce n when it actually does? If P_n produces n this fact can in principle be determined, simply by executing P_n with all possible values for r_0, one after another, until n is produced.[27] Thus human mathematicians could eventually determine that they are making mistakes. In Chapter 3 of *Shadows of the Mind* Penrose argues that this possibility is implausible:
I cannot really see that it is plausible that mathematicians are *really* using an *unsound* formal system F as the basis of their mathematical understandings and beliefs. I hope the reader will indeed agree with me that whether or not such a consideration is *possible*, it is certainly not at all *plausible*.
But in fact there has been a loss of certainty in the soundness and completeness of mathematics, as Penrose acknowledges and discusses in sections 2.10 and 3.4 of *Shadows of the Mind*. Here are the comments of several influential mathematicians:
I have told the story of this controversy in such detail, because I think that it constitutes the best caution against taking the immovable rigour of mathematics too much for granted. This happened in our own lifetime, and I know myself how humiliatingly easy my own views regarding the absolute mathematical truth changed during this episode, and how they changed three times in succession!... It is hardly possible to believe in the existence of an absolute, immutable concept of mathematical rigor, dissociated from all human experience.

John von Neumann (from [26])

Mathematics may be likened to a Promethean labor, full of life, energy and great wonder, yet containing the seed of an overwhelming self-doubt. It is good that only rarely do we pause to review the situation and to set down our thoughts on these deepest questions. During the rest of our mathematical lives we watch and perhaps partake in the glorious procession.... This is our fate, to live with doubts, to pursue a

subject whose absoluteness we are not certain of, in short to realize that the only ``true'' science is itself of the same mortal, perhaps empirical, nature as all other human undertakings.

Paul J. Cohen (from [6])

I wanted certainty in the kind of way in which people want religious faith. I thought that certainty is more likely to be found in mathematics than elsewhere. But I discovered that many mathematical demonstrations, which my teachers expected me to accept, were full of fallacies, and that, if certainty were indeed discoverable in mathematics, it would be in a new field of mathematics, with more solid foundations than those that had hitherto been thought secure.... After some twenty years of very arduous toil, I came to the conclusion that there was nothing more that I could do in the way of making mathematical knowledge indubitable.[19]... The splendid certainty which I had always hoped to find in mathematics was lost in a bewildering maze.[20]

Bertrand Russell

Only he who recognizes that he has nothing, that he cannot possess anything, that absolute certainty is unattainable, who completely resigns himself and sacrifices all, who gives everything, who does not know anything, does not want anything and does not want to know anything, who abandons and neglects everything, he will receive all; to him the world of freedom opens, the world of painless contemplation and of - nothing.

L.E.J.Brouwer (from [3])

Unassailable Mathematical Beliefs

In Chapter 3 of *Shadows of the Mind*, Penrose considers *beliefs* rather than truths. Consider for example the result (5). If P_l is sound then S_l is true and P_l does not produce l. Thus we know a true statement S_l not produced by the program P_l. But this holds only if we *know* that P_l is sound. We may very well *not know for sure* that P_l is sound. Then we don't know for sure that S_l is true. On the other hand, if we *believe* that P_l is sound (but we're not absolutely sure) then we *believe* (with the *same* level of confidence, on the basis of (5)) that S_l is true and l is not produced by P_l. Then we will believe (with the same level of confidence, on the basis of (7)) that all the programs $P_l, P_{l1}, P_{l2}, \square$ are sound. Thus we may construct ever more comprehensive programs which we believe to be sound with the same level of confidence that we believe P_l to be sound. In summary:

We can go beyond any program P_l believed to be sound, and obtain a statement S_l which we believe to be true and not produced by P_l.

A similar argument can be given concerning formal mathematical systems. Take for example a formal system F of the sort we have considered earlier. If F is sound, so all its theorems are true, then we can exhibit a formula ϕ which is true (*in our standard interpretation*) but which is not a theorem of F. But again, we can only do this if

we *know for sure* that F is sound. Perhaps we are not certain that F is sound, but we *believe* it to be. (If we are using F to deduce mathematical truths we certainly will believe that we are using a sound formal system.) Then we *believe* that ϕ is true (with the same level of confidence, on the basis of (9)) and hence we can *add ϕ as a new axiom* to the formal system F, obtaining a new formal system F_1. We will then believe that F_1 is sound with the same level of confidence that we believe F to be sound. We have not lost anything by broadening our formal system with the addition of the new axiom ϕ. We may repeat this procedure, exhibiting a formula ϕ_1 which we believe to be true with the same level of confidence that we believe F_1 and F to be sound. Proceeding in this way we obtain a sequence of formal systems $F, F_1, F_2, \ldots$ each more comprehensive than the preceding one (having an additional axiom which is not a theorem of the preceding formal system). We believe each of the formal systems F_j to be sound with the same level of confidence that we believe F is sound. This procedure was first studied by Turing in a paper [25], *Systems of logic based on ordinals*, published in 1939. It is still a subject of research. In summary:

We can go beyond any sufficiently broad formal system F believed to be sound and obtain a formula which is believed true (in our standard interpretation) and not a theorem of F.

Penrose uses [14,15] this result as the basis for his argument against the computational modelling of mathematical understanding. But we will show in section 6 that his argument is mistaken.

6. A Computational Model for Thought?

Do the mathematical theorems on computation and formal systems have implications concerning the computational modelling of the human intellect? Here, we mean computational modelling of the sort that is done when modelling a wide range of physical phenomena. Consider for example the steps involved in the computational modelling of the motion of a projectile:

1. Decide on which aspects of the phenomenon are to be modelled: in the case of the projectile, the position of the projectile at various moments of time.
2. Set up measuring instruments to code those aspects of the phenomenon into numerical quantities. (The meter readings serve as `Gödel numbers' which code those aspects of the phenomenon into numbers.) In the case of the projectile, we set up rulers and clocks.
3. Set up a mathematical theory (in this case, based on Newton's Laws of Motion) to compute relationships between the instrument readings. In the case of the projectile, write a program such that with the registers appropriately initialized, the program is completed with the register R_0 containing a number which

should, if the computational model is a good one, be very close to the number obtained by reading the ruler.

Let's invent a toy model, a naive computational model for the way the brain works, and let's see if any of the mathematical theorems we have discussed could be used to rule out such a model. We follow the basic ideas set out in Chapters 1 and 2:

1. The functioning of the brain is described by the connections between neurons. The state of the brain at any time is described by the state of firing or non-firing of each of the neurons.
2. A mathematical model of the brain is constructed by associating with the j^{th} neuron a mathematical variable $f_j(t)$ which takes the value 0 or 1, describing the state of the neuron at time t (the time being measured in discrete steps). The value 1 corresponds to the neuron firing, and the value 0 to non-firing. The configuration of all the neurons at a given time is thus modelled by all the values $f_0(t),f_1(t),\Box,f_N(t)$. The values $f_0(t+1),f_1(t+1),\Box,f_N(t+1)$ are determined by the values $f_0(t),f_1(t),\Box,f_N(t)$ together with a computational rule which models the dynamical behaviour of the neural network. That is, we give a mathematical rule which enables us to calculate each $f_j(t+1)$ given all the values $f_0(t),\Box,f_N(t)$. One such rule is described in Chapters 1 and 2, in terms of weights which correspond to neural connections. The weights w_{ji} are introduced, which take positive or negative integer values[28], and model the strength of the connection from the i^{th} neuron to the j^{th} neuron. Then the computational rule governing the behavior of the mathematical model is:

$$f_j(t+1) = \begin{cases} 1 & \text{if} \quad w_{j1}f_1(t)+\Box+w_{jN}f_N(t)\Box>\Box M_j \\ 0 & \text{if} \quad w_{j1}f_1(t)+\Box+w_{jN}f_N(t) \le M_j \end{cases} \qquad (13)$$

3. where M_j is a positive integer corresponding to the threshold of the j^{th} neuron. **Remark.** External input can be incorporated into the model by `plugging in by hand' the values $f_j(t)$ corresponding to the sensory neurons.
4. The preceding computational rule may be expressed[29] by a program P_l. The program will be constructed so that by executing the program we perform the computations expressed in (11). We initialize the registers with $r_j = f_j(t)$ (all others initialized to 0) and execute the program P_l. When we complete the program the registers will contain $r_j' = f_j(t+1)$, for $j = 0,\Box,N$.

In this way by executing the program P_l we may compute the values $f_0(t),\ldots,f_N(t)$ for all (discrete) times t, and consequently the sequence of states of the neural network.

A Computational Model for the Mind?

The computational model of the brain discussed above may or may not give a good approximation to the actual behavior of a biological neural network. Assuming we could construct a good computational model of the excitation states of the neurons in the brain, what would this model tell us about the thoughts and emotions experienced by the brain? In other words, what does the model tell us about the mind? This depends on the nature of mental representation in the brain. In his book [8] Daniel Dennett begins Chapter 3, *Brain Writing and Mind Reading*, this way:

What are we to make of the popular notion that our brains are somehow libraries of our thoughts and beliefs? Is it *in principle* possible that brain scientists might one day know enough about the workings of our brains to be able to ``crack the cerebral code'' and read our minds?

Let's imagine that there is a `dictionary' which can translate from our description $(f_0(t),\ldots,f_N(t))$ of the neuronal state of a brain to the beliefs which the mind holds at time t. Denote[30] these beliefs by $B_0(t),\ldots B_{M(t)}(t)$. These beliefs could be expressed in English (together with mathematical symbols), or in French, or in any formal language. Define some Gödel numbering so that each belief $B_j(t)$ is assigned a number and the collection of all beliefs at time t is assigned the number b(t). So, we will be able to decode the number b(t) and obtain all the beliefs $B_0(t),\ldots, B_{M(t)}(t)$. Suppose the dictionary is computational in the sense that there is a program P_d such that if we initialize the registers to $r_k = f_k(t)$ then when we finish executing the program P_d the register r_0 will contain the *Gödel belief number* b(t).

Hence to compute the beliefs held by a brain at time t initialize the registers to $r_k = f_k(0)$, $k = 0,\ldots,N$. Then execute the program P_l. When you complete the program you will have $r_k' = f_k(1)$ in the registers. Continue these computations until you have obtained $f_0(t),\ldots,f_N(t)$. Now execute the dictionary program P_d with the registers initialized to $r_k = f_k(t)$. When you complete the program P_d you will obtain a number b(t) which encodes all the beliefs held by the brain at time t. The number b(t) could be decoded to yield all the beliefs $B_0(t),\ldots B_N(t)$ at time t stated in English. Combining the above steps, we can write a program P so that, initializing the registers to $f_0(0),\ldots,f_N(0)$,t, then upon completing the program P the Gödel belief number b(t) will be in the register R_0.

Gödel's Objection?

The above computational model of the brain is as straightforward and naive as one could imagine. But if there is no way to rule out, on the basis of the mathematical results discussed earlier, such a simple model then, ipso facto, there is no way to rule out more complex and sophisticated computational models on the basis of those mathematical results. Let's use the term `computational mind' to denote a mind whose beliefs could be computed in the above fashion.[31] We will show:

The Gödel and Turing theorems have absolutely nothing to say about the beliefs which may be held by a computational mind. Claims to the contrary are based on a mistaken application of those theorems.

Before discussing the details, consider the fact that people often hold contradictory beliefs without realizing it. Or they may believe completely foolish things, such as `The moon is made of green cheese.' Why should such beliefs affect the consistency and correctness of your computation of those beliefs? Some mathematicians accept the logical principle of the law of the excluded middle, others do not. Some may base their mathematical beliefs on modal logic, or on non-constructive methods, or on intuitionistic philosophy. Why should these varying sorts of beliefs affect the way the brain works at the neuronal (or any other) level or how you compute those beliefs? They don't! Concerning the disagreements among mathematicians on a sound basis for developing mathematical beliefs, consider this quotation from E.T. Bell (quoted in chapter XI of [12]):

Experience has taught most mathematicians that much that looks solid and satisfactory to one mathematical generation stands a fair chance of dissolving into cobwebs under the steadier scrutiny of the next... Knowledge in any sense of a reasonably common agreement on the fundamentals of mathematics seems to be non-existent... The bald exhibition of the facts should suffice to establish the one point of human significance, namely, that equally competent experts have disagreed and do now disagree on the simplest aspects of any reasoning which makes the slightest claim, implicit or explicit, to universality, generality, or cogency.

The consistency, correctness, or otherwise of a person's beliefs has nothing whatever to do with the consistency or correctness of the formal system used to deduce that person's beliefs. And what about thoughts in general? Think of something crazy. Something you believe is not, and could not possibly be, true. Should that crazy thing be a part of the deductive system forming a theory of mind? Or is it only the things you believe to be true that should be theorems? Surely all thoughts and other aspects of mind should be treated in the same way. You can think about anything you like; it won't affect the consistency or soundness of the theory of mind.

The Mistaken Application of Gödel's Theorem

Penrose uses Gödel's theorem in various arguments against the computational modelling of mathematical understanding throughout Chapter 3 of *Shadows of the*

Mind, in particular in sections 3.2, 3.3, 3.14, and 3.16. All his arguments are subject to the criticism levelled at the argument below. Here is Penrose's reasoning about Gödel's theorem in a nutshell.

Suppose a human, who we call H, has a computational mind (associated with the program P), and that H is made aware of the program P which we are using to compute the properties of his brain. We may use a formal mathematical system F to logically deduce the behavior of P, and hence the behavior, beliefs, and other thoughts of H. We may say that the system F encapsulates all the knowledge and beliefs of H. Penrose argues that *if H believes statement X, then since F encapsulates all of H's beliefs, it must be possible to prove X in the system F.* Penrose then reasons that H will surely believe that the system F is sound. Consequently, H will believe (by Gödel's theorem) that the formula ϕ is true. But since, again by Gödel's theorem, ϕ is not a theorem of F, it follows that F *cannot* after all encapsulate all of H's beliefs.[32] By this contradiction Penrose concludes that H *cannot* in fact have a computational mind.

Penrose's confusion in this line of reasoning is italicized above. For if H believes **X** (on day 1) then it is necessary that F deduces that **H believes X on day 1**. It is not necessary that F deduces **X**. If on day 2 H changes his mind and believes the negation of X, $\neg$**X**, then it is necessary that F deduces that **H believes $\neg$X on day 2**. It is not necessary that F deduces $\neg$**X**. If on day 3 H goes crazy then it is necessary that F deduces that **H is crazy on day 3**. It is not necessary that F deduces all sorts of crazy formulas.

In short, although F cannot deduce ϕ, there is no reason why F cannot deduce that **H believes ϕ**. Gödel's theorem doesn't say anything about what can be proved concerning the state of H's mind!

The essential point is that H's beliefs *do not form part of the deductive system F*. If H believes two contradictory statements **X** and **Y**, these statements are *not* theorems of F and so it does not follow that F is inconsistent, as it would be if **X** and **Y** were theorems.[33]

Let's explain in detail the distinction between **X** and **H believes X**, using our `toy' computational model. The statement **X** is one of H's beliefs, say $B_k(t)$, and as such is encoded along with H's other beliefs in the Gödel belief number b(t). Using the formal system F we can deduce a theorem which states that when the program P is run with registers initialized to $r_k = f_k(0)$ then when the program is completed $r_0 = b(t)$. There is a world of difference between the theorem $r_0 = b(t)$ and the statements $B_0(t)$,□,$B_N(t)$ encoded in the number b(t). The confusion undoubtedly arose because the mathematical beliefs of H could be expressed in the same language used in the formal

system F. No one would think of incorporating a non-mathematical belief, such as `It will rain tomorrow', into the formal system F !

Discussion

Perhaps you do not believe that a `mind reading' program can be found which will translate brain states into the beliefs (and other thoughts) of that brain. Consider any means which H may use to communicate his beliefs to others. He may write them down with pencil and paper, or he may say them. Does Gödel's theorem impose any restrictions on what H may write down or say? No! By computing the state of the motor neurons, using our program P_l, we can determine the hand movements which H will carry out, and hence the geometrical pattern of marks on paper which will result. Any pattern whatsoever can be drawn by H. And if that pattern may be interpreted as a mathematical formula, that formula does not thereby become a theorem of the formal system F associated with P_l. Similarly, the way H's mouth and vocal cords move, which could be computed using P_l, are not restricted by Gödel's theorem. The theorem which would be proved using F is of the form $r_j(t) = 1$, indicating a marble in the bowl R_j and hence, by our modelling of H's brain, that the j^{th} neuron is firing. Translating the marks on paper into, say, an English sentence, that sentence plays the role of a *term*, not the role of a theorem.

Consider a present day computer. What sort of expressions might it print out? Consider the formal system F associated with the program it is running. Suppose the formula ϕ is not provable in F. Is there any reason why the formula ϕ cannot be printed out by the computer? No! Again, what comes out of the printer is not a theorem of F.

All this becomes clear when it is realized that what is printed out, or drawn, etc., need not satisfy any grammatical rules, which are required of theorems of F.

Related Ideas in the Literature

Our criticism of Penrose's use of Gödel's theorem rests on the idea that beliefs and other thoughts of a mind are not theorems of the theory of that mind. This idea can be found in the existing literature.

6.0.1 John von Neumann

In August 1955 John von Neumann was diagnosed as having bone cancer. In April 1956 he was admitted to Walter Reed Hospital which he did not leave until his death on February 8, 1957. He took with him to the hospital the manuscript of the Silliman Lectures (Yale University), which he hoped to be able to present. The title of the

lectures was to be *The Computer and the Brain*. The unfinished manuscript has been published as a book [27]. The last section of the book is entitled *The Language of the Brain Not the Language of Mathematics*, and the last lines of that section and of the book are these:

When we talk mathematics, we may be discussing a *secondary* language, built on the *primary* language truly used by the central nervous system. Thus the outward forms of *our* mathematics are not absolutely relevant from the point of view of evaluating what the mathematical or logical language *truly* used by the central nervous system is. However, the above remarks about reliability and logical and arithmetical depth prove that whatever the system is, it cannot fail to differ considerably from what we consciously and explicitly consider as mathematics.

We may interpret this quotation as stating the view that the mathematical procedures and beliefs of a mind are of a different category from the theorems of the formal system F associated with a computational model of the mind at the neuronal level. In short, mathematical beliefs are not theorems of F, which is the point of our criticism of Penrose's argument.

Robert Kirk

Robert Kirk, a philosopher from Nottingham University, published an article [11] in 1986 entitled *Mental Machinery and Gödel*. He levels the same criticism at Lucas's use of Gödel's theorem that we have levelled at Penrose; reproduced by kind permission of Kluwer Academic Publishers:

If there could be adequate mechanistic accounts which represented a person by means of a formal system, yet did not correlate beliefs, thoughts or statements with that system's theorems or `outputs', then the argument from Gödel fails. In fact, there is an approach which appears to satisfy this requirement. It deals in terms of the organism's physical states, inputs, and outputs, all at some rather low level of specification - perhaps the neurophysiological - and in possible transitions from state to state through time.... Certainly mechanists will maintain that those tokens of mathematical sentences which Alf produces are produced `mechanically'. However, in maintaining this they need not - and should not - say that the token sentences themselves correspond to theorems of a formal system which adequately represents Alf and his part of the world.

7. Can Computers Think?

Let's finally return to our original question and see if the preceding considerations have gotten us any closer to an answer. We have seen that Penrose's argument against the computational modelling of mental processes does not work, and that there is no obstruction to a computational mind, of the sort discussed in section 6, arising from

the mathematical theorems on computation and formal systems. We conclude that these theorems have not, in fact, gotten us any closer to an answer to the question: Will computers of the future be intelligent?

Certainly present-day computers aren't intelligent.[34] We have quoted in section 1 several views that intelligent computers are a likely development in the not too distant future. Here is the vision of the mathematical physicist David Ruelle, which he paints at the beginning of his book [18]:

Supercomputers will some day soon start competing with mathematicians and may well put them forever out of work.... Of course, present-day machines are useful mostly for rather repetitious and somewhat stupid tasks. But there is no reason why they could not become more flexible and versatile, mimicking the intellectual processes of man, with tremendously greater speed and accuracy. In this way, within fifty or a hundred years (or maybe two hundred), not only will computers help mathematicians with their work, but we shall see them take the initiative, introduce new and fruitful definitions, make conjectures, and then obtain proofs of theorems far beyond human intellectual capabilities.

Some Thoughts about Thinking

We close with some thoughts about thinking, which you might like to think about. Consider again our computational model for the trajectory of a projectile. By executing the program to compute the numbers read on the ruler, we are not ourselves moving along the trajectory of the projectile! And if a computer is set up to execute the program, it has not itself become a projectile. Although this is painfully obvious, the implication in connection with models of the mind is perhaps that by setting up a computer to execute a program which constitutes a `computational model of the mind' (in the sense considered in section 6) the computer is not thereby *thinking* ! Dennett, in Chapter 11 of his book [8], puts it this way:

It is never to the point in computer simulation that one's model be *indistinguishable* from the modelled. Consider, for instance, a good computer simulation of a hurricane, as might be devised by meteorologists. One would not expect to get wet or wind-blown in its presence. That ludicrous expectation would be akin to a use-mention error, like cowering before the word ``lion''.

And Searle [22] puts it this way:

No one would suppose that we could produce milk and sugar by running a computer simulation of the formal sequences in lactation and photosynthesis, but where the mind is concerned many people are willing to believe in such a miracle because of a deep and abiding dualism: the mind they suppose is a matter of formal processes and is independent of quite specific material causes in the way that milk and sugar are not.

How might we build a computer which *does* think?

How to Build a Robot

Let's try to build a thinking robot using our URM. Our robot will have:

- eyes made of video cameras
- ears made of microphones
- a speaker for a voice box
- chemical sensors for a nose
- mechanical legs for moving about
- mechanical arms for touching and moving things

Have we left anything out? Oh yes, a *brain*. That's the crucial bit!

To make the brain, we'll use the URM. Take all the bowls and marbles, and your program P_l which defines a good computational model of some particular brain, and sit down inside the robot. Now suppose that when certain bowls have a marble in them (modelling an excited neuron) an action is performed by the robot (a small arm movement, for example). Suppose also that input from the video cameras (or microphones, etc.) leads to certain other bowls having a marble dropped in them (corresponding to the excitation of sensory neurons). You now execute the program P_l. Then, during each time step t□ t+1, the marbles in the bowls change (modelling the varying excitation states of the neurons in the brain) causing the robot to react to external stimuli, to speak through the speaker, and so on. Of course, you must imagine yourself working faster and faster, and the bowls of marbles very small. In this way, the collection of bowls of marbles with you executing the program P_l becomes the brain of the robot.

The Mind Demon

Let's summarize the above discussion this way. A mechanical robot has a brain composed of a very large number of very small bowls containing marbles[35], with a tiny very fast working `mind demon' executing a program P_l, changing the numbers of marbles in the bowls. The demon knows *only* how to execute the basic instructions of the URM. He understands nothing else.

Among the bowls are `input bowls' whose contents reflect properties of the outside world, and `output bowls' whose contents determine actions by the robot.

The sequence of bowl contents $r_j(t)$ at times t = 0,1,2,□, for j = 0,□,N corresponds to the sequence of neuron firings of the j^{th} neuron in the brain which is modelled by the program P_l. The program P_l corresponds to the connectivity of the neural network (through the weights w_{ij}). The demon, executing the program P_l, corresponds to the

electro-chemical forces which induce the temporal development of the neural network (its varying neuronal firings).

Question. You know that *you* think[36]; how close does this robot come to doing the same?

Searle's Chinese Room

John Searle [22] argues that although such a robot may listen to questions and stories and make sensible replies, there is no understanding:

> Whatever purely formal principles you put into the computer, they will not be sufficient for understanding, since a human will be able to follow the formal principles without understanding anything.... I will argue that in the literal sense the programmed computer understands what the car and the adding machine understand, namely, exactly nothing. The computer understanding is not just (like my understanding of German) partial or incomplete; it is zero.

For `human' in this quote, substitute our `demon', whom we have agreed does not understand *anything* except the basic instructions required in the execution of the program P_I. Remembering that we said the demon corresponds[37] to the electro-chemical forces responsible for the temporal development of the neural network, we certainly would not ascribe `understanding' to those forces, nor to the demon. But although the demon does not understand anything of the outside world, does it follow that the robot has no understanding?

Imagine the robot in China, and the program P_I giving a good computational model of the brain of some Chinese person. The robot will function in China in the same way that the Chinese person does. It will make replies, in Chinese, to questions asked of it by Chinese people. Searle imagines himself taking the part of the demon. He then compares his excellent understanding of English with his total lack of understanding of Chinese, and concludes that also the robot could not `understand' Chinese. But again, the demon, or Searle-in-the-robot, is not where understanding is expected to reside. Searle's reference to the understanding of English which Searle-in-the-robot possesses involves a category-mistake, as pointed out by Margaret Boden [1]:

> Searle's description of the robot's pseudo-brain (that is, of Searle-in-the-robot) as understanding English involves a category-mistake comparable to treating the brain as the bearer - as opposed to the causal basis - of intelligence.

The `Feel' of Thinking

Can our robot with its `bowls-of-marbles + demon' brain see a sunset the same way that we see it? Does it really think and understand in the same way that we do? As we

don't expect to build a brain in this way (where would we find the demon?) the question is not very important. Suppose, instead of the bowls of marbles, we use silicon transistors, and instead of the demon, we use electromagnetic forces as they are exerted on the transistors of a silicon chip. Suppose, in other words, that we make a copy of a brain `in silicon', in such a way that the sequence of patterns of `on/off' transistors induced by electromagnetic forces is the same as the sequence of patterns of `firing/non-firing' neurons induced by electrochemical forces. Would our robot, with such a silicon chip central processing unit for a brain, see and think in the same way that we do? How would it be for the robot: Would it `feel' the same? Searle [22] doesn't think so:

The problem with the brain simulator is that it is simulating the wrong things about the brain. As long as it simulates only the formal structure of the sequence of neurone firings at the synapses, it won't have simulated what matters about the brain, namely its causal properties, its ability to produce intentional states.

But recall Turing's test based on *behavior*. The physicist Niels Bohr echoes Turing's ideas this way [10]:

I am quite prepared to talk of the spiritual life of an electronic computer; to say that it is considering or that it is in a bad mood. What really matters is the unambiguous description of its behaviour, which is what we observe. The question as to whether the machine *really* feels, or whether it merely looks as though it did, is absolutely as meaningless as to ask whether light is ``in reality'' waves or particles. We must never forget that ``reality'' too is a human word just like ``wave'' or ``consciousness.'' Our task is to learn to use these words correctly - that is, unambiguously and consistently.

Be that as it may, the `feel' of the functioning brain could very well depend on the detailed structure of the individual neurons, much as the tone of a violin depends on the quality of wood used and the manufacturing techniques employed. So a computer's silicon brain might `feel' different from our biological brain, but nevertheless, *it would be a thinking computer*.

References

[1] M.A. Boden, *Escaping from the Chinese Room*, in *The Philosophy of Artificial Intelligence*, ed. M.A. Boden, (Oxford University Press, 1990), pp.40-66

[2] L. Boltzmann and B. McGuinness, *Theoretical Physics and Philosophical Problems*, vol.5 (D. Reidel Publishing Company, 1974) p.57

[3] L.E.J. Brouwer, *Life, Art and Mysticism 1905 exerpts*; in Collected Works 1, ed. A. Heyting, (North-Holland, 1975)

[4]

J.N. Crossley, et.al., *What is Mathematical Logic?*, (Oxford University Press, 1978)

[5]
G. Chaitin, *Gödel's Theorem and Information*, International Journal of Theoretical Physics **22** (Plenum Publishing Corp., 1982) pp.941-954; reprinted in G.J. Chaitin, *Information, Randomness, and Incompleteness* (World Scientific,1987)

[6]
P.J. Cohen, *Comments on the Foundations of Set Theory*, in Proceedings of the Symposia in Pure Mathematics Volume XIII Part I, (American Mathematical Society, 1971)

[7]
N.J. Cutland, *Computability*, (Cambridge University Press, 1980)

[8]
D.C. Dennett, *Brainstorms*, (Harvester Wheatsheaf, 1981)

[9]
K. Fulves, *Self-Working Number Magic* (Dover, 1983)

[10]
J. Kalckar, *Niels Bohr and his youngest disciples*; in S. Rozental, ed., *Niels Bohr: His life and work as seen by his friends and colleagues*, (North-Holland, 1967)

[11]
R. Kirk, *Mental Machinery and Gödel*, Synthese **66** (D. Reidel Publishing Company, 1986) pp.437-452

[12]
M. Kline, *Mathematics: The Loss of Certainty*, (Oxford University Press, 1982)

[13]
J. Lucas, *Minds, Machines and Gödel*, Philosophy, Vol.XXXVI (Cambridge University Press, 1961); reprinted in *Minds and Machines*, ed. A.R. Anderson, (Prentice Hall, 1964)

[14]
R. Penrose, *Shadows of the Mind*, (Vintage, 1995)

[15]
R. Penrose, *Psyche*, Volume 2, Number 1 (1996) pp.89-129

[16]
H. Putnam, *The Best of all Possible Brains* (Review of Shadows of the Mind by R. Penrose), New York Times, 20 November 1994.

[17]
H. Putnam, *Mathematics without foundations*, Journal of Philosophy **64** (Journal of Philosophy Inc., 1967) pp.5-22; reprinted

in *Philosophy of Mathematics*, edited by P. Benacerraf and H. Putnam, (Cambridge University Press, 1985)

[18] D. Ruelle, *Chance and Chaos*, (Princeton University Press, 1991)

[19] B. Russell, Portraits from Memory, Bertrand Russell Peace Foundation (George Allen and Unwin, 1956)

[20] B. Russell, My Philosophical Development (Unwin, 1975).

[21] A. Scott, *Stairway to the Mind*, (Copernicus, 1995)

[22] J.R. Searle, *Minds, Brains, and Programs*, Behavioral and Brain Sciences **3** (Cambridge University Press, 1980) 417-24, reprinted in *The Philosophy of Artificial Intelligence*, ed. M.A. Boden, (Oxford University Press, 1990), pp.67-88

[23] J.C. Shepherdson and H.E. Sturgis, `Computability of Recursive Functions', *J. Assoc. Computing Machinery* **10** (1963) 217-255

[24] A. Turing, *Computing Machinery and Intelligence*, Mind **59** No.236 (Oxford University Press, 1950) 433-460, reprinted in *The Philosophy of Artificial Intelligence*, ed. M.A. Boden, (Oxford University Press, 1990), pp.40-66

[25] A. Turing, *Systems of Logic Based on Ordinals*, P. Lond. Math. Soc. (2) **45** (1939) 161-228

[26] J. von Neumann, *The Mathematician. ``Works of the Mind.''* in Collected Works Volume 1, ed. A.H. Taub, (Pergamon Press, 1961)

[27] J. von Neumann, *The Computer and the Brain*, (Yale University Press, 1958)

[28] K. Warwick, *March of the Machines*, (Century, 1997)

Footnotes:

[1] For more details see Intel's website

`http://www.intel.com/pressroom/archive/releases /cn121796.htm`
and Sandia's website
`http://www.sandia.gov/LabNews/LN06-20-97/teraflops_story.html`

[2] In addition to Putnam's review, discussed here, an on-line version of Soloman Feferman's review of *Shadows of the Mind* can be found at

`http://psyche.cs.monash.edu.au/v2/psyche-2-07-feferman.html`

You can find various other criticisms of Penrose's approach, together with replies by Penrose, at

`http://psyche.cs.monash.edu.au/v2/psyche-2-23-penrose.html`

A large bibliography on the subject of `mind and metamathematics' can be found at

`http://ling.ucsc.edu/~chalmers/biblio4.html .2`

which is *Part 4: Philosophy of Artificial Intelligence* of the Contemporary Philosophy of Mind: An Annotated Bibliography, at

`http://ling.ucsc.edu/~chalmers/biblio.html`

[3] Putnam's own view on the nature of human mentality is indicated by the following quote from [17]:

Why *should* all truths, even all empirical truths, be discoverable by probabilistic automata (which is what I suspect we are) using a finite amount of observational material?

[4] All quotations from Turing's article in this chapter are reprinted by permission of Oxford University Press.

[5] In 1990 Dr. Hugh Loebner pledged a Grand Prize of $100,000 for the first computer whose responses were indistinguishable from a human's. Each year a prize of $2000 is awarded to the most human computer, which this year was won on May 16, 1997 by the program Converse, entered by the University of Sheffield. The homepage of the Loebner prize is

`http://acm.org/~loebner/loebner-prize.htmlx`

[6] On May 11, 1997 the IBM computer Deep Blue won a six game match against the world chess champion Garry Kasparov. For more details see

`http://www.chess.ibm.com/home/html/b.html`

[7] The difficulty in defining each and every concept one uses is nicely put [2] by Ludwig Boltzmann (1844-1906); reprinted by kind permission of Kluwer Academic Publishers:

Let me begin by defining my position by way of a true story. While I was still at high school my brother (long since dead) tried often and in vain to convince me how absurd was my ideal of a philosophy that clearly defined each concept at the time of introducing it. At last he succeeded as follows: during a lesson a certain philosophic

work (I think by Hume) had been highly recommended to us for its outstanding consistency. At once I asked for it at the library, where my brother accompanied me. The book was available only in the original English. I was taken aback, since I knew not a word of English, but my brother objected at once: ``if the work fulfils your expectations, the language surely cannot matter, for then each word must in any case be clearly defined before it is used."

[8] To determine if the number of marbles in two bowls is the same, remove one marble from each bowl, then another marble from each bowl, and so on. If both bowls become empty at the same step then they contained the same number of marbles. If only one bowl is empty then they contained different numbers. (Then replace the marbles in their respective bowls.)

[9] We do not allow instructions of the sort, for example, $S(r_1)$, which adds a marble to the bowl with number equal to the content of register R_1. In fact, although we are using numbers as names for the bowls, any symbols could be used for these names. They do not have the same quantitative significance as the content of a bowl. We also do not allow instructions of the sort $J(1,3,r_0)$ which may cause a jump to the instruction with number equal to the content of register R_0. Again, an instruction number and a number of marbles are two different types of things. One is an *ordinal* number, representing an ordering (first, second, etc.), the other a *cardinal number*, representing an amount.

[10] The URM was invented by J.C.Shepherdson and H.E.Sturgis [23]. A nice discussion of the URM is given by N.J.Cutland [7].

[11] The working registers are R_0,R_1,R_2; the others remain unchanged and do not influence the computation.

[12] Used explicitly as part of the jump instructions and implicitly as part of the transfer instructions.

[13] The case n = m is dealt with by the first application of J(1,2,7).

[14] See Cutland [7] for further discussion of this and other assignments.

[15] Here is a program to add *l* to any number (which is placed in register R_0, all other registers initialized to 0): S(0),S(0),□,S(0) [*l* copies].

[16] Here is a program to subtract 1 from any number m (which is placed in register R_0, all other registers initialized to 0): S(1),J(0,1,5),S(1),S(2),J(0,0,1),T(2,0),Z(1),Z(2). If $m \geq 1$, when you complete the execution of this program there will be m-1 in register

R_0, all other registers containing 0. If m = 0, this program will not halt; i.e. you will never complete the program.

[17] Even if you could carry out your reasoning without language, you could not communicate to others how you arrived at your conclusions unless a common language is agreed.

[18] What rules of deduction should be included in the formal system? Is there only one correct logical system? Consider other areas of mathematics. What is the correct geometrical theory? The classical geometry of Euclid was considered the only correct geometry for thousands of years, but now we consider the correct geometry to be an *empirical* question. The curved space-time of Einstein's general relativity theory is considered a better mathematical model than the flat Euclidean geometry. What is the correct algebraic theory? The algebraic structure of a theory of observable quantities had been supposed to satisfy xy = yx (commutativity) and x(y+z) = xy+xz (distributivity). But now it is considered that quantum theory, involving non-commutative observables, is a better mathematical model of atomic phenomena. Quantum theory still keeps distributivity, but if the empirical data require it, then a further modification of the algebraic structure of the theory could be made. What is the correct logic? Perhaps there is no one correct logic. It may also depend on the properties of the system being modelled. We have *empirical geometry*, *empirical algebra*, why not *empirical logic*?

[19] We could use a method similar to the one used to encode programs, since programs are finite sequences of instructions and formulas are finite sequences of symbols - and proofs are finite sequences of formulas. So first give numbers to each symbol in the alphabet of L. If ϕ is the formula $a_1 \square a_b$, where m_j is the Gödel number of the symbol a_j, then ϕ is assigned the number $h(m_0,\square,m_b)$, where the function h is given in (4). Similarly, if $\phi_0,\square,\phi_b$ is a finite sequence of formulas where m_j is the Gödel number of the formula ϕ_j, then the sequence of formulas is assigned the number $h(m_0,\square,m_b)$.

[20] Different interpretations are often given to the same English sentence by different people. They may think they have a disagreement as to which statements are true, but actually they may simply have assigned a different meaning to the same words! This is called a problem of semantics.

[21] Consequently, if the formal system is sound and if there is a formula ϕ which is true in one interpretation and false in another, then neither ϕ nor its negation can be a theorem of the formal system.

[22] Someone else may interpret the formulas in a completely different way, as statements about some other activity or physical process.

[23] We suppose that when P_m halts, all its working registers other than R_0 contain 0. Similarly for P_M. This can always be arranged. In Figure we take the working registers of P_M to be R_k for $k \leq N_M$.

[24] In our interpretation ϕ_j expresses the statement S_j, that the j^{th} program does not produce the number j. However, in another interpretation ϕ_j no longer has this meaning. The relation between ϕ_j and S_j is lost.

[25] The element of self-reference in the definition is inevitable. The oracle must be able to refer to itself or to something dependent on the oracle. As we have seen in the case of formal systems, a formula can be written down which may be interpreted as stating that the self-same formula cannot be proved by the formal system.

[26] Recall the Feynman story above.

[27] On the other hand, if P_n does not produce n but a mathematician claims that it does (without any indication as to what input would produce n), it may not be possible to prove him wrong. There may be no way to determine the truth or falsity of S_n and the law of the excluded middle (every statement is either true or false) may not be applicable.

[28] Positive values of w_{ji} corresponds to exitation and negative values to inhibition. Given rational values for the weights, we can multiply through by a sufficiently large integer to rewrite the equations with integer-valued weights.

[29] Negative integers need not enter directly into the computation. Use may be made, for example, of the *difference function* Dif(n,m) = n-m if $n \geq m$ and = 0 if $n < m$.

[30] We shall naturally suppose that a (finite) brain can have only finitely many beliefs at any one time.

[31] What could be more computational than that?

[32] It may be that the formal system F does not satisfy the requirements to deduce Gödel's theorem for F. But the point of the argument is that there are statements which H believes to be true but which are not theorems of the formal system F. This will undoubtedly be the case for mathematical statements of a sort which cannot even be *expressed* in the particular formal system F, and even more so for non-mathematical beliefs.

[33] If H's beliefs are not theorems of F, what are they? The beliefs are encoded into the Gödel belief number b(t), which is a *term* of the formal language.

[34] They only think they are.

[35] To the reader with such a brain: `Don't lose your marbles!'

[36] Well, maybe.

[37] According to the construction of present-day computers, we could also say that the demon corresponds to the CPU (central processing unit) of the computer, which, while executing a program, manipulates the numbers (on/off transistor patterns) stored in its registers and RAM memory locations.