

Appendix A

Penrose's Argument, and a Response

Usefully, I have no need to rely so directly on intuitions to show why I believe Penrose's argument, the heart of which is neatly summarized in pages 72-77 of *Shadows of the Mind* (1994), to be seriously misleading. That is because Penrose himself retreats from arguments based on "obvious" truths, which he makes earlier in the book, to the seemingly safer ground of logic. (Further, he limits the required logic to no more than an introductory university-level understanding.) I will put Penrose's argument in my own words but be at pains (I trust) not to change it.

A.1 Penrose's Argument Step-by-Step

Penrose's argument is made via a version of the well-known halting problem, which is generally accepted to be algorithmically undecidable. Consider a set of functions \mathcal{C} all of whose members take as input an arbitrary natural number n and produce some other natural number x as output, by iterating over the natural numbers starting with zero and returning the first acceptable value of x that they find. Some possible members of \mathcal{C} are:

- $C_1(n) = x$ where x is odd and $x > n$.
- $C_2(n) = x$ where x is odd and is twice the value of n .
- $C_3(n) = x$ where x is not the sum of n squares.

It should be clear from grade school mathematics that the first function quickly terminates given any value of n and the second never does, regardless of n . On the other hand, the third one (Penrose's example) terminates only on 0, 1, 2, and 3 (according to the mathematicians), which is probably *not* immediately obvious.

What this means is that certain members of \mathcal{C} will terminate regardless of the value of n ; certain members will fail to terminate regardless of the value of n ; and certain members will terminate only on certain values of n . Let us refer to any specific member of \mathcal{C} as C_q , since we could, in principle, list all the possible members of \mathcal{C} and assign a natural number q as an index to each. It is essential to Penrose's argument that for any value of n , the listing should include every possible function that can take a single value n as input and return x . It is likewise essential that for any particular C_q , there is no value of n that it cannot take as input.

Next, consider another function A . A is a two-argument function that takes as input some C_q – namely, the q th C_q – and some value of n and terminates with an output of, say, 1 for “success” (the output really does not matter), if and only if that particular C_q does *not* terminate for that value of n . Otherwise, A does not terminate. It is important that A is sound (it does not make mistakes) and knowably sound (it can be strictly relied upon not to make mistakes).

Penrose suggests that we take A to encapsulate “*all* the procedures available to human mathematicians” (1994, p. 73) for deciding whether or not $C_q(n)$ terminates. Two things are essential here:

1. If $A(q,n)$ stops then $C_q(n)$ does not stop.
2. If $C_q(n)$ stops then $A(q,n)$ does not stop.

Note that the following do *not* hold: i.e., they do *not* follow from the previous two statements.

3. If $C_q(n)$ does not stop, then $A(q,n)$ stops.
4. If $A(q,n)$ does not stop, then $C_q(n)$ stops.

That is to say, the failure of either $A(q,n)$ or $C_q(n)$ to stop says nothing at all about whether the other will stop. This asymmetry is likewise critical, as shall be shown below.

What happens when $q = n$ (as must be a possibility, given that there are no restrictions on the value of either q or n)? In that case, we can re-write (1) as:

5. If $A(n,n)$ stops then $C_n(n)$ does not stop.

Remember, however, that \mathcal{C} contains every possible C_q that takes some n as input and returns some x . Since $A(n,n)$ is now a function of one argument, it must be on the list. If it is the k th element of the list, then, in that case, n takes the value k and:

6. If $A(k,k)$ stops then $C_k(k)$ does not stop.

At the same time, because $A(k,k)$ is the k th element of the list, $A(k,k) = C_k(k)$. Substituting back into (6), one gets:

7. If $C_k(k)$ stops then $C_k(k)$ does not stop.

If this looks like one half of the “eternal oscillation between two competing and contradictory points of view” I have discussed at several earlier points, it should. The successful termination of $C_k(k)$ yields a contradiction. “From this”, writes Penrose, “we must deduce that the computation... [$C_k(k)$] does *not* in fact stop” (1994, p. 75). Indeed, we can do so without any contradiction.

A.2 The Fly in the Ointment

In order for Penrose’s argument to go through, he needs to make the following assumptions about human mathematical reasoning: -Human mathematical reasoning is sound. That is, every statement that a competent human mathematician considers to be “unassailably true” actually is true. -The fact that human mathematical reasoning is sound is itself considered to be “unassailably true” (McCullough, 1995, p. 3).

What stops Penrose, and the rest of us, from completing the loop and being caught in the “eternal oscillation”? Why does (8) not follow (i.e., how is this situation crucially different from Russell's Paradox or the Epimenides Paradox)?

8. If $C_k(k)$ does not stop then $C_k(k)$ stops.

It is, as Penrose properly concludes, the incompleteness of A : i.e., its inability to produce *every* correct answer. A 's completeness would trap us in the very paradox from which Penrose wishes to show we are free:

Thus, our procedure A is incapable of ascertaining that this particular computation... [$C_k(k)$] does not stop even though it does not. Moreover, if we *know* that A is sound, then we *know* that... [$C_k(k)$] does not stop. Thus, we know something that A is unable to ascertain. It follows that A *cannot* encapsulate our understanding (Penrose, 1994, p. 75).

Unfortunately for Penrose, whether that last part follows depends quite critically on what exactly it is that we know. It depends, further, on what we *know* that we know¹: to know without being aware of that knowing, or only to *think* that we know, will not be enough. There cannot be possibility for error on our part in reaching our conclusion about $C_k(k)$.

I take it as untendentious that A cannot be a member of \mathbf{C} – if A is sound. (Remember Grush and Churchland's point that A might be benignly unsound.) The potential controversies arise, as Penrose is aware, with how A is understood. McCullough is concerned, as I am, with the ambiguities here. So, for example, the various C_q s can be understood, again untendentiously, as Turing machines; but Penrose considers actual, real-world computers *just to be* universal Turing machines, something to which I have objected.

Earlier Penrose invited us to consider that A encapsulates “*all* the procedures available to human mathematicians”. What Penrose wants us to conclude, of course, is that A encapsulates only the computational (or *algorithmically describable*) procedures. Mathematicians (and the rest of us) have access to a different, non-computational, function that includes non-computational as well as computational procedures and is able to decide what A cannot. Call this new function R . If A and R are thought of, for sake of argument, as sets of procedures, then A is contained within R : $A \subset R$.

It would seem to follow, however, that R must *also* be incomplete in order to break ourselves out of the “eternal oscillation”: i.e., Penrose's argument trades on the incompleteness not just of A but of R . Why is this? If R could produce every correct answer, and irrespective of diagonalization arguments², it would follow that:

9. If $R(q,n)$ stops then $C_q(n)$ does not stop (equivalent of (1); required by soundness).
 10. If $C_q(n)$ stops then $R(q,n)$ does not stop (equivalent of (2); required by soundness).
 11. **If $C_q(n)$ does not stop, then $R(q,n)$ stops (equivalent of (3)).**
 12. **If R does not stop, then $C_q(n)$ stops (equivalent of (4)).**

¹(Chalmers, 1995) echoes this concern.

²This is also what I understand to be Daryl McCullough's conclusion: see (1995, pp. 3-4). By subtly different argument, Chalmers (1995) argues that what I am calling R might be sound but cannot be (as Penrose requires) *knowably* sound.

Such is sufficient, I believe, to construct a Liar's-Paradox-type oscillation. Either R is, itself, a member of \mathcal{C} (and hence unsound) or R is (perhaps benignly) unsound.

As Daryl McCullough puts it, "... the Gödel argument doesn't prove that human reasoning must be noncomputable – it only proves that if human reasoning is computable, then it must either be unsound, or it must be inherently impossible for us to know both what our own reasoning powers are and to also know that they are sound" (1995, p. 3). Furthermore, "... Even though it might be the case that nothing false is ever judged to be unassailably true, this fact cannot be an unassailable truth" (1995, p. 5).

A.3 So What *Do* We Know?

The lesson I suggest we take from Gödel is precisely that resorting to a larger system – even a partly non-computational one (whatever we take that to mean and I, for one, am unclear, given Penrose's very broad definition) – does not prevent the replacement of one Gödel sentence with another, any more than the messy, informal system we call human language prevents us from constructing riddles like the Liar's Paradox. This is, precisely, Hofstadter's (1979) conclusion. Furthermore, the existence of paradoxes we can see, and not resolve, implies the possibility of paradoxes we cannot see, and therefore cannot consider resolving.

What is it that Penrose's argument actually allows us to conclude that we know? I suggest it is this: that there exist certain specific values of n for which we do, indeed, know that any function that attempts to calculate whether $C_n(n)$ stops will either be caught in an eternally oscillating loop or derive a contradiction. There is no evidence here, and no argument from Penrose:

Either that we can know for ourselves what those specific values of n are. (Of course we could, and would, know those values if R were complete, but R cannot be complete.)

Or that A cannot conclude (as a general principle) that there will exist *some* values of n that will produce undecidable propositions.

After all, what Penrose (and Gödel and Turing and Cantor) have shown is only that there are *certain specific* values of n that will produce undecidable propositions.

In conclusion, there is no evidence from Penrose that R is able to do anything that A cannot. In particular, he has not shown that a Gödel-type sentence cannot be constructed within the larger system: to wit, resorting to a "more complete" system does not, of itself, change anything. Is it still possible that R is more complete (or less incomplete) than A ? Of course it is – at least until such time as we have an untendentious definition of what is (and is not) computational or algorithmic. What R cannot be is complete (or rather, complete and consistent). Even then, a similarly structured argument gives us reason to think that a *completely* reliable decision procedure to distinguish what is computational from what is non-computational will not be forthcoming.