CS80—Senior Seminar
Discussion Questions—Penrose Chapters 3 & 4

Jim Rogers

jrogers@cs.earlham.edu

Fall 2000

These are a few questions for thought and discussion.

1. In the section *How to outdo an algorithm* Penrose observes that the machine (T_k) that Turing's proof constructs to confound the purported algorithm for the halting problem (H) must fail to halt when run on its own code number. Why is this?

   So, this gives us an example of a question that we know the answer to ("Does $T_k(k)$ halt?") that the algorithm must fail to get right. Is this, then, an example of human cognitive capacity that necessarily outstrips mechanical computation?

   Penrose then goes on to consider augmenting H with the fact $H(k; k) = 0$ (i.e., "$T_k(k)$ halts"). How does the augmented machine get around the contradiction that prevents the original H from getting this right to begin with? If we continue to patch up these algorithms like this we will, of course, only be able to cover finitely many errors. (So the original must have been wrong infinitely often!) Suppose, though, one had an algorithm that (magically) included patches for the whole infinite series of counterexamples for the flawed algorithms. Would this, then, necessarily solve the Halting Problem?

2. In Chapter 4, Penrose points out the parallel between this aspect of Turing's proof and Gödel's incompleteness proof. (Gödel's first great theorem, by the way, was his *Completeness Theorem*, a fact that has a satisfyingly paradoxical flavor.) He uses this to introduce the notion of a *reflection principle*, which he takes to be evidence of the distinction between human cognitive processes and computation. Are these reflection principle arguments, in fact, necessarily non-algorithmic? On what grounds does Penrose base his distinction?

3. In introducing the notion of *Recursively Enumerable* (**r.e.**) sets, Penrose points out that there is an effective procedure for finding a proof of any

true proposition. (This is not strictly algorithmic, since it must necessarily fail to halt on false propositions.) What happens if we start two copies of this procedure simultaneously, one searching for a proof of $P_k(k)$ (the counterexample of Gödel's proof) and the other on $\sim P_k(k)$? Penrose makes much of the role of insight in distinguishing mathematical reasoning from the simple enumeration of proofs. But there is a strong similarity between these two approaches, at least to the extent that neither is certain of arriving at an answer. What grounds are there for assuming that mathematical insight is not just exhaustive search under the covers?