# Degrees of Undecidability

Prakash Panangaden

24th November 2015

In this note we work with computable numerical functions, i.e. functions from $\mathbb{N}$ to $\mathbb{N}$; these may be total or partial. One of the early attempts to define "computable functions" is due to Gödel; he defined what we call today the *primitive recursive functions.* I will not give the formal definition here, but I note that these are precisely the functions that one could compute in a programming language that had bounded loops; i.e. like the `for` construct in FORTRAN. The important point is that all the primitive recursive functions *always terminate.* With `while` loops of course one gets all the (partial) computable functions.

Kleene showed that if one adds unbounded search to the primitive recursive functions then one gets all the (partial) computable functions. Kleene showed that they correspond precisely to what one can compute with a Turing machine. A key part of Kleene's proof of the theorem above is that one can define a primitive recursive function, $T$, such that $T(M, w, n)$ returns `true` if the Turing machine $M$ (encoded as a number) applied to the word $w$ (also encoded as a number) accepts (or halts, it does not really matter which) within $n$ steps. Now if you can search through all $n$ to find out if $T(M, w, n)$ is ever true, you can tell how many steps $M$ takes to accept $w$: of course, this unbounded search may never terminate. If we accept the existence of the Kleene $T$ predicate, then the fact that partial computable functions encode Turing machine computations is pretty clear. For details on the definition of $T$, see the book "Introduction to Metamathematics" by S. C. Kleene.

In order to work exclusively with numbers I will think of some enumeration of all the Turing machines $M_k$; for each value of the positive integer $k$ we get a Turing machine $M_k$ and every Turing machine is $M_k$ for some $k$. I will write $T(k, m, n)$ to mean that the $k$th Turing machine halts on input $m$ within $n$ steps. Now the halting problem can be described as

$$\mathsf{H}_{TM} = \{(k, m) \mid \exists n,\ \ T(k, m, n)\}.$$

This is the prototypical CE set, in fact it is CE complete: every CE set can be reduced to this one.

What about co-CE sets? The prototypical co-CE set is

$$\neg\mathsf{H}_{TM} = \{(k, m) \mid \forall n,\ \ \neg T(k, m, n)\}.$$

The existential quantifier has become a universal quantifier! What if we stack up more quantifiers? What does the following set consist of?

$$\{k \mid \forall m\, \forall\, n,\ \ \neg T(k, m, n)\}.$$

This says that Turing machine $k$ never halts on *any* input, in short this is $\mathsf{EMPTY}_{TM}$. This has two quantifers, but it is also only co-CE. Well we know that, by using a pairing function, we can combine a pair of integers as a single integer; so we do not really get a more complicated problem by using two universal quantifiers instead of one.

Can we get more complicated problems that are beyond CE and co-CE? We have seen an example of a problem that is neither CE nor co-CE. Here both quantifiers are universal. What happens if we combine universal and existential quantification?

Consider the set described below

$$\{k \mid \forall m, \;\; \exists n, \;\; T(k, m, n)\}.$$

This says that for every word the $k$th Turing machine halts. In short it tells you which Turing machines define *total* functions. This set $\mathsf{TOTAL}$ is beyond CE and co-CE. By Rice's theorem it is clearly not decidable. Let us show reductions to $\mathsf{HP}$ and $\neg\mathsf{HP}$. Given $(k, m)$ we want to know whether $M_k$ halts on input $m$. We define a new TM $M'$ by saying $M'(x)$ ignores $x$ and simulates $M_k$ on $m$, if $M_k(m)$ halts then so does $M'(x)$. Clearly if $M_k(m)$ halts $M'$ is total. Note this exact reduction shows that $\neg\mathsf{HP} \leq_m \mathsf{EMPTY}$; we have just shown that $\mathsf{HP} \leq_m \mathsf{TOTAL}$. Note we do not have a maping reduction of $\mathsf{HP}$ to $\mathsf{EMPTY}$.

Now consider the other reduction, we want to show that $\neg\mathsf{HP} \leq_m \mathsf{TOTAL}$. Given $(k, m)$ we define a TM $M'$ as follows. On input $x$ the TM $M'$ simulates $M_k$ on $m$ for $x$ steps. If $M_k$ has halted, $M'$ goes into an infinite loop, if $M_k$ has not yet halted then $M'$ halts. In this case if $M_k$ never halts on $m$ the TM $M'$ will *always* halt, thus we have the reduction we want. Note that we cannot have a reduction of $\mathsf{HP}$ to $\mathsf{EMPTY}$.

Let us consider the following set

$$\mathsf{FIN} := \{k \mid \exists m \; \forall x, n \text{ if } x > m \;\text{ then } \neg T(k, x, n)\}.$$

This means that there is some bound above which the TM $M_k$ never halts on any input, this means that its domain is *finite*. It is easy to come up with a reduction showing that $\mathsf{FIN}$ is not CE. The same reduction that we gave in the last paragraph also shows that $\mathsf{HP} \leq_m \mathsf{FIN}$ so this set is also not co-CE. It is *not*, however, equivalent to $\mathsf{TOTAL}$.

To give the classification of what is called the *arithmetic hierarchy* we introduce some notation. It is clear that what matters is the *alternation* of quantifiers; we cannot use pairing when we have different quatifiers next to each other. We look at the first quantifier appearing in the description. If it is an existential quantification we write $\Sigma$, if it is a universal quantification we write $\Pi$. We *always* put a superscript of 0. Now we start counting from 1, and read towards the right, every time we change the quatifier type we increment our counter. We use the final value of this counter as the subscript. Thus CE sets are $\Sigma^0_1$ and co-CE sets are $\Pi^0_1$. The set $\mathsf{FIN}$ is in $\Sigma^0_2$ and $\mathsf{TOTAL}$ is in $\Pi^0_2$. In this way we can go on and define the classes $\Sigma^0_n$ and $\Pi^0_n$ for any $n$. We write $\Delta^0_n$ for the intersection of $\Sigma^0_n$ and $\Pi^0_n$. Thus $\Delta^0_1$ is just the collection of decidable sets.

I will state without proof some key facts.

- The sets $\Sigma^0_n$, $\Pi^0_n$ and $\Delta^0_n$ are all different.

- For all $n$, $\Sigma_n^0 \subset \Sigma_{n+1}^0$, $\Pi_n^0 \subset \Pi_{n+1}^0$ and $\Delta_n^0 \subset \Delta_{n+1}^0$.
- For all $n$, $\Sigma_n^0 \subset \Pi_{n+1}^0$ and $\Pi_n^0 \subset \Sigma_{n+1}^0$.
- There are complete problems for every level of the $\Sigma$ and $\Pi$ hierarchies.

What could possibly live at level 3? Consider the following set

$$\mathsf{COF} := \{k \mid \exists x\ \forall y > x\ \exists n\ T(k, y, n)\}.$$

This describes the set of Turing machines that have a *cofinite* domain; recall that "cofinite" means "complement of a finite set." This is a $\Sigma_3^0$ set. What is an example of a $\Pi_3^0$ set? Consider

$$\{k \mid \forall x \exists m \text{ such that } (m > x) \text{ and } \forall n,\ \ \neg T(k, m, n)\}.$$

These are Turing machine that fail to terminate on infinitely many inputs.

This entire collection of sets is called the *arithmetic hierarchy.* Is it all there is? By no means! the world of degrees of unsolvability is immensely complicated. What if we allowed quantification over *functions* rather than just integers? Then we get $\Pi^1$ and $\Sigma^1$ and a whole new hierarchy called the *analytic hierarchy.* These questions are not just arbitrary abstractions. The $\Pi_1^1$ sets correspond to exactly the sets that can be defined by induction; yet another theorem due to Kleene. Natural questions in computer science that live in this hierarchy are matching problems in functional languages, termination problems, and deadlock and other problems in concurrent programming.