Math 29: Turing Reducibility

May 11th, 2022

1 Solving Problems Using Oracles

A set A is **Turing reducible** to B if there is an e such that $\chi_A = \Phi_e^B$. That is, there is an oracle machine which computes the characteristic function of A when using B as an oracle. We denote this with $A \leq_T B$. We often approximate this by saying B computes A. If A does not compute B and and B does not compute A, then we say A and B are **Turing incomparable** and write $A \perp_T B$. If $A \leq_T B$ and $B \leq_T A$, then B and A are **Turing equivalent**.

Notice that $A \leq_1 B$ and $A \leq_m B$ both imply $A \leq_T B$, and $A \equiv B$ implies $A \equiv_T B$. In other words, Turing reducibility encompasses all of the reducibilities we have previously discussed.

The equivalence classes of sets of naturals under $\equiv_T$ are called **Turing degrees**. Note that all of the computable sets form a degree. In fact, if C is computable, $C \leq_T X$ for any X: oracle machines are not required to query the oracle, and contain all of the features of regular machines, so since $\chi_C = \varphi_k$, there is some e such that $\chi_C = \Phi_e^X$ for any X which simply does not use the oracle.

We can finally state Post's Problem formally for the first time: Is there a c.e. set P such that $K \not\leq_T P$? So far, we have proven that creative sets, c.e. index sets, and effectively simple sets all do not satisfy this requirement. Fortunately, now that we have the concept of an oracle machine, we can build examples.

Theorem 1. *(Friedberg-Muchnik Theorem) There are Turing incomparable c.e. sets A and B.*

Notice that this solves Post's Problem, as both must be Turing below K but neither can compute K, as it would then compute the other. Similarly, neither can be computable because the other would then be able to compute it.

Our previous constructions used **no injury**: once a requirement has been met, we never have to worry about it again.

Now we will turn to **finite injury.** Here, requirements may be **injured**, i.e. something we have done previously to meet them is changed and they become unmet again. However, so long as they are only injured finitely often, eventually they will not be injured again, so we can be assured that all remain met at the end of the construction.

Proof: We work with the following requirements:

$$R_e = A \neq \Phi_e^B$$

$$Q_e = B \neq \Phi_e^A$$

R_e has higher priority than R_i or Q_i if and only if $e < i$, and R_e has higher priority than Q_e. That is, the priority of requirements is ordered as follows:

$$R_0 > Q_0 > R_1 > Q_1 > R_2 > \dots$$

We first describe the basic strategy for meeting a single requirement. Consider R_e. The strategy for Q_e will be essentially the same for B instead of A.

Pick x for R_e such that x has not been enumerated into A_s. Now wait until we see $\Phi_{e,s}^{B_s \upharpoonright u}(x) \downarrow = 0$, where u is the use of this computation. Notice that if this does not happen, then R_e is satisfied: $x \notin A$, but $\Phi_e^B(x)$ either diverges or doesn't converge to 0, so in either case $\Phi_e^B \neq \chi_A$. Therefore, if we see convergence to 0 at stage s, enumerate x into A_{s+1}. Now, if nothing else changes, the requirement is met: $x \in A$, but $\Phi_e^B(x) \downarrow = 0$, so $\Phi_e^B \neq \chi_A$.

This works in isolation, but the problem is that we cannot ensure that things will remain unchanged. As other requirements are acting in parallel, elements are being enumerated into B and A. Another requirement might put x in A despite $\Phi_e^B(x) \downarrow = 1$, forcing us to start over and pick a new x. Q requirements are enumerating elements into B, so the computation $\Phi_e^{B_s \upharpoonright u}(x)$ may change if elements are added below u. So we may have enumerated x to beat the computation, but then have the computation switch to 1, therefore un-meeting the requirement. This is an **injury**: even if we thought we had satisfied the requirement, we now need to go back and satisfy it again.

The solution to this is a **restraint function.** This will put a lower bound on which numbers can be enumerated into A and B for each requirements. If the restraint is higher than u and x, then we can be assured that the computation won't break because B will remain unchanged below the use.

Proof: (Cont.) We now describe the formal construction, then verify that it works. We let $\omega^{[y]}$ denote $\{\langle x, y\rangle : x \in \omega\}$. That is, it is the set of codes for pairs whose second elements are y. For each requirement R_e or Q_e, we will pick the potential witness x from $\omega^{[e]}$. This will ensure that x is not enumerated by any other requirement, removing one of our potential conflicts.

We set the restraints $r(e, s) = 0$ and $q(e, s) = 0$ when $s = 0$, and again whenever R_e or Q_e is injured at stage s for positive s. Whenever we have the corresponding restraint positive, the restraint is currently satisfied because this means we have chosen an x, which as discussed above means we win so long as we are not injured. If the restraint is 0, the requirement has either been injured or not acted yet in the first place. Let $A_0 = B_0 = \emptyset$.

For stage s, proceed as follows. Find the least e such that $r(e, s) = 0$ (i.e., the highest priority R_e which has not been satisfied with the following property) such that there is an x in $\omega^{[e]} \setminus A_s$ with $\Phi_{e,s}^{B_s}(x) \downarrow = 0$ and $r(i, s), q(i, s) < x$ for all $i < e$. (Notice that we can check this computably because only finitely many x converge in fewer than s stages by convention, and A_s will be finite, so it cannot contain all of $\omega^{[e]}$.) If there is no such e, then do nothing and continue to stage $s+1$. If there is an e, do the following:

- Set $A_{s+1} = A_s \cup \{x\}$.
- Set $r(e, s + 1) = s + 1$. This preserves the computation $\Phi_{e,s}^{B_s}$, as the use is certainly less than or equal to s.
- For all $j > e$, define $r(j, s+1) = q(j, s+1) = 0$. That is, we injure all lower priority requirements.
- For all $i < e$, define $r(i, s + 1) = r(i, s)$ and $q(i, s + 1) = q(i, s)$. (In other words, continue restraining based on higher priority requirements.)

To conclude the stage, repeat the process for the Q requirements with A and B swapped. Finally, continue to stage $s + 1$.

This completes the construction, so we let $A = \bigcup A_s$ and $B = \bigcup B_s$. We now need to check that $A \perp_T B$.

The following lemmas are, suitably, similar for Q requirements with B.

Lemma 2. *If a requirement R_e acts at stage s, and isn't injured later, then it is met and $r(e,t) = r(e,s)$ for all $t \geq s$.*

Proof: By our construction, if R_e acts at stage s, then $\Phi_{e,s}^{B_s}(x) \downarrow = 0$ for some $x \in \omega^{[e]}$. x is then enumerated into A. Since R_e isn't injured at a later stage, $r(e,s) = q(e,s) = s+1$ doesn't change at a later stage, so $r(e,t) = q(e,t) = r(e,s)$. Therefore, no $y \leq s$ is enumerated into B after stage s. Thus $B_s \upharpoonright s = B \upharpoonright s$, so $\Phi_e^B(x) \downarrow = 0 \neq 1 = \chi_A(x)$.

Lemma 3. *For every e, the requirement R_e is met, acts at most finitely often, and $r(e,s)$, $q(e,s)$ stabilizes at some finite stage.*

Proof: We argue by induction on e. Therefore, assume this is true for all R_i, Q_i for $i < e$. Let v be such that all R_i, Q_i do not act after stage v. If $r(e,v) > 0$, we are done. If not, then it is 0. If R_e acts after stage v, then it is met by our previous discussions, and not injured because all of the higher priority requirements do not act again. Then the restraint functions are set when it acts and do not change. If it does not act, then the restraint function is zero forever.

Suppose for the sake of contradiction that R_e is not met, i.e. $\Phi_e^B = \chi_A$. Then it must be the case that R_e does not act after stage v: if it were to act, it would not be injured, and thus it would be met. Therefore there must not be an $x \in \omega^{[e]} \setminus A_s$ for all $s > v$ with $\Phi_{e,s}^{B_s}(x) \downarrow = 0$ and x greater than the higher priority restraints. But this is a contradiction, as A_s contains at most finitely many elements of $\omega^{[e]}$, and $\Phi_e^B(x) \downarrow$ for all of them since it is total. As they are not already in A, they will never be in A since R_e does not act, giving us a contradiction since $\Phi_e^B(x) = \downarrow = \chi_A \neq 0$.

This completes the proof: every requirement is injured at most finitely often, because it is only injured when higher priority arguments act, and they act only finitely often. Once they are finished, R_e and Q_e will be guaranteed to be met by either the next action or by default.