Math 29: Relativization

May 13th, 2022

1 Relativization

A theorem, argument, or proof **relativizes** if it goes through unchanged when we replace "computable" with "X-computable" for an oracle X. Often, we use the term to mean it relativizes to any oracle, but we will occasionally use it when there are limitations on the oracle. In this case, we will list them.

This is best illustrated using an illustration. Recall the proof that the total computable functions are not uniformly computable: there is no single machine which computes all and only the total computable functions. For exactly the same reasoning, the total X-computable functions are not uniformly X-computable. That is, if there were such a machine listing them all, ψ_e^X, define $\rho(n) = \psi_n^X(n) + 1$. Then this will be total, X computable, but cannot be in the list. Therefore, such a listing does not exist.

Often, we will simply state that an argument relativizes when there is no creativity required to modify the proof. For example, the Padding Lemma relativizes simply by replacing computable with X computable. We provide here a relativization of the proof of the s-m-n theorem to illustrate how one would relativize a proof.

Lemma 1. *There is a total computable function $s : \omega^2 \to \omega$ such that*

$$\Phi_e^X(x_0, \dots, x_k) = \Phi_{s(e,x_k)}^X(x_0, \dots, x_{k-1})$$

Proof: Given e and x_k, let $s(e, x_k)$ return the code of the oracle machine M which takes in $x_0, \dots, x_{k-1}$ and starts with x_k-many R_k^+ nodes, then runs U with inputs e, x_0, …, x_k.

s is total computable because our coding of machines is effective, and we can compute its code because our coding of machines is effective.

Theorem 2. *(The s-m-n Theorem) For any n and m, there is a total computable function $s^m_n : \omega^{n+1} \to \omega$ such that, for all e,*

$$\Phi^X_e(y_0, \ldots, y_{m-1}, x_{n-1}, \ldots, x_0) = \Phi^X_{s^m_n(e,x_0,\ldots,x_{n-1})}(y_0, \ldots, y_{m-1})$$

Proof: We argue by induction on n. $n = 1$. Now suppose there is such an s^m_n for all m, and consider $n+1$. Define s^m_{n+1} via $s^m_{n+1}(e, x_0, \ldots, x_n) = s(s^{m+1}_n(e, x_0, \ldots, x_{n-1}), x_n)$. Then

$$\Phi^X_{s^m_{n+1}(e,x_0,\ldots,x_n)}(y_0, \ldots, y_{m-1}) = \Phi^X_{s(s^{m+1}_n(e,x_0,\ldots,x_{n-1}),x_n)}(y_0, \ldots, y_{m-1})$$

By the previous lemma, this is equal to

$$\Phi^X_{s^{m+1}_n(e,x_0,\ldots,x_{n-1})}(y_0, \ldots, y_{m-1}, x_n)$$

By the induction hypothesis, then, this is equal to

$$\Phi^X_e(y_0, \ldots, y_{m-1}, x_n, \ldots, x_0)$$

as desired. Thus we have shown the theorem for all n by induction, and the code for s^m_n can be computed because our coding of machines is effective.

Notice that the function in the s-m-n theorem is computable. If we were to replace every instance of computable with X-computable, it would remain true, but we prove the stronger statement that the function is computable. The exact statement after relativization can sometimes be subtle.

All of our core theorems about working with machines relativize.

Theorem 3. *Every total X-computable function $f : \omega \to \omega$ has an X-fixed point. Moreover, given an X-index for f, we can uniformly compute its fixed point.*

Proof: Consider the partial X-computable function g such that $g(x,y) = \Phi^X_{f(\Phi^X_x(x))}(y)$. By the s-m-n theorem, there is a total computable function s such that

$$\Phi^X_{f(\Phi^X_x(x))}(y) = \Phi^X_{s(x)}(y)$$

Then let m be an oracle machine index for s, which we can compute using an X-index for f. (That is, an index for an oracle machine which computes s without accessing the oracle. This exists because s is computable.)

Now notice that $\Phi^X_m(m)\downarrow$, as $\Phi^X_m(m) = s(m)$, and s is total. Thus

$$\Phi^X_{s(m)}(y) = \Phi^X_{\Phi^X_m(m)}(y) = \Phi^X_{f(\Phi^X_m(m))}(y)$$

Then $s(m) = \Phi^X_m(m) = \Phi^{\emptyset}_m(m)$ is a fixed point of f.

Notice that, of all possible relativizations, the strongest possible one is true. The fixed point is in terms of oracle machines with X as the oracle. It is true for all total X-computable functions, not just the computable ones. But, we do not need X in order to find the fixed point! It is enough to know which machine X uses to compute f, which is information we do not need X to have.

Most theorems relativize in some fashion, once you find the right way to combine things. For example, the Friedberg-Muchnik Theorem relativizes as follows: given an oracle X, there are X-c.e. sets A and B such that $A \perp^X_T B$, that is $A \not\leq^X_T B$ and $B \not\leq^X_T A$. Here Turing reducibility relativizes in a subtle way: $B \leq^X_T A$ if there is an e such that $\chi_B = \Phi^{A\oplus X}_e$. That is, we feed the oracle information for both X and A.

It is important to note that not all results relativize. Depending on what facts about the computable sets we use, they may no longer be true if we replace computable with X-computable. We are not equipped with the terminology to talk about some of the more common examples, but the following is certainly of interest. (Note, this involves relativization using computable oracles, but a restricted notion of machine.)

Theorem 4. *(Baker, Gill, Solovay) There are computable sets A and B such that $P^A = NP^A$ but $P^B \neq NP^B$*

In other words, any solution to the P vs NP problem must use some technique which does not relativize even to computable sets.