Oracle Computations
Proofs of Claims

This document includes details of proofs that were described in the lecture slides.

Relativized Universality Theorem

Recall that, for $n \geq 1$ $x_1, x_2, \ldots, x_n, x_{n+1} \in \mathbb{N}$, and a total function $G : \mathbb{N} \to \mathbb{N}$,

$$\Phi_G^{(n)}(x_1, x_2, \ldots, x_n, x_{n+1})$$

is the value produced as output when the $\mathcal{S}$-program with an oracle $\mathcal{P}$ such that $\#(\mathcal{P}) = x_{n+1}$ is executed, using G as an oracle, on inputs $x_1, x_2, \ldots, x_n$. This value is undefined if this G-computation does not terminate. The following was presented as "Theorem #8" in the lecture notes.

Relativized Universality Theorem: Let $G : \mathbb{N} \to \mathbb{N}$ be a total function and let $n \in \mathbb{N}$ such that $n \geq 1$. Then the program

$$\Phi_G^{(n)} : \mathbb{N}^{n+1} \to \mathbb{N}$$

is G-computable.

Proof. Consider the $\mathcal{S}$-program with an oracle that is obtained from the program in Figure 1, on page 2, after macro expansion. A review of the previous notes on encodings and "universality" should confirm that, after the execution of the steps at lines 1–3,

- Z_1 encodes the $\mathcal{S}$-program to be emulated as a Gödel number,
- S encodes the initial values of variables, and
- K is the number of the next instruction to be executed.

1. $Z_1 \leftarrow X_{n+1} + 1$
2. $S \leftarrow \prod_{i=1}^{n} p_{2i}^{X_i}$
3. $K \leftarrow 1$
4. $[C]$ IF $((K = \mathsf{Lt}(Z_1) + 1) \vee (K = 0))$ GOTO F
5. $U \leftarrow r((Z_1)_K)$
6. $P \leftarrow p_{r(U)+1}$
7. IF $\ell(U) = 0$ GOTO O
8. IF $\ell(U) = 1$ GOTO A
9. IF $(S \bmod P \neq 0)$ GOTO N
10. IF $\ell(U) = 2$ GOTO M
11. $K \leftarrow \min_{i \leq \mathsf{Lt}(Z_1)} (\ell((Z_1)_i) + 2 = \ell(U))$
12. GOTO C
13. $[O]$ $W \leftarrow (S)_{r(U)+1}$
14. $B \leftarrow W$
15. $B \leftarrow O(B)$
16. $S \leftarrow (S/P^W) \times P^B$
17. GOTO N
18. $[M]$ $S \leftarrow S/P$
19. GOTO N
20. $[A]$ $S \leftarrow S \times P$
21. $[N]$ $K \leftarrow K + 1$
22. GOTO C
23. $[F]$ $Y \leftarrow (S)_1$

Figure 1: An $\mathcal{S}$-Program with an Oracle

Following this, the test at line 4 is passed and execution continues at the statement with label F if (and only if) the program being emulated is about to halt. Otherwise K is the number of the instruction to be executed next.

Now, if the emulated program is not about to halt and if $\#(\mathcal{I}) = \langle a, \langle b, c\rangle\rangle$, where $\mathcal{I}$ is the instruction to be executed next, then (after step 5) $U = \langle b, c\rangle$. In this case P is assigned value p_i (the i^{th} prime) where $i = \#(V)$ for the variable V mentioned in the instruction to be executed next.

Recall that $\ell(U)$ encodes the *type* of the instruction $\mathcal{I}$ that is to be executed next. Thus the program continues, with the steps at lines 7–12, by checking the value of $\ell(U)$ in order to determine the type of the instruction to be executed next and branching accordingly:

- Execution continues with the statement with label O if the next instruction to be emulated is an oracle instruction.
- Control branches to the statement with label A if the next instruction is $V \leftarrow V + 1$ for some variable V.
- The text at line 9 is passed if and only if the variable V, included in the instruction $\mathcal{I}$, has value 0 — so that if the next instruction is either "IF $V \neq 0$ GOTO L" or "$V \leftarrow V - 1$," but the current value of the variable V is zero, then this is detected at line 9 and control branches to the statement with label N.
- If the next instruction is "$V \leftarrow V - 1$" and the current value of V is positive then execution continues with the statement with label M.
- In the remaining case — the next statement is

  $$\text{IF } V \neq 0 \text{ GOTO } L$$

  and the current value of V is positive — then (using "bounded minimization") the number of the next instruction to be emulated is calculated at line 11, and the execution continues with the statement at line 4 — so that emulation of the next instruction can begin.

If the next instruction to be emulated is an oracle statement then execution continues with the step (with label O) at line 13. After reviewing Gödel numbers and the encodings of programs as needed, it can be confirmed that step 13 sets W to have the current value of the variable V mentioned in the oracle statement. The oracle query at step 15 sets B to have the new value for V (or fails to halt if this value is undefined).Thus, if it is reached and executed, the step at line 16 updates the encoding S appropriately — replacing the old value for V with the new one. Control then branches to the statement with label N to finish things off.

If the next instruction to be executed is an instruction $V \leftarrow V + 1$ then the statement at line 20 (with label A) is reached, instead, and used to update the value of S accordingly. The statement at line 21 (with label N) is reached and used to finish things off, by updating the value of K, after that. Control is passed back to the statement at line 4 so the execution of the next instruction can begin.

The statement at line 18(with label M) is used to update S accordingly if the instruction being emulated was "$V \leftarrow V - 1$" and the current value of V was positive. Once again, the statement at line 21 (with label N) is reached after that. Remaining instructions update K and return control to the statement at line 4, so that emulation of the next instruction can begin.

The remaining cases are that $\mathcal{I}$ is either "$V \leftarrow V - 1$ or "IF V GOTO L" when $V = 0$. In these cases, control is passed directly to the statement at line 21, so that K can be updated and the emulation of the next instruction can begin.

The statement at line 23 is reached if and only if the emulated execution halts — at which point the output variable Y is set (using S) to have the value that should be returned. A consideration

of the details of this program thus confirms that it G-computes the partial function $\Phi_G^{(n)}$, as needed to prove the theorem. □

Relativized Step Counter Theorem

Now let $G : \mathbb{N} \to \mathbb{N}$ be a ***partial*** (or total) function. Let $n \geq 1$ and consider the total predicate

$$\mathsf{halted}_G^{(n)} : \mathbb{N}^{n+2} \to \{0, 1\}$$

such that, for all $x_1, x_2, \ldots, x_n, y, t \in \mathbb{N}$,

$$\mathsf{halted}_G^{(n)}(x_1, x_2, \ldots, x_n, y, t) = 1$$

if and only if there is a G-computation of the program with number y, beginning with inputs $x_1, x_2, \ldots, x_n$, whose length is at most $t+1$. This condition is ***not*** satisfied — so the predicate has value 0 — if the program with number y *crashes* when executed on inputs $x_1, x_2, \ldots, x_n$ after taking at most $t + 1$ steps.

The following result was stated as "Theorem #9" in the lecture. Note the requirement that G is a ***total*** function, here.

Relativized Step Counter Theorem: For any ***total function*** $G : \mathbb{N} \to \mathbb{N}$, and every number $n \geq 1$, the above predicate $\mathsf{halted}_G^{(n)}$ is G-computable.

Sketch of Proof. The $\mathcal{S}$-program with an oracle, described in the proof of the previous theorem and shown in Figure 1 on page 2, should be updated by including a "step counter" which keeps track of the number of statements emulated so far. This will be used to ensure that the required output is 1 if it really is, or to terminate the emulation and return 0 as soon as it has been confirmed that this is the case.

Steps 1–3 of the previous program should not be changed. As before, these ensure that Z_1 encodes the $\mathcal{S}$ to be emulated as a Gödel number, S encodes the initial values of variables, and K is the number of the next instruction to be executed.

Step 4,

$$[C] \quad \text{IF } ((K = \mathsf{Lt}(Z_1) + 1) \vee (K = 0)) \text{ GOTO } F$$

should be replaced by a block of statements that initializes and maintains the step counter, and uses it to terminate the emulation (if appropriate), while updating the program counter as before:

$$
\begin{array}{ll}
 & Q \leftarrow 0 \\
[C] & Q \leftarrow Q + 1 \\
 & \mathsf{IF}\ (Q > X_{n+2} + 1)\ \mathsf{GOTO}\ E \\
 & \mathsf{IF}\ ((K = \mathsf{Lt}(Z_1) + 1) \vee (K = 0))\ \mathsf{GOTO}\ F
\end{array}
$$

Steps 5–22 of the program (which carry out most of the rest of the emulation) are the same as before.

Finally, step 23 should be changed from

$$[F] \quad Y \leftarrow (S)_1$$

to

$$[F] \quad Y \leftarrow 1$$

in order to ensure that the value of the predicate is returned if this statement is reached.

Note that if step 23 is *not* reached then execution of this program terminated because the test in the statement *after* the one with label C (shown above) succeeded — so that the value 0 was correctly returned, because the emulated computation did not halt within the required number of steps. □

Finiteness Theorem

For each number $u \in \mathbb{N}$ consider the partial function $\{u\} : \mathbb{N} \to \mathbb{N}$ such that, for all $i \in \mathbb{N}$,

- if $i < \ell(u)$ then $\{u\}(i) = (r(u))_{i+1}$, and
- if $i \geq \ell(u)$ then $\{u\}(i)$ is undefined.

This definition implies the following.

- If $\ell(u) = 0$ then $\{u\}(i)$ is undefined for all $i \in \mathbb{N}$.
- If k is positive and
$$u = \langle k, [a_0, a_1, \ldots, a_{k-1}] \rangle$$
then $\{u\}(i) = a_i$ for $0 \leq i \leq k - 1$ and $\{u\}(i)$ is undefined for all $i \in \mathbb{N}$ such that $i \geq k$.

The next result was given as “Theorem #10” in the lecture note.

Lemma: The total predicate $P^{n+3} \rightarrow \{0,1\}$ such that, for all $x_1, x_2, \ldots, x_n, y, t, u \in \mathbb{N}$, $P(x_1, x_2, \ldots, x_n, y, t, u) = 1$ if and only if

$$\text{halted}^{(n)}_{\{u\}}(x_1, x_2, \ldots, x_n, y, t) = 1$$

is computable.

Proof. It suffices to modify the program in the proof of the Relativized Step Counter Theorem, above, in order to produce a regular $\mathcal{S}$-program that computes the above predicate. Since that program contained a single oracle statement it suffices to replace this with a pair of statements that evaluate $\{u\}$, where u is the current value of the input variable X_{n+3}, instead.

Suppose, in particular, that the oracle statement "$B \leftarrow O(B)$" is replaced by the following instead.

IF $\ell(X_{n+3}) \leq B$ GOTO E
$B \leftarrow (r(X_{n+3}))_{B+1}$

An examination of the definition of $\{u\}$ should confirm that this correctly returns output 0 if the desired value of $\{u\}$ is undefined (so that the computation would not halt) and that it returns the desired value, $\{u\}(B)$, as needed, if it *is* defined. □

Once again let $G : \mathbb{N} \rightarrow \mathbb{N}$ be a total function. Let us write

$$u \prec G$$

to mean that $\{u\}(i) = G(i)$ for every integer i such that $0 \leq i < \ell(u)$. The following result is given as "Theorem #11" in the lecture note.

Finiteness Theorem: Let G be a total function. Then, for every integer $n \geq 1$ and for all numbers $x_1, x_2, \ldots, x_n, y, t \in \mathbb{N}$,

$$\text{halted}^{(n)}_{G}(x_1, x_2, \ldots, x_n, y, t) = 1$$

if and only if there exists a number $u \in \mathbb{N}$ such that

$$u \prec G \quad \text{and} \quad \text{halted}^{(n)}_{\{u\}}(x_1, x_2, \ldots, x_n, y, t) = 1.$$

Proof. Suppose, first, that $\text{halted}^{(n)}_{G}(x_1, x_2, \ldots, x_n, y, t) = 1$ for $x_1, x_2, \ldots, x_n, y, t \in \mathbb{N}$. Let $\mathcal{P}$ be the $\mathcal{S}$-program with an oracle such that $\#(\mathcal{P}) = y$. Then there is a sequence of "snapshots"

$$\mathcal{S}_0, \mathcal{S}_1, \ldots, \mathcal{S}_k$$

such that $\mathcal{S}_1$ is the initial snapshot for the inputs $x_1, x_2, \ldots, x_n$, $\mathcal{S}_{i+1}$ is the G-successor of $\mathcal{S}_i$ for $0 \leq i \leq k-1$, $\mathcal{S}_k$ is terminal, and $k \leq t+1$.

Let M be the largest input value of any oracle query in this computation, and let

$$u = \langle M+1, [G(0), G(1), G(2), \ldots, G(M)] \rangle.$$

Then $u \prec G$ and $\{u\}(m) = G(m)$ for every number $m \in \mathbb{N}$ such that $m \leq M$.

This can be used to show that

$$\mathcal{S}_0, \mathcal{S}_1, \ldots, \mathcal{S}_k$$

is also a $\{u\}$-computation of $\mathcal{P}$ (just as it is a G-computation of $\mathcal{P}$). It follows that

$$\mathsf{halted}^{(n)}_{\{u\}}(x_1, x_2, \ldots, x_n, y, t) = 1,$$

as desired.

Suppose, instead, that there is a number $u \in \mathbb{N}$ such that $u \prec G$ and

$$\mathsf{halted}^{(n)}_{\{u\}}(x_1, x_2, \ldots, x_n, y, t) = 1.$$

Then there is a sequence of "snapshots"

$$\mathcal{S}_0, \mathcal{S}_1, \ldots, \mathcal{S}_k$$

such that $\mathcal{S}_1$ is the initial snapshot for the inputs $x_1, x_2, \ldots, x_n$, $\mathcal{S}_{i+1}$ is the $\{u\}$-successor of $\mathcal{S}_i$ for $0 \leq i \leq k-1$, $\mathcal{S}_k$ is terminal, and $k \leq t+1$.

Since this computation halts (after $k \leq t+1$ steps) $\{u\}(m)$ is defined for every number $m \in \mathbb{N}$ such the computation includes an oracle query to evaluate $\{u\}(m)$. However, since $u \prec G$ it follows that $\{u\}(m) = G(m)$ for every such number m. It follows from this that

$$\mathcal{S}_0, \mathcal{S}_1, \ldots, \mathcal{S}_k$$

is also a G-computation of $\mathcal{P}$. This implies that $\mathsf{halted}^{(n)}_{G}(x_1, x_2, \ldots, x_n, y, t) = 1$, as required to complete the proof. □