Chapter 14

Cubbyhole Mathematics

The arithmetical hierarchy provides us with lots of cubbyholes. Recursion theory seeks understanding of objects by putting them in the right holes. For when we do so, we know what the "main part" of the problem is. Problems within the class will be computationally trivial variants of one another.

Here are some examples. Some of them are familiar from chapter 6.

$$
\begin{array}{rcl}
Tot(x) & \iff & W_w = \mathbf{N} \\
Null(x) & \iff & W_w = \emptyset \\
Inf(x) & \iff & W_w \text{ is infinite} \\
Fin(x) & \iff & W_w \text{ is finite} \\
Cof(x) & \iff & W_w \text{ is co-finite} \\
Onto(x) & \iff & E_x = \mathbf{N} \\
Simp(x) & \iff & W_w \text{ is simple} \\
Creative(x) & \iff & \overline{W_x} \text{ is productive} \\
Subset(x) & \iff & W_{(x)_0} \subseteq W_{(x)_1} \\
Psubset(x) & \iff & W_{(x)_0} \subset W_{(x)_1} \\
Ident(x) & \iff & W_{(x)_0} = W_{(x)_1} \\
Rec(x) & \iff & W_w \text{ is recursive} \\
Comp(x) & \iff & W_w \equiv K
\end{array}
$$

14.1 Upper Bounds

Upper bounds are easily found by the *Tarski-Kuratowski algorithm*:

1. Define the relation in terms of the Kleene predicate.
2. Put the definition into prenex normal form.
   (a) First eliminate arrows using conjunctions, disjunctions and negations.
   (b) Then drive in all negations using DeMorgan's rules.
   (c) Then rename quantified variables so that they are all distinct in order to prevent clashes when the quantifiers are exported.
   (d) Then export quantifiers to the front of the formula in the most advantageous way (i.e., interleave them to minimize alternations without permuting the order of any quantifiers that were already nested.
3. The first quantifier determines whether the complexity class is Σ or Π.
4. The number of blocks of quantifiers of the same type determines the subscript.

14.1.1 Examples

$$\begin{aligned} Tot(x) &\iff W_x = \mathbf{N} \\ &\iff \forall z\ W_x(z) \\ &\iff \forall z\ \exists w\ U(x, (w)_0, (w)_1, \langle z \rangle) \\ &\iff \forall \exists R \text{ with R recursive.} \end{aligned}$$

So $\Pi_2(Tot)$.

$$\begin{aligned} Fin(x) &\iff W_x \text{ is finite} \\ &\iff \exists y\ \forall_{z \le y}\ \overline{W_x}(z) \\ &\iff \exists y\ \forall_{z \le y}\ \phi_x(z) \uparrow \\ &\iff \exists y\ \forall_{z \le y}\ \forall w\ \neg U(x, (w)_0, (w)_1, \langle z \rangle) \\ &\iff \exists \forall R \text{ with R recursive.} \end{aligned}$$

So $\Sigma_2(Fin)$.

These were automatic! Let's do one that requires a little bit of shuffling.

$$
\begin{aligned}
Ident(x) &\iff W_{(x)_0} = W_{(x)_1} \\
&\iff \forall z\ (W_{(x)_0}(z) \leftrightarrow W_{(x)_1}(z)) \\
&\iff \forall z\ \big((W_{(x)_0}(z) \wedge W_{(x)_1}(z)) \vee (\overline{W_{(x)_0}}(z) \wedge \overline{W_{(x)_1}}(z))\big) \\
&\iff \forall z\ \big[(\exists w\ U((x)_0,(w)_0,(w)_1,\langle z\rangle) \\
&\qquad \wedge \exists w\ U((x)_1,(w)_0,(w)_1,\langle z\rangle)) \\
&\qquad \vee (\forall w\ \neg U((x)_0,(w)_0,(w)_1,\langle z\rangle) \\
&\qquad \wedge \forall w\ \neg U((x)_1,(w)_0,(w)_1,\langle z\rangle))\big] \\
&\iff \forall\big((\exists \wedge \exists) \vee (\forall \wedge \forall)\big) \text{ (notice, the lead } \exists \text{ makes it most} \\
&\qquad\qquad \text{efficient to put all the } \exists \text{ quantifiers first).} \\
&\iff \forall\forall\forall\exists\exists
\end{aligned}
$$

So $\Pi_2(Ident)$.

Here is a more complicated one that makes use of work already done.

$$
\begin{aligned}
Simp(x) &\iff W_x \text{ is simple} \\
&\iff W_x \text{ is infinite } \wedge \forall y\ (W_y \text{ is infinite } \rightarrow W_x \cap W_y \neq \emptyset \\
&\iff Inf(x) \wedge \forall y\ (Inf(y) \rightarrow \forall z\ (\overline{W_x}(z) \vee \overline{W_y}(z))) \\
&\iff Inf(x) \wedge \forall y\ \big[\neg Inf(y) \vee \forall z \\
&\qquad (\forall w\ \neg U(x,(w)_0,(w)_1,\langle z\rangle) \vee \forall w\ \neg U(y,(w)_0,(w)_1,\langle z\rangle))\big] \\
&\iff \forall\exists \wedge \forall(\neg\forall\exists \vee \forall(\forall \vee \forall)) \\
&\iff \forall\exists \wedge \forall(\exists\forall \vee \forall(\forall \vee \forall)) \\
&\iff \forall\exists \wedge \forall(\exists\forall \vee \forall\forall\forall)) \\
&\iff \forall\exists \wedge \forall(\exists\forall\forall\forall\forall) \\
&\iff \forall\exists \wedge \forall\exists\forall\forall\forall \\
&\iff \forall\forall\exists\exists\forall\forall\forall \\
&\iff \exists\forall\exists
\end{aligned}
$$

So $\Pi_3(Simp)$.

Exercise 14.1 *Do five more examples. Include Comp.*

14.2 Lower Bounds

Lower bounds come by several techniques: diagonalization, reduction, or direct completeness arguments. We have already seen two ways to do diagonalization

in the last chapter, providing us with "seed" for reduction arguments. Let's begin with a direct completeness argument.

Proposition 14.1 *Fin is* Σ_2*-complete,* Inf *is* Π_2*-complete.*

Proof: suppose $\Sigma_2(P)$. So for some recursive R, we have for each x,

$$P(x) \iff \exists y\ \forall z\ R(\langle x, y, z\rangle)$$

Define

$$\psi(x, w) = \forall_{y \leq w}\ \exists z\ \overline{R}(\langle x, y, z\rangle)$$

This is partial recursive by the projection theorem. Apply the *s-m-n* theorem to obtain

$$\phi_{f(x)}(w) \approx \psi(x, w)$$

Hence,

$$\begin{array}{rcl} \overline{P}(x) & \Rightarrow & W_{f(x)} \text{ is total} \\ & \Rightarrow & Tot(x) \\ & \Rightarrow & Inf(x) \\ P(x) & \Rightarrow & W_{f(x)} \text{ is finite} \\ & \Rightarrow & Fin(x) \end{array} \qquad \dashv$$

Notice that the reduction shows more than we intended. It projects $\overline{P}$ into Tot and P into Fin. Following Soare, we may summarize this situation by the notation:

$$(P, \overline{P}) \leq_M (Fin, Tot)$$

When P stands for an arbitrary Σ_2 set, we may abbreviate the situation by writing

$$(\Sigma_2, \Pi_2) \leq_M (Fin, Tot)$$

Since Fin is in the complement of Tot, this reduction also establishes:

Corollary 14.2 *Tot is* Π_2*-complete.*

Proposition 14.3 *Subset is* Π_2*-complete.*

Exercise 14.2 *Prove the upper bound by Tarski-Kuratowski computation. Prove the lower bound by reduction of* Tot *or* Inf*.*
Hint: make the "subset" index be for $\mathbf{N}$ *and make the "superset" index depend on the given number.*

Exercise 14.3 *Peg the complexity of Onto in the hierarchy.*
No hints this time.

At the next level of complexity lower bounds become more complex.

Proposition 14.4 $(\Sigma_3, \Pi_3) \leq_M (Cof, Comp) \leq_M (Rec, Comp)$.

Corollary 14.5 *Cof, Rec are Σ_3-complete.*

Proof: suppose $\Sigma_3(P)$. So for some Σ_3 set R, we have for each x,

$$P(x) \iff \exists y\ R(\langle x, y\rangle)$$

We have already shown that there exists a total recursive g such that

$$R(\langle x, y\rangle) \iff W_{f(\langle x,y\rangle)} \text{ is infinite.}$$

Hence

$$\begin{aligned} P(x) &\iff \exists y\ R(\langle x, y\rangle)) \\ &\iff \exists y\ W_{f(\langle x,y\rangle)} \text{ is infinite.} \end{aligned}$$

We need to construct total recursive f such that

$$P(x) \Rightarrow W_{g(x)} \text{ is cofinite and}$$

$$\overline{P}(x) \Rightarrow W_{g(x)} \equiv K$$

I sketch the construction, showing how to enumerate $S = W_{g(x)}$ as a function of x:

- We start out with an array of "pointers" on the natural numbers labelled with the natural numbers.
- Let $pointer_x(y, s)$ be the number pointed to by the pointer labelled with y at stage s.
- At stage $s = 0$, the y^{th} pointer is initialized to point to number y: i.e., $pointer_x(y, 0) = y$.
- At stage $s+1$: Check for each $y \leq s$ whether either of the following occur:
  - $K(y)$ halts in exactly s steps (this can happen only once) or
  - $\exists_{w \leq s}\ W_{f(\langle x,y\rangle)}(w)$ halts in exactly s steps (this happens infinitely often for some y just in case $P(x)$).
- For each such y, add $pointer_x(y, s)$ to S.
- Now move all pointers to the right, without permuting them, so that positions already added to S are not pointed to, but without leaving any other gaps.

$$\begin{array}{rcl} P(x) & \Rightarrow & \exists y\ W_{f(\langle x,y\rangle)} \text{ is infinite} \\ & \Rightarrow & \exists y\ \lim_s pointer_x(y,s) = w \\ & \Rightarrow & S = W_{g(x)} \text{ is cofinite} \\ & \Rightarrow & Cof(g(x)) \\ \overline{P}(x) & \Rightarrow & \forall y\ W_{f(\langle x,y\rangle)} \text{ is finite} \\ & \Rightarrow & \forall y\ \lim_s pointer_x(y,s) \text{ is finite} \end{array}$$

Hence each terminal pointer position is missing from S.

Using the fact that each pointer bumps only finitely often, we can compute, given S, the least stage $s(y)$ such that

$$pointer(y, s(y)) = \lim_s pointer_x(y, s)$$

In other words, s is recursive in S.

To decide whether $K(y)$, check whether $pointer_x(y, s)$ ever bumped (prior to stage $s(y)$) when nothing new came out of the $W_{f(\langle x,y\rangle)}$ simulation.

In other words, the convergence of the pointer, which is recoverable from S, bounds the search through the reasons for adding y to S.

Hence, $S = W_{g(x)} \equiv K$, so $Comp(g(x))$.

Apply Church's thesis!

So recursiveness is worse than finiteness or infinity, but is the same as co-finiteness. Non-recursiveness is therefore the same as co-infinity.

How about $Comp$? The obvious Tarski-Kuratowski computation of $Comp$ takes us only down Σ_4, unfortunately, so our previous reduction doesn't lock in the complexity of $Comp$. Yates showed that $Comp$ is Σ_4-complete. He proves it via his "thickness" lemma, which is proved by an infinite injury argument.

Exercise 14.4 *This is from Soare, ex. 3.8. Define*

$$Ext(x) \iff \phi_x \textit{ is extendable to a total recursive function}$$

Show that Ext is Σ_3-complete.
Hint: use the preceding technique. Instead of building an r.e. set, $W_{g(x)}$, build a partial recursive function $\phi_{g(x)}$. When $W_{f(\langle x,y\rangle)}(w)$ halts in exactly s steps, define $\phi_{g(x)}(pointer_x(y,s))$ to be some value (e.g., 0). Also, if $\phi_y(pointer_x(y,s))$ is observed to halt in s steps, define $\phi_{g(x)}(pointer_x(y,s))$ to have a value different from $\phi_y(pointer_x(y,s))$. The resulting function is guaranteed to be partial recursive. If some $W_{f(\langle x,y\rangle)}$ is infinite, the function will itself be total recursive. If no $W_{f(\langle x,y\rangle)}$ is infinite, the function differs somewhere from each total recursive function. (This takes some arguing: how do you know that the y pointer sits around long enough to ensure that the constructed function differs from ϕ_y?)

14.3 Arithmetical Truth

Let A = the set of all Gödel numbers of true sentences of arithmetic.

Gödel's incompleteness construction shows only that A is productive. As we have seen, productive sets are merely co-r.e.-hard (i.e., non-r.e. and non-co-r.e.-incomplete. This is fairly weak, since it is consistent with A being co-r.e. If A were co-r.e., there would be an effective refutation procedure for A, which would tell us for each non-truth that it is false. A little thought reveals, however, that this is false.

In fact, A is not even arithmetical (i.e., it diagonalizes the whole hierarchy). We will see this as follows. Recall from the Gödel and uncomputability class that Q is a weak, incomplete system of arithmetic.

A relation $R(\vec{x})$ is representable in Q just in case there exists a formula $\Phi(\vec{x})$ in the language of arithmetic such that for all $\mathbf{b}$,

1. $R(\mathbf{b}) \Rightarrow Q \vdash \Phi(\mathbf{b})$
2. $\overline{R}(\mathbf{b}) \Rightarrow Q \vdash \neg\Phi(\mathbf{b})$

The main lemma *en route* to Gödel's incompleteness theorem is the representation theorem, which we will not reprove here:

Proposition 14.6 ***(Gödel)*** *Each recursive relation is representable in Q.*

This implies that each r.e. predicate has a purely existential definition in the language of Arithmetic.

Proposition 14.7 *A is not arithmetical.*

Proof: let $\Sigma_n(P)$. Then for some recursive relation R, we have

$$P(x) \iff \exists y_1 \ldots \exists y_n \; R(x, y_1, \ldots, y_n)$$

Using arithmetical representability, choose formula Φ representing R. Then:

$$\begin{array}{lcl} P(x) & \iff & \exists y_1 \ldots \exists y_n \; \Phi(x, y_1, \ldots, y_n) \\ & \iff & \text{the sentence ``}\exists y_1 \ldots \exists y_n \; \Phi(x, y_1, \ldots, y_n)\text{'' is true in arithmetic} \\ & \iff & A(\langle \exists y_1 \ldots \exists y_n \; \Phi(x, y_1, \ldots, y_n)\rangle) \end{array}$$

By Church's thesis, there is a total recursive f such that for all x,

$$f(x) = \langle \exists y_1 \ldots \exists y_n \; \Phi(x, y_1, \ldots, y_n)\rangle$$

Thus $P \leq_M A$.

Hence, for each n, $0^{(n+1)} \leq_M A$.

So by Post's theorem, for each n, $\neg\Sigma_n(A)$. $\dashv$

We can say more than this. We know what $S^{(n)}$ means. What should the limit jump $S^{(\omega)}$ mean? It should somehow reduce all the preceding jumps. The following notion will do:

$$S^{(\omega)}(x) \iff S^{((x)_1)}((x)_0)$$

Now we may state:

Proposition 14.8 $A \equiv 0^{(\omega)}$.

Proof: (Rogers Thm X, p. 318). $\dashv$
A question for us later: if A is not arithmetical, then where does A live?