Chapter 11

The Arithmetical Hierarchy

Think of $\neg K$ as posing the problem of induction for computational devices, for it is impossible to tell for sure whether a given computation will never halt. Thus, K is effectively refutable and $\neg K$ is effectively verifiable. We know from the philosophy of science that universal hypotheses are refutable and existential hypotheses are verifiable. This correspondence also holds, if we think of the expressions of the sets in Kleene normal form. Kleene normal form prenex normal form with U as the only predicate. Thus, we have

$$K(x) \iff \exists z\, U(x, (z)_1, (z)_2, \langle x \rangle)$$

$$\neg K(x) \iff \forall z\, \neg U(x, (z)_1, (z)_2, \langle x \rangle)$$

Thus, you may think of $K(x)$ as an "existential hypothesis" and of $\neg K(x)$ as a "universal hypothesis" given that instances of U are "observable" (e.g., the "scientist" is treating program x as a black box and watching what it does in various numbers of steps of computation on input x.

It is also familiar in the philosophy of science that most hypotheses are neither verifiable nor refutable. Thus, Kant's antinomies of pure reason include such statements as that space is infinite, matter is infinitely divisible, and the series of efficient causes is infinite. These hypotheses all have the form

$$(\forall x)(\exists y)\ \Phi(x, y).$$

For example, infinite divisibility amounts to "for every product of fission, there is a time by which attempts to cut it succeed" and the infinity of space amounts to "for each distance you travel, you can travel farther."

Computers face their own non-verifiable and non-refutable questions:

$$\begin{aligned} Tot(x) &\iff \phi_x \text{ is total} \\ &\iff (\forall w)(\exists z)\ U(x, (z)_1, (z)_2, \langle w \rangle); \\ Inf(x) &\iff W_x \text{ is infinite} \\ &\iff (\forall w)(\exists z)(\exists y > w)\ (U(x, (z)_1, (z)_2, \langle y \rangle)). \end{aligned}$$

Things get worse as the quantifier structure of the hypothesis becomes more complex, where complexity is measured as number of blocks of quantifiers: i.e., $\exists\forall\forall\forall\exists\exists$ counts as three blocks. You can count quantifier alternations as follows.

11.1 The Arithmetical Hierarchy

Let P be a relation over $\mathbf{N}$.

$$\begin{array}{rcl} \Sigma_0(P) & \Longleftrightarrow & P \text{ is recursive;} \\ \Sigma_{n+1}(P) & \Longleftrightarrow & (\exists R \in \Sigma_n)(\forall \vec{x})\ (P(\vec{x}) \Leftrightarrow (\exists y)\ \neg R(\vec{x}, y)); \\ \Pi_n(P) & \Longleftrightarrow & \Sigma_n(\neg P); \\ \Delta_n(P) & \Longleftrightarrow & \Sigma_n(P) \wedge \Pi_n(P); \\ Arith(P) & \Longleftrightarrow & (\exists n)\ \Sigma_n(P). \end{array}$$

Proposition 11.1 *(basic structure and closure laws)*

1. $\Delta_n =$ recursive;
2. $\Delta_n, \Sigma_n, \Pi_n$ are closed downward under $\leq$;
3. Δ_n is closed under $\wedge, \vee, \neg$;
4. Σ_n is closed under $\wedge, \vee, \exists$;
5. $\Pi[A]_n$ is closed under $\wedge, \vee, \forall$.

Exercise 11.1 *Prove it. Hint: use logical rules and some induction on n.*

11.2 The Arithmetical Hierarchy Theorem

We don't know yet whether the whole hierarchy collapses into some finite level. Recall that

$$W_0, W_1, W_2 \ldots$$

is an enumeration of the r.e. sets. and that the halting problem is just

$$\neg K(x) \Longleftrightarrow \neg W_x(x).$$

Wouldn't it be nice if for each n, Σ_n is enumerated by an analogous collection

$$W_0^n, W_1^n, W_2^n \ldots$$

so that when $n = 1$, $W_i = W_i^1$? Then we could define the nth generalization of the halting problem just as before:

$$\neg K^n(x) \Longleftrightarrow \neg W_x^n(x),$$

and then it would be clear that $\neg K^n \notin \Sigma_n$. Accordingly, define

$$\begin{aligned} W_i^1(\langle \vec{y} \rangle) &\iff W_i(\langle \vec{y} \rangle); \\ W_i^{n+1}(\vec{y}) &\iff (\exists y)\ \neg W_i^n(\langle \vec{x}, y \rangle). \end{aligned}$$

The first item is to show that each relation $R(\vec{x})$ in Σ_n is some $W_i^n(\langle \vec{x} \rangle)$.

Proposition 11.2 $(\forall n)(\forall R \in \Sigma_n)(\exists i)(\forall \vec{x})\ (R(\vec{x}) \Leftrightarrow W_i^n(\langle \vec{x} \rangle))$.

Proof: by induction on n.
Base case: immediate from fundamental theorem for r.e. sets.
Induction: let $\Sigma_{n+1}(R)$. Then by the definition of Σ_{n+1},

$$(\exists Q \in \Sigma_n)(\forall \vec{x})\ (R(\vec{x}) \Leftrightarrow (\exists y)\ \neg Q(\vec{x}, y)).$$

By the induction hypothesis,

$$(\exists i)(\forall \vec{x}, y)\ (Q(\vec{x}, y) \Leftrightarrow W_i^n(\langle \vec{x}, y \rangle)).$$

Substituting in the first formula and applying the definition of W_i^{n+1} yield

$$\begin{aligned} (\exists i)(\forall \vec{x})\ (R(\vec{x}) &\iff (\exists y)\ \neg W_i^n(\langle \vec{x}, y \rangle)) \\ &\iff W_i^{n+1}(\langle \vec{x} \rangle). \dashv \end{aligned}$$

Define the **universal relation** for Σ_n as follows:

$$U_n(i, x) \Leftrightarrow W_i^n(x).$$

Then you can show:

Proposition 11.3 $\Sigma_n(U_n)$.

Exercise 11.2 *Prove the preceding proposition by induction on n.*

Now define

$$K_n(x) \iff W_x^n(x).$$

So by the usual, Cantorian diagonal argument, we have that $\neg K_n$ differs from each W_i^n and hence is not Σ_n. On the other hand,

$$K_n(x) \iff U_n(x, x),$$

so since $\Sigma_n(U_n)$, you have $\Sigma_n(K_n)$. Hence,

Proposition 11.4 ***(arithmetical hierarchy theorem)***

$$\Sigma_n(K_n) \quad \textit{and} \quad \neg\Sigma_n(\neg K_n).$$

Exercise 11.3 *Why does the hierarchy theorem imply that $\neg\Sigma_n(K_{n+1})$?*

11.3 Cubbyhole Mathematics

The arithmetical hierarchy provides us with lots of shelves for problems. Recursion theorists are sort of like librarians: they want to locate everything on the right shelf, for all problems on a given shelf share a certain "family resemblance" that determines the best approach to all of them.

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

$$\begin{array}{rcl}
Tot(x) & \iff & W_w = \mathbf{N} \\
Und(x) & \iff & W_w = \emptyset \\
\mathit{Inf}(x) & \iff & W_w \text{ is infinite} \\
Fin(x) & \iff & W_w \text{ is finite} \\
Cof(x) & \iff & W_w \text{ is co-finite} \\
Onto(x) & \iff & E_x = \mathbf{N} \\
Subset(x) & \iff & W_{(x)_0} \subseteq W_{(x)_1} \\
Psubset(x) & \iff & W_{(x)_0} \subset W_{(x)_1} \\
Ident(x) & \iff & W_{(x)_0} = W_{(x)_1} \\
Rec(x) & \iff & W_w \text{ is recursive.}
\end{array}$$

11.4 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.

11.4.1 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 $\Pi_2(Tot)$.

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

So $\Sigma_2(Fin)$.

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

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

So $\Pi_2(Ident)$.

Exercise 11.4 *Do three more examples.*

11.5 Lower Bounds

Lower bounds on arithmetical complexity come by several techniques: diagonalization, reduction, or 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 11.5 *Fin is Σ_2-complete.*

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

$$(\forall \vec{x})\ (P(\vec{x}) \iff (\exists y)(\forall z)\ R(\vec{x}, y, z)).$$

Define

$$\psi(n, w) \simeq (\mu u)(\forall y \leq w)(\exists z)\ \neg R((n)_0, \ldots, (n)_{lh(n)-1}, y, z).$$

This is partial recursive by the projection theorem. Apply the *s-m-n* theorem to obtain total recursive f such that:

$$\phi_{f(n)}(w) \simeq \psi(n, w).$$

Hence,

$$\begin{array}{rcl} \neg P(\vec{x}) & \Rightarrow & (\forall y)(\exists z)\ \neg R(\vec{x}, y, z) \\ & \Rightarrow & \phi_{f(\langle \vec{x} \rangle)} = o \\ & \Rightarrow & W_{f(\langle \vec{x} \rangle)} = \mathbf{N} \\ & \Rightarrow & Tot(\langle \vec{x} \rangle) \\ & \Rightarrow & Inf(\langle \vec{x} \rangle); \\ P(x) & \Rightarrow & W_{f(x)} \text{ is finite} \\ & \Rightarrow & \neg Tot(\langle \vec{x} \rangle) \\ & \Rightarrow & Fin(x). \end{array} \qquad \dashv$$

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

$$(P, \neg P) \leq_m (Fin, Tot).$$

When P stands for an arbitrary Σ_2 set, you 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 11.6 *Tot is Π_2-complete.*

Proposition 11.7 *Subset is Π_2-complete.*

Exercise 11.5 *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 11.6 *Determine the complexity of Onto in the hierarchy. No hints this time.*

At the next level of complexity it is understandably more complicated to establish lower bounds.

Proposition 11.8 $(\Sigma_3, \Pi_3) \leq_M (Cof, \neg Rec)$.

Corollary 11.9 *Cof, Rec are Σ_3-complete.*

Proof: suppose that $\Sigma_3(P)$. So for some Σ_3 relation R, we have for each $\vec{x}$,

$$P(\vec{x}) \iff (\exists y)\ R(\vec{x}, y).$$

It has already been shown that there exists a total recursive f such that

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

Hence

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

We need to construct total recursive g such that

$$\begin{aligned} P(\vec{x}) &\Rightarrow W_{g(\vec{x})} \text{ is cofinite;} \\ \neg P(\vec{x}) &\Rightarrow W_{g(\vec{x})} \equiv_m K. \end{aligned}$$

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

> Let $\vec{x}$ be given. We start out with an infinite sequence of "pointers" on the natural numbers labelled with the natural numbers.
>
> Let $pointer_{\vec{x}}(y, s)$ 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., $pointer_{\vec{x}}(y, 0) = y$.
>
> At stage $s + 1$: check for each $y \leq s$ whether either of the following occur:
>
> A. $\phi_y(y)$ halts in exactly s steps (this can happen only once) or
> B. $(\exists w \leq s)\ \phi_{f(\langle \vec{x}, y\rangle)}(w)$ halts in exactly s steps (this happens infinitely often for some y just in case $P(\vec{x})$).
>
> For each such y, add $pointer_{\vec{x}}(y, s)$ 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.

Case: Suppose $P(\vec{x})$. Then some y satisfies (B) at infinitely many stages. Let y' be the least y such that y satisfies (B) at infinitely many stages. So let s' be the first stage after which no $y < y'$ satisfies (A or B). Let

$$k = pointer_{\vec{x}}(y', s').$$

Then for each $k' \geq k$, pointer y' eventually points at k' and condition (B) is subsequently satisfied, at which point k' is added to S. So $S = W_{g(\langle \vec{x} \rangle)}$ is cofinite and hence $g(\langle \vec{x} \rangle) \in Cof$, as required.

Case: Suppose $\neg P(\vec{x})$. Then for each y, y satisfies (A or B) at only finitely many stages. It remains only to show that $S = W_{g(\langle \vec{x} \rangle)}$ is not recursive. For suppose S were decidable. Then you could decide K as follows. For given n, decide whether n is in S. If not, return 0. If so, then simulate the preceding enumeration of S to determine the stage s at which n was added to S. According to the statement of (A), s provides an upper bound on the time required for $\phi_n(n)$ to halt, so run $\phi_n(n)$ for s steps and see whether it halts by then. If so, return 1. Otherwise, return 0. $\dashv$

Exercise 11.7 *(Bonus question) from Soare, ex. 3.8. Define*

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

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 $\phi_{f(\langle x,y \rangle)}(w)$ 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.

11.6 Arithmetical Truth

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

Gödel's incompleteness construction shows only that A is not r.e. One would like to know more than that. For example, if $\Pi_1(A)$, then it would at least be possible to have a "falsifiable" mathematics analogous to the testing of empirical laws in physics, for an empirical mathematician would have the assurance that each falsehood would eventually be "shot down" by future proofs.

In fact, A is *infinitely* worse than that, for A is not at any level in the arithmetical hierarchy.

Proposition 11.10 $(\forall n)\ \neg\Sigma_n(A)$.

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

$$P(x) \iff (\exists y_1)(\forall y_2)\ldots(\exists y_n)\ R(x, y_1, \ldots, y_n)$$

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

$$\begin{aligned} P(x) &\iff \text{the sentence "}(\exists y_1)\ldots\exists y_n\ \Phi(\mathbf{x}, y_1, \ldots, y_n)\text{" is true in arithmetic} \\ &\iff A(\langle \exists y_1 \ldots \exists y_n\ \Phi(x, y_1, \ldots, y_n)\rangle) \end{aligned}$$

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 $P \leq_m A$. Now choose $P = K_n$. $\dashv$