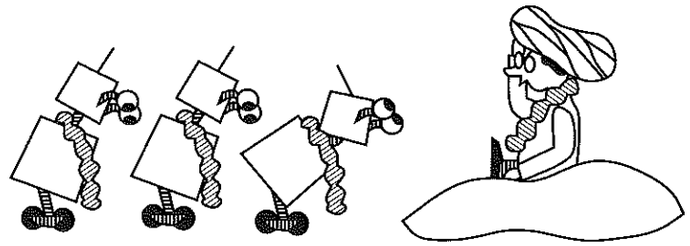
7

Computers in Search of the Truth



1. Ideal Epistemology and Computability

So far, our study of inductive methodology has acceded to the time-honored cult of the *ideal agent*. An ideal agent is a curious sort of deity who is limited in evidence-gathering ability more or less the way we are, but who suffers from no computational limitations whatsoever. An ideal agent can decide all mathematical relations in an instant, but in a dark room he bumps into the walls with the rest of us.

Whatever we are, we are not ideal agents. If cognitive psychologists are right, we are computers. But a methodologist needn't debate the point, since methods should be routinely followable, without appeal to arcane powers of insight that cannot be accessed at will. A. Turing's conception of the Turing machine was intended to explicate just this notion of explicitly directed, method-driven behavior. Accordingly, Turing computability seems a natural constraint to impose on methodological principles.

Ideal epistemologists propose norms for the conduct of ideal scientists. But computational agents cannot do what ideal agents can do. In fact, this is an extreme understatement: there are uncountably many distinct problems solvable by noncomputable agents, but there are at most countably many distinct problems that are solvable by computational agents. If *ought* implies *can*, then we should expect computationally bounded methodology to be different from ideal methodology. In this chapter, we will consider both the differences and the similarities between ideal and computationally bounded methodology, all from a logical reliabilist point of view.

The ideal epistemologist might respond that ideals also bind computable agents, who should strive to satisfy the ideal even though exact compliance is impossible. That is not my opinion, however. If ideals may be utterly indifferent

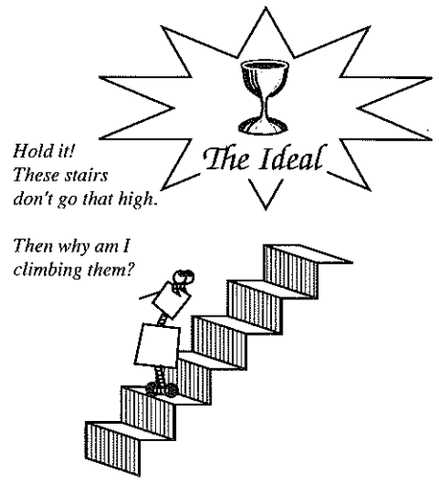


Figure 7.1

to our frailties, then inductive methodology would be the world's shortest subject; believe the whole truth and nothing but the truth. It is a candid admission of our severely limited perspective on the world that leads us to consider probabilities, inductive logics, and confirmation theories in the first place. To ignore computational boundedness after acknowledging perceptual boundedness makes little sense on the face of it (*Fig. 7.1*), and makes far less sense in light of the strong analogies between computability and ideal inductive inference that were discussed in the preceding chapter.

One solution is to find some hypothetical imperatives that justify approximations to the ideal directly. Maybe one can define degrees of *else*, so that it is clear to all that greater degrees of *else* are more horrible than smaller degrees of *else*. If one can also define degrees of approximation to the ideal that are achievable by bounded agents, then one can say strive harder and you will get more. This solves the problem, but it also seems to jettison the ideal. The degrees of the ideal are not ideal norms, but rather bounded norms achievable by bounded agents. The ideal, itself, ceases to do any work, and we are back to computationally bounded methodology.

A different approach is to hold that an ideal norm is binding in a given case if and only if it is achievable. If it is not achievable, then the violator is forgiven.[1] On this view, a computationally bounded agent is left entirely without

[1] "Standards of ideal rationality are not limiting cases of standards of bounded rationality. The standards are ideal in the sense that the capacity to satisfy them is one which an ideally rational agent, in contrast to a human agent, possesses. To advocate a standard of ideal rationality is not to promote realizable actions which approximate the prescriptions of such standards. Rather it is to promote conformity to such standards wherever that is feasible. The standard is an ideal in the sense that often we are incapable of meeting its requirements. In that case, we are urged to devise ways and means of satisfying these requirements in a larger number of cases" (Levi 1990): 214.

guidance in many cases unless unachievable ideals are augmented with achievable norms achievable in those cases.

Ideal methodology is not bad in itself. What a demigod can't do, a computer can't do. If a problem is so hard that gods cannot solve it, to say merely that it is uncomputable is like saying that Everest is a bit of a hill. Another reason for interest in ideal methodology is that it enables us to separate the purely topological (informational) dimensions of a problem from its purely computational dimensions by comparing the degree of underdetermination of a problem for ideal agents with its degree of underdetermination for computationally bounded agents.

Ideal results are fine so long as they are contrasted with and qualified by their computational counterparts. I depart company, however, when ideal results are portrayed as what is essential to methodology while computational considerations are brushed aside as bothersome details of application. As we saw in the preceding chapter, classical skepticism and the modern theory of computability are reflections of the same sorts of limitations and give rise to similar demonic arguments and hierarchies of underdetermination. I will accordingly treat computational boundedness as a full partner with perceptual boundedness, rather than as an accidental afterthought.

2. Computation as Internalized Inductive Inquiry

Some hypotheses lie beyond the scope of limiting, computable inquiry, even though they are decidable with certainty by an ideal agent. Consider, for example, a problem in which the first datum is an encoded clue that gives the problem away to an ideal agent, but that is too hard for a computer to decode. More precisely, let $S \subseteq \omega$, and let $\mathcal{P}_S = \{\varepsilon\colon \varepsilon_0 \in S\}$. Now let C, h be such that $C_h = \mathcal{P}_S$. Intuitively, h says that the first observation will be in S, so the members of S are clues that permit an ideal scientist to decide$_C$ h by time 1 (*Fig. 7.2*).

Now consider the following question. How does the purely computational complexity of S relate to the ability of a computer to decide$_C$ the truth value

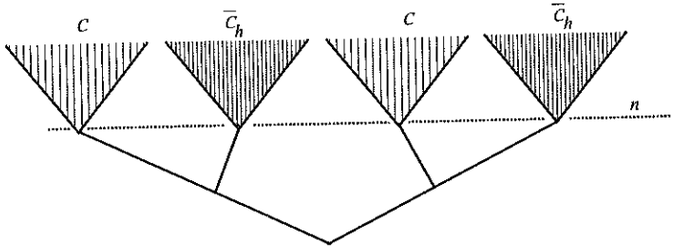


Figure 7.2

of h? If S is recursive, then a computer can succeed with certainty after scanning just the first datum, just like an ideal scientist. The computer uses its computable decision procedure to decide whether the first datum is in or out of S, and conjectures 1 or 0 forever after, accordingly.

Now suppose $S \in \Sigma_1^A - \Pi_1^A$. Then no computable method α can decide h with sample size 1, for otherwise α could be used as a decision procedure for S, contrary to assumption: to determine whether $n \in S$, feed $\alpha(h, \_)$ the sequence $n, 0, 0, 0, \ldots$ until the first '!' is returned and output the subsequent conjecture of α. On the other hand, some computable method α can verify h with certainty. This method simulates a positive test M for S for several steps on ε_1 every time a new datum arrives. α outputs 0 until M halts with 1, and outputs 1 thereafter. α treats membership in S as a kind of *internal* inductive problem that is, in fact, intrinsically harder than the ideal *external* empirical problem of deciding the truth of h. The required mind change is due entirely to the computational difficulty of this internal inductive question, rather than to the Borel complexity of $\mathcal{P}_S$ (*Fig. 7.3*). If $S \in \Pi_1^A$ and $C_h = \mathcal{P}_S$, then by similar reasoning, h can be refuted$_C$ with certainty by a computable scientist.

Suppose next that $S \in \Sigma_2^A - \Pi_2^A$. Then h cannot be decided$_C$ in the limit by a computable scientist. For suppose otherwise. Then we can construct a limiting recursive test M for S using the computable scientist α, as follows. To test $n \in \omega$, M feeds $\alpha(h, \_)$ the sequence $n, 0, 0, 0, \ldots$, which is clearly an effective task. Each time α makes an output, M makes the same output. Since α stabilizes to 1 if h is correct and to 0 otherwise, M stabilizes to 1 if $n \in S$ and to 0 otherwise, and hence is a limiting recursive test for S. But this is impossible by proposition 6.1. On the other hand, some computer can verify h in the limit by once again treating S as an internal inductive inference problem. Since $S \in \Sigma_2^A$, S is limiting r.e., so S has a limiting positive test M, by proposition 6.1. Our effective scientist α just feeds ε_1 to M and then repeats the outputs of M, ignoring all further empirical evidence. Similar points can be made concerning gradual verification and refutation.

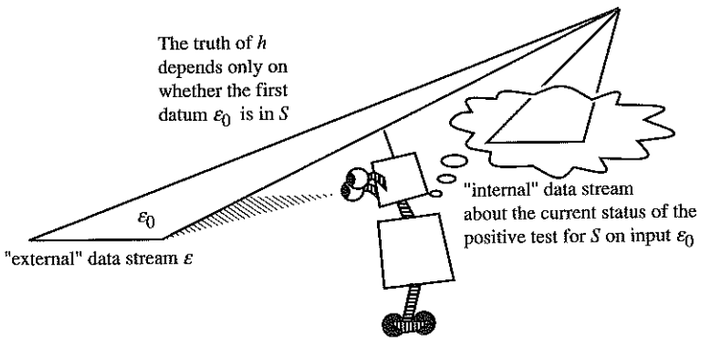


Figure 7.3

3. The Arithmetical Hierarchy over the Baire Space

We have just seen that a very trivial inductive problem in the ideal sense can be arbitrarily hopeless for a computer to solve. The problem is that clues that give away an inductive problem to an ideal agent can generate inductive difficulties for a computer. But our study so far has been restricted to a very particular type of inductive problem in which ideal and computational considerations are kept neatly separate to make the point as clearly as possible. It remains to characterize just how these two kinds of complexity—ideal and computational—interact to determine whether or not an inductive problem is solvable by a computer. Once again, we need a way to describe the computational structure of a hypothesis in a way that makes no reference to computable scientists or to decision.

Recall that the Borel hierarchy provides a natural characterization of the complexity of ideal inductive inference, because it is based on the notion of empirical verifiability. We will now consider the *arithmetical hierarchy of sets of functions*, built on the notion of empirical verifiability by a computer. This hierarchy will provide a natural setting for the characterization of the computational complexity of inductive inference.

Recall that a correctness relation $C(\varepsilon, h)$ is a relation of mixed type, holding between data streams (type 2 objects) and code numbers of hypotheses (type 1 objects). In general, a relation S might have any number of type 1 and type 2 arguments. For example, $S(\varepsilon, \tau, n, k)$ has two type 2 arguments and two type 1 arguments. In general, we say that S is of *type* (k, n) just in case it has k arguments of type 2 and n arguments of type 1. We will define arithmetical complexity all at once for relations of all types (k, n), not out of a misplaced love of generality, but because it makes the inductive definition easier to state. The arithmetical hierarchy defined in the preceding chapter will turn out to be a special case, in which only relations of type $(0, n)$ are considered.

In the case of ordinary computation, the base of the hierarchy is the set of recursive relations of type $(0, n)$. But now our relations have infinite data streams as arguments. How do we interpret a Turing machine as accepting an infinite input? The trick is to provide the machine with an extra input tape for each infinite input stream. But it might be objected that the machine still cannot read the whole infinite sequence written on the tape prior to making an output. That is correct, but it doesn't matter. A decision procedure is still required to output 1 or 0 after a finite amount of time, just as in the case of finite inputs. It follows that the decision procedure must extract whatever information it requires for an answer after scanning only some finite chunk of the infinite input tape carrying ε.[2]

With this picture in mind, we see that our previous definitions of decision procedure, positive test, limiting positive test, and so on all apply to the case of relations of type (k, n), so long as the procedure in question is provided with a separate tape for each infinite input. In particular, a *decision procedure* for a

[2] In the theory of computability, this is sometimes referred to as the *use principle*.

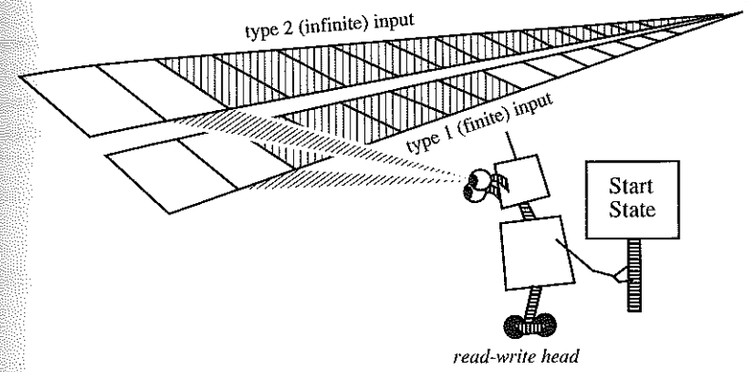


Figure 7.4

relation S of type (k, n) is a Turing machine that when provided with finite arguments in the usual way and with a separate tape for each of the k infinite arguments in the manner just described eventually halts with 1 if S holds of the inputs, and with 0 otherwise. S is said to be *recursive* just in case it has a decision procedure (*Fig. 7.4*).

It is straightforward to extend the arithmetical hierarchy defined in the preceding chapter to relations of mixed type. Let S be a type (k, m) relation. To eliminate tedious repetitions and subscripts, let $\bar{x}$ denote an n-vector of natural numbers and let $\bar{\varepsilon}$ denote a k-vector of data streams, so that we may write $S(\bar{\varepsilon}, \bar{x})$ instead of $S(\varepsilon[1], \ldots, \varepsilon[k], x_1, \ldots, x_m)$, where each $\varepsilon[i] \in \mathcal{N}$.

$S \in \Sigma_0^A \Leftrightarrow S$ *is recursive.*

$S \in \Sigma_{n+1}^A \Leftrightarrow$ *there is some type* $(k, m+1)$ *relation* $V \in \Sigma_n^A$ *such that for each* $\bar{\varepsilon} \in \mathcal{N}^k$, $\bar{x} \in \omega^m$, $S(\bar{\varepsilon}, \bar{x}) \Leftrightarrow \exists x_{m+1}$ *such that* $\neg V(\bar{\varepsilon}, \bar{x}, x_{m+1})$.

The Π_n^A classes and the ambiguous Δ_n^A classes are defined in terms of Σ_0^A in the usual way.

To illustrate the relationship between this hierarchy and the finite Borel hierarchy, we can extend the latter notion to cover relations of mixed type. Let $\mathfrak{T}_1, \ldots, \mathfrak{T}_n$ be topological spaces with countable bases $B_1, \ldots, B_n$, respectively. The *product space* $\mathfrak{T}_1 \times \cdots \times \mathfrak{T}_n$ then has the basis $B_1 \times \cdots \times B_n$. Now consider the space $\mathcal{N}^k \times \omega^n$. Each space $\mathcal{N}$ carries the Baire topology, and we will think of ω as having the *discrete topology*, in which each singleton $\{n\}$ is a basic open set. This choice is not arbitrary. Basic open sets correspond in our application to what is directly input to the inductive method at a given stage of inquiry. In the Baire space, finite chunks of data are given and, in hypothesis assessment, we assume that discrete hypotheses are given to be investigated.

Thus, each singleton $\{h\}$ should be basic open.[3] Accordingly, a basic open set in $\mathcal{N}^k \times \omega^n$ is a cross product of k fans and n singletons containing natural numbers. Now the definition of the finite Borel hierarchy can be rephrased so as to mimic the definition of the arithmetical hierarchy. Let S be of type (k, m). Then we have:

$$S \in \Sigma_0^B \Leftrightarrow S \text{ is clopen in } \mathcal{N}^k \times \omega^m.$$

$$S \in \Sigma_{n+1}^B \Leftrightarrow \text{there is some type } (k, m+1) \text{ relation } \mathcal{V} \in \Sigma_n^B \text{ such that for each } \bar{\varepsilon} \in \mathcal{N}^k, \bar{x} \in \omega^m, S(\bar{\varepsilon}, \bar{x}) \Leftrightarrow \exists x_{m+1} \text{ such that } \neg \mathcal{V}(\bar{\varepsilon}, \bar{x}, x_{m+1}).$$

The crucial point for our purposes is that this definition of *ideal* complexity differs from the preceding definition of *computational* complexity only in the base case. I will refer to this as the *new sense* of the Borel hierarchy until I show that it agrees with the one defined in chapter 4 over type (1, 0) relations.

The discrete topology on ω causes the structure of the type 1 part of a mixed type relation to wash out of its Borel complexity classification in the following sense:

Proposition 7.1

Let S be of type (k, n). Let $S_{\bar{x}} = \{\bar{\varepsilon}: S(\bar{\varepsilon}, \bar{x})\}$. Then $S \in \Sigma_n^B$ (in the new sense) $\Leftrightarrow$ for each $\bar{x} \in \omega^n$, $S_{\bar{x}} \in \Sigma_n^B$ (in the new sense). ■

It follows from this fact that the new version of the Borel hierarchy agrees with the old on type (1, 0) relations, so we needn't distinguish the two senses any longer for subsets of $\mathcal{N}$.

Proposition 7.2

Let $S \subseteq \mathcal{N}$. Then S is Σ_n^B in the old sense $\Leftrightarrow$ S is Σ_n^B in the new sense. ■

In light of proposition 7.2, the arithmetical hierarchy over $\mathcal{N}$ is sometimes referred to as the *effective, finite Borel hierarchy*. It follows from proposition 7.2 and the fact that the base case of the arithmetical hierarchy is at least as stringent as the base case of the Borel hierarchy, that membership in an arithmetical complexity class implies membership in the corresponding Borel complexity class. The example at the beginning of this section makes it clear that all the inclusions are proper (just insert arbitrarily complex subsets of ω in for S in the example). Thus:

[3] If, on the other hand, hypotheses were parametrized by a real-valued parameter θ, then it might be impossible to give one to a method except by specifying ever narrower intervals around it. In that case, the set of hypotheses would also carry a nontrivial topology in which basic open sets are intervals with rational-valued endpoints. Each singleton containing a hypothesis would then be closed but not open. In such a setting, determining which hypothesis one is supposed to investigate can be as hard as determining the truth of that hypothesis from data, so ideal inquiry becomes more analogous to computable inquiry in this respect.

Proposition 7.3

For each n, $\Sigma_n^A \subset \Sigma_n^B$. ■

Proposition 7.1 explains why we could characterize the scope of ideal inquiry in terms of the topological complexities of each C_h, instead of in terms of C as a relation, since the Borel complexity of C is entirely determined by the complexities of each C_h. But when we move to the computable case, the complexity of C can be arbitrarily high even when the computational complexity of each C_h is trivial. For the simplest example, consider $C(\varepsilon, h) \Leftrightarrow h \in X$, where X is some arbitrarily hard to decide subset of ω. Here, correctness does not depend on the data at all, so any particular hypothesis can be decided a priori by a method with a fixed conjecture. Nonetheless, the difficulty of matching the right a priori conjecture with the right hypothesis prevents a single computable method from assessing all hypotheses in ω.

The failure of proposition 7.1 in the computational case means that computable inquiry must be characterized in terms of the arithmetical complexity of the entire correctness relation C, rather than pointwise, in terms of each C_h, as in the ideal case. It also means that both the scientist's background assumptions $\mathcal{K}$ and the range H of hypotheses he might have to assess are relevant to the complexity of his problem. Accordingly, we must relativize arithmetical complexity both to $\mathcal{K}$ and to H.

$$S \in \Sigma[\mathcal{K}, H]_n^A \Leftrightarrow \exists \mathcal{V} \in \Sigma_n^A \text{ such that } \forall \bar{\varepsilon} \in \mathcal{K}^k, \bar{x} \in H^n, S(\bar{\varepsilon}, \bar{x}) \Leftrightarrow \mathcal{V}(\bar{\varepsilon}, \bar{x}).$$

4. Universal Relations and Hierarchy Theorems

It will be shown in this section that the Borel and arithmetical hierarchies are genuinely hierarchies, in the sense that each complexity class contains hypotheses not contained in any lower complexity class. The proof will be based on a way of assigning indices to all the problems in a given Σ_n^A class by means of a *universal relation* for the class. The idea of a universal relation depends essentially on the notion of encoding finite sequences of numbers and finite sequences of data streams. Let $\langle \_ \rangle$ be a 1–1, effective encoding of ω^* into ω, so that $\langle e \rangle$ is a code number for e. We may also encode a finite vector of

data streams into a single data stream using the same encoding. Let $\bar{\varepsilon} = (\varepsilon[1], \ldots, \varepsilon[n])$. Define:

$$\langle\bar{\varepsilon}\rangle = (\langle\varepsilon[1]_0, \ldots, \varepsilon[n]_0\rangle, \langle\varepsilon[1]_1, \ldots, \varepsilon[n]_1\rangle, \ldots, \langle\varepsilon[1]_m, \ldots, \varepsilon[n]_m\rangle, \ldots).$$

In other words, $\langle\bar{\varepsilon}\rangle_m = \langle\varepsilon[1]_m, \ldots, \varepsilon[n]_m\rangle$, for each $m \in \omega$.

The *Turing relation* $\mathcal{T}(i, \langle\bar{\varepsilon}\rangle, \langle\bar{x}\rangle, k)$ is to be understood as saying that machine M_i halts on inputs $\bar{\varepsilon}$, $\bar{x}$ within k steps of computation.[4] This relation is recursive by Church's thesis since it is a straightforward matter to decode i, $\langle\bar{\varepsilon}\rangle$, and $\langle\bar{x}\rangle$ and to simulate the computation $M_i[\bar{\varepsilon}, \bar{x}]$ for k steps, observing whether the computation halts by that time. Let $\mathcal{U}$ be a type (1, 2) relation. Now define:

$\mathcal{U}$ is universal for complexity class $\Theta \Leftrightarrow$

(1) *$\mathcal{U} \in \Theta$ and*

(2) *for each $S \in \Theta$, there is an i such that for all $\bar{\varepsilon}$, $\bar{x}$, $S(\bar{\varepsilon}, \bar{x}) \Leftrightarrow \mathcal{U}(i, \langle\bar{\varepsilon}\rangle, \langle\bar{x}\rangle)$.*

In other words, for each $S \in \Theta$, there is an index i such that $S = \{(\bar{\varepsilon}, \bar{x})\colon \mathcal{U}(i, \langle\bar{\varepsilon}\rangle, \langle\bar{x}\rangle)\}$. The question now is whether there exists a universal relation for each complexity class Σ_n^A. One candidate is defined inductively as follows:

$$\mathcal{U}_1^A(i, \langle\bar{\varepsilon}\rangle, \langle\bar{x}\rangle) \Leftrightarrow \exists k \text{ such that } \mathcal{T}(i, \langle\bar{\varepsilon}\rangle, \langle\bar{x}\rangle, k).$$

$$\mathcal{U}_{n+1}^A(i, \langle\bar{\varepsilon}\rangle, \langle\bar{x}\rangle) \Leftrightarrow \exists k \text{ such that } \neg\mathcal{U}_n^A(i, \langle\bar{\varepsilon}\rangle, \langle\bar{x}, k\rangle).$$

$\mathcal{U}_n^A$ is a type (1, 2) relation. The inductive clause follows the usual mechanism for building complexity in our hierarchies: existential quantification and negation. The trick is that the existential quantifier is added without adding to the arity of the relation at the lower level by means of coding the bound variable k into $\langle\bar{x}, k\rangle$. Since the coding is 1–1, it does not "erase" any extra complexity added by the quantifier. Now we have:

Proposition 7.4 The arithmetical indexing theorem[5]

(a) *For each $n \geq 1$, $\mathcal{U}_n^A$ is universal for Σ_n^A.*

(b) *For each $n \geq 1$, $\overline{\mathcal{U}_n^A}$ is universal for Π_n^A.*

[4] There is no ambiguity about the arities of $\bar{\varepsilon}$ and of $\bar{x}$ because the encoding $\langle\rangle$ is 1–1 from ω^* to ω.

[5] The following two proofs follow Hinman (1978).

Proof: (a) (1) A simple induction establishes that $\mathcal{U}_n^A \in \Sigma_n^A$. (2) We now show that $\mathcal{U}_n^A$ indexes all Σ_n^A relations. *Base*: Suppose $S \in \Sigma_1^A$. Let M_i be a positive test for S. Then for all $\bar{\varepsilon}$, $\bar{x}$, $S(\bar{\varepsilon}, \bar{x}) \Leftrightarrow \exists k$ such that $\mathcal{T}(i, \langle\bar{\varepsilon}\rangle, \langle\bar{x}\rangle, k) \Leftrightarrow \mathcal{U}_1^A(i, \langle\bar{\varepsilon}\rangle, \langle\bar{x}\rangle)$. *Induction*: Suppose the result for all $n' \leq n$. Let $S \in \Sigma_{n+1}^A$. Then for some $G \in \Sigma_n^A$, for all $\bar{\varepsilon}$, $\bar{x}$, $S(\bar{\varepsilon}, \bar{x}) \Leftrightarrow \exists k \neg G(\bar{\varepsilon}, \bar{x}, k)$. By the induction hypothesis there is some i such that for all $\bar{\varepsilon}$, $\bar{x}$, k, $G(\bar{\varepsilon}, \bar{x}, k) \Leftrightarrow \mathcal{U}_n^A(i, \langle\bar{\varepsilon}\rangle, \langle\bar{x}, k\rangle)$. Thus $S(\bar{\varepsilon}, \bar{x}) \Leftrightarrow \exists k \neg \mathcal{U}_n^A(i, \langle\bar{\varepsilon}\rangle, \langle\bar{x}, k\rangle)$. (b) is similar. ■

Now it is not hard to see that there is always something new at each level in the hierarchy.

Proposition 7.5 The arithmetical hierarchy theorem

For each n, $\Delta_n^A \subset \Sigma_n^A$.

Proof: The inclusions are immediate. That they are proper is shown as follows. Define the diagonal relation

$$\text{(a) } \mathcal{D}_n(x) \Leftrightarrow \mathcal{U}_n^A(x, \langle\,\rangle, \langle x\rangle),$$

where $\langle\,\rangle$ in the second argument position denotes the infinite sequence of code numbers for the empty vector $\mathbf{0}$. From the definition of $\mathcal{U}_n^A$ it is evident that $\mathcal{D}_n \in \Sigma_n^A$. Suppose $D_n \in \Delta_n^A$. Hence $\overline{\mathcal{D}_n} \in \Sigma_n^A$. Since $\mathcal{U}_n^A$ is universal for Σ_n^A (proposition 7.4), there is some b such that for all x,

$$\text{(b) } \overline{\mathcal{D}_n}(x) \Leftrightarrow \mathcal{U}_n^A(b, \langle\,\rangle, \langle x\rangle).$$

So in particular,

$$\text{(c) } \overline{\mathcal{D}_n}(b) \Leftrightarrow \mathcal{U}_n^A(b, \langle\,\rangle, \langle b\rangle).$$

But by (a),

$$\text{(d) } \mathcal{D}_n(b) \Leftrightarrow \mathcal{U}_n^A(b, \langle\,\rangle, \langle b\rangle).$$

Thus by (d) and (c),

$$\overline{\mathcal{D}_n}(b) \Leftrightarrow \mathcal{D}_n(b),$$

which is a contradiction. ■

All the complexity exhibited by $\mathcal{U}_n^A$ is computational, since the diagonalization is based entirely on numeric arguments rather than on infinite data streams. We know that Borel complexity is a lower bound on arithmetical complexity, so it would be very useful to be able to see how to build arbitrary complexity through diagonalization in the Borel hierarchy as well. It is thus useful at this

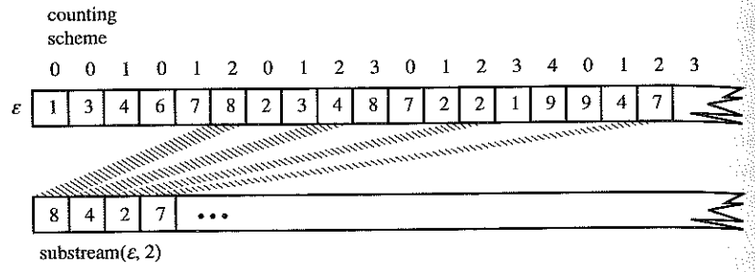


Figure 7.5

point to examine the proof of the analogous result for the finite Borel hierarchy, which was announced without proof as proposition 4.9. The argument is due to K. Kuratowski.[6] Once again, we proceed in two steps, constructing a universal relation for finite Borel classes and then diagonalizing to show that the hierarchy does not collapse.

Since there are already uncountably many open subsets of $\mathcal{N}$ (just think of all the ways of taking countable unions of disjoint fans), we cannot use natural numbers as indices as we did in the computational case. Kuratowski's idea was to use data streams as indices for Borel sets. Let ε be a data stream. Count off positions in ε according to the following fixed and effective pattern: 0; 0, 1; 0, 1, 2; ... 0, 1, 2, ..., n; .... Then *substream*(ε, n) is just the subsequence of ε whose positions are marked with n when we label successive positions in ε according to this fixed counting scheme (*Fig. 7.5*). Clearly, for each ω-sequence Ψ of data streams, there is a unique data stream ε such that for each n, Ψ_n = *substream*(ε, n): simply write the ith data stream listed in Ψ_n into the positions labeled i in the standard counting scheme. Now we may define the following candidate for a universal relation for Σ^B_n:

$\mathcal{U}^B_1(\tau, \varepsilon) \Leftrightarrow \exists i \in \omega$ *such that* $\tau_i = \langle e \rangle$ *and* $\varepsilon \in [e]$.

$\mathcal{U}^B_{n+1}(\tau, \varepsilon) \Leftrightarrow \exists i$ *such that* $\neg \mathcal{U}^B_n$ (*substream*(τ, i), ε).

Proposition 7.6 Borel indexing theorem

(a) *For each* $n \geq 1$, $\mathcal{U}^B_n$ *is universal for* Σ^B_n.

(b) *For each* $n \geq 1$, $\overline{\mathcal{U}^B_n}$ *is universal for* Π^B_n.

Proof: (a) A simple induction establishes that $\mathcal{U}^B_n \in \Sigma^B_n$. *Base*: Let $\mathcal{G} \subseteq \mathcal{N}$ be Σ^B_1. Then for some $G \subseteq \omega^*$, $\mathcal{G}$ is the union of all fans $[e]$ such that $e \in G$. Let τ be an enumeration of the code numbers of elements of G. Then $\varepsilon \in \mathcal{G} \Leftrightarrow \mathcal{U}^B_1(\tau, \varepsilon)$. *Induction*: Suppose $\mathcal{G} \in \Sigma^B_{n+1}$. Then $\mathcal{G}$ is a countable union of sets $\neg S_0 \cup \neg S_1 \cup \neg S_2 \cup \ldots$ such that each $S_i \in \Sigma^B_n$. By the induction hypothesis, for each S_i, there is a data stream $\tau[i]$ such that $S_i = \{\varepsilon: \mathcal{U}^B_n(\tau[i], \varepsilon)\}$.

[6] Kuratowski (1966): 368–371.

Construct τ so that for each i, $\tau[i]$ = *substream*(τ, i). Then $\varepsilon \in \mathcal{G} \Leftrightarrow \exists i\ \varepsilon \notin S_i \Leftrightarrow \exists i \neg \mathcal{U}^B_n(\tau[i], \varepsilon) \Leftrightarrow \exists i \neg \mathcal{U}^B_n$(*substream*($\tau$, i), ε) $\Leftrightarrow \mathcal{U}^B_{n+1}(\tau, \varepsilon)$. (b) is similar. ■

The Borel hierarchy theorem (proposition 4.9) can now be established by diagonalization on $\mathcal{U}^B_n$, just as in the arithmetical case:

Proposition 7.7 Borel hierarchy theorem

For each n, $\Delta^B_n \subset \Sigma^B_n$. ■

We now have two powerful techniques for constructing an inductive problem with arbitrarily high ideal or computational complexity when $\mathcal{K} = \mathcal{N}$ and $H = \omega$. Of course, restrictions on $\mathcal{K}$ and H can collapse the hierarchy to some finite level. For example, if $\mathcal{K} = \{\varepsilon\}$, then every subset of $\mathcal{K}$ is $\Delta[\mathcal{K}]^B_1$. The general question of how the hierarchy theorems depend on $\mathcal{K}$ and H is an interesting and important one that lies beyond the scope of this book.[7]

5. Characterization Theorems

The characterizations for the effective case look very much like the characterizations for the ideal case, except that the Borel hierarchy on $\mathcal{N}$ is exchanged for the arithmetical hierarchy on $\mathcal{N}$. As it turns out, very few modifications of the proofs given for the ideal case are necessary. First, all the basic correspondences established for ideal agents in chapter 3 hold in the computable case as well.

Proposition 7.8

(a) *H is verifiable*$_C$ $\left[\begin{array}{c}\textit{with certainty}\\ \textit{in the limit}\\ \textit{gradually}\end{array}\right]$ *by a Turing machine given* $\mathcal{K}$

$\Leftrightarrow$ *H is refutable*$_{\bar{C}}$ $\left[\begin{array}{c}\textit{with certainty}\\ \textit{in the limit}\\ \textit{gradually}\end{array}\right]$ *by a Turing machine given* $\mathcal{K}$.

(b) *H is decidable*$_C$ $\left[\begin{array}{c}\textit{with certainty}\\ \textit{in the limit}\end{array}\right]$ *by a Turing machine given* $\mathcal{K}$

$\Leftrightarrow$ *H is both verifiable*$_C$ *and refutable*$_C$ *by a Turing machine given* $\mathcal{K}$.

(c) *H is decidable*$_C$ *with n mind changes starting with* 1 *by a Turing machine given* $\mathcal{K} \Leftrightarrow$ *H is decidable*$_{\bar{C}}$ *with n mind changes starting with* 0 *by a Turing machine given* $\mathcal{K}$.

[7] Some ideas about this issue may be found in Moschavakis (1980).

(d) *For each r such that $0 < r < 1$, H is decidable$_C$ with n mind changes starting with $r \Leftrightarrow H$ is decidable$_C$ with n mind changes starting with 0 by a Turing machine given $\mathcal{K}$ and H is decidable$_C$ with n mind changes starting with 1 by a Turing machine given $\mathcal{K}$.* ∎

Now the characterizations of reliable inquiry by Turing-computable agents may be stated as follows:

Proposition 7.9[8]

(a) H *is* $\begin{bmatrix} \text{verifiable}_C \\ \text{refutable}_C \\ \text{decidable}_C \end{bmatrix}$ *with certainty by a Turing machine given* $\mathcal{K}$

$$\Leftrightarrow C \cap (\mathcal{K} \times H) \in \begin{bmatrix} \Sigma[\mathcal{K}, H]^A_1 \\ \Pi[\mathcal{K}, H]^A_1 \\ \Delta[\mathcal{K}, H]^A_1 \end{bmatrix}.$$

(b) H *is* $\begin{bmatrix} \text{verifiable}_C \\ \text{refutable}_C \\ \text{decidable}_C \end{bmatrix}$ *in the limit by a Turing machine given* $\mathcal{K}$

$$\Leftrightarrow C \cap (\mathcal{K} \times H) \in \begin{bmatrix} \Sigma[\mathcal{K}, H]^A_2 \\ \Pi[\mathcal{K}, H]^A_2 \\ \Delta[\mathcal{K}, H]^A_2 \end{bmatrix}.$$

(c) H *is* $\begin{bmatrix} \text{verifiable}_C \\ \text{refutable}_C \\ \text{decidable}_C \end{bmatrix}$ *gradually by a Turing machine given* $\mathcal{K}$

$$\Leftrightarrow C \cap (\mathcal{K} \times H) \in \begin{bmatrix} \Pi[\mathcal{K}, H]^A_3 \\ \Sigma[\mathcal{K}, H]^A_3 \\ \Delta[\mathcal{K}, H]^A_2 \end{bmatrix}.$$

Proof: By proposition 7.8, we need consider only the verification case of each statement, except for the case of gradual decidability, which follows from (b) along the lines of proposition 3.13(b). It is convenient to treat the ($\Rightarrow$) side of all three statements together. (a) If computable α verifies H with certainty given $\mathcal{K}$, then we have for all $\varepsilon \in \mathcal{K}$, $h \in H$,

$$C(\varepsilon, h) \Leftrightarrow \exists n[\alpha(h, \varepsilon|n) = \text{'!'} \,\&\, \alpha(h, \varepsilon|n+1) = 1 \quad \text{and} \quad \forall m < n, \alpha(h, \varepsilon|m) \neq \text{'!'}].$$

[8] (a) and (b) are in Gold (1965).

Since α is total recursive, the relation in square brackets is recursive. Hence,

$$C(\varepsilon, h) \cap (\mathcal{K} \times H) \in \Sigma[\mathcal{K}, H]^A_1.$$

(b) If computable α verifies H in the limit given $\mathcal{K}$, then we have for all $\varepsilon \in \mathcal{K}$, $h \in H$,

$$C(\varepsilon, h) \Leftrightarrow \forall n \exists m \geq n\, \alpha(h, \varepsilon|m) = 1.$$

Since $\alpha(h, \varepsilon|m)$ is a recursive relation, we have $C(\varepsilon, h) \cap (\mathcal{K} \times H) \in \Sigma[\mathcal{K}, H]^A_2$.
(c) If computable α verifies H gradually given $\mathcal{K}$, then we have for all $\varepsilon \in \mathcal{K}$, and for all $h \in H$,

$$C(\varepsilon, h) \Leftrightarrow \forall \text{ rational } r > 0\, \exists n \forall m \geq n\, 1 - \alpha(h, \varepsilon|m) < r,$$

so $C(\varepsilon, h) \cap (\mathcal{K} \times H) \in \Pi([\mathcal{K}, H]^A_3$, since the rationals may be effectively encoded 1–1 by natural numbers.

($\Leftarrow$) All we do is to show that the universal architectures introduced in chapter 5 can be implemented on Turing machines whenever C has the corresponding arithmetical complexity. (a) is immediate.

(b) Suppose $C \cap (\mathcal{K} \times H) \in \Sigma[\mathcal{K}, H]^A_2$. Then there is some co-r.e. relation $\mathcal{G}$ such that for each $\varepsilon \in \mathcal{N}$, $h \in H$, $C(\varepsilon, h) \Leftrightarrow \exists n\, \mathcal{G}(\varepsilon, h, n)$. Let M be a refutation procedure for $\mathcal{G}$. Then we implement an effective version of the bumping pointer architecture using M. Let $h \in H$ be given. On the empty data sequence $\mathbf{0}$, the method α outputs 0. When $e \neq \mathbf{0}$, define:

$$\text{pointer}(h, e) = \begin{cases} \text{the least } i \leq lh(e) \\ \quad \text{such that } M[e, h, i] \text{ does not return 0 in } lh(e) \text{ steps,} \\ \quad \text{if there is such an } i \\ lh(e) \text{ otherwise.} \end{cases}$$

The pointer is clearly recursive. Now let $\alpha(h, \mathbf{0}) = 1$ and define:

$$\alpha(h, e^*x) = \begin{cases} 1 & \text{if pointer}(h, e) = \text{pointer}(h, e^*x) \\ 0 & \text{otherwise.} \end{cases}$$

On data e, the pointer is moved to the least $i \leq lh(e)$ such that $M[e, h, i]$ does not return 0 in $lh(e)$ steps if there is one and to $lh(e)$ otherwise. Then the pointer is moved to the least $i \leq lh(e)$ such that $M[e^*x, h, i]$ does not return 0 in $lh(e)$ steps if there is one, and to $lh(e) + 1$ otherwise. If this position is different from the previous one, α conjectures 0. Otherwise, α conjectures 1 (*Fig. 7.6*).

It is readily seen that this procedure is computable. It is left to the reader to verify that it works.

(c) Suppose $C \cap (\mathcal{K} \times H) \in \Pi[\mathcal{K}, H]^A_3$. Then for some $\Sigma[\mathcal{K}, H]^A_2$ relation $\mathcal{G}$, we have for each $\varepsilon \in \mathcal{K}$, $h \in H$, $C(\varepsilon, h) \Leftrightarrow \forall n\, \mathcal{G}(\varepsilon, h, n)$. By an argument similar

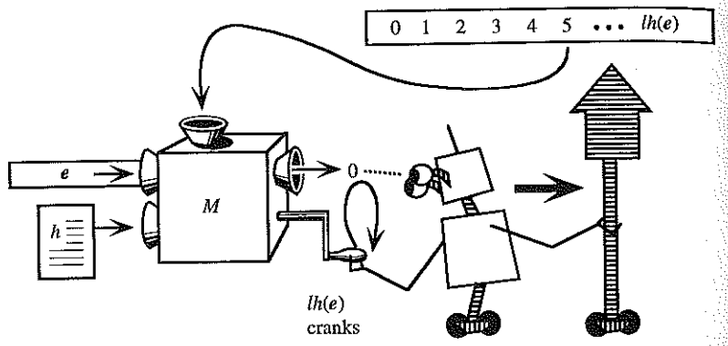


Figure 7.6

to the preceding, we obtain a single machine M that for each fixed n, $M[h, e, n]$ is a limiting verifier for $\mathcal{G}(\varepsilon, h, n)$. Since M is total, we can dispense with crank counting. On inputs e, h, our method α simulates M for $n = 0, 1, \ldots, lh(e)$, and then counts how many successive 1s starting from position 0 are in the sequence $M[e, h, 0], M[e, h, 1], \ldots, M[e, h, lh(e)]$. Call this number k. Then α conjectures $1 - 2^{-k}$ (*Fig. 7.7*).

Again, it is clear that this procedure is computable, and the verification of correctness is left to the reader. ■

Proposition 7.9 vindicates the promise in chapter 1 that the reliabilist approach to method provides a seamless analogy between computable and ideal methodology. The characterizations listed under proposition 7.9 are almost identical to the ideal characterizations, except that arithmetical complexity requires computable decidability in its base case whereas Borel complexity demands only ideal decidability. Moreover, the complete computable architectures introduced in the proof are but minor variants of the ideal methods seen in the preceding chapter.

The analogy can be carried through for decision with n mind changes as well. First we must introduce an effective version of the finite difference hierarchy, which will be denoted d. The d hierarchy is defined just like

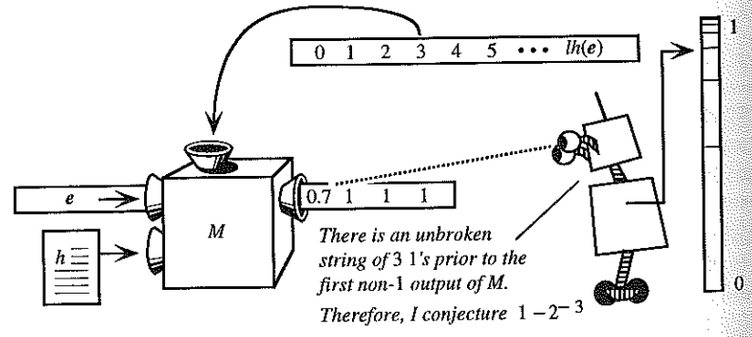


Figure 7.7

the D hierarchy, except that we substitute $\mathcal{K}$-r.e. sets and their complements for $\mathcal{K}$-open sets and $\mathcal{K}$-closed sets, respectively. Now we may state the characterization:

Proposition 7.10

For all r such that $0 < r < 1$, H is decidable$_C$ with n mind changes starting with $\begin{bmatrix} 0 \\ 1 \\ r \end{bmatrix}$ *by a Turing machine given $\mathcal{K}$*

$$\Leftrightarrow C \cap (\mathcal{K} \times H) \in \begin{bmatrix} \Sigma[\mathcal{K}, H]^d_n \\ \Pi[\mathcal{K}, H]^d_n \\ \Delta[\mathcal{K}, H]^d_n \end{bmatrix}.$$

Proof: Exercise 7.4. ■

The tools developed so far place us in a good position to compare ideal underdetermination to its computationally bounded counterpart. For example, we now know that each computationally underdetermined problem involves two components: a purely topological component reflective of purely empirical (or "external") underdetermination, together with a purely computational component reflective of formal (or "internal") underdetermination. Different problems partake of different mixtures of these two factors. For example, when $C_h = \mathcal{P}_S$ (i.e., the set of all data streams whose first data points are in S), the topological complexity of C_h is trivial, but the computational complexity of C_h can be arbitrarily high, depending on the nature of S (i.e., $S \in \Delta^B_1 - \Sigma^A_n$, where n may be as high as we please). On the other hand, in some problems, virtually all the computational difficulty is topological. This is true of the infinite divisibility hypothesis, which is in $\Pi^A_2 - \Sigma^B_2$, so there is no gap at all between the abilities of ideal and computable scientists so far as this problem is concerned. Other problems involve some intermediate mixture of the two factors. We will consider an interesting example of such a problem in section 7 of this chapter.

6. Data-Minimal Computable Methods

Chapter 4 presented the surprising result that every solvable limiting verification problem has a data-minimal solution. A natural question is whether computationally bounded inquiry can boast the same result. One might worry in this regard that the universal architecture for limiting verification tests a co-r.e. subset of C_h for just a bounded number of steps before pausing and asking for the next datum. This business of letting formal analysis pile up on the desk while new data comes in reflects the problem that there is no way to be sure

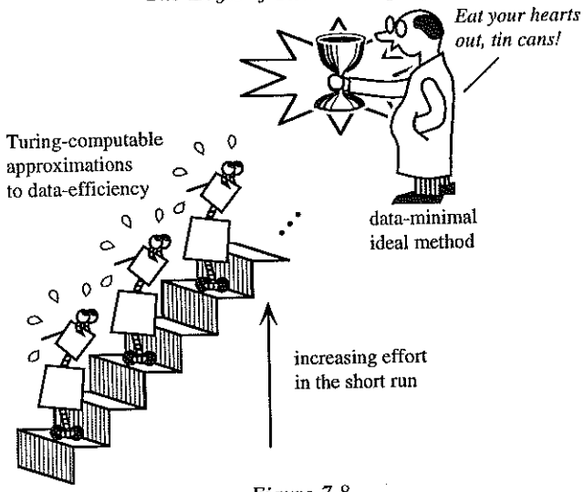


Figure 7.8

in the present that a proof of inconsistency between the data and a given co-r.e. set will never be found by computational means. Perhaps no matter how hard the computer works, there will in principle always be some computational backlog on its desk, so that a more industrious method that does more formal work at each stage converges to the truth sooner. Indeed, this is the case. Recall that $\mathcal{R} = \{\phi: \phi \text{ is total recursive}\}$ and let $\mathcal{P}_K = \{\varepsilon: \varepsilon_0 \in K\}$, where K is the halting problem (cf. chapter 6). Let $C_h = \mathcal{P}_K$. Then we have:

Proposition 7.11[9]

If $C_h = \mathcal{P}_K$ then $C_h \in \Sigma_1^A$ but no computable α is a weakly data-minimal limiting decider of h.

Proof: $C_h \in \Sigma_1^A$. Let computable α decide$_C$ h in the limit. There is an $\varepsilon \in \mathcal{N}$ such that $modulus_\alpha(\varepsilon) > 1$, else the computable function $\phi(x) = \alpha(h, (x))$ decides K. Suppose $\varepsilon \in \mathcal{P}_K$. Then $\varepsilon_0 \in K$. Define:

$$\beta(h, \boldsymbol{e}) = \begin{cases} 1 & \text{if } \boldsymbol{e}_0 = \varepsilon_0 \\ \alpha(h, \boldsymbol{e}) & \text{otherwise.} \end{cases}$$

Now $modulus_\beta(h, \varepsilon) < modulus_\alpha(h, \varepsilon)$, but for all other data streams, β's modulus remains unchanged if the first datum is not ε_0, and β's modulus is improved if the first datum is ε_0. The case in which $\varepsilon \notin \mathcal{P}_K$ is similar. ■

When it is impossible to avoid testing co-r.e. sets for consistency with the data, there is some ideal, data-minimal solution and a chain of ever better recursive approximations to data minimality that expend ever greater effort in finding refutation proofs in the short run (*Fig. 7.8*).

[9] This result is analogous to proposition 8.2.3.A in Osherson et al. (1986).

At the beginning of this chapter it was proposed that to regulate concrete action among bounded agents an ideal methodological norm should come with an argument to the effect that extra degrees of striving toward the ideal yield extra degrees of some value that exact achievement of the ideal would provide completely. Here we have exactly such a situation. A computable method may not be able to decide logical consistency with the data, but spending more time on the problem in the short run before examining the next datum may lead to a proof of refutation earlier and hence may sometimes lead to earlier convergence to the truth. Such a method must use some runtime bound to succeed at all, and hence can never duplicate the data-minimality of an ideal method; but increasing effort in the short run brings it ever closer to that unreachable ideal, so there is a reward for extra effort expended in the short run.

7. The Empirical Irony of Cognitive Science

A sentence h is *self defeating* when

h is true $\Rightarrow$ h is false.

For example:

All sentences involving more than three words are false.

A hypothesis h is *empirically self-defeating in the limit for us* just in case

h is true $\Rightarrow$ h is not verifiable in the limit by us.

Are any hypotheses of independent interest empirically self-defeating? It turns out that the cognitivist hypothesis that human behavior is computable provides such an example! Recall that $C_{h_{comp}}$ is the set of all computable data streams in 2^ω. We have already seen by a demonic argument that h_{comp} is verifiable$_C$ but not decidable$_C$ in the limit by ideal scientists, so $C_{h_{comp}}$ is in $\Sigma_2^B - \Delta_2^B$. On the computational side, the basic result is:

Proposition 7.12

$C_{h_{comp}} \in \Sigma_3^A - \Pi_3^A$.

Proof: Rogers (1987): 327, theorem XVI. ■

Hence:

h_{comp} is empirically self-defeating in the limit for us.

For suppose that h_{comp} is true. Then human behavior is computable, and hence

our inductive behavior is computable. So by proposition 7.12, we cannot verify$_C$ h_{comp} in the limit. If we are Turing computable, we cannot verify this fact in the limit because we are Turing computable. This result is interesting in its own right, but also because it illustrates a case in which the characterization theorem is useful in obtaining a negative result. A direct, demonic argument is not at all easy to provide in this case, as the reader may check (cf. exercise 7.13). A similar argument can be given if we replace verification in the limit with gradual verification (cf. exercise 7.6).

8. The Computable Assessment of Uncomputable Theories

Imagine an ideal scientist who investigates a complete, deterministic theory of a given system. So complete is this theory that it determines uniquely every future observation. We can imagine an ideal scientist who derives (without effort) successive predictions from the theory and then compares each of them with what is actually observed. In this way, the theory is eventually rejected by the scientist if and only if it misses a prediction.

Now suppose the scientist is a computer. The picture just described of sequentially obtaining new predictions from the theory then presupposes that the theory is computable, in the sense that its prediction for time n is a recursive function of n. We now face an important methodological question: Do the predictions of a theory have to be derivable from the theory by computable means if a computable method is to decide the truth value of the theory in the limit? Or do there exist theories whose predictions cannot be derived by computable means that can nonetheless be decided in the limit or even refuted with certainty by computable inductive methods?

In this discussion we will consider only hypotheses that make determinate predictions about each datum that will appear in the limit. Say that h is *empirically complete*$_C$ just in case for some ε, $C_h = \{\varepsilon\}$. That is, there is a unique data stream for which h is correct. When $C_h = \{\varepsilon\}$, we know from the preceding characterization theorems that h is effectively verifiable in the limit just in case $\{\varepsilon\} \in \Sigma_2^A$. So we may think of the arithmetical complexity of $\{\varepsilon\}$ as the *inductive* or *scientific complexity* of h. But the hypothesis h also has a deductive complexity that corresponds to the computational difficulty of formally deriving predictions from h. We take the deductive complexity of h to be the arithmetical complexity of ε, where ε is identified with its graph: $\{(0, \varepsilon(0)), (1, \varepsilon(1)), \ldots\}$. It is easily verified that for any function ε, if ε is in Σ_1^A, then ε is partial recursive. If ε is also total, then ε is total recursive and $\varepsilon \in \Delta_1^A$. Since a data stream is assumed to be total, we have that $\varepsilon \in \Sigma_1^A$ if and only if $\varepsilon \in \Delta_1^A$.

To summarize, the deductive complexity of an empirically complete hypothesis h such that $C_h = \{\varepsilon\}$ is just the arithmetical complexity of ε, whereas the inductive or scientific complexity of h is just the arithmetical complexity of $\{\varepsilon\}$. It is striking that the essential difference between the nature of induction and the nature of deduction can be captured by the difference in arithmetical complexity between ε and $\{\varepsilon\}$! But the subtle difference in notation belies a world of difference between the two kinds of complexity, as will soon be apparent.

Now we may restate our question. How does the deductive complexity of h depend on the scientific complexity of h? In other words, how does the computational complexity of deriving predictions from h depend on the computational complexity of determining the truth value of h from data?

An ideal scientist can decide any empirically complete$_C$ h with one mind change starting with 1. We can see this two ways. First, we already know that each singleton $\{\varepsilon\}$ is Π_1^B, so the result follows from the ideal characterization theorem. More directly, consider an ideal scientist that checks (in an uncomputable manner) successive entries in ε against successive entries in the data stream, and says 1 until some discrepancy is found. On the other hand, a simple demonic argument shows that no ideal scientist can verify such a hypothesis with certainty. Thus we have:

Proposition 7.13

Let $h_C = \{\varepsilon\}$. Then

(a) *h is ideally refutable$_C$ with certainty.*

(b) *h is not ideally verifiable$_C$ with certainty (i.e., $\{\varepsilon\} \in \Pi_1^B - \Sigma_1^B$).* ■

A computable scientist can play the same game, provided that $\varepsilon \in \Sigma_1^A$ (and hence $\varepsilon \in \Delta_1^A$ since ε is total). For then, the computable scientist can compute the next position of ε and check the outcome of this computation against the data, waiting for a discrepancy. From these simple observations, we already know that:

Proposition 7.14

Let $h_C = \{\varepsilon\}$. Then if ε is recursive, then h is refutable$_C$ with certainty by a Turing machine (i.e., $\varepsilon \in \Sigma_1^A \Rightarrow \{\varepsilon\} \in \Pi_1^A$). ■

But in the other direction, we would expect that a highly intractable ε would pose problems for a computable scientist trying to investigate h. For suppose that ε is not recursive. Then there must be infinitely many predictions entailed by the theory $\{\varepsilon\}$ that an arbitrary, computable scientist fails to derive correctly, either by drawing a mistaken prediction that the theory does not actually entail or by failing to derive any prediction at all. For suppose otherwise. Then the procedure that fails to derive only finitely many predictions could be "patched" with a finite lookup table to provide a procedure that computes ε. If there is no background knowledge, then any underived prediction can be wrong for all the scientist knows. This leads one to suspect that h is not effectively refutable with certainty unless ε is computable. It is hard to see how to succeed otherwise, unless there is some miraculous method of rejecting a hypothesis if and only if it is false without deriving its successive predictions and checking them against

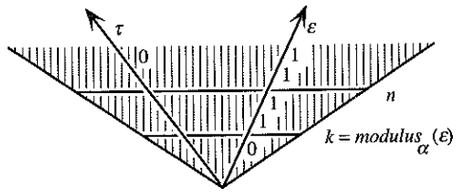


Figure 7.9

the data. As it turns out, the suspicion is correct in the special case in which h predicts only finitely many different kinds of outcomes. In fact, something much stronger is the case:

Proposition 7.15[10]

Let $h_C = \{\varepsilon\}$ and let $rng(\varepsilon)$ be finite. Then if h is verifiable$_C$ in the limit by a Turing machine, then ε is recursive (i.e., if $\{\varepsilon\} \in \Sigma_2^A$ then $\varepsilon \in \Delta_1^A$).

Proof: Suppose $rng(\varepsilon)$ is finite and recursive α verifies$_C$ h in the limit. Let $k = modulus_\alpha(\varepsilon)$. Thus, (*) for each $\tau \in rng(\varepsilon)^\omega$, we have that $\tau = \varepsilon \Leftrightarrow$ for each $m \geq k$, $\alpha(h, \varepsilon|m) = 1$. Think of $rng(\varepsilon)^*$ as an infinite but finitely branching tree in which ε is an infinite path. Each node in the tree may be labeled effectively with the conjecture $\alpha(h, e)$, where e is the unique path to that node from the root of the tree. Imagine that the tree is completely labeled. Then in virtue of (*), ε is the unique infinite path labeled with all 1s after position k. Say that a finite path e such that $lh(e) \leq m$ is *m-dead* just in case each extension of e of length m has a value less than 1 occurring on it between k and m. Then we have:

Lemma 7.16

If $n \geq k$ then for each $m \geq n$, $\varepsilon|n$ is not m-dead (Fig. 7.9). ■

Also, we have:

Lemma 7.17

If $n \geq k$ then there is some $m \geq n$ such that for each $e \in rng(\varepsilon)^n$ such that $e \neq \varepsilon|n$, e is m-dead.

Proof: Suppose $n \geq k$. Suppose for reductio that for all $m \geq n$, there is an $e \in rng(\varepsilon)^n$ *such that* $e \neq \varepsilon|n$ and e *is not m-dead.* Then for each m, there is an extension $e[m]$ of e of length m along which α conjectures only 1s for h. So $S = \{e[m]: m \geq n\}$ is an infinite, finitely branching tree. By König's lemma, S has an infinite path γ labeled entirely with 1s by α after position k. So by (*), $\gamma = \varepsilon$. Thus $e = \gamma|n = \varepsilon|n$, contrary to assumption. ■

[10] With Oliver Schulte.

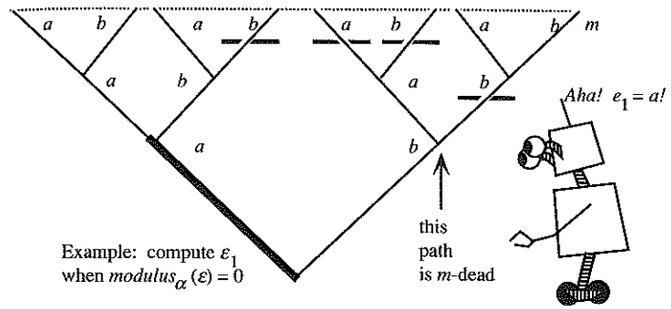


Figure 7.10

The procedure for computing ε is as follows. $M(n)$: Use α to effectively label the tree, from bottom to top, level by level, until for some m, the tree is labeled up to level m, and it is effectively verified (by exhaustion) that there is a unique path e of length n that is not m-dead. (By lemmas 1 and 2, $e = \varepsilon|n$). Return $e_n(=\varepsilon_n)$ (*Fig. 7.10*). ■

So now we know that if an empirically complete hypothesis h is effectively verifiable in the limit, then its predictions are computable, so long as it predicts only finitely many different kinds of events.

On the other hand, there is a hypothesis h predicting only finitely many different sorts of outcomes that can be refuted in the limit by a Turing machine even though its predictions are infinitely impossible to derive by computational means. At first blush, this seems like magic. By assumption, it is hopeless to expect a scientist to recover correctly all the predictions of h by computational means, and since there is no background knowledge, the demon can arrange the data to agree with any mistaken prediction the scientist derives. What could h be like? The answer is given in the following proof.

Proposition 7.18 (Hilbert and Bernays)[11]

There are C, h, ε such that $C_h = \{\varepsilon\}$ and for each n, $C_h \notin \Sigma_n^A$ but h is refutable$_C$ in the limit by a Turing machine (i.e., there is an ε such that for each n, $\varepsilon \notin \Sigma_n^A$ but $\{\varepsilon\} \in \Pi_2^A$).

Proof: Recall the universal relation $\mathcal{U}_n^A$ defined in section 7.4. Consider the characteristic function of $\mathcal{U}_n^A$:

$$v(\langle n, i, \bar{x}\rangle) = \begin{cases} 1 & \text{if } \mathcal{U}_n^A(i, \langle\rangle, \langle\bar{x}\rangle) \\ 0 & \text{otherwise.} \end{cases}$$

Recall from the hierarchy theorem that $\mathcal{D}_n \notin \Delta_n^A$. But v is at least as complex as

[11] Cf. Hinman (1978): 106–107.

$\mathcal{D}_n$ because for each z, $\mathcal{D}_n(z) \Leftrightarrow v(\langle n, i, \langle z\rangle\rangle) = 1$, where i is such that $\mathcal{D}_n(z) \Leftrightarrow \mathcal{U}_n^A(i, \langle\,\rangle, \langle z\rangle)$, by universality. It remains to show that $\{v\} \in \Pi_2^{\mathbb{A}}$. A simple induction shows that

$\varepsilon = v \Leftrightarrow$

(a) $\forall k \in \omega, \varepsilon_k \leq 1$ *and*

(b) $\forall i \in \omega, \bar{x} \in \omega^*[\varepsilon(\langle 1, i, \bar{x}\rangle) = 1 \Leftrightarrow \exists k \mathcal{T}(i, \langle\,\rangle, \langle\bar{x}\rangle, k)]$ *and*

(c) $\forall n, i \in \omega, \bar{x} \in \omega^*[\varepsilon(\langle n+1, i, \bar{x}\rangle) = 1$
$\Leftrightarrow \exists k$ *such that* $\varepsilon(\langle n, i, \langle\bar{x}, k\rangle\rangle = 0]$

Condition (a) says that ε is a characteristic function, condition (b) duplicates condition (1) of the definition of $\mathcal{U}_n^A$, and condition (c) duplicates condition (2) of the definition of $\mathcal{U}_n^A$. Hence, $\{v\} \in \Pi_2^{\mathbb{A}}$. ■

The curious hypothesis introduced in the proof of the proposition can be re-described in a more intuitive manner. Let V denote the set of all code numbers of truths of arithmetic. It turns out[12] that v is computationally equivalent to the problem of deciding V, so the hypothesis that a given black box yields a complete proof system for arithmetic is refutable in the limit by a computable method.

This point is relevant to a celebrated debate about the computational nature of mind. J. R. Lucas (1961) concludes from Gödel's theorem that humans cannot be computers since humans can always "intuit" the truth of the Gödel sentence for a consistent formal system, but each consistent formal system has a Gödel sentence that is true but that is not entailed by the system. In chapter 3, I argued that the real question is empirical, namely, whether human behavior is actually as claimed. The hypothesis involved in proposition 8.4 may be thought of as Lucas's hypothesis that a given human is an enumerator of the complete arithmetical truth. The correct resolution of the empirical issue is that the hypothesis is ideally refutable with certainty and effectively refutable in the limit.

The logical trick behind proposition 7.1 is that recursion on a function variable can simulate arbitrary alterations of numeric quantifiers, thereby building arbitrary arithmetical complexity into a single definition. To see how arbitrary arithmetical complexity can be unwound from the definition of v, consider the case of $v(\langle 4, i, \bar{x}\rangle)$. Each time the recursive clause (c) in the definition of v reduces the argument n by 1, another '$\exists\neg$' is added until the base case is reached, at which point '$\exists$' is added. Driving the negations through the sequence '$\exists\neg,\exists\neg,\exists\neg,\exists$' yields '$\exists\forall\exists\forall$'.

$$\begin{aligned} v(\langle 4, i, \bar{x}\rangle) = 1 &\Leftrightarrow \exists k_1 \neg v(\langle 3, i, \langle\bar{x}, k_1\rangle\rangle) = 1 \\ &\Leftrightarrow \exists k_1 \neg \exists k_2 \neg v(\langle 2, i, \langle\bar{x}, k_1, k_2\rangle\rangle) = 1 \\ &\Leftrightarrow \exists k_1 \neg \exists k_2 \neg \exists k_3 \neg v(\langle 1, i, \langle\bar{x}, k_1, k_2, k_3\rangle\rangle) = 1 \\ &\Leftrightarrow \exists k_1 \neg \exists k_2 \neg \exists k_3 \neg \exists k_4 \mathcal{T}(i, \langle\,\rangle, \langle\bar{x}, k_1, k_2, k_3\rangle, k_4) \\ &\Leftrightarrow \exists k_1 \forall k_2 \exists k_3 \forall k_4 \mathcal{T}(i, \langle\,\rangle, \langle\bar{x}, k_1, k_2, k_3\rangle, k_4). \end{aligned}$$

[12] Rogers (1987): 318, theorem X.

The distinction between the deductive and the inductive complexity of an empirically complete hypothesis coincides exactly with the logical distinction between explicit and implicit definability in arithmetic. An *explicit definition* of ε is a formula of arithmetic $\Phi(x, y)$ with free variables x and y such that $\Phi(x, y)$ is satisfied in arithmetic if and only if x and y are assigned to numbers n, m such that $\varepsilon(n) = m$. An *implicit definition* of ε is a formula of arithmetic $\Phi(\varepsilon)$ free only in function variable ε such that $\Phi(\varepsilon)$ is satisfied in arithmetic if and only if the variable ε is interpreted by the function ε. It turns out that ε is explicitly definable in arithmetic by a formula with $n-1$ quantifier alternations starting with $\forall[\exists]$ if and only if $\varepsilon \in \Pi_n^A[\Sigma_n^A]$. It also turns out that ε is implicitly definable in arithmetic by a formula with $n-1$ quantifier alternations starting with $\forall[\exists]$ if and only if $\{\varepsilon\} \in \Pi_n^A[\Sigma_n^A]$. When $C_h = \{\varepsilon\}$, the arithmetical complexity of $\{\varepsilon\}$ is just what I have called the inductive complexity of h, and the arithmetical complexity of ε is just what I have called the deductive complexity of h.

The original motivation for the study of implicit definability had nothing to do with the relationship between inductive and deductive complexity. It was intended, rather, as a precise analysis of the relative power of two different strategies for defining new concepts in arithmetic, with an eye toward the reduction of suspect notions in higher mathematics to the relatively clearer concept of the "natural" numbers. It is striking that the comparison of explicit and implicit complexity, a subject motivated by investigations into the foundations of mathematics, should have such direct and surprising implications for computable scientific inquiry.

If the range of predictions made by ε is finite, proposition 7.18 cannot be improved to yield a nonarithmetical ε such that $\{\varepsilon\}$ is refutable with certainty by a Turing machine, due to proposition 7.15. But what if the range of predictions made by ε is infinite? It seems odd to suppose that this might increase the power of computable inquiry, since it implies that there are more different sorts of predictions for the machine to worry about. But as a matter of fact, it can help, since there is an infinitely complex ε such that ε is refutable with certainty by a Turing machine. This seems even more magical than the preceding result, for now it is (infinitely) impossible for a machine to derive all predictions from ε, and ε could be wrong anywhere for all we know, but a Turing machine can eventually reject the hypothesis with certainty if and only if it is false! This result shows that the common picture of scientific inquiry, in which the scientist sequentially tests the predictions of a theory against the evidence, falls infinitely short of capturing the full potential of computable science.

Proposition 7.19[13]

There are C, h, ε such that $C_h = \{\varepsilon\}$ and for each n, $C_h \notin \Sigma_n^A$ but h is refutable$_C$ with certainty by a Turing machine (i.e., there is an ε such that for each n, $\varepsilon \notin \Sigma_n^A$ and $\{\varepsilon\} \in \Pi_1^A$).

[13] Hinman (1978): 106–107.

Proof: Let v be as in the proof of the preceding proposition. $\{v\} \in \Pi^A_2$, so there is a recursive relation $\mathcal{G}$ such that for all ε, $\varepsilon = v \Leftrightarrow \forall x \exists y \mathcal{G}(\varepsilon, x, y)$. Define:

$$\delta(x) = \langle v(x), \mu y \mathcal{G}(v, x, y)\rangle.$$

A Δ^A_n definition for δ would yield a Δ^A_n definition for v, since $v(x)$ can be recovered by decoding $\delta(x)$ and returning the first coordinate. Thus, for each n, $\delta \notin \Delta^A_n$. It remains only to show that $\{\delta\} \in \Pi^A_1$. Let $(\langle x, y\rangle)_1 = x$ and $(\langle x, y\rangle)_2 = y$. Let $(\alpha)_1$ denote the infinite sequence $(\alpha(0)_1, \alpha(1)_1, \alpha(2)_1, \ldots)$. It will now be shown that:

$$\varepsilon = \delta \Leftrightarrow$$

(a) $\forall x\, \mathcal{G}((\varepsilon)_1, x, (\varepsilon(x))_2)$ &

(b) $\forall x, y\, [y < (\varepsilon(x))_2 \Rightarrow \neg \mathcal{G}((\varepsilon)_1, x, y)]$.

which is a Π^A_1 definition for $\{\delta\}$. ($\Rightarrow$) Recall that $\forall x \exists y \mathcal{G}(v, x, y)$. Thus $\mathcal{G}$ holds if we choose the least such y: $\forall x \mathcal{G}(v, x, \mu y(\mathcal{G}(v, x, y))$. But $\mu y(\mathcal{G}(v, x, y) = \delta(x)_2$ and $v = (\varepsilon)_1$, so $\forall x \mathcal{G}((\varepsilon)_1, x, (\varepsilon(x))_2)$, which is (a). And (b) follows because $(\delta(x))_2$ is the least y such that $\mathcal{G}((\varepsilon)_1, x, y)$. ($\Leftarrow$) Suppose that (a) $\forall x \mathcal{G}((\varepsilon)_1, x, (\varepsilon(x))_2)$. Then $\forall x \exists y \mathcal{G}((\varepsilon)_1, x, y)$. Thus $(\varepsilon)_1 = v$. Assuming (b), we have that for all x, $(\varepsilon(x))_2$ is the least y such that $\mathcal{G}(v, x, y)$. Thus $\varepsilon = \delta$, as required. ■

The range of data stream δ may be infinite, because the minimization in the second coordinate has no bound. Thus there is no contradiction of our previous result that when $rng(\varepsilon)$ is finite, computable refutation with certainty implies that ε is recursive.

The results of this section provide a complete picture of how complex the deductive complexity of an empirically complete theory can be, given that the theory has some specified inductive complexity (*Fig. 7.11*).

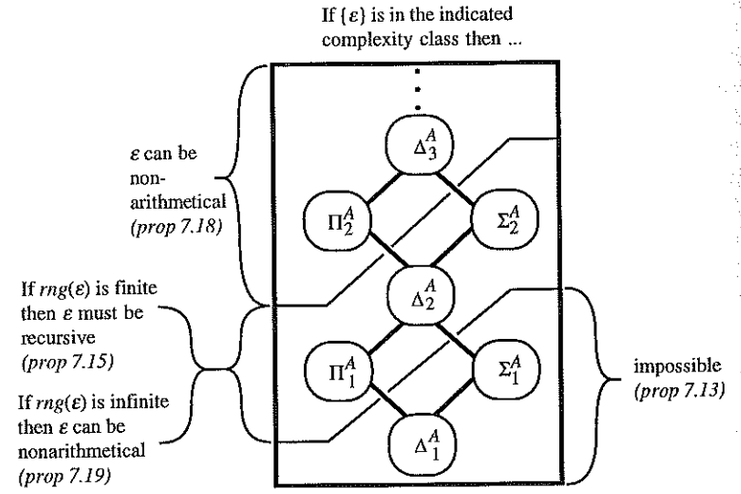


Figure 7.11

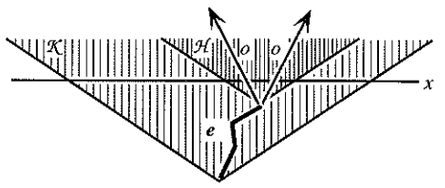


Figure 7.12

So far, we have considered only empirically complete hypotheses. Typically, a scientific theory entails predictions only in light of previous observations $\boldsymbol{e}$, and even then it may fail to make any prediction for a given time. Define h, $\boldsymbol{e} \vDash_C (n, o) \Leftrightarrow$ for each $\varepsilon \in C_h$ such that ε extends $\boldsymbol{e}$, $\varepsilon_n = o$ (*Fig. 7.12*). Define $PRED_C(h) = \{(\boldsymbol{e}, n, o)\colon h, \boldsymbol{e} \vDash_C (n, o)\}$. So if $(\boldsymbol{e}, n, o) \in PRED_C(h)$, then h predicts o at n given that $\boldsymbol{e}$ has been observed. Proposition 7.20, which is presented in *Figure 7.13*, summarizes the facts concerning both the empirically complete case and the general case. Each bound given in the table[14] can be shown best by means of an example. In Figure 7.13, assume that $\mathcal{K} = O^\omega$, where $O \subseteq \omega$.

For the proof of proposition 7.20, see exercise 7.12.

9. Ideal Norms and Computational Disasters

In the introduction to this chapter, I emphasized that bounded scientific norms are different from ideal norms. Recommendations that are good for ideal agents can be very bad for computationally bounded ones. A methodological norm is

Proposition 7.20 Given arithmetical bounds on C_h (cf. exercise 7.12)			Best arithmetical bound on $PRED_C(h)$			
			$\mathcal{H}$ is empirically complete		General case	
			O is finite	O is infinite	O is finite	O is infinite
certainty case	*a*	Δ^A_1	Impossible if $\lvert O\rvert > 1$		Δ^A_1	Π^A_1
	b	Σ^A_1			Π^A_1	Π^A_1
	c	Π^A_1	Δ^A_1	none	Σ^A_1	none
limiting case	*d*	Δ^A_2	Δ^A_1	none	Π^A_2	none
	e	Σ^A_2	Δ^A_1	none	Π^A_2	none
	f	Π^A_2	none	none	none	none

Figure 7.13

[14] Kelly and Schulte (1995).

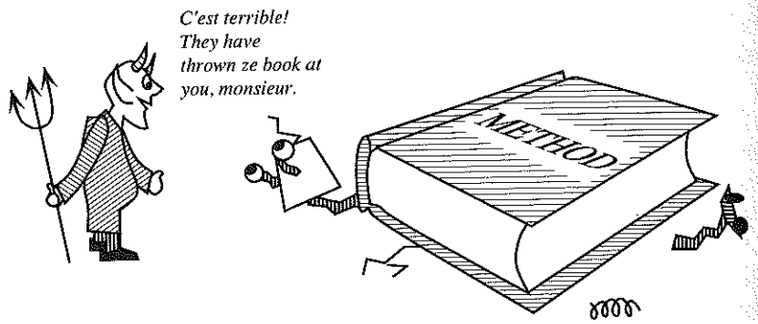


Figure 7.14

restrictive[15] for a class of agents just in case no agent in the class who obeys the norm can solve a given inductive problem that is solvable by a method of the same kind that violates the norm (*Fig. 7.14*).

Let's consider a concrete example of an ideal norm that is restrictive for computational agents. A standard, ideal requirement for science is that a hypothesis be rejected immediately when it is inconsistent with the available evidence. This requirement has been recommended, for example, by K. Popper.[16] We can think of it as a property of scientists, relative to a given inductive problem:

h is consistent$_C$ with $\boldsymbol{e}$ given $\mathcal{K} \Leftrightarrow \exists \varepsilon \in C_h \cap \mathcal{K} \cap [\boldsymbol{e}]$.

α is consistent for h given $\mathcal{K}$ $\Leftrightarrow$ ($\forall \boldsymbol{e} \in \omega^*$, h is not consistent$_C$ with $\boldsymbol{e}$ given $\mathcal{K} \Rightarrow \alpha(h, \boldsymbol{e}) = 0$).

Just a little reflection yields the following:

Proposition 7.21

Consistency is not restrictive for ideal methods in any of the paradigms of hypothesis assessment introduced in chapter 3. ■

Indeed, we saw in chapter 4 that consistency is necessary for ideal data-minimality and that each ideally solvable problem is solvable by a data-minimal method. Now let's consider the situation for computable methods. The relation of consistency may be nonrecursive. For example, recall the hypothesis $\mathcal{P}_K = \{\varepsilon\colon \varepsilon_0 \in K\}$, where K is the halting problem. If $x \notin K$, then (x) is inconsistent with $\mathcal{P}_K$, but no computer can decide whether $x \in K$. This suggests that consistency is restrictive for effective scientists. In fact, something far stronger

[15] Osherson, et al. (1986).
[16] Popper (1968).

is true. Say that

α is weakly conservative$_C$ for h given $\mathcal{K} \Leftrightarrow \forall \varepsilon \in \mathcal{K}$, $\varepsilon \in C_h \Rightarrow \alpha$ does not stabilize to 0 on h, ε

It is immediate that:

Proposition 7.22

$$\textit{If } \alpha \begin{bmatrix} \textit{verifies}_C \\ \textit{refutes}_C \\ \textit{decides}_C \end{bmatrix} h \begin{bmatrix} \textit{in the limit} \\ \textit{gradually} \end{bmatrix} \textit{ given } \mathcal{K},$$

then α *is weakly conservative$_C$ for h given $\mathcal{K}$.* ■

Then we have the following result:

Proposition 7.23

If $C_h = \{\varepsilon\}$ and α is consistent$_C$ for h given $\mathcal{K}$ and α is weakly conservative$_C$ for h given $\mathcal{K}$ and $\alpha \in \Sigma^A_n$, then $\varepsilon \in \Sigma^A_n$.

Proof: Given the conditions of the proposition, we claim:

$$\varepsilon_n = k \Leftrightarrow \exists \boldsymbol{e} \textit{ such that } lh(\boldsymbol{e}) \geq n \,\&\, \boldsymbol{e}_n = k \,\&\, \alpha(h, \boldsymbol{e}) \neq 0.$$

($\Rightarrow$) Suppose $\varepsilon_n = k$. Since α is weakly conservative, $\exists m \geq n$ such that $\alpha(h, \boldsymbol{e}) \neq 0$, so let $\boldsymbol{e} = \varepsilon|m$.

($\Leftarrow$) Suppose $\varepsilon_n \neq k$. Since α is consistent, for each $\boldsymbol{e}$ such that $h(\boldsymbol{e}) \geq n$ and $\boldsymbol{e}_n = k$, $\alpha(h, \boldsymbol{e}) = 0$.

Finally, since $\alpha \in \Sigma^A_n$, $\varepsilon \in \Sigma^A_n$. ■

Now recall the example $C_h = \{\delta\}$ of proposition 7.19. δ is not in any class Σ^A_n, but $\{\delta\} \in \Pi^A_1$. Then we have the following result, which illustrates the difficulties that accompany insistence on consistency, even for highly idealized, arithmetically definable methods.

Corollary 7.24

Let $C_h = \{\delta\}$, as in proposition 7.19. Then h is computably refutable with certainty but no consistent, arithmetically definable α can even gradually refute$_C$ or verify$_C$ h.

Proof: $\{\delta\} \in \Pi^A_1$ so h is computably refutable with certainty. Suppose that α gradually refutes$_C$ or verifies$_C$ h. Then by proposition 7.22, α is weakly conservative$_C$ for h given $\mathcal{K}$. Suppose α is also consistent$_C$ for h given $\mathcal{K}$. Then if $\alpha \in \Sigma^A_n$, so is δ, by proposition 7.23. But for each n, $\delta \notin \Sigma^A_n$. So for each n, $\alpha \notin \Sigma^A_n$. ■

Hence, insisting that a method notice immediately when data refutes a hypothesis entails a severe restriction on reliability. There is a hypothesis $C_h = \{\delta\}$ that a computable method can refute with certainty that no arithmetically definable method that notices refutation right away can even refute or verify gradually.

Nor can one require that a computable method be *maximally consistent*, in the sense that no method obeys consistency everywhere α does and somewhere else as well. Let α be an arithmetically definable method that verifies$_C$ h in the limit. Thus, α is not consistent. Then there is some e not extended by δ such that $\alpha(h, e) \neq 0$. Let α' be just like α except that $\alpha'(h, e) = 0$. α' is arithmetically definable (α' checks whether the current data is e, and then switches to a program for α if it is not) and α' is more consistent than α. But α' is still not entirely consistent, by corollary 7.24, so α' can be improved in the same way, and so on, forever. This situation is more the rule than the exception when we consider the reliability of computable methods satisfying ideal norms. Such norms are often restrictive and often fail to admit of optimal approximations among the reliable methods.

Intuitively, the problem with demanding consistency is that it requires a method to complete its consistency test before reading the next datum, whereas an arithmetically definable method may only be able to succeed by putting off the consistency test until after more data is read. It should not be supposed that this procrastination means that the consistency problem can be solved effectively in the limit but not in the short run. That would imply that $\delta \in \Sigma^A_2$, but in fact δ is not in any class Σ^A_n. The truth is more subtle. The computable method that refutes $C_h = \{\delta\}$ with certainty actually *uses* future data as an empirical oracle to help it decide whether past data is consistent with the hypothesis, so that not even a limiting, arithmetically definable method could decide whether past data is consistent with the hypothesis, given the past data alone (cf. exercise 7.15). This remarkable phenomenon underscores how formal and empirical problems blend into one another from a learning theoretic perspective.

10. Computable Inquiry

In this chapter, the difference between ideal and computationally bounded underdetermination in hypothesis assessment problems was characterized. The difference amounts to the difference in the base case between the arithmetical and the finite Borel hierarchies. We have seen that computational agents may treat formal difficulties as internal inductive inference problems solved in parallel with the ongoing external investigation. An important result of this chapter is that our ability to compute predictions from an empirical theory corresponds to its explicit definability in arithmetic, while the ability of a computer to determine the truth of the hypothesis in the limit corresponds to its implicit definability in arithmetic. This correspondence permits us to make use of standard results in mathematical logic to prove the surprising result that

theories whose predictions are infinitely impossible for a computer to derive can nonetheless be computably refutable with certainty. Without guidance from classical results in mathematical logic, such a possibility would be hard to imagine, much less to demonstrate.

This chapter extends the strong analogy between empirical and formal inquiry introduced in chapter 6. There are differences to be found between the two cases, and it is interesting to analyze just what they are, but the general picture is one of confluence rather than of sharp dichotomies between the respective logics of formal and empirical inquiry. That is not to say that ideal norms are good for computationally bounded agents, but rather that from the logical reliabilist point of view, inductive methodology for computationally bounded agents involves a smooth interleaving of formal and empirical considerations.

Exercises

7.1. Prove proposition 7.1 and use it to prove proposition 7.2.

7.2. Prove proposition 7.8 and show that the methods constructed in the proof of proposition 7.9 work as claimed.

7.3. Prove proposition 7.10.

7.4. (Putnam 1965) Recall the n-trial predicates defined in section 6.3. Define an effective, finite difference hierarchy for relations over the natural numbers, and characterize the n-trial predicates in terms of the cells of this hierarchy as follows: For each r such that $0 < r < 1$ and for each $n \in \omega$,

$$S \text{ is an } n\text{-trial predicate starting with } \begin{bmatrix} 0 \\ 1 \\ r \end{bmatrix} \Leftrightarrow S \in \begin{bmatrix} \Sigma^d_n \\ \Pi^d_n \\ \Delta^d_n \end{bmatrix}.$$

7.5. Prove proposition 7.7 following the strategy of proposition 7.5.

7.6. Show that h_{comp} is gradually empirically self-defeating for us.

7.7. Prove proposition 7.13.

7.8. Prove proposition 7.14.

7.9. Say that a method is *conservative*[17] with respect to C, $\mathcal{K}$, H just in case for each $h \in H$, it refuses to change its conjecture from 1 to 0 unless $C_h \cap \mathcal{K} = \emptyset$. Show that conservatism is restrictive for ideal verification in the limit.

[17] Osherson et al. (1986).

7.10. Prove proposition 7.21.

7.11. To apply the approach of chapter 5 to computable methods, we must introduce the notion of an *assessment* ***G*-reduction** as follows:

$$\mathcal{C} \leq_{\mathbf{G}} \mathcal{B} \Leftrightarrow \textit{there is a } \Phi \in \mathbf{G} \textit{ such that for each } \varepsilon \in \mathcal{N},\ h \in \omega,\ \mathcal{C}(\varepsilon, h) \Leftrightarrow \Phi(h, \varepsilon) \in \mathcal{B}.$$

Let Φ: $\omega \times \mathcal{N} \to \mathcal{N}$ be an operator. Define:

Φ *is recursive* $\Leftrightarrow$ *there is a partial recursive function* $\phi(h, e, x)$ *such that for all* $\varepsilon \in \mathcal{N}$, $h, x, y \in \omega$, $\Phi(h, \varepsilon) = y \Leftrightarrow$ *there exists* $e \subseteq \varepsilon$ *such that* $\phi(h, e, x) = y$.

Let ***Rec*** be the class of all recursive operators. Now prove analogues of propositions 5.3, and 5.4 for recursive assessment reducibility.

*7.12. Prove proposition 7.20 (*Fig. 7.13*).

Hints: (Oliver Schulte) to show that the infinite O, general case of (a) is optimal, let $O = \omega$ and let

$$\varepsilon \in C_h \Leftrightarrow [\varepsilon(\varepsilon(0)) \neq 0 \Rightarrow \phi_{\varepsilon(0)}(\varepsilon(0)) \textit{ halts within } \varepsilon(\varepsilon(0)) \textit{ steps}].$$

(Oliver Schulte) To show that the finite O, general case of (b) is optimal, let $O = \{0, 1\}$ and define the hypothesis:

$$\varepsilon \in C_h \Leftrightarrow \exists k[(\forall k' < k, \varepsilon(k') = 1) \textit{ and } \varepsilon(k) = 0 \text{ and } (\varepsilon(k) + 1 \neq 0 \Rightarrow k \in K)].$$

To show that the finite O, general case of (c) is optimal, let $O = \{0, 1\}$, let S be Π_2^A-complete and let $R \in \Sigma_2^A$ be such that for all n, $n \in S \Leftrightarrow \forall z\ R(n, z)$. Now define the hypothesis:

$$\varepsilon \in C_h \Leftrightarrow \forall x[x \in \bar{K} \quad \textit{or} \quad \varepsilon(x) = 0].$$

(Oliver Schulte) To show that the finite O, general case of (d) is optimal, define the hypothesis:

$$\varepsilon \in C_h \Leftrightarrow \textit{if } [\exists n\ (1^n)^*0^*1 \textit{ is extended by } \varepsilon] \textit{ then } [\exists n \exists m\ (1^n)^*0^*(1^m)^*0 \textit{ is extended by } \varepsilon \textit{ and } \neg R(n, m)].$$

The upper bound given in the infinite O, general case of (e) can be established as follows: let α verify$_C$ h in the limit. If $(e, n, o) \notin PRED_C(h)$, then α stabilizes to 1 on some infinite extension ε of e such that $\varepsilon_n \neq o$. Use this fact to get a Σ_2^B definition of $\overline{PRED_C(h)}$, from which the result follows. The rest of the proposition follows from elementary logical dependencies in the table or from results similar to those already proved in section 7.8.

*7.13. Take up the challenge of providing a direct demonic argument to show that h_{comp} is not gradually refutable$_C$ in the limit by a Turing machine (cf. proposition 7.12), from which it would follow that $C_{h_{comp}} \notin \Pi_3^A$. What is required is a demon who

presents a nonrecursive data stream if and only if the given method α approaches 0. Such a demon exists by proposition 5.6.

7.14. Develop the generalized setting for hypothesis assessment described in footnote 3 and characterize the various senses of success in this setting.

*7.15. Using the implicit definition of the hypothesis $C_h = \{\delta\}$ of proposition 7.19, explain how the obvious method for refuting this hypothesis with certainty uses the information in future data to determine if past data is inconsistent with the hypothesis.