Algorithmic Complexity and Triviality

Marcus Anthony Triplett
Supervisor: André Nies

October 27, 2014

Contents

1 Introduction

> *"By ratiocination, I mean* computation*. Now to compute, is either to collect the sum of many things that are added together, or to know what remains when one thing is taken out of another.* Ratiocination*, therefore, is the same with* Addition *or* Substraction*; and if any man adde* Multiplication *and* Division*, I will not be against it, seeing Multiplication is nothing but Addition of equals one to another, and Division nothing but a Substraction of equals one from another, as often as is possible. So that all ratiocination is comprehended in these two operations of the minde, Addition and Substraction."*
>
> - Hobbes, 1651

This report consists of a systematic study of some topics in both traditional and modern aspects of computability theory. Computability theory grew out of the foundational crisis at the turn of the 20th century in attempts to formalise the notion of an *algorithm*. A number of formalisations were created, most notable among them the lambda calculus of Alonzo Church, Kurt Gödel's recursive function theory, and the logical computing machines of Alan Turing; the latter now being known as Turing machines. These formalisations were constructed for different purposes, but in general were motivated by David Hilbert's program for the foundations of mathematics. In 1931 [8] Gödel was able to show using his recursive functions that the theory of Peano arithmetic is incomplete, and Turing in 1936 [29] used his machines to prove the unsolvability of the Entscheidungsproblem.

Following Church, Gödel, and Turing, two major figures emerged as leaders in computability. Stephen Kleene, a student of Church, proved many fundamental results for partial recursive functions and recursively enumerable sets, including the recursion theorem and various normal form theorems. It is largely due to Kleene that the field of computability was known as recursion theory during the 20th century [26]. Meanwhile Emile Post, already having discovered independently of Gödel the completeness theorem for first-order logic and undecidability in Principia Mathematica (see [28]), began research into degree theory. His work in this area established computability as an independent mathematical enterprise. Post also demonstrated some connections between classical mathematics and computability in his well known result establishing the undecidability of the word problem for semi-groups.

Historically, the theory of computation is full of limitative results, and computability theory is concerned specifically with the limits of computation. This is a theory of *what is computable*. By the Church-Turing thesis, it is a theory of what can and cannot be calculated by algorithm *in principle*. We do not consider constraints on time or space, so that what is proven incomputable is in fact incalculable by any known device, regardless of its processing power or memory. The division between computable and incomputable is too coarse, however. The difficulty of computational tasks is much more complex than a

mere dichotomy between 'possible' and 'impossible'. For instance, we will see that despite the incomputability of two sets A and B by Turing machine, the set A can be *more* incomputable than B. By this we mean that one set can be *reduced* to the other, while the other cannot be reduced to the one. This reveals an alternative motivation for the theory of computability. How does one understand and classify in some kind of hierarchical structure the difficulty of calculating sets of numbers, and of computing functions? More generally, how does one characterise the *complexity* of sets of natural numbers?

1.1 The complexity of sets of natural numbers

We will study a variety of complexity notions. Nies in [19] divides these into the following classes.

Descriptive complexity. We wish to understand how difficult a set is to describe. Section 2.3 contains some theorems regarding the first-order definability of properties of c.e. sets. For instance, by Theorem 2.25 effective inseparability is characterised by a Σ^0_3 formula over $\mathcal{E}$, the lattice of computably enumerable sets under inclusion. The representability of a class of sets in a first-order language reveals that the defining attribute common among them, such as effective inseparability or creativity, cannot be too complicated to describe. The arithmetical hierarchy (see Section 4.1) similarly calibrates the difficulty of describing sets, now in terms of the quantifier complexity. Although most sets are not arithmetical, the hierarchy provides a useful tool for the classification of sets and predicates one may study in formal arithmetic.

In a much broader sense, there are objects which are *easy* to describe and objects which are *difficult* to describe. In case the objects in question are finite strings one may define a complexity measure given by the length of the shortest description for the string. By a description we mean some input to a machine which generates the desired string accordingly. The strings capable of being described by short inputs can be viewed as simpler than those which cannot. This idea, whose origins lie with Kolmogorov [17], Chaitin [1], and Solomonoff [27], admits a formal definition for the randomness of finite strings. Namely that a string is random if it does not have a description shorter than the length of the string itself.

We may encode a set of natural numbers as an infinite binary string. Now we perform the same analysis on the *initial segments* of the infinite string, whereupon a set is *trivial* if each of its initial segments are easy to describe, and *random* if they are difficult to describe. By Corollary 5.14 there are trivial sets which are still incomputable, but by Theorem 5.15 no trivial set could be weak truth-table complete. In fact, no trivial set could be Turing complete.

Relative computational complexity. We wish to understand which of two sets is more useful. By this we mean how can we *reduce* one set to another, under which reducibilites is this possible, and is the reducibility necessarily strict? The modern concept of reducibility first arose in Turing's analysis of computation by means of oracle machines. Taking a set A as an oracle, one asks what can be computed by a machine Φ with access to it. If a set B may be computed by

such a machine with oracle A, then B is said to be Turing reducible to A. Many variations of this kind of reduction exist, and each induces a partial order on $2^{\mathbb{N}}$. By considering the equivalence classes of $2^{\mathbb{N}}$ modulo some reducibility, one obtains a degree structure formally defining classes of computational complexity. In terms of information content, the sets of minimum degree have none. On the other hand, $\emptyset^n$ is Σ^0_n-complete by Post's Theorem 4.9, and thus contains the maximum possible information content in the universe of Σ^0_n sets. Restricting our reduction to a computable function gives rise to the many-one degrees, wherein Post's notion of a creative set captures Σ^0_1-completeness by Theorem 2.13.

Absolute computational complexity. We wish to understand the computational strength or weakness of a set in an absolute sense. That is, without emphasising reductions between arbitrary sets, but perhaps by fixing some particular class of sets and using this as a measure for the information content of others. This is precisely what the notion of *lowness* captures. By Theorem 4.14 low sets exist. This gives a second solution to Post's problem, as discussed in Section 3.2. The absolute computational complexity of a set can also be calibrated by various other notions, such as computable domination. Now we ask *does an oracle help to compute quickly growing functions?* By Theorem 4.25 the Δ^0_2 degrees are hyperimmune, indicating that no incomputable Δ^0_2 set is capable of being computably dominated. On the other hand, by Theorem 4.26 any computably dominated set is already generalised low_2.

1.2 Sources

Much of the content of this report comes from Odifreddi's *Classical Recursion Theory* [20], Soare's *Recursively Enumerable Sets and Degrees* [25], Nies' *Computability and Randomness* [19], and Cooper's *Computability Theory* [2]. For an overview of the history of computability up to the year 1979 the pre-eminent source is Kleene himself in *Origins of Recursive Function Theory* [15]. Soare in *The history and concept of computability* [26] also provides an excellent account, especially regarding nomenclature.

1.3 Preliminaries

We assume a fixed enumeration of Turing Machines $\{\Phi_i\}_{i\in\mathbb{N}}$ given by some effective Gödel numbering. If a Turing Machine Φ_i halts on input x we write $\Phi_i(x)\downarrow$, otherwise we write $\Phi_i(x)\uparrow$. Similarly if a function f is defined on input x we write $f(x)\downarrow$, otherwise $f(x)\uparrow$. We denote by W_i the domain of the i-th Turing Machine; i.e., $W_i = \{x : \Phi_i(x)\downarrow\}$. Let $K \subseteq \mathbb{N}$. By a *partial computable function* we mean a function $f : K \to \mathbb{N}$ such that there is a Turing Machine Φ_i with $f(x) = \Phi_i(x)$ for all $x \in K$. If f is total we say f is a *computable function*. We say that a set A is *computably enumerable* (briefly, c.e.) if A is the domain of some partial computable function f. Equivalently A is c.e. if A is the range of a partial computable function. If the characteristic function χ_A

for A is computable we say that the set A is computable. The following lemma is used extensively:

Lemma 1.1 (Kleene [14])**.** *Let A be a set. Then A is computable iff A and $\overline{A}$ are computably enumerable.*

The following fundamental result was proven by Kleene in [13]. It may be extended in a number of ways, as in the two succeeding theorems.

Theorem 1.2 (The recursion theorem)**.** *For any computable function $f : \mathbb{N} \to \mathbb{N}$ there is $e \in \mathbb{N}$ such that $\Phi_{f(e)} = \Phi_e$.*

Theorem 1.3 (The double recursion theorem)**.** *For any two computable functions $f, g : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ there exist $a, b \in \mathbb{N}$ such that $\Phi_a = \Phi_{f(a,b)}$ and $\Phi_b = \Phi_{g(a,b)}$.*

Theorem 1.4 (The recursion theorem with parameters)**.** *For any computable function $f : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ there is a computable function $k : \mathbb{N} \to \mathbb{N}$ such that $\Phi_{k(y)} = \Phi_{f(k(y),y)}$.*

It is useful to specify how far along a computation may be. We write $\Phi_i(x)[s]$ for the Turing Machine Φ_i performing a computation with input x, limited to s computational steps. If $\Phi_i(x)$ halts in no more than s steps then $\Phi_i(x)[s]\downarrow$.

During constructions, such as those done in Section 3.3 we often use the *Cantor pairing function* to encode tuples of natural numbers.

Definition 1.5. *The Cantor pairing function is a bijection $\langle \cdot, \cdot \rangle : \mathbb{N}^2 \to \mathbb{N}$ given by*

$$\langle x, y \rangle = \frac{1}{2}(x + y)(x + y + 1) + y.$$

This may be extended to larger tuples by setting

$$\langle x_1, \ldots, x_n, x_{n+1} \rangle = \langle x_1, \ldots, \langle x_n, x_{n+1} \rangle \rangle.$$

We also use a notational device used to define recursive functions.

Definition 1.6 (Minimisation)**.** *The μ operator is defined by $\mu x.Fx =$* the least x such that Fx*, where F is some unary predicate over $\mathbb{N}$.*

2 Computably enumerable sets

In 1928 David Hilbert posed the *Entscheidungsproblem* – the *decision problem* – asking for an algorithm which determines in a finite number of steps, given as input a formula from first-order logic, whether that formula is valid or not. Alan Turing resolved this challenge negatively by *reducing* what became known as the halting problem to the Entscheidungsproblem. The halting problem is a decision problem asking whether, given an input consisting of a Turing machine

Φ and an input x for Φ, the machine Φ halts on x. The set of all pairs $\langle \Phi, x \rangle$ such that $\Phi(x) \downarrow$ can easily be seen to be computably enumerable. We may simply simulate in parallel each $\langle \Phi_i, x_i \rangle$, and take note of whenever such a pair halts. It is now common place to restrict the halting problem to deciding whether a machine Φ halts on its own code.

Definition 2.1. *We denote by $\emptyset'$ the halting problem. That is*

$$\emptyset' = \{x : \Phi_x(x) \downarrow\}.$$

It was shown in Turing's seminal 1936 paper *On Computable Numbers, with an application to the Entscheidungsproblem* [29] that this set is not computable.

Theorem 2.2. *$\emptyset'$ is incomputable.*

Proof. Assume for contradiction that $\emptyset'$ is computable. Then $\overline{\emptyset'}$ is c.e., and hence $\overline{\emptyset'} = W_k$ for some $k \in \mathbb{N}$. If $k \in W_k$ then $k \in \overline{\emptyset'}$ and $k \in \emptyset'$, which is impossible. Similarly if $k \notin W_k$ then $k \notin \emptyset'$ and $k \in \overline{\emptyset'}$. In either case we arrive at a contradiction, so no such k can exist. □

We begin by studying two varieties of computably enumerable sets which are incomputable: the *creative* sets, and the *simple* sets.

2.1 Creative sets

Emile Post introduced in [22] the notion of a *creative set*, intended to capture the underlying feature which gives rise to impossibility phenomena such as Gödel's incompleteness theorems and Theorem 2.2:

> "The conclusion is unescapable that even for such a fixed, well defined body of mathematical propositions, mathematical thinking is, and must remain, essentially creative. To the writer's mind, this conclusion must inevitably result in at least a partial reversal of the entire axiomatic trend of the late nineteenth and early twentieth centuries, with a return to meaning and truth as being of the essence of mathematics." ([22] p. 295)

Definition 2.3. *A set S is* productive *if there is a computable function p such that for any c.e. set W_e*

$$W_e \subseteq S \Rightarrow p(e) \in S - W_e.$$

A set is creative *if it is c.e. and coproductive.*

Creative sets are those which can be shown effectively (using their productive functions) to be incomputable. The productive function $p(e)$ guarantees that $\overline{C}$ cannot have a c.e. index.

Proposition 2.4. *$\emptyset'$ is creative via $p(x) = x$.*

Proof. Suppose $W_e \subseteq \emptyset'$. If $p(e) \in \emptyset'$ then $e \in W_e$ and hence $p(e) \notin \emptyset' - W_e$. On the other hand if $p(e) \notin \emptyset'$ then $p(e) \notin \emptyset' - W_e$. □

Here we give the first definition of a reduction. Broadly speaking, a *reduction* of one set, A, to another, B, is some method of deciding membership for A by appealing to answers from the set B. If the method can be carried out by Turing machines then we have an algorithm for converting A-problems in to B-problems. Since we are only interested in *computability*, we do not consider whether the reduction can, for instance, be computed in polynomial time, or can be computed subject to some memory constraint.

Definition 2.5. *Let A and B be sets. A is* many-one reducible *to B, written $A \leq_m B$, if there is a computable function f such that $x \in A \Leftrightarrow f(x) \in B$. If we can choose the reduction f to be injective we say that A is* one-one reducible *to B and write $A \leq_1 B$.*

Many-one reduction is a *strong* reducibility notion, preserving computability and enumerability. In subsequent sections we will study weaker reducibility notions which don't require a reduction to come from a computable function. In these cases a set A being weakly reducible to a c.e. set does not guarantee that A will be c.e.

Proposition 2.6. *If $A \leq_m B$ and B is c.e., then A is c.e. Moreover if B is computable, then A is computable.*

Proof. Suppose $A \leq_m B$ via f, and that B is c.e. Then $B = W_e$ for some $e \in \mathbb{N}$, so $A = \{f(w) : w \in W_e\}$ and hence is c.e. Now if B is computable then $\overline{B} = W_r$ for some $r \in \mathbb{N}$. Thus $\overline{A}$ is c.e., and so A is computable. □

Proposition 2.7. *Let B be c.e. If $A \leq_m B$ and A is creative, then B is creative.*

Proof. We show that B is coproductive. Let p be the productive function for A, and suppose $A \leq_m B$ via f computable. We can effectively find an index $i(e)$ such that $W_{i(e)} = f^{-1}(W_e)$, uniformly in e. Thus if $W_e \subseteq \overline{B}$ then $W_{i(e)} \subseteq \overline{A}$. Since A is creative one has $p(i(e)) \in \overline{A} - W_{i(e)}$. Then $f(p(i(e))) \in \overline{B} - W_e$. Hence $f \circ p \circ i$ is a productive function for B. □

Definition 2.8. *Let $K_0 = \{(x, y) : \Phi_y(x) \downarrow\}$.*

By Proposition 2.7 K_0 is also creative, for $x \in \emptyset'$ iff $(x, x) \in K_0$. Hence the map $x \mapsto (x, x)$ is a many-one reduction.

Example 2.9. $K_0 \leq_m \mathsf{Tot} := \{x : \Phi_x(y) \downarrow \text{ for all } y \in \mathbb{N}\}$.

Proof. Let ψ be the partial computable function which on input (x, y) first simulates the computation $\Phi_y(x)$ and, if it converges, returns 1. Then

$$(x, y) \in K_0 \Rightarrow \Phi_y(x) \downarrow \Rightarrow \Phi_{\psi((x,y))}(z) = 1 \text{ for all } z \in \mathbb{N} \Rightarrow \psi((x, y)) \in Tot$$
$$(x, y) \notin K_0 \Rightarrow \Phi_y(x) \uparrow \Rightarrow \Phi_{\psi((x,y))}((x, y)) \uparrow \Rightarrow \psi((x, y)) \notin Tot$$

and the proposition holds. □

Remark 2.10. *That a productive function is computable allows us to devise an algorithm for generating an infinite c.e. set in the complement of any creative set. Let C be creative via p. The process is as follows.*

- *Compute an index $e_0 \in \mathbb{N}$ such that $W_{e_0} = \emptyset$. Since $W_{e_0} = \emptyset \subseteq \overline{C}$ one has $p(e_0) \in \overline{C}$.*
- *Given a finite set $\{c_1, \ldots, c_n\} \subseteq \overline{C}$ compute an index $e_n \in \mathbb{N}$ such that $W_{e_n} = \{c_1, \ldots, c_n\}$. Note that for any finite set one can effectively determine a c.e. index for it. Hence $W_{e_n} \subseteq \overline{C}$ and thus via the creativity of C one has $\{c_1, \ldots, c_n, p(e_n)\} \subseteq \overline{C}$, where $p(e_n) \neq c_i$ for $i = 1, \ldots, n$.*

This construction computably enumerates an infinite subset

$$\{p(e_0), p(e_1), p(e_2), \ldots\}$$

of $\overline{C}$. By the following proposition this method may be strengthened to finding an infinite computable *subset of $\overline{C}$.*

Proposition 2.11. *Every infinite c.e. set S contains an infinite computable subset.*

Proof. Let ψ be the computable function such that $\mathsf{ran}(\psi) = S$. Define f by the following recursive scheme:

$$\begin{aligned} f(0) &= \psi(0) \\ f(n+1) &= \psi(x), \text{ where } x \text{ is least s.t. } \psi(x) > f(n). \end{aligned}$$

Let $V = \mathsf{ran}(f)$. Note that V is infinite since S is infinite. To compute whether $x \in V$ it suffices to check if $x \in \{f(0), f(1), \ldots, f(x)\}$. □

Every reducibility notion induces a corresponding *completeness* notion for computably enumerable sets. Sets which are complete for a reducibility are maximally incomputable c.e. sets, in the sense that (i) they are c.e. and incomputable, and (ii) they are above any other c.e. sets with respect to that reducibility. For now will restrict our consideration to *m-completeness*.

Definition 2.12. *Say that a c.e. set A is* m-complete *if for every c.e. set B one has $B \leq_m A$.*

Thus should one be capable of computing an m-complete set, one could further compute every (incomputable) c.e. set. Clearly then, m-complete sets are incomputable themselves. Post's notion of a creative set in fact captures precisely the m-complete sets.

Theorem 2.13. *A set is creative iff it is m-complete.*

Proof. "⇒" Let C be creative via f, and let A be c.e. Let g be a partial computable function such that

$$W_{g(x,y)} = \begin{cases} \{f(x)\} & \text{if } y \in A \\ \emptyset & \text{otherwise.} \end{cases}$$

By the recursion theorem with parameters 1.4 there is a computable function k with $W_{g(k(y),y)} = W_{k(y)}$. Then

$$y \in A \Rightarrow W_{k(y)} = \{f(k(y))\} \Rightarrow W_{k(y)} \not\subseteq \overline{C} \Rightarrow f(k(y)) \in C, \text{ and}$$
$$y \notin A \Rightarrow W_{k(y)} = \emptyset \Rightarrow W_{k(y)} \subseteq \overline{C} \Rightarrow f(k(y)) \in \overline{C} - W_{k(y)} \Rightarrow f(k(y)) \notin C.$$

Hence $A \leq_m C$.

"$\Leftarrow$" Suppose C is m-complete. Then $\emptyset' \leq_m C$. By Proposition 2.4 $\emptyset'$ is creative, and so C is creative by Proposition 2.7. □

As a corollary of Theorem 2.13 we learn that the halting problem $\emptyset'$ is m-complete, and thus maximally unsolvable relative to c.e. sets.

Let $r \in \{1, m\}$. Say that sets A and B are *r-equivalent* if $A \leq_r B$ and $B \leq_r A$. In this case we write $A \equiv_r B$. We note the following useful theorem, asserting that for any two 1-equivalent sets there is a computable function permuting one into the other. This allows us to prove many results about creative sets in general by considering just one creative set (we do precisely this in Section 2.3). A proof can be found in [25] I.5.4.

Theorem 2.14 (Myhill's Isomorphism Theorem)**.** *Let $A, B \subseteq \mathbb{N}$. If $A \equiv_1 B$ then there exists a computable permutation π such that $\pi(A) = B$.*

2.2 Simple sets

Another class of incomputable c.e. sets distinct from the creative sets are the *simple sets*.

Definition 2.15. *An infinite set S is* immune *if it does not contain an infinite c.e. subset. A set S is* simple *if it is c.e. and co-immune.*

In other words, simple sets are so large that they intersect every c.e. set. Any simple set S cannot be computable, otherwise S would intersect its complement.

Proposition 2.16. *No simple set is creative.*

Proof. Let C be creative. By Remark 2.10 $\overline{C}$ contains an infinite c.e. subset W_e. But then if C were simple $C \cap W_e \neq \emptyset$, which is absurd. □

As an example we give an easy recharacterisation of simple sets.

Example 2.17. *A co-infinite c.e. set S is simple iff it has no co-infinite computable superset.*

Proof. "$\Rightarrow$" Assume for contradiction that S has a co-infinite computable superset H. Then $\overline{H}$ is infinite and c.e., and so by Proposition 2.11 contains an infinite computable subset. But as $S \subseteq H$, one has $\overline{H} \subseteq \overline{S}$, so that $\overline{S}$ has an

infinite computable subset - contradicting the fact is S was co-immune.

"$\Leftarrow$" Assume S is not simple. Then $\overline{S}$ contains an infinite c.e. subset. In fact $\overline{S}$ contains an infinite computable subset H. But then $H \subseteq \overline{S}$, so $S \subseteq \overline{H}$ with $\overline{H}$ computable and co-infinite. □

We present in this report a number of proofs for the existence of simple sets. The first is Post's original construction. In Section 4.2 we build a *low* simple set in order to resolve Post's problem, and in Section 5 we give a construction of a simple set due to Kolmogorov (the set of compressible strings is simple), and another using a cost function construction.

Theorem 2.18 (Post [22])**.** *There is a simple set.*

Proof. Let ψ be the partial computable function which, on input x, returns the first element no less than $2x$ enumerated into W_x. Let S be the range of ψ. Then S is c.e., so it remains to check that S is co-infinite. But this follows by construction since the number of elements in S less than $2x$ is at most x. □

2.3 Lattice-theoretic properties of c.e. sets

We denote by $\mathcal{E}$ the lattice of computably enumerable sets under inclusion:

$$\mathcal{E} = \langle \{W_e\}_{e \in \mathbb{N}}, \cup, \cap, \subseteq, \emptyset, \mathbb{N} \rangle.$$

A property P of sets is *definable in* $\mathcal{E}$ if there is some first-order formula φ over the signature $\langle \cup, \cap, \subseteq, \emptyset, \mathbb{N} \rangle$ with one free variable such that a set $X \in \mathcal{E}$ has property P iff $\mathcal{E} \models \varphi(X)$. We say that a property is *lattice theoretic* if it is definable in $\mathcal{E}$.

It was shown by Leo Harrington that creativity is lattice-theoretic. It seems that the first published proof of this was in [25, p. 399]. Soare and Harrington together in a series of papers (e.g. [9–12]) in the 1990's developed a significant body of work regarding automorphisms and definability in the lattice. For instance, it is shown in [9] that there is a lattice-theoretic property guaranteeing that a set is both incomputable and Turing incomplete (this term will be defined and explored from Section 3.2 onwards). It is also shown in [12] that there is a lattice-theoretic property implying simplicity and Turing completeness. Here we will recreate the proof of Harrington's theorem about creative sets, and then modify it to give a lattice-theoretic characterisation of another property, namely effective inseparability.

Let X, Y be c.e. sets with $X \subseteq Y$. If $Y - X$ is c.e. we write $X \sqsubset Y$. It is important to note that if $X \not\sqsubset Y$, then there is an infinite stream of elements which are first enumerated in Y, and then later appear also in X. This will act as a resource in the upcoming construction.

Theorem 2.19 (Lattice-theoretic characterisation of creativity, Harrington)**.** *Creativity is lattice-theoretic. In particular, if C is c.e. then C is creative iff*

$$\exists F \; \forall Z \; \exists R \; [R \cap F \not\subseteq R \wedge R \cap C = R \cap Z] \tag{1}$$

where quantification is over $\mathcal{E}$.

Proof. "⇒" We show that $\widehat{C} = \{\langle x, e\rangle : \langle x, e\rangle \in W_e\}$ is creative, satisfies (1) and hence obtain, via Myhill's Isomorphism Theorem 2.14, that every creative set satisfies (1). Indeed, if $\widehat{C}$ satisfies (1) and C is any other creative set, there is a computable permutation π such that $\pi(\widehat{C}) = C$. Applying the permutation across $\mathbb{N}$ shows that C satisfies (1) also.

To see that $\widehat{C}$ is 1-complete, let e be an index such that $W_e = \emptyset' \times \mathbb{N}$. Then $x \in \emptyset' \Leftrightarrow \langle x, e\rangle \in W_e \Leftrightarrow \langle x, e\rangle \in \widehat{C}$. The map $x \mapsto \langle x, e\rangle$ gives the required 1-reduction.

Next, let $F = \emptyset' \times \mathbb{N}$. Given $Z = W_k$, take R to be the computable set $\mathbb{N} \times \{k\}$. Then $R - F = \overline{\emptyset'} \times \{k\}$ which is not c.e., and so $R \cap F \not\sqsubseteq R$.

For each $\langle x, k\rangle \in R$

$$\langle x, k\rangle \in \widehat{C} \Leftrightarrow \langle x, k\rangle \in W_k \Leftrightarrow \langle x, k\rangle \in Z.$$

Hence $R \cap \widehat{C} = R \cap Z$.

"⇐" Suppose C satisfies (1) via F. We construct Z in stages, giving an infinite number of c.e candidate witnesses R_i for Z in expression (1). For each such witness, we define a potential productive function p_i. In the limit some R_i will in fact be the correct witness, thereby yielding a total p_i.

Construction. Stage $s = 0$. Fix a one-one enumeration of F, and for each e an enumeration of W_e. Let $Z_0 = \emptyset$ and declare $p_{i,0}(e)$ undefined for each i, e.

Stage $s > 0$. Let i be least such that

i. If $t < s$ is the largest stage such that $t = 0$ or p_i had a new value defined at stage t then

$$(R_i \cap C) \upharpoonright_t [s] = (R_i \cap Z) \upharpoonright_t [s].$$

ii. There is an $x \in R_{i,s-1} \cap F_{at\ s}$

If i exists define $p_{i,s}(e) = x$, where $e = \mu y.p_{i,s-1}(y)\uparrow$. If x is enumerated into $W_{e,u}$ for some stage $u \geq s$, put also x into Z_u. This completes the construction.

Verification.

- There is some total p_i

  Let i be least such that R_i is the correct witness for Z in (1). If p_k is total for some $k < i$ the claim holds. So suppose for each $k < i$ that p_k is not total. Then there is some stage s_0 such that we do not choose any values of p_k with $k < i$ at a stage $s \geq s_0$, since only finitely many values of p_k are defined. There is an infinite flow of elements from R_i into F since $R_i \cap F \not\sqsubseteq R_i$. Hence at infinitely many stages s we choose x from $R_{i,s-1} \cap F_{at\ s}$. This ensures that p_i is total because we always define the least element on which p_i is undefined.

- C is creative via the above p_i

  Suppose $p_i(e) \in W_e \cup C$. We want to ensure that $W_e \cap C \neq \emptyset$. We have $R_i \cap C = R_i \cap Z$ as we always confirm i in the construction. Given e, we define $p_i(e) = x$ at some stage s. Now x is not defined as a value of any $p_k(m)$ at stage $s-1$ since $x \in F_{at\ s}$. Also x is not defined as a value of any $p_k(m)$ at a stage $s' > s$ since we chose a one-one enumeration of F, so x cannot be an element of $F_{at\ s'}$. Since $x \in R_i$, if x is enumerated into W_e it is put into Z, and hence into C. Moreover this is the only way items are enumerated into Z, so if x never enters W_e it never enters Z either.

□

There is an analogue of creativity for pairs of disjoint c.e. sets known as *effective inseparability*. It should be no surprise following Theorem 2.19 such analogous sets are also lattice-theoretic. This suggests that sets of this nature have an inherently low descriptive complexity.

Definition 2.20. *Disjoint c.e. sets A_0 and A_1 are* effectively inseparable *if there is a computable function p such that for every x, y we have*

$$[A_0 \subseteq W_x \ \wedge \ A_1 \subseteq W_y \ \wedge \ W_x \cap W_y = \emptyset] \ \Rightarrow \ p(x,y) \notin W_x \cup W_y. \qquad (2)$$

We call p the productive function *for (A_0, A_1).*

Effectively inseparable sets (A_0, A_1) are incapable of having their disjointness witnessed by computable sets. Similar to our definition of creativity (Definition 2.3) the productive function for effectively inseparable sets automatically gives a counterexample to any possible computable B such that $A_0 \subseteq B$ and $A_1 \cap B = \emptyset$.

Example 2.21. *Let $A_0 = \{e : \Phi_e(e) = 0\}$ and $A_1 = \{e : \Phi_e(e) = 1\}$. Then A_0 and A_1 are effectively inseparable.*

Proof. For c.e. sets W_x, W_y write $W_x \setminus W_y$ for the elements which are enumerated into W_x before W_y (if ever). Define a partial computable function p by

$$\Phi_{p(x,y)}(z) = \begin{cases} 1 \text{ if } z \in W_x \setminus W_y \\ 0 \text{ if } z \in W_y \setminus W_x \\ \uparrow \text{ otherwise.} \end{cases}$$

Suppose that $A_0 \subseteq W_x, A_1 \subseteq W_y$, and $W_x \cap W_y = \emptyset$. For contradiction, assume that $p(x,y) \in W_x \cup W_y$. If $p(x,y) \in W_x$ then $\Phi_{p(x,y)}(p(x,y)) = 1$, but also $p(x,y) \notin W_y$ and hence $p(x,y) \notin A_1$ so that $\Phi_{p(x,y)}(p(x,y)) \neq 1$. In case $p(x,y) \in W_y$ we obtain a similar contradiction. Hence $p(x,y) \notin W_x \cup W_y$, and p is productive for (A_0, A_1). □

To obtain a result similar to Theorem 2.19 for effectively inseparable sets we must build a repertoire of similar tools. These include corresponding notions of reducibility and completeness.

Definition 2.22. *Let (A_0, A_1) and (B_0, B_1) be two pairs of disjoint c.e. sets. We write $(A_0, A_1) \leq_m (B_0, B_1)$ if there is a computable function f such that $f(B_0) \subseteq A_0$, $f(B_1) \subseteq A_1$, and $f(\overline{B_0 \cup B_1}) \subseteq \overline{A_0 \cup A_1}$. If we can choose f to be one-one we write $(A_0, A_1) \leq_1 (B_0, B_1)$.*

Lemma 2.23 (Smullyan)**.** *Assume that (A_0, A_1) is a pair of effectively inseparable c.e. sets with productive function p. If (B_0, B_1) is a pair of disjoint c.e. sets then $(B_0, B_1) \leq_1 (A_0, A_1)$.*

Proof. By the double recursion theorem 1.3 there are one-one computable functions g and h with

$$W_{g(z)} = \begin{cases} A_0 \cup \{p(\langle g(z), h(z)\rangle)\} \text{ if } z \in B_1 \\ A_0 \text{ otherwise,} \end{cases}$$

and

$$W_{h(z)} = \begin{cases} A_1 \cup \{p(\langle g(z), h(z)\rangle)\} \text{ if } z \in B_0 \\ A_1 \text{ otherwise.} \end{cases}$$

Let $f(z) = p(\langle g(z), h(z)\rangle)$. We show that f gives the required reduction. Let $b \in B_0$. Then

$$\begin{aligned} &f(b) = p(\langle g(b), h(b)\rangle), \\ &W_{g(b)} = A_0, \text{ and} \\ &W_{h(b)} = A_1 \cup \{p(\langle g(b), h(b)\rangle)\}. \end{aligned}$$

We have $f(b) \in W_{g(b)} \cup W_{h(b)}$. Then $W_{g(b)} \cap W_{h(b)} \neq \emptyset$ since A_0 and A_1 are effectively inseparable. Thus $f(b) \in A_0$, for otherwise A_0 and A_1 would not be disjoint. By a symmetric argument, if $b \in B_1$ then $f(b) \in A_1$.

For the remaining details it suffices to check that if $b \notin B_0 \cup B_1$ then $f(b) \notin A_0 \cup A_1$. To this end suppose $b \notin B_0 \cup B_1$. Then

$$\begin{aligned} &f(b) = p(\langle g(b), h(b)\rangle), \\ &W_{g(b)} = A_0, \text{ and} \\ &W_{h(b)} = A_1. \end{aligned}$$

We know by assumption that $A_0 \cap A_1 = \emptyset$. Hence by the effective inseparability of A_0 and A_1 we have that $p(\langle g(b), h(b)\rangle) \notin W_{g(b)} \cup W_{h(b)}$; i.e., $f(b) \notin A_0 \cup A_1$, as required. □

Lemma 2.24. *Let $\{(\widehat{W}_{e_0}, \widehat{W}_{e_1})\}_{e \in \mathbb{N}}$ be an effective enumeration of the disjoint pairs of c.e. sets.*

$$\begin{aligned} A_0 &= \{\langle x, \langle e_0, e_1\rangle\rangle : \langle x, \langle e_0, e_1\rangle\rangle \in \widehat{W}_{e_0}\}, \\ A_1 &= \{\langle x, \langle e_0, e_1\rangle\rangle : \langle x, \langle e_0, e_1\rangle\rangle \in \widehat{W}_{e_1}\}. \end{aligned}$$

Then (A_0, A_1) is effectively inseparable.

Proof. We m-reduce the effectively inseparable pair from Example 2.21 to (A_0, A_1). Let $B_0 = \{e : \Phi_e(e) = 0\}, B_1 = \{e : \Phi_e(e) = 1\}$. Let (W_{k_0}, W_{k_1}) be the disjoint pair of c.e. sets given by

$$W_{k_0} = \{\langle x, n\rangle : x \in B_0 \ \wedge \ n \in \mathbb{N}\}$$
$$W_{k_1} = \{\langle x, n\rangle : x \in B_1 \ \wedge \ n \in \mathbb{N}\}.$$

Then $x \in B_i \leftrightarrow \langle x, \langle k_0, k_1\rangle\rangle \in A_i$, for $i = \{0, 1\}$. Let f be the computable function such that $f(x) = \langle x, \langle k_0, k_1\rangle\rangle$. Then $f(B_0) \subseteq A_0$, $f(B_1) \subseteq A_1$, and if $x \notin B_0 \cup B_1$ then we immediately have $\langle x, \langle k_0, k_1\rangle\rangle \notin A_0 \cup A_1$. □

Theorem 2.25 (Lattice-theoretic characterisation of effective inseparability, Nies)**.** *The following are equivalent for any pair of disjoint c.e. sets A_0, A_1:*

1. *A_0 and A_1 are effectively inseparable*
2. *$\exists F \, \forall Z_0, Z_1$ disjoint $\exists R \, [R \cap F \not\subseteq R \wedge R \cap A_0 = R \cap Z_0 \wedge R \cap A_1 = R \cap Z_1]$.*

Proof. "1 $\Rightarrow$ 2" Let (A_0, A_1) be as in Lemma 2.24. Let $F = \emptyset' \times \mathbb{N}$. Write $Z_0 = \widehat{W}_{e_0}$, $Z_1 = \widehat{W}_{e_1}$, and take $R = \mathbb{N} \times \{\langle e_0, e_1\rangle\}$. Then $R - F = \overline{\emptyset'} \times \{\langle e_0, e_1\rangle\}$, which is not c.e. So $R \cap F \not\subseteq R$. As in Theorem 2.19 $R \cap A_i = R \cap W_{e_i}$ for $i = 0, 1$. By Lemma 2.23 any other pair of disjoint c.e. sets is 1-equivalent to (A_0, A_1), and so satisfy 2 by Myhill's Isomorphism Theorem 2.14.

"2 $\Rightarrow$ 1" Suppose A_0, A_1 satisfy 2 via F. We proceed in a similar fashion to the proof of Theorem 2.19. We construct disjoint c.e. sets Z_0, Z_1 which generate a number of candidate witnesses R_i to the second criterion in the original statement, each of which allows us to define a potential function p_i to satisfy the effective inseparability of A_0 and A_1.

Construction. Stage $s = 0$. Fix a one-one enumeration of F, and an enumeration of the pairs of disjoint c.e. sets (W_x, W_y). Let $Z_{0,0}, Z_{1,0} = \emptyset$, and declare $p_{i,0}(e)$ undefined for each i, e.

Stage $s > 0$. Let i be least such that

i. If $t < s$ is the largest stage such that $t = 0$ or p_i had a new value defined at stage t then

$$(R_i \cap A_0) \upharpoonright_t [s] = (R_i \cap Z_0) \upharpoonright_t [s], \text{ and}$$
$$(R_i \cap A_1) \upharpoonright_t [s] = (R_i \cap Z_1) \upharpoonright_t [s].$$

ii. There is $z \in R_{i,s-1} \cap F_{at\ s}$

If i exists define $p_{i,s}(x, y) = z$, where $\langle x, y\rangle$ is least such that $p_{i,s-1}(x, y) \uparrow$. If z is enumerated into either $W_{x,u}$ or $W_{y,u}$ at some stage $u \geq s$, put also z into both $Z_{0,u}$ and $Z_{1,u}$. This completes the construction.

Verification.

- There is some total p_i

  As in Theorem 2.19.

- A_0 and A_1 are effectively inseparable via p_i

  Suppose that $p_i(x,y) \in W_x \cup W_y \ \wedge \ A_0 \subseteq W_x \ \wedge \ A_1 \subseteq W_y$. We want to ensure $W_x \cap W_y \neq \emptyset$. Given x, y, we define $p_i(x,y) = z$ at some stage s. For the same reasons as in Theorem 2.19 z is not taken as a value for any other p_k at any other stage. Now since $z \in R_i$, if z is enumerated into $W_x \cup W_y$ then z is put into $A_0 \cap A_1$, and hence $W_x \cap W_y \neq \emptyset$.

□

3 Structure of the Turing degrees

3.1 Turing reducibility

The reducibility we saw in Section 2 was a strong reducibility, and in fact too strong for some purposes. For instance, an incomputable set A does not in general m-reduce to its complement. However, *if we could compute* $\overline{A}$, then certainly we could compute A by reversing the answer we get. This motivates the following weaker reduction, introduced by Turing in [30].

Definition 3.1 (Oracle Machines)**.** *By an* oracle machine Φ_i^A *we mean a Turing machine* Φ_i *with access to an* oracle *for* A*. The oracle machine can query the oracle for* A *by posing a finite but unbounded number of questions of the form "*$x \in A$*?" to the oracle – to which the oracle replies truthfully. We define* $W_i^A = \{x : \Phi_i^A(x) \downarrow\}$ *the domain of the oracle machine* Φ_i^A*. We denote by* use $\Phi_i^A(x)$ *the* use *of the computation* $\Phi_i^A(x)$*; that is,* $1+$ *the largest oracle question made to* A*.*

Definition 3.2 (Turing Reduction)**.** *We say that a function* f *is* Turing reducible *to a set* B*, written* $f \leq_T B$ *if there is an oracle machine* Φ_i^B *with an oracle for* B *such that* $f(x) = \Phi_i^B(x)$ *for all* x*. We say that a set* A *is Turing reducible to* B*,* $A \leq_T B$*, if the characteristic function for* A *is Turing reducible to* B*.*

Although this definition may seem ethereal, oracle machines are simply mathematical devices for gauging the information content of a set. Turing remarked:

> "We shall not go any further into the nature of this oracle apart from saying that it cannot be a machine." ([30, pp. 172-173])

Definition 3.3.

- *For sets* A *and* B*, if* $A \leq_T B$ *and* $B \leq_T A$ *we write* $A \equiv_T B$*. Note that* $\equiv_T$ *is an equivalence relation.*

- *We denote the equivalence class of a set A modulo $\equiv_T$ by $\mathsf{deg}(A)$, the* Turing degree *of A.*
- *If $A \leq_T B$ we write $\mathsf{deg}(A) \leq \mathsf{deg}(B)$.*
- *The collection of all Turing degrees is denoted by $\mathcal{D}$.*

The Turing degrees were the first example of a degree structure. Initially they were known as the *degrees of unsolvability*, introduced by Post in 1948 [21]. This was followed by significant work done by Kleene and Post [16], Sacks [23], and Shoenfield [24].

We will tend to use boldface lowercase letters $\mathbf{a}, \mathbf{b}, \ldots$ for Turing degrees. Thus $\mathsf{deg}(\emptyset)$ is just the class of computable sets – using an $\emptyset$-oracle does not provide us with any new information, for anything we could compute with the help of an $\emptyset$-oracle we could already do. To see that $\leq_T$ really is a weaker reducibility notion, consider the following proposition.

Proposition 3.4. *Let A and B be sets. If $A \leq_m B$ then $A \leq_T B$.*

Proof. Let $A \leq_m B$ via f. To determine whether $x \in A$ simply ask a B-oracle whether $f(x) \in B$. □

Of course the converse does not hold. If $\overline{\emptyset'} \leq_m \emptyset'$ then by Proposition 2.6 $\overline{\emptyset'}$ would be c.e., and hence $\emptyset'$ would be computable.

Before the preceding proposition it was stated that using $\emptyset$ as an oracle does not provide any extra information. Indeed, $\mathsf{deg}(\emptyset)$ is minimal in $\mathcal{D}$. We denote $\mathsf{deg}(\emptyset)$ by $\mathbf{0}$.

Proposition 3.5. *Let $\boldsymbol{a}$ be a Turing degree. Then $\boldsymbol{0} \leq_T \boldsymbol{a}$.*

Proof. Write $\mathbf{a} = \mathsf{deg}(A)$, and $\mathbf{0} = \mathsf{deg}(B)$. Since B is computable its characteristic function is given by some Φ_i. Hence there is an oracle machine Φ_j^A which computes the characteristic function for B without querying its oracle. □

Definition 3.6. *We say that a Turing degree is computably enumerable (c.e.) if it contains a c.e. set.*

Kleene and Post in [16] introduced the jump operator on sets of numbers, relativising the halting problem to oracle machines. By the jump theorem below, the jump operator is invariant under Turing equivalence. This ensures we obtain a well-defined definition for the jump of a Turing degree.

Definition 3.7. *Let A be a set. We define the* jump *of A, written A', by $A' = \{x : \Phi_x^A(x)\downarrow\}$. The $(n+1)$-th jump of A, $A^{(n+1)}$, is defined to be $(A^n)'$. The n-th jump of a Turing degree $\boldsymbol{a} = \mathsf{deg}(A)$ is given by $\boldsymbol{a}^n = \mathsf{deg}(A^n)$.*

If a set A is the domain of an oracle machine Φ_i^B we say that A is *c.e. in B*. Intuitively, with a B-oracle we can enumerate A which was perhaps not computably enumerable. If the characteristic function of A is given by an oracle machine Φ_i^B, we say that A is *computable in B*.

Proposition 3.8. *Define* $K_0^A = \{(x, y) : \Phi_y^A(x)\downarrow\}$. *Then* $A' \equiv_m K_0^A$

Proof. Relativise the notions in Section 2.1 □

Theorem 3.9 (The Jump Theorem)**.** *Let* $A, B \subseteq \mathbb{N}$.

1. A' *is c.e. in* A
2. B *is c.e. in* A *iff* $B \leq_m A'$
3. $A' \not\leq_T A$
4. *If* $A \equiv_T B$ *then* $A' \equiv_T B'$

Proof. 1. Consider the partial A-computable function ψ which, on input x simulates $\Phi_x^A(x)$ and returns 1 if $\Phi_x^A(x)\downarrow$. Then $A' = \mathsf{dom}(\psi)$.

2. If B is c.e. in A then $B = \mathsf{dom}(\Phi_i^A)$ for some index i. Thus

$$x \in B \Leftrightarrow \Phi_i^A(x)\downarrow \Leftrightarrow (i, x) \in K_0^A \tag{3}$$

So $B \leq_m K_0^A \leq_m A'$, the second inequality given by Proposition 3.8. Conversely if $B \leq_m A'$ via f we set ψ to be the oracle machine which takes as input x and simulates $\Phi_{f(x)}^A(f(x))$. Then $\mathsf{dom}(\psi) = B$, whence B is c.e. in A.

3. Assume for contradiction that $A' \leq_T A$. Then $\overline{A'}$ is c.e. in A, and hence $\overline{A} = W_k^A$ for some index k. Now if $k \in W_k^A$ then $k \in A'$ and $k \in \overline{A'}$. Similarly if $k \notin W_k^A$ then $k \notin \overline{A'}$ and $k \notin A'$.

4. If $A \equiv_T B$ then A is c.e. in B. Now A' is c.e. in A, so A' is c.e. in B. By item 2 $A' \leq_m B'$, and so by Proposition 3.4 $A' \leq_T B'$. Symmetrically one has $B' \leq_T A'$.

□

Definition 3.10. *Let* $A, B \subseteq \mathbb{N}$. *Define the* computable join *of* A *and* B *by*

$$A \oplus B = \{2x : x \in A\} \cup \{2x + 1 : x \in B\}.$$

It is not difficult to see that $(\mathcal{D}, \leq)$ is a partial order. The following fact shows that $\mathcal{D}$ is in fact an upper semi-lattice.

Proposition 3.11. *Let* $\boldsymbol{a} = \mathsf{deg}(A)$ *and* $\boldsymbol{b} = \mathsf{deg}(B)$ *be Turing degrees. Then the least upper bound of* $\boldsymbol{a}$ *and* $\boldsymbol{b}$ *exists and is given by* $\boldsymbol{a} \vee \boldsymbol{b} = \mathsf{deg}(A \oplus B)$.

Proof. The maps $x \mapsto 2x$ and $x \mapsto 2x + 1$ are m-reductions from A and B to $A \oplus B$, so $A, B \leq_T A \oplus B$. Now suppose that $A, B \leq_T C$. Then the characteristic functions χ_A and χ_B are given by some oracle machines Φ_m^C and Φ_k^C respectively. The characteristic function $\chi_{A\oplus B}$ may then be given by Φ_p^C, where $\Phi_p^C(2x) = \Phi_m^C(x)$ and $\Phi_p^C(2x + 1) = \Phi_k^C(x)$. Hence $A \oplus B \leq_T C$. □

3.2 Post's problem

In 1944 [22] Post asked whether it was possible for a computably enumerable set to be both incomputable and incapable of solving $\emptyset'$, or if instead any incomputable c.e. set automatically solves every other c.e. set. Let us formally introduce this notion.

Definition 3.12. *Let $A \subseteq \mathbb{N}$. We say that A is* Turing complete *if A is c.e., and for every c.e. set B one has $B \leq_T A$.*

By Proposition 3.4 and Theorem 2.13 the halting problem $\emptyset'$ is Turing complete. Thus Post's problem asks whether there is a c.e. set which is both Turing incomplete and incomputable. We will often write suppress the Turing prefix and simply say incomplete.

We have already solved a parallel of Post's problem with respect to m-completeness: $\emptyset'$ is m-complete, and by Proposition 2.18 there exists a (c.e.) simple set A which, by Proposition 2.16, is neither m-complete nor computable.

In the next section 3.3 we will visit the attempt made by Kleene and Post to solve this problem. They got as far as constructing Turing incomparable sets below $\emptyset'$ which are not c.e. We show how to resolve Post's problem definitively in Section 3.4, and discuss other approaches to solving Post's problem using K-triviality in Section 5.

3.3 The finite extension method

The method of finite extensions was introduced in [16] by Kleene and Post resolving a weaker statement of Post's problem. They constructed Turing incomparable sets A_0 and A_1 below $\emptyset'$ in stages. The strategy is as follows. At each stage s one uses a $\emptyset'$-oracle to determine whether there is an extension τ to the set A_i for which $\Phi_n^\tau(x)\downarrow$ (note that at every stage A_i is finite), whence we extend A_j to ensure that the characteristic function for A_j on input x returns the opposite value. This is a *requirement* which must be met at all stages for the construction to be successful. In the limit one has constructed two infinite sets which, when used as oracles, cannot be used to compute each others' characteristic functions.

Theorem 3.13 (Kleene-Post [16])**.** *The Turing degrees are not linearly ordered. In particular, there are incomputable sets $A_0, A_1 \leq_T \emptyset'$ such that $A_0|_T A_1$.*

We present a proof of a stronger theorem using a modification of the method by Kleene and Post. It is indicative of rich structure in the sets below $\emptyset'$. First we need some notation for strings.

A *string* is a (possibly infinite) sequence of 0s and 1s. Given strings σ, τ, we write $\sigma \prec \tau$ if σ is a prefix of τ. We write $\sigma^\frown\tau$ for the concatenation of σ and τ. If σ is finite we write $|\sigma|$ for the length of σ. For $n \in \mathbb{N}$ we write $\sigma(n)$ for the nth digit of σ.

Theorem 3.14. *There exists an infinite sequence $\boldsymbol{a}_0, \boldsymbol{a}_1, \boldsymbol{a}_2, \ldots$ of degrees $\leq \boldsymbol{0}'$ such that $\boldsymbol{a}_i$ and $\boldsymbol{a}_j$ are incomparable whenever $i \neq j$.*

Proof. We construct sets $A_0, A_1, A_2, \ldots$ by defining sequences of strings $\tau_0^i \prec \tau_1^i \prec \ldots$ and setting $A_i = \bigcup_{j \geq 0} \tau_j^i$ for each i. We can think of the resulting infinite string as describing a set of natural numbers: a natural number i is in the set A if the ith digit of the string A is 1. Otherwise $i \notin A$. For each n, m, e we aim to meet the requirement

$$R_{n,m,e} : \exists k[A_n(k) \neq \Phi_e^{A_m}(k)].$$

Construction. Stage 0. Declare $\tau_0^k = \emptyset$ for each k.

Stage $i + 1 = \langle n, m, e \rangle$. Let $k = |\tau_i^m|$. Using a $\emptyset'$-oracle check whether there is $\tau \succ \tau_i^m$ such that $y = \Phi_e^\tau(k) \downarrow$. If such a τ exists let $\tau_{i+1}^m = \tau$ and $\tau_{i+1}^n = \tau_i^n \hat{} (1 - y)$. Otherwise let $\tau_{i+1}^n = \tau_i^n \hat{} 0$, $\tau_{i+1}^m = \tau_i^m \hat{} 0$.

Verification. Let $n, m \in \mathbb{N}$. Clearly $A_n, A_m \leq_T \emptyset'$ as our construction is computable relative to $\emptyset'$. We claim that $A_n \not\leq_T A_m$. Consider any $e \in \mathbb{N}$ - we show that $A_n \neq \Phi_e^{A_m}$. At stage $i + 1 = \langle n, m, e \rangle$ if we can find τ extending τ_i^m then

$$\Phi_e^{A_m}(|\tau_i^m|) \downarrow = \Phi_e^{\tau_{i+1}^m}(|\tau_i^m|) = y \neq 1 - y = \tau_{i+1}^n(|\tau_i^m|) = A_n(|\tau_i^m|). \quad (4)$$

Otherwise there is no τ extending τ_i^m such that $\Phi_e^\tau(|\tau_i^m|) \downarrow$. Hence we meet requirement R since A_m is an extension of τ_i^m:

$$\Phi_e^{A_m}(|\tau_i^m|) \uparrow \neq A_n(|\tau_i^m|); \quad (5)$$

proving our claim. Similarly we obtain $A_m \not\leq_T A_n$ at stage $\langle m, n, e \rangle$, so that $A_n |_T A_m$ indeed. □

We can think of a construction as an inductive definition. The base case initiates our various sets, and at each stage we contribute a finite amount of information to each of these, subject to any requirements we might enforce.

By the *cone* above a Turing degree $\mathbf{a}$ we mean the set $\{\mathbf{b} : \mathbf{a} \leq \mathbf{b}\}$. We can use the method of finite extensions to show that any degree in the cone above $\mathbf{0}'$ is the image of a degree under the jump operator. This is known as the Friedberg jump inversion theorem. Originally this result was known as the Friedberg completeness criterion, since it gives a criterion for a degree to above $\mathbf{0}'$ (and hence Turing complete); namely that it is the least upper bound of two degrees, one of them being $\mathbf{0}'$.

Theorem 3.15 (Friedberg Jump Inversion Theorem [6])**.** *For any $\boldsymbol{b} \in \mathcal{D}$, if $\boldsymbol{b} > \boldsymbol{0}'$ then there exists $\boldsymbol{a} \in \mathcal{D}$ such that $\boldsymbol{a}' = \boldsymbol{b}$.*

Proof. Write $\mathbf{b} = \deg(B)$. We construct $A = \bigcup_{i \in \mathbb{N}} \tau_i$ such that $A' \equiv_T B$ via finite extension.

Construction. Stage 0. Set $\tau_0 = \emptyset$.

Stage $s+1 = 2e+2$. We aim to meet the requirement

$$R_e \ : \exists \sigma \preceq A \ [\Phi_e^\sigma(e)\downarrow \ \vee \ (\forall \sigma' \succeq \sigma \ \Phi_e^{\sigma'}(e)\uparrow)].$$

To this end use a $\emptyset'$-oracle to determine whether

$$\exists \sigma \exists t \ [\tau_s \preceq \sigma \wedge \Phi_e^\sigma(e)[t]\downarrow]. \tag{6}$$

If such a σ and t exist, set $\tau_{s+1} = \sigma$. Otherwise set $\tau_{s+1} = \tau_s$.

Stage $s+1 = 2e+1$. We code $B(e)$ into A. With a B-oracle set $\tau_{s+1} = \tau_s{}^\frown B(e)$. This completes the construction.

Verification. We show that $A' \equiv_T A \oplus \emptyset' \equiv_T B$.

- $A \oplus \emptyset' \leq_T A'$

  This is obvious since $A, \emptyset' \leq_T A'$.

- $A' \leq_T B$

  We compute whether $e \in A'$ using a B-oracle to determine if (6) holds for e. If it does then $\Phi_e^A(e)\downarrow$ and so $e \in A'$. Otherwise $\Phi_e^\sigma(e)\uparrow$ for every $\sigma \succeq \tau_{2e}$, giving $\Phi_e^A(e)\uparrow$ so that $e \notin A'$.

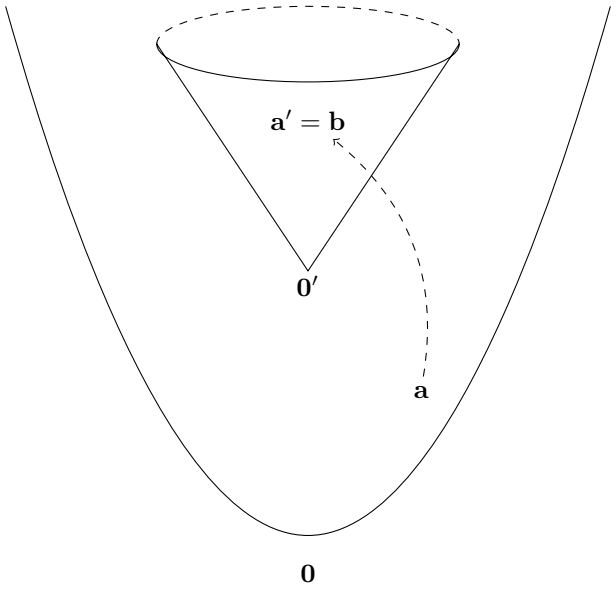
- $B \leq_T A \oplus \emptyset'$

  By induction we show that the sequence $\{\tau_i\}_{i\in\mathbb{N}}$ is computable in $A \oplus \emptyset'$. Given $\{\tau_0, \ldots, \tau_{2e}\}$ we compute τ_{2e+1} using $\emptyset'$ as in the construction, and then $\tau_{2e+2} = \tau_{2e+1}{}^\frown A(|\tau_{2e+1}|)$.

  Now to compute $B(e)$ we can use $A \oplus \emptyset'$ to determine τ_{2e+2}, whose last digit is precisely $B(e)$.

□

The set which we constructed actually belongs to a class of sets called the *generalised low* sets (see Definition 4.16 below). Theorem 3.15 gives us some additional perspective on the Turing degrees, namely that the jump operator is onto the cone above $\mathbf{0}'$:



On the other hand, the jump operator is not injective on the degrees. Should we choose $\mathbf{b} = \mathbf{0}''$ then by Theorem 3.15 there is a degree $\mathbf{a}$ such that $\mathbf{a}' = \mathbf{0}''$, but $\mathbf{a} \neq \mathbf{0}'$ by construction (otherwise $\emptyset'' \leq_T \emptyset'$, which is impossible by Theorem 3.9). Thus there are sets which are Turing incomparable, but determine the same halting sets for their respective oracle machines.

3.4 The finite injury priority method

It took 12 years following the publication of Post's problem in 1944 [22] for a solution to emerge. Friedberg and Muchnik independently resolved Post's problem positively using a novel construction called a *priority argument with finite injury*. In these arguments one constructs sets, say A and B, in stages to meet a number of requirements. These requirements are ordered according to *priority*. Meeting a requirement amounts to enumerating particular elements into A (or B). At the same time, what has so far been constructed of A is being used as an oracle in the construction of B. Hence enumerating an element into A changes A as an oracle, and could potentially *injure* one of B's requirements. All is not lost, so long as we return at a later stage and repair any possible injury. Priority is in place to ensure that one can satisfy a particular requirement R at a stage s only if doing so does not injure a requirement of greater priority. Ultimately one shows that each requirement is injured at most finitely many times, thereby ensuring that in the limit every requirement will be met.

Theorem 3.16 (Friedberg [7]-Muchnik [18])**.** *There exist c.e. Turing degrees* $\boldsymbol{a}, \boldsymbol{b} > \boldsymbol{0}$ *such that* $\boldsymbol{a} | \boldsymbol{b}$.

Proof. We build $A = \bigcup_{i \in \mathbb{N}} A_i$ and $B = \bigcup_{i \in \mathbb{N}} B_i$ by computable enumeration such

that the following requirements are met:

$$\begin{aligned} R_{2e} &: \exists n[A(n) \neq \Phi_e^B(n)] \\ R_{2e+1} &: \exists n[B(n) \neq \Phi_e^A(n)], \end{aligned}$$

subject to the priorities $R_0 > R_1 > R_2 > \ldots$ To meet R_{2e} and R_{2e+1} we require *witnesses* to the inequalities. We choose witnesses for R_{2e} and R_{2e+1} from $\mathbb{N} \times \mathbb{N} \times \{0\}$ and $\mathbb{N} \times \mathbb{N} \times \{1\}$ respectively. Intuitively, the even and odd requirements have their own planes of natural numbers at their disposal. This is to ensure that the witnesses don't conflict, and damage our construction.

Construction. Stage $s = 0$. Let $A_0 = B_0 = \emptyset$. Set initial witnesses to R_{2e} and R_{2e+1} by defining $a_e^0 = \langle e, 0, 0\rangle$, $b_e^0 = \langle e, 0, 1\rangle$.

Stage $s + 1 = 2e + 2$. We aim to satisfy R_{2e}. Given B_s and witness a_e^s

1. For each $e \leq s$, if $\Phi_e^{B_s}(a_e^s)[s] = 0$ and $a_e^s \notin A_s$, enumerate a_e^s into A.
2. Define the *restraint functions* $r^A(e, s)$ and $r^B(e, s)$ by

$$\begin{aligned} r^A(e, s) &= \mathsf{use}\ \Phi_e^{A_s}(b_e^s)[s] \\ r^B(e, s) &= \mathsf{use}\ \Phi_e^{B_s}(a_e^s)[s]. \end{aligned}$$

   We say that requirement R_{2e} (respectively R_{2e+1}) has been *injured* if there is some $x \in B_{s+1} - B_s$ ($x \in A_{s+1} - A_s$) such that $x \leq r^A(e, s)$ ($x \leq r^B(e, s)$).

   The restraint function establishes a wall for our witnesses. Once we have calculated how much a computation uses an oracle the wall is erected. We then only choose witnesses beyond the wall, so as to avoid infinitely recurring injury. Over time the wall is pushed further and further back so that each requirement may be injured at most finitely many times. We update the witness as follows

$$a_e^{s+1} = \begin{cases} \mu x > a_e^s\ [\forall i \leq e\ (x > r^B(i, s)) & \text{if } R_{2e+1} \text{ has been injured} \\ \qquad\qquad \wedge\ \exists z \leq x\ (\langle e, z, 0\rangle = x)] & \\ a_e^s & \text{otherwise.} \end{cases}$$

Stage $s + 1 = 2e + 1$. We aim to satisfy R_{2e+1}. Given A_s and witness b_e^s

1. For each $e \leq s$, if $\Phi_e^{A_s}(b_e^s)[s] = 0$ and $b_e^s \notin B_s$, enumerate b_e^s into B.
2. We update the witness for R_{2e+1}.

$$b_e^{s+1} = \begin{cases} \mu x > b_e^s\ [\forall i \leq e\ (x > r^A(i, s)) & \text{if } R_{2e} \text{ has been injured} \\ \qquad\qquad \wedge\ \exists z \leq x\ (\langle e, z, 1\rangle = x)] & \\ b_e^s & \text{otherwise.} \end{cases}$$

This completes the construction.

Verification. There exists s_0 such that at any stage $s > s_0$ the requirement R_{2e} is not injured. To see this observe that R_{2e} can only be injured if $\Phi_i(a_i^s)[s] = 0$ for some $i < e$, so that a_i^s is enumerated into A_{s+1}. Hence R_{2e} can be injured at most e times, and in particular only finitely many times. Since our witness a_e^s only changes in case R_{2e} is injured, $a_e := \lim_{s\to\infty} a_e^s$ exists. Now if $\Phi_e^B(a_e)\uparrow$ then R_{2e} is met. Otherwise $a_e \in A \Leftrightarrow \Phi_e^B(a_e) = 0$, so R_{2e} is again met. Similarly R_{2e+1} is met. □

We will use the finite injury priority method later in Section 4.2 to construct a *low* simple set. This provides another (arguably more perspicuous) solution to Post's problem which makes use of this kind of argument.

4 Absolute complexity

Recall from Section 1.1 that we understand an absolute complexity notion as one which gauges the complexity of a set by appealing to some fixed measure. In this section we will focus on three such notions: quantifier complexity, lowness, and hyperimmunity.

4.1 The arithmetical hierarchy

We have seen in Section 2 that some properties of computably enumerable sets may be definable in the first-order language of $\mathcal{E}$. We will now study definability in a different context, and investigate which relations are definable in $(\mathbb{N}, +, \cdot)$. It is known that the computable relations are definable in $(\mathbb{N}, +, \cdot)$, and hence any sentence obtained by stacking quantifiers in front of a computable relation is also definable in $(\mathbb{N}, +, \cdot)$. The number of (alternating) quantifiers in front of such a relation gives some notion of descriptive complexity, or of distance to computability. The relations definable in this way are called arithmetical, and exist in some level of the *arithmetical hierarchy*. Kleene introduced the hierarchy in [14] in order to give a classification of the predicates used in elementary number theory. Consider the following expressions, where R is a computable relation.

$\exists x\ [Rax]$	$\forall x\exists y\ [Raxy]$	$\exists x\forall y\exists z\ [Raxyz]$	$\ldots$
$\forall x\ [Rax]$	$\exists x\forall y\ [Raxy]$	$\forall x\exists y\forall z\ [Raxyz]$	$\ldots$

(7)

For each expression from the first row of (7) Kleene was able to show that there is a relation expressible in that form, but not in the corresponding form in the row below using the same number of quantifiers, or in any of the forms with fewer quantifiers. According to Kleene:

> "this result (obtained in 1940) marks the beginning of the use of applications and adaptions of recursive function theory to reveal

structure in parts of classical mathematics where effectiveness does not in general obtain." ([15, p. 63])

Definition 4.1 (The arithmetical hierarchy)**.** *Let $A \subseteq \mathbb{N}$.*

- $A \in \Sigma^0_0, \Pi^0_0, \Delta^0_0$ *iff A is computable.*
- $A \in \Sigma^0_{n+1}$ *iff there exists $B \in \Pi^0_n$ such that for all $x \in \mathbb{N}$,*

$$x \in A \leftrightarrow \exists y[(x, y) \in B].$$

- $A \in \Pi^0_{n+1}$ *iff there exists $B \in \Sigma^0_n$ such that for all $x \in \mathbb{N}$,*

$$x \in A \leftrightarrow \forall y[(x, y) \in B].$$

- $A \in \Delta^0_{n+1}$ *iff $A \in \Sigma^0_{n+1} \cap \Pi^0_{n+1}$.*

With the machinery of the arithmetical hierarchy we can calibrate the complexity of some incomputable sets. For instance, any c.e. set is Σ^0_1 and hence is low down on the hierarchy. Such sets are *almost* computable.

Proposition 4.2. *Let $A \subseteq \mathbb{N}$. Then A is c.e. iff $A \in \Sigma^0_1$.*

Proof. "$\Rightarrow$" Write $A = W_e$ for some index e. Then $x \in A \leftrightarrow \exists s[\Phi_e(x)[s] \downarrow]$, and hence $A \in \Sigma^0_1$.

"$\Leftarrow$" There is a computable set B such that $x \in A \leftrightarrow \exists y[(x, y) \in B]$. Define the partial computable function ψ by

$$\psi(x) = \begin{cases} 1 & \text{if } \exists y[(x, y) \in B] \\ \uparrow & \text{otherwise.} \end{cases}$$

Intuitively, ψ searches B (possibly forever) for a witness y to $x \in A$. Now since ψ is partial computable, there is a code i with $\psi = \Phi_i$. Then $A = \mathsf{dom}(\Phi_i) = W_i$. So A is c.e. □

Example 4.3. $\mathsf{Tot} \in \Pi^0_2$.

Proof. Observe that $i \in \mathsf{Tot} \leftrightarrow \forall x \Phi_i(x) \downarrow \leftrightarrow \forall x \exists y[\Phi_i(x)[y] \downarrow]$. □

Example 4.4. $\mathsf{Cof} := \{e : W_e \text{ is cofinite}\} \in \Sigma^0_3$.

Proof. $e \in \mathsf{Cof} \leftrightarrow \overline{W_e}$ is finite $\leftrightarrow \exists x \forall y \exists z[y \in W_{e,z} \vee y \leq x]$. □

Notice that if $A \in \Sigma^0_n$ for some n, then $\overline{A} \in \Pi^0_n$. For any first-order relation R, if R is computable, then so is $\neg R$. Thus if $x \in A \leftrightarrow \exists y_1 \forall y_2 \exists \ldots [(y_1, \ldots, y_n) \in B]$ we can push negations through the quantifiers, obtaining an equivalent sentence: $x \notin A \leftrightarrow \forall y_1 \exists y_2 \forall \ldots [(y_1, \ldots, y_n) \notin B]$. One also notices that if $A \in \Sigma^0_n$ then $A \in \Pi^0_{n+1}$, since we could add dummy quantifiers to the front of the first-order description of A. In fact we have the following inclusions:

$$\Sigma^0_0, \Pi^0_0 \subseteq \Delta^0_1 \subseteq \Sigma^0_1, \Pi^0_1 \subseteq \Delta^0_2 \subseteq \cdots \subseteq \Sigma^0_n, \Pi^0_n \subseteq \Delta^0_{n+1} \subseteq \ldots$$

The inclusions are strict for $n \geq 1$, so the hierarchy never reaches a plateau.

There is an identical structure obtainable by relativising the arithmetical hierarchy.

Definition 4.5 (Relativised arithmetical hierarchy)**.** *Let $A, B \subseteq \mathbb{N}$.*

- $A \in \Sigma_0^{0,B}, \Pi_0^{0,B}, \Delta_0^{0,B}$ *iff A is computable in B.*
- $A \in \Sigma_{n+1}^{0,B}$ *iff there exists $C \in \Pi_n^{0,B}$ such that for all $x \in \mathbb{N}$,*

$$x \in A \leftrightarrow \exists y[(x, y) \in C].$$

- $A \in \Pi_{n+1}^{0,B}$ *iff there exists $C \in \Sigma_n^{0,B}$ such that for all $x \in \mathbb{N}$,*

$$x \in A \leftrightarrow \forall y[(x, y) \in C].$$

- $A \in \Delta_{n+1}^{0,B}$ *iff $A \in \Sigma_{n+1}^{0,B} \cap \Pi_{n+1}^{0,B}$.*

Proposition 4.6. *Let $A, B \subseteq \mathbb{N}$. Then A is c.e. in B iff $A \in \Sigma_1^{0,B}$.*

Proof. Relativise Proposition 4.2. □

We have seen that $\mathsf{Tot} \in \Pi_2^0$, and so $\mathsf{Tot} \in \Pi_n^0$ for all $n > 2$. But how do we know that $\mathsf{Tot} \notin \Pi_1^0$? Writing a first-order description for a set A only gives an upper bound on its complexity relative to the arithmetical hierarchy. As we will see, in some cases we can do better.

Definition 4.7. *Say that a set A is Σ_n^0-complete if $A \in \Sigma_n^0$, and for every $B \in \Sigma_n^0$ we have $B \leq_m A$.*

Theorem 4.8. *$\emptyset'$ is Σ_1^0-complete.*

Proof. Let $A \in \Sigma_1^0$. Then A is c.e. by Proposition 4.2. Now $\emptyset'$ is m-complete by Theorem 2.13, so $A \leq_m \emptyset'$ as required. □

Post was able to show that the n-th jump of $\emptyset$ is Σ_n^0-complete, for any n.

Theorem 4.9 (Post's Theorem)**.** *Let $A \subseteq \mathbb{N}$, $n \in \mathbb{N}$. Then*

1. *$A \in \Sigma_{n+1}^0$ iff A is c.e. in $\emptyset^n$*
2. *$\emptyset^n$ is Σ_{n+1}^0-complete*
3. *$A \in \Delta_{n+1}^0$ iff $A \leq_T \emptyset^n$.*

Proof. 1. We induct on n. The base case is Proposition 4.2. Assume the claim holds for all $B \in \Sigma_n^0$. If $A \in \Sigma_{n+1}^0$ then there is $B \in \Pi_n^0$ such that $x \in A \leftrightarrow \exists y[(x, y) \in B]$. Hence $A \in \Sigma_1^{0,B}$. Now $\overline{B} \in \Sigma_n^0$, and so c.e. in $\emptyset^{n-1}$ by the inductive hypothesis. By Theorem 3.9 $\overline{B} \leq_m \emptyset^n$, so $A \in \Sigma_1^{0,\emptyset^n}$, whence A is c.e. in $\emptyset^n$ by Proposition 4.6. We obtain the converse with a reverse argument.

2. If $A \in \Sigma^0_{n+1}$ then A is c.e. in $\emptyset^n$ by item 1. Hence $A \leq_m \emptyset^{n+1}$ by Theorem 3.9.

3. The following statements are equivalent.

   (a) $A \in \Delta^0_{n+1}$

   (b) $A \in \Sigma^0_{n+1}$ and $\overline{A} \in \Sigma^0_{n+1}$

   (c) A and $\overline{A}$ are c.e. in $\emptyset^n$

   (d) $A \leq_T \emptyset^n$.

□

As a corollary of Post's Theorem we obtain the following characterisation of the Δ^0_2 sets.

Theorem 4.10 (Shoenfield Limit Lemma)**.** *Let $A \subseteq \mathbb{N}$. Then*

$$A \in \Delta^0_2 \text{ iff } A \leq_T \emptyset'.$$

Shoenfield's original approach to Theorem 4.10 involved *computable approximations.*

Definition 4.11. *Let $A \subseteq \mathbb{N}$. If $(A_s)_{s\in\mathbb{N}}$ is a sequence of finite sets such that $A(x) = \lim_{s\to\infty} A_s(x)$ for all $x \in \mathbb{N}$, we say that $(A_s)_{s\in\mathbb{N}}$ is a* computable approximation *for A.*

The simplest examples of computable approximations are *computable enumerations* $(A_s)_{s\in\mathbb{N}}$, where $A_s \subseteq A_{s+1}$. Any c.e. set A has a standard computable approximation, namely its computable enumeration. The Δ^0_2 sets turn out to be precisely those sets A which have computable approximations. We will hereafter use this fact in establishing further results. A full proof in this spirit may be found in [19] 1.4.2.

4.2 Lowness notions

There is an intermediate reducibility between Turing reducibility and many-one reducibility known as *truth-table* reducibility. If one can bound the size of the queries to an oracle in a Turing reduction one has a stronger reduction.

Definition 4.12 (Truth-table Reduction)**.** *Suppose $f \leq_T B$ via some oracle machine Φ^B.*

- *If there is a computable function r such that $\forall n$ use $\Phi^B(n) \leq r(n)$ say that f is* weak truth-table reducible *to B and write $f \leq_{wtt} B$.*
- *If f is the characteristic function for B we write $A \leq_{wtt} B$.*
- *If furthermore Φ^Z is total for every oracle Z call such a reduction a* truth-table reduction *and write $\leq_{tt}$.*

The machine Φ is called a (weak) truth-table reduction procedure.

These reducibilities induce *lowness notions* on the computational strength of oracles.

Definition 4.13. *Say that a set A is* low *if $A' \equiv_T \emptyset'$. Call A* superlow *if $A' \equiv_{tt} \emptyset'$.*

Intuitively, oracles which are *low* are computationally weak as an oracle, and thus cannot be Turing complete. Although they may be incomputable, they have minor information content in terms of decision problems. Indeed if A is low but Turing complete then $A \equiv_T \emptyset' \equiv_T A'$, which is impossible by Theorem 3.9. Thus we can give an alternative solution to Post's problem with the construction of a *low* incomputable c.e. set.

Theorem 4.14. *There is a low simple set.*

Proof. We build $A = \bigcup_{s \in \mathbb{N}} A_s$ by finite injury to meet the simplicity requirements S_e, and the lowness requirements L_e:

$$\begin{aligned} S_e &: |W_e| = \infty \Rightarrow W_e \cap A \neq \emptyset \\ L_e &: \exists^\infty s \ \Phi_e^{A_s}(e)[s]\downarrow \Rightarrow \ \Phi_e^A(e)\downarrow . \end{aligned}$$

We give priority $S_0 > L_0 > S_1 > L_1 > \ldots$ The lowness requirements ensure that $A' \equiv_T \emptyset'$ in the following way. Let g be defined by

$$g(e,s) = \begin{cases} 1 & \text{if } \Phi_e^{A_s}(e)[s]\downarrow \\ 0 & \text{otherwise.} \end{cases}$$

If L_e is met then $A'(e) = \lim_s g(e,s)$. Hence by the Limit Lemma ([25] III.3.3) $A' \leq_T \emptyset'$.

Construction. Stage 0. Let $A_0 = \emptyset$.

Stage $s+1$. Given A_s, choose the least $i \leq s$ such that

$$W_{i,s} \cap A = \emptyset \tag{8}$$

and

$$\exists x[x \in W_{i,s} \wedge x > 2i \wedge \forall e \leq i[x > r(e,s)]]. \tag{9}$$

If i exists, enumerate the least x satisfying (9) into A. Otherwise do nothing. This completes the construction.

Verification. Say that x *injures* L_e at stage $s+1$ if $x \in A_{s+1} - A_s$ and $x \leq r(e,s)$. Notice that S_i contributes at most one element x to A via (8). Hence L_e can only be injured by S_i if $i < e$. In particular, L_e is injured at most finitely many times.

- *L_e is met.*

  Choose s_0 such that L_e is not injured at any stage $s > s_0$. If $\Phi_e^{A_s}(e)\downarrow$ for some $s > s_0$ then $\Phi_e^{A_t}(e)\downarrow = \Phi_e^{A_s}(e)$ for all $t \geq s$, and hence $\Phi_e^{A}(e)\downarrow$.

- *S_e is met.*

  Let W_e be infinite. We stated above that for $t \geq s$, $\Phi_e^{A_t}(e) = \Phi_e^{A_s}(e)$. Since these machines make the same queries we get $r(e,s) = r(e,t)$, and hence $r(e) = \lim_s r(e,s)$ exists. Let $s' \geq s$ be any stage such that no simplicity requirement of higher priority receives attention after s', and let $t > s$ be any stage such that

  $$\exists x[x \in W_{e,t} \wedge x > 2e \wedge \forall i \leq e[x > r(i)]].$$

  Note that t must exist since we chose W_e to be infinite. Now either $W_{e,t} \cap A \neq \emptyset$ in which case S_e is met, or else $W_{e,t} \cap A = \emptyset$ in which case S_e is met at stage $t+1$.

This completes the proof. □

Corollary 4.15. *There is a superlow simple set A.*

The corollary requires a minor definition. Say that a function g is *ω-c.e.* if there is a sequence of computable functions $(g_s)_{s\in\mathbb{N}}$ such that $\forall x\ \lim_s g_s(x) = g(x)$, and there is a computable bound b with $b(x) \geq |\{s > x : g_{s-1}(x) \neq g_s(x)\}|$. We may set $g_s(x) = g(x,s)$.[1] If the characteristic function for a set is ω-c.e. we say that the set itself is ω-c.e.

Proof of Corollary 4.15. Let A and g be as above. Since each requirement L_e can only be injured by S_i if $i \leq e$, the total number of stages causing injury to L_e is at most $e+1$. At each stage the status of $g(e,s)$ may change at most twice. Hence $|\{s > e : g(e,s-1) \neq g(e,s)\}| \leq 2e+2$. By definition A' is ω-c.e., and hence by [19] 1.4.4 we have $A' \leq_{tt} \emptyset$. □

Definition 4.16. *Let $A \subseteq \mathbb{N}$. Say that A is* generalised low *if $A' \equiv_T A \oplus \emptyset'$.*

Generalised lowness is only slightly more refined than the preceding lowness notions, but still captures an interesting understanding of being weak as an oracle; namely that $\emptyset'$ is complete in the universe of A-machines. Recall Theorem 3.15. The set A was constructed to be generalised low. It follows that any set capable of solving the halting problem is the join of $\emptyset'$ with a generalised low set. Moreover, by Proposition 3.11 any degree $\mathbf{b} > \mathbf{0}$ is the least upper bound of $\mathbf{0}$ and $\mathsf{deg}(A)$, for some generalised low set A.

When proving that a particular set is generalised low we will tend to only show that $A' \leq_T A \oplus \emptyset'$. This is sufficient because the jump theorem 3.9 guarantees that $A, \emptyset' \leq_T A'$.

[1] This is actually due to the enumeration theorem, see [25] 3.4

4.3 Hyperimmune degrees

A function f *dominates* a (possibly partial) function g if

$$\forall^\infty x\ [g(x)\downarrow \to f(x) \geq g(x)].$$

We introduce another lowness notion known as *computable domination*. The set A can be seen as computationally weak if using A as an oracle does not enable one to compute functions which grow faster than one could compute before. That is, if using A as an oracle does not help one to dominate all computable functions.

Definition 4.17. *Let $A \subseteq \mathbb{N}$. Say that A is* computably dominated *if every function $g \leq_T A$ is dominated by some computable function.*

We can use domination to strengthen the definition of simplicity in the following way. For a set $A = \{a_0 < a_1 < a_2 < \dots\}$, denote by p_A the *principal function for A*, where $p_A(i) = a_i$.

Definition 4.18. *Let $A \subseteq \mathbb{N}$. Call A* hyperimmune *if A is infinite and p_A is not dominated by a computable function. If S is c.e. and $\overline{S}$ is hyperimmune we say that S is* hypersimple.

Theorem 4.19 (Dekker [3])**.** *Every nonzero c.e. Turing degree $\boldsymbol{a}$ contains a hypersimple set.*

Proof. Let $A \in \mathbf{a}$ be computably enumerable, and let f be a computable one-one enumeration of A. Define the *stages of deficiency* D of A by

$$D = \{x : \exists y > x\ [f(y) < f(x)]\} \tag{10}$$

We claim that i. D is hypersimple, and ii. $D \equiv_T A$. Assume to the contrary of i. that $p_{\overline{D}}$ is dominated by a computable function g. Then there is $x_0 \in \mathbb{N}$ such that for all $x > x_0$ we have $g(x) \geq p_{\overline{D}}(x)$. Now given x, let $z = \max\{x, x_0\}$. Then $g(z) \geq p_{\overline{D}}(z)$. Now for all $y > p_{\overline{D}}(z)$ we have $f(y) > f(p_{\overline{D}}(z))$. Hence $f(g(z)) > f(p_{\overline{D}}(z))$. Notice that no element of A less than x can be enumerated by f after $f(g(z))$, otherwise $p_{\overline{D}}(z) \in D$. Hence to determine if $z \in A$ we need only check to see if z is in the set $\{f(0), f(1), \dots, f(g(z))\}$. This contradicts A being incomputable. We complete the proof of claim i. by observing that D is Σ^0_1, and hence c.e.

Now $x \in A$ iff $x \in \{f(0), \dots, f(p_{\overline{D}}(x))\}$ since no element of A less than x is enumerated after $f(p_{\overline{D}}(x))$, and hence $A \leq_T D$.

Finally we show that $D \leq_T A$. Let $(A_s)_{s\in\mathbb{N}}$ be the computable enumeration of A given by $A_s = \{f(0), \dots, f(s)\}$. Given x, use an A-oracle to determine the least t such that $A_t \upharpoonright_{f(x)} = A \upharpoonright_{f(x)}$. If $t > x$ then $x \in D$ since $f(t) < f(x)$. If $t \leq x$ then $x \in \overline{D}$ since for every $y > x$ we have $f(y) \notin A \upharpoonright_{f(x)}$.

□

Theorem 4.20 (Kuznecov, Medvedev, Uspenskii)**.** *A is* not *computably dominated iff there is a hyperimmune set E such that $A \equiv_T E$.*

Proof. "⇐" Notice that $p_E \equiv_T E$. Hence A is not computably dominated since $p_E \leq_T A$ and p_E is not dominated by any computable function.

"⇒" There is a function $g \leq_T A$ which is not dominated by any computable function. We build a hyperimmune set $E \equiv_T A$ using g in the following way.

$$\begin{aligned} e_0 &= g(0), \\ e_{2n+1} &= e_{2n} + g(n) + 1, \\ e_{2n+2} &= e_{2n+1} + p_A(n) + 1. \end{aligned}$$

Set $E = \{e_i : i \in \mathbb{N}\}$. Clearly $E \equiv_T A$. Moreover E is hyperimmune since

$$e_{2n+2} > e_{2n+1} > g(n)$$

which by our hypothesis is not computably dominated. □

Definition 4.21. *Let $\boldsymbol{d} \in \mathcal{D}$. We say that $\boldsymbol{d}$ is* hyperimmune *if $\boldsymbol{d}$ contains a hyperimmune set. If $\boldsymbol{d}$ contains no hyperimmune set say that $\boldsymbol{d}$ is* computably dominated.

Theorem 4.19 tells us that every non-zero c.e. Turing degree is in fact hyperimmune. We extend this further, and show that the degree of any Δ^0_2 set is hyperimmune.

Definition 4.22. *Let $A \in \Delta^0_2$, and let $(A_s)_{s\in\mathbb{N}}$ be a computable approximation for A. Define the* computation function c_A *by*

$$c_A(x) = \mu s > x \ [A_s \upharpoonright_x = A \upharpoonright_x]. \tag{11}$$

Theorem 4.23. *The following hold for any set $A \subseteq \mathbb{N}$ and function g.*

1. *$c_A \equiv_T A$*
2. *If g dominates c_A then $A \leq_T g$.*

Proof. 1. $A \leq_T c_A$ since for any x we have that $x \in A$ iff $x \in A_{c_A(x)}$. Conversely, given x, enumerate $(A_s)_{s\in\mathbb{N}}$ until condition 4.22 holds. Hence $c_A \leq_T A$.

2. Without loss of generality suppose $g(x) > c_A(x)$ for all x. Define

$$h(x) = \mu z > x \ \forall t \ [z \leq t \leq g(z) \to A_t \upharpoonright_x = A_z \upharpoonright_x]. \tag{12}$$

Then the interval $[z, g(z)]$ must contain some stage t with $A_t \upharpoonright_z = A \upharpoonright_z$ since $c_A(z) < g(z)$. Since $t > x$ we also have $A_t \upharpoonright_x = A \upharpoonright_x$. Thus $A(x) = A_{h(x)}(x)$ by the consequent of (12). Since h may be obtained effectively from g we have $A \leq_T g$.

□

Corollary 4.24. *If c_A is dominated by a computable function then A is computable.*

Theorem 4.25. *Suppose $\emptyset <_T A \leq_T \emptyset'$. Then A is of hyperimmune degree.*

Proof. By Corollary 4.24 c_A is not dominated by any computable function, and hence by Theorem 4.23 A is not computably dominated. Thus by Theorem 4.20 there is a hyperimmune set $E \equiv_T A$. □

Definition 4.16 can be extended in the following way. Call a set A *generalised low$_2$* if $A'' \equiv_T (A \oplus \emptyset')'$. Thus if A is generalised low then A is generalised low$_2$ since $A'' = (A')' \equiv_T (A \oplus \emptyset')'$. It happens that being generalised low$_2$ is a stronger lowness notion than being computably dominated.

Theorem 4.26. *Let $A \subseteq \mathbb{N}$ be computably dominated. Then $A'' \leq_T A' \oplus \emptyset''$. In particular, A is generalised low$_2$.*

Proof. Note that $\overline{A''} \equiv_m \mathsf{Tot}^A := \{e : \Phi_e^A \text{ is total}\}$, and so $A'' \equiv_T \mathsf{Tot}^A$. Given e, let g be the function given by

$$g(x) = \mu t.\Phi_e^A(x)[t]. \tag{13}$$

Then g is partial computable in A. We may effectively find an index e_0 such that $g = \Phi_{e_0}^A$, and hence Φ_e^A is total iff $\Phi_{e_0}^A$ is. Using an $A' \oplus \emptyset''$-oracle find the least $\langle i, n\rangle$ such that $i \in \mathsf{Tot}$ and either

$$\exists x < n \ \forall s[\Phi_{e_0}^A(x)[s] \uparrow] \tag{14}$$

or

$$\forall x \geq n \ \forall s[\Phi_{e_0}^A(x)[s] \downarrow \Rightarrow \ \Phi_{e_0}^A(x)[s] \leq \Phi_i(x)]. \tag{15}$$

Such a pair $\langle i, n\rangle$ must exist, for if $\Phi_{e_0}^A$ is total then $\Phi_{e_0}^A$ is computably dominated by the hypothesis; giving (15). If $\Phi_{e_0}^A$ is not total then $\Phi_{e_0}^A$ is undefined on some input x irrespective of stage; giving (14). Finally, $e \in \mathsf{Tot}^A$ iff for all $x < n$ one has $\Phi_e^A(x) \downarrow$, and for all $x \geq n$ there exists $s \leq \Phi_i(x)$ such that $\Phi_e^A(x)[s] \downarrow$. The truth of conditions (14) and (15) can be computed relative to A', so the theorem follows. □

5 K-triviality

The *randomness* of a string may be measured by the presence or absence of structure. Intuitively, a string is random if it is not structured. Well structured strings are highly compressible since their patterns can be exploited for short descriptions, and thus are not random. If the structure is not algorithmic, however, the string 'looks' random to any machine. This is a difference between genuine randomness (should it exist) and effective randomness. On the other hand, a string which lacks any definite structure appears very complicated and will likely have no short description. Kolmogorov complexity allows us to be formal about these notions.

Definition 5.1. *Let K denote the prefix-free Kolmogorov complexity, i.e. for a fixed optimal universal prefix-free machine $\mathbb{U}$ operating over the language $\{0,1\}^*$, we define $K(x) = \min\{|\sigma| : \mathbb{U}(\sigma) = x\}$. The K operator can be relativized for an oracle A by setting $K^A(x) = \min\{|\sigma| : \mathbb{U}^A(\sigma) = x\}$.*

If $K(x) \leq |x| - b$, we say that x is *b-compressible*. If we can choose $b = 1$ then x is *compressible*, and if there is no such b then x is *incompressible*. It is not difficult to see that there is at least one incompressible string of every length. Indeed, for $n > 1$ there are $2^0 + 2^1 + \cdots + 2^{n-1} = 2^n - 1$ binary strings of length less than n, while there are 2^n strings of length n.

Theorem 5.2 (Kolmogorov)**.** *The set $A = \{x : K(x) < |x|\}$ of compressible strings is simple.*

Proof. By the preceding remark $\overline{A}$ is infinite. Moreover A is Σ^0_1 (and hence c.e.) since $x \in A \leftrightarrow \exists\sigma\exists s[|\sigma| < |x| \wedge \mathbb{U}_s(\sigma) = x]$. Finally, assume for contradiction that $\overline{A}$ contains some infinite c.e. subset B. Let M be the prefix-free machine which, on input n waits until an element y_n is enumerated into B such that $|y_n| > n$, and halts outputting y_n. Let c be the length of the encoding of M with doubled bits. Then $K(y_n) \geq |y_n| \geq n$ since $y_n \in \overline{A}$. But M together with n can describe y_n, hence $c + \log_2(n) \geq K(y_n) \geq n$ which is false for sufficiently large n. □

5.1 Lowness and triviality

We introduce another lowness notion in terms of Kolmogorov complexity.

Definition 5.3. *Say that a set A is* low for K *if there is a constant b such that*

$$\forall x[K^A(x) \geq K(x) - b].$$

The intuition for sets A which are low for K is that using A as an oracle doesn't help to detect new regularities in strings which can then be exploited for compression. Clearly computable sets are low for K since if A is computable we can replace our queries to the oracle with a computation of χ_A, thereby increasing the length of a description only by a fixed constant. This lowness notion implies that A is computationally weak in the sense of Definition 4.13 and Definition 4.16.

Theorem 5.4. *If A is low for K then A is generalised low.*

Proof. Let A be low for K via b. We show how to compute A' in $A \oplus \emptyset'$. Let M_c be the prefix-free machine such that $M_c^X(0^e1) = \mu s.\Phi_e^X(e)[s]\downarrow$. Then $M_c^X(0^e1) = \mathbb{U}^X(0^{c-1}10^e1)$. Hence

$$K^A(M_c^A(0^e1)) \leq K^A(0^{c-1}10^e1) \leq c + e + 1. \tag{16}$$

Since A is low for K via b we have

$$K(M_c^A(0^e1)) \leq b + c + e + 1. \tag{17}$$

We use $\emptyset'$ to determine t such that $t = \max\{\mathbb{U}(\sigma) : |\sigma| < b + c + e + 1\}$. Then $\Phi_e^A(e) \downarrow$ iff $\Phi_e^A(e)[t] \downarrow$. Hence $A' \leq_T A \oplus \emptyset'$. □

We identify a set $A \subseteq \mathbb{N}$ with an infinite binary string $x = x_0x_1x_2\ldots$ by $x_i = 1$ if $i \in A$, and $x_i = 0$ otherwise. Thus results about the randomness of strings can carry over to sets. Of course, the work done so far has been using finite objects and so we must introduce new notions for infinite strings.

Definition 5.5. *Let Z be a set. We say that Z is* 1-random *if there is a constant $b \in \mathbb{N}$ such that*

$$\forall n \ [K(Z \upharpoonright_n) > n - b]$$

That is, the initial segments of 1-random sets are b-incompressible. On the opposite end of the randomness spectrum is the notion of K-triviality. *K-triviality* is in fact an *anti-randomness* notion. A set which is K-trivial is one whose initial segments are no more complicated than their length, in terms of algorithmic complexity. Intuitively K-trivials are sets whose initial segments are easy to describe.

Definition 5.6. *Say that A is K-trivial via $b \in \mathbb{N}$ if*

$$\forall n \in \mathbb{N} \ [K(A \upharpoonright_n) \leq K(n) + b].$$

It is a striking phenomenon that the inability of an oracle A to detect regularities in strings (i.e. being low for K) in fact coincides with K-triviality, or having slowly growing initial segment complexity. One side of this coincidence is not difficult to prove.

Proposition 5.7. *Every set A which is low for K is K-trivial.*

Proof. Let M be the prefix-free oracle machine which, on input n with oracle X, computes $X \upharpoonright_n$. Let c be the coding constant for M. Then $K^A(A \upharpoonright_n) \leq K(n) + c$. If A is low for K via b then for every n we have $K(A \upharpoonright_n) \leq K(n) + c + b$. □

Chaitin proved that there are only $O(2^b)$ sets which are K-trivial via b. As a consequence the K-trivials are Δ_2^0 sets,[2] and so by Theorem 4.10 have computable approximations. We will use this fact in the following section to prove the existence of incomputable K-trivial sets using cost functions.

5.2 The cost function method

The cost function method is an approach to building c.e. sets which satisfy particular lowness properties. Given a Δ_2^0 set, we assign a *cost* to the act of changing items in its computable approximation at a stage s. That is, putting x into (or taking x out of) A_s has a cost $\mathbf{c}(x, s)$. In this subsection we use the cost function method to build our first example of an incomputable K-trivial set.

[2] For each b there is a binary tree whose paths coincide with the sets which are K-trivial via b. Each of these paths is isolated, and so computable in $\emptyset'$. See [19] 5.2.4 for details.

Definition 5.8. *By a* cost function $\boldsymbol{c}$ *we mean a computable function*

$$\boldsymbol{c} : \mathbb{N} \times \mathbb{N} \to \{x \in \mathbb{Q} : x \geq 0\}.$$

Let $(A_s)_{s\in\mathbb{N}}$ be a computable approximation. If the sum

$$\mathbf{c}(A_s) = \sum_{x,s} \mathbf{c}(x,s)[\![x < s \wedge x \text{ is least s.t. } A_{s-1}(x) \neq A_s(x)]\!] \tag{18}$$

is finite, we say that $(A_s)_{s\in\mathbb{N}}$ *obeys* $\mathbf{c}$ and write $(A_s) \models \mathbf{c}$. If $A = \lim_{s\to\infty} A_s$ we write $A \models \mathbf{c}$ and say that A obeys $\mathbf{c}$. A cost function $\mathbf{c}$ is *monotonic* if $\mathbf{c}(x+1,s) \leq \mathbf{c}(x,s) \leq \mathbf{c}(x,s+1)$ for all $x < s$. We say that $\mathbf{c}$ satisfies the *limit condition* if

$$\forall e \forall^\infty x \forall s > x[\mathbf{c}(x,s) \leq 2^{-e}]. \tag{19}$$

The limit condition guarantees the existence of obedient sets.

Theorem 5.9. *For any cost function* $\boldsymbol{c}$ *satisfying the limit condition there is an obedient simple set* A*.*

Proof. We give a computable enumeration of $A = \bigcup_{s\geq 0} A_s$. We aim to meet the simplicity requirements

$$S_e \ : \ |W_e| = \infty \Rightarrow A \cap W_e \neq \emptyset.$$

Construction. At stage $s = 0$ declare $A_0 = \emptyset$. At stage $s > 0$, for each $e < s$, if S_e has not yet been met and there is $x \geq 2e$ such that $\mathbf{c}(x,s) \leq 2^{-e}$, enumerate x into A_s.

Clearly $A \models \mathbf{c}$ since $\mathbf{c}(A_s)$ is bounded above by $\sum_{e\in\mathbb{N}} 2^{-e} = 2$. Since $\mathbf{c}$ satisfies the limit condition, there is x_0 such that for all $x > x_0$ and $s > x$ one has $\mathbf{c}(x,s) \leq 2^{-e}$. Suppose W_e is infinite. Then it must contain such an x_0, and so we choose $x > x_0$ at some stage $s > x$ and meet S_e. □

Proposition 5.10. *If a computable approximation* $(A_s)_{s\in\mathbb{N}}$ *for an incomputable set* A *obeys a monotonic cost function* $\boldsymbol{c}$ *then* $\boldsymbol{c}$ *satisfies the limit condition.*

Proof. We argue by contrapositive. Assume there is $\varepsilon > 0$ such that

$$\exists^\infty x \exists s > x[\mathbf{c}(x,s) > 2^{-\varepsilon}]. \tag{20}$$

If $A \models \mathbf{c}$ then there is a stage s_0 such that

$$\sum_x \sum_{s>s_0} \mathbf{c}(x,s)[\![x < s \wedge \ x \text{ is least s.t. } A_{s-1}(x) \neq A_s(x)]\!] \leq 2^{-\varepsilon}. \tag{21}$$

We compute A as follows. On input w, search for a stage s such that $s > s_0$, $s > w$, and $\mathbf{c}(w,s) > 2^{-\varepsilon}$. Note that (20) guarantees such a stage: there is $v > w$ and $s > s_0$ such that $\mathbf{c}(v,s) > 2^{-\varepsilon}$, while monotonicity gives $\mathbf{c}(w,s) \geq \mathbf{c}(v,s)$. We claim that $A_s(w) = A(w)$. Otherwise there is $u \leq w$ with $A_{s-1}(u) \neq A_s(u)$ and $\mathbf{c}(u,s) \geq \mathbf{c}(w,s) > 2^{-\varepsilon}$, contradicting (21). □

A cost function can characterise a class of c.e. sets much in the same way a formula φ may characterise a class of models. For instance, Löb's modal formula $\Box(\Box p \to p) \to \Box p$ characterises the class of frames $\mathfrak{F} = (W, R)$ such that R is transitive and reverse well-founded. The following cost function characterises the K-trivials.

Definition 5.11. *The* standard cost function $\boldsymbol{c}_K$ *is defined by*

$$\boldsymbol{c}_K(x, s) = \sum_{w=x+1}^{s} 2^{-K_s(w)}$$

where $K_s(w) = \min\{|\sigma| : \mathbb{U}_s(\sigma) = w\}$. *By convention* $K_s(w) = \infty$ *if there is no such minimum.*

Note that $\mathbf{c}_K$ satisfies the limit condition. To see this, notice that $\sum_w 2^{-K(w)} \leq 1$, so for any e there is x_0 such that $\sum_{w \geq x_0} 2^{-K(w)} \leq 2^{-e}$. Thus for any $x > x_0$ and $s > x$ one has

$$\sum_{w \geq x_0}^{s} 2^{-K_s(w)} \leq \sum_{w \geq x_0} 2^{-K(w)} \leq 2^{-e}.$$

We note the following theorem, known as the Machine Existence Theorem or the Kraft-Chaitin Theorem, which we will use in the upcoming proof. By a *bounded request set* we mean a set of *requests* $W = \{\langle r_i, x_i\rangle : i < N\}$, where $N \in \mathbb{N} \cup \{\infty\}$, $r_i \in \mathbb{N}$, $x_i \in \{0,1\}^*$, such that

$$\sum_{\langle r_i, x_i\rangle \in W} 2^{-r_i} \leq 1.$$

Intuitively, a request $\langle r, x\rangle$ asks for a description of the string x of length r. The Machine Existence Theorem tells us that there is a machine which meets exactly those requests.

Theorem 5.12 (The Machine Existence Theorem)**.** *Let* W *be a c.e. bounded request set. Then one can effectively obtain a prefix-free machine* $M = M_d$, $d > 1$, *such that*

$$\forall r, x[\langle r, x\rangle \in W \leftrightarrow \exists w\ (|w| = r\ \wedge\ M(w) = x)].$$

A proof can be found in [19] 2.2.17.

Theorem 5.13 ([5])**.** *Let* $(A_s)_{s\in\mathbb{N}}$ *be a computable approximation for a set* A. *If* $A \models \boldsymbol{c}_K$ *then* A *is K-trivial.*

Proof. Let p be least such that $\mathbf{c}(A) \leq 2^p$. We enumerate a bounded request set W for A which ensures that A is K-trivial. At stage s enumerate the request $\langle K_s(w) + p + 1, A_s \upharpoonright_w\rangle$ into W whenever $w \leq s$ and

1. $K_s(w) < K_{s-1}(w)$, or
2. $K_s(w) < \infty \ \wedge \ A_{s-1} \upharpoonright_w \neq A_s \upharpoonright_w$.

Requests enumerated into W as a result of criterion 1 contribute no more than

$$\sum_{s,w} 2^{-(K_s(w)+p+1)} \leq \sum_{\sigma} 2^{-(|\sigma|+p+1)} \leq \frac{1}{2^{p+1}} \tag{22}$$

to the weight of W. Given that $(A_s)_{s\in\mathbb{N}}$ converges to A, for every w there are a finite number of stages s such that $A_{s-1} \upharpoonright_w \neq A_s \upharpoonright_w$. Hence there are a finite number of requests $\langle K_s(w)+p+1, A_s \upharpoonright_w\rangle$ enumerated into W, contributing at most $\frac{1}{2^{p+1}}$ as in (22). Thus W is a bounded request set.

Let M_d be the prefix-free machine for W obtained via Theorem 5.12. We show that $K(A \upharpoonright_w) \leq K(w) + d + p + 1$. Given w, let s be greatest such that $s = 0$ or $A_{s-1} \upharpoonright_w \neq A_s \upharpoonright_w$. If $s > 0$ then there is a request $\langle K_s(w)+p+1, A_s \upharpoonright_w\rangle$ enumerated into W. Hence there is $t > s$ such that $K_t(A \upharpoonright_w) \leq K_s(w) + d + p + 1$. If $K_s(w) = K(w)$ we meet the claim. Otherwise there is $j > s$ with $K_j(w) = K(w)$ such that $\langle K_j(w) + p + 1, A_j \upharpoonright_w\rangle$ is enumerated into W as a result of criterion 1. In this case $A_j \upharpoonright_w = A \upharpoonright_w$ since we chose s greatest. Now if $s = 0$ then $A_j \upharpoonright_w = A \upharpoonright_w$ for all j. There will be some least $t \geq w$ such that $K_t(w) = K(w)$. At stage t we put $\langle K(w) + p + 1, A \upharpoonright_w\rangle$ in to W, from which we have $K(A \upharpoonright_w) \leq K(w) + d + p + 1$. □

Combining Theorem 5.13 and Theorem 5.9 we have the following corollary.

Corollary 5.14. *There is a simple K-trivial set.*

5.3 Incompleteness

Let A be a simple K-trivial set. Then A is Δ^0_2 and hence below $\emptyset'$. Also A is incomputable and hence strictly above $\emptyset$. Thus, showing that K-trivial sets are not Turing complete would provide an injury-free solution to Post's problem. Here we show that every K-trivial set is weak truth-table incomplete. The proof that K-trivials are Turing incomplete requires a so-called decanter construction and extends the following proof.

Theorem 5.15 ([19])**.** *Every K-trivial set A is wtt-incomplete.*

Proof. Assume for contradiction that A is K-trivial via b and wtt-complete. We enumerate a c.e. set B. By the recursion theorem 1.2 we are given a c.e. index, and therefore a many-one reduction f from $B = W_e$ to $\emptyset'$. Since A is wtt-complete, there is a fixed wtt-reduction from $\emptyset'$ to A. Combining this reduction with f we have a Turing reduction Γ such that $B = \Gamma^A$ and computable use g of Γ. We begin by enumerating a single element $\langle 0, n\rangle$ into a set L. Then L will be a bounded request set, and hence by Theorem 5.12 there is a prefix-free machine M_d meeting precisely that request. By the recursion theorem we may assume we know the constant d in advance.

Construction. Let $c = 2^{b+d}$, and let $n = g(c)$. Fix a computable approximation $(A_s)_{s\in\mathbb{N}}$ of A.

Stage $t = 0$. Put the request $\langle 0, n\rangle$ into L.

Stage $t > 0$. If $B \upharpoonright_c [t] = \Gamma^A \upharpoonright_c [t]$ and $K_t(A_t \upharpoonright_n) \leq b + d$ then put x into B, where x is greatest such that $x < c$ and $x \notin B$. Otherwise do nothing.

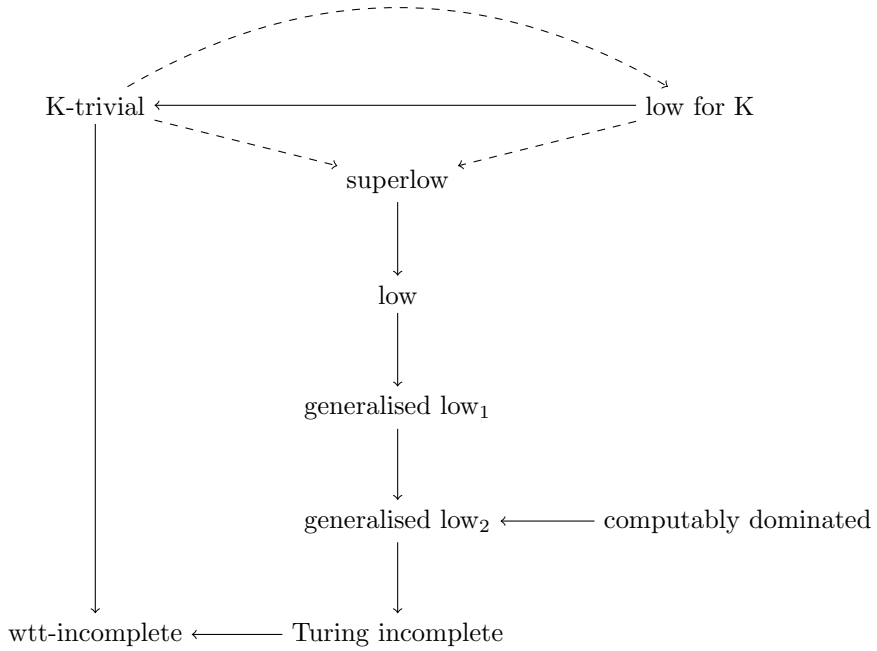
Verification. Enumerating a fresh $x < c$ into B at a stage t causes a change in B. This change requires a change in the approximation A_t to ensure that $B \upharpoonright_c = \Gamma^A \upharpoonright_c$ in the limit. Since the shortest description of $A_t \upharpoonright_n$ is at most $b+d$, this change contributes a weight of at least $2^{-(b+d)}$. As we build B, we put a total of c such elements x into B, causing the total measure of the different descriptions of $A \upharpoonright_n$ to be at least $(c+1)2^{-(b+d)} = 1 + 2^{-(b+d)} > 1$, which is impossible as no prefix-free machine can meet these requests.[3] □

The preceding construction tries to overload the amount of measure needed to describe A below some particular use n. If we don't have a computable bound g on the use of Γ, then after we force some A_t-change the use of Γ could exceed n. Future changes in the description of $A \upharpoonright_n$ can now contribute less than $2^{-(b+d)}$ to the total measure, and we do not attain the desired contradiction.

Thus this approach does not work in case we only know that Γ is a Turing reduction. The *decanter method* was created to tolerate such difficulties. Decanter constructions can be used to do what the argument in Theorem 5.15 tries to do for Turing reductions. Namely, they can be used to show that the K-trivials are incomplete, and furthermore that the K-trivials are low for K. See [4] Section 11.3 or [19] Section 5.4 for details.

We summarise some implications between lowness notions and completeness in the following diagram. The implications with dashed lines are true, but not proven in this report.

[3] Alternatively, the measure of the open set $[A \upharpoonright_n]^{\prec}$ cannot exceed 1.



References

[1] Gregory J. Chaitin. On the length of programs for computing finite binary sequences. *Journal of the ACM (JACM)*, 13(4):547–569, 1966.

[2] S. Barry Cooper. *Computability theory*. CRC Press, 2003.

[3] J.C.E. Dekker. A theorem on hypersimple sets. *Proceedings of the American Mathematical Society*, 5(5):791–796, 1954.

[4] Rodney G. Downey and Denis R. Hirschfeldt. *Algorithmic randomness and complexity*. Springer, 2010.

[5] Rodney G. Downey, Denis R. Hirschfeldt, André Nies, and Frank Stephan. Trivial reals. *Electronic Notes in Theoretical Computer Science*, 66(1):36–52, 2002.

[6] Richard M. Friedberg. A criterion for completeness of degrees of unsolvability. *The Journal of Symbolic Logic*, 22(02):159–160, 1957.

[7] Richard M. Friedberg. Two recursively enumerable sets of incomparable degrees of unsolvability (solution of post's problem, 1944). *Proceedings of the National Academy of Sciences of the United States of America*, 43(2):236, 1957.

[8] Kurt Gödel. Über formal unentscheidbare sätze der principia mathematica und verwandter systeme i. *Monatshefte für mathematik und physik*, 38(1):173–198, 1931.

[9] Leo Harrington and Robert I. Soare. Post's program and incomplete recursively enumerable sets. *Proceedings of the National Academy of Sciences*, 88(22):10242–10246, 1991.

[10] Leo Harrington and Robert I. Soare. Dynamic properties of computably enumerable sets. *LONDON MATHEMATICAL SOCIETY LECTURE NOTE SERIES*, pages 105–122, 1995.

[11] Leo Harrington and Robert I. Soare. Definability, automorphisms, and dynamic properties of computably enumerable sets. *Bulletin of Symbolic Logic*, 2(02):199–213, 1996.

[12] Leo Harrington and Robert I. Soare. Definable properties of the computably enumerable sets. *Annals of pure and applied logic*, 94(1):97–125, 1998.

[13] Stephen C. Kleene. On notation for ordinal numbers. *The Journal of Symbolic Logic*, 3(4):150–155, 1938.

[14] Stephen C. Kleene. Recursive predicates and quantifiers. *Transactions of the American Mathematical Society*, 53(1):41–73, 1943.

[15] Stephen C. Kleene. Origins of recursive function theory. In *Foundations of Computer Science, 1979., 20th Annual Symposium on*, pages 371–382. IEEE, 1979.

[16] Stephen C. Kleene and Emil L. Post. The upper semi-lattice of degrees of recursive unsolvability. *Annals of mathematics*, pages 379–407, 1954.

[17] Andrei N. Kolmogorov. Three approaches to the quantitative definition of information. *Problems of information transmission*, 1(1):1–7, 1965.

[18] Albert A. Muchnik. On the unsolvability of the problem of reducibility in the theory of algorithms. In *Dokl. Akad. Nauk SSSR*, volume 108, page 1, 1956.

[19] André Nies. *Computability and randomness*, volume 51. Oxford University Press, 2009.

[20] Piergiorgio Odifreddi. Classical recursion theory: The theory of functions and sets of natural numbers, vol. 1 (studies in logic and the foundations of mathematics, vol. 125), 1992.

[21] Emil L. Post. Degrees of recursive unsolvability-preliminary report. In *Bulletin of the American Mathematical Society*, volume 54, pages 641–642. AMER MATHEMATICAL SOC 201 CHARLES ST, PROVIDENCE, RI 02940-2213, 1948.

[22] Emil L. Post. Recursively enumerable sets of positive integers and their decision problems. *Mathematical Logic in the 20th Century*, page 352, 2003.

[23] Gerald E. Sacks. *Degrees of unsolvability*. Number 55. Princeton University Press, 1963.

[24] J. Shoenfield. *Degrees of unsolvability*, volume 2. Mathematical Studies 2, North-Holland, Amsterdam, 1971.

[25] Robert I. Soare. *Recursively enumerable sets and degrees: A study of computable functions and computably generated sets*. Springer, 1987.

[26] Robert I. Soare. The history and concept of computability. *1999 Handbook of Computability Theory, Elsevier*, 1999.

[27] Ray J. Solomonoff. A formal theory of inductive inference. part i. *Information and control*, 7(1):1–22, 1964.

[28] John Stillwell. Emil Post and his anticipation of Gödel and Turing. *Mathematics Magazine*, pages 3–14, 2004.

[29] Alan M. Turing. On computable numbers, with an application to the Entscheidungsproblem. *J. of Math*, 58:345–363, 1936.

[30] Alan M. Turing. Systems of logic based on ordinals. *Proceedings of the London Mathematical Society*, 2(1):161–228, 1939.