Exercises from *Computability*

John Peloquin

Chapter 1

Exercises 3.3

Exercise 1. We show that the functions below are computable by exhibiting procedures which compute them.

(a) $f(x) = \begin{cases} 0 & \text{if } x = 0 \\ 1 & \text{if } x \neq 0 \end{cases}$

1. J(1,2,4)
2. Z(1)
3. S(1)

(b) $f(x) = 5$

1. Z(1)
2. S(1)
⋮ ⋮
6. S(1)

(c) $f(x,y) = \begin{cases} 0 & \text{if } x = y \\ 1 & \text{if } x \neq y \end{cases}$

Same procedure as in (a).

(d) $f(x,y) = \begin{cases} 0 & \text{if } x \leq y \\ 1 & \text{if } x > y \end{cases}$

1. J(1,3,5)
2. J(2,3,7)
3. S(3)
4. J(1,1,1)
5. Z(1)
6. J(1,1,9)
7. Z(1)
8. S(1)

The idea in this procedure is to run a counter (in R_3) which is repeatedly checked against first x and then y. If x is hit first, we know $x \leq y$. If y is hit first, we know $y < x$.

(e) $f(x) = \begin{cases} \frac{1}{3}x & \text{if } x \text{ is a multiple of 3} \\ \text{undefined} & \text{otherwise} \end{cases}$

See Example 3.3(c).

(f) $f(x) = \lfloor 2x/3 \rfloor$

We merely sketch a procedure: first load $2x$ into a register. Run a counter k in one register and $3(k+1)$ in another register, stopping at the least k such that $3(k+1) > 2x$. Then k is the desired value.

Exercise 2. Let P be the program in Example 2.1. We claim

$$f_P^{(2)} = \begin{cases} x - y & \text{if } x \geq y \\ \text{undefined} & \text{otherwise} \end{cases}$$

Proof. We note that P repeatedly checks whether $r_1 = r_2$, and if not increments r_2 as well as the counter r_3. If and when $r_1 = r_2$, P halts and returns the value r_3.

Clearly then if $x < y$, P does not halt. If $x \geq y$, then P returns the value k satisfying $y + k = x$; in other words, P returns $x - y$ as desired. □

Exercise 3. Let P be a program with no jump instructions. Then there exists $m \in \mathbb{N}$ such that either

$$f_P^{(1)}(x) = m \qquad \text{or} \qquad f_P^{(1)}(x) = m + x$$

for all $x \in \mathbb{N}$.

Proof. We prove a more general claim. Let $f_{P,R_k}^{(1)}(x)$ denote the final contents of the register R_k under the computation $P(x)$ (or be undefined if $P(x)$ does not halt). We claim that for each $k \geq 1$, there exists $m_k \in \mathbb{N}$ such that either $f_{P,R_k}^{(1)}(x) = m_k$ or $f_{P,R_k}^{(1)}(x) = m_k + x$. The result then follows since $f_P^{(1)}(x) = f_{P,R_1}^{(1)}(x)$.

We prove the claim by induction on the length n of P. If $n = 0$, then trivially $f_{P,R_1}^{(1)}(x) = x$ and $f_{P,R_k}^{(1)}(x) = 0$ for all $k > 1$, so the claim holds. Now suppose $n > 0$ and the claim holds for all programs of length $n-1$ with no jump instructions. Let P^* be the program consisting of the first $n-1$ instructions of P. Then by assumption the claim holds for P^*, and hence the contents of register R_k after the first $n-1$ instructions in the computation $P(x)$ is given by $f_{P^*,R_k}^{(1)}(x)$, which is m_k or $m_k + x$ for some $m_k \in \mathbb{N}$.

Now consider instruction I_n in P by cases:

(i) If I_n is Z(q), then $f_{P,R_q}^{(1)}(x) = 0$.

(ii) If I_n is S(q), then $f_{P,R_q}^{(1)}(x) = f_{P^*,R_q}^{(1)}(x) + 1$, giving either $m_q + 1$ or $(m_q + 1) + x$.

(iii) If I_n is T(p,q), then $f_{P,R_q}^{(1)}(x) = f_{P^*,R_p}^{(1)}(x)$, giving either m_p or $m_p + x$.

In all cases, $f_{P,R_k}^{(1)}(x) = f_{P^*,R_k}^{(1)}(x)$ for all $k \neq q$. Therefore $f_{P,R_k}^{(1)}(x)$ is of the desired form for all $k \geq 1$, so the claim holds for P as desired. □

Exercise 4. For each transfer instruction T(m,n), with $m, n \geq 1$, there exists a program $P_{m,n}$ having the same effect on the register machine configuration and using no transfer instructions.

Proof. For $m = n$, let $P_{m,n}$ be the empty program. For $m \neq n$, let $P_{m,n}$ be given by:

1. Z(n)
2. J(m,n,5)
3. S(n)
4. J(1,1,2)

□

Chapter 2

Exercises 3.4

Exercise 1. For every $m \in \mathbb{N}$, the functions $\mathbf{m}(x) = m$ and mx, are computable.

Proof. We proceed by induction on m.

For the first class of functions, we know that $\mathbf{0}$ is computable by Lemma 1.1(a). Suppose that $\mathbf{m}$ is computable. Then $\mathbf{m}+\mathbf{1}$ is obtained by composition of $\mathbf{m}$ and the successor function, and hence is computable by Lemma 1.1(b) and Theorem 3.1.

For the second class of functions, $0x = \mathbf{0}$ is computable. Now suppose mx is computable. Since

$$(m+1)x = mx + x$$

is simply the composite of mx and the identity map x with the addition function, it is also computable by Theorem 3.1. □

Exercise 3. Let $g(x)$ be total computable. Define a binary predicate $M(x,y)$ by

$$M(x,y) \equiv g(x) = y$$

Then $M(x,y)$ is decidable.

Proof. We must show that the characteristic function

$$\varphi_M(x,y) = \begin{cases} 1 & \text{if } M(x,y) \\ 0 & \text{otherwise} \end{cases} = \begin{cases} 1 & \text{if } g(x) = y \\ 0 & \text{otherwise} \end{cases}$$

is computable. Let G be a procedure in standard form computing g and let k be the number of instructions in G and $r = \rho(G)+1$. Define a procedure P as follows:

$$\begin{array}{rl} 1. & \mathrm{T}(2,r) \\ 2. & \mathrm{Z}(2) \\ 3. & G \\ 3+k. & \mathrm{J}(1,r,6+k) \\ 4+k. & \mathrm{Z}(1) \\ 5+k. & \mathrm{J}(1,1,8+k) \\ 6+k. & \mathrm{Z}(1) \\ 7+k. & \mathrm{S}(1) \end{array}$$

Since G is total, P always halts, and P computes φ_M as desired. □

Exercises 4.16

Exercise 1. We prove that the following functions are computable:

(a) Any polynomial function $a_0 + a_1 x + \cdots + a_n x^n$, where $a_0, \ldots, a_n \in \mathbb{N}$.

Proof. Note any x^k is just the composite of the constant function $\mathbf{k}$ with the exponential function x^y, and hence is computable. Therefore any term $a_k x^k$ is computable since it is the composite of functions $\mathbf{a_k}$ and x^k with the product function xy. Finally, a polynomial is just obtained from iterated composition of these terms with the sum function $x+y$. □

(b) $\lfloor \sqrt{x} \rfloor$ *Proof:* $\lfloor \sqrt{x} \rfloor = \mu z < x+2\,(x < z^2) \dot{-} 1$.

(c) $\mathrm{LCM}(x,y)$ *Proof:* $\mathrm{LCM}(x,y) = \mu z < xy(z > 0 \text{ and } x|z \text{ and } y|z)$.

(d) $\mathrm{HCF}(x,y)$ *Proof:* $\mathrm{HCF}(x,y) = \mathrm{qt}(\mathrm{LCM}(x,y), xy)$ (assumed $x, y > 0$).

(e) $f(x) =$ number of prime divisors of x *Proof:* $f(x) = \sum_{z \le x} \mathrm{Pr}(\mathrm{div}(z,x) \cdot z)$.

(f) $\varphi(x)$ *Proof:* $\varphi(x) = \sum_{1 \le z < x} \overline{\mathrm{si}}(\mathrm{HCF}(z,x) \dot{-} 1)$.

Exercise 2. Let $\pi(x,y) = 2^x(2y+1) - 1$. Then π is a computable bijection from $\mathbb{N}^2$ to $\mathbb{N}$, and the functions π_1, π_2 such that $\pi(\pi_1(n), \pi_2(n)) = n$ for all n are computable.

Proof. Clearly π is a computable function.

To see π is injective, suppose $\pi(v,w) = \pi(x,y)$ for $v, w, x, y \in \mathbb{N}$. Then we have $2^v(2w+1) = 2^x(2y+1)$. If $v = 0$ or $x = 0$, then we must have both $v = 0$ and $x = 0$ lest one side of the equation be odd and the other even. It then also follows that $w = y$. If $v \neq 0$ and $x \neq 0$, note that $2^v | 2^x(2y+1)$, but since 2^v is even and $2y+1$ is odd, we must have $2^v | 2^x$. Similarly $2^x | 2^v$. Therefore $2^v = 2^x$, so $x = v$, and it then also follows that $w = y$. In either case, $(v,w) = (x,y)$, establishing injectivity of π.

To see π is surjective, let $n \in \mathbb{N}$ be arbitrary and let 2^x be the highest power of 2 dividing $n+1$ (note this is defined since $n+1 \neq 0$). Then the quotient of $n+1$ by 2^x must be odd of the form $2y+1$, so $n = 2^x(2y+1) - 1 = \pi(x,y)$.

Note $\pi_1(n) = \mu z \le n+1(2^z > n+1) \dot{-} 1$ is clearly computable, from which it follows that $\pi_2(n) = \mathrm{qt}(2, \mathrm{qt}(2^{\pi_1(n)}, n+1) \dot{-} 1)$ is also computable. □

Exercise 5. Recall any $x \in \mathbb{N}$ can be expressed uniquely in the form $x = \sum_{i=0}^{\infty} \alpha_i 2^i$ where $\alpha_i \in \{0,1\}$ for all i (this is the *binary expansion* of x). Hence if $x > 0$ there are unique expressions for x of the forms

$$x = 2^{b_1} + 2^{b_2} + \cdots + 2^{b_l} \qquad (l \ge 1,\ 0 \le b_1 < b_2 < \cdots < b_l) \qquad \text{and}$$
$$x = 2^{a_1} + 2^{a_1+a_2+1} + \cdots + 2^{a_1+\cdots+a_k+k-1}$$

For these expressions, define $\alpha(i,x) = \alpha_i$ and

$$l(x) = \begin{cases} l & \text{if } x > 0 \\ 0 & \text{otherwise} \end{cases} \qquad b(i,x) = \begin{cases} b_i & \text{if } x > 0, 1 \le i \le l \\ 0 & \text{otherwise} \end{cases} \qquad a(i,x) = \begin{cases} a_i & \text{if } x > 0, 1 \le i \le l \\ 0 & \text{otherwise} \end{cases}$$

Then α, l, b, a are all computable.

Proof. We exhibit each:

(i) To calculate α_i, we can right shift and then mod the binary expansion of x in order to isolate the i-th bit, so $\alpha(i,x) = \mathrm{rm}(2, \mathrm{qt}(2^i, x))$, which is computable.

(ii) Note l is the number of 1's in the binary expansion of x, so $l(x) = \sum_{i \le x} \alpha(i,x)$, which is also computable.

(iii) Note b_i is the digit index of the i-th 1 in the binary expansion of x (if such a 1 exists), so we can calculate it by sequentially searching digits until we have accumulated i 1's. Specifically,

$$b(i,x) = \begin{cases} \mu z \le x \big(\sum_{j \le z} \alpha(j,x) = i \big) & \text{if } x > 0 \text{ and } 1 \le i \le l(x) \\ 0 & \text{otherwise} \end{cases}$$

Therefore $b(i,x)$ is computable.

(iv) Note a_i measures the number of 0's between the $(i-1)$-th and i-th 1's in the binary expansion of x (where the 0-th 1 fictionally occurs just prior to the start of the binary expansion). Therefore, for $x > 0$ and $1 \le i \le l(x)$, we obtain a recursive definition for $a(i,x)$ of the form

$$\begin{aligned} a(1,x) &= b(1,x) \\ a(i+1,x) &= b(i+1,x) \dot{-} b(i,x) \dot{-} 1 \end{aligned}$$

By cases, we set $a(i,x) = 0$ otherwise. It follows that $a(i,x)$ is computable.

□

This exercise demonstrates that the third expression for x above can be regarded as a coding for the sequence $(a_1, \ldots, a_l)$, and each a_i can be effectively calculated from x. The coding uses a binary expansion like a delimited string, with 0's acting as tally marks for the values in the sequence and 1's acting as delimiters.

As an example, consider the following coding:

$$(1,2,3) \mapsto 100010010 = 1 \underbrace{000}_{3} 1 \underbrace{00}_{2} 1 \underbrace{0}_{1}$$

Exercises 5.4

Exercise 1. Let $f(x)$ be a total injective computable function. Then the inverse function $f^{-1}(y)$ is computable.

Proof. Note $f^{-1}(y) = \mu z(f(z) = y)$. □

Chapter 3

Exercises 4.6

Exercise 1. The unary function computed by the Turing machine in Example 4.2 is $f(x) = \left\lfloor \frac{x+1}{2} \right\rfloor$. *Proof:* The machine starts with $x+1$ 1's on the tape, and then zeros out every other one, leaving $\left\lfloor \frac{x+1}{2} \right\rfloor$ remaining.

Exercises 5.4

Exercise 1. Let $\Sigma = \{a, b\}$. We examine the ways in which the production

$$(\pi)\ S_1 bS_2 aaS_3 b \to S_3 abS_1$$

applies to the string $\sigma = babaabbbaab$.

The following are exhaustive possibilities:

(1) If the first b in π matches the first b in σ, then $S_1 = \Lambda$.

   (a) If the aa in π matches the first aa in σ, then $S_2 = ab$ and $S_3 = bbaa$. In this case the string generated is $bbaaab$.
   (b) If the aa in π matches the second aa in σ, then $S_2 = abaabb$ and $S_3 = \Lambda$. In this case the string generated is ab.

(2) If the first b in π matches the second b in σ, then $S_1 = ba$.

   (a) If the aa in π matches the first aa in σ, then $S_2 = \Lambda$ and $S_3 = bbaa$. In this case the string generated is $bbaaabba$.
   (b) If the aa in π matches the second aa in σ, then $S_2 = aabb$ and $S_3 = \Lambda$. In this case the string generated is $abba$.

(3) If the first b in π matches the third b in σ, then $S_1 = babaa$, $S_2 = b$, and $S_3 = \Lambda$. The string generated is $abbabaa$.

(4) If the first b in π matches the fourth b in σ, then $S_1 = babaab$ and $S_2 = S_3 = \Lambda$. The string generated is $abbabaab$.

Exercises 7.2

Exercise 1. Suppose $f(x)$ and $g(x)$ are effectively computable. Then

$$h(x) = \begin{cases} x & \text{if } x \in \mathrm{Dom}(f) \cap \mathrm{Dom}(g) \\ \text{undefined} & \text{otherwise} \end{cases}$$

is URM-computable.

Proof. Since f and g are computable, their values can be computed by algorithms. Consider the following algorithm to compute $h(x)$: 'Run an algorithm to compute $f(x)$. If it halts, then run an algorithm to compute $g(x)$. If it also halts, then return x.' This algorithm computes $h(x)$, for it halts (and returns x) iff the algorithms for computing $f(x)$ and $g(x)$ both halt, which occurs iff $x \in \mathrm{Dom}(f) \cap \mathrm{Dom}(g)$. By Church's Thesis then, it follows that h is URM-computable. □

Chapter 4

Exercises 1.6

Exercise 1.

(c) Let P be the program T(3,4), S(3), Z(1). Then

$$\begin{aligned}\beta(\mathrm{T}(3,4)) &= 4\pi(3-1,4-1)+2 \qquad \beta(\mathrm{S}(3)) = 4(3-1)+1 \quad \beta(\mathrm{Z}(1)) = 4(1-1)\\ &= 4[2^2(2\cdot 3+1)-1]+2 \qquad\quad = 9 \qquad\qquad\qquad = 0\\ &= 4\cdot 27+2\\ &= 110\end{aligned}$$

Therefore $\gamma(P) = \tau(110,9,0) = 2^{110}+2^{120}+2^{121}$.

(d) Let $P = P_{100}$. Note $100 = 1100100_2 = 2^2+2^5+2^6$, so $a_1 = 2$, $a_2 = 2$, and $a_3 = 0$. Now $2 = 4\cdot 0+2$, hence $\beta^{-1}(2) = \mathrm{T}(1,1)$. Similarly $\beta^{-1}(0) = \mathrm{Z}(1)$. Therefore P is the program T(1,1), T(1,1), Z(1).

Exercises 2.3

Exercise 1. Let f be computable. Then f has infinitely many indices.

Proof. Write $f = \phi_k^{(n)}$, so P_k computes f. Set $r = \rho(P_k)+1$. Then for any fixed m, the program

$$\begin{array}{l} P_k \\ \mathrm{Z(r)} \\ \vdots \qquad m \text{ times} \\ \mathrm{Z(r)} \end{array}$$

also computes f. Denote this program by P_k^m, so $\gamma(P_k^m)$ is also an index of f. By the injectivity of γ, the set $\Gamma = \{\gamma(P_k^m) \mid m \in \mathbb{N}\}$ gives infinitely many indices of f. □

Exercises 3.2

Exercise 3. Let $f : \mathbb{N} \to \mathbb{N}$ be partial and $m \in \mathbb{N}$. There exists a non-computable function g such that $g(x) \simeq f(x)$ for $x \le m$.

Proof. Define g as follows:

$$g(x) \simeq \begin{cases} f(x) & \text{if } x \le m \\ \phi_{x-(m+1)}(x)+1 & \text{if } x > m \text{ and } \phi_{x-(m+1)}(x) \text{ is defined} \\ 0 & \text{if } x > m \text{ and } \phi_{x-(m+1)}(x) \text{ is undefined} \end{cases}$$

By construction, $g(x) \simeq f(x)$ for $x \le m$ and g is non-computable since for any ϕ_k,

$$g(k+m+1) \not\simeq \phi_k(k+m+1)$$

□

Note that the above proof works by shifting the diagonal used in the diagonalization argument right by a finite number of places. A corollary of this exercise is that for any computable function ϕ_k, there exist non-computable functions agreeing with ϕ_k for up to any finite number of values.

Exercise 4.

(a) The set of all functions from $\mathbb{N}$ to $\mathbb{N}$ is uncountable.

Proof. The set is clearly infinite since, for example, there are infinitely many constant functions. The set is seen to be uncountably infinite using a standard diagonalization argument. □

(b) The set of all total non-computable functions from $\mathbb{N}$ to $\mathbb{N}$ is uncountable.

Proof. Let $\mathscr{T}$ be the set of all total functions from $\mathbb{N}$ to $\mathbb{N}$. Then it is immediate by diagonalization that $\mathscr{T}$ is uncountable. For each $\alpha \in \mathscr{T}$ define a function $f_\alpha : \mathbb{N} \to \mathbb{N}$ as follows:

$$f_\alpha(n) = \begin{cases} \phi_n(n) + \alpha(n) + 1 & \text{if } \phi_n(n) \text{ is defined} \\ \alpha(n) & \text{otherwise} \end{cases}$$

It is immediate that each f_α is total and non-computable. And clearly if $\alpha \neq \beta$, then $f_\alpha \neq f_\beta$. Therefore $\{ f_\alpha \mid \alpha \in \mathscr{T} \}$ is uncountable, establishing the result. □

Exercises 4.4

Exercise 1. There is a total computable function k such that $k(n)$ is an index of the function $\lfloor \sqrt[n]{x} \rfloor$.

Proof. Note $f(x,y) = \lfloor \sqrt[x]{y} \rfloor = \mu z(z^x > y) \dot{-} 1$ is computable. The result thus follows from the simple s-m-n theorem. □

Exercise 3. For fixed $n \geq 1$, there is a total computable function s such that

$$W^{(n)}_{s(x)} = \{ (y_1, \ldots, y_n) \mid y_1 + \cdots + y_n = x \}$$

Proof. Fix $n \geq 1$ and define

$$f(x, y_1, \ldots, y_n) = \begin{cases} 1 & \text{if } x = y_1 + \cdots + y_n \\ \text{undefined} & \text{otherwise} \end{cases}$$

Clearly f is computable, so $f = \phi^{(n)}_k$ for some index k. By the s-m-n theorem, there exists a total computable function $s(x) = s^1_n(k, x)$ such that for all x,

$$\phi^{(n)}_{s(x)}(y_1, \ldots, y_n) = f(x, y_1, \ldots, y_n)$$

Therefore by construction of f, $W^{(n)}_{s(x)}$ is as desired. □

Exercise 4. The functions s_n^m in in the s-m-n theorem are primitive recursive.

Proof. We merely sketch: since the encoding and decoding functions used by each s_n^m are all primitive recursive, each s_n^m itself is also primitive recursive. □

Exercise 5 (s-m-n theorem)**.** For each m there is a total computable $(m+1)$-ary function s^m such that for all $e, n, \vec{x}$,

$$\phi^{(n)}_{s^m(e,\vec{x})}(\vec{y}) \simeq \phi^{(m+n)}_e(\vec{x},\vec{y})$$

Proof. In the proof of Theorem 4.3, n is used only to count how many registers should be right shifted initially. But since P_e uses only registers $R_1,\ldots,R_{\rho(e)}$, it works equally well to shift $\rho(e)$ registers. Now $\rho(e)$ can be effectively computed from e by syntactically examining the finitely many instructions of P_e. Therefore n is not needed in the proof, and by constructing a similar proof but using $\rho(e)$ instead of n, one obtains a total computable function s^m as desired. □

Chapter 5

Exercises 1.5

Exercise 1.

(i) There exists a decidable predicate $Q(x,y,z)$ such that $y \in E_x$ iff $\exists z\, Q(x,y,z)$ and if $y \in E_x$ then $\phi_x((z)_1) = y$.

Proof. Note $y \in E_x$ iff there exists some s such that $\phi_x(s) = y$, that is, iff the computation $P_x(s)$ halts with output y after a finite number t of steps. Thus under the effective coding $z = 2^s 3^t$ we define

$$Q(x,y,z) \equiv S_1(x,s,y,t) = S_1(x,(z)_1,y,(z)_2)$$

Now Q is decidable by Corollary 1.3(a) and substitution, and the remaining properties follow easily. □

(ii) There exists a computable function $g(x,y)$ such that $y \in E_x$ iff $g(x,y)$ is defined, and if $y \in E_x$ then $g(x,y) \in W_x$ and $\phi_x(g(x,y)) = y$.

Proof. Define $g(x,y) \simeq \big(\mu z\, Q(x,y,z)\big)_1$. □

Exercise 2. Suppose f and g are unary computable functions. Then the function h defined by

$$h(x) = \begin{cases} 1 & \text{if } x \in \mathrm{Dom}(f) \cup \mathrm{Dom}(g) \\ \text{undefined} & \text{otherwise} \end{cases}$$

is computable.

Proof. Write $f = \phi_m$, $g = \phi_n$. Then $h(x) = \mathbf{1}(\mu t(H_1(m,x,t) \text{ or } H_1(n,x,t)))$. □

Note this exercise illustrates the theoretical possibility of effective *multitasking* or *multithreading*. Since any finite initial segment of a computation can be effectively simulated, a program can run through steps of multiple computations, giving the effect of 'running multiple programs simultaneously' (see Example 3.7.1.2).

Exercises 3.2

Remark. In these exercises we apply the *s-m-n* theorem and universal functions to prove effectiveness of operations on computable functions and predicates.

Exercise 1. There is a total computable function $k(e)$ such that for all e, if ϕ_e is the characteristic function of a decidable predicate $M(x)$, then $\phi_{k(e)}$ is the characteristic function for its negation 'not $M(x)$'.

Proof. Define $f(e,x) = 1 - \phi_e(x) = 1 \dot{-} \psi_U(e,x)$. Then f is computable, and for fixed e if $\phi_e(x)$ characterizes $M(x)$, then $f(e,x)$ characterizes its negation. By the *s-m-n* theorem, there exists a total computable function $k(e)$ such that $\phi_{k(e)}(x) \simeq f(e,x)$, establishing the result. □

Exercise 2. There exists a total computable function $k(e)$ such that $E_{k(e)} = W_e$.

Proof. Define $f(e,x) = x \cdot \mathbf{1}(\phi_e(x)) = x \cdot \mathbf{1}(\psi_U(e,x))$. Then f is computable, so by the *s-m-n* theorem there exists a total computable function $k(e)$ such that for all e $\phi_{k(e)}(x) \simeq f(e,x)$. By construction $x \in E_{k(e)}$ iff $x \in W_e$, so $E_{k(e)} = W_e$ as desired. □

Exercise 3. There exists a total computable function $s(x,y)$ such that, for all x and y, $E_{s(x,y)} = E_x \cup E_y$.

Proof. For a pair x and y, we must construct a single computable function the range of which consists of all elements from the ranges of ϕ_x and ϕ_y. Thus we partition our domain and define the computable function

$$f(x,y,z) \simeq \begin{cases} \phi_x\left(\frac{z}{2}\right) & \text{if } z \text{ is even} \\ \phi_y\left(\frac{z-1}{2}\right) & \text{if } z \text{ is odd} \end{cases}$$

By the *s-m-n* theorem, there exists a total computable function $s(x,y)$ such that for all x and y, $\phi_{s(x,y)}(z) \simeq f(x,y,z)$. Clearly $E_{s(x,y)} = E_x \cup E_y$. □

Exercise 5 (Effectiveness of substitution and minimalization)**.**

(a) Fix $m, n \geq 1$. There is a total computable function $s(e, e_1, \ldots, e_m)$ such that

$$\phi^{(n)}_{s(e,e_1,\ldots,e_m)} = \mathrm{Sub}(\phi^{(m)}_e; \phi^{(n)}_{e_1}, \ldots, \phi^{(n)}_{e_m})$$

Proof. Define the computable function

$$f(e, e_1, \ldots, e_m, \vec{x}) \simeq \phi^{(m)}_e\left(\phi^{(n)}_{e_1}(\vec{x}), \ldots, \phi^{(n)}_{e_m}(\vec{x})\right)$$

Now apply the *s-m-n* theorem to obtain s. □

(b) Fix $n \geq 1$. There is a total computable function $k(e)$ such that for all e

$$\phi^{(n)}_{k(e)}(\vec{x}) \simeq \mu y \left(\phi^{(n+1)}_e(\vec{x}, y) = 0\right)$$

Proof. Define

$$f(e, \vec{x}) \simeq \mu y \left(\phi^{(n+1)}_e(\vec{x}, y) = 0\right)$$

and apply the s-m-n theorem. □

Chapter 6

Exercises 1.8

Exercise 1. The following predicates are undecidable:

(a) '$x \in E_x$'

Proof. If '$x \in E_x$' is decidable, then the function

$$f(x) = \begin{cases} x & \text{if } x \notin E_x \\ \text{undefined} & \text{if } x \in E_x \end{cases}$$

is computable, say $f = \phi_m$. But then for all x, $x \in E_m$ iff $x \notin E_x$, so in particular $m \in E_m$ iff $m \notin E_m$—a contradiction. □

(b) '$W_x = W_y$'

Proof. Fix k with $\phi_k = \mathbf{0}$. Then $W_x = W_k$ iff ϕ_x is total, which is undecidable, hence '$W_x = W_y$' is undecidable. □

(c) '$\phi_x(x) = 0$'

Proof. Let $c(x)$ be the characteristic function of the predicate. Then for all x, $c(x) = 0$ iff $\phi_x(x) \neq 0$, hence $c \neq \phi_x$. Since c differs from all unary computable functions, c is not computable. □

(d) '$\phi_x(y) = 0$'. *Proof:* '$\phi_x(x) = 0$' reduces to this problem.

(e) '$x \in E_y$'. *Proof:* '$x \in E_x$' reduces to this problem.

(f) 'ϕ_x is total and constant'

Proof. Apply Rice's Theorem, or note that '$\phi_x = \mathbf{0}$' reduces to this problem since $\phi_x = \mathbf{0}$ iff ϕ_x is total and constant and $\phi_x(0) = 0$. □

(g) '$W_x = \emptyset$'. *Proof:* Apply Rice's Theorem with $\mathscr{B} = \{f \in \mathscr{C}_1 \mid \mathrm{Dom}\, f = \emptyset\}$.

(h) 'E_x is infinite'. *Proof:* Apply Rice's Theorem with $\mathscr{B} = \{f \in \mathscr{C}_1 \mid \mathrm{Ran}\, f \text{ is infinite}\}$.

(i) '$\phi_x = g$'. *Proof:* Apply Rice's Theorem with $\mathscr{B} = \{g\}$.

Exercise 2. There does not exist a total computable function $f(x,y)$ such that for all x, y, if $P_x(y)$ halts, it does so in at most $f(x,y)$ steps.

Proof. Suppose such a function $f(x,y)$ exists. Construct a program P which, on input x, runs for at least $f(x,x)+1$ steps and halts. Let $n = \gamma(P)$. Then ϕ_n is total, and the computation $\phi_n(n)$ runs for at least $f(n,n)+1 > f(n,n)$ steps—contradicting the definition of f.

Alternately, define the predicate $H(x,y) \equiv H_1(x,y,f(x,y))$, which is decidable if $f(x,y)$ is computable. Note $H(x,y)$ holds iff $P_x(y)$ halts, so the halting problem reduces to H. □

Exercises 6.14

Exercise 1. The following predicates are partially decidable:

(a) '$E_x^{(n)} \neq \emptyset$' (n fixed)

Proof. The predicate '$y \in E_x^{(n)}$' is partially decidable (Example 6.7.1), hence $M(x) \equiv \exists y(y \in E_x^{(n)})$ is partially decidable by Corollary 6.6, and $E_x^{(n)} \neq \emptyset$ iff $M(x)$ holds. □

(b) '$\phi_x(y)$ is a perfect square'

Proof. Note $M(z) \equiv$ 'z is a perfect square' is decidable, and therefore partially decidable, so its partial characteristic function c_M is computable. Hence the partial characteristic function $c(x,y)$ of the desired predicate is computed by

$$c(x,y) \simeq c_M(\phi_x(y)) \simeq c_M(\psi_U(x,y))$$

□

(c) 'n is a Fermat number'

Proof. By definition, n is a Fermat number iff $\exists x \exists y \exists z (x^n + y^n = z^n)$, which is partially decidable by Corollary 6.6. □

(d) 'There exists a run of exactly x consecutive 7's in the decimal expansion of π'

Proof. For fixed n it is decidable whether there exists such a run within the first n digits of the decimal expansion of π. Denote this predicate $M(x,n)$. Then the desired predicate is given by $\exists n\, M(x,n)$, which is partially decidable by Theorem 6.4. □

Exercise 2. Let $G = \langle S \mid R \rangle$ be a finitely presented group. Then the word problem for G is partially decidable.

Proof. For a word $w \in G$, if $w = 1$, then by definition of a group presentation this is a consequence of the relations in R and the group axioms, that is, the relation $w = 1$ is derivable from R in G. Derivability from R in G is partially decidable just like provability in the predicate calculus (Example 6.7.2). Therefore, the word problem is partially decidable. □

Exercise 4. Suppose $M(\vec{x})$ and $N(\vec{x})$ are partially decidable. Then the predicates $M(\vec{x}) \wedge N(\vec{x})$ and $M(\vec{x}) \vee N(\vec{x})$ are partially decidable, but $\neg M(\vec{x})$ is not necessarily partially decidable.

Proof. If $\phi_j^{(n)}$ and $\phi_k^{(n)}$ are partial characteristic functions for M and N, respectively, then it follows that

$$c_{M\wedge N}(\vec{x}) \simeq \mathbf{1}(\phi_j^{(n)}(\vec{x}), \phi_k^{(n)}(\vec{x})) \simeq \mathbf{1}(\psi_U^{(n)}(j, \vec{x}), \psi_U^{(n)}(k, \vec{x}))$$

is a computable partial characteristic function for $M \wedge N$, and similarly

$$c_{M\vee N}(\vec{x}) \simeq \mathbf{1}(\mu t[H_n(j, \vec{x}, t) \vee H_n(k, \vec{x}, t)])$$

is a computable partial characteristic function for $M \vee N$.

To see that $\neg M(\vec{x})$ is not necessarily partially decidable, let $M(x) \equiv x \in W_x$. Then $M(x)$ is partially decidable, but $\neg M(x)$ is not partially decidable (see Examples 6.2). □

Note the partial decision procedure for $M \vee N$ uses multithreading (see the comment after Exercise 5.1.5.2).

Exercise 5. Suppose $M(\vec{x}, y)$ is partially decidable. Then

(a) '$\exists y < z\, M(\vec{x}, y)$' is partially decidable.

Proof. Let $\phi_e^{(n+1)}$ denote the partial characteristic function for M. Then the partial characteristic function for this predicate is computed by

$$f(\vec{x}, z) \simeq \mathbf{1}(\mu t[\exists y < z H_{n+1}(e, \vec{x}, y, t)])$$

□

Note this partial decision procedure multithreads computation of the values $\phi_e^{(n+1)}(0), \ldots, \phi_e^{(n+1)}(z-1)$.

(b) '$\forall y < z\, M(\vec{x}, y)$' is partially decidable.

Proof. The partial characteristic function is computed by

$$f(\vec{x}, z) \simeq \prod_{y<z} \phi_e^{(n+1)}(\vec{x}, y)$$

□

(c) '$\forall y\, M(\vec{x}, y)$' is not necessarily partially decidable.

Proof. Set $M(x, y, t) \equiv \neg H_1(x, y, t)$. Then M is decidable but

$$\forall t\, M(x, y, t) \iff P_x(y) \text{ diverges}$$

which is not partially decidable (Corollary 6.12). □

Exercise 7.

(a) There does not exist a partially decidable predicate $M(x)$ and total computable function $k(x)$ such that $x \in W_x$ iff not $M(k(x))$.

Proof. For such a predicate, $M(k(x))$ holds iff $x \notin W_x$, which is not partially decidable (Example 6.2.4). □

(b) 'ϕ_x is not total' is not partially decidable.

Proof. By the *s-m-n* theorem, there exists a total computable function $k(x)$ such that for all x,

$$\phi_{k(x)}(y) = \begin{cases} 1 & \text{if } x \in W_x \\ \text{undefined} & \text{otherwise} \end{cases}$$

Thus $\phi_{k(x)}$ is not total iff $x \notin W_x$, so it follows from part (a) that 'ϕ_x is not total' is not partially decidable. □

(c) 'ϕ_x is total' is not partially decidable.

Proof. By the *s-m-n* theorem, there exists a total computable function $k(x)$ such that for all x,

$$\phi_{k(x)}(y) = \begin{cases} 1 & \text{if } \neg H_1(x, x, y) \\ \text{undefined} & \text{otherwise} \end{cases}$$

Thus $\phi_{k(x)}$ is total iff $x \notin W_x$, so 'ϕ_x is total' is not partially decidable by (a). □

This proof illustrates an important technique: if $M(x)$ is partially decidable and $\neg M(x)$ is not, then even though there is no partial decision procedure for $\neg M(x)$, we can characterize the failure to halt of a decision procedure for $M(x)$ in terms of other properties of computable functions, and thereby establish a lack of partial decidability for such properties. This technique is facilitated by the effectiveness of simulation and the *s-m-n* theorem, and is also used in the proof of the Rice-Shapiro Theorem (Theorem 7.2.16).

Exercise 9. Suppose $M(x_1, \ldots, x_n)$ is partially decidable and $g_1, \ldots, g_n$ are m-ary computable functions. Then the predicate

$$N(\vec{y}) \equiv M(g_1(\vec{y}), \ldots, g_n(\vec{y}))$$

is partially decidable.

Proof. If c_M is the partial characteristic function of M, then the partial characteristic function of N is computed by $c_N(\vec{y}) \simeq c_M(g_1(\vec{y}), \ldots, g_n(\vec{y}))$. □

Chapter 7

Exercises 1.4

Exercise 2.

(a) Let $B \subseteq \mathbb{N}$ be recursive and $n > 1$. Define the predicate

$$M(x_1, \ldots, x_n) \equiv \prod_{i=1}^{n} p_i^{x_i} \in B$$

Then M is decidable.

Proof. The characteristic function is given by $c_M(x_1, \ldots, x_n) = c_B(\prod p_i^{x_i})$, which is computable. □

(b) For $n > 1$, call $A \subseteq \mathbb{N}^n$ *recursive* iff the predicate '$\vec{x} \in A$' is decidable. Then A is recursive iff the set $B = \{\prod p_i^{x_i} \mid (x_1, \ldots, x_n) \in A\}$ is recursive.

Proof. This follows from (a) since $c_A(\vec{x}) = c_M(\vec{x})$. □

This exercise shows that we experience no loss of generality in restricting attention to recursive subsets of $\mathbb{N}$, since recursive subsets of more general sets can be fully characterized in $\mathbb{N}$ using effective codings.

Exercises 2.18

Exercise 1. For $a, e \in \mathbb{N}$, the set ${}^aW_e = \{x \mid \phi_e(x) = a\}$ is r.e., and ${}^aW_1, {}^aW_2, \ldots$ is an enumeration of all r.e. sets.

Proof. The set ${}^aW_e$ is r.e. since '$\phi_e(x) \simeq a$' is partially decidable (Theorem 6.6.13).

Now if A is r.e., $A = W_k$ for some k. Define $f(x) = \mathbf{a}(\phi_k(x))$. Then f is computable, so $f = \phi_m$ for some m, and

$$x \in A \iff x \in W_k \iff x \in W_m \iff \phi_m(x) = a \iff x \in {}^aW_m$$

Hence $A = {}^aW_m$. Since A was arbitrary, this establishes the enumeration. □

Exercise 2. The set $\mathscr{B} = \{x \mid \phi_x \text{ is not injective}\}$ is r.e.

Proof. The predicate $M(x, y, z) \equiv \phi_x(y) = \phi_x(z)$ is partially decidable, and

$$x \in \mathscr{B} \iff \exists y \exists z \big(y \neq z \wedge M(x, y, z) \big)$$

□

Exercise 3. There exist total computable functions k and l such that for all x,

$$W_x = E_{k(x)} \quad \text{and} \quad E_x = W_{l(x)}$$

Proof. We already know k exists by Exercise 5.3.2.2. Define

$$f(x,y) \simeq \mu z[\, S_1(x,(z)_1,y,(z)_2)\,]_1$$

By the s-m-n theorem, there exists a total computable function $l(x)$ such that for all x and y, $\phi_{l(x)}(y) = f(x,y)$. Then $y \in E_x$ iff $y \in W_{l(x)}$, hence $E_x = W_{l(x)}$. □

Exercise 4. If A is r.e., then $\bigcup_{x\in A} W_x$ and $\bigcup_{x\in A} E_x$ are r.e.

Proof. By definition,

$$y \in \bigcup_{x\in A} W_x \iff \exists x(x \in A \wedge y \in W_x)$$
$$y \in \bigcup_{x\in A} E_x \iff \exists x(x \in A \wedge y \in E_x)$$

and the predicates on the right are partially decidable. □

Exercise 5. Let f be a unary computable function and $A \subseteq \mathrm{Dom} f$. Then A is r.e. iff $g = f|_A$ is computable.

Proof. If g is computable, then $A = \mathrm{Dom} g$ is r.e. by Theorem 2.3. If A is r.e. and c_A is the partial characteristic function of A, then $g(x) = f(x \cdot c_A(x))$ is computable. □

Exercise 6. Let f be a unary function and set $A = \{2^x 3^{f(x)} \mid x \in \mathrm{Dom} f\}$. Then f is computable iff A is r.e.

Proof. If f is computable, $\mathrm{Dom} f$ is r.e. and hence is empty or enumerated by a total unary computable function g. If empty, $A = \emptyset$ is r.e. If not, define $h(x) = 2^{g(x)}3^{f(g(x))}$. Then h is computable and $A = \mathrm{Ran}\, h$, so A is r.e.

Conversely, if B is r.e., then '$2^x 3^y \in B$' is partially decidable, and $f(x) \simeq y$ iff $2^x 3^y \in B$, hence f is computable by Theorem 6.6.13. □

Exercise 7. Let A be infinite and r.e. Then A can be enumerated without repetitions by a total computable function.

Proof. Let f be a total unary computable function with $A = \mathrm{Ran} f$. Define

$$g(0) = f(0)$$
$$g(n) = f(\mu t[f(t) \notin \{f(0),\ldots,f(n-1)\}]) \qquad (n > 0)$$

Then g is well-defined, total, and computable, and by construction enumerates A without repetitions. □

Exercise 8. We consider sets:

A	A r.e.	$\overline{A}$ r.e.	A recursive
$\{x \mid x \in E_x\}$	Y	N	N
$\{x \mid x$ a perfect square$\}$	Y	Y	Y
$\{x \mid \phi_x$ injective$\}$	N	Y	N
$\{x \mid \exists \geq x$ 7's in $\pi\}$	Y	N	N
$\{x \mid P_m(x)\uparrow\}$	N	Y	N

Exercise 10. Let A be recursive, B r.e., and f a total unary computable function. Then $f^{-1}(A)$ is recursive, and $f(A)$, $f(B)$, and $f^{-1}(B)$ are r.e. but not necessarily recursive. If f is bijective on $\mathbb{N}$, then $f(A)$ is recursive.

Proof. $f^{-1}(B)$ is r.e. since $x \in f^{-1}(B) \iff f(x) \in B$, which is partially decidable since B is r.e. Similarly, $y \in f(B) \iff \exists x(x \in B \wedge f(x) \simeq y)$, which is also partially decidable, so $f(B)$ is r.e.

It follows that $f(A)$ and $f^{-1}(A)$ are r.e. since A is r.e., and $\overline{f^{-1}(A)} = f^{-1}(\overline{A})$ is r.e., so $f^{-1}(A)$ is recursive.

Note $f^{-1}(B)$ and $f(B)$ are not necessarily recursive; for example, take $B = K$ and f the identity map. Also $f(A)$ is not necessarily recursive; for example, take $A = \mathbb{N}$ and $f = \phi_k$ where ϕ_k is total, injective, and $K = E_k$ (cf. Exercise 7).

These examples show that we do not gain additional information if we assume f is injective. However, if f is bijective on $\mathbb{N}$, then $y \in f(A) \iff f^{-1}(y) \in A$, which is decidable by Exercise 2.5.4.1, so $f(A)$ is recursive. □

Exercise 11 (Rice-Shapiro)**.** The Rice-Shapiro Theorem implies Rice's Theorem.

Proof. Let $\mathscr{B} \subseteq \mathscr{C}_1$ with $\mathscr{B} \neq \emptyset, \mathscr{C}_1$. We must prove that $\mathscr{B}$ is not recursive using the Rice-Shapiro Theorem.

Fix $f \in \mathscr{B}$ and $g \in \overline{\mathscr{B}}$. Note $\emptyset \subseteq f, g$. If $\mathscr{B}$ is recursive, then $\mathscr{B}$ and $\overline{\mathscr{B}}$ are both r.e., so by the Rice-Shapiro Theorem, $\emptyset \notin \mathscr{B}$ since $g \notin \mathscr{B}$ and $\emptyset \notin \overline{\mathscr{B}}$ since $f \notin \overline{\mathscr{B}}$. But this is a contradiction since $\emptyset \in \mathscr{C}_1 = \mathscr{B} \cup \overline{\mathscr{B}}$. Therefore $\mathscr{B}$ is not recursive. □

Exercise 13.

(a) Define sets $K_0 = \{x \mid \phi_x(x) = 0\}$ and $K_1 = \{x \mid \phi_x(x) = 1\}$. Then K_0 and K_1 are recursively inseperable.

Proof. Trivially K_0 and K_1 are r.e. and $K_0 \cap K_1 = \emptyset$. If C is recursive and $K_0 \subseteq C$ and $K_1 \subseteq \overline{C}$, note that the characteristic function of C differs from ϕ_x at x for all x—a contradiction. □

(b) Let A and B be disjoint. Then A and B are recursively inseparable iff whenever $A \subseteq W_a$ and $B \subseteq W_b$ with $W_a \cap W_b = \emptyset$, then there exists $x \notin W_a \cup W_b$.

Proof. If for all x, $x \in W_a \cup W_b$, then W_a, W_b partition $\mathbb{N}$, so $W_b = \overline{W_a}$ and W_a, W_b are recursive. But then A and B are not recursively inseparable.

Conversely, if A and B are recursively inseparable, choose C recursive with $A \subseteq C$ and $B \subseteq \overline{C}$. Since C and $\overline{C}$ are r.e., there exist $a, b \in \mathbb{N}$ with $C = W_a$ and $\overline{C} = W_b$, and hence for all x, $x \in W_a \cup W_b$. □

Note intuitively that A and B are recursively inseparable iff there does not exist a general effective procedure by which to *rule out*, for an arbitrary x, one or the other of the alternatives $x \in A$ or $x \in B$.

Exercises 3.13

Exercise 1. The following sets are productive:

(a) $\{x \mid W_x \text{ is finite}\}$

(b) $\{x \mid \phi_x \text{ is not surjective}\}$

(c) $\{x \mid \phi_x \text{ is injective}\}$

(d) $\{x \mid \phi_x \text{ is not a polynomial}\}$

Proof. By Theorem 3.4. For (a)–(d), the empty function is in the set; for (a), (b), and (d), the identity function is not in the set; and for (c), the zero function is not in the set. □

Exercise 2. The following sets are creative:

(a) $\{x \mid x \in E_x\}$

(b) $\{x \mid E_x^{(n)} \neq \emptyset\}$

(c) $\{x \mid \phi_x \text{ is not injective}\}$

(d) $\{x \mid \phi_x(x) \in B\}$ where B is a nonempty r.e. set

(e) $\{x \mid \phi_x(x) = f(x)\}$ where f is a total computable function

Proof. By Theorem 3.8.

For (a) and (b), we know the sets are r.e., and we know the sets are not recursive hence nontrivial and proper.

For (c), we know the set is r.e. (Exercise 2.18.2), and the zero function is in but the identity function is not.

For (d), we know the set is r.e. since '$\phi_x(x) \in B$' is partially decidable. Fix $b \in B$. Then the function **b** is in the set, but the empty function is not.

For (e), we know the set is r.e. since '$\phi_x(x) = f(x)$' is partially decidable, and f is in the set but the empty function is not. □

Exercise 3. Let A and B be sets and suppose B is r.e. and $A \cap B$ is productive. Then A is productive.

Proof. First we claim there exists a total computable function k such that for all x, $W_{k(x)} = W_x \cap B$. Indeed, since B is r.e., by the s-m-n theorem there exists k with

$$\phi_{k(x)}(y) = \begin{cases} \phi_x(y) & \text{if } y \in B \\ \text{undefined} & \text{otherwise} \end{cases}$$

Then $W_{k(x)} = W_x \cap B$ as desired.

Now let f be a productive function for $A \cap B$. We claim $g = f \circ k$ is productive for A. Indeed, g is total computable, and given $W_x \subseteq A$, $W_{k(x)} \subseteq A \cap B$, so

$$g(x) = f(k(x)) \in A \cap B - W_{k(x)} = A \cap B - W_x \cap B \subseteq A - W_x$$

Thus A is productive. □

Exercise 4. Let A and C be disjoint sets and suppose A is r.e. and C is creative. Then $A \cup C$ is creative.

Proof. Since A and C are r.e., $A \cup C$ is r.e.

We claim that $\overline{A \cup C} = \overline{A} \cap \overline{C}$ is productive. Fix a total computable function k such that for all x, $W_{k(x)} = W_x \cup A$ (cf. Exercise 3), and let f be a productive function for $\overline{C}$. We claim that $g = f \circ k$ is productive for $\overline{A} \cap \overline{C}$. Indeed, g is total computable, and given $W_x \subseteq \overline{A} \cap \overline{C}$, $W_{k(x)} \subseteq \overline{C}$, so

$$g(x) = f(k(x)) \in \overline{C} - W_{k(x)} = \overline{C} - W_x \cup A \subseteq \overline{A} \cap \overline{C} - W_x$$

Thus $\overline{A} \cap \overline{C}$ is productive, and so $A \cup C$ is creative. □

This exercise shows that adjoining additional r.e. elements to a creative set preserves creativity, while the previous exercise shows that adjoining additional non-r.e. elements to a productive set preserves productivity.

Exercise 7. Let $\mathscr{B} \subseteq \mathscr{C}_1$, and suppose there exists $g \in \mathscr{B}$ such that for all finite $\theta \subseteq g$, $\theta \notin \mathscr{B}$. Then $\mathscr{B}$ is productive.

Proof. By the s-m-n theorem, there exists k such that for all x,

$$\phi_{k(x)}(y) = \begin{cases} g(y) & \text{if } \neg H_1(x,x,y) \\ \text{undefined} & \text{otherwise} \end{cases}$$

If $x \in \overline{K}$, then $\phi_{k(x)} = g \in \mathscr{B}$, and if $x \notin \overline{K}$, then $\phi_{k(x)} \subseteq g$ is finite, hence $\phi_{k(x)} \notin \mathscr{B}$. By Theorem 3.2, $\mathscr{B}$ is productive. □

Exercise 8. The following sets are productive:

(a) $\{x \mid \phi_x \text{ is total}\}$

(b) $\{x \mid \phi_x \text{ is a polynomial}\}$

Proof. Note the identity function is in both sets, but no finite function is in either set, so the sets are productive by Exercise 7. □

Exercise 9.

(a) The sets $K_0 = \{x \mid \phi_x(x) = 0\}$ and $K_1 = \{x \mid \phi_x(x) = 1\}$ are effectively recursively inseparable.

Proof. By the s-m-n theorem, there exists a total computable function $k(a,b)$ such that for all a, b with $W_a \cap W_b = \emptyset$,

$$\phi_{k(a,b)}(x) = \begin{cases} 1 & \text{if } x \in W_a \\ 0 & \text{if } x \in W_b \\ \text{undefined} & \text{otherwise} \end{cases}$$

Now suppose $K_0 \subseteq W_a$ and $K_1 \subseteq W_b$ with $W_a \cap W_b = \emptyset$. Set $n = k(a,b)$. Then if $n \in W_a \cup W_b$, $\phi_n(n) = 0$ iff $\phi_n(n) = 1$—a contradiction. Hence $n \notin W_a \cup W_b$. □

(b) Suppose A and B are r.e. sets which are effectively recursively inseparable. Then A and B are creative.

Proof. We must prove that $\overline{A}$ and $\overline{B}$ are productive.

To see $\overline{A}$ is productive, fix k such that for all x, $W_{k(x)} = W_x \cup B$ (cf. Exercise 3) and let f be a witness function for A and B. Define $g(x) = f(a, k(x))$, where $A = W_a$. Then given $W_x \subseteq \overline{A}$, $g(x) \notin A \cup W_{k(x)} = A \cup B \cup W_x$, so $g(x) \in \overline{A} - W_x$. So g is a productive function or $\overline{A}$.

To see $\overline{B}$ is productive, proceed similarly. □

Exercises 4.4

Exercise 2. Let f be an injective total computable function such that $\mathrm{Ran} f$ is not recursive. Then the set

$$A = \{x \mid \exists y > x\big(f(y) < f(x)\big)\}$$

is simple.

Proof. Note A is r.e. since membership is partially decidable.

If $\overline{A}$ is finite, then there exists some x_0 such that for all $x \geq x_0$, $x \in A$. But then we can recursively define

$$x_{n+1} = \mu x > x_n [f(x_{n+1}) < f(x_n)]$$

Now $f(x_0) > f(x_1) > \cdots$ is infinitely descending—a contradiction. Thus $\overline{A}$ is infinite.

To see that $\overline{A}$ contains no infinite r.e. subset, suppose towards a contradiction that $B \subseteq \overline{A}$ is infinite r.e. Then we can decide whether $y \in \mathrm{Ran} f$ as follows: search for $b \in B$ with $f(b) > y$ (such a b much exist since B is infinite and f is injective). Then we know for all $c > b$, $y < f(b) < f(c)$. Now check whether any of $f(0), \ldots, f(b-1)$ equal y. If so, $y \in \mathrm{Ran} f$; if not, then by the previous remark, $y \notin \mathrm{Ran} f$. Since $\mathrm{Ran} f$ is not decidable, this is a contradiction. □

Chapter 9

Exercises 1.7

Exercise 1.

(a) $K \leq_{\mathrm{m}} \{x \mid \phi_x(x) = 0\}$

Proof. By the s-m-n theorem, there exists a total computable function s such that for all x,

$$\phi_{s(x)}(y) = \begin{cases} 0 & \text{if } x \in W_x \\ \text{undefined} & \text{otherwise} \end{cases}$$

Clearly $x \in W_x$ iff $\phi_{s(x)}(s(x)) = 0$. □

(b) $K \leq_{\mathrm{m}} \{x \mid x \in E_x\}$

Proof. By the s-m-n theorem, there exists a total computable function t such that for all x,

$$\phi_{t(x)}(y) = \begin{cases} y & \text{if } x \in W_x \\ \text{undefined} & \text{otherwise} \end{cases}$$

Then $x \in W_x$ iff $t(x) \in E_{t(x)}$. □

Exercise 3.

(a) $\{x \mid \phi_x = \mathbf{0}\} \leq_{\mathrm{m}} \{x \mid \phi_x \text{ is total and constant}\}$

Proof. By the s-m-n theorem, there exists a total computable function k such that for all x,

$$\phi_{k(x)}(y) = y \sum_{z=0}^{y} \phi_x(z)$$

If $\phi_x = \mathbf{0}$, then $\phi_{k(x)} = \mathbf{0}$, so $\phi_{k(x)}$ is total and constant. If $\phi_x \neq \mathbf{0}$, then either ϕ_x is total or it is not. If ϕ_x is not total, then neither is $\phi_{k(x)}$ (in fact, if $\phi_x(y)$ is undefined, then $\phi_{k(x)}(z)$ is undefined for all $z \geq y$). If ϕ_x is total, choose y such that $\phi_x(y) > 0$. Then

$$\phi_{k(x)}(y+1) = (y+1) \sum_{z=0}^{y+1} \phi_x(z) \geq (y+1) \sum_{z=0}^{y} \phi_x(z) \geq \phi_{k(x)}(y) + \phi_x(y) > \phi_{k(x)}(y)$$

Hence $\phi_{k(x)}$ is not constant. □

(b) $\{x \mid \phi_x \text{ is total}\} \leq_{\mathrm{m}} \{x \mid W_x \text{ is infinite}\}$

Proof. Using the same function k from part (a). □

Exercise 5. Suppose A and B are r.e., $A \cup B = \mathbb{N}$ and $A \cap B \neq \emptyset$. Then $A \leq_{\mathrm{m}} A \cap B$.

Proof. Write $A = W_a$, $B = W_b$, and define $\tau(x) = \mu t[H_1(a,x,t) \vee H_1(b,x,t)]$. Then τ is total and computable, and hence so is

$$\pi(x) = \begin{cases} 0 & \text{if } H_1(a,x,\tau(x)) \\ 1 & \text{otherwise} \end{cases}$$

Note $\pi^{-1}(0) \subseteq A$ and $\pi^{-1}(1) \subseteq B$. Now fix $w \in A \cap B$ and define total computable

$$f(x) = (1 - \pi(x))\, w + \pi(x)\, x$$

If $x \in A$, either $\pi(x) = 0$ and $f(x) = w \in A \cap B$, or else $\pi(x) = 1$ so $x \in B$ and hence $f(x) = x \in A \cap B$. If $x \notin A$, then $x \in B$ and $\pi(x) = 1$, so $f(x) = x \notin A \cap B$. Therefore $x \in A$ iff $f(x) \in A \cap B$, so $f : A \leq_{\mathrm{m}} A \cap B$. □

Exercises 2.9

Exercise 1. The following sets are m-equivalent to K:

(a) $\{x \mid x \in E_x\}$

(b) $\{x \mid \phi_x(x) = 0\}$

Proof. These sets are m-reducible to K since they are r.e. (Theorem 1.6), and K is m-reducible to each of them by Exercise 1.7.1. □

Exercise 3. It is not true in general that if $A \equiv_{\mathrm{m}} \overline{A}$, then A is recursive.

Proof. Let $\mathbf{a} = \mathbf{0}'_m \cup \mathbf{0}'^*_m > \mathbf{0}$. Then $\mathbf{a} = \mathbf{a}^*$ by Exercise 5(d). Hence for $A \in \mathbf{a}$, A is not recursive but $A \equiv_{\mathrm{m}} \overline{A}$. □

Exercise 4. The following sets are all m-equivalent:

(a) $A = \{x \mid \phi_x = \mathbf{0}\}$

(b) $B = \{x \mid \phi_x \text{ is total and constant}\}$

(c) $C = \{x \mid W_x \text{ is infinite}\}$

Proof. We know $A \leq_{\mathrm{m}} B$ by Exercise 1.7.3(a). To see $B \leq_{\mathrm{m}} C$, note by the *s-m-n* theorem there exists a total computable function s such that for all x, $\phi_{s(x)}$ is defined recursively as follows:

$$\phi_{s(x)}(0) = \phi_x(0)$$

$$\phi_{s(x)}(y+1) = \begin{cases} \phi_x(y+1) & \text{if } \phi_{s(x)}(y) \text{ is defined and } \phi_x(y+1) = \phi_{s(x)}(y) \\ \text{undefined} & \text{otherwise} \end{cases}$$

If ϕ_x is total and constant, then $\phi_{s(x)} = \phi_x$ so $W_{s(x)} = \mathbb{N}$ is infinite. If ϕ_x is not total, then there exists y such that $\phi_x(y)$ is undefined; by construction then, $\phi_{s(x)}(z)$ is undefined for all $z \geq y$, so $W_{s(x)}$ is finite. If ϕ_x is total but not constant, then there exists a least $y > 0$ such that $\phi_x(y) \neq \phi_x(0)$; by construction again, $\phi_{s(x)}(z)$ is undefined for all $z \geq y$, so $W_{s(x)}$ is finite. Thus $s : B \leq_{\mathrm{m}} C$.

Finally, to see $C \leq_{\mathrm{m}} A$, note by the *s-m-n* theorem there exists a total computable function u such that for all x,

$$\phi_{u(x)}(y) = \mathbf{0}\big(\mu t\big[\bigvee_{i=0}^{t} H_1(x, y+i, t)\big]\big)$$

If W_x is infinite, then for any y there exists $z \geq y$ such that $\phi_x(z)$ is defined, that is, for any y there exists i and $t \geq i$ such that the computation $P_x(y+i)$ halts within t steps, so $\phi_{u(x)}(y) = 0$. Since y was arbitrary, $\phi_{u(x)} = \mathbf{0}$. Conversely, if W_x is finite, there exists some y such that for all $z \geq y$, the computation $P_x(z)$ never halts, so $\phi_{u(x)}(y)$ is undefined and $\phi_{u(x)} \neq \mathbf{0}$. Therefore we have $u : C \leq_{\mathrm{m}} A$.

Since $A \leq_{\mathrm{m}} B \leq_{\mathrm{m}} C \leq_{\mathrm{m}} A$, $A \equiv_{\mathrm{m}} B \equiv_{\mathrm{m}} C$ by Theorem 1.3. □

Exercise 5. Let $\mathbf{a}, \mathbf{b}$ be m-degrees.

(a) The least upper bound $\mathbf{a} \cup \mathbf{b}$ of $\mathbf{a}$ and $\mathbf{b}$ is uniquely determined.

Proof. We have existence by Theorem 2.8. If $\mathbf{c}$ and $\mathbf{d}$ are both least upper bounds, then $\mathbf{c} \leq_{\mathrm{m}} \mathbf{d}$ and $\mathbf{d} \leq_{\mathrm{m}} \mathbf{c}$ by definition, so $\mathbf{c} = \mathbf{d}$ by Theorem 2.6. □

(b) If $\mathbf{a} \leq_{\mathrm{m}} \mathbf{b}$, then $\mathbf{a} \cup \mathbf{b} = \mathbf{b}$.

Proof. By definition $\mathbf{b} \leq_{\mathrm{m}} \mathbf{a} \cup \mathbf{b}$. Now $\mathbf{b}$ is an upper bound of $\mathbf{a}$ and $\mathbf{b}$, hence $\mathbf{a} \cup \mathbf{b} \leq_{\mathrm{m}} \mathbf{b}$ by definition, so $\mathbf{a} \cup \mathbf{b} = \mathbf{b}$. □

(c) If $\mathbf{a}$ and $\mathbf{b}$ are r.e., then so is $\mathbf{a} \cup \mathbf{b}$.

Proof. By the proof of Theorem 2.8, since if A and B are r.e. sets, so is their direct sum. □

(d) If $\mathbf{a}^*$ denotes the m-degree of $\overline{A}$ for $A \in \mathbf{a}$, then $(\mathbf{a} \cup \mathbf{a}^*)^* = \mathbf{a} \cup \mathbf{a}^*$.

Proof. Note $\mathbf{a}^*$ is well-defined by Theorem 1.3(b).

Write $\mathbf{b} = \mathbf{a} \cup \mathbf{a}^*$, so we must prove $\mathbf{b}^* = \mathbf{b}$. We first claim $\mathbf{b}^*$ is an upper bound for $\mathbf{a}$ and $\mathbf{a}^*$, so that $\mathbf{b} \leq_{\mathrm{m}} \mathbf{b}^*$. Indeed, for $X \in \mathbf{b}^*$, $\overline{X} \in \mathbf{b}$ by definition. Thus for $Y \in \mathbf{a}$, $\overline{Y} \in \mathbf{a}^* \leq_{\mathrm{m}} \mathbf{b}$, so $\overline{Y} \leq_{\mathrm{m}} \overline{X}$ and $Y \leq_{\mathrm{m}} X$, and hence $\mathbf{a} \leq_{\mathrm{m}} \mathbf{b}^*$. By symmetry, $\mathbf{a}^* \leq_{\mathrm{m}} \mathbf{b}^*$, establishing our claim.

Now $\mathbf{b} \leq_{\mathrm{m}} \mathbf{b}^*$, so $\mathbf{b}^* \leq_{\mathrm{m}} (\mathbf{b}^*)^* = \mathbf{b}$ (Theorem 1.3(b)), so $\mathbf{b} = \mathbf{b}^*$ as desired. □

Exercise 6.

(a) Every nontrivial m-degree is denumerable.

Proof. Let $\mathbf{a}$ be an m-degree with $\mathbf{a} \neq \mathbf{o}, \mathbf{n}$. To see that $\mathbf{a}$ is infinite, fix $A \in \mathbf{a}$ and consider the sets

$$A_n = A + n = \{a + n \mid a \in A\}$$

Clearly $A_n \in \mathbf{a}$ for all n, and the A_n are pairwise distinct since if λ is least in A, $\lambda + n$ is least in A_n.

To see that $\mathbf{a}$ is countable, note for every $B \in \mathbf{a}$ there exists a distinct pair (f_B, g_B) of total computable functions with $f_B : A \leq_{\mathrm{m}} B$ and $g_B : B \leq_{\mathrm{m}} A$. Choose such a pair for each $B \in \mathbf{a}$ and define

$$\begin{aligned} \Phi : \mathbf{a} &\to \mathscr{C}_1 \times \mathscr{C}_1 \\ B &\mapsto (f_B, g_B) \end{aligned}$$

Then Φ is injective, and $\mathscr{C}_1 \times \mathscr{C}_1$ is countable, so $\mathbf{a}$ is countable. □

(b) There exist uncountably many m-degrees.

Proof. Let $\mathscr{D}$ be the set of m-degrees. If $\mathscr{D}$ is countable, then since each of the m-degrees is countable, we have

$$\mathscr{P}(\mathbb{N}) \subseteq \bigcup_{\mathbf{a}\in\mathscr{D}} \mathbf{a}$$

is countable—a contradiction since $\mathscr{P}(\mathbb{N})$ is uncountable. □

(c) There are only countably many r.e. m-degrees.

Proof. If there are uncountably many r.e. m-degrees, there are uncountably many distinct r.e. sets—a contradiction. □

Exercises 4.10

Exercise 1. If χ and ψ are total unary functions and ϕ_e^{χ} is total, ϕ_e^{ψ} need not be total.

Proof. Let e be the index of the following URMO program:

1. T(1,2)
2. O(2)
3. J(2,3,6)
4. S(1)
5. J(1,1,1)

Note P_e on input x searches for the first zero of the oracle function that is $\geq x$. Hence for $\chi = \mathbf{0}$, ϕ_e^{χ} is the total identity function, but for $\psi = \mathbf{1}$, ϕ_e^{ψ} is the empty function. □

Exercise 3. Let $\chi_1, \ldots, \chi_k$ be total unary functions and let $\mathscr{R}^{\chi_1,\ldots,\chi_k}$ be the class of $\chi_1, \ldots, \chi_k$-partial recursive functions.
Define $\chi(x) = \prod_{i=1}^{k} p_i^{\chi_i(x)}$. Then $\mathscr{C}^{\chi} = \mathscr{R}^{\chi_1,\ldots,\chi_k}$.

Proof. By Theorem 4.5, $\mathscr{C}^{\chi} = \mathscr{R}^{\chi}$, so it suffices to prove $\mathscr{R}^{\chi} = \mathscr{R}^{\chi_1,\ldots,\chi_k}$. By definition of χ, it is immediate that $\chi \in \mathscr{R}^{\chi_1,\ldots,\chi_k}$, hence $\mathscr{R}^{\chi} \subseteq \mathscr{R}^{\chi_1,\ldots,\chi_k}$. Conversely, note

$$\chi_i(x) = (\chi(x))_i \quad (1 \leq i \leq k)$$

Hence $\chi_i \in \mathscr{R}^{\chi}$ for $1 \leq i \leq k$, so $\mathscr{R}^{\chi_1,\ldots,\chi_k} \subseteq \mathscr{R}^{\chi}$. □

This exercise shows that we can fully characterize partial recursiveness relative to multiple oracles in terms of our existing notion of computability relative to a single oracle. We could also define directly a notion of computability relative to multiple oracles, but we would gain nothing by doing so, as this exercise illustrates.

Exercise 3. If χ is computable, then $\mathscr{C} = \mathscr{C}^{\chi}$.

Proof. Since the URMO extends the URM, $\mathscr{C} \subseteq \mathscr{C}^\chi$.

To prove $\mathscr{C}^\chi \subseteq \mathscr{C}$, we must show that for any URMO program P, there exists a URM program P^* such that P^* URM-computes the same functions that P^χ URMO-computes, that is, so that $f^{(n)}_{P^*} = f^{(n)}_{P^\chi}$ for all n. We sketch a construction.

Let P be a URMO program. Let Q be a URM program computing χ. Set $p = \rho(P)$, $q = \rho(Q)$, and $m = \max(p, q)$. Now construct a URM program P^* from P by replacing every instance of the oracle instruction O(n) in P by a sequence of URM instructions performing the following (and then adjusting jump instructions appropriately):

1. Transfer the contents of $R_1, \ldots, R_q$ to $R_{m+1}, \ldots, R_{m+q}$
2. Transfer the contents of R_n to R_1
3. Zero $R_2, \ldots, R_q$
4. Run Q
5. Transfer the contents of R_1 to R_n
6. Transfer the contents of $R_{m+1}, \ldots, R_{m+(n-1)}, R_{m+(n+1)}, \ldots, R_{m+q}$ to $R_1, \ldots, R_{n-1}, R_{n+1}, \ldots, R_q$

Among registers $R_1, \ldots, R_p$, this sequence effects the same configuration change on the URM as the oracle instruction O(n) does on the URMO when $1 \leq n \leq p$ and χ is in the oracle. Therefore, since these are the only registers referenced by P, the program P^* computes the same functions as P^χ. □

Exercise 6. Let A be a set.

(a) If B is r.e., then $B = W^A_e$ for some e.
Proof: $B = W_k$ for some k. Let e code P_k as a URMO program. Then $B = W^A_e$.

(b) If A is recursive, then W^A_e is r.e. for all e.
Proof: By Theorem 4.3(c).

(c) If A is recursive, K^A is r.e. but not recursive.
Proof: By Theorem 4.3(c) and Theorem 4.9(c).

Exercise 7. Let A, B, C be sets.

(a) If A is B-recursive and B is C-recursive, then A is C-recursive.
Proof: By Theorem 4.3, $c_A \in \mathscr{C}^A \subseteq \mathscr{C}^B \subseteq \mathscr{C}^C$.

(b) If A is B-r.e. and B is C-recursive, then A is C-r.e.
Proof: By Theorem 4.3, $c^*_A \in \mathscr{C}^B \subseteq \mathscr{C}^C$.

(c) If A is B-recursive and B is C-r.e., A need not be C-r.e.
Proof: Set $A = \overline{K}$, $B = K$, and $C = \emptyset$.

Exercise 8 (Relativized completeness)**.** For sets A and B, B is A-r.e. iff $B \leq_{\mathrm{m}} K^A$.

Proof. If $B \leq_{\mathrm{m}} K^A$, then B is A-r.e. since K^A is A-r.e.

If B is A-r.e., by the relativized s-m-n theorem there exists a total computable function $k(x)$ such that for all x, $\phi^A_{k(x)}(y) = c^*_B(x)$. If $x \in B$, $\phi^A_{k(x)} = \mathbf{1}$, so $k(x) \in K^A$; and if $x \notin B$, $\phi^A_{k(x)} = \emptyset$, so $k(x) \notin K^A$. Thus $k : B \leq_{\mathrm{m}} K^A$. □

Exercise 9. There exists k such that $K^A = W_k^A$ for all A.

Proof. By the remark following Theorem 4.7. Let Q be the universal program for unary URMO-computable functions (independent of the oracle), and let k be the index of a program computing $Q(x,x)$. Then for any A, $K^A = W_k^A$. □

Remark. Comparing Exercises 1 and 9, we see that with relative computability some results are independent of the oracle chosen while other results are not. Generally speaking, results fundamentally about *programs* (including the relativized *s-m-n* theorem and existence of universal programs) will be independent of the oracle, while results fundamentally about *functions* will not be. This may be viewed as a difference between *intensional* and *extensional* properties of programs.

Exercises 5.21

Exercise 1. The following sets are T-complete:

(a) $\{x \mid x \in E_x\}$

(b) $\{x \mid W_x \neq \emptyset\}$

Proof. Both of these sets are m-complete, hence T-complete by Theorem 5.2(c). □

Exercise 3. There exists a total computable function f such that for all A and B, if $c_A = \phi_e^B$ then $\phi_{f(e)} : K^A \leq_{\mathrm{m}} K^B$.

Proof. Let k be the index of the universal program for unary URMO-computable functions (see the remark following Theorem 4.7). Let $\sigma(e)$ be the total computable function which performs the following steps:

1. Decodes k to obtain P_k
2. Decodes e to obtain P_e
3. Replaces each oracle instruction in P_k with a 'subroutine call' to P_e (cf. the proof of Exercise 4.10.3).
4. Encodes the resulting program and returns the code.

Then clearly if $c_A = \phi_e^B$, $\phi_{\sigma(e)}^B(x,y) = \psi_U^A(x,y)$. Now define

$$\tau(e,x,y) = \phi_{\sigma(e)}^B(x,x)$$

Clearly τ is B-computable. Hence by the relativized *s-m-n* theorem (applied twice), there exists a total computable function f such that for all e, if $n = \phi_{f(e)}(x)$, then $\phi_n^B(y) = \phi_{\sigma(e)}^B(x,x)$ for all y. Thus $n \in K^B$ iff $x \in K^A$, so $\phi_{f(e)} : K^A \leq_{\mathrm{m}} K^B$. □

Exercise 4. Let A be a set. Define the following sequence:

$$\begin{aligned} A^{(0)} &= A \\ A^{(n)} &= K^{A^{(n-1)}} \quad (n > 0) \\ A^{(\omega)} &= \{\pi(m,n) \mid m \in A^{(n)}\} \end{aligned}$$

(a) $A^{(n)} <_{\mathrm{T}} A^{(\omega)}$ for all n.

Proof. Notice $A^{(n)} \leq_{\mathrm{m}} A^{(\omega)}$ since $x \in A^{(n)}$ iff $\pi(x,n) \in A^{(\omega)}$; hence $A^{(n)} \leq_{\mathrm{T}} A^{(\omega)}$. If $A^{(\omega)} \leq_{\mathrm{T}} A^{(n)}$, then in particular

$$K^{A^{(n)}} = A^{(n+1)} \leq_{\mathrm{T}} A^{(\omega)} \leq_{\mathrm{T}} A^{(n)} \leq_{\mathrm{T}} K^{A^{(n)}}$$

So $K^{A^{(n)}} \equiv_{\mathrm{T}} A^{(n)}$—contradicting Theorem 4.9(c). Therefore $A^{(n)} <_{\mathrm{T}} A^{(\omega)}$. □

(b) There exists a total computable function h such that $c_{A^{(n)}} = \phi^{A^{(\omega)}}_{h(n)}$ for all n.

Proof. Define $C(n,x) = c_{A^{(\omega)}}(\pi(x,n))$. Then C is $c_{A^{(\omega)}}$-computable and for all n, x, $C(n,x) = c_{A^{(n)}}(x)$. Thus by the relativized s-m-n theorem applied to C there exists a total computable function h such that $\phi^{A^{(\omega)}}_{h(n)} = c_{A^{(n)}}$ for all n. □

(c) Suppose B is a set and f is a total computable function such that $c_{A^{(n)}} = \phi^{B}_{f(n)}$ for all n. Then $A^{(\omega)} \leq_{\mathrm{T}} B$.

Proof. Note $c_{A^{(\omega)}}(x) = c_{A^{(\pi_2(x))}}(\pi_1(x)) = \phi^{B}_{f(\pi_2(x))}(\pi_1(x))$. □

(d) If $A \leq_{\mathrm{T}} B$, then $A^{(n)} \leq_{\mathrm{T}} B^{(n)}$ for all n and $A^{(\omega)} \leq_{\mathrm{T}} B^{(\omega)}$.

Proof. By assumption $A^{(0)} = A \leq_{\mathrm{T}} B = B^{(0)}$, and if $A^{(n)} \leq_{\mathrm{T}} B^{(n)}$, by Theorem 5.7(d)

$$A^{(n+1)} = K^{A^{(n)}} \leq_{\mathrm{T}} K^{B^{(n)}} = B^{(n+1)}$$

Therefore by induction the result holds for all n.

To prove $A^{(\omega)} \leq_{\mathrm{T}} B^{(\omega)}$, we appeal to the fact that $A^{(n)} \leq_{\mathrm{T}} B^{(n)} \leq_{\mathrm{T}} B^{(\omega)}$ and $A^{(\omega)}$ is simplest among the sets harder than all the $A^{(n)}$ (part (c)). To formalize this argument, we must make things effective.

Fix e with $c_A = \phi^B_e$, let f be as in Exercise 3, and let h be as in part (b) for $B^{(\omega)}$. Consider

$$\tau(n,e,x) = \phi^{B^{(\omega)}}_{h(n)}(\phi_{f(e)}(x))$$

Clearly τ is $B^{(\omega)}$-computable, so by the relativized s-m-n theorem, there exists a total computable function $\theta(n,e)$ such that $\phi^{B^{(\omega)}}_{\theta(n,e)} = \phi^{B^{(\omega)}}_{h(n)} \circ \phi_{f(e)}$ for all n, e. Now recursively define a total computable function $g(n)$ as follows:

$$g(0) = e$$
$$g(n+1) = \theta(n+1, g(n))$$

By induction, $\phi^{B^{(\omega)}}_{g(n)} = c_{A^{(n)}}$ for all n, so $A^{(\omega)} \leq_{\mathrm{T}} B^{(\omega)}$ by part (c). □

Chapter 10

In the following exercises, we say that an operator Φ is *finitely recursive* if it satisfies condition (b) of Theorem 1.5.

Exercises 1.7

Exercise 2. Let Φ be a recursive operator. Then if f is computable, so is $\Phi(f)$.

Proof. Let ϕ witness recursiveness of Φ. By definition, for all $\vec{x}$ and y,

$$\Phi(f)(\vec{x}) \simeq y \iff \exists \theta[\theta \subseteq f \wedge \phi(\widetilde{\theta}, \vec{x}) \simeq y]$$

Since f is computable, the predicate on the right is partially decidable, therefore $\Phi(f)(\vec{x}) \simeq y$ is partially decidable and $\Phi(f)$ is computable. □

This exercise shows that recursive operators preserve computability.

Exercise 3. For each of the operators below, we determine whether it is monotone, continuous, or recursive.

(a) $\Phi(f)(x) \simeq \begin{cases} f(x) & \text{if } \mathrm{Dom} f \text{ is finite} \\ \text{undefined} & \text{if } \mathrm{Dom} f \text{ is infinite} \end{cases}$

Φ is not monotone. Indeed, consider functions $f(x) \simeq \begin{cases} 0 & \text{if} x = 0 \\ \text{undefined} & \text{if } x \neq 0 \end{cases}$ and $g(x) = x$. Then $f \subseteq g$, but $\Phi(f) = f \not\subseteq f_\emptyset = \Phi(g)$. By Theorem 1.4, Φ is also neither continuous nor recursive.

(b) $\Phi(f)(x) \simeq \begin{cases} 0 & \text{if } f(x) \text{ is defined} \\ \text{undefined} & \text{otherwise} \end{cases}$

Φ is clearly continuous (and monotone) since $\Phi(f)(x) \simeq 0$ iff there exists $\theta \subseteq f$ with $\Phi(\theta)(x) \simeq 0$ (for the forward direction, take any $\theta \subseteq f$ with $x \in \mathrm{Dom}\theta$). Φ is also finitely recursive since

$$\phi(\widetilde{\theta}, x) \simeq \begin{cases} 0 & \text{if } \theta(x) \text{ is defined} \\ \text{undefined} & \text{otherwise} \end{cases}$$

is computable. Therefore Φ is recursive by Theorem 1.5.

(c) $\Phi(f)(x) \simeq \begin{cases} 0 & \text{if } f(x) \simeq n \in K \\ 1 & \text{if } f(x) \simeq n \notin K \\ \text{undefined} & \text{if } f(x) \text{ is undefined} \end{cases}$

Φ is clearly continuous (and monotone), but Φ is not recursive by the halting problem. Indeed, if ϕ witnesses recursiveness of Φ, then

$$c_K(x) = \overline{\mathrm{si}}(\phi(3^{x+1}, 0))$$

is a computable characteristic function for K—a contradiction.

(d) $\Phi(f)(x) \simeq \begin{cases} \text{undefined} & \text{if } \mathrm{Dom} f \text{ is finite} \\ f(x) & \text{if } \mathrm{Dom} f \text{ is infinite} \end{cases}$

Φ is monotone since if $f \subseteq g$, then either f is finite and $\Phi(f) = f_\emptyset \subseteq \Phi(g)$, or else f is infinite and hence g is also infinite, so $\Phi(f) = f \subseteq g = \Phi(g)$. Φ is clearly not continuous, and hence also not recursive.

Exercise 4. Let $\Phi : \mathscr{F}_m \to \mathscr{F}_n$ and $\Psi : \mathscr{F}_n \to \mathscr{F}_q$ be recursive operators. Then the composite $\Psi \circ \Phi : \mathscr{F}_m \to \mathscr{F}_q$ is also recursive.

Proof. Write $\Pi = \Psi \circ \Phi$. Since Φ and Ψ are continuous, Π is continuous (Exercise 7). We claim Π is also finitely recursive, so Π is recursive by Theorem 1.5.

Indeed, let ϕ and ψ witness finite recursiveness of Φ and Ψ, respectively, and define the function

$$\pi(z, \vec{x}) = \begin{cases} \Pi(\theta)(\vec{x}) & \text{if } z = \widetilde{\theta} \text{ for } \theta \in \mathscr{F}_m \\ \text{undefined} & \text{otherwise} \end{cases}$$

Then π is computable since

$$\begin{aligned} \pi(z, \vec{x}) \simeq y &\iff \exists \theta \big[z = \widetilde{\theta} \wedge \Pi(\theta)(\vec{x}) \simeq y \big] \\ &\iff \exists \theta \big[z = \widetilde{\theta} \wedge \Psi(\Phi(\theta))(\vec{x}) \simeq y \big] \\ &\iff \exists \theta \big[z = \widetilde{\theta} \wedge \exists \theta' [\theta' \subseteq \Phi(\theta) \wedge \Psi(\theta')(\vec{x}) \simeq y] \big] \\ &\iff \exists \theta \big[z = \widetilde{\theta} \wedge \exists \theta' [\forall \vec{w} \in \mathrm{Dom}\, \theta' [\theta'(\vec{w}) \simeq \phi(\widetilde{\theta}, \vec{w})] \wedge \psi(\widetilde{\theta'}, \vec{x}) \simeq y] \big] \end{aligned}$$

and the predicate on the right is partially decidable. Therefore π witnesses the finite recursiveness of Π, so Π is recursive as desired. □

This exercise shows the class of recursive operators is closed under composition.

Exercise 7. Let $\Phi : \mathscr{F}_m \to \mathscr{F}_n$ and $\Psi : \mathscr{F}_n \to \mathscr{F}_q$ be continuous operators. Then the composite $\Psi \circ \Phi : \mathscr{F}_m \to \mathscr{F}_q$ is also continuous.

Proof. Write $\Pi = \Psi \circ \Phi$. Then

$$\begin{aligned} \Pi(f)(\vec{x}) \simeq y &\iff \Psi(\Phi(f))(\vec{x}) \simeq y \\ &\iff \exists \theta \subseteq \Phi(f) [\Psi(\theta)(\vec{x}) \simeq y] && \text{by continuity of } \Psi \\ &\iff \exists \theta' \subseteq f \big[\exists \theta \subseteq \Phi(\theta') [\Psi(\theta)(\vec{x}) \simeq y] \big] && \text{by continuity of } \Phi \\ &\iff \exists \theta' \subseteq f [\Psi(\Phi(\theta'))(\vec{x}) \simeq y] && \text{by continuity of } \Psi \\ &\iff \exists \theta' \subseteq f [\Pi(\theta')(\vec{x}) \simeq y] \end{aligned}$$

Therefore Π is continuous. □

This exercise shows the class of continuous operators is closed under composition.

Exercise 8 (Enumeration operators)**.** We extend the notion of recursiveness to operators on sets of natural numbers.

Definition. Let $\Phi : \mathscr{P}(\mathbb{N}) \to \mathscr{P}(\mathbb{N})$ be an operator on sets of natural numbers. Then Φ is *recursive* if the operator $\Phi^* : \mathscr{F}_1 \to \mathscr{F}_1$ mapping $c^*_A \to c^*_{\Phi(A)}$ for $A \subseteq \mathbb{N}$ (where c^*_A denotes the partial characteristic function of A, and Φ^* maps all other functions to $f_\emptyset$, say) is recursive on partial characteristic functions.

Note by this definition, if $\Phi : \mathscr{P}(\mathbb{N}) \to \mathscr{P}(\mathbb{N})$ is recursive there exists a computable function $\phi^*(z,x)$ such that for all $A \subseteq \mathbb{N}$,

$$x \in \Phi(A) \iff \text{there exists finite } X \subseteq A \text{ such that } \phi^*(\widetilde{c^*_X}, x) \simeq 1$$

We can just think of $\widetilde{c^*_X}$ as a coding of X.

Definition. Let $\Phi : \mathscr{P}(\mathbb{N}) \to \mathscr{P}(\mathbb{N})$.

(a) Φ is *continuous* if for all $A \subseteq \mathbb{N}$ and $x \in \mathbb{N}$, $x \in \Phi(A)$ iff there exists finite $X \subseteq A$ such that $x \in \Phi(X)$.

(b) Φ is *monotone* if for all $A \subseteq B \subseteq \mathbb{N}$, $\Phi(A) \subseteq \Phi(B)$.

Theorem. *Recursive operators are continuous and monotone.*

Proof. Let $\Phi : \mathscr{P}(\mathbb{N}) \to \mathscr{P}(\mathbb{N})$ be recursive, and let ϕ^* be a witness as above.

To prove continuity, fix $A \subseteq \mathbb{N}$. If $x \in \Phi(A)$, then there exists finite $X \subseteq A$ such that $\phi^*(\widetilde{c^*_X}, x) \simeq 1$. Since $X \subseteq X$, this implies $x \in \Phi(X)$. Conversely, if there exists finite $X \subseteq A$ with $x \in \Phi(X)$, then there exists $W \subseteq X$ such that $\phi^*(\widetilde{c^*_W}, x) \simeq 1$. But since $W \subseteq A$ is also finite, this implies $x \in \Phi(A)$.

To prove monotonicity, fix $A \subseteq B \subseteq \mathbb{N}$. If $x \in \Phi(A)$, by continuity choose finite $X \subseteq A$ with $x \in \Phi(X)$. Then X is also a finite subset of B, so by continuity again $x \in \Phi(B)$. Therefore $\Phi(A) \subseteq \Phi(B)$. □

Theorem. *Let $\Phi : \mathscr{P}(\mathbb{N}) \to \mathscr{P}(\mathbb{N})$. Then Φ is recursive iff (a) Φ is continuous and (b) there exists a computable function $\phi^*(z,x)$ such that for all finite $X \subseteq \mathbb{N}$ and $x \in \mathbb{N}$, $c^*_{\Phi(X)}(x) \simeq \phi^*(\widetilde{c^*_X}, x)$.*

Proof. Suppose Φ is recursive. Then (a) holds since Φ is continuous by the previous theorem. Let ϕ^{**} be a witness function as above and define

$$\phi^*(z,x) = \begin{cases} c^*_{\Phi(X)}(x) & \text{if } z = \widetilde{c^*_X} \text{ for some finite } X \subseteq \mathbb{N} \\ \text{undefined} & \text{otherwise} \end{cases}$$

Then ϕ^* is computable since

$$\phi^*(z,x) \simeq y \iff \exists X[z = \widetilde{c^*_X} \wedge \exists Y[Y \subseteq X \wedge \phi^{**}(\widetilde{c^*_Y}, x) \simeq 1]]$$

and the predicate on the right is partially decidable, so ϕ^* satisfies (b).

Conversely, suppose (a) and (b) hold. We claim ϕ^* witnesses recursiveness of Φ^* on partial characteristic functions. Indeed, note

$$\begin{aligned}
\Phi^*(c_A^*)(x) \simeq c_{\Phi(A)}^*(x) \simeq 1 &\iff x \in \Phi(A) && \text{by definition of } \Phi^* \\
&\iff \exists \text{ finite } X \subseteq A[x \in \Phi(X)] && \text{by continuity of } \Phi \\
&\iff \exists \text{ finite } X \subseteq A[c_{\Phi(X)}^*(x) \simeq 1] \\
&\iff \exists \text{ finite } X \subseteq A[\phi^*(\widetilde{c_X^*}, x) \simeq 1] && \text{by (b)} \\
&\iff \exists \theta \subseteq c_A^*[\phi^*(\widetilde{\theta}, x) \simeq 1]
\end{aligned}$$

Thus Φ^* is recursive, so Φ is recursive. □

Exercises 2.6

Exercise 1. Let $\Phi : \mathscr{F}_m \to \mathscr{F}_n$ and $\Psi : \mathscr{F}_n \to \mathscr{F}_q$ be recursive operators. Then the composite $\Psi \circ \Phi : \mathscr{F}_m \to \mathscr{F}_q$ is also recursive.

Proof. We give an alternate proof using Myhill-Shepherdson (cf. Exercise 1.7.4).

Fix total computable j, k such that $\Phi(\phi_e) = \phi_{j(e)}$ and $\Psi(\phi_e) = \phi_{k(e)}$ for all e. Then

$$\Psi \circ \Phi(\phi_e) = \Psi(\Phi(\phi_e)) = \Psi(\phi_{j(e)}) = \phi_{k(j(e))} = \phi_{k \circ j(e)}$$

Now $\Psi \circ \Phi$ is continuous since Φ and Ψ are continuous (Exercise 1.7.7), hence $\Psi \circ \Phi$ is the unique continuous operator satisfying this equation, and is also recursive. □

Exercises 3.4

Exercise 2. Consider the recursive operator

$$\Phi(f)(x) = \begin{cases} f(f(x+11)) & x \leq 100 \\ x - 10 & x > 100 \end{cases}$$

Then the only fixed point of Φ is $m(x) = \begin{cases} 91 & x \leq 100 \\ x - 10 & x > 100 \end{cases}$.

Proof. It is immediate that $\Phi(m) = m$ since for $x \leq 100$, $91 \leq m(x+11) \leq 101$, so $m(m(x+11)) = 91 = m(x)$.

Suppose $\Phi(g) = g$, so $g(x) = g(g(x+11))$ for $x \leq 100$ and $g(x) = x - 10$ for $x > 100$. We claim $g = m$. First note $g(100 - y) = 91$ for $y \leq 100$ by induction on y. Indeed, $g(100) = g(g(100+11)) = g(101) = 91$. If $0 < y \leq 10$ and the result holds for $y-1$, then

$$g(100 - y) = g(g(100 - y + 11)) = g(100 - (y - 1)) = 91$$

In particular $g(91) = 91$. Finally, if $10 < y \leq 100$ and the result holds for $z < y$,

$$g(100 - y) = g(g(100 - (y - 11))) = g(91) = 91$$

By induction then, $g(x) = 91 = m(x)$ for all $x \leq 100$. Since $g(x) = x - 10 = m(x)$ for all $x > 100$, $g = m$ as claimed.

Therefore m is the only fixed point of Φ. □

Chapter 11

Exercises 1.10

Exercise 1. There exists n such that $\phi_n(x) = \left\lfloor \sqrt[n]{x} \right\rfloor$.

Proof. The function $f(e,x) = \left\lfloor \sqrt[e]{x} \right\rfloor$ is computable, so by the s-m-n theorem there is a total computable function $s(e)$ such that $\phi_{s(e)}(x) = f(e,x)$ for all x. By the second recursion theorem then, there is n with $\phi_n(x) = \phi_{s(n)}(x) = f(n,x) = \left\lfloor \sqrt[n]{x} \right\rfloor$ for all x. □

Exercise 5. Let $\mathscr{A} \subseteq \mathscr{C}_1$ and $A = \{ x \mid \phi_x \in \mathscr{A} \}$. Then $A \not\leq_{\mathrm{m}} \overline{A}$.

Proof. Suppose $f : A \leq_{\mathrm{m}} \overline{A}$. Then for all x, $x \in A \iff f(x) \notin A$. But by the second recursion theorem, there exists n with $\phi_n = \phi_{f(n)}$, so

$$n \in A \iff f(n) \in A \iff n \notin A$$

—a contradiction. Therefore $A \not\leq_{\mathrm{m}} \overline{A}$. □

Note if $\emptyset \subset A \subset \mathbb{N}$, then A cannot be recursive lest $A \equiv_{\mathrm{m}} \overline{A}$ (Theorem 9.1.3). This is just Rice's Theorem (Theorem 1.6).

Exercise 6. There exists a total computable function f such that

(i) if ϕ_n is total, then $\phi_{f(n)}$ is total

(ii) there is no fixed point n for f with ϕ_n total

Proof. By the s-m-n theorem, there exists a total computable function f with

$$\phi_{f(n)}(x) \simeq \phi_n(x) + 1$$

Clearly if ϕ_n is total, then so is $\phi_{f(n)}$, and in this case $\phi_n \neq \phi_{f(n)}$. □

Note $f_\emptyset$ is the least fixed point of the operation used in the proof.

Exercise 7 (Recursion theorem)**.** If f is a total computable function, for any $k \geq 1$ there exists n with $\phi^{(k)}_{f(n)} = \phi^{(k)}_n$.

Proof. Proceed identically to the proof for case $k = 1$ (Theorem 1.1), except using the k-ary s-m-n theorem with a k-ary universal function. □

Exercise 8 (Recursion theorem)**.** If f is a total computable function, there exists an increasing total computable function $n(t)$ such that for all t, $\phi_{f(n(t))} = \phi_{n(t)}$.

Proof. We first require a lemma:

Lemma. *There exists a total computable function $\alpha(e,k)$ such that for all e and k, $\phi_{\alpha(e,k)} = \phi_e$ and $\alpha(e,k) \geq k$.*

Proof. We give an informal algorithm for α and appeal to Church's Thesis.

Given e and k, let $r = \rho(P_e) + 1$ and let P_e^n denote the program obtained from P_e by appending n instructions $Z(r)$. Calculate $e_n = \gamma(P_e^n)$ for $1 \leq n \leq k$. By injectivity of program coding, the set $\{e_0, e_1, \ldots, e_k\}$ has cardinality $k+1$, so there must exist e_i with $e_i \geq k$. Let $\alpha(e,k)$ be the least such e_i. By construction, $\phi_{\alpha(e,k)} = \phi_e$. □

We now claim that there exists a total computable function $\psi(e,k)$ such that if ϕ_e is total, $\psi(e,k)$ is a fixed point for ϕ_e and $\psi(e,k) \geq k$. Indeed, first define

$$S(e,x,y) \simeq \phi_{\phi_e(\alpha(\phi_x(x),x))}(y)$$

(where the expression at right is undefined if the index is undefined). Then by the s-m-n theorem (applied twice), there exists a total computable function $\sigma(e)$ such that for all e, $\phi_{\sigma(e)}$ is total and for all x and y,

$$\phi_{\phi_{\sigma(e)}(x)}(y) \simeq \phi_{\phi_e(\alpha(\phi_x(x),x))}(y)$$

Now define $\sigma^*(e,k) = \alpha(\sigma(e),k)$ and $\psi(e,k) = \alpha(\phi_{\sigma^*(e,k)}(\sigma^*(e,k)), \sigma^*(e,k))$. We verify ψ has the desired properties. Note ψ is total and computable. Now let ϕ_e be total, and set $s = \sigma^*(e,k)$ and $n = \psi(e,k)$. Then $n \geq s \geq k$, and substituting into the identity above we obtain

$$\phi_{\phi_e(n)} = \phi_{\phi_e(\alpha(\phi_s(s),s))} = \phi_{\phi_s(s)} = \phi_{\alpha(\phi_s(s),s)} = \phi_n$$

Therefore n is a fixed point for ϕ_e and $n \geq k$, establishing our claim.

Finally, define $n(t)$ recursively by writing $f = \phi_e$ and setting

$$n(0) = \psi(e,0)$$
$$n(t+1) = \psi(e, n(t)+1)$$

Clearly n is an increasing total computable function whose values are fixed points for f, as required. □

Note the idea behind this proof is best understood relative to the discussion in the book on pp. 207–9. To find $\psi(e,k)$, we first use α to reindex the functions on the diagonal enumeration $\mathbf{D}$ so that the indices are kept suitably large along $\mathbf{D}$. We then use α again to find a suitably large index for the ϕ_e-transformed enumeration $\mathbf{D}^*$. The fixed point obtained using this index is guaranteed to be $\geq k$.

Exercises 3.4

Exercise 1 (Recursion theorem)**.** For all $k \geq 0$, there exists a $(k+1)$-ary total computable function $n(e,\vec{z})$ such that for all $\vec{z}$, if $\phi_e^{(k+1)}(x,\vec{z})$ is total, then $n(e,\vec{z})$ is a fixed point for $\phi_e^{(k+1)}$, that is,

$$\phi_{\phi_e^{(k+1)}(n(e,\vec{z}),\vec{z})} = \phi_{n(e,\vec{z})}$$

Proof. Fix $k \geq 0$ and define

$$f(x, e, \vec{z}) \simeq \phi_e^{(k+1)}(x, \vec{z})$$

Then f is computable (though not total), so by the proof of the second recursion theorem (Theorem 3.1), there exists a total computable function $n(e, \vec{z})$ such that if $f(n(e, \vec{z}), e, \vec{z})$ is defined, then $n(e, \vec{z})$ is a fixed point for f relative to $e, \vec{z}$.

Now fix e and $\vec{z}$ and suppose $\phi_e^{(k+1)}(x, \vec{z})$ is total. Then $f(x, e, \vec{z})$ is total, so $n(e, \vec{z})$ is a fixed point for f, that is

$$\phi_{n(e,\vec{z})} = \phi_{f(n(e,\vec{z}),\vec{z})} = \phi_{\phi_e^{(k+1)}(n(e,\vec{z}),\vec{z})}$$

But this just means $n(e, \vec{z})$ is a fixed point for $\phi_e^{(k+1)}$, as desired. □

Chapter 12

Exercises 1.8

Exercise 1. Let f be a total computable function taking only values in $\{0, 1\}$. For all m, there exists a program F computing f such that $t_F(x) \leq 2x + 3$ for all $x \leq m$.

In particular, if b is total computable and $b(x) > 2x + 3$ for all x, the restriction to almost all n in Theorem 4.1 is best case for b.

Proof. Let P be a program in standard form computing f. Construct F from P by prepending a finite table lookup for inputs $x \leq m$:

1.	J(1,2,$r + 2f(0)$)
2.	S(2)
3.	J(1,2,$r + 2f(1)$)
4.	S(2)
$\vdots$	$\vdots$
$2m + 1$.	J(1,2,$r + 2f(m)$)
	Z(2)
	P
	J(1,1,$r + 4$)
r.	Z(1)
	J(1,1,$r + 4$)
	Z(1)
	S(1)

Clearly F computes f. For inputs $x \leq m$, two instructions are executed for each $0 \leq y < x$, then three additional instructions are executed before the computation halts, giving $t_F(x) = 2x + 3$ for $x \leq m$ as desired.

The remainder of the exercise follows immediately, since if $b(x) > 2x + 3$, then no matter which function f is chosen in Theorem 4.1, we know there exist programs computing f in time bounded by b on arbitrarily large finite subsets of $\mathbb{N}$. □

Exercise 2. Let Φ_e be the storage space computational complexity measure from Example 1.5.2. Then

(i) if $\phi_e(x)$ is defined, then $\Phi_e(x) \geq \max(x, \phi_e(x))$.

(ii) if f is a total computable function and $X \subseteq \mathbb{N}$ is finite, there exists a program P_e computing f with $\Phi_e(x) = \max(x, f(x))$ for all $x \in X$.

Proof. If $\phi_e(x)$ is defined, then clearly $\Phi_e(x) \geq x$ since x appears in R_1 at the start of the computation and $\Phi_e(x) \geq \phi_e(x)$ since $\phi_e(x)$ appears in R_1 at the end of the computation. Therefore $\Phi_e(x) \geq \max(x, \phi_e(x))$.

Let f be total and suppose P computes f. If $X \subseteq \mathbb{N}$ is finite, then we can construct from P a program P_e computing f which performs a finite table lookup for inputs $x \in X$ (cf. the proof of Exercise 1). Then for inputs $x \in X$, P_e will only store numbers $\leq \max(x, f(x))$. It follows that $\Phi_e(x) = \max(x, f(x))$ for $x \in X$ as desired. □

References

[1] Cutland, N. J. *Computability: An introduction to recursive function theory.* New York: Cambridge, 1980.