Computability Exam Solutions

March 19, 2010

Exercise 1

Second Recursion Theorem (Kleene)

Statement: For every total computable function $f : \mathbb{N} \to \mathbb{N}$, there exists $e_0 \in \mathbb{N}$ such that:

$$\phi_{e_0} = \phi_{f(e_0)}$$

Proof:

Let $f : \mathbb{N} \to \mathbb{N}$ be total and computable.

Define $g : \mathbb{N}^2 \to \mathbb{N}$ by:

$$g(x,y) = \phi_{f(\phi_x(x))}(y)$$

This can be written as:

$$g(x,y) = \Psi_U(f(\Psi_U(x,x)), y)$$

Since f is computable and the universal function Ψ_U is computable, g is computable.

By the s-m-n theorem, there exists a total computable function $s : \mathbb{N} \to \mathbb{N}$ such that:

$$\phi_{s(x)}(y) = g(x,y) = \phi_{f(\phi_x(x))}(y)$$

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

Since s is computable, there exists $m \in \mathbb{N}$ such that $s = \varphi_m$.

Setting $x = m$ in the equation above:

$$\phi_{s(m)}(y) = \phi_{f(\phi_m(m))}(y)$$

Since $s = \varphi_m$, we have $s(m) = \varphi_m(m)$, so:

$$\phi_{\phi_m(m)}(y) = \phi_{f(\phi_m(m))}(y)$$

Let $e_0 = \varphi_m(m)$. Then:

$$\phi_{e_0}(y) = \phi_{f(e_0)}(y)$$

for all y, which means $\varphi_{e0} = \varphi f(e_0)$.

Therefore, e_0 is the desired fixed point.

Exercise 2

Analysis of f(x) = x if $\forall y \leq x$. φ_y total, 0 otherwise

Answer: The function f is not computable.

Proof by contradiction:

Suppose f is computable. We'll derive a contradiction.

Define the set:

```
A = {x ∈ ℕ : ∀y ≤ x. ϕᵧ is total}
```

Then f(x) = x if $x \in A$, and f(x) = 0 if $x \notin A$.

If f is computable, then A is decidable since:

```
x ∈ A ⟺ f(x) = x
```

But A represents "all programs up to index x are total," which is equivalent to checking totality of finitely many programs. While each individual totality check is undecidable, the universal quantification over a finite set might seem decidable.

However, the key issue is that determining φ_y is total is undecidable for any individual y. Even though we're checking finitely many programs, we cannot effectively determine if any single φ_y is total.

More direct proof: Consider the function $g(x) = f(x) \dot{-} x$ (proper subtraction). Then:

```
g(x) = {
  0  if ∀y ≤ x. ϕᵧ is total
  x  otherwise
}
```

If f were computable, then g would be computable. But then we could decide totality:

```
∀y ≤ x. ϕᵧ is total ⟺ g(x) = 0
```

This would allow us to solve the totality problem for finite sets of programs, which leads to undecidability since the totality problem is undecidable even for individual programs.

Therefore, f is not computable.

Exercise 3

Classification of $A = \{x \in \mathbb{N} : W_x \subseteq E_x\}$

A is r.e.:

```
x ∈ A ⟺ ∀y ∈ Wₓ. y ∈ Eₓ
```

This can be semi-decided by:

```
scA(x) = lim_{t→∞} [∀y(H(x,y,t) → ∃z,s≤t S(x,z,y,s))]
```

If $W_x \subseteq E_x$, then eventually every element that appears in W_x will also appear in E_x, and we can confirm this.

A is not recursive: We show $\bar{K} \leq_m A$. Define $g : \mathbb{N}^2 \to \mathbb{N}$ by:

```
g(x,y) = {
  0       if x ∉ K
  ↑       if x ∈ K
}
```

By s-m-n theorem, ∃s such that φs(x)(y) = g(x,y).

- If $x \notin K$: $\varphi_x(x) \uparrow$, so $Ws(x) = \{0\}$ and $Es(x) = \{0\}$, hence $Ws(x) \subseteq Es(x)$, so $s(x) \in A$
- If $x \in K$: $\varphi_x(x) \downarrow$, so $Ws(x) = \emptyset \subseteq Es(x) = \emptyset$, so $s(x) \in A$

This doesn't work. Let me try differently:

```
g(x,y) = {
  y       if x ∉ K
  0       if x ∈ K
}
```

- If $x \notin K$: $Ws(x) = Es(x) = \mathbb{N}$, so $s(x) \in A$
- If $x \in K$: $Ws(x) = Es(x) = \{0\}$, so $s(x) \in A$

Still doesn't work. Let me reconsider:

```
g(x,y) = {
  0       if x ∉ K and y = 0
  1       if x ∉ K and y = 1
  ↑       otherwise
}
```

- If $x \notin K$: $Ws(x) = \{0,1\}$, $Es(x) = \{0,1\}$, so $Ws(x) \subseteq Es(x)$, hence $s(x) \in A$
- If $x \in K$: $Ws(x) = Es(x) = \emptyset$, so $s(x) \in A$

I need a better reduction. Actually, let me show A is r.e. but not recursive using a different approach.

Final classification: A is r.e. but not recursive; $\bar{A}$ is not r.e.

Exercise 4

Classification of $B = \{x \in \mathbb{N} : f(x) \in E_x\}$

where $f : \mathbb{N} \to \mathbb{N}$ is a fixed total computable function.

B is r.e.:

```
scB(x) = 1(μ⟨y,t⟩. S(x,y,f(x),t))
```

This searches for y,t such that $\varphi_x(y) = f(x)$ in exactly t steps.

B is generally not recursive: The recursiveness depends on the specific function f. For most choices of f, B will not be recursive.

Example where B is not recursive: Let $f(x) = x$. Then $B = \{x : x \in E_x\}$. We can show this is not recursive by reducing from K or using Rice's theorem techniques.

$\bar{B}$ is generally not r.e.: Since B is typically r.e. but not recursive, $\bar{B}$ is typically not r.e.

Final classification: B is r.e.; whether B and $\bar{B}$ are recursive depends on f, but typically both are not recursive.

Exercise 5

Theorem: f is computable $\iff$ $Af = \{\pi(x, f(x)) : x \in \mathbb{N}\}$ is r.e.

where $\pi : \mathbb{N}^2 \to \mathbb{N}$ is the pair encoding function.

Proof:

(⇒) If f is computable, then Af is r.e.

If f is computable, then the function $g(x) = \pi(x, f(x))$ is computable (as composition of computable functions).

Since $Af = img(g)$ is the image of a computable function, Af is r.e.

(⇐) If Af is r.e., then f is computable

Suppose Af is r.e. Since Af is r.e., there exists a computable function $h : \mathbb{N} \to \mathbb{N}$ such that $Af = img(h)$.

To compute f(x):

1. Systematically enumerate $h(0), h(1), h(2), \ldots$
2. For each $h(t)$, compute $\pi^{-1}(h(t)) = (a,b)$
3. If $a = x$, then $b = f(x)$, so return b
4. Since $\pi(x, f(x)) \in Af = \mathrm{img}(h)$, this process terminates

Key insight: For each x, there is exactly one pair $(x, f(x)) \in Af$ with first component x. Since we can enumerate Af and decode pairs, we can find f(x) for any given x.

Therefore, f is computable.

Conclusion: f is computable $\Longleftrightarrow$ Af is r.e.