CS5311 Computation Theory Midterm Solutions
Spring 2002

1. **[10 points]** The following is an incorrect proof of EMPTY $\leq_{\mathrm{m}}$ TOTAL, where EMPTY $= \{ x \mid W_x = \emptyset \}$ and TOTAL $= \{ x \mid \Phi_x \text{ is total} \}$. Point out the problem *explicitly* and explain why it is a mistake.

   - Define a function g as follows:

$$g(y,x) = \begin{cases} 0 & x \in \text{EMPTY} \\ \uparrow & \text{otherwise} \end{cases}$$

   - Because g is partially computable, let $\#g$ be the program number that computes g.
   - By the *s-m-n* Theorem, there is a primitive recursive function $f(x) = S_1^1(x, \#g)$ such that $\Phi(y, x, \#g) = \Phi(y, S_1^1(x, \#g)) = \Phi(y, f(x))$.
   - Now, we have the following:

$$\begin{aligned} x \in \text{EMPTY} &\Rightarrow (\forall_y)[g(y,x) = 0] \\ &\Rightarrow (\forall_y)[\Phi(y, f(x)) = 0] \\ &\Rightarrow W_{f(x)} = \mathcal{N} \\ &\Rightarrow f(x) \in \text{TOTAL} \end{aligned}$$

   - On the other hand, we also have

$$\begin{aligned} x \notin \text{EMPTY} &\Rightarrow (\forall_y)[g(y,x)\uparrow] \\ &\Rightarrow (\forall_y)[\Phi(y, f(x))\uparrow] \\ &\Rightarrow W_{f(x)} = \emptyset \\ &\Rightarrow f(x) \notin \text{TOTAL} \end{aligned}$$

   - Therefore, $x \in \text{EMPTY} \Leftrightarrow f(x) \in \text{TOTAL}$ and EMPTY $\leq_{\mathrm{m}}$ TOTAL.

Answer: It is wrong, or at least not completely correct, to say that g is not computable because EMPTY is not r.e. There have been many examples covered in class and in homework showing to you that we could use K even though the reduction is meant for $\bar{K}$. Thus, even though EMPTY is not r.e., we could still use the complement of EMPTY to write a program for g. Since we know that x is in EMPTY if and only if $W_x = \emptyset$, the complement of EMPTY is $\overline{\text{EMPTY}} = \{ x \mid W_x \neq \emptyset \}$. It is easy to show (by the Projection Theorem) that $\overline{\text{EMPTY}}$ is r.e. By the definition of r.e. sets, there is a partially computable function f such that $\overline{\text{EMPTY}} = \{ x \mid f(x)\downarrow \}$. Therefore, g can be rewritten as follows:

$$g(y,x) = \begin{cases} 0 & \text{otherwise} \\ \uparrow & f(x)\downarrow \end{cases}$$

Intuitively, no program can implement g completely because we have no way to return 0 if $f(x)\uparrow$. However, a rigorous proof is always required.

If g is partially computable, by the Recursion Theorem, there is an e such that $g(e,x) = \Phi(x,e)$. Now, we have

$$\begin{aligned} \Phi(e,e)\downarrow &\Leftrightarrow g(e,e)\downarrow \\ &\Leftrightarrow f(e)\uparrow \quad \text{(definition of } g\text{)} \\ &\Leftrightarrow e \in \text{EMPTY} \quad \text{(definition of } f\text{)} \end{aligned}$$

This is a contradiction, because $\Phi(e,e)\downarrow$ means $e \in W_e \neq \emptyset$ and $e \notin$ EMPTY. Therefore, g is not partially computable. ∎

2. **[10 points]** Suppose A is r.e. such that $\bar{A}$ is infinite and contains no infinite r.e. subset. Show that A cannot be recursive.

   Answer: Since $\bar{A}$ does not have any infinite r.e. subset, $\bar{A}$ is not r.e. Otherwise, $\bar{A}$ is an infinite r.e. subset of itself, which contradicts to the assumption. If A is recursive, then A and $\bar{A}$ must be both r.e. which is a contradiction. ∎

3. **[10 points]** Does $A \subseteq B$ imply $A \leq_{\rm m} B$? State your claim first and prove your claim. Providing only a claim receives **no** credit.

   Answer: The answer is no. Let $A = \bar{K}$ and $B = \mathcal{N}$. If $\bar{K} \subseteq \mathcal{N}$ implies $\bar{K} \leq_{\rm m} \mathcal{N}$, $\bar{K}$ is r.e., which is a contradiction. ∎

4. **[10 points]** Is there a recursive m-complete set? State your claim first and prove your claim. Providing only a claim receives **no** credit.

   Answer: No m-complete set can be recursive. Let X be a recursive m-complete set. Since every r.e. set reduces to X, every r.e. set is recursive. However, we know this is not true because K is r.e. but non-recursive. Therefore, we have a contradiction. ∎

5. **[15 points]** Let MONOTONE = $\{\ x \mid \Phi_x$ is total and $(\forall_y)[\Phi_x(y) \leq \Phi_x(y+1)]\ \}$. Show that $\bar{K} \leq_{\rm m}$ MONOTONE.

   Answer: Since we cannot use $\bar{K}$ directly, we shall use STEP(). Compare this with Problem 1. Define function g as follows:

$$g(y,x) = \begin{cases} y & \neg\text{STEP}(x,x,y) \\ \uparrow & \text{STEP}(x,x,y) \end{cases}$$

   This is a partially computable function. Let the number of the program that computes g be $\#g$. By the *s-m-n* Theorem, we have $g(y,x) = \Phi(y,x,\#g) = \Phi(y, S_1^1(x,\#g)) = \Phi(y, f(x))$, where $f(x) = S_1^1(x,\#g)$. Now, we can show $\bar{K} \leq_{\rm m}$ MONOTONE as follows:

$$\begin{aligned} x \in \bar{K} &\Rightarrow \Phi(x,x)\uparrow \\ &\Rightarrow (\forall_y)[\neg\text{STEP}(x,x,y)] \\ &\Rightarrow (\forall_y)[g(y,x) = y] \\ &\Rightarrow (\forall_y)[\Phi(y,f(x)) = y] \\ &\Rightarrow f(x) \text{ is the identity function} \\ &\Rightarrow f(x) \in \text{MONOTONE} \end{aligned}$$

   On the other hand, we also have

$$\begin{aligned} x \notin \bar{K} &\Rightarrow x \in K \\ &\Rightarrow \Phi(x,x)\downarrow \\ &\Rightarrow (\exists_y)[\text{STEP}(x,x,y)] \\ &\Rightarrow (\exists_y)[g(y,x)\uparrow] \\ &\Rightarrow (\forall_y)[\Phi(y,f(x))\uparrow] \\ &\Rightarrow f(x) \text{ is not total} \\ &\Rightarrow f(x) \notin \text{MONOTONE} \end{aligned}$$

   Therefore, $\bar{K} \leq_{\rm m}$ MONOTONE. ∎

6. **[15 points]** Show that there is a number n such that $W_n = W_{n\ominus 1} \cup W_{n+1}$ holds, where $n \ominus 1$ is $n-1$ if $n \geq 1$ and 0 otherwise.

   Answer: Let function g be defined as follows:

$$g(y,x) = \begin{cases} 1 & x \in W_{y\ominus 1} \cup W_{y+1} \\ \uparrow & \text{otherwise} \end{cases}$$

   Notice that the first condition corresponds exactly to the desired result. However, before we can proceed, we must prove function g is partially computable because it is not so trivial.

   Since $W_{y\ominus 1}$ and W_{y+1} are both r.e., $W_{y\ominus 1} \cup W_{y+1}$ is also r.e. Therefore, there exists a partially computable function f such that $f(x)\downarrow$ if and only if $x \in W_{y\ominus 1} \cup W_{y+1}$. Thus, function g can be implemented as follows:

```
Φ(x, #f);     // run f on x
y := 1;       // if f halts on x, return 1
```

   Therefore, g is partially computable. By the Recursion Theorem, there is an n such that $g(n,x) = \Phi(x,n)$. Hence, we have $x \in W_n \Leftrightarrow \Phi(x,n)\downarrow \Leftrightarrow g(n,x)\downarrow \Leftrightarrow x \in W_{n\ominus 1} \cup W_{n+1}$. This means $W_n = W_{n\ominus 1} \cup W_{n+1}$. ∎

7. **[15 points]** Use the Recursion Theorem to prove that FINITE is not r.e., where FINITE $= \{x \mid W_x \text{ is a finite set }\}$.

   Answer: If FINITE is r.e., then there is a partially computable function f such that FINITE $= \{\ x \mid f(x)\downarrow\ \}$. Consider the following function:

$$g(y,x) = \begin{cases} x & \text{if } f(y)\downarrow \\ \uparrow & \text{otherwise} \end{cases}$$

   Then, $g(y,x)$ is partially computable. Why? Let $\#g$ be the program that computes g. By the Recursion Theorem, there is a e such that $\Phi(x,e) = g(e,x)$. Is e in FINITE?

$$\begin{aligned} e \in \text{FINITE} &\Rightarrow f(e)\downarrow \quad \text{(definition of } f\text{)} \\ &\Rightarrow (\forall_x)[g(e,x) = x] \\ &\Rightarrow (\forall_x)[\Phi(x,e) = x] \\ &\Rightarrow e \text{ is the identity function} \\ &\Rightarrow W_e = \mathcal{N} \\ &\Rightarrow e \notin \text{FINITE} \end{aligned}$$

   On the other hand, we have

$$\begin{aligned} e \notin \text{FINITE} &\Rightarrow f(e)\uparrow \quad \text{(definition of } f\text{)} \\ &\Rightarrow (\forall_x)[g(e,x)\uparrow] \\ &\Rightarrow (\forall_x)[\Phi(x,e)\uparrow] \\ &\Rightarrow e \text{ is defined nowhere} \\ &\Rightarrow W_e = \emptyset \\ &\Rightarrow e \in \text{FINITE} \end{aligned}$$

   Therefore, we have $e \in$ FINITE $\Leftrightarrow e \notin$ FINITE, which is a contradiction. Hence, FINITE is not r.e. ∎

8. **[15 points]** Let $A = \{ x \mid \Phi_x \text{ is not one-to-one} \}$. Is A recursive, r.e. but not recursive, or not r.e.? State your claim first and prove your claim. Providing only a claim receives **no** credit.

   Answer: This problem tests if you have the right sense that can distinguish r.e. and non-r.e. sets.

   A cannot be recursive. If A is recursive, $\bar{A} = \{ x \mid \Phi_x \text{ is one-to-one} \}$ is r.e., which is a contradiction since we showed in class that $\bar{A}$ is not r.e.

   While we know testing if a function is one-to-one is not r.e., testing if a function is not one-to-one is easy, because what we must do is to find two values that are mapped to the same result by a given function. Suppose x is a program number. The following program does the trick in a way we have seen many times:

```
Input: program number x

for(t=1; ; t++)      // try all possible steps
    for (u = 0; u ≤ t; u++)      // find the first value
        if STEP(u, x, t) then      // if x running on input u stops in ≤ t steps
            f_u := Φ(u, x);      // then run x on u and save the result
            for (v = 0; v < u; v++)      // find the second value v
                if STEP(v, x, t) then      // if x running on input v stops in ≤ t steps
                    f_v := Φ(v, x);      // then run x on v and save the result
                    if f_u = f_v then
                        return;      // x is not one-to-one
                    end if
                end if
        end if
```

   This program takes program number x and search for u and v such that $\Phi(u, x) = \Phi(v, x)$. If such a pair can be found, program x is not one-to-one and the above program returns. Otherwise, the above program loops. Therefore, $x \in A$ if and only if the above program running on x halts, and A is r.e. You may use the Projection Theorem to prove this fact. ∎