Computational Complexity - Scrimmage II (Solutions)

Y. Wan
LDCSEE,
West Virginia University,
Morgantown, WV
{wan@csee.wvu.edu}

D. Desovski
LDCSEE,
West Virginia University,
Morgantown, WV
{desovski@csee.wvu.edu}

1 Problems

1. Prove that an infinite set is decidable if and only if it can be enumerated in increasing order by a one-to-one computable function.

   Proof:

   (a) Prove : An infinite set S is decidable $\Rightarrow$ S can be enumerated in increasing order by a one-to-one computable function.
   If S is decidable, then the characteristic function is computable :

   $$\begin{aligned} f_s(x) &= 0, x \in S \\ &= 1, x \notin S \end{aligned}$$

   We can define $g(x) = f_s(x) + x$. $g(x)$ is a computable, one-to-one function such that:

   $$\begin{aligned} g(x) &= x, x \in S \\ &= 1 + x, x \notin S \end{aligned}$$

   So, $S = \{g(0), g(1), ...\}$. Clearly, if $a < b$ then, $g(a) < g(b)$. So, we can list out all the g's as $g(a), g(b), g(c), ...$ where $a < b < c < ...$ Therefore, S can be enumerated in increasing order by a one-to-one computable function g.

   (b) Prove: A set S can be enumerated in increasing order by a one-to-one computable function $\Rightarrow$ S is decidable.
   Assume S can be enumerated in increasing order by a one-to-one computable function. Then, there is a function $f : N ... S$ such that:

   $$\begin{aligned} f(a_1) &= b_1 \\ f(a_2) &= b_2 \\ f(a_3) &= b_3 \\ &\dots \\ &\dots \end{aligned}$$

   Then, we can have the following algorithm :

```
Function F(x)
i = 0
while TRUE do
  if (x = a_i) then
    if (f(x) = b_i) then
      RETURN(YES)
    else
      RETURN(NO)
    end if
  else
    if (x < a_i) then
      RETURN(NO)
    else
      i++
    end if
  end if
end while
```

Algorithm 1.1: Algorithm

By using the algorithm above, we can decide if $x \in S$. Therefore, S is decidable.

□

2. Prove : $L_u \leq_m K$.

   Proof: We first need to construct a many-to-one mapping function $f :< e_1, w > \rightarrow e_2$. To do this, we encode the input word w into the finite control of Turing Machine e_1 creating a new Turing Machine e_2, which on every input x, simulates the computation of e_1 on w, ignoring the input x. It is easy to see that if e_1 halts on w (i.e., $< e_1, w > \in L_u$) then e_2 will halt on itself (i.e., $e_2 \in K$) and vice versa. □

3. Show that there must exist a program e such that $W_e = \{e^2\}$.

   Proof: We can define a partial computable function $\psi(e, x)$ as :

$$\begin{aligned} \psi(e,x) &= e^2, \ \textit{for all e and x} \\ &= \uparrow, \ \textit{otherwise} \end{aligned}$$

   Then, $dom \ \ \psi(e,x) = \{e^2\}$, and by the S-M-N theorem, $\phi_{f(e)}(x) = \psi(e,x)$ for every partial computable function. Thus, $dom \ \psi(e,x) = dom \ \phi_{f(e)}(x) = \{e^2\}$. By Corollary 3.6 we have: $W_e = W_{f(e)}$. Therefore, $\{e^2\} = dom \ \ \phi_{f(e)} = W_{f(e)} = W_e$. □

4. Given a collection C of c.e. sets. Is C regular?

   Proof: Let $C = \{all \ \ regular \ \ sets\}$. The index set corresponding to C is:

$$\begin{aligned} P_C &= \{e \mid range(\phi_e) \in C\} \\ &= \{e \mid e \ \ computes \ \ a \ \ regular \ \ set\} \end{aligned}$$

   Clearly, $P_c \neq \emptyset$, and $P_c \neq N$, since $0 \in N$ but $0 \notin P_c$, since $\phi_0 = 0 \Rightarrow range(\phi_0) = 0 \notin C$. So, $P_c \neq \emptyset$ and $P_c \neq N$, and by Rice's Theorem we can conclude that P_c is undecidable. Since P_c is undecidable, C is undecidable. Therefore, we cannot decide if C is regular. □

5. Prove $A \leq_m B \Rightarrow A \leq_T B$.

   Proof: Given $A \leq_m B$, we know that $x \in A \Leftrightarrow f(x) \in B$. Thus, we can define an Oracle Turing Machine M^B to compute the total computable function f as follows: on every input x, if $f(x) \in B$, then M^B accepts, otherwise M^B rejects. Then, since $x \in A \Leftrightarrow f(x) \in B$ on input x, if M^B accepts, we know that $x \in A$, and if M^B rejects, we know that $x \notin A$. So, we can decide set A using M^B. Therefore, $A \leq_T B$. □