Lista 3 de Problemas y Ejercicios
Lógica Matemática

Cristo Daniel Alvarado

24 de noviembre de 2024

3.1. Ejercicios

Ejercicio 3.1.1
Demuestre que todo subconjunto cofinito de $\mathbb{N}$ (es decir, cuyo complemento es finito) es computable.

Demostración:
Sea $X \subseteq \mathbb{N}$ un conjunto cofinito, entonces su complemento $\mathbb{N} \setminus X$ es finito. Sea $N \in \mathbb{N}$, se tiene que la función característica $\chi_{\{N\}}$ es computable, pues tiene como algoritmo:

```
int chi_N(int n){
    if(n == N) return 1;
    else return 0;
}
```

por lo que el conjunto $\{N\}$ es computable, en particular el conjunto:

$$\mathbb{N} \setminus X = \left\{ n \in \mathbb{N} \middle| n \notin X \right\}$$

es computable (por ser finito) ya que es unión finita de conjuntos numerables, luego su complemento el cual es X es computable. ■

Ejercicio 3.1.2
Suponga que $X \subseteq \mathbb{N}$ es computable, y sea $f : \mathbb{N} \to \mathbb{N}$ una función total computable. Demuestre que:

$$f^{-1}[X] = \left\{ n \in \mathbb{N} \middle| f(n) \in X \right\}$$

es un conjunto computable.

Demostración:
Considere el siguiente algoritmo de la función característica de $f^{-1}[X]$:

```
int f_1_[X](int n){
    if(chi_X(f(n))) return 1;
    else return 0;
}
```

como f es total computable, entonces $f(n)$ existe para todo n, luego al ser X un conjunto computable, en una cantidad finita de tiempo se obtiene si χ_X evaluada en $f(n)$ es cero o uno, en cuyo caso se retorna cero o uno en el algoritmo definido anteriormente, el cual siempre retorna algo. ■

Ejercicio 3.1.3
Defina la función $f : \mathbb{N} \to \mathbb{N}$ mediante:

$$f(n) = \begin{cases} 1 & \text{si la expansión decimal de } \pi \text{ contiene una sucesion de al menos } n \text{ digitos} \\ & \text{consecutivos iguales a 7.} \\ 0 & \text{en otro caso.} \end{cases}$$

Demuestre (sin usar ningún hecho especial sobre π) que la función f es total computable.

Demostración:
Es inmediato del siguiente algoritmo:

```
int f(int n){
    recorrer digito por digito la expansion decimal de pi hasta encontrar un 7{
        int cont = 1;
        cont cuenta el numero de 7s despues del primer 7;
        if(n <= cont) return 1;
    }
}
```

que f es computable. Si f no fuese total computable (es decir, que f no sea la función constante uno), entonces existiría al menos un $N \in \mathbb{N}$ tal que $f(N)$ no está bien definido, por la forma en que definimos el algoritmo de f, se tendría que $N+1, \ldots$ tampoco estarían bien definidos. Sea n_0 el mínimo entero no negativo tal que $f(n_0)$ no está bien definido (es decir que el algoritmo anterior sigue funcionando). Construímos el algoritmo:

```
int f_2(int n){
    if(n < n_0) return f(n);
    else return 0;
}
```

esta es el algoritmo de la función f, mismo que es total. ■

Observación 3.1.1
¿Puedo elegir tal n_0 en la demostración anterior?

Ejercicio 3.1.4
Suponga que $g : \mathbb{N} \to \mathbb{N}$ es una función no-creciente, es decir que $g(n+1) \leq g(n)$ para todo $n \in \mathbb{N}$. Pruebe que g debe ser total computable.

Demostración:
Como g es no creciente, g solo puede tomar un número finito de valores en $\mathbb{N}$, digamos $c_k < c_{k-1} < \cdots < c_1 = c_0$, ya que debe quedarse o disminuir a partir de su valor inicial, el cual es $g(0) = c_0$.

Por ser g no decreciente, además debe suceder que $g^{-1}(c_i)$ es un conjunto finito, para todo $i = 0, 1, \ldots, k-1$ y el conjunto $g^{-1}(c_k)$ es infinito. Por tanto, si hacemos:

$$g^{-1}(c_i) = \{x_{i,0}, x_{i,1}, \ldots, x_{i,k_i}\}, \quad \forall i = 0, 1, \ldots, k-1$$

entonces el algoritmo de g sería:

```
int g(int n){
    if(n == x_0_0 || n == x_0_1 || ... || n == x_0_k_0) return c_0;
    if(n == x_1_0 || n == x_1_1 || ... || n == x_1_k_1) return c_1;
    ... ... ;
    if(n == x_(k-1)_0 || n == x_(k-1)_1 || ... || n == x_(k-1)_k_(k-1)) return c_(k-1);
    else return c_k;
}
```

por lo cual, la función g sería total computable. ■

Ejercicio 3.1.5
Demuestre que la función $f:\mathbb{N}^3\to\mathbb{N}$ dada por:

$$f(x,y,z)=\begin{cases} y & \text{si} \quad x=0. \\ z & \text{si} \quad x\neq 0. \end{cases}$$

es total computable.

Demostración:
Veamos que el algoritmo de f sería:

```
int f(int x, int y, int z){
    if(x == 0) return y;
    else return z;
}
```

por lo que, f es total computable. ■

Ejercicio 3.1.6
Considere una retícula de calles que conste de n calles que van de este a oeste, atravesadas por m calles que van de norte a sur, de tal suerte que se genere un mapa rectangular con mn intersecciones. Si un peatón se propone caminar (utilizando dichas calles) para llegar desde la esquina noreste hasta la suroeste, caminando únicamente hacia el este o hacia el sur, y cambiando de dirección únicamente en las esquinas, denote por $r(n,m)$ a la cantidad de posibles rutas que nuestro peatón puede tomar. Demuestre que la función $r:\mathbb{N}^2\to\mathbb{N}$ es total computable.

Demostración:
■

Ejercicio 3.1.7
Proporcione un ejemplo de una función no-total, $g:\mathbb{N}^2\to\mathbb{N}$, tal que la función $h:\mathbb{N}\to\mathbb{N}$ obtenida por medio de una búsqueda no acotada:

$$h(x)=(\mu y)(g(x,y)=0)$$

sí es total.

Solución:
Considere la función no total $g:;\mathbb{N}\to\mathbb{N}$ dada por:

```
int g(int x, int y){
    if(y == 0) return 0;
    else{
        for(int i; 1; i++){}
    }
}
```

la función g sería:

$$g(x,y)=\begin{cases} 0 & \text{si} \quad y=0 \\ \uparrow & \text{e.o.c.} \end{cases}, \quad \forall (x,y)\in\mathbb{N}^2$$

Esta función no es total computable, pero la función h:

$$h(x)=(\mu y)(g(x,y)=0)$$

siempre está definida para todo x, pues el mínimo valor de g ocurre cuando $y = 0$ sin importar la entrada de x, así que μ siempre en la búsqueda no acotada encuentra el valor mínimo, así que,

$$h(x) = 0, \quad \forall x \in \mathbb{N}$$

la cual es total computable. $\square$

Definición 3.1.1
El operador μ significa **el mínimo tal que**, en caso de que exista (y la función queda sin definir en caso de que no). El acto de invocar a μ se conoce como búsqueda no acotada.

Ejercicio 3.1.8
Demuestre que si $X \subseteq \mathbb{N}$ es el conjunto de números de Gödel de máquinas de Turing (es decir, $n \in X$ si y sólo si $\varphi(n, \cdot)$ es una máquina de Turing válida, en donde φ es la máquina de Turing universal), entonces la función característica χ_X es total computable.

Demostración:
Debido a que en el simulador de máquinas de Turing, el mismo puede detectar si una máquina de Turing es válida o no (a partir de sintaxis, estados, movimientos, etc...), llamemos a este algoritmo `verificarMaquinaTuring`. Se tiene pues el algoritmo siguiente:

```
int chi_X(int n){
    convertir el entero n a la maquina de Turing correspondiente y guardarlo en la variable strMaquina;
    return verificarMaquinaTuring(strMaquina);
}
```

ya que la función `verificarMaquinaTuring` siempre evalúa si una máquina de Turing es válida o no, por lo que X es un conjunto total computable. $\blacksquare$

Ejercicio 3.1.9
Haga lo siguiente:

(a) Construya una función h que **eventualmente domine** a todas las funciones computables, es decir, que para toda función computable f exista un $N \in \mathbb{N}$ tal que $f(n) \leq h(n)$ para todo $n \geq N$.

Sugerencia. Hay por lo menos tres maneras naturales de definir a h.

(b) ¿Es posible lograr en el inciso anterior que la función h sea total computable?

Solución:
De *(a)*: Como el conjunto

$$\left\{ f : \mathbb{N} \to \mathbb{N} \middle| f \text{ es computable} \right\}$$

es numerable, podemos hacer una enumeración de todas las funciones, digamos $\left\{ f_i \middle| i \in \mathbb{N} \right\}$. Considere ahora la función $h : \mathbb{N} \to \mathbb{N}$ dada por:

$$h(n) = \max_{0 \leq i \leq n} f_i(n)$$

Claramente la función h es computable, veamos que eventualmente domina a todas las funciones computables. En efecto, si $f:\mathbb{N}\to\mathbb{N}$ es computable, entonces existe $N\in\mathbb{N}$ tal que $f=f_N$. Ahora, para todo $n\geq N$ se cumple que:

$$
\begin{aligned}
h(n) &= \max_{0\leq i\leq n} f_i(n)\\
&\geq f_N(n)\\
&= f(n)
\end{aligned}
$$

De *(b)*: Si h fuese total computable, entonces en particular, $h+1$ sería total computable, en particular computable, luego existiría $N\in\mathbb{N}$ tal que:

$$
h(n)\geq h(n)+1,\quad \forall n\geq N
$$

lo cual no puede suceder, por lo que en algún momento la función h no debe de poder estar definida (para que no haya contradicciones), así que h no puede ser total computable. □

Ejercicio 3.1.10
Dado un algoritmo $\mathcal{A}$ y un $t\in\mathbb{N}\cup\{0\}$, la *foto instantánea de* $\mathcal{A}$ *en el tiempo* t es una compliación de toda la información que se encuentra en el algoritmo en el instante de tiempo t.

Demuestre que, si un algoritmo eventualmente se detiene, entonces todas las fotos instantáneas previas a la foto instantánea terminal (es decir, aquella que corresponde al tiempo en el cual el algoritmo se detiene) deben de ser distintas dos a dos.

Demostración:
Probaremos la contrapositiva. Si hubiera dos fotos iguales sucesivas en el los tiempos t y $t+1$, entonces cuando siguiera corriendo el programa en el tiempo $t+2$, la información pasaría exactamente igual que cuando pasó del tiempo t al tiempo $t+1$, es decir que sería la información del tiempo t.

Por inducción se puede probar rápidamente que para todo tiempo $t+n$ con $n\in\mathbb{N}$, la información del algoritmo es la misma que en el tiempo t, por lo que el algoritmo nunca se detiene (a menos que se haya detenido en el tiempo t). ■

Ejemplo 3.1.1
En el ejercicio anterior, un ejemplo sería si concebimos a nuestro algoritmo como una máquina de Turing, entonces la foto instantánea en el tiempo t es un conjunto que contiene al t-ésimo estado visitado, así como el contenido de la cinta (visto como una sucesión finita de símbolos del alfabeto correspondiente junto con el símbolo *en blanco*) justo en el momento en que se visita ese t-ésimo estado, así como la posición exacta del cabezal lector/escritor en ese momento.

Por otra parte, si concebimos a nuestro algoritmo como un programa en algún lenguaje de programación, entonces la foto instantánea en el tiempo t consta de la información acerca de los valores que tienen las variables en el momento de correr la t-ésima instrucción, así como el reglón del programa que se está corriendo en ese momento.

Ejercicio 3.1.11
Recuerde que un conjunto no vacío $X\subseteq\mathbb{N}$ es **computablemente enumerable** si y sólo si X es el rango de alguna función total computable. Demuestre ahora que un conjunto no vacío $X\subseteq\mathbb{N}$ es computable si y sólo si $X=\operatorname{ran}(f)$ para alguna función $f:\mathbb{N}\to X$ total computable que es *no decreciente*.

Demostración:

■

Ejercicio 3.1.12
Sea $A \subseteq \mathbb{N}$ un conjunto infinito, computablemente enumerable.

(a) Demuestre que existe una función total computable $g : \mathbb{N} \to \mathbb{N}$ que es estrictamente creciente tal que $\mathrm{ran}(g) \subseteq A$.

(b) Concluya que todo conjunto computablemente enumerable infinito contiene un subconjunto computable infinito.

Demostración:

$\blacksquare$

Ejercicio 3.1.13
Decimos que un conjunto $X \subseteq \mathbb{N}^k$ es Σ_1^0 si existe algún conjunto computable $A \subseteq \mathbb{N}^m$ tal que:

$$X = \left\{(x_1, ..., x_k)\middle|\exists a_1, ..., a_{m-k} \text{ tales que } (x_1, ..., x_k, a_1, ..., a_{m-k}) \in A\right\}$$

Por otra parte, decimos que un conjunto $Y \subseteq \mathbb{N}^k$ es Π_1^0 si existe algún conjunto computable $B \subseteq \mathbb{N}^m$ tal que:

$$Y = \left\{(y_1, ..., y_k)\middle|\forall b_1, ..., b_{m-k} \text{ tales que } (y_1, ..., y_k, b_1, ..., b_{m-k}) \in B\right\}$$

Demuestre que todo conjunto que es al mismo tiempo Σ_1^0 y Π_1^0 es computable.

Demostración:

$\blacksquare$

Ejercicio 3.1.14
Dado un conjunto $X \subseteq \mathbb{N}^k$, demuestre que las siguientes condiciones son equivalentes:

(a) X es computablement enumerable.

(b) O bien $X = \emptyset$, o bien X es el rango de alguna función parcial computable.

(c) Existe una sucesión computable de conjuntos finitos $Y_s \subseteq \mathbb{N}^k$ (lo cual realmente significa: existe un conjunto computable $Y \subseteq \mathbb{N}^{k+1}$ tal que, para cada $s \in \mathbb{N}$

$$Y_s = \left\{(x_1, ..., x_k)\middle|(x_1, ..., x_k, s) \in Y\right\}$$

) que satisfacen:

$$Y_s \subseteq Y_{s+1}, \quad \forall s \in \mathbb{N}$$

de tal suerte que $X = \bigcup_{s\in\mathbb{N}} Y_s$

Demostración:

$\blacksquare$

Ejercicio 3.1.15
Demuestre que todo conjunto infinito computablemente enumerable es el rango de alguna función total computable $f : \mathbb{N} \to \mathbb{N}$ que es inyectiva.

Demostración:

■

Ejercicio 3.1.16
Demuestre que si $X \subseteq \mathbb{N}$ es computable y $Y \subseteq \mathbb{N}$ es computable enumerable, entonces Y es computable si y sólo si $X \setminus Y$ es computable.

Demostración:

■

Ejercicio 3.1.17
Demuestre la *propiedad de reducción*: para cualesquiera dos conjuntos computablemente enumerables X y Y existen conjuntos computablemente enumerables $A \subseteq X$ y $B \subseteq Y$ tales que $A \cap B = \emptyset$ y $A \cup B = X \cup Y$.

Demostración:

■

Ejercicio 3.1.18
Demuestre que todo conjunto infinito $X \subseteq \mathbb{N}$ contiene un subconjunto que no es computable.

Demostración:

■

Ejercicio 3.1.19
Dados dos conjuntos $X, Y \subseteq \mathbb{N}$, definimos la **yunta** de X y Y como:

$$X \oplus Y = \left\{2n \middle| n \in X\right\} \cup \left\{2n+1 \middle| n \in Y\right\}$$

(a) Demuestre que, si X y Y son ambos computables, entonces $X \oplus Y$ también lo es.

(b) Demuestre que, si X y Y son ambos computablemente enumerables, entonces $X \oplus Y$ también lo es.

(c) Demuestre que, si $X \oplus Y$ es computable, entonces tanto X como Y también son computables.

(d) Demuestre que, si $X \oplus Y$ es computablemente enumerable, entonces tanto X como Y también son computablemente enumerables.

Demostración:

■

Observación 3.1.2
En términos de grados de Turing, la yunta es importante porque representa al supremo de los grados de Turing de los conjuntos correspondientes.

Ejercicio 3.1.20
Considere una codificación de los algoritmos de enumeración por medio de números naturales, y denote al conjunto impreso por el n-ésimo algoritmo como W_n (en otras palabras, W_n representa al n-ésimo conjunto computablemente enumerable, de acuerdo con alguna enumeración efectiva).

(a) Demuestre que, si $f : \mathbb{N} \to \mathbb{N}$ es una función total computable, entonces

$$\bigcup_{n=1}^{\infty} W_{f(n)}$$

es computablemente enumerable (en otras palabras, las uniones computables de conjuntos computablement enumerables son computablemente enumerables).

(b) ¿Es posible afirmar algo acerca de la computabilidad/computable enumerabilidad del conjunto $\bigcap_{n=1}^{\infty} W_{f(n)}$?

(c) Demuestre que existe una función parcial computable $f :; \mathbb{N} \to \mathbb{N}$ tal que $f(n)$ está definida siempre que $W_n \neq \emptyset$, en cuyo caso $f(n) \in W_n$ (en otras palabras, se cumple el **axioma de elección computable**).

(d) Dada una relación binaria $R \subseteq \mathbb{N}^2$ computablemente enumerable, exhiba una función parcial computable $f :; \mathbb{N} \to \mathbb{N}$ tal que $f(n)$ esté definida siempre que $(n, m) \in R$ para algun $m \in \mathbb{N}$, en cuyo caso, además debe cumplirse que $(n, f(n)) \in R$.

Demostración:

■

Ejercicio 3.1.21
Definamos los siguientes subconjuntos de $\mathbb{N}$:

$$X = \left\{ n \in \mathbb{N} \middle| \varphi(n, n) = 0 \right\},$$
$$Y = \left\{ n \in \mathbb{N} \middle| \varphi(n, n) = 1 \right\},$$

(donde φ reprsenta la máquiana de Turing universal).

(a) Demuestre que X y Y son conjuntos computablemente enumerables, amén de que $X \cap Y = \emptyset$.

(b) Demuestre que X y Y son **computablemente inseprables**: en otras palabras, no existe ningún conjunto computable Z tal que $X \subseteq Z$ y $Y \cap Z = \emptyset$.

Sugerencia. Diagonalización, es decir, si existiera tal Z, su función característica sería $\varphi(d, \cdot)$ para algún $d \in \mathbb{N}$; luego, ¿qué podemos decir de $\varphi(d, d)$?

Demostración:

■