8 Funciones recursivas generales III

José de Jesús Lavalle Martínez

Benemérita Universidad Autónoma de Puebla
Facultad de Ciencias de la Computación
Computabilidad CCOS 257

Contenido

Motivación

El operador de búsqueda (a menudo llamado minimización u operador μ) proporciona una forma útil de definir una función en términos de una "búsqueda" para la primera vez que se cumple una condición dada.

Búsqueda acotada I

Definición 1

Para una relación P de aridad $(k+1)$, el número $(\mu t < y)P(\vec{x}, t)$ se define mediante la ecuación:

$$(\mu t < y)P(\vec{x}, t) = \begin{cases} \text{el mínimo } t \text{ tal que } t < y \text{ y } P(\vec{x}, t), \text{ si existe} \\ y \text{ si no existe tal } t \end{cases}$$

Por ejemplo, si definimos

$$f(x, y) = \mu t < y[t \text{ es primo y } x < t]$$

entonces $f(6, 4) = 4$ y $f(6, 8) = f(6, 800) = 7$.

Búsqueda acotada II

Teorema 1

Si P es una relación primitiva recursiva, entonces la función

$$f(\vec{x}, y) = (\mu t < y) P(\vec{x}, t)$$

es una función recursiva primitiva.

Búsqueda acotada III

Prueba: Aplicaremos la recursividad primitiva. Trivialmente $f(\vec{x}, 0) = 0$, por lo que aquí no hay problema. El problema es ver cómo $f(\vec{x}, y+1)$ (llámelo b) depende de $f(\vec{x}, y)$ (llámelo a):

- Si $a < y$, entonces $b = a$. (La búsqueda para números menores que y tuvo éxito).

Búsqueda acotada III

Prueba: Aplicaremos la recursividad primitiva. Trivialmente $f(\vec{x}, 0) = 0$, por lo que aquí no hay problema. El problema es ver cómo $f(\vec{x}, y+1)$ (llámelo b) depende de $f(\vec{x}, y)$ (llámelo a):

- Si $a < y$, entonces $b = a$. (La búsqueda para números menores que y tuvo éxito).
- Si $a = y$ y $P(\vec{x}, y)$, entonces $b = y$.

Búsqueda acotada III

Prueba: Aplicaremos la recursividad primitiva. Trivialmente $f(\vec{x}, 0) = 0$, por lo que aquí no hay problema. El problema es ver cómo $f(\vec{x}, y+1)$ (llámelo b) depende de $f(\vec{x}, y)$ (llámelo a):

- Si $a < y$, entonces $b = a$. (La búsqueda para números menores que y tuvo éxito).
- Si $a = y$ y $P(\vec{x}, y)$, entonces $b = y$.
- En caso contrario, $b = y + 1$.

Búsqueda acotada IV

- Así, si definimos

$$g(a, \vec{x}, y) = \begin{cases} a & \text{si } a < y \\ y & \text{si } a \not< y \text{ y } P(\vec{x}, y) \\ y + 1 & \text{si } a \not< y \text{ y no } P(\vec{x}, y), \end{cases}$$

entonces f se obtiene por recursión primitiva de las funciones $\vec{x} \mapsto 0$ y g.

Búsqueda acotada IV

- Así, si definimos

$$g(a, \vec{x}, y) = \begin{cases} a & \text{si } a < y \\ y & \text{si } a \not< y \text{ y } P(\vec{x}, y) \\ y + 1 & \text{si } a \not< y \text{ y no } P(\vec{x}, y), \end{cases}$$

  entonces f se obtiene por recursión primitiva de las funciones $\vec{x} \mapsto 0$ y g.
- Como P es una relación recursiva primitiva, se deduce que g es recursiva primitiva (por definición, por casos) y, por tanto, f es recursiva primitiva. $\dashv$

Búsqueda acotada IV

- Así, si definimos

$$g(a, \vec{x}, y) = \begin{cases} a & \text{si } a < y \\ y & \text{si } a \nless y \text{ y } P(\vec{x}, y) \\ y+1 & \text{si } a \nless y \text{ y no } P(\vec{x}, y), \end{cases}$$

  entonces f se obtiene por recursión primitiva de las funciones $\vec{x} \mapsto 0$ y g.
- Como P es una relación recursiva primitiva, se deduce que g es recursiva primitiva (por definición, por casos) y, por tanto, f es recursiva primitiva. $\dashv$
- Hay otra prueba de este teorema, que se basa en la siguiente notable ecuación:

$$(\mu t < y) P(\vec{x}, t) = \sum_{u<y} \prod_{t \leqslant u} C_{\overline{P}}(\vec{x}, t)$$

Búsqueda acotada V

- Un operador de búsqueda relacionado es una *maximización* acotada.

Búsqueda acotada V

- Un operador de búsqueda relacionado es una *maximización* acotada.
- Defina el operador $\overline{\mu}$ de la siguiente manera:

$$(\overline{\mu}t < y)P(\vec{x}, t) = \begin{cases} \text{el número } t \text{ más grande tal que } t \leqslant y \text{ y } P(\vec{x}, t), \text{ si lo hay} \\ 0 \quad \text{si no existe tal } t \end{cases}$$

Búsqueda acotada V

- Un operador de búsqueda relacionado es una *maximización* acotada.
- Defina el operador $\overline{\mu}$ de la siguiente manera:

$$(\overline{\mu}t < y)P(\vec{x}, t) = \begin{cases} \text{el número } t \text{ más grande tal que } t \leqslant y \text{ y } P(\vec{x}, t), \text{ si lo hay} \\ 0 \quad \text{si no existe tal } t \end{cases}$$

Búsqueda acotada V

- Un operador de búsqueda relacionado es una *maximización* acotada.
- Defina el operador $\overline{\mu}$ de la siguiente manera:

$$(\overline{\mu}t < y)P(\vec{x}, t) = \begin{cases} \text{el número } t \text{ más grande tal que } t \leqslant y \text{ y } P(\vec{x}, t), \text{ si lo hay} \\ 0 \quad \text{si no existe tal } t \end{cases}$$

Teorema 2

Si P es una relación recursiva primitiva, entonces la función

$$f(\vec{x}, y) = (\overline{\mu}t < y)P(\vec{x}, t)$$

es una función recursiva primitiva. *Prueba:*

$$(\overline{\mu}t < y)P(\vec{x}, t) = y \dot{-} (\mu s < y)P(\vec{x}, y \dot{-} s).$$

Esta ecuación capta la idea de buscar hacia abajo a partir de y. $\dashv$

-

Búsqueda acotada VI

- Euclides observó que el conjunto de los números primos es ilimitado.

Búsqueda acotada VI

- Euclides observó que el conjunto de los números primos es ilimitado.
- Por tanto, la función

$$h(x) = \text{el menor número primo mayor que } x$$

es total.

Búsqueda acotada VI

- Euclides observó que el conjunto de los números primos es ilimitado.
- Por tanto, la función

$$h(x) = \text{el menor número primo mayor que } x$$

es total.

- También es primitiva recursiva porque

$$h(x) = \mu t < (x! + 2)[t \text{ es primo y } x < t].$$

Búsqueda acotada VI

- Euclides observó que el conjunto de los números primos es ilimitado.
- Por tanto, la función

$$h(x) = \text{el menor número primo mayor que } x$$

es total.

- También es primitiva recursiva porque

$$h(x) = \mu t < (x! + 2)[t \text{ es primo y } x < t].$$

- El límite superior $x! + 2$ es suficiente porque para cualquier factor primo p de $x! + 1$, tenemos $x < p \leqslant x! + 1$.

Búsqueda acotada VI

- Euclides observó que el conjunto de los números primos es ilimitado.
- Por tanto, la función

$$h(x) = \text{el menor número primo mayor que } x$$

es total.

- También es primitiva recursiva porque

$$h(x) = \mu t < (x! + 2)[t \text{ es primo y } x < t].$$

- El límite superior $x! + 2$ es suficiente porque para cualquier factor primo p de $x! + 1$, tenemos $x < p \leqslant x! + 1$.
- Así que cualquier búsqueda de un número primo mayor que x no necesita ir más allá de $x! + 1$.

Búsqueda acotada VII (Digresión)

- Aquí hay un resultado interesante en la teoría de números.

Búsqueda acotada VII (Digresión)

- Aquí hay un resultado interesante en la teoría de números.
- El "postulado de Bertrand" establece que para cualquier $x > 3$, siempre habrá un número primo p con $x < p < 2x - 2$.

Búsqueda acotada VII (Digresión)

- Aquí hay un resultado interesante en la teoría de números.
- El "postulado de Bertrand" establece que para cualquier $x > 3$, siempre habrá un número primo p con $x < p < 2x - 2$.
- El postulado de Bertrand implica que en el párrafo anterior basta con usar simplemente $h(x) = \mu t < (2x - 2)[t \text{ es primo y } x < t]$.

Búsqueda acotada VII (Digresión)

- Aquí hay un resultado interesante en la teoría de números.
- El "postulado de Bertrand" establece que para cualquier $x > 3$, siempre habrá un número primo p con $x < p < 2x - 2$.
- El postulado de Bertrand implica que en el párrafo anterior basta con usar simplemente $h(x) = \mu t < (2x - 2)[t$ es primo y $x < t]$.
- En 1845, el matemático francés Joseph Bertrand, usando tablas de números primos, verificó esta afirmación para x por debajo de tres millones.

Búsqueda acotada VII (Digresión)

- Aquí hay un resultado interesante en la teoría de números.
- El "postulado de Bertrand" establece que para cualquier $x > 3$, siempre habrá un número primo p con $x < p < 2x - 2$.
- El postulado de Bertrand implica que en el párrafo anterior basta con usar simplemente $h(x) = \mu t < (2x - 2)[t$ es primo y $x < t]$.
- En 1845, el matemático francés Joseph Bertrand, usando tablas de números primos, verificó esta afirmación para x por debajo de tres millones.
- Luego, en 1850, el ruso P.L. Chebyshev (Tchebychef) demostró el resultado en general.

Búsqueda acotada VII (Digresión)

- Aquí hay un resultado interesante en la teoría de números.
- El "postulado de Bertrand" establece que para cualquier $x > 3$, siempre habrá un número primo p con $x < p < 2x - 2$.
- El postulado de Bertrand implica que en el párrafo anterior basta con usar simplemente $h(x) = \mu t < (2x - 2)[t$ es primo y $x < t]$.
- En 1845, el matemático francés Joseph Bertrand, usando tablas de números primos, verificó esta afirmación para x por debajo de tres millones.
- Luego, en 1850, el ruso P.L. Chebyshev (Tchebychef) demostró el resultado en general.
- En 1932, el húngaro Paul Erdős dio una prueba mejor, que ahora se puede encontrar en los libros de texto universitarios de teoría de números.

Búsqueda acotada VII (Digresión)

- Aquí hay un resultado interesante en la teoría de números.
- El "postulado de Bertrand" establece que para cualquier $x > 3$, siempre habrá un número primo p con $x < p < 2x - 2$.
- El postulado de Bertrand implica que en el párrafo anterior basta con usar simplemente $h(x) = \mu t < (2x - 2)[t$ es primo y $x < t]$.
- En 1845, el matemático francés Joseph Bertrand, usando tablas de números primos, verificó esta afirmación para x por debajo de tres millones.
- Luego, en 1850, el ruso P.L. Chebyshev (Tchebychef) demostró el resultado en general.
- En 1932, el húngaro Paul Erdős dio una prueba mejor, que ahora se puede encontrar en los libros de texto universitarios de teoría de números.
- El origen de

Chebyshev lo dijo
Así que lo diré de nuevo
Siempre hay un primo
Entre N and $2N$

se desconoce, lo cual puede ser mejor.

Ejemplo 18 I

- Defina p_x como el $(x+1)$ primer número primo, de modo que

$$p_0 = 2, p_1 = 3, p_2 = 5, p_3 = 7, p_4 = 11,$$

etcétera.

Ejemplo 18 I

- Defina p_x como el $(x+1)$ primer número primo, de modo que

$$p_0 = 2, p_1 = 3, p_2 = 5, p_3 = 7, p_4 = 11,$$

  etcétera.
- En otras palabras, p_x es el x-ésimo primo impar, excepto que $p_0 = 2$.

Ejemplo 18 I

- Defina p_x como el $(x+1)$ primer número primo, de modo que

$$p_0 = 2, p_1 = 3, p_2 = 5, p_3 = 7, p_4 = 11,$$

  etcétera.
- En otras palabras, p_x es el x-ésimo primo impar, excepto que $p_0 = 2$.
- El teorema de los números primos nos dice que p_x crece a una velocidad similar a $x \ln x$, pero eso no viene al caso.

Ejemplo 18 I

- Defina p_x como el $(x+1)$ primer número primo, de modo que

$$p_0 = 2, p_1 = 3, p_2 = 5, p_3 = 7, p_4 = 11,$$

  etcétera.
- En otras palabras, p_x es el x-ésimo primo impar, excepto que $p_0 = 2$.
- El teorema de los números primos nos dice que p_x crece a una velocidad similar a $x \ln x$, pero eso no viene al caso.
- La función $x \mapsto p_x$ es recursiva primitiva porque tenemos las ecuaciones de recursión

$$\begin{aligned} p_0 &= 2 \\ p_{x+1} &= h(p_x), \end{aligned}$$

  donde h es la función anterior que encuentra el siguiente primo.

Ejemplo 18 I

- Defina p_x como el $(x+1)$ primer número primo, de modo que

$$p_0 = 2, p_1 = 3, p_2 = 5, p_3 = 7, p_4 = 11,$$

  etcétera.
- En otras palabras, p_x es el x-ésimo primo impar, excepto que $p_0 = 2$.
- El teorema de los números primos nos dice que p_x crece a una velocidad similar a $x \ln x$, pero eso no viene al caso.
- La función $x \mapsto p_x$ es recursiva primitiva porque tenemos las ecuaciones de recursión

$$\begin{aligned} p_0 &= 2 \\ p_{x+1} &= h(p_x), \end{aligned}$$

  donde h es la función anterior que encuentra el siguiente primo.
- Es fácil ver que siempre tenemos $x + 1 < p_x$; una prueba formal puede utilizar la inducción.

Ejemplo 18 II

- Necesitaremos métodos para codificar una cadena de números mediante un solo número.

Ejemplo 18 II

- Necesitaremos métodos para codificar una cadena de números mediante un solo número.
- Un método que es conceptualmente simple utiliza potencias de números primos.

Ejemplo 18 II

- Necesitaremos métodos para codificar una cadena de números mediante un solo número.
- Un método que es conceptualmente simple utiliza potencias de números primos.
- Definimos la "notación entre corchetes" de la siguiente manera.

$$
\begin{aligned}
[] &= 1 \\
[x] &= 2^{x+1} \\
[x, y] &= 2^{x+1} 3^{y+1} \\
[x, y, z] &= 2^{x+1} 3^{y+1} 5^{z+1} \\
&\dots \\
[x_0, x_1, \dots, x_k] &= 2^{x_0+1} 3^{x_1+1} \cdots p_k^{x_k+1}.
\end{aligned}
$$

Ejemplo 18 III

- Por ejemplo, $[2, 1] = 72$ y $[2, 1, 0] = 360$.

Ejemplo 18 III

- Por ejemplo, $[2,1] = 72$ y $[2,1,0] = 360$.
- Claramente, para cualquier valor de k, la función

$$< x_0, x_1, \ldots, x_k > \mapsto [x_0, x_1, \ldots, x_k]$$

es una función recursiva primitiva de aridad $(k+1)$.

Ejemplo 18 III

- Por ejemplo, $[2,1] = 72$ y $[2,1,0] = 360$.
- Claramente, para cualquier valor de k, la función

$$< x_0, x_1, \ldots, x_k > \mapsto [x_0, x_1, \ldots, x_k]$$

  es una función recursiva primitiva de aridad $(k+1)$.
- Pero codificar es inútil, a menos que podamos decodificarlo.

Ejemplo 18 III

- Por ejemplo, $[2,1] = 72$ y $[2,1,0] = 360$.
- Claramente, para cualquier valor de k, la función

$$< x_0, x_1, \ldots, x_k > \mapsto [x_0, x_1, \ldots, x_k]$$

  es una función recursiva primitiva de aridad $(k+1)$.
- Pero codificar es inútil, a menos que podamos decodificarlo.
- El "teorema fundamental de la aritmética" es la afirmación de que todo número entero positivo tiene una factorización en números primos, única excepto por el orden.

Ejemplo 18 III

- Por ejemplo, $[2,1] = 72$ y $[2,1,0] = 360$.
- Claramente, para cualquier valor de k, la función

$$< x_0, x_1, \ldots, x_k > \mapsto [x_0, x_1, \ldots, x_k]$$

  es una función recursiva primitiva de aridad $(k+1)$.
- Pero codificar es inútil, a menos que podamos decodificarlo.
- El "teorema fundamental de la aritmética" es la afirmación de que todo número entero positivo tiene una factorización en números primos, única excepto por el orden.
- Para decodificar, estamos explotando implícitamente la unicidad de la factorización de números primos.

Ejemplo 18 III

- Por ejemplo, $[2, 1] = 72$ y $[2, 1, 0] = 360$.
- Claramente, para cualquier valor de k, la función

$$< x_0, x_1, \ldots, x_k > \mapsto [x_0, x_1, \ldots, x_k]$$

  es una función recursiva primitiva de aridad $(k + 1)$.
- Pero codificar es inútil, a menos que podamos decodificarlo.
- El "teorema fundamental de la aritmética" es la afirmación de que todo número entero positivo tiene una factorización en números primos, única excepto por el orden.
- Para decodificar, estamos explotando implícitamente la unicidad de la factorización de números primos.
- El Ejemplo 19 dará una función de decodificación recursiva primitiva.

Ejemplo 18 IV (Digresión I)

- Usar potencias de números primos no es de ninguna manera la única forma de codificar una cadena de números.

Ejemplo 18 IV (Digresión I)

- Usar potencias de números primos no es de ninguna manera la única forma de codificar una cadena de números.
- Es un método muy conveniente para nuestros propósitos actuales, pero existen otros métodos.

Ejemplo 18 IV (Digresión I)

- Usar potencias de números primos no es de ninguna manera la única forma de codificar una cadena de números.
- Es un método muy conveniente para nuestros propósitos actuales, pero existen otros métodos.
- Aquí hay un enfoque muy diferente:

$$< x_0, x_1, \ldots, x_k > \mapsto 1\underbrace{00\cdots0}_{x_0}1\underbrace{00\cdots0}_{x_1}1\cdots\cdots1\underbrace{00\cdots0}_{x_k}$$

Ejemplo 18 IV (Digresión I)

- Usar potencias de números primos no es de ninguna manera la única forma de codificar una cadena de números.
- Es un método muy conveniente para nuestros propósitos actuales, pero existen otros métodos.
- Aquí hay un enfoque muy diferente:

$$< x_0, x_1, \ldots, x_k > \mapsto 1\underbrace{00\cdots0}_{x_0}1\underbrace{00\cdots0}_{x_1}1\cdots\cdots1\underbrace{00\cdots0}_{x_k}$$

- Es decir, una secuencia de longitud n puede codificarse mediante el número cuya representación binaria tiene n unos.

Ejemplo 18 IV (Digresión I)

- Usar potencias de números primos no es de ninguna manera la única forma de codificar una cadena de números.
- Es un método muy conveniente para nuestros propósitos actuales, pero existen otros métodos.
- Aquí hay un enfoque muy diferente:

$$< x_0, x_1, \ldots, x_k > \mapsto 1\underbrace{00\cdots0}_{x_0}1\underbrace{00\cdots0}_{x_1}1\cdots\cdots1\underbrace{00\cdots0}_{x_k}$$

- Es decir, una secuencia de longitud n puede codificarse mediante el número cuya representación binaria tiene n unos.
- El número de ceros que siguen al i-ésimo 1 en la representación corresponde al i-ésimo componente de la secuencia.

Ejemplo 18 IV (Digresión I)

- Usar potencias de números primos no es de ninguna manera la única forma de codificar una cadena de números.
- Es un método muy conveniente para nuestros propósitos actuales, pero existen otros métodos.
- Aquí hay un enfoque muy diferente:

$$< x_0, x_1, \ldots, x_k > \mapsto 1\underbrace{00\cdots0}_{x_0}1\underbrace{00\cdots0}_{x_1}1\cdots\cdots1\underbrace{00\cdots0}_{x_k}$$

- Es decir, una secuencia de longitud n puede codificarse mediante el número cuya representación binaria tiene n unos.
- El número de ceros que siguen al i-ésimo 1 en la representación corresponde al i-ésimo componente de la secuencia.
- Aquí hay unos ejemplos:

$$< 0, 3, 2 > \mapsto 11000100 = 196$$
$$< 2, 1, 0 > \mapsto 100101 = 37$$
$$<> \mapsto 0 = 0$$
$$< 7 > \mapsto 10000000 = 128$$
$$< 0, 0, 0, 0 > \mapsto 1111 = 15$$

Ejemplo 18 IV (Digresión II)

- En particular, supongamos que queremos codificar una secuencia de dos números.

Ejemplo 18 IV (Digresión II)

- En particular, supongamos que queremos codificar una secuencia de dos números.
- Este método produce

$$< m, n > \mapsto 1\underbrace{00\cdots0}_{m}1\underbrace{00\cdots0}_{n} = 2^{m+n+1} + 2^n = 2^n(2^{m+1} + 1).$$

Ejemplo 18 IV (Digresión II)

- En particular, supongamos que queremos codificar una secuencia de dos números.
- Este método produce

$$< m, n > \mapsto 1\underbrace{00\cdots0}_{m}1\underbrace{00\cdots0}_{n} = 2^{m+n+1} + 2^n = 2^n(2^{m+1} + 1).$$

- La notación entre corchetes produce simplemente $[m, n] = 2^{m+1} \cdot 3^{n+1}$.

Ejemplo 18 IV (Digresión II)

- En particular, supongamos que queremos codificar una secuencia de dos números.
- Este método produce

$$< m, n > \mapsto 1\underbrace{00\cdots0}_{m}1\underbrace{00\cdots0}_{n} = 2^{m+n+1} + 2^n = 2^n(2^{m+1} + 1).$$

- La notación entre corchetes produce simplemente $[m, n] = 2^{m+1} \cdot 3^{n+1}$.
- Ambas "funciones de emparejamiento" tienen la característica de que crecen exponencialmente a medida que m y n aumentan.

Ejemplo 18 IV (Digresión III)

- Curiosamente, existen funciones de emparejamiento *polinomial*, y aquí hay una:

$$J(m,n) = \frac{1}{2}((m+n)^2 + 3m + n).$$

Ejemplo 18 IV (Digresión III)

- Curiosamente, existen funciones de emparejamiento *polinomial*, y aquí hay una:

$$J(m,n) = \frac{1}{2}((m+n)^2 + 3m + n).$$

- La función J es uno a uno, por lo que el par $< m, n >$ es recuperable a partir del valor $J(m,n)$.

Ejemplo 18 IV (Digresión III)

- Curiosamente, existen funciones de emparejamiento *polinomial*, y aquí hay una:

$$J(m,n) = \frac{1}{2}((m+n)^2 + 3m + n).$$

- La función J es uno a uno, por lo que el par $< m, n >$ es recuperable a partir del valor $J(m,n)$.
- De hecho, la función J asigna $\mathbb{N} \times \mathbb{N}$ uno a uno a $\mathbb{N}$.

Ejemplo 18 IV (Digresión III)

- Curiosamente, existen funciones de emparejamiento *polinomial*, y aquí hay una:

$$J(m,n) = \frac{1}{2}((m+n)^2 + 3m + n).$$

- La función J es uno a uno, por lo que el par $< m, n >$ es recuperable a partir del valor $J(m,n)$.
- De hecho, la función J asigna $\mathbb{N} \times \mathbb{N}$ uno a uno a $\mathbb{N}$.
- ¿Y de dónde viene J?

Ejemplo 18 IV (Digresión III)

- Curiosamente, existen funciones de emparejamiento *polinomial*, y aquí hay una:

$$J(m,n) = \frac{1}{2}((m+n)^2 + 3m + n).$$

- La función J es uno a uno, por lo que el par $< m, n >$ es recuperable a partir del valor $J(m,n)$.
- De hecho, la función J asigna $\mathbb{N} \times \mathbb{N}$ uno a uno a $\mathbb{N}$.
- ¿Y de dónde viene J?
- Aquí hay una pista.

Ejemplo 18 IV (Digresión III)

- Curiosamente, existen funciones de emparejamiento *polinomial*, y aquí hay una:

$$J(m,n) = \frac{1}{2}((m+n)^2 + 3m + n).$$

- La función J es uno a uno, por lo que el par $< m, n >$ es recuperable a partir del valor $J(m,n)$.
- De hecho, la función J asigna $\mathbb{N} \times \mathbb{N}$ uno a uno a $\mathbb{N}$.
- ¿Y de dónde viene J?
- Aquí hay una pista.
- Calcule $J(m,n)$ para todos los valores pequeños de m y n, digamos $m + n \leqslant 4$.

Ejemplo 18 IV (Digresión III)

- Curiosamente, existen funciones de emparejamiento *polinomial*, y aquí hay una:

$$J(m,n) = \frac{1}{2}((m+n)^2 + 3m + n).$$

- La función J es uno a uno, por lo que el par $< m, n >$ es recuperable a partir del valor $J(m,n)$.
- De hecho, la función J asigna $\mathbb{N} \times \mathbb{N}$ uno a uno a $\mathbb{N}$.
- ¿Y de dónde viene J?
- Aquí hay una pista.
- Calcule $J(m,n)$ para todos los valores pequeños de m y n, digamos $m + n \leqslant 4$.
- Luego haga una gráfica en el plano, colocando el número $J(m,n)$ en el punto del plano con coordenadas $< m, n >$.

Ejemplo 18 IV (Digresión III)

- Curiosamente, existen funciones de emparejamiento *polinomial*, y aquí hay una:

$$J(m,n) = \frac{1}{2}((m+n)^2 + 3m + n).$$

- La función J es uno a uno, por lo que el par $< m, n >$ es recuperable a partir del valor $J(m,n)$.
- De hecho, la función J asigna $\mathbb{N} \times \mathbb{N}$ uno a uno a $\mathbb{N}$.
- ¿Y de dónde viene J?
- Aquí hay una pista.
- Calcule $J(m,n)$ para todos los valores pequeños de m y n, digamos $m + n \leqslant 4$.
- Luego haga una gráfica en el plano, colocando el número $J(m,n)$ en el punto del plano con coordenadas $< m, n >$.
- Compruebe si está surgiendo un patrón.

Ejemplo 19 I

- Existe una función primitiva recursiva de "decodificación" de aridad dos, cuyo valor en $< x, y >$ se escribe $(x)_y$, con la propiedad de que siempre que $y \leqslant k$,

$$([x_0, x_1, \ldots, x_k])_y = x_y.$$

Ejemplo 19 I

- Existe una función primitiva recursiva de "decodificación" de aridad dos, cuyo valor en $< x, y >$ se escribe $(x)_y$, con la propiedad de que siempre que $y \leqslant k$,

$$([x_0, x_1, \ldots, x_k])_y = x_y.$$

- Esto es

$$(\text{código para una secuencia})_y = \text{el } (y+1)\text{-ésimo término de la secuencia.}$$

Ejemplo 19 I

- Existe una función primitiva recursiva de "decodificación" de aridad dos, cuyo valor en $< x, y >$ se escribe $(x)_y$, con la propiedad de que siempre que $y \leqslant k$,

$$([x_0, x_1, \ldots, x_k])_y = x_y.$$

- Esto es

$$(\text{código para una secuencia})_y = \text{el } (y+1)\text{-ésimo término de la secuencia.}$$

- Por ejemplo, $(72)_0 = 2$ y $(72)_1 = 1$ porque $72 = [2, 1]$.

Ejemplo 19 II

- Primero, observe que el exponente de un primo q en la factorización de un entero positivo x es

$$\mu e \left(q^{e+1} \nmid x \right),$$

la e más pequeña para la cual $e + 1$ sería demasiado.

Ejemplo 19 II

- Primero, observe que el exponente de un primo q en la factorización de un entero positivo x es

$$\mu e \left(q^{e+1} \nmid x\right),$$

  la e más pequeña para la cual $e+1$ sería demasiado.
- Podemos acotar la búsqueda en x porque si $q^e \mid x$, entonces $e < q^e \leqslant x$.

Ejemplo 19 II

- Primero, observe que el exponente de un primo q en la factorización de un entero positivo x es

$$\mu e \left(q^{e+1} \nmid x\right),$$

  la e más pequeña para la cual $e+1$ sería demasiado.
- Podemos acotar la búsqueda en x porque si $q^e \mid x$, entonces $e < q^e \leqslant x$.
- Es decir, el exponente de q en la factorización de x es

$$\left(\mu e < x\right) \left(q^{e+1} \nmid x\right).$$

Ejemplo 19 II

- Primero, observe que el exponente de un primo q en la factorización de un entero positivo x es

$$\mu e \left(q^{e+1} \nmid x\right),$$

la e más pequeña para la cual $e+1$ sería demasiado.

- Podemos acotar la búsqueda en x porque si $q^e \mid x$, entonces $e < q^e \leqslant x$.
- Es decir, el exponente de q en la factorización de x es

$$(\mu e < x) \left(q^{e+1} \nmid x\right).$$

- Ahora supongamos que el primo q es p_y. Definimos

$$(x)_y^* = (\mu e < x) \left(p_y^{e+1} \nmid x\right)$$

así que $(x)_y^*$ es el exponente de p_y en la factorización prima de x.

Ejemplo 19 III

- En segundo lugar, para nuestra función de decodificación, necesitamos uno menos que el exponente del primo p_y en la factorización del código de secuencia.

Ejemplo 19 III

- En segundo lugar, para nuestra función de decodificación, necesitamos uno menos que el exponente del primo p_y en la factorización del código de secuencia.
- En consecuencia, definimos

$$(x)_y = (x)^*_y \dot{-} 1 = (\mu e < x) \left(p_y^{e+1} \nmid x\right) \dot{-} 1.$$

Ejemplo 19 III

- En segundo lugar, para nuestra función de decodificación, necesitamos uno menos que el exponente del primo p_y en la factorización del código de secuencia.
- En consecuencia, definimos

$$(x)_y = (x)_y^* \dot{-} 1 = (\mu e < x) \left(p_y^{e+1} \nmid x\right) \dot{-} 1.$$

- El lado derecho de esta ecuación está escrito en nuestro lenguaje, por lo que la función es recursiva primitiva.

Ejemplo 19 III

- En segundo lugar, para nuestra función de decodificación, necesitamos uno menos que el exponente del primo p_y en la factorización del código de secuencia.
- En consecuencia, definimos

$$(x)_y = (x)_y^* \dot{-} 1 = (\mu e < x) \left(p_y^{e+1} \nmid x\right) \dot{-} 1.$$

- El lado derecho de esta ecuación está escrito en nuestro lenguaje, por lo que la función es recursiva primitiva.
- La función prueba potencias de p_y hasta que encuentra la más grande en la factorización de x, y luego retrocede en 1. Si p_y no divide a x, entonces $(x)_y = 0$, lo cual es bastante inofensivo.

Ejemplo 19 III

- En segundo lugar, para nuestra función de decodificación, necesitamos uno menos que el exponente del primo p_y en la factorización del código de secuencia.
- En consecuencia, definimos

$$(x)_y = (x)_y^* \dot{-} 1 = (\mu e < x) \left(p_y^{e+1} \nmid x\right) \dot{-} 1.$$

- El lado derecho de esta ecuación está escrito en nuestro lenguaje, por lo que la función es recursiva primitiva.
- La función prueba potencias de p_y hasta que encuentra la más grande en la factorización de x, y luego retrocede en 1. Si p_y no divide a x, entonces $(x)_y = 0$, lo cual es bastante inofensivo.
- También $(0)_y = 0$, pero por una razón diferente.

Operación de búsqueda I

- Obtenemos la clase de funciones parciales recursivas generales al permitir que las funciones se construyan mediante el uso de la búsqueda (además de la composición y la recursividad primitiva).

Operación de búsqueda I

- Obtenemos la clase de funciones parciales recursivas generales al permitir que las funciones se construyan mediante el uso de la búsqueda (además de la composición y la recursividad primitiva).
- La búsqueda (también llamada minimización) corresponde a un operador-μ ilimitado.

Operación de búsqueda I

- Obtenemos la clase de funciones parciales recursivas generales al permitir que las funciones se construyan mediante el uso de la búsqueda (además de la composición y la recursividad primitiva).
- La búsqueda (también llamada minimización) corresponde a un operador-μ ilimitado.
- Para una función parcial g de aridad $(k+1)$, definimos

$$\mu y[g(\vec{x},y)=0] = \begin{cases} \text{el menor número } y \text{ tal que } g(\vec{x},y)=0 \text{ y para todo } t \\ \text{menor que } y, \text{ el valor } g(\vec{x},t) \text{ está definido y es distinto} \\ \text{de cero, si existe tal } y \\ \text{indefinido, si no existe tal } y. \end{cases}$$

Operación de búsqueda I

- Obtenemos la clase de funciones parciales recursivas generales al permitir que las funciones se construyan mediante el uso de la búsqueda (además de la composición y la recursividad primitiva).
- La búsqueda (también llamada minimización) corresponde a un operador-μ ilimitado.
- Para una función parcial g de aridad $(k+1)$, definimos

$$\mu y[g(\vec{x},y)=0] = \begin{cases} \text{el menor número } y \text{ tal que } g(\vec{x},y)=0 \text{ y para todo } t \\ \text{menor que } y, \text{ el valor } g(\vec{x},t) \text{ está definido y es distinto} \\ \text{de cero, si existe tal } y \\ \text{indefinido, si no existe tal } y. \end{cases}$$

- Esta cantidad puede no estar definida para algunos (o todos) los valores de $\vec{x}$, incluso si g resulta ser una función total.

Operación de búsqueda II

- **Ejemplo:** Supongamos que conocemos la siguiente información sobre la función g:

$$g(0,0) = 7 \qquad g(0,1) = 0$$
$$g(1,0) = \uparrow \qquad g(1,1) = 0$$

Operación de búsqueda II

- **Ejemplo:** Supongamos que conocemos la siguiente información sobre la función g:

$$\begin{aligned} g(0,0) &= 7 & g(0,1) &= 0 \\ g(1,0) &= \uparrow & g(1,1) &= 0 \end{aligned}$$

- Entonces $\mu y[g(0,y) = 0]$ es 1 y $\mu y[g(1,y) = 0]$ está indefinida.

Operación de búsqueda II

- **Ejemplo:** Supongamos que conocemos la siguiente información sobre la función g:

$$\begin{aligned} g(0,0) &= 7 & g(0,1) &= 0 \\ g(1,0) &= \uparrow & g(1,1) &= 0 \end{aligned}$$

- Entonces $\mu y[g(0,y) = 0]$ es 1 y $\mu y[g(1,y) = 0]$ está indefinida.
- Se dice que una función parcial h de aridad k se obtiene a partir de g mediante *búsqueda* si la ecuación

$$h(\vec{x}) = \mu y[g(\vec{x}, y) = 0]$$

se cumple para todo $\vec{x}$, con el entendimiento habitual de que para que una ecuación se cumpla, ambos lados no están definidos o ambos lados están definidos y son iguales.

Operación de búsqueda III

- Luego decimos que una función parcial es *recursiva general* si se puede construir a partir de las funciones cero, sucesor y proyección, donde se nos permite usar composición, recursividad primitiva y búsqueda.

Operación de búsqueda III

- Luego decimos que una función parcial es *recursiva general* si se puede construir a partir de las funciones cero, sucesor y proyección, donde se nos permite usar composición, recursividad primitiva y búsqueda.
- La colección de funciones parciales recursivas generales incluye todas las funciones recursivas primitivas (que son todas totales) y más.

Operación de búsqueda III

- Luego decimos que una función parcial es *recursiva general* si se puede construir a partir de las funciones cero, sucesor y proyección, donde se nos permite usar composición, recursividad primitiva y búsqueda.
- La colección de funciones parciales recursivas generales incluye todas las funciones recursivas primitivas (que son todas totales) y más.
- Como ejemplo extremo, la función vacía de aridad uno (es decir, la función con dominio vacío) es una función parcial recursiva general; se obtiene mediante búsqueda a partir de la función constante $g(x, y) = 3$.

Ejemplo 11A I

- Si f es una función parcial recursiva general, entonces también lo son las funciones s y p:

$$s(\vec{x}, y) = \sum_{t<y} f(\vec{x}, t) \quad \text{y} \quad p(\vec{x}, y) = \prod_{t<y} f(\vec{x}, t).$$

Ejemplo 11A I

- Si f es una función parcial recursiva general, entonces también lo son las funciones s y p:

$$s(\vec{x}, y) = \sum_{t<y} f(\vec{x}, t) \quad \text{y} \quad p(\vec{x}, y) = \prod_{t<y} f(\vec{x}, t).$$

- Para cualquier $\vec{x}$ particular, estas funciones se definen para todo y o para un segmento inicial finito de los números naturales.

Ejemplo 11A I

- Si f es una función parcial recursiva general, entonces también lo son las funciones s y p:

$$s(\vec{x}, y) = \sum_{t<y} f(\vec{x}, t) \quad \text{y} \quad p(\vec{x}, y) = \prod_{t<y} f(\vec{x}, t).$$

- Para cualquier $\vec{x}$ particular, estas funciones se definen para todo y o para un segmento inicial finito de los números naturales.
- Definimos una relación R como una *relación recursiva general* si su función característica C_R (que por definición es siempre total) es una función recursiva general.

Ejemplo 11A I

- Si f es una función parcial recursiva general, entonces también lo son las funciones s y p:

$$s(\vec{x}, y) = \sum_{t<y} f(\vec{x}, t) \quad \text{y} \quad p(\vec{x}, y) = \prod_{t<y} f(\vec{x}, t).$$

- Para cualquier $\vec{x}$ particular, estas funciones se definen para todo y o para un segmento inicial finito de los números naturales.
- Definimos una relación R como una *relación recursiva general* si su función característica C_R (que por definición es siempre total) es una función recursiva general.
- Como caso especial de búsqueda, siempre que R es una relación recursiva general de aridad $(k+1)$, entonces la función h de aridad k definida por la ecuación

$$h(\vec{x}) = \mu y R(\vec{x}, y)$$

es una función parcial recursiva general.

Ejemplo 11A II

- Nuevamente tenemos una regla de sustitución: siempre que Q sea una relación recursiva general de aridad n y $g_1, \ldots, g_n$ sean funciones recursivas generales totales de aridad k, entonces la relación de aridad k

$$\{\vec{x} \mid < g_1(\vec{x}), \ldots, g_n(\vec{x}) > \in Q\}$$

es recursiva general porque su función característica se obtiene de C_Q y $g_1, \ldots, g_n$ por composición.

Ejemplo 11A II

- Nuevamente tenemos una regla de sustitución: siempre que Q sea una relación recursiva general de aridad n y $g_1, \ldots, g_n$ sean funciones recursivas generales totales de aridad k, entonces la relación de aridad k

$$\{\vec{x} \mid < g_1(\vec{x}), \ldots, g_n(\vec{x}) > \in Q\}$$

  es recursiva general porque su función característica se obtiene de C_Q y $g_1, \ldots, g_n$ por composición.
- Pero esto no es necesariamente cierto si las funciones g_i no son totales.

Ejemplo 11A II

- Nuevamente tenemos una regla de sustitución: siempre que Q sea una relación recursiva general de aridad n y $g_1, \ldots, g_n$ sean funciones recursivas generales totales de aridad k, entonces la relación de aridad k

$$\{\vec{x} \mid < g_1(\vec{x}), \ldots, g_n(\vec{x}) > \in Q\}$$

  es recursiva general porque su función característica se obtiene de C_Q y $g_1, \ldots, g_n$ por composición.
- Pero esto no es necesariamente cierto si las funciones g_i no son totales.
- En ese caso, la composición no nos da C_Q completo, sino una subfunción no total del mismo.

Ejemplo 11A III

- Por ejemplo, para cualquier función recursiva general total f, su gráfica

$$\{\vec{x}, y \mid f(\vec{x}) = y\}$$

es una relación recursiva general.

Ejemplo 11A III

- Por ejemplo, para cualquier función recursiva general total f, su gráfica

$$\{\vec{x}, y \mid f(\vec{x}) = y\}$$

  es una relación recursiva general.
- De manera similar, la gráfica de cualquier función recursiva primitiva será una relación recursiva primitiva).

Ejemplo 11A III

- Por ejemplo, para cualquier función recursiva general total f, su gráfica

$$\{\vec{x}, y \mid f(\vec{x}) = y\}$$

  es una relación recursiva general.
- De manera similar, la gráfica de cualquier función recursiva primitiva será una relación recursiva primitiva).
- Veremos más adelante que esto puede fallar en el caso de una función no total.

Ejemplo 11A III

- Por ejemplo, para cualquier función recursiva general total f, su gráfica

$$\{\vec{x}, y \mid f(\vec{x}) = y\}$$

  es una relación recursiva general.
- De manera similar, la gráfica de cualquier función recursiva primitiva será una relación recursiva primitiva).
- Veremos más adelante que esto puede fallar en el caso de una función no total.
- **Teorema:**

Ejemplo 11A III

- Por ejemplo, para cualquier función recursiva general total f, su gráfica

$$\{\vec{x}, y \mid f(\vec{x}) = y\}$$

  es una relación recursiva general.
- De manera similar, la gráfica de cualquier función recursiva primitiva será una relación recursiva primitiva).
- Veremos más adelante que esto puede fallar en el caso de una función no total.
- **Teorema:**

  (d) Si Q y R son relaciones recursivas generales de aridad k, entonces también lo son $\overline{Q}$, $Q \cap R$ y $Q \cup R$.

Ejemplo 11A III

- Por ejemplo, para cualquier función recursiva general total f, su gráfica

$$\{\vec{x}, y \mid f(\vec{x}) = y\}$$

  es una relación recursiva general.
- De manera similar, la gráfica de cualquier función recursiva primitiva será una relación recursiva primitiva).
- Veremos más adelante que esto puede fallar en el caso de una función no total.
- **Teorema:**

  (d) Si Q y R son relaciones recursivas generales de aridad k, entonces también lo son $\overline{Q}$, $Q \cap R$ y $Q \cup R$.

  (e) Si Q es una relación recursiva general de aridad $(k+1)$, entonces también lo son las relaciones

$$\{< \vec{x}, y > | \ (\forall t < y) Q(\vec{x}, t)\} \quad \text{y} \quad \{< \vec{x}, y > | \ (\exists t < y) Q(\vec{x}, t)\}.$$

Ejemplo 11A III

- Por ejemplo, para cualquier función recursiva general total f, su gráfica

$$\{\vec{x}, y \mid f(\vec{x}) = y\}$$

  es una relación recursiva general.
- De manera similar, la gráfica de cualquier función recursiva primitiva será una relación recursiva primitiva).
- Veremos más adelante que esto puede fallar en el caso de una función no total.
- **Teorema:**
  - (d) Si Q y R son relaciones recursivas generales de aridad k, entonces también lo son $\overline{Q}$, $Q \cap R$ y $Q \cup R$.
  - (e) Si Q es una relación recursiva general de aridad $(k+1)$, entonces también lo son las relaciones

$$\{< \vec{x}, y > \mid (\forall t < y) Q(\vec{x}, t)\} \quad \text{y} \quad \{< \vec{x}, y > \mid (\exists t < y) Q(\vec{x}, t)\}.$$

- La prueba no ha cambiado.

Ejemplo 11A IV

- La definición por casos sigue siendo válida, pero debemos ser más cuidadosos con su demostración.

Ejemplo 11A IV

- La definición por casos sigue siendo válida, pero debemos ser más cuidadosos con su demostración.
- Supongamos que g es una función parcial recursiva general de aridad k y que Q es una relación recursiva general de aridad k.

Ejemplo 11A IV

- La definición por casos sigue siendo válida, pero debemos ser más cuidadosos con su demostración.
- Supongamos que g es una función parcial recursiva general de aridad k y que Q es una relación recursiva general de aridad k.
- Defina g^Q por la ecuación

$$g^Q(\vec{x}) = \begin{cases} g(\vec{x}) & \text{si } Q(\vec{x}) \\ 0 & \text{si no } Q(\vec{x}) \end{cases}$$

Ejemplo 11A IV

- La definición por casos sigue siendo válida, pero debemos ser más cuidadosos con su demostración.
- Supongamos que g es una función parcial recursiva general de aridad k y que Q es una relación recursiva general de aridad k.
- Defina g^Q por la ecuación

$$g^Q(\vec{x}) = \begin{cases} g(\vec{x}) & \text{si } Q(\vec{x}) \\ 0 & \text{si no } Q(\vec{x}) \end{cases}$$

- Entonces g^Q también es una función parcial recursiva general.

Ejemplo 11A IV

- La definición por casos sigue siendo válida, pero debemos ser más cuidadosos con su demostración.
- Supongamos que g es una función parcial recursiva general de aridad k y que Q es una relación recursiva general de aridad k.
- Defina g^Q por la ecuación

$$g^Q(\vec{x}) = \begin{cases} g(\vec{x}) & \text{si } Q(\vec{x}) \\ 0 & \text{si no } Q(\vec{x}) \end{cases}$$

- Entonces g^Q también es una función parcial recursiva general.
- Pero no podemos escribir simplemente $g^Q(\vec{x}) = g(\vec{x}) \cdot C_Q(\vec{x})$ porque puede haber algunas $\vec{x}$ que no estén en el dominio de g (por lo que el lado derecho no estará definido) y no en Q (por lo que el lado izquierdo será 0).

Ejemplo 11A V

- En cambio, primero podemos usar la recursividad primitiva para construir la función.

$$G(\vec{x}, 0) = 0$$
$$G(\vec{x}, y+1) = g(\vec{x})$$

que es como g excepto que tiene un "interruptor de encendido y apagado".

Ejemplo 11A V

- En cambio, primero podemos usar la recursividad primitiva para construir la función.

$$G(\vec{x}, 0) = 0$$
$$G(\vec{x}, y+1) = g(\vec{x})$$

que es como g excepto que tiene un "interruptor de encendido y apagado".

- Entonces tenemos la ecuación

$$g^Q(\vec{x}) = G(\vec{x}, C_Q(\vec{x})).$$

mostrando que g^Q es una función parcial recursiva general.

Ejemplo 11A VI

- Ahora, si también tenemos otra función parcial recursiva general f de aridad k y definimos

$$h(\vec{x}) = \begin{cases} f(\vec{x}) & \text{si } Q(\vec{x}) \\ g(\vec{x}) & \text{si no } Q(\vec{x}) \end{cases}$$

entonces h es una función parcial recursiva general porque $h(x) = f^{Q}(\vec{x}) + g^{\overline{Q}}(\vec{x})$.

Ejemplo 26A

- Asuma que g es una función parcial recursiva general de aridad $(k+2)$ y sea f la única función de aridad $(k+1)$ para la cual

$$f(\vec{x}, y) = g(\overline{f}(\vec{x}, y), \vec{x}, y)$$

para todo $\vec{x}$ y y.

Ejemplo 26A

- Asuma que g es una función parcial recursiva general de aridad $(k+2)$ y sea f la única función de aridad $(k+1)$ para la cual

$$f(\vec{x}, y) = g(\overline{f}(\vec{x}, y), \vec{x}, y)$$

para todo $\vec{x}$ y y.

- Si g no es total, entonces es posible que para algunos valores de $\vec{x}$, la cantidad $f(\vec{x}, y)$ se defina sólo para un número finito de ys.

Ejemplo 26A

- Asuma que g es una función parcial recursiva general de aridad $(k+2)$ y sea f la única función de aridad $(k+1)$ para la cual

$$f(\vec{x}, y) = g(\overline{f}(\vec{x}, y), \vec{x}, y)$$

  para todo $\vec{x}$ y y.
- Si g no es total, entonces es posible que para algunos valores de $\vec{x}$, la cantidad $f(\vec{x}, y)$ se defina sólo para un número finito de ys.
- Entonces f también es una función parcial recursiva general.

Ejemplo 26A

- Asuma que g es una función parcial recursiva general de aridad $(k+2)$ y sea f la única función de aridad $(k+1)$ para la cual

$$f(\vec{x}, y) = g(\overline{f}(\vec{x}, y), \vec{x}, y)$$

  para todo $\vec{x}$ y y.
- Si g no es total, entonces es posible que para algunos valores de $\vec{x}$, la cantidad $f(\vec{x}, y)$ se defina sólo para un número finito de ys.
- Entonces f también es una función parcial recursiva general.
- La prueba es como antes.

Ejemplo 26A

- Asuma que g es una función parcial recursiva general de aridad $(k+2)$ y sea f la única función de aridad $(k+1)$ para la cual

$$f(\vec{x}, y) = g(\overline{f}(\vec{x}, y), \vec{x}, y)$$

  para todo $\vec{x}$ y y.
- Si g no es total, entonces es posible que para algunos valores de $\vec{x}$, la cantidad $f(\vec{x}, y)$ se defina sólo para un número finito de ys.
- Entonces f también es una función parcial recursiva general.
- La prueba es como antes.
- Anteriormente se argumentó que la colección de funciones recursivas primitivas no puede contener todas las funciones totales efectivamente calculables.

Ejemplo 26A

- Asuma que g es una función parcial recursiva general de aridad $(k+2)$ y sea f la única función de aridad $(k+1)$ para la cual

$$f(\vec{x}, y) = g(\overline{f}(\vec{x}, y), \vec{x}, y)$$

  para todo $\vec{x}$ y y.
- Si g no es total, entonces es posible que para algunos valores de $\vec{x}$, la cantidad $f(\vec{x}, y)$ se defina sólo para un número finito de ys.
- Entonces f también es una función parcial recursiva general.
- La prueba es como antes.
- Anteriormente se argumentó que la colección de funciones recursivas primitivas no puede contener todas las funciones totales efectivamente calculables.
- Pero la tesis de Church implica que la colección de funciones parciales recursivas generales las contiene todas, así como las funciones no totales efectivamente calculables.

Ejemplo 26A

- Asuma que g es una función parcial recursiva general de aridad $(k+2)$ y sea f la única función de aridad $(k+1)$ para la cual

$$f(\vec{x}, y) = g(\overline{f}(\vec{x}, y), \vec{x}, y)$$

  para todo $\vec{x}$ y y.
- Si g no es total, entonces es posible que para algunos valores de $\vec{x}$, la cantidad $f(\vec{x}, y)$ se defina sólo para un número finito de ys.
- Entonces f también es una función parcial recursiva general.
- La prueba es como antes.
- Anteriormente se argumentó que la colección de funciones recursivas primitivas no puede contener todas las funciones totales efectivamente calculables.
- Pero la tesis de Church implica que la colección de funciones parciales recursivas generales las contiene todas, así como las funciones no totales efectivamente calculables.
- Como se ha indicado informalmente, no es posible "diagonalizar" la colección de funciones parciales recursivas generales.

Ejercicios I

1. ¿Entiendes la recursividad primitiva? ¿Eres positivo? Si eres positivo, ve al Ejercicio 2.
2. Resta 1. Ve al ejercicio 1.
3. Dé un árbol de construcción completo para la multiplicación (item 3).
4. Demuestre que la función que eleva al cuadrado $f(x) = x^2$ es recursiva primitiva proporcionando un árbol de construcción que muestre en detalle cómo se puede construir a partir de funciones iniciales mediante el uso de composición y recursividad primitiva. (En las hojas del árbol, debes tener sólo funciones iniciales; por ejemplo, si quieres usar la suma, debes construirla).
5. Demuestre que la función

$$\text{pos}(x) = \begin{cases} 1 & \text{si } x > 0 \\ 0 & \text{si } x = 0 \end{cases}$$

es recursiva primitiva dando un árbol de construcción.

Ejercicios II

6. Demuestre que la función de paridad

$$C_{\text{odd}}(x) = \begin{cases} 1 & \text{si } x \text{ es impar} \\ 0 & \text{si } x \text{ es par} \end{cases}$$

es recursiva primitiva dando un árbol de construcción.

7. Demuestre que la función $< x, y > \mapsto |x - y|$ es recursiva primitiva.
8. Demuestre que la función $< x, y > \mapsto \text{máx} < x, y >$ es recursiva primitiva.
9. Demuestre que la función $< x, y > \mapsto \text{mín} < x, y >$ es recursiva primitiva.
10. Utilice el postulado de Bertrand para demostrar (por inducción) que $p_x \leqslant 2^{x+1}$ y que la igualdad se cumple sólo para $x = 0$.