7 Funciones recursivas generales II

José de Jesús Lavalle Martínez

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

Contenido

Motivación

En la sesión anterior empezamos a construir un catálogo de funciones recursivas primitivas, esta vez vamos a continuar construyendo ese catálogo.

Ejemplo 13 I

- De manera similar, la función h que verifica sus dos entradas x y y para ver si o no $x \leqslant y$

$$h(x,y) = \begin{cases} 1 & \text{si } x \leqslant y \\ 0 & \text{en caso contrario} \end{cases}$$

es recursiva primitiva porque $h(x,y) = z(x \dot{-} y)$.

Ejemplo 13 I

- De manera similar, la función h que verifica sus dos entradas x y y para ver si o no $x \leqslant y$

$$h(x,y) = \begin{cases} 1 & \text{si } x \leqslant y \\ 0 & \text{en caso contrario} \end{cases}$$

  es recursiva primitiva porque $h(x,y) = z(x \dot{-} y)$.
- Los ítems 12 y 13 pueden reformularse en términos de relaciones (en lugar de funciones).

Ejemplo 13 I

- De manera similar, la función h que verifica sus dos entradas x y y para ver si o no $x \leqslant y$

$$h(x,y) = \begin{cases} 1 & \text{si } x \leqslant y \\ 0 & \text{en caso contrario} \end{cases}$$

  es recursiva primitiva porque $h(x,y) = z(x \dot{-} y)$.
- Los ítems 12 y 13 pueden reformularse en términos de relaciones (en lugar de funciones).
- Supongamos que R es una relación de aridad k sobre los números naturales, es decir, R es algún conjunto de tuplas-k de números naturales: $R \subseteq N^k$.

Ejemplo 13 II

- Definimos R como una relación recursiva primitiva si su función característica

$$C_R(x_1, \ldots, x_k) = \begin{cases} 1 & \text{si } (x_1, \ldots, x_k) \in R \\ 0 & \text{en caso contrario} \end{cases}$$

  es una función recursiva primitiva.
- Por ejemplo, el item 13 establece que la relación de ordenamiento $\{< x, y > | x \leqslant y\}$ es una relación binaria recursiva primitiva.
- Y el item 12 establece que $\{0\}$ es una relación unaria recursiva primitiva.

Ejemplo 13 III

- De la composición derivamos la regla de sustitución: si Q es una relación recursiva primitiva de aridad n, y $g_1, \ldots, g_n$ son funciones recursivas primitivas de aridad k, entonces la relación de aridad k.

$$\{\vec{x} | g_1(\vec{x}), \ldots, g_n(\vec{x})\}$$

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

Ejemplo 13 III

- De la composición derivamos la regla de sustitución: si Q es una relación recursiva primitiva de aridad n, y $g_1, \ldots, g_n$ son funciones recursivas primitivas de aridad k, entonces la relación de aridad k.

$$\{\vec{x} | g_1(\vec{x}), \ldots, g_n(\vec{x})\}$$

  es recursiva primitiva porque su función característica se obtiene de C_Q y $g_1, \ldots, g_n$ por composición.
- De una relación Q, podemos formar su *complemento* $\overline{Q}$:

$$\overline{Q} = \{\vec{x} | \vec{x} \text{ no está en } Q\}.$$

Ejemplo 13 III

- De la composición derivamos la regla de sustitución: si Q es una relación recursiva primitiva de aridad n, y $g_1, \ldots, g_n$ son funciones recursivas primitivas de aridad k, entonces la relación de aridad k.

$$\{\vec{x} | g_1(\vec{x}), \ldots, g_n(\vec{x})\}$$

  es recursiva primitiva porque su función característica se obtiene de C_Q y $g_1, \ldots, g_n$ por composición.
- De una relación Q, podemos formar su *complemento* $\overline{Q}$:

$$\overline{Q} = \{\vec{x} | \vec{x} \text{ no está en } Q\}.$$

- A partir de dos relaciones Q y R de aridad k (para el mismo k), podemos formar su *intersección*,

$$Q \cap R = \{\vec{x} | \text{ ambos } \vec{x} \in Q \text{ y } \vec{x} \in R\}$$

Ejemplo 13 III

- De la composición derivamos la regla de sustitución: si Q es una relación recursiva primitiva de aridad n, y $g_1, \ldots, g_n$ son funciones recursivas primitivas de aridad k, entonces la relación de aridad k.

$$\{\vec{x} | g_1(\vec{x}), \ldots, g_n(\vec{x})\}$$

  es recursiva primitiva porque su función característica se obtiene de C_Q y $g_1, \ldots, g_n$ por composición.
- De una relación Q, podemos formar su *complemento* $\overline{Q}$:

$$\overline{Q} = \{\vec{x} | \vec{x} \text{ no está en } Q\}.$$

- A partir de dos relaciones Q y R de aridad k (para el mismo k), podemos formar su *intersección*,

$$Q \cap R = \{\vec{x} | \text{ ambos } \vec{x} \in Q \text{ y } \vec{x} \in R\}$$

- y su *unión*

$$Q \cup R = \{\vec{x} | \text{ o } \vec{x} \in Q \text{ o } \vec{x} \in R \text{ o ambos}\}$$

Ejemplo 13 IV

- Podemos simplificar ligeramente la notación escribiendo, en lugar de $\vec{x} \in Q$, simplemente $Q(\vec{x})$.

Ejemplo 13 IV

- Podemos simplificar ligeramente la notación escribiendo, en lugar de $\vec{x} \in Q$, simplemente $Q(\vec{x})$.
- En esta notación,

$$\begin{aligned} \overline{Q} &= \{\vec{x} | \text{ no } Q(\vec{x})\}, \\ Q \cap R &= \{\vec{x} | \text{ ambos } Q(\vec{x}) \text{ y } R(\vec{x})\}, \\ Q \cup R &= \{\vec{x} | \text{ o } Q(\vec{x}) \text{ o } R(\vec{x}) \text{ o ambos}\}. \end{aligned}$$

Ejemplo 13 IV

- Podemos simplificar ligeramente la notación escribiendo, en lugar de $\vec{x} \in Q$, simplemente $Q(\vec{x})$.
- En esta notación,

$$\begin{aligned} \overline{Q} &= \{\vec{x} | \text{ no } Q(\vec{x})\}, \\ Q \cap R &= \{\vec{x} | \text{ ambos } Q(\vec{x}) \text{ y } R(\vec{x})\}, \\ Q \cup R &= \{\vec{x} | \text{ o } Q(\vec{x}) \text{ o } R(\vec{x}) \text{ o ambos}\}. \end{aligned}$$

- El siguiente teorema nos asegura que estas construcciones preservan la recursividad primitiva.

Ejemplo 13 IV

- Podemos simplificar ligeramente la notación escribiendo, en lugar de $\vec{x} \in Q$, simplemente $Q(\vec{x})$.
- En esta notación,

$$\begin{aligned} \overline{Q} &= \{\vec{x} | \text{ no } Q(\vec{x})\}, \\ Q \cap R &= \{\vec{x} | \text{ ambos } Q(\vec{x}) \text{ y } R(\vec{x})\}, \\ Q \cup R &= \{\vec{x} | \text{ o } Q(\vec{x}) \text{ o } R(\vec{x}) \text{ o ambos}\}. \end{aligned}$$

- El siguiente teorema nos asegura que estas construcciones preservan la recursividad primitiva.
- Es decir, cuando se aplican a relaciones recursivas primitivas, producen relaciones recursivas primitivas.

Ejemplo 13 IV

- Podemos simplificar ligeramente la notación escribiendo, en lugar de $\vec{x} \in Q$, simplemente $Q(\vec{x})$.
- En esta notación,

$$\begin{aligned} \overline{Q} &= \{\vec{x} | \text{ no } Q(\vec{x})\}, \\ Q \cap R &= \{\vec{x} | \text{ ambos } Q(\vec{x}) \text{ y } R(\vec{x})\}, \\ Q \cup R &= \{\vec{x} | \text{ o } Q(\vec{x}) \text{ o } R(\vec{x}) \text{ o ambos}\}. \end{aligned}$$

- El siguiente teorema nos asegura que estas construcciones preservan la recursividad primitiva.
- Es decir, cuando se aplican a relaciones recursivas primitivas, producen relaciones recursivas primitivas.
- Este teorema será útil para ampliar nuestro catálogo de relaciones y funciones recursivas primitivas.

Ejemplo 13 V

Teorema *Supongamos que Q y R son relaciones recursivas primitivas de aridad k. Entonces las siguientes relaciones también son recursivas primitivas.*

1. *El complemento $\overline{Q}$ de Q:*

$$\overline{Q} = \{\vec{x} | \text{ no } Q(\vec{x})\}$$

Ejemplo 13 V

Teorema *Supongamos que Q y R son relaciones recursivas primitivas de aridad k. Entonces las siguientes relaciones también son recursivas primitivas.*

1. *El complemento $\overline{Q}$ de Q:*

$$\overline{Q} = \{\vec{x} | \text{ no } Q(\vec{x})\}$$

2. *La intersección $Q \cap R$ de Q y R:*

$$Q \cap R = \{\vec{x} | \text{ ambos } Q(\vec{x}) \text{ y } R(\vec{x})\}$$

Ejemplo 13 V

Teorema *Supongamos que Q y R son relaciones recursivas primitivas de aridad k. Entonces las siguientes relaciones también son recursivas primitivas.*

1. *El complemento $\overline{Q}$ de Q:*

$$\overline{Q} = \{\vec{x} | \text{ no } Q(\vec{x})\}$$

2. *La intersección $Q \cap R$ de Q y R:*

$$Q \cap R = \{\vec{x} | \text{ ambos } Q(\vec{x}) \text{ y } R(\vec{x})\}$$

3. *La unión $Q \cup R$ de Q y R:*

$$Q \cup R = \{\vec{x} | \text{ o } Q(\vec{x}) \text{ o } R(\vec{x}) \text{ o ambos}\}$$

Ejemplo 13 VI

1. $$C_{\overline{Q}}(\vec{x}) = z(C_Q(\vec{x}))$$

   donde z es la función del item 12. Es decir, $C_{\overline{Q}}$ se obtiene por composición de funciones que se sabe que son recursivas primitivas. Las demás partes se prueban de manera similar; necesitamos hacer la función característica a partir de partes recursivas primitivas.

Ejemplo 13 VI

1. $$C_{\overline{Q}}(\vec{x}) = z(C_Q(\vec{x}))$$

   donde z es la función del item 12. Es decir, $C_{\overline{Q}}$ se obtiene por composición de funciones que se sabe que son recursivas primitivas. Las demás partes se prueban de manera similar; necesitamos hacer la función característica a partir de partes recursivas primitivas.

2. $$C_{Q \cap R}(\vec{x}) = C_Q(\vec{x}) \cdot C_R(\vec{x})$$

Ejemplo 13 VI

1. $$C_{\overline{Q}}(\vec{x}) = z(C_Q(\vec{x}))$$

   donde z es la función del item 12. Es decir, $C_{\overline{Q}}$ se obtiene por composición de funciones que se sabe que son recursivas primitivas. Las demás partes se prueban de manera similar; necesitamos hacer la función característica a partir de partes recursivas primitivas.

2. $$C_{Q\cap R}(\vec{x}) = C_Q(\vec{x}) \cdot C_R(\vec{x})$$

3. $$C_{Q\cup R}(\vec{x}) = \text{pos}[C_Q(\vec{x}) + C_R(\vec{x})]$$

   donde pos es la función del Ejercicio 5 de este capítulo. Por ejemplo, podemos aplicar este teorema para concluir que $>$ y $=$ son relaciones recursivas primitivas: $\dashv$

Ejemplo 14

La relación $\{< x, y > | x > y\}$ es recursiva primitiva porque es el complemento de la relación $\leqslant$ del item 13.

Ejemplo 15 I

- La relación $\{< x, y > | x = y\}$ es recursiva primitiva porque es la intersección de las relaciones $\leqslant$ y $\geqslant$, y $\geqslant$ se obtiene de $\leqslant$ mediante transformación explícita.

Ejemplo 15 I

- La relación $\{< x, y > | x = y\}$ es recursiva primitiva porque es la intersección de las relaciones $\leqslant$ y $\geqslant$, y $\geqslant$ se obtiene de $\leqslant$ mediante transformación explícita.
- Del item 15 y de la regla de sustitución se deduce que para cualquier función recursiva primitiva f, su gráfica

$$\{< \vec{x}, y > | f(\vec{x}) = y\}$$

Ejemplo 15 I

- La relación $\{< x, y > | x = y\}$ es recursiva primitiva porque es la intersección de las relaciones $\leqslant$ y $\geqslant$, y $\geqslant$ se obtiene de $\leqslant$ mediante transformación explícita.
- Del item 15 y de la regla de sustitución se deduce que para cualquier función recursiva primitiva f, su gráfica

$$\{< \vec{x}, y > | f(\vec{x}) = y\}$$

- es una relación recursiva primitiva.

Ejemplo 15 II

Definición por casos: *Supongamos que Q es una relación recursiva primitiva de aridad k, y que f y g son funciones recursivas primitivas de aridad k. Entonces la función h definida por la ecuación*

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

también es recursiva primitiva.
Prueba: $h(\vec{x}) = f(\vec{x}) \cdot C_Q(\vec{x}) + g(\vec{x}) \cdot C_{\overline{Q}}(\vec{x})$. $\dashv$

Ejemplo 15 III

- Este resultado puede extenderse a más de dos casos; vea el Ejercicio 12 de este capítulo.

Ejemplo 15 III

- Este resultado puede extenderse a más de dos casos; vea el Ejercicio 12 de este capítulo.
- Por ejemplo, es posible que deseemos manejar una ecuación de la forma

$$h(\vec{x}) = \begin{cases} f_1(\vec{x}) & \text{si } Q(\vec{x}) \text{ y } R(\vec{x}) \\ f_2(\vec{x}) & \text{si } Q(\vec{x}) \text{ y no } R(\vec{x}) \\ f_3(\vec{x}) & \text{si } R(\vec{x}) \text{ y no } Q(\vec{x}) \\ f_4(\vec{x}) & \text{si ni } Q(\vec{x}) \text{ ni } R(\vec{x}) \end{cases}$$

Ejemplo 15 III

- Este resultado puede extenderse a más de dos casos; vea el Ejercicio 12 de este capítulo.
- Por ejemplo, es posible que deseemos manejar una ecuación de la forma

$$h(\vec{x}) = \begin{cases} f_1(\vec{x}) & \text{si } Q(\vec{x}) \text{ y } R(\vec{x}) \\ f_2(\vec{x}) & \text{si } Q(\vec{x}) \text{ y no } R(\vec{x}) \\ f_3(\vec{x}) & \text{si } R(\vec{x}) \text{ y no } Q(\vec{x}) \\ f_4(\vec{x}) & \text{si ni } Q(\vec{x}) \text{ ni } R(\vec{x}) \end{cases}$$

- o una de la forma

$$h(\vec{x}) = \begin{cases} f_1(\vec{x}) & \text{si } Q_1(\vec{x}) \\ f_2(\vec{x}) & \text{si } Q_2(\vec{x}) \\ \cdots & \cdots \\ f_9(\vec{x}) & \text{si } Q_9(\vec{x}) \\ f_{10}(\vec{x}) & \text{si ninguna de las anteriores se cumple} \end{cases}$$

en una situación en la que se sabe que no hay dos de $Q_1, \ldots, Q_9$ que puedan cumplirse simultáneamente.

Ejemplo 15 IV

- Además, a partir de una relación Q de aridad k, podemos formar

$$\{< x_1, \ldots, x_{k-1}, y > \mid \text{para todo } t < y, < x_1, \ldots, x_{k-1}, t > \in Q\}$$

que se puede escribir en mejor notación como

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

donde el símbolo $\forall$ se lee "para todos".

Ejemplo 15 IV

- Además, a partir de una relación Q de aridad k, podemos formar

$$\{< x_1, \ldots, x_{k-1}, y > \mid \text{para todo } t < y, < x_1, \ldots, x_{k-1}, t > \in Q\}$$

que se puede escribir en mejor notación como

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

donde el símbolo $\forall$ se lee "para todos".

- Con el mismo espíritu, podemos formar

$$\{< x_1, \ldots, x_{k-1}, y > \mid \text{para algún } t < y, < x_1, \ldots, x_{k-1}, t > \in Q\}$$

que está mejor escrito como

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

donde el símbolo $\exists$ se lee como "existe . . . tal que".

Ejemplo 15 IV

- Además, a partir de una relación Q de aridad k, podemos formar

$$\{< x_1, \ldots, x_{k-1}, y > \mid \text{para todo } t < y, < x_1, \ldots, x_{k-1}, t > \in Q\}$$

que se puede escribir en mejor notación como

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

donde el símbolo $\forall$ se lee "para todos".

- Con el mismo espíritu, podemos formar

$$\{< x_1, \ldots, x_{k-1}, y > \mid \text{para algún } t < y, < x_1, \ldots, x_{k-1}, t > \in Q\}$$

que está mejor escrito como

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

donde el símbolo $\exists$ se lee como "existe . . . tal que".

- Nuevamente, estas construcciones preservan la recursividad primitiva:

Ejemplo 15 V

Teorema: *Supongamos que Q es una relación recursiva primitiva de aridad $(k+1)$. Entonces las siguientes relaciones también son recursivas primitivas:*

(a)
$$\{< \vec{x}, y > \mid (\forall t < y)\ Q(\vec{x}, t)\}$$

(b)
$$\{< \vec{x}, y > \mid (\exists t < y)\ Q(\vec{x}, t)\}$$

Prueba:

(a) El valor de la función característica en $< \vec{x}, y >$ es
$$\prod_{t<y} C_Q(\vec{x}, t)$$
Aplicar el item 11.

(b) El valor de la función característica en $< \vec{x}, y >$ es
$$\text{pos}\left[\sum_{t<y} C_Q(\vec{x}, t)\right]$$
donde pos es la función del ejercicio 5. Esta es primitiva recursiva en virtud del item 11 y del Ejercicio 5.

Ejemplo 15 VI

- Por ejemplo, podemos aplicar estos resultados para demostrar que la relación

$$\{< x, y > \mid (\exists q < y + 1)[x \cdot q = y]\}$$

es recursiva primitiva.

Ejemplo 15 VI

- Por ejemplo, podemos aplicar estos resultados para demostrar que la relación

$$\{< x, y > \mid (\exists q < y + 1)[x \cdot q = y]\}$$

  es recursiva primitiva.
- Hacemos esto observando la forma en que está escrita la línea anterior y luego completando los detalles.

Ejemplo 15 VI

- Por ejemplo, podemos aplicar estos resultados para demostrar que la relación

$$\{< x, y > \mid (\exists q < y + 1)[x \cdot q = y]\}$$

es recursiva primitiva.

- Hacemos esto observando la forma en que está escrita la línea anterior y luego completando los detalles.
- En primer lugar, la relación ternaria

$$R_1(x, y, q) \Longleftrightarrow x \cdot q = y$$

se obtiene de la relación de igualdad sustituyendo las siguientes funciones $< x, y, q > \mapsto x \cdot q$ y I_2^3.

Ejemplo 15 VI

- Por ejemplo, podemos aplicar estos resultados para demostrar que la relación

$$\{< x, y > \mid (\exists q < y + 1)[x \cdot q = y]\}$$

es recursiva primitiva.

- Hacemos esto observando la forma en que está escrita la línea anterior y luego completando los detalles.
- En primer lugar, la relación ternaria

$$R_1(x, y, q) \Longleftrightarrow x \cdot q = y$$

se obtiene de la relación de igualdad sustituyendo las siguientes funciones $< x, y, q > \mapsto x \cdot q$ y I_2^3.

- En segundo lugar, una aplicación del teorema anterior muestra que la relación ternaria

$$R_2(x, y, z) \Longleftrightarrow (\exists q < z)[x \cdot q = y]$$

es recursiva primitiva.

Ejemplo 15 VI

- Por ejemplo, podemos aplicar estos resultados para demostrar que la relación

$$\{< x, y > \mid (\exists q < y + 1)[x \cdot q = y]\}$$

es recursiva primitiva.

- Hacemos esto observando la forma en que está escrita la línea anterior y luego completando los detalles.
- En primer lugar, la relación ternaria

$$R_1(x, y, q) \Longleftrightarrow x \cdot q = y$$

se obtiene de la relación de igualdad sustituyendo las siguientes funciones $< x, y, q > \mapsto x \cdot q$ y I_2^3.

- En segundo lugar, una aplicación del teorema anterior muestra que la relación ternaria

$$R_2(x, y, z) \Longleftrightarrow (\exists q < z)[x \cdot q = y]$$

es recursiva primitiva.

- Finalmente aplicamos sustitución:

$$(\exists q < y + 1)[x \cdot q = y] \Longleftrightarrow R_2(x, y, y + 1)$$

Ejemplo 15 VI

- Por ejemplo, podemos aplicar estos resultados para demostrar que la relación

$$\{< x, y > \mid (\exists q < y + 1)[x \cdot q = y]\}$$

es recursiva primitiva.

- Hacemos esto observando la forma en que está escrita la línea anterior y luego completando los detalles.
- En primer lugar, la relación ternaria

$$R_1(x, y, q) \Longleftrightarrow x \cdot q = y$$

se obtiene de la relación de igualdad sustituyendo las siguientes funciones $< x, y, q > \mapsto x \cdot q$ y I_2^3.

- En segundo lugar, una aplicación del teorema anterior muestra que la relación ternaria

$$R_2(x, y, z) \Longleftrightarrow (\exists q < z)[x \cdot q = y]$$

es recursiva primitiva.

- Finalmente aplicamos sustitución:

$$(\exists q < y + 1)[x \cdot q = y] \Longleftrightarrow R_2(x, y, y + 1)$$

- En resumen, podemos demostrar que esta relación es recursiva primitiva examinando la forma sintáctica de su definición y verificando que se ha construido utilizando sólo piezas que se sabe que son recursivas primitivas.

Ejemplo 16 I

- La relación de divisibilidad $x|y$, es decir, la relación

$$\{< x, y > \mid x \text{ divide } y \text{ con resto } 0\},$$

es recursiva primitiva.

Ejemplo 16 I

- La relación de divisibilidad $x|y$, es decir, la relación

$$\{< x, y > \mid x \text{ divide } y \text{ con resto } 0\},$$

  es recursiva primitiva.
- Aquí adoptamos la convención de que 0 se divide a sí mismo, pero no divide a ningún entero positivo.

Ejemplo 16 I

- La relación de divisibilidad $x|y$, es decir, la relación

$$\{< x, y > \mid x \text{ divide } y \text{ con resto } 0\},$$

  es recursiva primitiva.
- Aquí adoptamos la convención de que 0 se divide a sí mismo, pero no divide a ningún entero positivo.
- Esto se debe a que

$$\begin{aligned} x|y &\iff \text{ para algún cociente } q, \text{ tenemos que } x \cdot q = y \\ &\iff (\exists q \leqslant y)[x \cdot q = y] \\ &\iff (\exists q < y + 1)[x \cdot q = y]. \end{aligned}$$

Ejemplo 16 I

- La relación de divisibilidad $x|y$, es decir, la relación

$$\{< x, y > \mid x \text{ divide } y \text{ con resto } 0\},$$

  es recursiva primitiva.
- Aquí adoptamos la convención de que 0 se divide a sí mismo, pero no divide a ningún entero positivo.
- Esto se debe a que

$$\begin{aligned} x|y &\iff \text{ para algún cociente } q, \text{ tenemos que } x \cdot q = y \\ &\iff (\exists q \leqslant y)[x \cdot q = y] \\ &\iff (\exists q < y + 1)[x \cdot q = y]. \end{aligned}$$

- Es decir, ¡la relación que examinamos en el Ejemplo anterior no es más que la relación de divisibilidad!

Ejemplo 16 II

- En efecto, estamos construyendo un determinado lenguaje de tal manera que se garantiza que cualquier función o relación definible en el lenguaje será recursiva primitiva. (Para la divisibilidad, el hecho crucial fue que la expresión "$(\exists q < y + 1)[x \cdot q = y]$" pertenecía a este lenguaje).

Ejemplo 16 II

- En efecto, estamos construyendo un determinado lenguaje de tal manera que se garantiza que cualquier función o relación definible en el lenguaje será recursiva primitiva. (Para la divisibilidad, el hecho crucial fue que la expresión "$(\exists q < y + 1)[x \cdot q = y]$" pertenecía a este lenguaje).
- Este lenguaje incluye lo siguiente:

Ejemplo 16 II

- En efecto, estamos construyendo un determinado lenguaje de tal manera que se garantiza que cualquier función o relación definible en el lenguaje será recursiva primitiva. (Para la divisibilidad, el hecho crucial fue que la expresión "$(\exists q < y + 1)[x \cdot q = y]$" pertenecía a este lenguaje).
- Este lenguaje incluye lo siguiente:
  - Variables: Las funciones de proyección son recursivas primitivas.

Ejemplo 16 II

- En efecto, estamos construyendo un determinado lenguaje de tal manera que se garantiza que cualquier función o relación definible en el lenguaje será recursiva primitiva. (Para la divisibilidad, el hecho crucial fue que la expresión "$(\exists q < y + 1)[x \cdot q = y]$" pertenecía a este lenguaje).
- Este lenguaje incluye lo siguiente:
  - Variables: Las funciones de proyección son recursivas primitivas.
  - Constantes (numerales): Las funciones constantes son recursivas primitivas.

Ejemplo 16 II

- En efecto, estamos construyendo un determinado lenguaje de tal manera que se garantiza que cualquier función o relación definible en el lenguaje será recursiva primitiva. (Para la divisibilidad, el hecho crucial fue que la expresión "$(\exists q < y + 1)[x \cdot q = y]$" pertenecía a este lenguaje).
- Este lenguaje incluye lo siguiente:
  - Variables: Las funciones de proyección son recursivas primitivas.
  - Constantes (numerales): Las funciones constantes son recursivas primitivas.
  - Símbolos de función: Podemos usar símbolos para cualquier función recursiva primitiva en la lista que se está acumulando, de acuerdo a las convenciones usuales ($+, \times, \dot{-} \ldots$ y habrá más por venir).

Ejemplo 16 II

- En efecto, estamos construyendo un determinado lenguaje de tal manera que se garantiza que cualquier función o relación definible en el lenguaje será recursiva primitiva. (Para la divisibilidad, el hecho crucial fue que la expresión "$(\exists q < y + 1)[x \cdot q = y]$" pertenecía a este lenguaje).
- Este lenguaje incluye lo siguiente:
  - Variables: Las funciones de proyección son recursivas primitivas.
  - Constantes (numerales): Las funciones constantes son recursivas primitivas.
  - Símbolos de función: Podemos usar símbolos para cualquier función recursiva primitiva en la lista que se está acumulando, de acuerdo a las convenciones usuales ($+, \times, \dot{-} \ldots$ y habrá más por venir).
  - Combinaciones: $\sum_{x<y}, \prod_{x<y}$, y habrá más por venir.

Ejemplo 16 II

- En efecto, estamos construyendo un determinado lenguaje de tal manera que se garantiza que cualquier función o relación definible en el lenguaje será recursiva primitiva. (Para la divisibilidad, el hecho crucial fue que la expresión "$(\exists q < y + 1)[x \cdot q = y]$" pertenecía a este lenguaje).
- Este lenguaje incluye lo siguiente:
  - Variables: Las funciones de proyección son recursivas primitivas.
  - Constantes (numerales): Las funciones constantes son recursivas primitivas.
  - Símbolos de función: Podemos usar símbolos para cualquier función recursiva primitiva en la lista que se está acumulando, de acuerdo a las convenciones usuales ($+, \times, \dot{-} \ldots$ y habrá más por venir).
  - Combinaciones: $\sum_{x<y}, \prod_{x<y}$, y habrá más por venir.
  - Símbolos de relación: Podemos usar símbolos para cualquier relación recursiva primitiva en la lista que estamos construyendo ($\leqslant, =, |, \ldots$, con más por venir).

Ejemplo 16 II

- En efecto, estamos construyendo un determinado lenguaje de tal manera que se garantiza que cualquier función o relación definible en el lenguaje será recursiva primitiva. (Para la divisibilidad, el hecho crucial fue que la expresión "$(\exists q < y + 1)[x \cdot q = y]$" pertenecía a este lenguaje).
- Este lenguaje incluye lo siguiente:
  - Variables: Las funciones de proyección son recursivas primitivas.
  - Constantes (numerales): Las funciones constantes son recursivas primitivas.
  - Símbolos de función: Podemos usar símbolos para cualquier función recursiva primitiva en la lista que se está acumulando, de acuerdo a las convenciones usuales ($+, \times, \dot{-} \ldots$ y habrá más por venir).
  - Combinaciones: $\sum_{x<y}, \prod_{x<y}$, y habrá más por venir.
  - Símbolos de relación: Podemos usar símbolos para cualquier relación recursiva primitiva en la lista que estamos construyendo ($\leqslant, =, |, \ldots$, con más por venir).
  - Más combinaciones: "no", "y", "o" se pueden aplicar a las relaciones.

Ejemplo 16 II

- En efecto, estamos construyendo un determinado lenguaje de tal manera que se garantiza que cualquier función o relación definible en el lenguaje será recursiva primitiva. (Para la divisibilidad, el hecho crucial fue que la expresión "$(\exists q < y + 1)[x \cdot q = y]$" pertenecía a este lenguaje).
- Este lenguaje incluye lo siguiente:
  - Variables: Las funciones de proyección son recursivas primitivas.
  - Constantes (numerales): Las funciones constantes son recursivas primitivas.
  - Símbolos de función: Podemos usar símbolos para cualquier función recursiva primitiva en la lista que se está acumulando, de acuerdo a las convenciones usuales ($+, \times, \dot{-} \ldots$ y habrá más por venir).
  - Combinaciones: $\sum_{x<y}, \prod_{x<y}$, y habrá más por venir.
  - Símbolos de relación: Podemos usar símbolos para cualquier relación recursiva primitiva en la lista que estamos construyendo ($\leqslant, =, |, \ldots$, con más por venir).
  - Más combinaciones: "no", "y", "o" se pueden aplicar a las relaciones.
  - "Cuantificadores" acotados: $\forall x < y$ y $\exists x < y$. (Aquí se necesita el límite superior y).

Ejemplo 16 II

- En efecto, estamos construyendo un determinado lenguaje de tal manera que se garantiza que cualquier función o relación definible en el lenguaje será recursiva primitiva. (Para la divisibilidad, el hecho crucial fue que la expresión "$(\exists q < y + 1)[x \cdot q = y]$" pertenecía a este lenguaje).
- Este lenguaje incluye lo siguiente:
  - Variables: Las funciones de proyección son recursivas primitivas.
  - Constantes (numerales): Las funciones constantes son recursivas primitivas.
  - Símbolos de función: Podemos usar símbolos para cualquier función recursiva primitiva en la lista que se está acumulando, de acuerdo a las convenciones usuales ($+, \times, \dot{-} \ldots$ y habrá más por venir).
  - Combinaciones: $\sum_{x<y}, \prod_{x<y}$, y habrá más por venir.
  - Símbolos de relación: Podemos usar símbolos para cualquier relación recursiva primitiva en la lista que estamos construyendo ($\leqslant, =, |, \ldots$, con más por venir).
  - Más combinaciones: "no", "y", "o" se pueden aplicar a las relaciones.
  - "Cuantificadores" acotados: $\forall x < y$ y $\exists x < y$. (Aquí se necesita el límite superior y).
- Tenemos teoremas que nos aseguran que las funciones o relaciones expresables en este lenguaje seguramente serán recursivas primitivas.

Ejemplo 17

- Por ejemplo, a continuación agregamos el conjunto de números primos (como una relación unaria) a nuestra lista:

Ejemplo 17

- Por ejemplo, a continuación agregamos el conjunto de números primos (como una relación unaria) a nuestra lista:
- El conjunto $\{2, 3, 5, \ldots\}$ de números naturales primos (como relación unaria sobre $\mathbb{N}$) es recursivo primitivo. Para ver esto, observe que

$$x \text{ es primo } \iff 1 < x \text{ y } (\forall u < x)(\forall v < x)[uv \neq x],$$

y el lado derecho está escrito en el lenguaje disponible para nosotros.

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.