LyC: Práctica 1 - Parte II

Tomás González

Clase basada en clases anteriores por
Herman Schinca y María Emilia Descotte

Lo que ya sabemos

- Demostrar que funciones son primitivas recursivas
  - Por recursión primitiva
  - Por composición de funciones p.r.

Lo que ya sabemos

- Demostrar que funciones son primitivas recursivas
  - Por recursión primitiva
  - Por composición de funciones p.r.
- Usar predicados primitivos recursivos
  - Funciones: alpha, negación (NOT), conjunción (AND), disyunción (OR)
  - Funciones partidas; por casos

Lo que vamos a ver

- Nuevas funciones p.r. para usar
    - Sumatoria
    - Productoria

Lo que vamos a ver

- Nuevas funciones p.r. para usar
  - Sumatoria
  - Productoria
- Nuevas codificaciones
  - Listas
  - Tuplas

Lo que vamos a ver

- Nuevas funciones p.r. para usar
    - Sumatoria
    - Productoria
- Nuevas codificaciones
    - Listas
    - Tuplas
- Nuevos predicados p.r.
    - Cuantificadores **acotados**: universal (para todo), existencial (existe)

Lo que vamos a ver

- Nuevas funciones p.r. para usar
  - Sumatoria
  - Productoria
- Nuevas codificaciones
  - Listas
  - Tuplas
- Nuevos predicados p.r.
  - Cuantificadores **acotados**: universal (para todo), existencial (existe)
- Demostrar que funciones son primitivas recursivas
  - Esquemas de recursión más complejos (por ejemplo, fibonacci)
  - Generar y probar

Nuevas funciones p.r.: Sumatorias y productioras

- Sumatoria (desde 0 o 1)

$$g(y, x_1, .., x_n) = \sum_{t=0}^{y} f(t, x_1, .., x_n)$$

Nuevas funciones p.r.: Sumatorias y productioras

- Sumatoria (desde 0 o 1)

$$g(y, x_1, .., x_n) = \sum_{t=0}^{y} f(t, x_1, .., x_n)$$

- Productoria (desde 0 o 1)

$$g(y, x_1, .., x_n) = \prod_{t=0}^{y} f(t, x_1, .., x_n)$$

Nuevas funciones p.r.: Sumatorias y productioras

- Sumatoria (desde 0 o 1)

$$g(y, x_1, .., x_n) = \sum_{t=0}^{y} f(t, x_1, .., x_n)$$

- Productoria (desde 0 o 1)

$$g(y, x_1, .., x_n) = \prod_{t=0}^{y} f(t, x_1, .., x_n)$$

- Ejercicio: Demostrar que la función que calcula el factorial de un número es p.r. (usando lo que acabamos de aprender)

Nuevas funciones p.r.: Sumatorias y productioras

- Sumatoria (desde 0 o 1)

$$g(y, x_1, .., x_n) = \sum_{t=0}^{y} f(t, x_1, .., x_n)$$

- Productoria (desde 0 o 1)

$$g(y, x_1, .., x_n) = \prod_{t=0}^{y} f(t, x_1, .., x_n)$$

- Ejercicio: Demostrar que la función que calcula el factorial de un número es p.r. (usando lo que acabamos de aprender)

- Para el hogar: Demostrar que la sumatoria y productoria desde cualquier número natural es p.r.

Nuevas codificaciones: Tuplas

- Definición

$$z = \langle x, y \rangle = 2^x(2y + 1) - 1$$

Nuevas codificaciones: Tuplas

- Definición

$$z = \langle x, y \rangle = 2^x(2y + 1) - 1$$

- Operaciones

$$l(z) = x$$

$$r(z) = x$$

Nuevas codificaciones: Tuplas

- Definición

$$z = \langle x, y \rangle = 2^x(2y+1) - 1$$

- Operaciones

$$l(z) = x$$
$$r(z) = x$$

- Ejercicio: Demostrar que $g(\langle a, \langle b, c \rangle \rangle) = b$ es p.r.

Nuevas codificaciones: Secuencias

- Definición

$$x = [a_1, .., a_n] = \prod_{i=1}^{n} primo(i)^{a_i}$$

Nuevas codificaciones: Secuencias

- Definición

$$x = [a_1, .., a_n] = \prod_{i=1}^{n} primo(i)^{a_i}$$

- Operaciones

$$|x| = n$$

$$x[i] = a_i$$

Nuevas codificaciones: Secuencias

- Definición

$$x = [a_1, .., a_n] = \prod_{i=1}^{n} primo(i)^{a_i}$$

- Operaciones

$$|x| = n$$

$$x[i] = a_i$$

- Ejercicio: Demostrar que el producto escalar de dos vectores (representados por listas de la misma longitud)

Ejercicio

Ejercicio 1. *Demostrar que la siguiente función* $f : \mathbb{N} \to \mathbb{N}$ *es primitiva recursiva:*

$$\begin{aligned} f(0) &= 1 \\ f(1) &= 2 \\ f(2) &= 4 \\ f(n+3) &= f(n) + f(n+1)^2 + f(n+2)^3 \end{aligned}$$

¿Qué hicimos?

- Identificamos el esquema de recursión de *f*
  - Casos base
  - Caso recursivo
    - Cuantos pasos anteriores usa
    - **Cuales** pasos anteriores usa

¿Qué hicimos?

- Identificamos el esquema de recursión de *f*
    - Casos base
    - Caso recursivo
        - Cuantos pasos anteriores usa
        - **Cuales** pasos anteriores usa
- Definimos una nueva función *g* que guarda los pasos anteriores que necesitamos en una secuencia. (*f(x1,..,xn,t)* va a pertenecer a la secuencia)

¿Qué hicimos?

- Identificamos el esquema de recursión de *f*
    - Casos base
    - Caso recursivo
        - Cuantos pasos anteriores usa
        - **Cuales** pasos anteriores usa
- Definimos una nueva función *g* que guarda los pasos anteriores que necesitamos en una secuencia. (*f(x1,..,xn,t)* va a pertenecer a la secuencia)
- Probamos que *g* es p.r. usando recursión primitiva

¿Qué hicimos?

- Identificamos el esquema de recursión de *f*
    - Casos base
    - Caso recursivo
        - Cuantos pasos anteriores usa
        - **Cuales** pasos anteriores usa
- Definimos una nueva función *g* que guarda los pasos anteriores que necesitamos en una secuencia. (*f(x1,..,xn,t)* va a pertenecer a la secuencia)
- Probamos que *g* es p.r. usando recursión primitiva
- Vemos cómo obtener *f* de *g* (acceso a listas)

¿Qué hicimos?

- Identificamos el esquema de recursión de *f*
  - Casos base
  - Caso recursivo
    - Cuantos pasos anteriores usa
    - **Cuales** pasos anteriores usa
- Definimos una nueva función *g* que guarda los pasos anteriores que necesitamos en una secuencia. (*f(x1,..,xn,t)* va a pertenecer a la secuencia)
- Probamos que *g* es p.r. usando recursión primitiva
- Vemos cómo obtener *f* de *g* (acceso a listas)
- *f* es p.r. pues es una composición del acceso a listas con *g*

Nuevos predicados p.r.

- Cuantificador universal **acotado**

$$(\forall t)_{\leq y}\ p(t, x_1, .., x_n)$$

Nuevos predicados p.r.

- Cuantificador universal **acotado**

$$(\forall t)_{\leq y}\ p(t, x_1, .., x_n)$$

- Cuantificador existencial **acotado**

$$(\exists t)_{\leq y}\ p(t, x_1, .., x_n)$$

Nuevos predicados p.r.

- Cuantificador universal **acotado**

$$(\forall t)_{\leq y}\ p(t, x_1, .., x_n)$$

- Cuantificador existencial **acotado**

$$(\exists t)_{\leq y}\ p(t, x_1, .., x_n)$$

- Minimización **acotada**

$$\min_{t \leq y} p(t, x_1, .., x_n)$$

Nuevos predicados p.r.

- Cuantificador universal **acotado**

$$(\forall t)_{\leq y}\ p(t, x_1, .., x_n)$$

- Cuantificador existencial **acotado**

$$(\exists t)_{\leq y}\ p(t, x_1, .., x_n)$$

- Minimización **acotada**

$$\min_{t \leq y}\ p(t, x_1, .., x_n)$$

- Para el Hogar: hacer el ejercicio 7 de la práctica

Ejercicios con cuantificadores

- Demostrar que las siguientes funciones son p.r.
  - pertenece(L , x): es 1 si x pertenece a la lista L, 0 si no

Ejercicios con cuantificadores

- Demostrar que las siguientes funciones son p.r.
  - pertenece(L , x): es 1 si x pertenece a la lista L, 0 si no
  - índice(L , x): devuelve el índice de x en la lista L
    - Si x no está en la lista, devuelve 0
    - Si x está varias veces en la lista, devuelve cualquiera de los índices en los que se encuentra x

Ejercicios con cuantificadores

- Demostrar que las siguientes funciones son p.r.
  - pertenece(L , x): es 1 si x pertenece a la lista L, 0 si no
  - índice(L , x): devuelve el índice de x en la lista L
    - Si x no está en la lista, devuelve 0
    - Si x está varias veces en la lista, devuelve cualquiera de los índices en los que se encuentra x

- ¿Qué estamos haciendo en estas funciones?

Ejercicios con cuantificadores

- Demostrar que las siguientes funciones son p.r.
  - pertenece(L , x): es 1 si x pertenece a la lista L, 0 si no
  - índice(L , x): devuelve el índice de x en la lista L
    - Si x no está en la lista, devuelve 0
    - Si x está varias veces en la lista, devuelve cualquiera de los índices en los que se encuentra x

- ¿Qué estamos haciendo en estas funciones?
  - Establecemos un conjunto de posibles soluciones (cota superior)

Ejercicios con cuantificadores

- Demostrar que las siguientes funciones son p.r.
  - pertenece(L , x): es 1 si x pertenece a la lista L, 0 si no
  - índice(L , x): devuelve el índice de x en la lista L
    - Si x no está en la lista, devuelve 0
    - Si x está varias veces en la lista, devuelve cualquiera de los índices en los que se encuentra x

- ¿Qué estamos haciendo en estas funciones?
  - Establecemos un conjunto de posibles soluciones (cota superior)
  - Probamos las soluciones posibles (predicados) y devolvamos la primer solución que cumpla lo pedido

Ejercicio: Generar y probar

Ejercicio 5. *Una lista l de pares de naturales se llama camino de longitud n que comienza en a_0 si $l = [\langle a_0, a_1\rangle, \langle a_1, a_2\rangle, ..., \langle a_{n-2}, a_{n-1}\rangle, \langle a_{n-1}, a_n\rangle]$. Sea $f : \mathbb{N}^3 \to \mathbb{N}$ una función tal que $f(l, a_0, n)$ devuelve el camino más largo de longitud menor o igual a n que comienza en a_0 y tal que todos los elementos del camino (es decir, los pares de naturales) son elementos de la lista l. Si no existe tal camino, la función debe devolver cero; si existe más de uno, debe devolver cualquiera de ellos. Demostrar que f es primitiva recursiva (por simplicidad, se considera que el par $\langle 0, 0\rangle \notin l$).*

¿Qué hicimos? Generate + Test

1. Identificamos si ibamos a usar un existencial o minimización

¿Qué hicimos? Generate + Test

1. Identificamos si ibamos a usar un existencial o minimización
    - Existencial si me basta con saber si existe la solución que quiero
    - Minimización si quiero esa solución

¿Qué hicimos? Generate + Test

1. Identificamos si íbamos a usar un existencial o minimización
   - Existencial si me basta con saber si existe la solución que quiero
   - Minimización si quiero esa solución
2. Identificamos el conjunto de las posibles soluciones

¿Qué hicimos? Generate + Test

1. Identificamos si íbamos a usar un existencial o minimización
   - Existencial si me basta con saber si existe la solución que quiero
   - Minimización si quiero esa solución
2. Identificamos el conjunto de las posibles soluciones
3. Acotamos el conjunto de las posibles soluciones

¿Qué hicimos? Generate + Test

1. Identificamos si íbamos a usar un existencial o minimización
   - Existencial si me basta con saber si existe la solución que quiero
   - Minimización si quiero esa solución
2. Identificamos el conjunto de las posibles soluciones
3. Acotamos el conjunto de las posibles soluciones
4. Identificamos las condiciones que tiene que cumplir la solución (en español)

¿Qué hicimos? Generate + Test

1. Identificamos si íbamos a usar un existencial o minimización
   - Existencial si me basta con saber si existe la solución que quiero
   - Minimización si quiero esa solución
2. Identificamos el conjunto de las posibles soluciones
3. Acotamos el conjunto de las posibles soluciones
4. Identificamos las condiciones que tiene que cumplir la solución (en español)
5. Tradujimos las condiciones del español a un predicado primitivo recursivo con el cuantificador que identificamos en el paso 1.

¿Qué hicimos? Generate + Test

1. Identificamos si íbamos a usar un existencial o minimización
   - Existencial si me basta con saber si existe la solución que quiero
   - Minimización si quiero esa solución
2. Identificamos el conjunto de las posibles soluciones
3. Acotamos el conjunto de las posibles soluciones
4. Identificamos las condiciones que tiene que cumplir la solución (en español)
5. Tradujimos las condiciones del español a un predicado primitivo recursivo con el cuantificador que identificamos en el paso 1.
6. Usamos minimización acotada
   - La cota es el elemento máximo del conjunto que acotamos en el paso 2
   - El predicado es la traducción de las condiciones que realizamos en el paso 5

Ejercicio: Esquema de recursión

Ejercicio 3. *Sea* $k \in \mathbb{N}$, $f : \mathbb{N} \to \mathbb{N}$ *una función tal que* $f(x+1) < x+1$ *para todo* x, *y sea* $h : \mathbb{N}^2 \to \mathbb{N}$ *definida como*

$$\begin{aligned} h(x,0) &= k \\ h(x,t+1) &= g(x,t,h(x,f(t+1))) \end{aligned}$$

Demostrar que si f *y* g *pertenecen a una clase PRC* $\mathcal{C}$, *entonces también* h *pertenece a* $\mathcal{C}$.

Fin

¡Consulten! ¡Hagan los ejercicios!