

Prueba de oposición 2017: Probar que cierto predicado es primitivo recursivo

(Sin marearse)

Guillermo Mosse

Contexto

- Ejercicio 6 de la práctica 2.
- Los chicos ya conocen bastante a las funciones primitivas recursivas por la práctica 1.
- Hacer este tipo de ejercicios en forma concisa y completa tiene sus dificultades.
- Suele ser un tipo de ejercicio que se toma en un parcial.

Enunciado

Ejercicio: Un programa P en el lenguaje S con instrucciones I_1, \dots, I_n se dice optimista si $\forall i = 1, \dots, n$, si I_i es la instrucción 'IF $V \neq 0$ GOTO L' entonces L no aparece como etiqueta de ninguna instrucción I_j con $j \leq i$.

Demostrar que el siguiente predicado es primitivo recursivo:

$$r(x) = \begin{cases} 1 & \text{si el programa cuyo número es } x \text{ es optimista} \\ 0 & \text{caso contrario} \end{cases}$$

Entendamos el enunciado

Queremos ver que el predicado $r(x)$ es primitivo recursivo.

¿Cómo se prueba algo así?

Entendamos el enunciado

Queremos ver que el predicado $r(x)$ es primitivo recursivo.

¿Cómo se prueba algo así?

Recuerdo: una función es primitiva recursiva (p.r.) si se puede obtener a partir de las funciones iniciales por un número finito de aplicaciones de composición y recursión primitiva.

Entendamos el enunciado

Queremos ver que el predicado $r(x)$ es primitivo recursivo.

¿Cómo se prueba algo así?

Recuerdo: una función es primitiva recursiva (p.r.) si se puede obtener a partir de las funciones iniciales por un número finito de aplicaciones de composición y recursión primitiva.

¿Qué herramientas tenemos a nuestra disposición?:

Entendamos el enunciado

Queremos ver que el predicado $r(x)$ es primitivo recursivo.

¿Cómo se prueba algo así?

Recuerdo: una función es primitiva recursiva (p.r.) si se puede obtener a partir de las funciones iniciales por un número finito de aplicaciones de composición y recursión primitiva.

¿Qué herramientas tenemos a nuestra disposición?:

- El conjunto de predicados p.r. es cerrado por operadores lógicos (\neg, \vee, \wedge)
- El conjunto de funciones p.r. es cerrado por definición por casos
- Y por cuantificadores y minimizadores acotados
- También puedo usar codificación y decodificación de pares y secuencias, y longitud de secuencias
- Operaciones básicas (¡jojo! la resta no está, hay que usar resta con punto)
- Los operadores $\leqslant, \geqslant, =, \neq$, etc

¿Cómo se codificaba el número de programa?

Recuerdo: ¿cómo se codifican los programas?

¿Cómo se codificaba el número de programa?

Recuerdo: ¿cómo se codifican los programas?

Ordenamos las variables, asignándole un número a cada una. (Este orden está fijo, no depende del programa) Lo mismo para sus etiquetas.

¿Cómo se codificaba el número de programa?

Recuerdo: ¿cómo se codifican los programas?

Ordenamos las variables, asignándole un número a cada una. (Este orden está fijo, no depende del programa) Lo mismo para sus etiquetas.

Codificamos a la instrucción I con $\#(I) = < \mathbf{a}, < \mathbf{b}, \mathbf{c} >>$ donde:

- 1 Si I tiene etiqueta L entonces $\mathbf{a} = \#(L)$, si no $\mathbf{a} = 0$.
- 2 Si la variable mencionada en I es V entonces $\mathbf{c} = \#(V) - 1$
- 3 Si la instrucción I es:
 - 3.1 $V \leftarrow V$ entonces $\mathbf{b} = 0$
 - 3.2 $V \leftarrow V + 1$ entonces $\mathbf{b} = 1$
 - 3.3 $V \leftarrow V - 1$ entonces $\mathbf{b} = 2$
 - 3.4 $IF\ V \neq 0\ GOTO\ L'$ entonces $\mathbf{b} = \#(L') + 2$

Si P es un programa con instrucciones I_1, \dots, I_n , su codificación va a ser $\#(P) = [\#(I_1), \dots, \#(I_n)] - 1$

Empecemos

La idea del ejercicio va a ser partir al predicado $r(x)$ en funciones que sabemos que son p.r. usando que la clase de funciones p.r. es cerrada por ciertos esquemas.

Quiero expresar el siguiente predicado de manera p.r.: dado un programa con número de programa x ,

$$r(x) = \begin{cases} 1 & \text{si el programa cuyo n\'umero es } x \text{ es optimista} \\ 0 & \text{caso contrario} \end{cases}$$

Empecemos en lenguaje (medio) coloquial

Un programa con número x es optimista si:

$$\forall i \leq n,$$

[Si I_i "es" 'IF $V \neq 0$ GOTO L' $\Rightarrow \forall j \leq i, I_j$ no tiene la etiqueta L],

donde n es la cantidad de líneas del programa x .

Vamos transformando la expresión...

Un programa con número x es optimista si:

$$\forall i \leq |x + 1|,$$

[Si I_i "es" 'IF $V \neq 0$ GOTO L' $\Rightarrow \forall j \leq i$, I_j no tiene la etiqueta L]

Partámosla en trozos más manejables

Un programa con número x es optimista si:

$$\forall i \leq |x + 1|,$$

[*Si I_i "es" 'IF $V \neq 0$ GOTO L' $\Rightarrow \forall j \leq i$, I_j no tiene la etiqueta L*]

Primera parte

I_i "es" 'IF $V \neq 0$ GOTO L' :

El número de programa es x . Expresemos este predicado de manera p.r.:

$$p(i, x) = \begin{cases} 1 & \text{si } I_i \text{ "es" 'IF } V \neq 0 \text{ GOTO } L' \\ 0 & \text{caso contrario} \end{cases}$$

Primera parte

I_i "es" 'IF V ≠ 0 GOTO L':

El número de programa es x. Expresemos este predicado de manera p.r.:

$$p(i, x) = \begin{cases} 1 & \text{si } (x + 1)[i] \text{ "es" 'IF V } \neq 0 \text{ GOTO L}' \\ 0 & \text{caso contrario} \end{cases}$$

Primera parte

I_i "es" 'IF $V \neq 0$ GOTO L' :

Recordemos que una instrucción es de la pinta $\langle a, \langle b, c \rangle \rangle$, donde a tiene información de la etiqueta, b del tipo de instrucción, y c de la variable mencionada.

Luego queremos que, si $I_i = \langle a, \langle b, c \rangle \rangle$, entonces $b \geq 3$. Queda:

$$p(i, x) = \begin{cases} 1 & \text{si } I(r((x+1)[i])) \geq 3 \\ 0 & \text{caso contrario} \end{cases}$$

Segunda parte

$\forall j \leq i, l_j$ no tiene la etiqueta L:

Segunda parte

$\forall j \leq i, l_j$ no tiene la etiqueta L:

$$n(i, j, x) = \begin{cases} 1 & \text{si } \forall j \leq i, (x+1)[j] \text{ no tiene la etiqueta L} \\ 0 & \text{caso contrario} \end{cases}$$

Segunda parte

$\forall j \leq i, l_j$ no tiene la etiqueta L:

$$n(x, i, j) = \begin{cases} 1 & \text{si } \forall j \leq i, [I((x+1)[j]) \neq I(r((x+1)[i])) - 2] \\ 0 & \text{caso contrario} \end{cases}$$

Segunda parte

$\forall j \leq i, l_j$ no tiene la etiqueta L:

$$n(x, i, j) = \begin{cases} 1 & \text{si } \forall j \leq i, [I((x + 1)[j]) \neq I(r((x + 1)[i])) - 2] \\ 0 & \text{caso contrario} \end{cases}$$

porque si l_i es un GOTO, $I(r((x + 1)[i])) - 2$ es la etiqueta L, y,
OJO, la resta con punto es p.r., la resta no.

Todo

Unamos todo. Queda:

$$r(x) = \begin{cases} 1 & \text{si } \forall i \leq (|x+1|), [I(r((x+1)[i]) \geq 3) \Rightarrow (\forall j \leq i ([I((x+1)[j]) \neq I(r((x+1)[i])) \div 2])] \\ 0 & \text{caso contrario} \end{cases}$$

Todo

Unamos todo. Queda:

$$r(x) = \begin{cases} 1 & \text{si } \forall i \leq (|x+1|), [I(r((x+1)[i]) \geq 3) \Rightarrow (\forall j \leq i ([I((x+1)[j]) \neq I(r((x+1)[i])) \div 2)])] \\ 0 & \text{caso contrario} \end{cases}$$

Tenemos que el predicado $p \Rightarrow n$ es equivalente a $n \vee \neg p$

Todo

Unamos todo. Queda:

$$r(x) = \begin{cases} 1 & \text{si } \forall i \leq (|x+1|), [I(r((x+1)[i]) \geq 3) \Rightarrow (\forall j \leq i ([I((x+1)[j]) \neq I(r((x+1)[i])) \div 2])] \\ 0 & \text{caso contrario} \end{cases}$$

Tenemos que el predicado $p \Rightarrow n$ es equivalente a $n \vee \neg p$

Luego puedo expresar $r(x)$ así:

Todo

Unamos todo. Queda:

$$r(x) = \begin{cases} 1 & \text{si } \forall i \leq (|x+1|), [(\textcolor{red}{I(r((x+1)[i]) \geq 3)} \Rightarrow (\forall j \leq i ([I((x+1)[j]) \neq I(r((x+1)[i])) - 2)] \\ 0 & \text{caso contrario} \end{cases}$$

Tenemos que el predicado $p \Rightarrow n$ es equivalente a $n \vee \neg p$

Luego puedo expresar $r(x)$ así:

$$r(x) = \begin{cases} 1 & \text{si } \forall i \leq (|x+1|), [(\forall j \leq i (I(x+1)[j] \neq I(r((x+1)[i])) - 2)) \vee \neg (\textcolor{red}{I(r((x+1)[i]) \geq 3)})] \\ 0 & \text{caso contrario} \end{cases}$$

,

Todo

Unamos todo. Queda:

$$r(x) = \begin{cases} 1 & \text{si } \forall i \leq (|x + 1|), [(\mathcal{I}(r((x + 1)[i]) \geq 3) \Rightarrow (\forall j \leq i ([\mathcal{I}((x + 1)[j]) \neq \mathcal{I}(r((x + 1)[i])) \div 2)])] \\ 0 & \text{caso contrario} \end{cases}$$

Tenemos que el predicado $p \Rightarrow n$ es equivalente a $n \vee \neg p$

Luego puedo expresar $r(x)$ así:

$$r(x) = \begin{cases} 1 & \text{si } \forall i \leq (|x + 1|), [(\forall j \leq i (\mathcal{I}(x + 1)[j] \neq \mathcal{I}(r((x + 1)[i])) \div 2)) \vee \neg (\mathcal{I}(r((x + 1)[i]) \geq 3))] \\ 0 & \text{caso contrario} \end{cases}$$

,

que es p.r. por estar definido por casos por funciones p.r., donde la segunda función es la constante cero y la primera está formada por \neg , \vee , dos cuantificadores acotados, la longitud de una secuencia, observadores de secuencias, el predicado \leq y las operaciones elementales suma y resta con punto, que sabemos que son p.r.

Es parecido:

Ejercicio de parcial: Decimos que un programa es "vueltero" si en algún lugar de su código aparece la siguiente secuencia de instrucciones:

$$V \leftarrow V + 1$$

[L] IF $V \neq 0$ GOTO L

para alguna variable V cualquiera y alguna etiqueta L que no aparece en las líneas anteriores del código del programa. Demostrar que el conjunto $\{x \mid x \text{ es el número de un programa "vueltero"}\}$ es p.r.

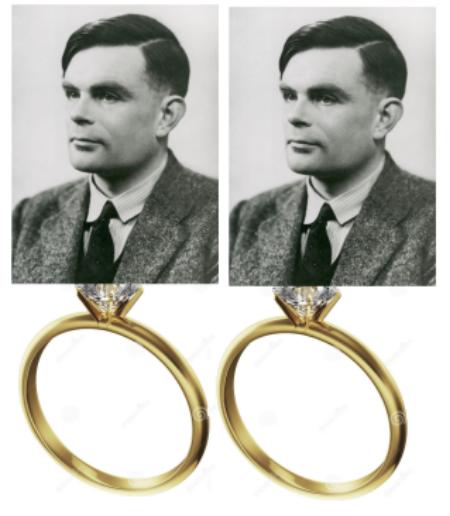
¿Preguntas?

¡Gracias por escuchar!

¿Alguna pregunta?



(a) 1 ring



(b) Tu-rings

Mini apéndice

$$x \div y = \begin{cases} x - y & \text{si } y \leq x \\ 0 & \text{si } y > x \end{cases}$$

Codif de pares: $\langle x, y \rangle = 2^x(2 \cdot y + 1) \div 1$

Decodif (izq) de pares: $l(z) = \min_{x \leq z}((\exists y)_{\leq z} z = \langle x, y \rangle)$

Decodif (der) de pares: $r(z) = \min_{y \leq z}((\exists x)_{\leq z} z = \langle x, y \rangle)$

Codif de secuencias: $[a_1, \dots, a_n] = \prod_{i=1}^n p_i^{a_i}$, donde p_i es el i-ésimo primo.

Decodif de secuencias: $x[i] = \min_{t \leq x}(\neg p_i^{t+1} | x)$

Tamaño de secuencias: $|x| = \min_{i \leq x}(x[i] \neq 0 \wedge (\forall j)_{\leq x}(j \leq i \vee x[j] = 0))$ 17

Bibliografía

- Teóricas de LyC, diapositivas 39-51, 75-78
- Práctica 1 LyC
- Parcial de Computabilidad del 2do cuatrimestre del 2015 ([Link](#))