Concurso Area Teoría

Brian Bokser

Contexto

- Los alumnos están familiarizados con el lenguaje S
- Los alumnos tuvieron la teórica de codificación de programas y diagonalización
- El ejercicio podría ser un ejercicio de parcial

Enunciado

Un programa P en el lenguaje S se dice optimista si $\forall$ i $= 1, \ldots,$ 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

Sea $r(x) = \begin{cases} 1 & \text{si el programa de numero x es optimista} \\ 0 & \text{si no} \end{cases}$

Demostrar que el predicado r(x) es primitivo recursivo.

Motivación

- Entender la codificación de programas (programas como números)
- Practicar conceptos de recursividad primitiva
- Dividir en subproblemas

Metodología

¡Idea Importante!

Resolución **Top-Down**: *Dividir el problema en* **subproblemas**

Metodología

¡Idea Importante!

Resolución **Top-Down**: *Dividir el problema en* **subproblemas**

Los subproblemas son definiciones del enunciado.

Metodología

¡Idea Importante!

Resolución **Top-Down**: *Dividir el problema en* **subproblemas**

Los subproblemas son definiciones del enunciado.

Si el enunciado nos habla de instrucciones válidas, saber identificar una de estas puede ser un subproblema

Metodología

¡Idea Importante!

Resolución **Top-Down**: *Dividir el problema en* **subproblemas**

Los subproblemas son definiciones del enunciado.

Si el enunciado nos habla de instrucciones válidas, saber identificar una de estas puede ser un subproblema

Tratar de resolver cada subproblema "en un linea"

Codificación

Codificación de instrucciones de $\mathcal{S}$

Codificamos a la instrucción I con

$$\#(I) = \langle a, \langle b, c \rangle \rangle$$

donde

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

Por ejemplo,

- $\#(X \leftarrow X + 1) = \langle 0, \langle 1, 1 \rangle \rangle = \langle 0, 5 \rangle = 10$
- $\#([A] \quad X \leftarrow X + 1) = \langle 1, \langle 1, 1 \rangle \rangle = \langle 1, 5 \rangle = 21$
- $\#(\text{IF } X \neq 0 \text{ GOTO } A) = \langle 0, \langle 3, 1 \rangle \rangle = \langle 0, 23 \rangle = 46$
- $\#(Y \leftarrow Y) = \langle 0, \langle 0, 0 \rangle \rangle = \langle 0, 0 \rangle = 0$

Todo número x representa a una única instrucción I.

Codificación de programas en $\mathcal{S}$

Un programa P es una lista (finita) de instrucciones $I_1, \ldots, I_k$

Codificamos al programa P con

$$\#(P) = [\#(I_1), \ldots, \#(I_k)] - 1$$

Por ejemplo, para el programa P

$$\begin{array}{ll} [A] & X \leftarrow X + 1 \\ & \text{IF } X \neq 0 \text{ GOTO } A \end{array}$$

tenemos

$$\#(P) = [\#(I_1), \#(I_2)] - 1 = [21, 46] - 1 = 2^{21} \cdot 3^{46} - 1$$

Primer problema: ¿Qué es r?

Primer problema: ¿Qué es r?

$$\text{Es optimista} \equiv \text{No hay una instruccion que sea inválida}$$

Primer problema: ¿Qué es r?

Es optimista $\equiv$ No hay una instruccion que sea inválida

$\mathsf{r}(\mathsf{x}) \equiv \alpha(\exists_{i \leq |x+1|}(\mathit{esSaltoAtras}(i, x+1)))$

Primer problema: ¿Qué es r?

Es optimista $\equiv$ No hay una instruccion que sea inválida

$\mathsf{r}(\mathsf{x}) \equiv \alpha(\exists_{i \leq |x+1|}(\mathit{esSaltoAtras}(i, x+1)))$

válida $\equiv$ Si tengo un if, no salta a una etiqueta anterior.

Primer problema: ¿Qué es r?

Es optimista $\equiv$ No hay una instruccion que sea inválida

$r(x) \equiv \alpha(\exists_{i \leq |x+1|}(esSaltoAtras(i, x+1)))$

válida $\equiv$ Si tengo un if, no salta a una etiqueta anterior.

esSaltoAtras(i, x) $\equiv$
$esUnIf(i, x) \implies \exists_{j \leq i}(etiqueta(j, x) = etiquetaDelf(x[i]))$

Es un if si $a = \#(L) + 2$ en $< a, < b, c >>$

Es un if si $a = \#(L) + 2$ en $< a, < b, c >>$

esUnIf(i, x) $\equiv l(r(x)) > 2$

Es un if si $a = \#(L) + 2$ en $< a, < b, c >>$

esUnIf(i, x) $\equiv l(r(x)) > 2$

La etiqueta es la primer parte de la tripla

Es un if si $a = \#(L) + 2$ en $< a, < b, c >>$

esUnIf(i, x) $\equiv l(r(x)) > 2$

La etiqueta es la primer parte de la tripla

etiqueta(j, x) $\equiv l(x[j])$

Es un if si $a = \#(L) + 2$ en $< a, < b, c >>$

esUnIf(i, x) $\equiv l(r(x)) > 2$

La etiqueta es la primer parte de la tripla

etiqueta(j, x) $\equiv l(x[j])$

Etiqueta de un if es la segunda parte menos dos

Es un if si $a = \#(L) + 2$ en $< a, < b, c >>$

esUnIf(i, x) $\equiv l(r(x)) > 2$

La etiqueta es la primer parte de la tripla

etiqueta(j, x) $\equiv l(x[j])$

Etiqueta de un if es la segunda parte menos dos

etiquetaDeIf(x) $\equiv l(r(x)) - 2$

Conclusiones

r(x) es primitivo recursivo pues es composición de p.r.

El existencial **acotado** es p.r.

Los observadores de lista y pares son p.r.

No olvidarse de mencionar esto en la justificación

¿Preguntas?