

①

⑩ Elementos de la función RNF

- Tupla de parámetros formales $\bar{x} \equiv x, n, i$
- Caso Base? $(\bar{x}) \equiv i=0$
- Sol(x, \cancel{n}, i) $\equiv x[i]$
- Suc(x, n, i) $\equiv x, n, i+1$
- Función combinación $\text{Comb}(x, n, i, f) \equiv x[i] + f$

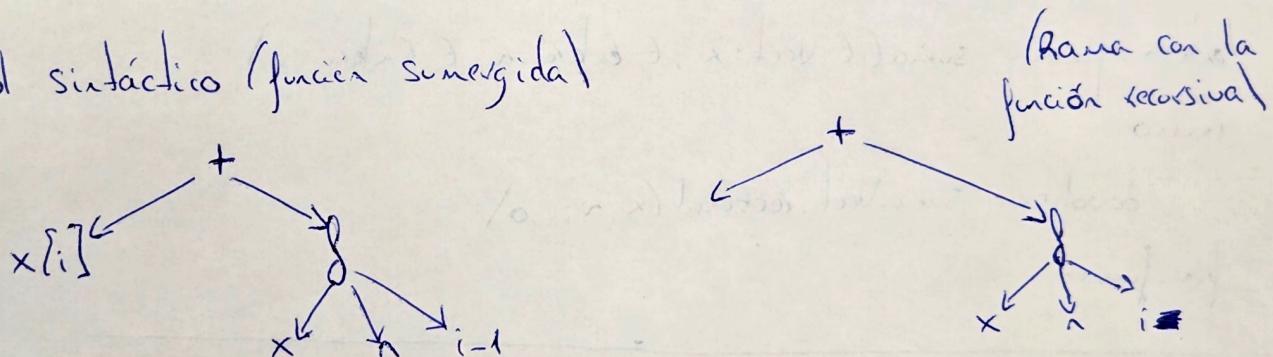
RNF a FRF

a) Generalización:

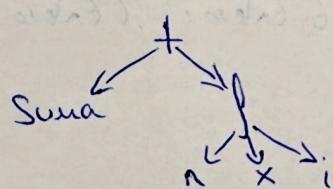
- Función Sumergida:

$$\text{sumaVect_rec}(x, n, i) = x[i] + \text{sumaVect_rec}(x, n, i-1)$$

- Árbol sintáctico (función sumergida)



- Árbol sintáctico con los parámetros de invocación.



- Función Inversora:

sumaVect-vec-Final ($x, n, i, suma$) = suma + sumaVect-vec (x, n, i)

- Valores Iniciales:

Elemento neutro de la suma = 0

- Llamada Inicial:

sumaVect-vecFinal ($x, n, i, 0$) = 0 + sumaVect-vec (x, n, i)

- Función que realiza la llamada a la función RF:

// Cabecera: entero: suma (E Vect: x, E Entero: n, E Entero: i)

// Precondición: $x = A[1..n] \wedge n > 0 \wedge 0 \leq i \leq n$

// Postcondición: devolver el valor $\sum_{\alpha=1}^i x[\alpha]$

entro: función suma (E Vect: x, E Entero: n, E Entero: i)

inicio

devolver sumaVect-vecFinal ($x, n, i, 0$)

fin-función

b) Desplegado:

- Diseño de Función Inversora:

// Cabecera: entero: sumaVect-vecFinal (E Vector: x, E Entero: n, E Entero: i, E Entero: suma)

// Precondiciones: $x = A[1..N] \wedge n > 0 \wedge 0 \leq i \leq n$

// Postcondiciones: devolver el valor: $\sum_{\alpha=1}^i x[\alpha]$

entro: sumaVect-vec (E Vector: x, E Entero: n, E Entero: i, E Entero: suma)

inicio

si $i = 0$ entonces

devolver suma + 0

si $\neg 0$

devolver suma + $x[i] + sumaVect-vec(x, n, i - 1)$

fin-si

fin-función.

②

- Reorganización del Caso General:

$$(suma + x[i]) + sumaVect_vec(x, n, i-1)$$

- Función RF (aún depende a la función RNF):

// Cabeza: E Vector: x, E Entero: n, E Entero: i, E Entero: suma

// Precondición: la misma

// Postcondición: la misma.

externo: función sumaVect_rectifinal (E Vector: x, E Entero: n, E Entero: i, E Entero: suma)

inicio

si $i=0$ entonces:

devolver suma + 0

si - no

devolver $(suma + x[i]) + sumaVect_vec(x, n, i-1)$

fin-si

fin-función

c) Plegado:

- Sucesores:

$$suc(suma) \equiv suma + x[i]$$

- Versión Recursiva Final:

// Cabeza: externo: sumaVect_rectifinal (E Vector: x, E Entero: n, E Entero: i, E Entero: suma)

// Precondición: la misma

// Postcondición: La misma

externo: función ~~Cabeza~~

inicio

si $i=0$ entonces

devolver suma + 0

si - no

devolver sumaVect_rectifinal (x, n, i-1, suma + x[i])

fin-si

fin-función.

FRNF a F.I.

- Transformación de la FRNF a F.I.

// Calcular: entero : sumaVector (E Vector: x, E Entero: n, E Entero: i)

// Precondic.: la misma

// Postcond.: la misma

entero : función ~~Calcular~~

var

entero : C, res

inicio

i < 0

mientras $\neg(i = 0)$ hacer

C \leftarrow C + 1

i \leftarrow i - 1

fin-mientras

res \leftarrow 0

mientras C \neq 0 hacer

C \leftarrow C - 1

i \leftarrow i + 1

res \leftarrow res + x[i]

fin-mientras

devolver res

fin-función.

FRF a F.I.

- Transformación función R.F. a R.I.

// Cabecera: entero : suma (E vect: x, E Entero: n, E: Entero: i)

// Recorrel.: la misma

// Postcond.: La misma

externo: función ~~**~~ Cabecera ~~**~~

inicio

declarar sumaVectIter (x, n, i, o)

fin-función

// Cabecera: entero : sumaVectIter (E vect: x, E Entero: n, E: Entero: i, EEntero: suma)

// Recorrel.: la misma

// Postcond.: La misma

externo: función! ~~**~~ Cabecera ~~**~~

inicio

mientras $i \neq 0$ hacer

suma = suma + x[i]

$i = i - 1$

fin-mientras

declarar suma + 0

fin-función.

- optimización del código:

// Cabeza: entero sumavectIter(E vector: x, E Entero: n, E Entero: i)
// Precondición: la misma
// Postcond.: la misma
externo función *** Cabeza ***

Var

entro: C, res

inicio

c <= n - i

i <= 0

res <= 0

mientras c >= 0 hacer:

c <= c - 1

i <= i + 1

res <= res + x[i]

fin-mientras

deshacer res

fin-función.

④

- Optimización del código.

// Cabeza: entro: sumaVectIter(EVect<X>, EEntero<n>, EEntero<i>, EEntero<suma>)

// Precond.: la misma

// Postcond.: la misma

externo: función, & Cabeza & *

inicio

suma \leftarrow 0

i \leftarrow n - 1

mientras $\neg (i = 0)$ hacer

 suma = suma + x[i]

 i = i - 1

fin-mientras

dejar x suma + 0

fin-función.