13/Sep/2020

Práctica 2 -

TADs con comportamiento automático

Pablo Barenbaum

Impresionante despliegue del uso de vim para la escritura de TADs

Recursión sobre listas

extender secuencia con "primeros"

(mio)

$$primeros : nat\ n \times secu(\alpha)\ s \to secu(\alpha) \quad \{n \leq long(s)\}$$

$$primeros(3, [2,3,7,8]) \equiv [2,3,7]$$

$$prim(0, s) \equiv \langle\rangle$$

$$prim(n, a \cdot s) \equiv a \cdot prim(n-1, s)$$

(pablo)

```
primeros (n,s) ≡
        if n =0?
           then <>
        else
           prim(s) • primeros(n-1, fin(s))
```

Subcadenas

```
subcadenas : secu(α) x nat → conj(secu(α))
subcadenas (n, s) ≡
    if long(s) < n then
        ∅
    else
        Ag(primeros(n, s), subcadenas(n, fin(s)))
```

si n=0: "Agrega" muchas veces la secu. vacía

Subsecuencias

+ no necesariamente contiguas

```
subsec : secu(α) x nat → conj(secu(α))
subsec ≡
if long(s) < n then
    ∅
else if n =0? then
        Ag(<>, ∅)
    else
        agregar Ad(prim(s), subsecuencias(fin(s), n-1))
        ∪ subsec(fin(s), n)
    fi
fi
```

agregarAd : $\alpha \times conj(secu(\alpha)) \rightarrow conj(secu(\alpha))$

agregarAd $(a, c) \equiv$

if $\emptyset?(c)$ then

$\emptyset$

else

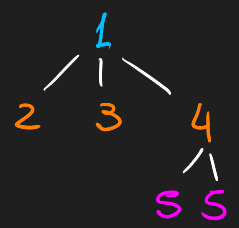
Ag ($a \bullet$ dameUno(c) , agregarAd $(a$, sinUno $(c))$)

(dameUno: una secu)

fi

Parte 2 - Recursión sobre árboles

TAD Rosetree




```
rose(1, [
  rose(2, []),
  rose(3, []),
  rose(4, [
    rose(5, []),
    rose(5, [])
  ])
])
```

incrementar : rosetree(nat) $\times$ nat $\rightarrow$ rosetree(nat)

incrementar$(r, n) \equiv$ rose (raiz$(r) + n$, incrementarMuchos (hijos(r) , n))

```
incrementarMuchos : secu(roset(nat)) × nat → secu(roset(nat))
incrementarMuchos(s, n) ≡
    if vacía?(s) then
        <>
    else
        incrementar(prim(s), n) • incr.Muchos(fin(s), n)
    fi
```

Suma de niveles

```
sumaDeNiveles : roset(nat) → secu(nat)
sumaDeNiveles(r) ≡ raíz(r) • sumaNivelesDeMuchos(hijos(r))

sumaNiveDeMuchos : secu(roset(nat)) → secu(nat)
sumaNiveDeMuchos(s) ≡
    if vacía?(s) then
        <>
    else
        sumaListas(
            sumaDeNiveles(prim(s)) •
            • sumaNiveDeMuchos(fin(s))
        )
    fi
```

```
sumaListas : secu(nat) × secu(nat) → secu(nat)
```

```
sumaListas(s, t) ≡
  if vacía?(s) then
    t
  else
    if vacía?(t) then
      s
    else
      (prim(s) + prim(t)) • sumaListas(fin(s), fin(t))
    fi
  fi
```

Parte 3 - Comportamiento automático

Fábrica de Empanadas

Enunciado

Se quiere especificar el comportamiento de una fábrica de empanadas que está totalmente automatizada. A medida que se encuentran listas, las empanadas van saliendo de una máquina una a una y son depósitadas en una caja para empanadas. En la caja caben 12 empanadas y cuando esta se llena, es automáticamente despachada y reemplazada por una caja vacía.

Requerimientos

Se quiere saber:

- Cuántas cajas de empanadas se despacharon en total.
- Cuántas empanadas hay en la caja que está actualmente abierta.

Caja abierta con 12 empanadas ? JAMÁS!

↳ Comportamiento automático

TAD FábricaDeEmpanadas

Lun. 14/Sep/2020

géneros fábrica

observadores

cajasDespechadas : fábrica → nat

empanadasEnCaja : fábrica → nat

generadores

crearFábrica : → fábrica

cocEmpanada : fábrica → fábrica {sin rest!}

-- despacharCaja : es comportamiento automático!

axiomas

cajasDesp(crearFábrica(f)) ≡ 0

cajasDesp(cocEmpanada(f)) ≡

cajasDesp(f) + -- 1 si #empa = 11

if empaEnCaja(f) = 11 then

1

else

0

fi

empanadasEnCaja(crearFábrica(f)) ≡ 0

empanadasEnCaja(cocEmpanada(f)) ≡

if empaEnCaja(f) = 11 then

0

else

empaEnCaja(f) + 1

Ejercicio: El ascensor inútil

Enunciado

Se quiere especificar el comportamiento de un ascensor que lleva personas entre dos pisos. La capacidad máxima del ascensor es de 3 personas. El ascensor se pone en funcionamiento cuando ingresan 3 personas, sin necesidad de apretar ningún botón. Cuando el ascensor se pone en movimiento, se desplaza del piso en el que está hacia el otro. En cuanto llega al piso de destino, las personas que están en el interior del ascensor lo desocupan inmediatamente. En cualquier momento pueden llegar personas a cualquiera de los dos pisos. En el caso que el ascensor no esté, las personas forman una fila y esperan a que el ascensor las venga a buscar.

×

TAD Piso

géneros piso

generadores

A : → piso

B : → piso

otras op.

otroPiso : piso → piso

otroPiso(A) ≡ B

otroPiso(B) ≡ A

TAD Ascensor

géneros ascensor

igualdad observacional

$$(\forall a, a' : \text{ascensor})\Big(a =_{obs} a' \iff \wedge_{p : \text{piso}} \big(\text{personasEn}(a, p) = \text{personasEn}(a', p) \wedge \text{piso?}(a) = \text{piso}(a')\big)\Big)$$

(=obs de TAD Piso.)

observadores

personasEn : ascensor × piso → nat

piso? : ascensor → piso

generadores

crearAscensor : → ascensor

llegaPersona : ascensor × piso → ascensor

axiomas

personasEn(crearAscensor, p) ≡ 0

personasEn(llegaPersona(a, pl), p) ≡

```
if piso?(a) = pl ∧ personasEn(a, pl) = 2
    then
        if pl = p then
            0
        else
            if personasEn(a, p) < 3 then
                personasEn(a, p)
            else
                personasEn(a, p) - 3
            fi
        fi
    else
        if pl = p then
            personasEn(a, p) + 1
        else
            personasEn(a, p)
        fi
fi
```

$piso?(crearAscensor) \equiv A$ -- elijo porque sí

$piso?(llegaPersona(a, pl)) \equiv$

if $piso?(a) = pl \wedge personasEn(a, pl) = 2$

then

if $personasEn(a, otroPiso(pl)) < 3$

then

$otroPiso(piso?(a))$

else

$piso?(a)$

fi

else

$piso?(a)$

fi