Algoritmos y Estructuras de Datos II

Departamento de Computación
Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires

Trabajo Práctico 1: Especificación

Lollapatuza

Alias de grupo: MVNWOFAHTVRARZOSADUY

Integrante	LU	Correo electrónico
Rafael Montero	1546/21	rafamontero1000@gmail.com
Beto Calero	620/15	calero.bd@gmail
Mateo Lazarte	539/22	mateolazarte07@gmail.com
Esteban Mena	540/22	estebanpetiso@gmail.com

Reservado para la cátedra

Instancia	Docente	Nota
Primera entrega		
Segunda entrega		

1. TADs

TAD PERSONA **ES** STRING
TAD ITEM **ES** STRING
TAD PRECIO **ES** NAT/N>0
TAD DESCUENTO **ES** NAT/N<100

TAD LOLLAPATUZA

géneros Lolla

usa BOOL, NAT, TUPLA(α), CONJ(α), PUESTODECOMIDA

igualdad observacional

$$(\forall L_1, L_2 : \text{Lolla}) \left(L_1 =_{\text{obs}} L_2 \iff \begin{pmatrix} \text{personasHabilitadas}(L_1) =_{\text{obs}} \text{personasHabilitadas}(L_2) \\ \wedge\ \text{puestosLolla}(L_1) =_{\text{obs}} \text{puestosLolla}(L_2) \\ \wedge_{\text{L}}\ (\forall q\text{: persona}) \\ (\text{q} \in \text{personasHabilitadas}(L_1) \Rightarrow_{\text{L}} (\forall p\text{: puesto}) \\ (\text{p} \in \text{puestosLolla}(L_1) \Rightarrow_{\text{L}} \text{registroLolla}(L_1, \text{p}, \text{q}) =_{\text{obs}} \\ \qquad \text{regristroLolla}(L_2, \text{p}, \text{q}))) \end{pmatrix} \right)$$

observadores básicos

personasHabilitadas	: Lolla	$\longrightarrow$	conj(persona)
puestosLolla	: Lolla	$\longrightarrow$	conj(puesto)
registroLolla	: Lolla L $\times$ puesto p $\times$ persona q	$\longrightarrow$	Lolla
			{p $\in$ puestosLolla(L) $\wedge$ q $\in$ personasHabilitadas(L)}

generadores

crearLolla	: conj(puesto) ps $\times$ conj(persona) qs	$\longrightarrow$	Lolla
	$\begin{cases} (\forall p_1, p_2 : \text{puesto})\ (p_1 \in \text{ps} \wedge p_2 \in \text{ps} \Rightarrow_{\text{L}} \neg(\exists t_1, t_2 : \text{item x nat})\ (t_1 \in \text{Menu}(p_1) \wedge t_2 \in \text{Menu}(p_2)) \\ \wedge\ \neg(t_1 = t_2) \wedge_{\text{L}} \text{precio?}(p_1, \pi_1(t_1)) \neq \text{precio?}(p_2, \pi_1(t_2)))) \end{cases}$		
venderLolla	: Lolla L $\times$ puesto p $\times$ item i $\times$ nat n $\times$ persona q	$\longrightarrow$	Lolla
	$\begin{cases} \text{p} \in \text{puestosLolla(L)} \wedge \text{q} \in \text{personasHabilitadas(L)} \wedge_{\text{L}} (\exists t\text{: item} \times \text{nat})(\text{t} \in \text{Menu(p)} \wedge \pi_1(\text{t}) = \text{i}) \\ \wedge_{\text{L}}\ 0 < \text{n} \leq \text{Stock(p, i)} \end{cases}$		
hackearLolla	: Lolla L $\times$ puesto p $\times$ item i $\times$ persona q	$\longrightarrow$	Lolla
	$\begin{cases} \text{p} \in \text{puestosLolla(L)} \wedge \text{q} \in \text{personasHabilitadas(L)} \wedge_{\text{L}} \\ (\exists t\text{: item} \times \text{nat})(\text{t} \in \text{Registro(p, q)} \wedge \pi_1(\text{t}) = \text{i}) \wedge (\forall n\text{: nat})(\text{Promocion(p, i, n)} = 0) \end{cases}$		

otras operaciones

gastoTotal	: Lolla L $\times$ persona q	$\longrightarrow$	nat
			{q $\in$ personasHabilitadas(L)}
gastoTotalAux	: persona $\times$ conj(puesto)	$\longrightarrow$	nat
masGastó	: Lolla	$\longrightarrow$	persona
masGastóAux	: Lolla $\times$ conj(persona)	$\longrightarrow$	persona
mayorGasto	: Lolla	$\longrightarrow$	nat
mayorGastoAux	: Lolla $\times$ conj(persona)	$\longrightarrow$	nat

axiomas $\forall L$: Lolla, $\forall ps$: conj(puesto), $\forall qs$: conj(persona), $\forall p, p_1$: puesto, $\forall i$: item, $\forall n$: nat, $\forall q, q_1$: persona

personasHabilitadas(crearLolla(ps, qs))	$\equiv$	qs
personasHabilitadas(venderLolla(L, p, i, n, q))	$\equiv$	personasHabilitadas(L)
personasHabilitadas(hackearLolla(L, p, i q))	$\equiv$	personasHabilitadas(L)
puestosLolla(crearLolla(ps, qs))	$\equiv$	ps
puestosLolla(venderLolla(L, p, i, n, q))	$\equiv$	Ag(vender(p, i, n, q), puestosLolla(L) $-$ {p})
puestosLolla(hackearLolla(L, p, i, q))	$\equiv$	Ag(hackearCompra(p, i, q), puestosLolla(L) $-${p})
registroLolla(crearLolla(ps, qs), p, q)	$\equiv$	$\emptyset$
registroLolla(venderLolla(L, p_1, i, n, q_1), p, q)	$\equiv$	**if** p = p_1 $\wedge$ q = q_1 **then**
		Registro(vender(p, i, n, q), q)
		else
		Registro(p, q)
		fi

registroLolla(hackearLolla(L, p_1, i, q_1), p, q) $\equiv$ **if** p = $p_1 \wedge$ q = q_1 **then**
Registro(hackearCompra(p, i, q), q)
else
Registro(p, q)
fi

gastoTotal(L, q) $\equiv$ gastoTotalAux(q, puestosLolla(L))

gastoTotalAux(q, ps) $\equiv$ **if** $\emptyset$?(ps) **then**
0
else
gastosDePersona(dameUno(ps), q) +
gastoTotalAux(q, sinUno(ps))
fi

mayorGasto(L) $\equiv$ mayorGastoAux(L, personasHabilitadas(L))

mayorGastoAux(L, qs) $\equiv$ máx(gastoTotal(L, dameUno(qs)),
mayorGastoAux(L, sinUno(qs)))

masGastó(L) $\equiv$ masGastóAux(L, personasHabilitadas(L))

masGastóAux(L, qs) $\equiv$ **if** mayorGasto(L) = gastoTotal(L, dameUno(qs)) **then**
dameUno(qs)
else
masGastóAux(L, sinUno(qs))
fi

Fin TAD

TAD PUESTODECOMIDA

géneros puesto

exporta puesto, generadores, observadores, gastosDePersona, precio?

usa BOOL, NAT, TUPLA(α), CONJ(α)

igualdad observacional

$$(\forall p_1, p_2 : \text{puesto}) \left(p_1 =_{\text{obs}} p_2 \iff \begin{pmatrix} \text{Menu}(p_1) =_{\text{obs}} \text{Menu}(p_2) \wedge_{\text{L}} (\forall t\text{: item x precio}) \\ (\text{t} \in \text{Menu}(p_1) \Rightarrow_{\text{L}} \text{Stock}(p_1, \pi_1(\text{t})) =_{\text{obs}} \text{Stock}(p_2, \pi_1(\text{t})) \\ \wedge\ (\forall n\text{: nat})(\text{Promocion}(p_1,\pi_1(\text{t}),\text{n}) =_{\text{obs}} \\ \qquad \text{Promocion}(p_2,\pi_1(\text{t}),\text{n}))) \quad \wedge \\ (\forall q\text{: persona})(\text{Registro}(p_1,\text{q}) =_{\text{obs}} \text{Registro}(p_2,\text{q})) \end{pmatrix} \right)$$

observadores básicos

Menu	: puesto p	$\longrightarrow$ conj(item $\times$ precio)	
Stock	: puesto $p \times$ item i	$\longrightarrow$ nat	$\{(\exists$ t : item x precio)(t $\in$ Menu(p) $\wedge\ \pi_1$(t) = i)$\}$
Promocion	: puesto $p \times$ item $i \times$ nat n	$\longrightarrow$ descuento	$\{(\exists$ t : item x precio)(t $\in$ Menu(p) $\wedge\ \pi_1$(t) = i)$\}$
Registro	: puesto $\times$ persona	$\longrightarrow$ conj(item $\times$ nat)	

generadores

crearPuesto	: conj(item $\times$ nat $\times$ precio) c	$\longrightarrow$ puesto	$\left\{ \begin{matrix} \neg(\exists\ t_1, t_2 : \text{item x nat x precio})(\ t_1 \in \text{c} \wedge t_2 \in \text{c} \wedge \neg(t_1 = t_2) \wedge \pi_1(t_1) = \pi_1(t_2)\ ) \wedge \\ (\forall t\text{: item x nat x precio})(\text{t} \in \text{c}\ \Rightarrow\ \pi_2(\text{t}) > 0) \end{matrix} \right\}$
implementarPromocion	: puesto $p \times$ item $i \times$ nat $n \times$ descuento d	$\longrightarrow$ puesto	$\{\neg$hayPromocion?(p, i, n) $\wedge\ 0 <$ n $\leq$ Stock(p, i)$\}$
vender	: puesto $p \times$ item $i \times$ nat $n \times$ persona q	$\longrightarrow$ puesto	$\{(\exists$ t : item x precio)(t $\in$ Menu(p) $\wedge\ \pi_1$(t) = i) $\wedge_{\text{L}}\ 0 <$ n $\leq$ Stock(p, i)$\}$
hackearCompra	: puesto $p \times$ item $i \times$ persona q	$\longrightarrow$ puesto	$\{(\exists t$: item $\times$ nat)(t $\in$ Registro(p, q) $\wedge\ \pi_1$(t) = i) $\wedge_{\text{L}}\ (\forall n$: nat)(Promocion(p, i, n) = 0)$\}$

otras operaciones

#ItemsComprados	: item $i \times$ conj(item $\times$ nat) c	$\longrightarrow$ nat	$\{(\exists t$: item $\times$ nat)(t $\in$ c $\wedge\ \pi_1$(t) = i)$\}$
hayPromocion?	: puesto $p \times$ item $i \times$ nat n	$\longrightarrow$ bool	$\{$vendeItem?(p, i)$\}$
precio?	: puesto $p \times$ item i	$\longrightarrow$ precio	$\{$vendeItem?(p, i)$\}$
buscaPrecio	: item $i \times$ conj(item $\times$ precio) m	$\longrightarrow$ precio	$\{$encontroItem?(i, m)$\}$
promocionMaxima	: puesto $p \times$ item $i \times$ nat n	$\longrightarrow$ descuento	$\{$vendeItem?(p, i)$\}$

hayPromocionMaxima?	: puesto p × item i × nat n	⟶ bool	{vendeItem?(p, i)}
aplicarDescuento	: nat × descuento	⟶ nat	
div	: nat n × nat k	⟶ nat	{0 < k}
gastosDePersona	: puesto × persona	⟶ nat	
gastosDelRegistro	: puesto × conj(item × nat)	⟶ nat	
vendeItem?	: puesto × item	⟶ bool	
encontroItem?	: item × conj(item × precio)	⟶ bool	

axiomas $\forall c$: conj(item × nat × precio), $\forall i, i_2$: item, $\forall n, n_2, k$: nat, $\forall q, q_2$: persona, $\forall d$: descuento, $\forall r$: conj(item × nat), $\forall m$: conj(item × precio), $\forall p$: puesto

Menu(crearPuesto(c)) ≡ **if** $\emptyset$?(c) **then**
  $\emptyset$
else
  Ag(⟨π_1(dameUno(c)), π_3(dameUno(c))⟩,
    Menu(crearPuesto(sinUno(c))))
fi

Menu(implementarPromocion(p, i, n, d)) ≡ Menu(p)

Menu(vender(p, i, n, q)) ≡ Menu(p)

Menu(hackearCompra(p, i, q)) ≡ Menu(p)

Stock(crearPuesto(c), i) ≡ **if** π_1(dameUno(c)) = i **then**
  π_2(dameUno(c))
else
  Stock(crearPuesto(sinUno(c), i))
fi

Stock(implementarPromocion(p, i, n, d), i_2) ≡ Stock(p, i_2)

Stock(vender(p, i, n, q), i_2) ≡ **if** ¬(i = i_2) **then**
  Stock(p, i_2)
else
  Stock(p, i_2) - n
fi

Stock(hackearCompra(p, i, q), i_2) ≡ **if** i = i_2 **then**
  Stock(p,i_2) + 1
else
  Stock(p,i_2)
fi

Promocion(crearPuesto(c), i, n) ≡ 0

Promocion(implementarPromocion(p, i, n, d), i_2, n_2) ≡ **if** n = n_2 ∧ i = i_2 **then**
  d
else
  Promocion(p, i_2, n_2)
fi

Promocion(vender(p, i, n, q), i_2, n_2) ≡ Promocion(p, i_2, n_2)

Promocion(hackearCompra(p, i, q), i_2, n) ≡ Promocion(p, i_2, n)

Registro(crearPuesto(c), q) ≡ $\emptyset$

Registro(implementarPromocion(p, i, n, d), q) ≡ Registro(p, q)

Registro(vender(p, i, n, q), q_2) ≡ **if** q = q_2 **then**
  Ag(⟨i, n⟩, Registro(p, q_2))
else
  Registro(p, q_2)
fi

Registro(hackearCompra(p, i, q), q_2) ≡ **if** ¬(q = q_2) **then**
  Registro(p, q_2)
else
  Ag(
    ⟨i, #ItemsComprados(i, Registro(p, q_2)) - 1⟩,
    Registro(p, q_2) -
    {⟨ i, #ItemsComprados(i, Registro(p, q_2)) ⟩}
)
fi

#ItemsComprados(i, r) ≡ **if** i = π_1(dameUno(r)) **then**
    π_2(dameUno(r))
else
    #ItemsComprados(i, sinUno(r))
fi

hayPromocion?(p, i, n) ≡ **if** Promocion(p, i, n) = 0 **then**
    false
else
    true
fi

precio?(p, i) ≡ buscaPrecio(i, Menu(p))

buscaPrecio(i, m) ≡ **if** π_1(dameUno(m)) = i **then**
    π_2(dameUno(m))
else
    buscaPrecio(i, sinUno(m))
fi

promocionMaxima(p, i, n) ≡ **if** n = 0 **then**
    0
else
    if hayPromocion?(p, i, n) **then**
        Promocion(p, i, n)
    else
        promocionMaxima(p, i, n - 1)
    fi
fi

hayPromocionMaxima?(p, i, n) ≡ **if** promocionMaxima(p, i, n) = 0 **then**
    false
else
    true
fi

div(n, k) ≡ **if** n < k **then** 0 **else** 1 + div(n - k, k) **fi**

aplicarDescuento(n, d) ≡ div(n × (100 - d), 100)

gastosDePersona(p, q) ≡ gastosDelRegistro(p, Registro(p, q))

gastosDelRegistro(p, r) ≡ **if** ∅?(r) **then**
    0
else
    aplicarDescuento(
        precio?(π_1(dameUno(r))) × π_2(dameUno(r)),
        promocionMaxima(
            p,
            π_1(dameUno(r)),
            π_2(dameUno(r))
)
) + gastosDelRegistro(p, sinUno(r))
fi

vendeItem?(p, i) ≡ encontroItem?(i, Menu(p))

encontroItem?(i, m) ≡ **if** ∅?(m) **then**
    false
else
    if π_1(dameUno(m)) = i **then**
        true
    else
        encontroItem?(i, sinUno(m))
    fi
fi

Fin TAD

2. Decisiones

- Consideramos personas habilitadas a comprar a toda aquella persona que esté dentro del Lollapatuza. Es decir, aquél conjunto de personas con las que creamos un Lolla. Se podría interpretar que estas personas compraron una entrada y por eso pueden acceder.
- Asumimos que si una persona compra es porque tiene el dinero para hacerlo. En ningún lugar del enunciado indica que tengamos que modelar preocupándonos por si las personas tienen o no dinero.
- Tomamos la decisión de permitir en nuestro menú ítems que tengan stock 0, en lugar de eliminar estos ítems del menú. Esto es así porque pensamos en qué pasaría si se hackea un ítem que ya no tiene stock pero que en el momento de realizarse la compra estaba en el sistema. Imaginemos que una persona compra el último ítem que tenemos en stock, por lo que la información de ese item se elimina del menú. Luego, se hackea a esa persona y se elimina ese registro, ya que no tenía descuento. Pero no podremos saber su precio y si tenía o no descuento si lo eliminamos del menú una vez que su stock llegó a 0. Por eso permitimos ítems cuyo stock es 0, sin embargo, no permitimos ventas de ítems cuyo stock sea 0. Esa restricción la ponemos en vender, pidiendo que $0 < n \leq$ Stock(p, i), y cumpliendo de este modo con la condición del enunciado de que "El local sólo puede vender los items de su menú que tiene disponible".
- Si algún ítem en un puesto no tiene descuento, permitimos que el observador "Promoción"nos devuelva descuento 0 para facilitar los cálculos a la hora de aplicar el descuento total a una venta.