Algoritmos y Estructuras de Datos II

Trabajo Práctico 1

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

Lollapatuza

Integrante	LU	Correo electrónico
Agustin Fernandez Aragon	998/21	f.a.agustin@gmail.com
Bruno Muschietti	924/21	brunomuschi@gmail.com
Felipe Saidón	1436/21	felipesaidon@gmail.com
Matias Daniel Diaz Sarmiento	704/19	mdds.2017@gmail.com

Reservado para la cátedra

Instancia	Docente	Nota
Primera entrega		
Segunda entrega		

1. Renombres de TADs

TAD Comida es String

TAD Persona es String

TAD Descuentos es dicc(Tupla<Comida,Cantidad>,nat)

TAD Cantidad es Nat

TAD Menu es Diccionario(Comida,Nat)

TAD stock es Diccionario(comida,cantidad)

TAD Venta es Tupla<Persona,Comida,Cantidad>

2. Desarrollo

2.1. TAD Puestos de Comida

TAD Puesto

igualdad observacional

$$(\forall pc1, pc2 : \text{puesto}) \left(pc1 =_{\text{obs}} pc2 \iff \begin{pmatrix} \text{ID(pc1)}=_{\text{obs}}\text{ID(pc2)} \wedge \\ \text{stock(pc1)}=_{\text{obs}}\text{stock(pc2)} \wedge \\ \text{menu(pc1)}=_{\text{obs}}\text{menu(pc2)} \wedge \\ \text{descuento(pc1)}=_{\text{obs}}\text{descuento(pc2)} \wedge \\ \text{ventasTotales(pc1)}=_{\text{obs}}\text{ventasTotales(pc2)} \end{pmatrix} \right)$$

géneros puesto

exporta puesto, generadores, observadores, operaciones

usa string, diccionario(clave,significado), nat, conjunto(tupla$< \pi_1, \pi_2, \pi_3 >$), festival, persona

generadores

crearPuesto : string id × stock s × menu m × descuentos d $\longrightarrow$ puesto
$\left\{ \begin{array}{l} \text{claves(m)} = \text{claves(s)} \wedge (\forall \text{c} : \text{tupla<com,cant>})(\text{def?(c,d)} \rightarrow_L \text{def?}(\pi_1\text{(c)}, \text{s})) \wedge \\ (\neg \exists \text{ c} : \text{tupla<com,cant>})(\text{def?(c,d)} \wedge_L (\text{obtener(c,d)} > 100) \wedge (\pi_2\text{(c)} \leq 0)\ ) \end{array} \right\}$

vender : puesto pc × persona p × comida c × cantidad $cant$ $\longrightarrow$ puesto
{c ∈ claves(stock(pc)) $\wedge_L$ cant≤obtener(c,stock(pc)) ∧ cant > 0}

hackearPuesto : Puesto pc × venta v $\longrightarrow$ Puesto {v ∈ ventasTotales(pc)}

observadores básicos

ID : puesto $\longrightarrow$ nat
stock : puesto $\longrightarrow$ stock
menu : puesto $\longrightarrow$ menu
descuentos : puesto $\longrightarrow$ dicc(tupla<comida,cant>, nat)
ventasTotales : puesto $\longrightarrow$ multiconj(venta)

otras operaciones

comprasDePersona : puesto pc × persona per $\longrightarrow$ multiconj(venta)
gastoDePersona : puesto pc × persona per $\longrightarrow$ nat
obtenerCompras : multiconj(venta) × persona $\longrightarrow$ multiconj(venta)
huboDescuento : puesto pc × comida com × cantidad $cant$ $\longrightarrow$ Bool
elDescuento : puesto pc × comida com × cantidad $cant$ $\longrightarrow$ nat
CalculaGasto : multiconjunto(venta) v × puesto pc $\longrightarrow$ nat {v ∈ ventasTotales(pc)}

axiomas

ID(crearPuesto(id, s, m, d)) ≡ id
ID(vender(pc,per,com,cant)) ≡ ID(pc)
ID(hackearPuesto(pc,v)) ≡ ID(pc)
stock(crearPuesto(id,s, m, d)) ≡ s
stock(vender(pc,per,com,cant) ≡ definir(com, obtener(com,stock(pc)) − cant, borrar(com,stock(pc)))
stock(hakearPuesto(pc,v)) ≡ definir(π_2(v), obtener(π_2(v), stock(pc)) +1, borrar(π_2(v),stock(pc)))
menu(crearPuesto(id, s, m, d)) ≡ m
menu(vender(pc,per,com,cant)) ≡ menu(pc)

menu(hackearPuesto(pc,v)) ≡ menu(pc)
descuentos(crearPuesto(*id*, s, m, d)) ≡ d
descuentos(venta(pc,com)) ≡ descuentos(pc)
descuentos(hackearPuesto(pc,v)) ≡ descuentos(pc)
ventasTotales(crearPuesto(id,s,m,des)) ≡ {}
ventasTotales(vender(pc,per,com.cant)) ≡ Ag(Tupla<per,com,cant>, ventasTotales(pc))
ventasTotales(hackearPuesto(pc,v)) ≡ **if** π_3(v)=1 **then**
ventasTotales(pc) - v
else
Ag(<π_1(v),π_2(v),π_3(v) − 1 >, ventasTotales(pc) - v)
fi
comprasDePersona(pc,per) ≡ obtenerCompras(ventasTotales(pc),per)
obtenerCompras(ventas,per) ≡ **if** esVacio?(ventas) **then**
∅
else
if π_1(dameUno(ventas))=per
then
Ag(dameUno(ventas), obtenerCompras(sinUno(ventas),per))
else
obtenerCompras(sinUno(ventas),per)
fi
fi
huboDescuento(pc,com,cant) ≡ **if** cant=0 **then**
False
else
if def?(tupla<com,cant>,descuentos(pc)) **then**
True
else
huboDescuento(pc,com,cant-1)
fi
fi
elDescuento(pc,com,cant) ≡ **if** cant=0 **then**
0
else
if def?(tupla<com,cant>,descuentos(pc)) **then**
obtener(<com,cant>,descuentos(pc))
else
elDescuento(pc,com,cant-1)
fi
fi
gastoDePersona(pc,per) ≡ CalculaGasto(comprasDePersona(pc,per),pc)
CalculaGasto(compras,pc) ≡ **if** esVacio?(compras) **then**
0
else
aplicarDescuento(obtener(π_2(dameUno(compras)),menu(pc))×
π_3(dameUno(compras)),
elDescuento(pc,π_2(dameUno(compras)),π_3(dameUno(compras)))) +
CalculaGasto(sinUno(compras),pc)
fi

Fin TAD

2.2. TAD Festival

TAD FESTIVAL

igualdad observacional

$$(\forall f1, f2 : \text{festival}) \left(f1 =_{\text{obs}} f2 \iff \begin{pmatrix} \text{personas(f1)}=_{\text{obs}}\text{personas(f2)} \wedge \\ \text{puestos(f1)}=_{\text{obs}}\text{puestos(f2)} \end{pmatrix} \right)$$

géneros festival

exporta festival, generadores, observadores, operaciones

usa string, multiconjunto(tupla$< \pi_1, \pi_2, \pi_3 >$), nat, bool, conjunto(α), persona, puesto

generadores

abrir : conj(personas) $\times$ conj(puestos) *pcs* $\longrightarrow$ festival

$$\left\{ \begin{array}{l} (\forall \text{c} : \text{comida})(\forall \text{pc1, pc2} : \text{Puesto})(((\text{pc1,pc2} \in \text{pcs}) \wedge_L (\text{def?(c,menu(pc1))} \wedge \text{def?(c,menu(pc2))})) \\ \rightarrow_L \text{obtener(c,menu(pc1))} = \text{obtener(c,menu(pc2))})) \end{array} \right\}$$

hacker : festival f $\times$ persona *per* $\times$ comida *com* $\longrightarrow$ festival

$$\left\{ \begin{array}{l} (\text{per} \in \text{personas(f)}) \wedge \exists\ (\text{p1} : \text{Puesto, v1} : \text{Venta})(\text{p1} \in \text{puestos(f)} \wedge \text{v1} \in \text{ventas(p1)}) \\ \wedge\ \pi_1(\text{v1}) = \text{per} \wedge \pi_2(\text{v1}) = \text{com} \wedge \neg\ \text{huboDescuento(p1,com,}\pi_3\text{(v1)))} \end{array} \right\}$$

compra/venta : festival f $\times$ persona p $\times$ puesto *pc* $\times$ comida *com* $\times$ cantidad *cant* $\longrightarrow$ festival

$$\left\{ \begin{array}{l} \text{p} \in \text{personas(f)} \wedge \text{pc} \in \text{puestos(f)} \wedge \text{com} \in \text{claves(stock(pc))} \wedge \text{cant} \leq \text{obtener(com,stock(pc))} \wedge \\ \text{cant}>0 \end{array} \right\}$$

observadores básicos

personas : festival $\longrightarrow$ conj(personas)
puestos : festival $\longrightarrow$ conj(puestos)

otras operaciones

compraParaHackear : conjunto(puesto) *pcs* $\times$ persona *per* $\times$ comida *com* $\longrightarrow$ venta {$\neg$esVacio?(pcs)}
existeCompra : puesto $\times$ multiconjunto(venta) $\times$ persona $\times$ comida $\longrightarrow$ Bool
laCompra : puesto $\times$ multiconjunto(venta) *ventas* $\times$ persona *per* $\times$ comida *com* $\longrightarrow$ tupla<Puesto,Venta>
{($\exists$ v: venta)(v $\in$ ventas $\wedge$ π_1(v) = per $\wedge$ π_2(v) = com}
mayorGastadorEnFest : festival $\longrightarrow$ persona {$\neg$esVacio?(personas(f))}
mayorGastador : conjunto(persona) *pers* $\times$ conjunto(puesto) $\longrightarrow$ persona {$\neg$esVacio?(pers)}
gastoMaximo : conjunto(persona) *pers* $\times$ conjunto(puesto) $\longrightarrow$ nat {$\neg$esVacio?(pers)}
gastoTotal : persona $\times$ conjunto(puesto) $\longrightarrow$ nat

axiomas

personas(abrir(pers,pcs)) $\equiv$ pers
personas(hacker(f,per,com) $\equiv$ personas(f)
personas(compra/venta(f,per,pc,com,cant $\equiv$ personas(f)
puestos(abrir(pers,pcs) $\equiv$ pcs
puestos(hacker(f,per,com) $\equiv$ Ag(hackearPuesto(π_1(compraParaHackear(puestos(f),per,com),
π_2(compraParaHackear(puestos(f),per,com)),puestos(f)-
π_1(compraParaHackear(puestos(f),per,com)))))
puestos(compra/venta(f,per,pc,com,cant)) $\equiv$ Ag(vender(pc,per,com,cant),puestos(f)-pc)
compraParaHackear(pcs,per,com) $\equiv$ **if** existeCompra(dameUno(pcs),ventasTotales(dameUno(pcs)),per,com)
then
<dameUno(pcs),
laCompra(dameUno(pcs),ventasTotales(dameUno(pcs)),per,com)>
else
compraParaHackear(sinUno(pcs),per,com)
fi
existeCompra(pc, ventas,per,com) $\equiv$ **if** esVacio?(ventas) **then**
False
else
if $\pi_1$dameUno(ventas)= per $\wedge$ $\pi_2$dameUno(ventas) = com $\wedge$ $\neg$ huboDescuento(pc,com,cant) **then**
True
else
existeCompra(pc,sinUno(ventas),per,com)
fi
fi

laCompra(pc, ventas,per,com) ≡ **if** $\pi_1$dameUno(ventas)= per ∧ $\pi_2$dameUno(ventas) = com ∧ ¬ huboDescuento(pc,com,cant) **then**
dameUno(ventas)
else
laCompra(pc,sinUno(ventas),per,com)
fi
mayorGastadorEnFest(f) ≡ mayorGastador(personas(f),puestos(f))
mayorGastador(pers,pcs) ≡ **if** gastoMaximo(pers,pcs) = gastoTotal(dameUno(pers),pcs) **then**
dameUno(pers)
else
mayorGastador(sinUno(pers),pcs)
fi
gastoMaximo(pers,pcs) ≡ **if** esVacio?(pers) **then**
0
else
max(gastoTotal(dameUno(pers),pcs)),gastoMaximo(sinUno(pers),pcs))
fi
gastoTotal(per,pcs) ≡ **if** esVacio?(pcs) **then**
0
else
gastoTotal(dameUno(pcs),per) + gastoTotal(per,sinUno(pcs))
fi

Fin TAD

3. Comentarios

3.1. Tad Puestos

comprasDePersona: A partir de un puesto y una persona, devuelve las compras realizadas por esa persona en ese puesto

gastoDePersona: Calcula el valor de todas las compras hechas por una persona en un festival (con su respectivo descuento)

3.2. Tad Festival

compraParaHackear: Usada para axiomatizar el hackeo, devuelve una tupla que contiene en la primera posicion un puesto y en la segunda una venta de ese puesto pero que cumple con las condiciones de hacker, es decir, fue una compra de la persona a hackear, de la comida a hackear y dicha venta no tuvo descuento

existeCompra: Verifica si en un puesto dado hay una venta registrada que cumpla con las condiciones del hackeo

gastoTotal: Calcula la suma de los gastos en cada puesto de una sola persona

gastoMaximo: Devuelve una natural que es el mayor gasto realizado en l festival por alguna persona

mayorGastadorEnFest: Busca y retorna la persona en el festival que su gasto se corresponde con el gasto maximo