DEPARTAMENTO DE COMPUTACION

Facultad de Ciencias Exactas y Naturales - UBA

Trabajo práctico

Especificacion TAD'S

October 24, 2025 — Algoritmos y Estructuras de Datos

Los llocabas

Integrante	LU	Correo electrónico
Gonzalez, Tomás Ezequiel	80/25	tomasgonzalez31506@gmail.com
Salvatierra Laclau, Tomás Eduardo	847/25	salvatierra21tomas@gmail.com
Cortés, Imanol	290/25	imanolcortes364@gmail.com
Kessler, Iván Matías	820/25	ivankessler2005@gmail.com

Facultad de Ciencias Exactas y Naturales

Universidad de Buenos Aires

Ciudad Universitaria - (Pabellón I/Planta Baja)

Intendente Güiraldes 2610 - C1428EGA

Ciudad Autónoma de Buenos Aires - Rep. Argentina

Tel/Fax: (++54 +11) 4576-3300

http://www.exactas.uba.ar

Examen : $\mathsf{dict}\langle p : \mathbb{Z}, r : \mathbb{Z}, \rangle$
Coordenadas : $\langle f : \mathbb{Z}, c : \mathbb{Z} \rangle$
Estudiante : $\langle examen : Examen,\ pos : Coordenadas,\ entrega : \mathsf{bool} \rangle$
`TAD` EdR {

`obs` conjuntoEstudiantes : $\mathsf{conj}\langle Estudiante \rangle$
`obs` examenQueToman : $Examen$

`proc EdR`(in $filas : \mathbb{Z}$, in $columnas : \mathbb{Z}$, in $examenQueToman : Examen$, in $cantidadEstudiantes : \mathbb{Z}$) : EdR {
`requiere` { $dimensionesCorrectas(filas, columnas, cantidadDeEstudiantes) \land (filas = columnas) \land (filas > 0) \land (columnas > 0) \land respuestasEntre0y9(examenQueToman) \land |examenQueToman| > 0$ }
`asegura` { $EdR.examenQueToman = examenQueToman \land |EdR.conjuntoEstudiantes| = cantidadEstudiantes \land bienSentados(filas, columnas, EdR.conjuntoEstudiantes) \land repartirExamenes(EdR.conjuntoEstudiantes)$ }
}

`pred dimensionesCorrectas`($filas : \mathbb{Z}, columnas : \mathbb{Z}, cantidadDeEstudiantes : \mathbb{Z}$) { $(filas\ mod\ 2 \neq 0 \implies cantidadDeEstudiantes \leq ((filas + 1) * columnas)/2) \land (filas\ mod\ 2 = 0 \implies cantidadDeEstudiantes \leq filas * columnas/2)$ }

`pred respuestasEntre0y9`($examen : Examen$) { $(\forall c \in claves(examen))(examen[c] \in \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\})$ }

`pred bienSentados`($filas : \mathbb{Z}, columnas : \mathbb{Z}, estudiantes : \mathsf{conj}\langle Estudiante \rangle$) { $(\forall e \in conjuntoEstudiantes)(\forall e2 \in conjuntoEstudiantes)(0 \leq e.pos.f < filas \land 0 \leq e.pos.c < columnas \land 0 \leq e2.pos.f < filas \land 0 \leq e2.pos.c < columnas \land e.pos.f = e2.pos.f \land e \neq e2) \rightarrow_L (|e.pos.c - e2.pos.c| \geq 2)$ }

`pred repartirExamenes`($conjuntoEstudiantes : \mathsf{conj}\langle Estudiante \rangle$) { $(\forall e \in conjuntoEstudiantes)(e.examen = \{\} \land e.entrega = False)$ }

`proc igualdad`(in $EdR1 : EdR$, in $EdR2 : EdR$) : bool {
`requiere` { $True$ }
`asegura` { $res = true \longleftrightarrow (EdR1.conjuntoEstudiantes = EdR2.conjuntoEstudiantes \land EdR1.examenQueToman = EdR2.examenQueToman)$ }
}

$\texttt{proc copiarse}(\text{in } copion : Estudiante, \text{inout } EdR : EdR) : \{$
$\quad \texttt{requiere } \{ sinEntregar(copion) \ \wedge$
$\quad estudiantePertenece(copion, EdR.conjuntoEstudiantes) \ \wedge$
$\quad existeRespuestaParaCopiar(copion, EdR.conjuntoEstudiantes) \ \wedge \ EdR_0 =$
$\quad EdR$
$\quad \}$
$\quad \texttt{asegura } \{ |EdR_0.conjuntoEstudiantes| = |EdR.conjuntoEstudiantes| \ \wedge$
$\quad restoEstudiantesIgual(copion,$
$\quad EdR_0.conjuntoEstudiantes, EdR.conjuntoEstudiantes) \ \wedge$
$\quad EdR.examenQueToman = EdR_0.examenQuetoman \ \wedge$
$\quad copiadoCorrectamente(copion, EdR_0.conjuntoEstudiantes, EdR.conjuntoEstudiantes)$
$\quad \}$
$\}$
$\texttt{pred sinEntregar}(e : Estudiante) \ \{ e.entrega = False \}$

$\texttt{pred estudiantePertenece}(e : Estudiante, conjEstudiantes : \text{conj}\langle Estudiante \rangle)$
$\{ e \in conjEstudiantes \}$

$\texttt{pred existeRespuestaParaCopiar}(copion : Estudiante, conjEstudiantes : \text{conj}\langle Estudiante \rangle)$
$\{ (\exists e \in conjEstudiantes)(esVecino(e, copion, conjEstudiantes) \ \wedge$
$(\exists c \in claves(e))(c \notin claves(copion))) \}$

$\texttt{pred esVecino}(vecino : Estudiante; copion : Estudiante, conjEstudiantes : \text{conj}\langle Estudiante \rangle)$
$\{ (copion.pos.f = vecino.pos.f \wedge |copion.pos.c - vecino.pos.c| = 2) \vee (copion.pos.c =$
$vecino.pos.c \ \wedge \ copion.pos.f + 1 = vecino.pos.f) \}$

$\texttt{pred restoEstudiantesIgual}(EstuDiferente : Estudiante, conjEstudiantes0 :$
$\text{conj}\langle Estudiante \rangle, : \text{conj}\langle Estudiante \rangle) \ \{ (\forall e0 \in conjEstudiantes0)$
$(e0 \in conjEstudiantes \vee e0 = EstuDiferente) \}$

$\texttt{pred copiadoCorrectamente}(copion : Estudiante, conjEstudiantes0 : \text{conj}\langle Estudiante \rangle,$
$conjEstudiantes : \text{conj}\langle Estudiante \rangle) \ \{ (copion \notin conjEstudiantes) \ \wedge$
$(\exists copionYaCopiado \in conjEstudiantes)(copionYaCopiado.pos = copion.pos \ \wedge$
$copionYaCopiado.entrega = copion.entrega \ \wedge$
$|claves(copion.examen)| + 1 = |copionYaCopiado.examen|) \ \wedge$
$clavesViejasEnNuevoCopion(copionYaCopiado, copion) \ \wedge$
$claveExtraCopiadaDeVecino(copionYaCopiado, copion, conjEstudiante0) \}$

$\texttt{pred clavesViejasEnNuevoCopion}(copionYaCopiado : Estudiante, copion : Estudiante)$
$\{ (\forall c \in claves(copion.examen))(c \in claves(copionYaCopiado.examen) \wedge copion.examen[c] =$
$copionYaCopiado.examen[c]) \}$

pred claveExtraCopiadaDeVecino($copionYaCopiado : Estudiante, copion : Estudiante,$
$conjEstudiante :$ conj$\langle Estudiante \rangle$) {
$(\exists w \in claves(copionYaCopiado.examen))((w \notin claves(copion.examen)) \land$
$(\exists vecino \in conjEstudiantes)(esVecino(vecino, copionYaCopiado, conjEstudiantes) \land$
$w \in claves(vecino.examen) \land copionYaCopiado.examen[w] = vecino.examen[w]$
}

---

proc consultarDarkWeb(in $cantidadAccesos : \mathbb{Z}$; in $examenWeb : Examen$;
inout $EdR : EdR$) : {
requiere { $mismosEjercicios(examenWeb, EdR.examenQueToman) \land$
$respuestasEntre0y9(examenWeb) \land 0 \leq cantidadAccesos \land EdR_0 = EdR$
}
asegura { $|EdR_0.conjuntoEstudiantes| = |EdR.conjuntoEstudiantes| \land$
$EdR.examenQueToman = Edr_0.examenQueToman; \land$
$cantidadCopionesDentroDelRango(cantidadAccesos, EdR_0.conjuntoEstudiantes,$
$EdR.conjuntoEstudiantes) \land$
$copionesSeCopiaronDelExamenWeb(examenWeb, EdR_0.conjuntoEstudiantes,$
$EdR.conjuntoEstudiantes) \land$
$copionesNoEntregaron(EdR_0.conjuntoEstudiantes, EdR.conjuntoEstudiantes) \land$
$copionesMantienenPosicion(EdR_0.conjuntoEstudiantes, EdR.conjuntoEstudiantes) \land$
$noCopionesMantienenExamen(EdR_0.conjuntoEstudiantes,$
$EdR.conjuntoEstudiantes, examenWeb)$ }
}
pred mismosEjercicios($examen1 : Examen, examen2 : Examen$) { $(\forall c \in claves(examen1))$
$(c \in claves(examen2))$ }

pred respuestasEntre0y9($examen : Examen$) { $(\forall c \in claves(examen))(examen[c] \in$
$\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\})$ }

pred cantidadCopionesDentroDelRango($cantidadAccesos : \mathbb{Z}, conjEstudiantes0 :$
conj$\langle Estudiante \rangle, conjEstudiantes :$ conj$\langle Estudiante \rangle$) {
$(\forall e \in conjEstudiantes)(\forall e0 \in conjEstudiantes0)$
$\sum_{e \in conjEstudiantes}(IfThenElse(e.pos = e0.pos \land e.examen \neq e0.examen, 1, 0)) \leq$
$cantidadAccesos$ }

pred copionesSeCopiaronDelExamenWeb($examenWeb : Examen, conjEstudiantes0 :$
conj$\langle Estudiante \rangle, conjEstudiantes :$ conj$\langle Estudiante \rangle$) {
$(\forall e \in conjEstudiantes)(\forall e0 \in conjEstudiantes0)$
$((e.pos = e0.pos \land e.examen \neq e0.examen) \implies e.examen = examenWeb)$ }

pred copionesNoEntregaron($conjEstudiantes0 :$ conj$\langle Estudiante \rangle, conjEstudiantes :$
conj$\langle Estudiante \rangle$) { $(\forall e \in conjEstudiantes)(\forall e0 \in conjEstudiantes0)$
$((e.pos = e0.pos \land e.examen \neq e0.examen) \implies e0.entrega = False)$ }

pred copionesMantienenPosicion($conjEstudiantes0$: conj$\langle Estudiante\rangle$, $conjEstudiantes$: conj$\langle Estudiante\rangle$) {
$(\forall e \in conjEstudiantes)(\forall e0 \in conjEstudiantes0)$
$((e.pos = e0.pos \ \wedge \ e.examen \neq e0.examen) \implies e.pos = e0.pos)$ }

pred noCopionesMantienenExamen($conjEstudiantes0$: conj$\langle Estudiante\rangle$, $conjEstudiantes$: conj$\langle Estudiante\rangle$, $examenWeb : Examen$) {
$(\forall e \in conjEstudiantes)(\forall e0 \in conjEstudiantes0)$
$((e.pos = e0.pos \ \wedge \ e.examen \neq examenWeb) \implies e.examen = e0.examen)$ }

---

proc resolver(inout $EdR : EdR$; in $secExamenes$: seq$\langle Examen\rangle$; in $estudiante$: $Estudiante$) : seq$\langle$Examen$\rangle$ {
requiere { $estudiantePertenece(estudiante, EdR.conjuntoEstudiantes) \ \wedge$
$estudianteConExamenVacio(estudiante) \wedge estudianteNoEntrego(estudiante) \wedge$
$primerExamenVacio(secExamenes) \wedge cadaExamenAgregaUnEjer(secExamenes) \wedge$
$longitudCorrecta(secExamenes, EdR.examenQueToman) \ \wedge$
$resEntre0y9(secExamenes) \ \wedge$
$ejerciciosValidos(secExamenes, EdR.examenQueToman) \wedge EdR_0 = EdR$
}
asegura { $|EdR_0.conjuntoEstudiantes| = |EdR.conjuntoEstudiantes| \ \wedge$
$EdR.examenQueToman = Edr_0.examenQueToman \wedge$
$restoEstudiantesIgual(estudiante,$
$EdR_0.conjuntoEstudiantes, EdR.conjuntoEstudiantes) \ \wedge$
$resuelto(estudiante, EdR_0.conjuntoEstudiantes, EdR.conjuntoEstudiantes, secExamenes)$
}
}
pred estudiantePertenece($e : Estudiante, conjEstudiantes$: conj$\langle Estudiante\rangle$)
{ $e \in conjEstudiantes$ }

pred estudianteConExamenVacio($e : Estudiante$) { $e.examen = \{\}$ }

pred cadaExamenAgregaUnEjer($secExamenes$: seq$\langle Examen\rangle$) { $(\forall i \in \mathbb{Z})(0 \leq i < |secExamenes| - 1 \rightarrow_L |secExamenes[i]| + 1 = |secExamenes[i+1]|)$ }

pred estudianteNoEntrego($e : estudiante$) { $e.entrega = False$ }

pred longitudCorrecta($secExamenes$: seq$\langle Examen\rangle, examenQueToman : Examen$)
{ $|secExamenes| \leq |claves(examenQueToman)| + 1$ }

pred primerExamenVacio($secExamenes$: seq$\langle Examen\rangle$) { $secExamenes_0 = \{\}$
}

pred resEntre0y9($secExamenes$: seq$\langle Examen\rangle$) { $(\forall i \in \mathbb{Z})(0 \leq i < |secExamenes| \rightarrow_L$
$(\forall c \in claves(secExamenes[i]))(secExamenes[c] \in \{0,1,2,3,4,5,6,7,8,9\}))$ }

pred ejerciciosValidos($secExamenes$: seq$\langle Examen\rangle$, $examenQueToman$: $Examen$)
{ $(\forall i \in \mathbb{Z})(0 \leq i < |secExamenes| \rightarrow_L (\forall c \in claves(secExamenes[i]))(c \in claves(examenQueToman)))$ }

pred restoEstudiantesIgual($EstuDiferente$: $Estudiante$, $conjEstudiantes0$: conj$\langle Estudiante\rangle$,
$conjEstudiantes$: conj$\langle Estudiante\rangle$) { $(\forall e0 \in conjEstudiantes0)$
$(e0 \in conjEstudiantes \vee e0 = EstuDiferente)$ }

pred resuelto($estudiante$: $Estudiante$, $conjEstudiantes0$: conj$\langle Estudiante\rangle$,
$conjEstudiantes$: conj$\langle Estudiante\rangle$, $secExamenes$: seq$\langle Examen\rangle$) { $estudiante \notin conjEstudiantes0 \wedge (\exists e \in conjEstudiantes)(e.pos = estudiante.pos \wedge e.entrega = estudiante.entrega \wedge e.examen = secExamenes[|secExamenes| - 1])$ }

---

proc entregar(in $estudiante$: $Estudiante$; inout EdR : EdR;) : {
requiere { $estudiantePertenece(estudiante, EdR.conjuntoEstudiantes) \wedge noEntrego(estudiante) \wedge EdR_0 = EdR$ }
asegura { $EdR.examenQueToman = Edr_0.examenQueToman \wedge |EdR.conjuntoEstudiante| = |EdR_0.conjuntoEstudiantes| \wedge estudianteEntrego(estudiante) \wedge restoEstudiantesIgual(estudiante, EdR_0.conjuntoEstudiantes, EdR.conjuntoEstudiantes) \wedge examenIgual(EdR.conjuntoEstudiantes, Edr_0.conjuntoEstudiantes, estudiante)$ }
}
pred estudiantePertenece(e : $Estudiante$, $conjEstudiantes$: conj$\langle Estudiantes\rangle$)
{ $e \in conjEstudiantes$ }

pred noEntrego(e : $Estudiante$) { $e.entrega = False$ }

pred estudianteEntrego(e : $Estudiante$) { $e.entrega = True$ }

pred restoEstudiantesIgual($EstuDiferente$: $Estudiante$, $conjEstudiantes0$: conj$\langle Estudiante\rangle$, $conjEstudiantes$: conj$\langle Estudiante\rangle$) { $(\forall e0 \in conjEstudiantes0)$
$(e0 \in conjEstudiantes \vee e0 = EstuDiferente)$ }

pred examenIgual($conjEstudiantes$: conj$\langle Estudiante\rangle$, $conjEstudiantes0$: conj$\langle Estudiante\rangle$,
$estudiante$: $Estudiante$) {
$(\forall e0 \in conjEstudiantes0)(\forall e \in conjEstudiantes)(e0.pos = e.pos = estudiante.pos \implies e.examen = e0.examen)$ }

---

proc chequearCopias(in $EdR : EdR$) : seq$\langle Estudiante \rangle$ {
requiere { $todosEntregaron(EdR.conjuntoEstudiantes)$
}
asegura { $(\forall e \in EdR.conjuntoEstudiantes)(e \in res \longleftrightarrow$
$(seCopiaDeVecino(e, EdR.conjuntoEstudiantes) \ \vee$
$seCopiaGeneral(e, EdR.conjuntoEstudiantes)))$ }
}

pred todosEntregaron($conjEstudiantes$: conj$\langle Estudiante \rangle$) {
$(\forall e \in conjEstudiantes)(e.entrega = True)$ }

pred seCopiaDeVecino(e : $Estudiante, conjEstudiantes$: conj$\langle Estudiante \rangle$) {
$(\exists e1 \in conjEstudiantes)(sonVecinos(e, e1) \ \wedge \ seCopiaEDeE1(e, e1))$ }

pred sonVecinos(e : $Estudiante, e1$: $Estudiante$) { $(e.pos.f = e1.pos.f \ \wedge$
$|e.pos.c - e1.pos.c| = 2) \ \vee \ (e.pos.f - 1 = e1.pos.f \ \wedge \ e.pos.c = e1.pos.c)$ }

pred seCopiaEDeE1($e : Estudiante, e1 : Estudiante$) {
$$\frac{respuestasIguales(e.examen, e1.examen)}{|claves(e.examen)|} > 0.6 \}$$

aux respuestasIguales($examen1 : Examen, examen2 : Examen$) : $\mathbb{Z} =$
$\sum_{p \in claves(examen1)} ifThenElse(examen1[p] = examen2[p], 1, 0)$

pred seCopiaGeneral(e: Estudiante, conjEstudiantes: conj<Estudiante>){
$$\frac{examenesIguales(e.pos, e.examen, conjEstudiantes)}{|conjEstudiantes|} > 0.25 \}$$

aux examenesIguales($posicion : Coordenadas, examen : Examen, conjEstudiantes$:
conj$\langle Estudiante \rangle$) : $\mathbb{Z} = \sum_{e \in conjEstudiantes}(ifThenElse(examen = e.examen, 1, 0))$

---

proc corregir(in $EdR : EdR, $) : seq$\langle < Estudiante, \mathbb{R} > \rangle$ {

requiere { $(\forall e \in EdR.conjuntoEstudiantes)(e.entrega = True)$ }

asegura { $(perteneceAConjEstudiantes(EdR.conjuntoEstudiantes, res) \land$
$sinRepetidosEnRes(res)$
$\land (\forall e \in EdR.conjuntoEstudiantes)((e, ponerNota(e, EdR.examenQueToman)) \in$
$res) \longleftrightarrow (\neg seCopiaGeneral(e, EdR.conjuntoEstudiantes) \land$
$\neg seCopiaDeVecino(e, EdR.conjuntoEstudiante))))$ }

}

pred perteneceAConjEstudiantes($e : conj$\langle Estudiantes \rangle$, $f : seq$\langle Estudiante, \mathbb{R} \rangle$)
{ $(\forall i \in \mathbb{Z})((0 < i \leq |f|) \rightarrow_L (f[i]_0 \in e))$ }

pred sinRepetidosEnRes($listaNotas : seq$\langle Estudiante, \mathbb{R} \rangle$) { $((\forall i \in \mathbb{Z})(0 < i \leq$
$|listaNotas|)(\forall j \in \mathbb{Z})(0 < j \leq |listaNotas|) \land i \neq j) \rightarrow_L (listaNotas[i]_0 \neq$
$listaNotas[j]_0)$ }

aux ponerNota($e : Estudiante, examenQueToman : Examen$) : $\mathbb{R} =$
$$\frac{cantidadRtasCorrectas(e.examen, examenQueToman)}{|examenQueToman|}$$

aux cantidadRtasCorrectas($examen : Examen, examenQueToman : Examen$) :
$\mathbb{Z} = \sum_{c \in claves(examen)}(IfThenElse(respuestaIgual(examenQueToman, examen, c), 1, 0))$

pred respuestaIgual($examenQueToman : Examen, examenEstudiante : Examen, clave :$
$\mathbb{Z}$) { $(clave \in claves(examenQueToman) \land$
$examenEstudiante[clave] = examenQueToman[clave])$ }