DEPARTAMENTO
DE COMPUTACION
Facultad de Ciencias Exactas y Naturales - UBA

Trabajo práctico

Especificación de TAD's

February 5, 2026 — Algoritmos y Estructuras de Datos

Java Haters

Integrante	LU	Correo electrónico
Burunov, Celina	23/25	celinaburunov@gmail.com
Doublier, Galo	629/25	galodoublier@gmail.com
Hoton, Thiago	1013/23	thiagohoton@gmail.com
Otero Zappa, Facundo	339/20	facuotero20.88@outlook.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

1 Introducción

1.1 Macros para escribir en LaTeX

Para facilitar la escritura del trabajo práctico en LaTeX, les proveemos de varias macros que les permitirán escribir especificaciones en lógica de primer orden. Por ejemplo, podrán hacer uso de comandos que definen los conectores lógicos: $\wedge, \vee, \rightarrow$ y también los operadores de la lógica trivaluada $\wedge_L$, $\vee_L$, $\rightarrow_L$.

Además, disponen de comandos para cuantificadores: $(\forall x : \mathbb{Z})\ (x \geq 0 \vee x < 0)$, $(\exists x : \mathbb{Z})\ (x \geq 0)$. A estos comandos se les puede agregar un `Largo` al final para que ocupen múltiples renglones.

1.2 Ejemplos de uso de las macros

```
TAD EjemploDeTAD {
    obs secuenciaDeCaracteres : seq⟨char⟩
    obs conjuntoDeTuplas : conj⟨⟨nombre : seq⟨char⟩, apellido : seq⟨char⟩⟩⟩

    proc crearTAD(in caracteres : seq⟨char⟩, in secuenciaDeNombres : seq⟨⟨nombre : seq⟨char⟩,
    apellido : seq⟨char⟩⟩⟩) : EjemploDeTAD {
        requiere { predicadoUnilinea(caracteres) }
        asegura { res.secuenciaDeCaracteres = caracteres }
        asegura {
            |res.conjuntoDeTuplas| = |secuenciaDeNombres| ∧
            predicadoMultilinea(secuenciaDeNombres, res.conjuntoDeTuplas)
        }
    }


    pred predicadoMultilinea(s : seq⟨char⟩, conjunto : conj⟨⟨seq⟨char⟩, seq⟨char⟩⟩⟩)  {
         (∀i : ℤ) (0 ≤ i < |s| →_L s[i] ∈ c)
    }

    pred predicadoUnilinea(s : seq⟨char⟩)  { |s| > 0 }

    aux unAuxiliar(k : ℤ, l : ℤ) : ℤ = |k − l|
}
```

2 Enunciado

En GPT5, un recóndito planeta del Sur Galáctico, hay vida inteligente. Allí, en la cima de la pirámide evolutiva se ubican los Iroquai, unos seres amables y pacíficos, muy similares en forma y tamaño a los seres humanos. Los Iroquai han hecho prodigiosos avances en medicina. Al igual que en la Tierra, todos los años los/as médicos/as deben hacer un examen para iniciar la especialización, el cual por comodidad denominaremos "Examen de Residencia" (EdR).

Este examen es muy exigente y condiciona el futuro profesional de los y las médicos/as. Quienes obtienen los mejores puntajes pueden elegir la especialidad y el centro médico en el que desarrollarán su aprendizaje. Los Iroquai en condiciones normales son honestos/as, pero la dificultad del examen y la perspectiva de no alcanzar el puntaje esperado, puede tentarlos/as a tratar de fraguar el examen de distintas formas.

El EdR consiste de una lista de ejercicios de elección múltiple de 10 opciones (numeradas de 0 a 9) con una única respuesta correcta. El mismo se rinde en un aula con suficiente capacidad para albergar a los y las estudiantes, e incluso pueden quedar asientos vacíos por ausencias. Al comienzo del examen, cada estudiante recibe una copia del mismo con los ejercicios sin responder, y se sienta de manera tal que haya al menos un asiento de distancia con respecto a su compañero/a más próximo de su misma fila. En el transcurso del examen, cada estudiante resuelve los ejercicios en cualquier orden. Sin embargo, algunos/as estudiantes pueden optar por copiarse un ejercicio de un/a compañero/a cercano/a para intentar obtener una nota más alta. El rango de visión solo les permite hacerlo si alguien se encuentra sentado en a lo sumo dos asientos de distancia hacia los costados o uno hacia delante, pero no hacia atrás. Además, aprovechando los avances tecnológicos, personas ajenas al examen, denominadas "proveedores", son capaces de subir una posible resolución del examen a la dark web con un número limitado de accesos, la cual puede ser consultada por estudiantes tomando el examen.

Atentos a esta problemática, una vez que todos y todas las estudiantes entregaron su examen y se da por finalizado, los y las docentes realizan un chequeo de copias, donde establecen el siguiente criterio para determinar si una resolución dada es sospechosa de haberse copiado: Proporción de respuestas equivalentes a un/a compañero/a cercano/a > 60 % ó examen equivalente a más del 25 % del resto del alumnado.

Llegado el momento de la corrección, aquellos Iroquai sospechosos/as de haberse copiado no reciben nota, mientras que al resto se les asigna la proporción de respuestas correctas como nota final.

3 Consignas

En base al enunciado, especificar el TAD EdR con sus correspondientes observadores y los siguientes procedimientos:

- **EdR:** dadas las dimensiones de un aula con la misma cantidad de filas de asientos que columnas, una solución al examen y una cantidad de estudiantes, crea una instancia del sistema.
- **igualdad:** dada dos instancias del TAD, determina si son iguales.
- **copiarse:** un estudiante dado se copia un ejercicio que aún no resolvió de un/a compañero/a cercano/a.
- **consultarDarkWeb:** dada una posible resolución del examen y N cantidad de posibles accesos, a lo sumo N estudiantes utilizan todas las respuestas de la resolución en su examen (incluyendo ejercicios ya resueltos por el/la estudiante).
- **resolver:** dada una secuencia de t pasos de resolución del examen y un/a estudiante que aún no ha comenzado a resolver su examen, devuelve una secuencia con $t+1$ pasos de resolución, donde el paso $(t+1)$-ésimo representa el examen del estudiante a la salida del procedimiento. El primer elemento de la secuencia de entrada corresponde al examen del estudiante antes de que comience a resolverlo (es decir, es un examen sin ninguna respuesta), y el último elemento es el examen inicial luego de t pasos de resolución. El elemento $(t+1)$-ésimo de la salida corresponde al examen del estudiante en el paso t con una respuesta adicional y corresponde a la nueva instancia de examen del estudiante. Por ejemplo, si la entrada es $[examen0, examen1, examen2]$, $examen0$ es el examen del estudiante sin ninguna resolución, $examen1$ contiene una respuesta (por ejemplo, al ejercicio 3), $examen2$ contiene dos respuestas (ejercicio 3 y, por ejemplo, ejercicio 1), y la salida debe ser $[examen0, examen1, examen2, examen3]$, donde $examen3$ contiene las respuestas de $examen2$ más una respuesta adicional (por ejemplo, al ejercicio 5). En este ejemplo, $examen3$ corresponde a la nueva instancia de examen del estudiante.
- **entregar:** un estudiante dado entrega su examen y se retira del aula.
- **chequearCopias:** devuelve una lista de estudiantes sospechosos de haberse copiado, según los criterios detallados en el enunciado.
- **corregir:** devuelve una secuencia de tuplas $\langle estudiante, nota \rangle$ con las correcciones de los exámenes de los y las estudiantes que no fueron sospechosos/as de copiarse, otorgándoles como nota la proporción de respuestas correctas.

Se recomienda utilizar renombres de tipos para los parámetros de los procedimientos así como predicados en los `requiere` y `asegura`. Es posible utilizar `structs` para los tipos (por ejemplo, "punto es struct$\langle x : \mathbb{Z}, y : \mathbb{Z} \rangle$").

4 Resolución

Matriz$\langle T \rangle$ ES seq$\langle$seq$\langle T \rangle\rangle$
Examen ES struct$\langle entregado$: bool, $ejercicios$: seq$\langle item \rangle\rangle$
Item ES struct$\langle respondido$: bool, $opcion$: $\mathbb{Z}\rangle$

TAD EdR {

obs aula : Matriz$\langle$examen$\rangle$
obs resuelto : examen

// Ejercicio 1

proc nuevoEdR(in $dimension : \mathbb{Z}$, in $cantidadDeAlumnos : \mathbb{Z}$, in $solucion$: examen) : EdR {
requiere { $dimension > 0$ }
requiere { $cantidadDeAlumnos > 0$ }
requiere { $esSolucionValida(solucion)$ }
requiere { $elAulaEsSuficientementeGrande(dimension, cantidadDeAlumnos)$ }
asegura {
$esCuadrada(res.aula) \wedge_L |res.aula| = dimension \wedge$
$bienSentados(res.aula) \wedge conExamenEnBlanco(res, solucion) \wedge$
$contarAlumnos(res) = cantidadDeAlumnos \wedge$
$res.resuelto = solucion$
}
}

pred esMatriz(s : Matriz$\langle T \rangle$) { $(\exists l : \mathbb{Z})\ ((\forall i : \mathbb{Z})\ (0 \leq i < |s| \rightarrow_L |s[i]| = l))$ }

pred elAulaEsSuficientementeGrande($dimension : \mathbb{Z}, cantidadDeAlumnos : \mathbb{Z}$) {
$(\exists M$: Matriz$\langle$examen$\rangle)$ (
$esCuadrada(M) \wedge |M| = dimension \wedge$
$bienSentados(M) \wedge contarOcupantes(M) = cantidadDeAlumnos$
)
}

aux contarOcupantes(M : Matriz$\langle$examen$\rangle$) : $\mathbb{Z} = \sum_{f,c=0}^{|M|-1} IfThenElse(M[f][c] \neq \langle\rangle, 1, 0)$

pred esCuadrada(s : Matriz$\langle T \rangle$) { $esMatriz(s) \wedge_L (\forall i : \mathbb{Z})\ (0 \leq i < |s| \rightarrow_L |s[i]| = |s|)$ }

pred enAula($i : \mathbb{Z}, j : \mathbb{Z}, M$: aula) { $esCuadrada(M) \wedge_L 0 \leq i < |M| \wedge 0 \leq j < |M[0]|$ }

pred bienSentados(M : Matriz$\langle T \rangle$) {
$(\forall f : \mathbb{Z})\ (0 \leq f < |M| \wedge_L$
$(\forall c : \mathbb{Z})\ ((0 \leq c < |M| - 1 \wedge_L M[f][c] \neq \langle\rangle) \rightarrow_L M[f][c+1] = \langle\rangle))$
}

pred conExamenEnBlanco(A : EdR, $solucion$: examen) {
$(\forall f, c : \mathbb{Z})$ (
$(enAula(f, c, A.aula) \wedge A.aula[f][c] \neq \langle\rangle) \rightarrow_L$
$A.aula[f][c].entregado = False \wedge |A.aula[f][c].ejercicios| = |solucion.ejercicios| \wedge$
$(\forall k : \mathbb{Z})\ (0 \leq k < |A.aula[f][c].ejercicios| \rightarrow_L$
$(A.aula[f][c].ejercicios[k].respondido = False)))$

}

pred esSolucionValida($prueba$: examen) {
$(\forall k : \mathbb{Z})\ (0 \leq k < |prueba.ejercicios| \rightarrow_L (prueba.ejercicios[k].respondido = True \wedge$
$0 \leq prueba.ejercicios[k].opcion \leq 9))$
}

pred esEstudiante($f : \mathbb{Z}, c : \mathbb{Z}, A$: EdR) {
$enAula(f, c, A.aula) \wedge bienSentados(A.aula) \wedge A.aula[f][c] \neq \langle\rangle$
}

aux contarAlumnos(A : EdR) : $\mathbb{Z} = \sum_{i,j=0}^{|A.aula|-1} IfThenElse(esEstudiante(i, j, A), 1, 0)$

// Ejercicio 2

proc igualdad(in $instancia_1$: EdR, in $instancia_2$: EdR) : bool {
requiere { $True$ }
asegura {
$res = True \leftrightarrow ((instancia_1.aula = instancia_2.aula) \wedge$
$(instancia_1.resuelto = instancia_2.resuelto))$
}
}

// Ejercicio 3

proc copiarse(in $f : \mathbb{Z}$, in $c : \mathbb{Z}$, inout A : EdR) {
requiere { $A = A_0$ }
requiere { $esEstudiante(f, c, A_0)$ }
requiere { $A_0.aula[f][c].entregado = False$ }
requiere { $hayEstudianteCercanoConRespuesta(f, c, A_0)$ }
asegura {
$(\exists i, j, k : \mathbb{Z})$ (
$(0 \leq k < |A.aula[f][c].ejercicios| \wedge_L sonVecinos(f, c, i, j, A)) \wedge_L$
$(\neg ejercicioCompletado(f, c, k, A_0) \wedge ejercicioCompletado(i, j, k, A_0))$
$\wedge (A.aula[f][c].ejercicios[k].respondido = True \wedge$
$A.aula[f][c].ejercicios[k].opcion = A_0.aula[i][j].ejercicios[k].opcion) \wedge$
$losOtrosEjerciciosNoCambian(A_0.aula[f][c].ejercicios, A.aula[f][c].ejercicios, k) \wedge$
$A.aula[f][c].entregado = A_0.aula[f][c].entregado$
$) \wedge$
$losOtrosExamenesNoCambian(f, c, A_0.aula, A.aula) \wedge$
$A_0.resuelto = A.resuelto$
}
}

pred losOtrosEjerciciosNoCambian($ejercicios_1$: seq$\langle item\rangle, ejercicios_2$: seq$\langle item\rangle, k : \mathbb{Z}$) {
$|ejercicios_1| = |ejercicios_2| \wedge_L (\forall i : \mathbb{Z})$ (
$0 \leq i < |ejercicios_1| \wedge i \neq k \rightarrow_L ejercicios_1[i] = ejercicios_2[i])$
}

pred losOtrosExamenesNoCambian($f : \mathbb{Z}, c : \mathbb{Z}, matriz_1$: Matriz$\langle T\rangle, matriz_2$: Matriz$\langle T\rangle$) {
$(\forall i, j : \mathbb{Z})\ ((enAula(i, j, matriz_1) \wedge (i \neq f \vee j \neq c)) \rightarrow_L matriz_1[i][j] = matriz_2[i][j])$

}

pred ejercicioCompletado($f : \mathbb{Z}, c : \mathbb{Z}, k : \mathbb{Z}, A :$ EdR) {
$esEstudiante(f, c, A) \land 0 \leq k < |A.aula[f][c].ejercicios| \land_L$
$(A.aula[f][c].ejercicios[k].respondido = True \land 0 \leq A.aula[f][c].ejercicios[k].opcion \leq 9)$
}

pred sonVecinos($f : \mathbb{Z}, c : \mathbb{Z}, i : \mathbb{Z}, j : \mathbb{Z}, A :$ EdR) {
$enAula(i, j, A.aula) \land enAula(f, c, A.aula) \land$
$((i = f - 1 \land j = c) \lor (i = f \land (j = c - 2 \lor j = c + 2)))$
}

pred hayEstudianteCercanoConRespuesta($f : \mathbb{Z}, c : \mathbb{Z}, A :$ EdR) {
$esEstudiante(f, c, A) \land_L$
$(\exists i, j, k : \mathbb{Z})$ (
$esEstudiante(i, j, A) \land_L 0 \leq k < |A.aula[f][c].ejercicios|) \land_L$
$(sonVecinos(f, c, i, j, A) \land ejercicioCompletado(i, j, k, A)) \land$
$\neg ejercicioCompletado(f, c, k, A)$
)
}

// Ejercicio 4

proc consultarDarkWeb(in $N : \mathbb{Z}$, in $solDarkWeb :$ examen, inout $A :$ EdR) {
requiere { $esSolucionValida(solDarkWeb) \land$
$|solDarkWeb.ejercicios| = |A_0.resuelto.ejercicios|$ }
requiere { $A = A_0$ }
asegura {
$A_0.resuelto = A.resuelto \land$
$(\exists s : \text{seq}\langle\langle \mathbb{Z}, \mathbb{Z}\rangle\rangle)$ (
$(|s| \leq N) \land_L$
$(\forall f, c, k : \mathbb{Z})$ (
$esEstudiante(f, c, A_0) \land 0 \leq k < |A_0.aula[f][c].ejercicios| \land_L (\langle f, c\rangle \in s)) \rightarrow_L$
$(A.aula[f][c].ejercicios[k].respondido = True \land$
$A.aula[f][c].ejercicios[k].opcion = solDarkWeb[k].opcion)$
$\lor$
$esEstudiante(f, c, A_0) \land$
$0 \leq k < |A_0.aula[f][c].ejercicios| \land_L (\langle f, c\rangle \neg \in s)) \rightarrow_L$
$(A_0.aula[f][c] = A.aula[f][c])))$
}
}

// Ejercicio 5

proc resolver(in $f : \mathbb{Z}$, in $c : \mathbb{Z}$, inout $A :$ EdR, in $s : \text{seq}\langle$examen$\rangle$) : seq$\langle$examen$\rangle$ {
requiere { $A = A_0 \land |s| > 0 \land_L s[0] = examenVacio(A_0.aula[f][c])$ }
requiere { $enEstudiante(f, c, A_0)$ }
requiere { $siguienteNoModificaAnteriores(s) \land siguienteTieneUnoMas(s)$ }
requiere { $A_0.aula[f][c].entregado = False \land s[|s| - 1] = A_0.aula[f][c]$ }
requiere { $(\exists k : \mathbb{Z})$ (
$0 \leq k < |A_0.aula[f][c].ejercicios| \land_L A_0.aula[f][c].ejercicios[k].respondido = False$

```
    ) }
    asegura {
        ((|res| = |s| + 1) ∧_L (∀i : ℤ) (0 ≤ i < |s| →_L s[i] = res[i])) ∧_L
        (siguienteNoModificaAnteriores(res) ∧ siguienteTieneUnoMas(res)) ∧_L
        (∀i : ℤ) (
        (0 ≤ i < |res| − 1) →_L
        (cantidadEjerciciosCompletados(res[i+1]) = 1+cantidadEjerciciosCompletados(res[i]))
        ) ∧
        res[|res| − 1] = A.aula[f][c] ∧
        A_0.resuelto = A.resuelto ∧
        losOtrosExamenesNoCambian(f, c, A_0.aula, A.aula)
    }
}

pred siguienteTieneUnoMas(s : seq⟨examen⟩) {
    (∀i : ℤ) (
    (0 ≤ i < |s| − 1) →_L
    (∃k : ℤ) (
        (0 ≤ k < |s[0].ejercicios|) ∧_L
        ¬ejercicioCompletadoEnLista(k, s[i]) ∧ ejercicioCompletadoEnLista(k, s[i + 1])
    )
    )
}

pred siguienteNoModificaAnteriores(s : seq⟨examen⟩) {
    (∀i, k : ℤ) (
    ((0 ≤ i < |s| − 1) ∧_L (0 ≤ k < |s[0].ejercicios|) ∧_L ejercicioCompletadoEnLista(k, s[i])) →_L
    ejercicioCompletadoEnLista(k, s[i + 1]) ∧
    (s[i + 1].ejercicios[k].opcion = s[i].ejercicios[k].opcion)
    )
}

pred ejercicioCompletadoEnLista(k : ℤ, prueba : examen) {
    0 ≤ k < |prueba.ejercicios| ∧_L
    (prueba.ejercicios[k].respondido = True ∧ 0 ≤ prueba.ejercicios[k].opcion ≤ 9)
}

aux cantidadEjerciciosCompletados(prueba : examen) : ℤ =
    Σ_{k=0}^{|prueba.ejercicios|−1} IfThenElse(ejercicioCompletadoEnLista(k, prueba), 1, 0)
```

// Ejercicio 6

```
proc entregar(in f : ℤ, in c : ℤ, inout A : EdR) {
    requiere { A = A_0 }
    requiere { esEstudiante(f, c, A_0) ∧ A_0.aula[f][c].entregado = False }
    asegura {
        A.aula[f][c].entregado = True ∧ A.aula[f][c].ejercicios = A_0.aula[f][c].ejercicios ∧
        (∀i, j : ℤ) (
        (enAula(i, j, A.aula) ∧ (i ≠ f ∨ j ≠ c)) →_L
```

$A_0.aula[i][j] = A.aula[i][j]$
) ∧
$A_0.resuelto = A.resuelto$
}
}

// Ejercicio 7

proc chequearCopias(in A : EdR) : seq$\langle\langle f : \mathbb{Z}, c : \mathbb{Z}\rangle\rangle$ {
requiere { $todosLosEstudiantesEntregaron(A)$ }
asegura {
$(\forall f, c : \mathbb{Z})$ $(\langle f, c\rangle \in res \leftrightarrow (esEstudiante(f, c, A) \wedge_L sospechoso(f, c, A)))$
}
}

pred todosLosEstudiantesEntregaron(A : EdR) {
$(\forall f, c : \mathbb{Z})$ $(esEstudiante(f, c, A) \rightarrow_L A.aula[f][c].entregado = True)$
}

pred sospechoso($f : \mathbb{Z}, c : \mathbb{Z}, A$: EdR) {
$seCopioDeEstudianteCercano(f, c, A) \vee examenParecidoAlResto(f, c, A)$
}

pred seCopioDeEstudianteCercano($f : \mathbb{Z}, c : \mathbb{Z}, A$: EdR) {
$(\forall i, j : \mathbb{Z})$ (
$(esEstudiante(i, j, A) \wedge sonVecinos(f, c, i, j, A) \wedge A.aula[f][c].ejercicios \neq \langle\rangle) \rightarrow_L$
$examenMuyParecido(i, j, f, c, A)$
)
}

pred examenMuyParecido($i : \mathbb{Z}, j : \mathbb{Z}, f : \mathbb{Z}, c : \mathbb{Z}, A$: EdR) {
$porcentajeEjerciciosParecidos(i, j, f, c, A) > \frac{3}{5} * |A.aula[f][c].ejercicios|$
}

pred examenParecidoAlResto($f : \mathbb{Z}, c : \mathbb{Z}, A$: EdR) {
$\frac{cantidadExamenesIguales(f, c, A)}{contarExamenes(A)} > \frac{contarExamenes(A)}{4}$
}

aux porcentajeEjerciciosParecidos($i, j, f, c : \mathbb{Z}, A$: EdR) : $\mathbb{R} = \frac{ejerciciosIguales(i, j, f, c, A)}{|A.aula[f][c].ejercicios|}$

aux contarExamenes(A : EdR) : $\mathbb{R} = \sum_{f,c=0}^{|A.aula|-1} IfThenElse(A.aula[f][c].entregado = True, 1, 0)$

aux ejerciciosIguales($i, j, f, c : \mathbb{Z}, A$: EdR) : $\mathbb{R} =$
$\sum_{k=0}^{|A.aula[f][c]|-1} IfThenElse(A.aula[f][c].ejercicios[k] = A.aula[i][j].ejercicios[k], 1, 0)$

aux cantidadExamenesIguales($f, c : \mathbb{Z}, A$: EdR) : $\mathbb{R} =$

$$\sum_{i,j=0}^{|A.aula|-1} IfThenElse((f \neq i \wedge c \neq j) \wedge ejerciciosIguales(i,j,f,c,A) = |A.aula[f][c].ejercicios|, 1, 0)$$

// Ejercicio 8

```
proc corregir(in A : EdR) : seq⟨⟨f : ℤ, c : ℤ⟩, ℝ⟩ {
    requiere { |res| > 0 }
    asegura {
        (∀f, c, n : ℤ) (
        ⟨⟨f, c⟩, n⟩ ∈ res ↔
        esEstudiante(f, c, A) ∧_L A.aula[f][c].entregado = True ∧
        ¬sospechoso(f, c, A)) ∧ n = calcularNota(A.aula[f][c], A.resuelto)
        )
    }
}
```

$$\texttt{aux calcularNota}(prueba, sol : \text{examen}) : \mathbb{R} = \frac{respuestasCorrectas(prueba.ejercicios, sol.ejercicios)}{|sol.ejercicios|}$$

$$\texttt{aux respuestasCorrectas}(s, sol : \text{seq}\langle\text{item}\rangle) : \mathbb{R} = \sum_{i=0}^{|s|-1} IfThenElse(s[i].opcion = sol[i].opcion, 1, 0)$$

}