DEPARTAMENTO
DE COMPUTACION
Facultad de Ciencias Exactas y Naturales - UBA

Trabajo Práctico de Especificación

Algoritmos y Estructuras de Datos II - Grupo 36

Integrante	LU	Correo electrónico
Sebastián Silvera	680/17	sebaok2011@gmail.com
Luciano Zinik	290/17	lzinik@gmail.com
Martín Funes	342/16	martinfunesfunes@gmail.com
Fernando Regert	282/15	fernandostds9@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

1. Decisiones tomadas y Notas

En este trabajo, tomamos la siguiente decisión:

- Definimos el mapa como una tupla de *nats* desde la celda $\langle 0, 0 \rangle$, de modo que sus valores puedan aumentar hasta el infinito en forma de damero.

Notas:

- En el **TAD** Mapa:
  - El generador «nuevoMapa» recibe como parámetro un conj(⟨nat, bool⟩), este conjunto define los ríos, que, citando el enunciado: «*corren en líneas infinitas horizontales o verticales*». Las tuplas de este conjunto definen lo siguiente: el segundo parámetro de la tupla, indica si el río se encuentra en una fila o una columna, y el primero indica el número de la fila o columna.:
    - true: Horizontal.
    - false: Vertical.

    Entonces, si es ⟨ 3, true ⟩, lo que nos dice es que toda la **fila** 3 de nuestra grilla, esta ocupada por un río. También, podemos tener ⟨ 4, false ⟩, lo que nos dice, es que toda la **columna** 4 de nuestra grilla, estará ocupada por un río.
  - La «otra operación» **casasADistManhattan** está axiomatizada sobre los generadores. Nos permitimos esto, ya que la información que nos brinda se puede deducir únicamente de los observadores, por lo cual podemos asegurar que no hay riesgo de romper la congruencia. Además, tomamos esta decisión porque que la información sea deducible de los observadores, no significa que esto pueda hacerse fácilmente, ya que habría que mirar todas las celdas a distancia 3 de la celda dada, y en dicho caso, preguntar si hay una casa.
- En el **TAD** Juego:
  - Para hacer frente al conflicto que se genera al intentar combinar dos juegos, en particular cuando haya dos construcciones que ocupen el mismo casillero, tomamos las siguientes decisiones.
    En primer lugar priorizaremos aquella que tenga el **nivel más alto.**
    En caso de tener el mismo nivel, las construcciones quedarían de la siguiente manera:
    - Casa Vs. Comercio: Casa
    - Casa Vs. Casa: Casa (es indistinto ya que ambas tienen el mismo nivel)
    - Comercio Vs. Comercio: Comercio (es indistinto ya que ambas tienen el mismo nivel)
  - En cuanto a la combinación de mapas, tomamos la siguiente decisión:
    - Baldio Vs. Río: Río.

2. TADs

TAD Celda **es** $\langle nat, nat \rangle$

TAD Terreno **es** String

TAD Mapa

géneros mapa

usa conj, dicc, nat

exporta observadores, generadores

igualdad observacional

$$(\forall\, m, m' : \text{mapa})\ \left(m =_{\text{obs}} m' \iff \begin{pmatrix} (\forall\, c : \text{celda}) \\ (\text{esRio?}(m,\, c) =_{\text{obs}} \text{esRio}(m',\, c)) \end{pmatrix} \right)$$

observadores básicos

esRio? : mapa × celda ⟶ bool

generadores

nuevoMapa : conj(⟨nat × bool⟩) ⟶ mapa

combinarMapas : mapa × mapa ⟶ mapa

otras operaciones

distanciaEntreCeldas : celda × celda ⟶ nat

rios : mapa ⟶ conj(⟨nat, bool⟩)

axiomas $\forall c, c1$: celda, $\forall rios$: conjunto(<nat, bool>)

esRio?(nuevoMapa(rios), c) ≡ **if** $\emptyset$ **then**
false
else
if (seg(dameUno(rios)) ∧
prim(dameUno(rios)) = prim(c)) ∨
(¬ seg(dameUno(rios)) ∧
prim(dameUno(rios)) = seg(c)) **then**
true
else
esRio(nuevoMapa(sinUno(rios), c))
fi
fi

esRio?(combinarMapas($m1$, $m2$), c) ≡ esRio?($m1$, c) ∨ esRio?($m2$, c)

distanciaEntreCeldas($c1$, $c2$) ≡ max(π_1(c1), π_1(c2)) - min(π_1(c1), π_1(c2))
+
max(π_2(c1), π_2(c2)) - min(π_2(c1), π_2(c2))

rios(nuevoMapa(c)) ≡ c

rios(combinarMapas($m1$, $m2$)) ≡ rios($m1$) ∪ rios($m2$)

Fin TAD

TAD Juego

géneros juego

igualdad observacional

$$(\forall\, j, j' : \text{juego}) \left(j =_{\text{obs}} j' \iff \begin{pmatrix} \text{turno}(j) =_{\text{obs}} \text{turno}(j') \ \wedge \\ \text{mapa?}(j) =_{\text{obs}} \text{mapa?}(j') \wedge \\ \text{hayConstNuevas?}(j) =_{\text{obs}} \text{hayConstNuevas?}(j') \wedge \\ (\forall c\text{: celda})(\text{queHay?}(j, c) =_{\text{obs}} \text{queHay?}(j', c) \wedge \\ \text{nivel}(j, c) =_{\text{obs}} \text{nivel}(j', c)) \wedge \\ \text{popularidad}(j) =_{\text{obs}} \text{popularidad}(j') \end{pmatrix} \right)$$

observadores básicos

turno	: juego	⟶ nat	
mapa?	: juego	⟶ mapa	
hayConstNuevas?	: juego	⟶ bool	
queHay?	: juego j × celda c	⟶ terreno	{¬ esRio(mapa?(j),c)}
nivel	: juego × celda	⟶ nat	
popularidad	: juego	⟶ nat	

generadores

nuevoJuego	: mapa	⟶ juego	
agregarCasa	: juego j × celda c	⟶ juego	{queHay?(j, c) = baldío}
agregarComercio	: juego j × celda c	⟶ juego	{queHay?(j, c) = baldío}
siguienteTurno	: juego j	⟶ juego	{hayConstNuevas?(j)}
combinarJuegos	: juego j1 × juego j2	⟶ juego	{sePuedenCombinarJuegos(construcciones($j1$), construcciones($j2$)}

otras operaciones

hayCasa?	: juego × celda	⟶ bool	
casasADist3	: juego × celda	⟶ conj(celda)	
maxNivel	: juego × conj(celda)	⟶ nat	
nivelManhattan	: juego j × celda c	⟶ nat	{queHay?(j, c) = comercio}
construcciones	: juego	⟶ conj(celda)	
construccionesDeMaxNivel	: juego × conj(celda)	⟶ conj(celda)	
sePuedeCombinarCelda	: juego × juego × celda	⟶ bool	
sePuedeCombinarJuegos	: juego × juego	⟶ bool	

axiomas $\forall c$: celda, $\forall j$: juego, $\forall m$: mapa

turno(nuevoJuego(m))	≡ 0
turno(agregarCasa(j, c))	≡ turno(j)
turno(agregarComercio(j, c))	≡ turno(j)
turno(siguienteTurno(j))	≡ turno(j) + 1
turno(combinarJuegos($j1$, $j2$))	≡ max(turno($j1$), turno($j2$))

mapa?(nuevoJuego(m))	≡ m
mapa?(agregarCasa(j, c))	≡ mapa?(j)
mapa?(agregarComercio(j, c))	≡ mapa?(j)
mapa?(siguienteTurno(j))	≡ mapa?(j)
mapa?(combinarJuegos($j1$, $j2$))	≡ combinarMapas(mapa?($j1$), mapa?($j2$))

hayConstNuevas?(nuevoJuego(m))	≡ false
hayConstNuevas?(agregarCasa(j, c))	≡ true
hayConstNuevas?(agregarComercio(j, c))	≡ true
hayConstNuevas?(siguienteTurno(j))	≡ false

hayConstNuevas?(combinarJuegos($j1$, $j2$)) $\equiv$ construcciones($j1$) $\neq$ construcciones($j2$)
queHay?(nuevoJuego(conj(n,h)) $\equiv$ baldio
queHay?(agregarCasa(j, $c1$), $c2$) $\equiv$ **if** c1=c2 **then** casa **else** queHay?(j, $c2$) **fi**

queHay?(agregarComercio(j, $c1$), $c2$) $\equiv$ **if** c1=c2 **then** comercio **else** queHay?(j, $c2$) **fi**

queHay?(siguienteTurno(j), c) $\equiv$ queHay?(j, c)

queHay(combinarJuegos($j1$, $j2$), c) $\equiv$ **if** nivel($j1$, c) > nivel($j2$, c) **then**
queHay?($j1$, c)
else
if nivel($j1$, c) < nivel($j2$, c) **then**
queHay?($j2$, c)
else
if queHay?($j1$, c) = queHay?($j2$, c) **then**
queHay?($j1$, c)
else
casa
fi
fi
fi

nivel(nuevoJuego(m), c) $\equiv$ 0

nivel(agregarCasa(j, c), $c1$) $\equiv$ **if** c=c1 **then** 0 **else** nivel(j, $c1$) **fi**

nivel(agregarComercio(j, c), $c1$) $\equiv$ **if** c=c1 **then**
nivelManhattan(j, $c1$)
else
nivel(j, $c1$)
fi

nivel(siguienteTurno(j), c) $\equiv$ nivel(j, c) + 1

nivel(combinarJuego($j1$, $j2$), c) $\equiv$ max(nivel($j1$, c), nivel($j2$, c))

popularidad(nuevoJuego(m)) $\equiv$ 0
popularidad(agregarCasa(j, c)) $\equiv$ popularidad(j)
popularidad(agregarComercio(j, c)) $\equiv$ popularidad(j)
popularidad(siguienteTurno(j)) $\equiv$ popularidad(j)
popularidad(combinarJuegos($j1$, $j2$)) $\equiv$ 1 + popularidad($j1$) + popularidad($j2$)

hayCasa?(j, c) $\equiv$ queHay?(j, c) = casa

casasADist3(nuevoJuego(conj(n, b)), c) $\equiv$ $\emptyset$

casasADist3(agregarCasa(j, c), $c1$) $\equiv$ **if** 0 < distanciaEntreCeldas(c, $c1$) $\leq$ 3
then
Ag(c, casasADist3(j, $c1$))
else
casasADist3(j, $c1$)
fi

casasADist3(agregarCasa(j, c), $c1$) $\equiv$ **if** $0 <$ distanciaEntreCeldas(c, $c1$) ≤ 3 **then**
Ag(c, casasADist3(j, $c1$))
else
casasADist3(j, $c1$)
fi

casasADist3(agregarComercio(j, c), $c1$) $\equiv$ casasADist3(j, $c1$)

casasADist3(siguienteTurno(j), c) $\equiv$ casasADist3(j, c)

maxNivel(j, A) $\equiv$ **if** $\emptyset?(A)$ **then**
0
else
max(nivel(j, dameUno(A)),
maxNivel(j, sinUno(A)))
fi

nivelManhattan(j, c) $\equiv$ maxNivel(j, casasADist3(j, c))

construcciones(nuevoJuego(m) $\equiv$ $\emptyset$

contrucciones(agregarCasa(j, c)) $\equiv$ Ag(c, contrucciones(j))

contrucciones(agregarComercio(j, c)) $\equiv$ Ag(c, construcciones(j))

contrucciones(siguienteTurno(j)) $\equiv$ construcciones(j)

construcciones(combinarJuegos($j1$, $j2$)) $\equiv$ construcciones($j1$) $\cup$ construcciones($j2$)

construccionesDeMaxNivel(j, C) $\equiv$ **if** $\emptyset?(C)$ **then**
$\emptyset$
else
if nivel(j, dameUno(C)) = maxNivel(j, C) **then**
Ag(dameUno(c), construccionesDeMaxNivel(j,
sinUno(C)))
else
construccionesDeMaxNivel(j, sinUno(C))
fi
fi

sePuedeCombinarCelda($j1$, $j2$, c) $\equiv$ $\neg$(
($c \in$ construccionesDeMaxNivel($j1$,construcciones($j1$))
$\wedge$ esConstruccion($j2$, c))
$\vee$
(esConstruccion($j1$, c) $\wedge$
$c \in$ construccionesDeMaxNivel($j2$,construcciones($j2$)))
$\vee$
(c $\in$ construcciones($j1$) $\wedge$ esRio?(mapa($j2$),c))
$\vee$
(c $\in$ construcciones($j2$) $\wedge$ esRio?(mapa($j1$), c))
)

sePuedenCombinarJuegos($j1$, $j2$) $\equiv$ **if** $\emptyset$?(construcciones($j1$)) $\vee$ $\emptyset$?(construcciones($j2$)) **then**
true
else
(sePuedeCombinarCelda($j1$, $j2$, dameUno(construcciones($j1$))) $\wedge$
sePuedenCombinarJuegos($j1$, $j2$, sinUno(construcciones($j1$)))
$\wedge$
(sePuedeCombinarCelda($j1$, $j2$, dameUno(construcciones($j2$))) $\wedge$
sePuedenCombinarJuegos($j1$, $j2$, sinUno(construcciones($j2$)))
fi

Fin TAD