DEPARTAMENTO
DE COMPUTACION
Facultad de Ciencias Exactas y Naturales - UBA

Trabajo Práctico de Especificación

Buscaminas

23/09/22 Algoritmos y Estructuras de Datos 1

Integrante	LU	Correo electrónico
Lamonica, Ivo	66/22	ivolamonicam@gmail.com
Neville, Matías	88/22	nevillematias@gmail.com
Sanguinetti, Ivan	331/22	ivan.sanguinetti18@gmail.com
Schiro, Mateo	657/22	mateo.schiro8@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

Índice

1. Ejercicio 1

`pred estáEnTablero` (p: pos, t: tablero) {
$esTableroVálido(t) \wedge (0 \leq p_0 < |t|) \wedge (0 \leq p_1 < |t|)$
}

`pred esTableroVálido` (t: tablero) {
$|t| < 0 \wedge_L (\forall i : \mathbb{Z})(0 \leq i < |t| \longrightarrow_L |t[i]| = |t|)$
}

`aux minasAdyacentes` (t: tablero, p: pos) : $\mathbb{Z}$ =
$\sum_{i=-1}^{1}\sum_{j=-1}^{1}$if $estáEnTablero((p_0+i, p_1+j), t) \wedge ((p_0+i, p_1+j) \,!= p) \wedge_L (hayMina(t, (p_0+i, p_1+j)))$ then 1 else 0 fi ;

`pred hayMina` (t: tablero, p: pos) {
$t[p_0][p_1] = true$
}

Este auxiliar realiza una sumatoria doble para acceder a las nueve posiciones alrededor (e inclusive) de la posición p ingresada. Para cada casilla recorrida, se fija lo siguiente: que la casilla se encuentre en el tablero, que sea distinta a la posición p, y que en esa posición recorrida, haya una mina en el tablero. Si cumple estas tres condiciones, se sumará +1 con tal de contar las minas adyacentes.

Respecto al predicado, tomando una posición p y el tablero como entrada, este simplemente se fijará que el tablero tenga al menos una casilla, que ambas de las coordenadas de la posición por separado sean mayores o iguales a cero, y que no se encuentren por fuera de los límites del mismo.

2. Ejercicio 2

`pred juegoVálido` (t: tablero, j: jugadas) {
$(\forall i : \mathbb{Z})(0 \leq i < |j| \longrightarrow_L estáEnTablero(j[i]_0, t) \wedge j[i]_1 = minasAdyacentes(t, j[i]_0) \wedge$
$(\forall k : \mathbb{Z})(0 \leq k < |j| \longrightarrow_L (j[i]_0 \neq j[k]_0 \wedge i \neq k)) \wedge \neg(\exists n : \mathbb{Z})(0 \leq n < |j| - 1 \wedge_L hayMina(t, j[n]_0))$
}

Tomando la secuencia de jugadas y el tablero, un juego se considerará válido si se cumplen las siguientes condiciones: todas las jugadas están dentro del tablero; en cada jugada, la cantidad de minas adyacentes es la correspondiente; y ninguna jugada se encuentra repetida.

3. Ejercicio 3

`proc plantarBanderita` (in t: tablero, in j: jugadas, in p: pos, inout b: banderitas) {
`Pre` $\{(estaEnTablero(p, t)) \wedge (banderitaValida(b, t)) \wedge (\neg estáDescubierto(p, j)) \wedge (b = B_0) \wedge (\neg tieneBanderita(p, b))\}$
`Post` $\{|b| = |B_0| + 1 \wedge tieneBanderita(p, b) \wedge (\forall i : \mathbb{Z})(0 \leq i < |B_0| \longrightarrow_L tieneBanderita(B_0[i], b))\}$
}

`pred estáDescubierto` (p: pos, j: jugadas) {
$(\exists i : \mathbb{Z})(0 \leq i < |j| \wedge_L p = j[i]_0)$
}

`pred tieneBanderita` (p: pos, b: banderitas) {
$(\exists i : \mathbb{Z})(0 \leq i < |b| \wedge_L b[i] = p)$

}

pred banderitaValida (b: banderita, t: tablero) {
$(\forall i : \mathbb{Z})(0 \leq i < |b| \longrightarrow_L estaEnTablero(b[i], t) \wedge_L \neg(\exists k : \mathbb{Z})(0 \leq k < |b| \wedge k \neq i \wedge_L b[k] = b[i]))$
}

Como precondición en este procedimiento, pedimos que la posición en la que se desea plantar una banderita se encuentre dentro del tablero, que no sea ya una casilla jugada, y que no haya una banderita previamente colocada. Si se cumplen dichas condiciones, aseguramos que la nueva secuencia de banderitas va a ser igual a la de entrada, pero agregandole la posición con la nueva banderita.

Respecto a los últimos dos predicados, se especifica que dada una posición p y una secuencia de tipo jugadas y una secuencia de tipo banderitas, respectivamente, que exista algun elemento entre estas que sea igual a p.

4. Ejercicio 4

proc perdió (in t: tablero, in j: jugadas, out res: Bool) {
Pre $\{juegoVálido(t, j)\}$
Post $\{res = true \leftrightarrow hayMinaEnÚltima(t, j)\}$
}

pred hayMinaEnÚltima (t: tablero, j: jugadas) {
$hayMina(t, j[|j| - 1]_0)$
}

Tomando como precondición que el juego es válido, el procedimiento devolverá true si el jugador ha perdido, fijandose si hay una mina en la última posición jugada. Nuestra interpretación a la secuencia de jugadas es la siguiente: luego de hacer un "clic" en alguna casilla permitida, se agrega dicha posición a la secuencia de jugadas, y luego se ejecuta el procedimiento mencionado. Como este proceso se realiza jugada a jugada, puede darse el caso en que la última jugada realizada haya caído en una mina, que es lo que detecta el procedimiento Perdió.

Respecto al predicado, dada una secuencia de jugadas y el tablero, se especifica que si hay una mina en la posición de la última jugada realizada, sea verdadero.

5. Ejercicio 5

proc ganó (in t: tablero, in j: jugadas, out res: Bool) {
Pre $\{juegoVálido(t, j)\}$
Post $\{res = true \leftrightarrow (|j| = casillasSinBombas(t)) \wedge \neg hayMinaEnÚltima(t, j)\}$
}

aux casillasSinBombas (t:tablero) : $\mathbb{Z} = |t| * |t| - cantidadBombas(t)$;

aux cantidadBombas (t: tablero) : $\mathbb{Z} = \sum_{i=0}^{|t|-1} \sum_{j=0}^{|t|-1}$ if $(t[i][j] = true)$ then 1 else 0 fi ;

Tomando de vuelta que el juego es válido, consideramos que un jugador habrá ganado si la cantidad de jugadas realizadas es igual a la cantidad de casillas en el tablero que no poseen minas, y que la última jugada realizada no haya sido sobre una mina.

Respecto a las auxiliares, se calcula que la cantidad de casillas sin minas es igual a la diferencia entre la cantidad de casillas totales y la cantidad de casillas con minas. Para determinar la cantidad total de minas, se utiliza una doble sumatoria para recorrer cada casilla dentro del tablero, y se suma +1 al total si en dicha casilla hay una mina.

6. Ejercicio 6

```
proc jugar (in t: tablero, in b: banderitas, in p: pos inout j: jugadas) {
```
Pre $\{juegoEnCurso(t, j) \land (j = J_0) \land \neg tieneBanderita(p, b) \land \neg estáDescubierto(p, j) \land_L$
$(\forall i : \mathbb{Z})(0 \leq i < |j| \longrightarrow_L \neg tieneBanderita(j[i]_0, b))\}$
Post $\{j = concat(J_0, < (p, minasAdyacentes(t, p)) >)\}$
}

```
pred juegoEnCurso (t: tablero, j: jugadas) {
```
$|j| < casillasSinBombas(t) \land_L \neg hayMinaEnÚltima(t, j) \land juegoVálido(t, j)$
}

Tomando como precondición que el juego aún está en curso, que la jugada a realizar no posee una banderita y que tampoco haya sido previamente jugada, en el procedimiento se especifica como postcondición que, al salir, jugadas sea igual a agregarle la posición de la nueva jugada a la lista de jugadas realizadas preivamente, junto con su cantidad de minas adyacentes.

Respecto al predicado, consideramos que un juego está en curso cuando no hay una mina en la última jugada realizada, y la cantidad de jugadas totales es menor a la cantidad de casilleros sin bombas (lo que significaría victoria).

7. Ejercicio 7

```
pred caminoLibre (t: tablero, P0: pos, P1: pos) {
```
$(minasAdyacentes(t, P0) = 0 \land minasAdyacentes(t, P1) \geq 1) \longrightarrow (\exists l : seq\langle pos\rangle)(\forall i : \mathbb{Z})(0 \leq i < |l| - 1 \longrightarrow$
$((sonAdyacentes(l[i], l[i{+}1])) \land (juegoVálido(t, < (l[i], 0) >)) \land (juegoVálido(t, < (l[|l|{-}1], 0) >)) \land (sonAdyacentes(P0, l[0])) \land$
$(sonAdyacentes(P1, l[|l| - 1])))$
}

```
pred sonAdyacentes (s: pos, t: pos) {
```
$(\exists i, l : \mathbb{Z})(((-1 \leq i \leq 1) \land (-1 \leq j \leq 1)) \land_L ((s_0 = t_0 + i) \land (s_1 = t_1 + j) \land (s \neq t)))$
}

Dadas dos posiciones p1 y p2 para las cuales p1 no tiene minas adyacentes, y p2 tiene al menos una, se considerará que existe un camino libre entre ellas si existe una secuencia l de posiciones tales que: todas son adyacentes a la posición siguiente, todas tienen 0 minas adyacentes, y las posiciones en los extremos son adyacentes a p1 y p2, respectivamente.
Para el predicado, dadas dos posiciones p1 y p2, las mismas se consideran adyacentes si son distintas, y una se encuentra en las 8 casillas adyacentes de la otra

8. Ejercicio 8

```
proc jugarPlus (in t: tablero, in b: banderitas, in p: pos, inout j: jugadas) {
```
Pre $\{juegoEnCurso(t, j) \land (j = J_0) \land \neg tieneBanderita(p, b) \land \neg estáDescubierto(p, j)\}$
Post $\{juegoVálido(j, t) \land J_0 = subseq(j, 0, |J_0|) \land incluyeCaminoLibre(t, p, j, J_0) \land noIncluyeCasillasExtra(t, p, j, J_0) \land$
$estáDescubierta(p, j)\}$
}

pred incluyeCaminoLibre (t: tablero, p: pos,j: jugadas, J0 : jugadas) {
$(\forall z : pos)(estáEntablero(z,t) \wedge \neg estáDescubierta(z, J_0) \wedge caminoLibre(t,p,z) \longrightarrow_L (\forall l : seq < pos >)(sonAdyacentes(p, l[0]) \wedge$
$sonAdyacentes(z, l[|l|-1]) \wedge (\forall i : \mathbb{Z})(0 \leq i < |l| - 1 \longrightarrow_L sonAdyacentes(l[i], l[i+1]) \wedge$
$(\forall n : \mathbb{Z})(0 \leq n < |l| \longrightarrow_L (minasAdyacentes(t, l[n]) = 0)) \longrightarrow_L (\forall k : \mathbb{Z})(0 \leq k < |l| \longrightarrow_L estáDescubierta(l[k], j)))))$
}

pred noIncluyeCasillasExtra (t: tablero, p: pos, J0,j: jugadas) {
$\neg(\exists i : \mathbb{Z})(|J_0| \leq i < |j| \wedge_L j[i]_0 \neq p \wedge_L \neg caminoLibreAux(t,p,j[i]_0))$
}

pred caminoLibreAux (t: tablero, P0: pos, P1: pos) {
$(minasAdyacentes(t, P0) = 0) \longrightarrow (\exists l : seq\langle pos\rangle)(\forall i : \mathbb{Z})(0 \leq i < |l| - 1 \longrightarrow$
$((sonAdyacentes(l[i], l[i+1])) \wedge (juegoVálido(t, < (l[i], 0) >)) \wedge (juegoVálido(t, < (l[|l|-1], 0) >)) \wedge (sonAdyacentes(P0, l[0])) \wedge$
$(sonAdyacentes(P1, l[|l| - 1])))$
}

Como precondicion del procedimiento, se especifica lo mismo que en ejercicio 6, pues la situación de juego es la misma. En la post, en el caso en que no exista un camino libre entre la posición p ingresada, y alguna otra posicion z (que jugaría el rol de P1 del predicado caminoLibre), simplemente se se comportará igual que en el ejercicio 6, y sumará la posición jugada a la lista de jugadas, junto con su cantidad de minas adyacentes. Caso contrario, existirá un posible camino libre, y se especifica que en jugadas aparezcan todas estas secuencias de posiciones descubiertas automaticamente, con el fin de descubrir la mayor cantidad de casillas posibles, sin repeticiones.

El predicado incluyeCaminoLibre es el que especifica que todas las secuencias l que cumplen los requisitos de ser caminos libres, aparecerán en la secuencia de jugadas fijándose mediante juegoValido.

9. Ejercicio 9

proc sugerirAutomático121 (in t: tablero, in b: banderitas, in j: jugadas, out p: pos) {
Pre $\{juegoEnCurso(t,j) \wedge (\exists x, y, z : pos)(están Horizontal121(x,y,z,t,j,b) \vee están Vertical121(x,y,z,t,j,b)\}$
Post $\{(\exists x, y, z : pos)((están Horizontal121(x,y,z,t,j,b) \longrightarrow_L (p = (y_0 + 1, y_1)) \vee (p = (y_0 - 1, y_1)) \wedge$
$(están Vertical121(x,y,z,t,j,b) \longrightarrow_L (p = (y_0, y_1+1) \vee (p = (y_0, y_1-1)) \wedge (\neg estaDescubierta(p,j) \wedge estaEntablero(p,t)))\}$
}

pred estánHorizontal121 (x,y,z: pos, t: tablero, j: jugadas, b: banderitas) {
$(estáDescubierta(x,j) \wedge_L estáDescubierta(y,j) \wedge_L estáDescubierta(z,j)) \wedge$
$juegoVálido(t, < (x,1) >) \wedge juegoVálido(t, < (y,2) >) \wedge juegoVálido(t, < (z,1) >) \wedge (x_0 = y_0) \wedge (y_0 = z_0) \wedge$
$(x_1 = y_1 - 1) \wedge (y_1 = z_1 - 1) \wedge ((\neg estáDescubierta((y_0 - 1, y_1), j)) \wedge \neg tieneBanderita((y_0 - 1, y_1), b)) \vee$
$((\neg estáDescubierta((y_0 + 1, y_1), j)) \wedge \neg tieneBanderita((y_0 + 1, y_1), b))$
}

pred estánVertical121 (x,y,z: pos, t: tablero, j: jugadas, b: banderitas) {
$(estáDescubierta(x,j) \wedge_L estáDescubierta(y,j) \wedge_L estáDescubierta(z,j)) \wedge$
$juegoVálido(t, < (x,1) >) \wedge juegoVálido(t, < (y,2) >) \wedge juegoVálido(t, < (z,1) >) \wedge (x_1 = y_1) \wedge (y_1 = z_1) \wedge$
$(x_0 = y_0 - 1) \wedge (y_0 = z_0 - 1) \wedge ((\neg estáDescubierta((y_0, y_1 - 1), j)) \wedge \neg tieneBanderita((y_0, y_1 - 1), b)) \vee$
$((\neg estáDescubierta((y_0, y_1 + 1), j)) \wedge \neg tieneBanderita((y_0, y_1 + 1), b))$
}

Se especifica como precondición que exista al menos tres casillas adyacentes de manera horizontal o vertical, con el fin de que el juego pueda devolver esta posición p sugerida como jugada segura. Respecto a la postcondición, p será la posición que se encuentra arriba o abajo, o a la izquierda o a la derecha (dependiendo la disposición del 1-2-1) de la casilla que tiene dos minas adyacentes, tal que sea una posición no descubierta dentro del tablero, y que sea segura de jugar.
Los predicados son los que especifican que existan estas tres posiciones adyacentes (estrictamente al lado, pues en diagonal no cuentan), y que la casilla que luego será sugerida en el procedimiento no esté descubierta ni tampoco tenga banderita, pues sinó no podrá clickearse esa posición.