ISIS 2103 Diseño de Algoritmos
Semestre 2010-1 - Parcial No. 1
Prof. Rodrigo Cardoso
Marzo 4, 2010

1 [30 puntos]

1a **[15/30]** El significado de `S1;S2`, en la semántica axiomática de Dijkstra, se establece al definir la precondición más débil correspondiente:

`wp(S1;S2|R) ≡ wp(S1|wp(S2|R))`   `[D;]`

Por otro lado, el Cálculo de Hoare propone la regla de inferencia:

$$\frac{\{Q\}\ S1\ \{R1\},\ \{R1\}\ S2\ \{R\}}{\{Q\}\ S1;S2\ \{R\}} \qquad [H;]$$

Muestre que la adecuación de `[H;]` se puede probar a partir de `[D;]`, i.e., si se suponen `[D;]` y las hipótesis de `[H;]` se puede deducir la conclusión de `[H;]`.

AYUDA: Necesitará la Ley de Monotonía de la `wp`.

Dem:

```
Hip: [D;], {Q}S1{R1}, {R1}S2{R}
   true
=     〈 Hip: {R1}S2{R} 〉
   {R1}S2{R}
=     〈 Def corrección 〉
   R1 ⇒ wp(S2|R)
⇒     〈 Monotonía wp 〉
   wp(S1|R1) ⇒ wp(S1|wp(S2|R))
⇒     〈 {Q}S1{R1} ≡ (Q ⇒ wp(S1|R1)); Transitividad 〉
   Q ⇒ wp(S1|wp(S2|R))
=     〈 Hip: [D;] 〉
   Q ⇒ wp(S1;S2|R)
=     〈 Def corrección 〉
   {Q} S1;S2 {R}
```

[15/30]

1b **[15/30]** Explique por qué, en general, no es calculable `wp(S|R)`, con `S` una instrucción y `R` una condición.

Porque se podría calcular, para cualquier programa, `wp(S|true)`, el conjunto de estados a partir de los cuales el programa `S` se ejecuta y termina normalmente (en `true`). Es decir, se tendría resuelto el problema de la parda, que es indecidible.

[15/30]

2 [30 puntos]

Con la notación, para $0 \le k \le n$:

`lasml(k)` = "longitud del subarreglo ascendente más largo en `b[0..k-1]`"
`lasmlf(k)` = "longitud del subarreglo ascendente más largo en `b[0..k-1]` que finaliza en `b[k-1]`"

considere un programa GCL de la forma:

```
[ Ctx C:  n≥1 ∧ b: array[0..n-1] of nat

  INIC;
```

```
 {Inv P : 0≤k≤n ∧ x=lsaml(k) ∧ y=lsamlf(k) }
 {Cota t: y}

 do B  →    S     od

 {R: x = lsaml(n)}
]
```

2a **[15/30]** Escriba código GCL (INIC, B, S) que satisfaga la especificación dada.

```
[ Ctx C:  n≥1 ∧ b: array[0..n-1] of nat

 k,x,y:= 1,1,1;

 {Inv P : 0≤k≤n ∧ x=lsaml(k) ∧ y=lsamlf(k) }
 {Cota t: y}

 do k≠n      →     if b[k-1] ≤ b[k]  →     y:= y+1
                   [] b[k-1] > b[k]  →     y:= 1
                   fi;
                   if y>x      → x,k:= y,k+1
                   [] y≤x      → k:= k+1
                   fi
 od

{R: x = lsaml(n)}
]
```

[15/30]

Variantes

Cambios en S, por ejemplo:

```
      if b[k-1] ≤ b[k]  →     y:= y+1;
                              if y>x      → x:= y
                              [] y≤x      → skip
                              fi
      [] b[k-1] > b[k]  →     y:= 1
      fi;
      k:= k+1
```

[15/30]

2b **[8/30]** Indique qué técnica(s) puede(n) explicar la definición del invariante P a partir de la especificación dada.

El invariante resulta de cambiar la constante n por la variable k, 0≤k≤n. Además, es necesario fortalecer el invariante ("poner a trabajar al invariante") con la condición adicional y=lsamlf(k).

[8/30]

2c **[7/30]** Cuente asignaciones como operaciones básicas y estime el orden de complejidad de su solución (como θ(...)). Explique su respuesta.

Sea `T(n)` el número de asignaciones, en el peor caso. Entonces:

$T(n) = T_{INIC}(n) + n*T_S(n)$

donde

$T_{INIC}(n) = 3$

$T(S) = T(IF1) + T(IF2) = 1+2 = 3$

Así:

$T(n) = 3 + 3*n = \theta(n)$.

[7/30]

3 **[40/100]**

Se quiere resolver el problema de decidir si una matriz

`b[0..m-1,0..n-1]`**of nat**

es ascendente en cada fila.

3a **[5/40]** Especifique el problema (contexto, pre y poscondición).

```
Ctx C: b[0..m-1,0..n-1]of nat
Pre Q: true
Pos R: af = "b es ascendente en cada fila"
```

[5/40]

3b **[25/40]** Desarrolle una solución para el problema especificado en **3a**. Para esto, describa:

i *Plan de desarrollo*: una explicación esquemática que indique en qué consiste su solución (v.gr., "un ciclo para lograr …"; "una asignación para lograr …"; "un paradigma X para lograr …"; …).

ii *Código GCL*.

Documente todo ciclo o paradigma que utilice.

Advertencia: La calificación dependerá de la complejidad de la solución desarrollada (una solución ineficiente será calificada de manera inferior).

i *Plan de desarrollo*

BLI de una fila desordenada

BLI de un desorden en una fila

Si se encuentra el desorden en la fila, indicar fila desordenada.

en otro caso (b asc. en fila) continuar.

Si se encuentra fila desordenada, indicar desorden en filas

en otro caso (b asc. en toda fila), continuar.

Si se encontró algún desorden, responder no; en otro caso, responder sí.

[5/25]

ii *Código GCL*

Para BLI en filas:

E = [0..m-1]

$e_0 = 0$

$e_{ult} = m$

suc.i = i+1

p(i) ≡ "filas 0,1,…,i-1" ordenadas

Para BLI en fila i:

E = [1..n-1]

$e_0 = 1$

$e_{ult} = n$

suc.j = j+1

```
p(j) ≡ b[i,j-1]> b[i,j]
```

[5/25]

```
[  Ctx C: b[0..m-1,0..n-1]of nat
   {Pre Q: true}

   i,icent:= 0,m;
   do i≠icent     →    j,jcent:= 1,n;
                       do j≠jcent  →    if b[i,j-1]>b[i,j] → jcent:= j
                                        [] b[i,j-1]≤b[i,j] → j:= j+1
                                        fi
                       od;
                       if j=n →   i:= i+1
                       [] j≠n →   icent:= i
                       fi
   od;
   af:= (i=m)

   {Pos R: af = "b es ascendente en cada fila"}
]
```

[15/25]

Variantes

Pueden no usarse paradigmas BLI. En ese caso debe incluirse invariantes en cada uno de los ciclos.

3c **[10/40]** Cuente asignaciones como operaciones básicas y estime el orden de complejidad de su solución (como $\theta(\ldots)$). Explique su estimación.

Hay que hacer n-1 comparaciones en cada una de m, filas, en el peor caso. Es decir:

$$T(m,n) = m*(n-1) = \theta(mn)$$

[10/10]