CRA. Tema 5: Semántica y verificación de programas (tercera parte)

José Enrique Morais San Miguel

Universidad de Alcalá

Ejercicio 4 del examen de 8 de mayo de 2014

Demostrar que el siguiente programa, que toma como entrada dos enteros m y n, con $m \geq 0$ y $n > 0$, calcula el cociente de la división euclídea de m por n (corrección parcial):

$$
\begin{array}{l}
\{m \geq 0,\ n > 0\} \\
\quad x := m;\ z := 0 \\
\qquad \texttt{mientras}\ (x \geq n)\ \texttt{hacer} \\
\qquad\quad z := z + 1;\ x := x - n \\
\qquad \texttt{fmientras} \\
\{z = coc(m, n)\}
\end{array}
$$

Ejercicio 4 del examen de 8 de mayo de 2014

$\{m \geq 0,\ n > 0\}$
  $x := m;\ z := 0$
    `mientras` $(x \geq n)$ `hacer`
      $z := z + 1;\ x := x - n$
    `fmientras`
$\{z = coc(m, n)\}$

Ejercicio 4 del examen de 8 de mayo de 2014

$\{m \geq 0,\ n > 0\}$
$\quad x := m;\ z := 0$
$\quad\quad$ `mientras` $(x \geq n)$ `hacer`
$\quad\quad\quad z := z + 1;\ x := x - n$
$\quad\quad$ `fmientras`
$\{z = coc(m, n)\}$

Invariante: $I \equiv (zn + x = m)$

$$wp(W, I) = wp(z := z + 1;\ x := x - n, I) =$$

$$wp(z := z + 1, zn + x - n = m) = ((z + 1)n + x - n = m)$$

$$\equiv (zn + x = m) \equiv I$$

Ejercicio 5 del examen de 14 de mayo de 2014

Demostrar que el siguiente programa, que toma como entrada un entero a y una lista de enteros L, devuelve una lista que es la lista L una vez eliminadas todas las apariciones de a en la misma (corrección parcial) (NOTA: Se asume que se dispone de las funciones correctas `reverse`, `cons`, `hd` y `tl`):

$\{T\}$

$L1$:=`reverse`(L); $L2$:= `nil`;

`mientras` ($L1 \neq$ `nil`) `hacer`

`si` `hd`($L1$) $= a$ `entonces` $L1$:= `tl`($L1$)

`si no` $L2 :=$ `cons` (`hd`($L1$),$L2$); $L1$:=`tl`($L1$)

`fsi`

`fmientras`

$\{L2 = elim(a, L)\}$

Ejercicio 4 del examen de 14 de mayo de 2014

$\{T\}$
```
  L1:=reverse(L); L2:= nil;
    mientras (L1 ≠ nil) hacer
      si hd(L1) = a entonces L1:= tl(L1)
        si no L2 := cons (hd(L1),L2);  L1:=tl(L1)
      fsi
    fmientras
```
$\{L2 = elim(a, L)\}$

$\{T\}$
```
   L1:=reverse(L);  L2:= nil;
      mientras (L1 ≠ nil) hacer
         si hd(L1) = a entonces L1:= tl(L1)
            si  no L2 := cons (hd(L1),L2);  L1:=tl(L1)
         fsi
      fmientras
```
$\{L2 = elim(a, L)\}$

Invariante: $I \equiv app(rev(elim(a, L1)), L2) = elim(a, L)$

$$wp(W, I) = (hd(L1) = a \wedge wp(L1 := tl(L1), I)) \vee$$

$$\vee (hd(L1) \neq a \wedge wp(L2 := cons(hd(L1), L2) \; ; L1 := tl(L1), I) =$$

$$= (hd(L1) = a \wedge app(rev(elim(a, tl(L1)), L2) = elim(a, L)) \vee$$

$$\vee (hd(L1) \neq a \wedge app(rev(elim(a, tl(L1))), cons(hd(L1), L2)) = elim(a, L)) \equiv$$

$$\equiv (hd(L1) = a \wedge I) \vee (hd(L1) \neq a \wedge I) \equiv I$$

Ejercicio 4 del examen de 14 de mayo de 2014

En lo anterior hemos hecho de las siguientes propiedades:

a) Si $hd(L) = a$, entonces $elim(a, L) = elim(a, tl(L))$
b) Si $hd(L) \neq a$, entonces $elim(a, L) = cons(hd(L), elim(a, tl(L)))$
c) $append(L, cons(x, L1)) = append(append(L, cons(x, nil)), L1)$
d) $append(reverse(L), cons(x, nil)) = reverse(cons(x, L))$

Por ejemplo:

$append(reverse(elim(a, tl(L1))), cons(hd(L1), L2))$ es igual a $append(append(reverse(elim(a, tl(L))), cons(hd(L1), nil)), L2)$ por la propiedad c). Aplicando la propiedad d), esta última expresión es igual a $append(reverse(cons(hd(L1), elim(a, tl(L1)))), L2)$ que, en el caso de que $hd(L) \neq a$, es igual a $append(reverse(elim(a, L1), L2)$ por la propiedad b).

Ejercicio 4 del examen de 21 de mayo de 2018

Demostrar que el siguiente programa, que toma como entrada dos listas l_1 y l_2 (la segunda sin repeticiones), devuelve una lista cuyos elementos constituyen la unión (conjuntista) de las listas l_1 y l_2 (corrección parcial) (NOTA: Se asume que se dispone de las funciones correctas `pertenece` que verifica si un elemento pertenece a una lista, `cons`, `hd` y `tl`):

$\{l_2 \; \textit{sin repeticiones}\}$
$\quad L1 := l_1; \; L2 := l_2;$
$\quad\quad$ `mientras` $(L1 \neq \texttt{nil})$ `hacer`
$\quad\quad\quad$ `si pertenece(hd(`$L1$`),`$L2$`) entonces` $L1 := \texttt{tl}(L1)$
$\quad\quad\quad\quad$ `si no` $L2 := \texttt{cons}\,(\texttt{hd}(L1), L2); \quad L1 := \texttt{tl}(L1)$
$\quad\quad\quad$ `fsi`
$\quad\quad$ `fmientras`
$\{L2 = \textit{union}(l_1, l_2)\}$

Ejercicio 4 del examen de 21 de mayo de 2018

$\{l_2 \textit{ sin repeticiones}\}$

$L1 := l_1;\ L2 := l_2;$

mientras $(L1 \neq \texttt{nil})$ hacer

si pertenece(hd($L1$), $L2$) entonces $L1$:= tl($L1$)

si no $L2$:= cons (hd($L1$),$L2$); $L1$:=tl($L1$)

fsi

fmientras

$\{L2 = union(l_1, l_2)\}$

$\{l_2$ *sin repeticiones*$\}$
```
    L1 := l1; L2 := l2;
        mientras (L1 ≠ nil) hacer
            si pertenece(hd(L1), L2) entonces L1:= tl(L1)
                si no L2 := cons (hd(L1),L2);  L1:=tl(L1)
            fsi
        fmientras
```
$\{L2 = union(l_1, l_2)\}$

Invariante: $I \equiv (union(L1, L2) = union(l_1, l_2))$.

Para probar que I es un invariante basta considerar cuál es la definición recursiva de la unión:

$$union(L_1, L_2) = \begin{cases} L_2 & \text{si} \quad L_1 = nil \\ union(tl(L_1), L_2) & \text{si} \quad hd(L_1) \in L_2 \\ union(tl(L_1), cons(hd(L_1), L_2)) & \text{si} \quad hd(L_1) \notin L_2 \end{cases}$$

Derivación de programas: Raíz cuadrada entera

Desarrollaremos dos métodos para encontrar la raíz cuadrada entera de un entero no negativo, es decir, partimos del aserto:

$$\{0 \leq a\}\ S\ \{0 \leq x^2 \leq a < (x+1)^2\}.$$

Usaremos un bucle para calcular valores de `x` hasta que se verifique la postcondición. Supongamos que tomamos la primera parte de la postcondición como invariante, $p \equiv \{0 \leq x^2 \leq a^2\}$, y no tratamos de verificar la segunda parte hasta que acabe el bucle.

$\{0 \leq a\}$

```
x:=?;
mientras B(x,a) hacer
```

$\quad\{0 \leq x^2 \leq a^2\}$

```
   x:= ?;
fmientras
```

$\{0 \leq x^2 \leq a < (x+1)^2\}.$

Derivación de programas: Raíz cuadrada entera

Dada la precondición $0 \leq a$, necesariamente la primera expresión ha de ser `x:=0`. La postcondición del bucle es $p \wedge \neg$`B(x,a)`, por lo que `B(x,a)` debe escogerse para que se verifique la postcondición: `B(x,a)` ha de ser la negación de $a < (x+1)^2$, es decir, `B(x,a)` es $\texttt{(x+1)}^2 \leq \texttt{a}$. Finalmente, como el bucle termina cuando `x` es suficientemente grande, en el cuerpo del bucle podemos incrementar el valor de `x` en una unidad. Se tiene, pues el programa (anotado):

$\{0 \leq a\}$
`x:=0;`
`mientras` $\texttt{(x+1)}^2 \leq \texttt{a}$ `hacer`
  $\{0 \leq x^2 \leq a\}$
  `x:= x+1;`
`fmientras`
$\{0 \leq x^2 \leq a < (x+1)^2\}$.

Para ver la corrección parcial del programa se debe probar que $\{p \land B\}\ S\ \{p\}$, es decir,

$$\{0 \leq x^2 \leq a \land (x+1)^2 \leq a\}\ \texttt{x:=x+1}\{0 \leq x^2 \leq a\}.$$

Ahora, tenemos

$$\{0 \leq (x+1)^2 \leq a\}\ \texttt{x:=x+1}\{0 \leq x^2 \leq a\},$$

pero

$$(0 \leq x^2 \leq a \land (x+1)^2 \leq a) \to (0 \leq (x+1)^2 \leq a),$$

así que el hecho de que p es invariante se sigue de la regla de la consecuencia.

Derivación de programas: Raíz cuadrada entera

La derivación de un programa está determinada, normalmente, por la forma en la que se escribe la postcondición. Escribamos la postcondición introduciendo una nueva variable:

$$(0 \leq x^2 \leq a < y^2) \wedge (y = x + 1).$$

La introducción de la nueva variable hace que el bucle deba afectar a ambas variables. Como en la postcondición se tiene $y = x + 1$, esta es la condición que finalizará el bucle. El valor de `y` debe ser siempre mayor que el de `x`. Además, no tiene sentido que el valor de `y` sea mayor que `a+1`, luego inicializamos `y := a+1` y añadimos la condición $x < y \leq a + 1$ al invariante y se obtiene el candidato a invariante

$$p = (0 \leq x^2 \leq a < y^2) \wedge (x < y \leq a + 1)$$

El bosquejo del programa (anotado) queda así:

$\{0 \leq a\}$

```
x:=0; y:=a+1;
mientras y≠x+1 hacer
```

$\quad\{(0 \leq x^2 \leq a < y^2) \wedge (x < y \leq a+1)\}$

```
   ?;
fmientras
```

$\{0 \leq x^2 \leq a < (x+1)^2\}.$

Ahora, en el bucle podemos incrementar el valor de `x` o reducir el valor de `y` siempre que se mantenga el invariante. Llamemos `z:=(x+y) div 2`. El valor de `z` se lo debemos asignar a `y` o a `x` pero manteniendo el invariante. Con estas consideraciones, el bucle debería verificar $\{p \wedge B\}\ S1\{p\}$. Por lo tanto, el bucle (anotado) debería ser de la forma

$\{p \wedge (y \neq x+1)\}$
```
z:=(x+y) div 2;
```
$\{p \wedge (y \neq x+1) \wedge (z = \lfloor (x+y)/2 \rfloor)\}$
```
si Cond(x,y,z) entonces
```
$\{p\{x \leftarrow z\}\}$
```
   x:=z;
   si no
```
$\{p\{y \leftarrow z\}\}$
```
      y:=z;
fsi
```
$\{p\}$

Derivación de programas: Raíz cuadrada entera

La condición `Cond(x,y,z)` debe escogerse tal que verifique, con $p = (0 \leq x^2 \leq a < y^2) \wedge (x < y \leq a+1))$

$$\{p \wedge (y \neq x+1) \wedge (z = \lfloor (x+y)/2 \rfloor) \wedge Cond(x,y,z)\} \rightarrow$$

$$((0 \leq z^2 \leq a < y^2) \wedge (z < y \leq a+1))$$

y

$$\{p \wedge (y \neq x+1) \wedge (z = \lfloor (x+y)/2 \rfloor) \wedge \neg Cond(x,y,z)\} \rightarrow$$

$$((0 \leq x^2 \leq a < z^2) \wedge (x < z \leq a+1)).$$

`Cond(x,y,z)=` $z^2 \leq a$ hace que se verifiquen las dos implicaciones anteriores. ($x < y \leq a+1$ implica $x < \lfloor (x+y)/2 \rfloor < y \leq a+1$ si $y \neq x+1$)

Derivación de programas: Raíz cuadrada entera

El programa final, cuya corrección parcial está probada, es:

$\{0 \leq a\}$

```
x:=0; y:= a+1;
mientras y≠x+1 hacer
```

$\{(0 \leq x^2 \leq a < y^2) \wedge x < y \leq a+1\}$

```
     z:=(x+y) div 2;
     si z²≤a entonces x:=z si no y:=z fsi
fmientras
```

$\{0 \leq x^2 \leq a < (x+1)^2\}$

Derivación de programas: Selección por eliminación

Sean dados dos números naturales m y n, con $m \leq n$ y sea P una propiedad cualquiera que pueden cumplir, o no, los naturales entre m y n. Supongamos que queremos saber si dados naturales r y s entre m y n, hay algún otro natural α, $r \leq \alpha \leq s$ que verifique la propiedad P. Partimos, pues, del aserto

$$\{m \leq r \leq s \leq n\}\ S\ \{b = \exists \alpha\ r \leq \alpha \leq s \wedge P(\alpha)\}.$$

Desarrollaremos un programa recursivo. La precondición nos asegura que el dominio del cuantificador existencial es no vacío.

- Si $r = s$, $b := P(r)$

Derivación de programas: Selección por eliminación

- $r \neq s$: haremos uso de $P \longrightarrow Q \equiv ((P \vee Q) = Q)$ [a]

$$\exists \alpha (r \leq \alpha \leq s \wedge P(\alpha)) \equiv P(r) \vee \exists \alpha (r+1 \leq \alpha \leq s \wedge P(\alpha))$$

$$\equiv (r \neq s) P(r) \vee P(s) \vee \exists \alpha (r+1 \leq \alpha \leq s-1 \wedge P(\alpha)) \equiv$$

$$\equiv (\text{si } P(r) \longrightarrow P(s)) P(s) \vee \exists \alpha (r+1 \leq \alpha \leq s-1 \wedge P(\alpha)) \equiv$$

$$\equiv \exists \alpha (r+1 \leq \alpha \leq s \wedge P(\alpha))$$

[a] En lo que sigue haremos uso de la siguiente notación $A \equiv (C)B$ para indicar que bajo la condición de que C sea cierta A y B son equivalentes.

Derivación de programas: Selección por eliminación

- $r \neq s$: Razonando igual que antes, se tiene:

$$\exists\alpha(r \leq \alpha \leq s \wedge P(\alpha)) \equiv$$

$$\equiv (\text{si } P(s) \longrightarrow P()) \; \exists\alpha(r \leq \alpha \leq s-1 \wedge P(\alpha))$$

Por otro lado,

$$(P(r) \longrightarrow P(s)) \vee (P(s) \longrightarrow P(r)) \equiv \neg P(r) \vee P(s) \vee \neg P(s) \vee P(r) \equiv T,$$

luego tenemos todos los casos cubiertos.

Derivación de programas: Selección por eliminación

Hemos obtenido el siguiente programa, cuya corrección parcial está probada:

$\{m \leq n\}$

```
r:=m; s:= n;
mientras r≠s hacer
```

Inv:

$\{m \leq r \leq s \leq n \wedge \exists\alpha(r \leq \alpha \leq s \wedge P(\alpha)) = \exists\alpha(m \leq \alpha \leq n \wedge P(\alpha))\}$

si $P(\mathrm{r}) \longrightarrow P(\mathrm{s})$ `entonces r:=r+1`

```
  si no s:=s-1
fsi
fmientras
b:=P(r).
```

$\{b := \exists\alpha(m \leq \alpha \leq n \wedge P(\alpha))\}$

La última asignación garantiza que si b se devuelve como valor cierto, r es un elemento que cumple P y puede devolverse si es oportuno.

Derivación de programas: Selección por eliminación

¿Qué ocurre cuando tenemos la seguridad de que hay elementos que cumplen la propiedad? En este caso, la precondición es $\{\exists\alpha(m \leq \alpha \leq n \wedge P(\alpha))\}$. Como invariante tenemos $\{m \leq r \leq s \leq n \wedge \exists\alpha(r \leq \alpha \leq s \wedge P(\alpha))\}$. Se tiene, en consecuencia, el programa:

$\{\exists\alpha\,(m \leq \alpha \leq n \wedge P(\alpha))\}$

```
r:=m; s:= n;
mientras r≠s hacer
```

Inv: $\{m \leq r \leq s \leq n \wedge \exists\alpha(r \leq \alpha \leq s \wedge P(\alpha))\}$

```
        si P(r) ⟶ P(s) entonces r:=r+1
          si no s:=s-1
        fsi
fmientras
```

$\{m \leq r \leq n \wedge P(r)\}$

Como aplicación de lo anterior, consideramos el problema de localizar el valor máximo de un vector no nulo de naturales de longitud N. Consideremos como especificación del programa a derivar

$$\{1 \leq N\}\ S\ \{1 \leq k \leq N \wedge \forall \beta\ 1 \leq \beta \leq N \longrightarrow a(\beta) \leq a(k)\},$$

donde $a(i)$ designa el natural que ocupa el lugar $i-$ésimo del vector a. Nótese que como precondición es necesario exigir $1 \leq N$, pues de lo contrario es imposible establecer $1 \leq k \leq N$. Por otro lado, para aligerar la notación en la discusión que sigue, definimos

$$P(a,i) \equiv \forall \beta\ (1 \leq \beta \leq N) \longrightarrow (a(\beta) \leq a(i)).$$

Derivación de programas: Localización del máximo

a)
- $a(i) \leq a(j) \longrightarrow (P(a,i) \longrightarrow P(a,j))$
- $a(j) \leq a(i) \longrightarrow (P(a,j) \longrightarrow P(a,i))$

b) $T \equiv (a(i) \leq a(j)) \vee (a(i) \leq a(j))$

Con esto en cuenta y lo hecho previamente, se deriva el siguiente programa:

$\{1 \leq N \wedge \exists \alpha\, (1 \leq \alpha \leq N \wedge P(a,\alpha))\}$

```
i:=1; j:= N;
mientras i≠j hacer
```

Inv: $1 \leq i \leq j \leq N \wedge \exists \alpha (i \leq \alpha \leq j \wedge P(a,\alpha))\}$

```
      si a(i) ≤ a(j) entonces i:=i+1
        si no j:=j-1
      fsi
fmientras
```

$\{1 \leq i \leq N \wedge P(a,i)\}$