Práctica 5 : Demostración de programas completos

Algoritmos y Estructuras de Datos

13 de de septiembre de 2023

Prueba de correctitud del programa completo

Recordemos: Hay que probar que

- $Pre \longrightarrow wp(\textbf{codigo previo al ciclo}, P_C)$
- $P_C \longrightarrow wp(\textbf{ciclo}, Q_C)$
  (**con el teorema del invariante, porque no se puede calcular esa wp en general**)
- $Q_C \longrightarrow wp(\textbf{codigo posterior al ciclo}, Post)$

Si probamos estas tres cosas, **por corolario de monotonía** (teórica 3) sabemos que
$Pre \longrightarrow wp(\texttt{programa completo}, Post)$
y, por lo tanto, **el programa es correcto con respecto a la especificación**.

Teorema de corrección de un ciclo:

1. $P_C \Rightarrow I$
   (**"El invariante se cumple antes de iniciar el ciclo"**)
2. $\{I \wedge B\}\ S\ \{I\}$
   (**"Ejecutar el cuerpo del ciclo preserva el invariante"**)
3. $I \wedge \neg B \Rightarrow Q_C$
   (**"El invariante y la no guarda implican la postcondición del ciclo"**)
4. $\{I \wedge B \wedge v_0 = fv\}\ \mathbf{S}\ \{fv < v_0\}$
   (**"La función variante es estrictamente decreciente"**)
5. $I \wedge fv \leq 0 \Rightarrow \neg B$
   (**"Cuando la función variante llega a 0, deja de valer la guarda"**)

Tips: Probando implicaciones

Estos ejercicios consisten principalmente en
(a) calcular wps y
(b) probar que ciertas implicaciones son tautologías.

Algunas cosas que es útil hacer para simplificar pruebas del estilo $A \longrightarrow B$:

- conviene transformar B para que se parezca *lo más posible a* A(**pero no necesariamente igual**).
- Si A afirma algo útil para transformar B, **lo puedo usar**, porque estoy asumiendo en la prueba que $A \equiv Verdadero$.
- ¡podemos reemplazar una parte de A o de B por el cuerpo de un `pred` o un `aux` que tengamos definido! (si hacer esto nos ayuda).
- Recordar equivalencias de fórmulas proposicionales que podamos aplicar (**Ej.** $(p \land q) \lor (\neg p \land q)$ **es equivalente a** q) .

No hay recetas mecánicas. Requiere práctica para aprender a identificar qué reescrituras necesitamos hacer, y qué pasos seguir para conseguirlas.

Especificación - problema hayMasImpares

Dada una secuencia de números enteros, se pide devolver verdadero o falso de acuerdo a si contiene o no más números impares.

proc `hayMasImpares` (in s: $seq\langle\mathbb{Z}\rangle$, out res: Bool) {
  `requiere` $\{s = S_0\}$
  `asegura` $\{s = S_0 \land res = True \leftrightarrow (|s| > 2 * contarPares(s)\}$
  `aux contarPares` (s: $seq\langle\mathbb{Z}\rangle$) : $\mathbb{Z} \ = \sum_{k=0}^{|s|-1} if \ \ s[k] \ mod \ 2 \ = \ 0 \ then \ 1 \ else \ 0 \ fi$;
}

Implementación de hayMasImpares en SmallLang

```
i := 0;
j := 0;
while (i < s.size()) do
    if (s[i] mod 2 = 0) then
        j := j + 1
    else
        skip
    endif;
    i := i + 1
endwhile;
if (s.size() > 2*j)
    res := true
else
    res := false
endif
```

Elección de P_C, Q_C, B, I, fv - justificación

- $P_C \equiv s = S_0 \land i = 0 \land j = 0$
  Es lo mínimo que podemos pedir a partir de Pre y del efecto de las dos primeras instrucciones
- $Q_C \equiv s = S_0 \land j = \sum_{k=0}^{|s|-1} \mathit{if}\ \ s[k]\ \mathit{mod}\ 2\ =\ 0\ \mathit{then}\ 1\ \mathit{else}\ 0\ \mathit{fi}$
  Es análogo a lo que pide Post, salvo que no habla del contenido de result (que será actualizado luego de terminar el ciclo)
- $B \equiv i < |s|$
  La traducción directa a lógica de la guarda del ciclo
- $I \equiv s = S_0 \land 0 \leq i \leq |s| \land_L j = \sum_{k=0}^{i-1} \mathit{if}\ \ s[k]\ \mathit{mod}\ 2\ =\ 0\ \mathit{then}\ 1\ \mathit{else}\ 0\ \mathit{fi}$
  Dado que el ciclo incrementa i, este invariante especifica que j contiene el conteo parcial de elementos pares hasta la posición i inclusive
- $fv = |s| - i$
  Como i siempre se incrementa en cada iteración, $|s| - i$ *es una función monótona decreciente*

1. $Pre \longrightarrow_L wp(\texttt{i:=0;}\ \ \texttt{j:=0}, P_C)$

1. Para esto calculamos esta wp:
$wp(\texttt{i:=0;}\ \ \texttt{j:=0}, P_C) \equiv wp(i := 0, wp(j := 0, P_C))$

2. Calculamos $wp(j := 0, P_C)$:
$wp(j := 0, \{s = S_0 \land i = 0 \land j = 0\}) \equiv def(0) \land_L s = S_0 \land i = 0 \land 0 = 0 \equiv$
$s = S_0 \land i = 0 \equiv E_1$

3. Calculamos $wp(i := 0, E_1)$:
$wp(i := 0, E_1) \equiv def(0) \land_L s = S_0 \land 0 = 0 \equiv$
$s = S_0 \equiv E_2$
Y como $Pre \equiv E_2$, $Pre \longrightarrow E_2$ ✓

2. $Q_C \longrightarrow_L wp(\texttt{if...then..else...fi}, Post)$

1. Calculamos la wp

$wp(\texttt{if s.size() > 2*j then res := true else res := false endif}, Post) \equiv$
$def(|s| > 2 * j) \wedge_L ((|s| > 2 * j \wedge wp(res := true, Post)) \vee (|s| \leq 2 * j \wedge wp(res := false, Post)) \equiv$
$(|s| > 2 * j \wedge wp(res := true, Post)) \vee$
$(|s| \leq 2 * j \wedge wp(res := false, Post))$

2a. $wp(res := true, Post) \equiv def(true) \wedge_L Post^{res}_{true} \equiv$
$s = S_0 \wedge true = true \iff (|s| > 2 * contarPares(s)) \equiv$
($|s| > 2 * contarPares(s)$ **debe ser Verdadero**)
$s = S_0 \wedge (|s| > 2 * contarPares(s))$

2b. $wp(res := false, Post) \equiv def(false) \wedge_L Post^{res}_{false} \equiv$
$s = S_0 \wedge false = true \iff (|s| > 2 * contarPares(s)) \equiv$
($false = true$ **es Falso, entonces** $|s| > 2 * contarPares(s)$ **debe ser Falso, invierto desigualdad**)
$s = S_0 \wedge (|s| \leq 2 * contarPares(s))$

$Q_C \longrightarrow_L wp(\texttt{if...then..else...fi}, Post)$ (2)

3. $wp(\texttt{if...}, Post) \equiv$

$\big($**Uso wps resultado de 2a y 2b, y saco $s = S_0$ afuera de la disyunción**$\big)$

$s = S_0 \land ($

$(|s| > 2 * j \land (|s| > 2 * contarPares(s)) \lor$

$(|s| \leq 2 * j \land (|s| \leq 2 * contarPares(s)))) \equiv$

$\big($**aplico** $(p \land q) \lor (\neg p \land \neg q) \equiv p \iff q\big)$

$\{s = S_0 \land (|s| > 2 * j \iff (|s| > 2 * contarPares(s)))\} \equiv E_3$

$Q_C \longrightarrow_L wp(\texttt{if...then..else...fi}, Post)$ (3)

4. Chequeo $Q_C \longrightarrow_L E_3$

$E_3 \equiv \{s = S_0 \land (|s| > 2 * j \iff (|s| > 2 * contarPares(s)))\}$

$Q_C \equiv (s = S_0 \land j = \sum_{k=0}^{|s|-1} if\ s[k]\ mod\ 2\ =\ 0\ then\ 1\ else\ 0\ fi) \equiv$

(**reemplazo sintáctico de la sumatoria por def. de contarPares**)

$\{s = S_0 \land j = contarPares(s)\} \equiv Q_C$

Vemos si esto implica a E_3:

$s = S_0 \land j = contarPares(s) \longrightarrow_L s = S_0 \land (|s| > 2 * j \iff (|s| > 2 * contarPares(s)))$

(**asumo Q_C Verdadero, analizo E_3, reemplazo j por equivalente en el consecuente**)

$s = S_0 \land (|s| > 2 * contarPares(s) \iff (|s| > 2 * contarPares(s)))$

(**$s = S_0$ es Verdadero por Q_C, y aplico que $p \iff p$ es tautología**)

$True \land True \equiv True$ ✓

3. $P_C \longrightarrow_L wp(\texttt{while...}, Q_C)$

Esto consiste en hacer la prueba completa de correctitud de ciclos para mostrar que la tripla de Hoare
$\{P_C\}$ `while...` $\{Q_C\}$
es válida.

$P_C \longrightarrow I$

- $P_C \equiv s = S_0 \wedge i = 0 \wedge j = 0$
- $I \equiv s = S_0 \wedge 0 \leq i \leq |s| \wedge_L j = \sum_{k=0}^{i-1} if\ \ s[k]\ mod\ 2\ = \ 0\ then\ 1\ else\ 0\ fi$

$s = S_0 \wedge i = 0 \wedge j = 0 \longrightarrow$

- $s = S_0$ ✓ (**trivial**)
- $0 \leq i \leq |s|$ ✓ (**trivial**)
- $j = \sum_{k=0}^{i-1} if\ \ s[k]\ mod\ 2\ = \ 0\ then\ 1\ else\ 0\ fi$ ✓

(**porque la sumatoria es vacía y suma 0**)

$I \wedge \neg B \longrightarrow Q_C$

- $I \equiv s = S_0 \wedge 0 \leq i \leq |s| \wedge_L j = \sum_{k=0}^{i-1} if \;\; s[k] \;\; mod \;\; 2 \;\; = \;\; 0 \;\; then \;\; 1 \;\; else \;\; 0 \;\; fi$
- $B \equiv i < |s|$
- $Q_C \equiv s = S_0 \wedge j = \sum_{k=0}^{|s|-1} if \;\; s[k] \;\; mod \;\; 2 \;\; = \;\; 0 \;\; then \;\; 1 \;\; else \;\; 0 \;\; fi$

- $s = S_0$ ✓(**lo afirma** I)
- $j = \sum_{k=0}^{|s|-1} if \;\; s[k] \;\; mod \;\; 2 \;\; = \;\; 0 \;\; then \;\; 1 \;\; else \;\; 0 \;\; fi$ ✓

(**porque según** $I \wedge \neg B$ **sé que** $i = |S|$ **y lo aplico a definición de** j **en** I)

$I \wedge fv \leq 0 \longrightarrow \neg B$

- $I \equiv s = S_0 \wedge 0 \leq i \leq |s| \wedge_L j = \sum_{k=0}^{i-1} if\ \ s[k]\ mod\ 2\ = \ 0\ then\ 1\ else\ 0\ fi$
- $fv = |s| - i$
- $B \equiv i < |s|$

$I \wedge fv \leq 0 \equiv$

(**definición de fv**)

$I \wedge |s| - i \leq 0 \equiv$

(**sumo i de ambos lados de la desigualdad**)

$I \wedge |s| \leq i \equiv$

($\neg B \equiv |s| \leq i$)

$I \wedge \neg B \longrightarrow \neg B$ ✓

$\{I \wedge B\}\ S\ \{I\}$

- $B \equiv i < |s|$
- $I \equiv s = S_0 \wedge 0 \leq i \leq |s| \wedge_L j = \sum_{k=0}^{i-1} if\ \ s[k]\ mod\ 2\ = \ 0\ then\ 1\ else\ 0\ fi$

Veo si $I \wedge B \longrightarrow wp($`if...; i:= i + 1`$, I)$

$wp(i := i+1, I) \equiv def(i+1) \wedge_L I^i_{i+1} \equiv$

$s = S_0 \wedge 0 \leq i+1 \leq |s| \wedge_L j = \sum_{k=0}^{i} if\ \ s[k]\ mod\ 2\ = \ 0\ then\ 1\ else\ 0\ fi \equiv E_4$

$wp(if\ldots, E_4) \equiv$

$def(s[i]\ \bmod 2 = 0) \wedge_L (\ (s[i]\ \bmod 2 = 0 \wedge wp(j := j+1, E_4)) \vee\ (s[i]\ \bmod 2 \neq 0 \wedge wp(skip, E_4))\ ) \equiv$

$(s[i]\ \bmod 2 = 0 \wedge {E_4}^j_{j+1}) \vee\ (s[i]\ \bmod 2 \neq 0 \wedge E_4)$

(**reemplazo** def**, simplifico** $s = S_0 \wedge 0 \leq i+1 \leq |s|$ **en la subexpresión, paso restando el 1 que suma a j**)

$s = S_0 \wedge 0 \leq i+1 \leq |s| \wedge_L ($

$(s[i]\ mod\ 2 = 0 \wedge j = \sum_{k=0}^{i} if\ \ s[k]\ mod\ 2\ = \ 0\ then\ 1\ else\ 0\ fi - 1)\ \vee$

$(s[i]\ mod\ 2 \neq 0 \wedge j = \sum_{k=0}^{i} if\ \ s[k]\ mod\ 2\ = \ 0\ then\ 1\ else\ 0\ fi))) \equiv$

Necesitamos llevar esto a algo similar al Invariante...

... ¡reescribiendo los términos!

$\{I \wedge B\}\ S\ \{I\}$ (2)

Reescribimos las sumatorias restando términos if..then..else equivalentes a restar 0 y restar 1 respectivamente... iguales a losque usan las sumatorias!

$s = S_0 \wedge 0 \leq i+1 \leq |s| \wedge_L ($
$(s[i]\ mod\ 2 = 0 \wedge j = \sum_{k=0}^{i} if\ \ s[k]\ mod\ 2\ = \ 0\ then\ 1\ else\ 0\ fi$
$- if\ \ s[i]\ mod\ 2\ = \ 0\ then\ 1\ else\ 0\ fi) \vee$
$(s[i]\ mod\ 2 \neq 0 \wedge j = \sum_{k=0}^{i} if\ \ s[k]\ mod\ 2\ = \ 0\ then\ 1\ else\ 0\ fi$
$- if\ \ s[i]\ mod\ 2\ = \ 0\ then\ 1\ else\ 0\ fi))) \equiv$
(**restamos los teminos a las sumatorias y ajustamos los indices**)
$s = S_0 \wedge 0 \leq i+1 \leq |s| \wedge_L ($
$(s[i]\ mod\ 2 = 0 \wedge j = \sum_{k=0}^{i-1} if\ \ s[k]\ mod\ 2\ = \ 0\ then\ 1\ else\ 0\ fi) \vee$
$(s[i]\ mod\ 2 \neq 0 \wedge j = \sum_{k=0}^{i-1} if\ \ s[k]\ mod\ 2\ = \ 0\ then\ 1\ else\ 0\ fi))) \equiv$
(**aplico** $(p \wedge q) \vee (\neg p \wedge q) \equiv q$)
$s = S_0 \wedge 0 \leq i+1 \leq |s| \wedge_L j = \sum_{k=0}^{i-1} if\ \ s[k]\ mod\ 2\ = \ 0\ then\ 1\ else\ 0\ fi \equiv E_5$

$\{I \land B\}\ S\ \{I\}$ (3)

- $I \equiv s = S_0 \land 0 \leq i \leq |s| \land_L j = \sum_{k=0}^{i-1} \textit{if } s[k] \textit{ mod } 2 = 0 \textit{ then } 1 \textit{ else } 0 \textit{ fi}$
- $B \equiv i < |s|$
- $E_5 \equiv s = S_0 \land 0 \leq i+1 \leq |s| \land_L j = \sum_{k=0}^{i-1} \textit{if } s[k] \textit{ mod } 2 = 0 \textit{ then } 1 \textit{ else } 0 \textit{ fi}$

Chequeamos si $I \land B \longrightarrow E_5$:

- $s = S_0$ ✓(*I* **afirma esto**)
- $0 \leq i+1 \leq |s|$ ✓(*I* **afirma que** $0 \leq i \leq |s|$ **y B afirma que** $i < |s|$)
- $j = \sum_{k=0}^{i-1} \textit{if } s[k] \textit{ mod } 2 = 0 \textit{ then } 1 \textit{ else } 0 \textit{ fi}$ ✓ (*I* **afirma esto**)

$$\{I \land B \land v_0 = |s| - i\} \ \textbf{S} \ \{|s| - i < v_0\}$$

Veo si $I \land B \land v_0 = |s| - i \longrightarrow wp(\texttt{if...; i:= i + 1}, |s| - i < v_0)$

$$wp(i := i + 1, |s| - i < v_0) \equiv def(i+1) \land_L |s| - i - 1 < v_0 \equiv |s| - i < v_0 + 1$$

$wp(if\ldots, |s| - i < v_0 + 1) \equiv def(s[i] \mod 2 = 0) \land_L ($
$(s[i] \mod 2 = 0 \land wp(j := j + 1, |s| - i < v_0 + 1)) \lor$
$(s[i] \mod 2 \neq 0 \land wp(skip, |s| - i < v_0 + 1))\ ) \equiv$
$(s[i] \mod 2 = 0 \land \{|s| - i < v_0 + 1\}^{j}_{j+1})) \lor$
$(s[i] \mod 2 \neq 0 \land |s| - i < v_0 + 1) \equiv$
(**Como j no aparece en $|s| - i < v_0 + 1$, el reemplazo es igual al predicado original**)
$(s[i] \mod 2 = 0 \land |s| - i < v_0 + 1) \lor \ (s[i] \mod 2 \neq 0 \land |s| - i < v_0 + 1) \equiv$
(**aplico** $(p \land q) \lor (\neg p \land q) \equiv q$)
$|s| - i < v_0 + 1$

Vemos si vale la implicación:
$I \land i < |s| \land |s| - i = v_0 \longrightarrow |s| - i < v_0 + 1$ ✓
(**Sí, porque** $n = m \longrightarrow n < m + 1$)

Prueba de correctitud del programa completo

De acuerdo a lo anterior, probamos:

- $Pre \longrightarrow wp($**codigo previo al ciclo**$, P_C)$
- $P_C \longrightarrow wp($**ciclo**$, Q_C)$
- $Q_C \longrightarrow wp($**codigo posterior al ciclo**$, Post)$

Al probar estas tres cosas, **por corolario de monotonía** (teórica 3) sabemos que
$Pre \longrightarrow wp(\texttt{programa completo}, Post)$
y, por lo tanto, **el programa es correcto con respecto a la especificación**.