1. Precondición más débil en SmallLang

Ejercicio 1. Calcular las siguientes expresiones, donde a, b son variables reales, i una variable entera y A es una secuencia de reales.

a) $\text{def}(a+1)$.

b) $\text{def}(a/b)$.

c) $\text{def}(\sqrt{a/b})$.

d) $\text{def}(A[i]+1)$.

e) $\text{def}(A[i+2])$.

f) $\text{def}(0 \leq i \leq |A| \ \wedge_L \ A[i] \geq 0)$.

Ejercicio 2. Calcular las siguientes precondiciones más débiles, donde a, b son variables reales, i una variable entera y A es una secuencia de reales.

a) $wp(\textbf{a := a+1; b := a/2}, b \geq 0)$.

b) $wp(\textbf{a := A[i] + 1; b := a*a}, b \neq 2)$.

c) $wp(\textbf{a := A[i] + 1; a := b*b}, a \geq 0)$.

d) $wp(\textbf{a := a-b; b := a+b}, a \geq 0 \wedge b \geq 0)$.

Ejercicio 3. Sea $Q \equiv (\forall j :\mathbb{Z})(0 \leq j < |A| \rightarrow_L A[j] \geq 0)$. Calcular las siguientes precondiciones más débiles, donde i es una variable entera y A es una secuencia de reales.

a) $wp(\textbf{A[i] := 0}, Q)$.

b) $wp(\textbf{A[i+2] := 0}, Q)$.

c) $wp(\textbf{A[i+2] := -1}, Q)$.

d) $wp(\textbf{A[i] := 2 * A[i]}, Q)$.

e) $wp(\textbf{A[i] := A[i-1]}, Q)$.

Ejercicio 4. Calcular $wp(S, Q)$, para los siguientes pares de programas S y postcondiciones Q.

a) $S \equiv$

```
if( a < 0 )
  b := a
else
  b := -a
endif
```

$Q \equiv (b = -|a|)$

b) $S \equiv$

```
if( a < 0 )
  b := a
else
  b := -a
endif
```

$Q \equiv (b = |a|)$

c) $S \equiv$

```
if( i > 0 )
  s[i] := 0
else
  s[0] := 0
endif
```

$Q \equiv (\forall j : \mathbb{Z})(0 \leq j < |s| \rightarrow_L s[j] \geq 0)$

d) $S \equiv$

```
if( i > 1 )
  s[i] := s[i-1]
else
  s[i] := 0
endif
```

$Q \equiv (\forall j : \mathbb{Z})(1 \leq j < |s| \rightarrow_L s[j] = s[j-1])$

e) $S \equiv$

```
if( s[i] < 0 )
  s[i] := -s[i]
else
  skip
endif
```

$Q \equiv 0 \leq i < |s| \ \wedge_L \ s[i] \geq 0$

f) $S \equiv$

```
if( s[i] > 0 )
  s[i] := -s[i]
else
  skip
endif
```

$Q \equiv (\forall j : \mathbb{Z})(0 \leq j < |s| \ \rightarrow_L \ s[j] \geq 0)$

Ejercicio 5. Escribir programas para los siguientes problemas y demostrar formalmente su corrección usando la precondición más débil.

a) `proc problema1` (in s: $seq\langle\mathbb{Z}\rangle$, in i: $\mathbb{Z}$, inout a: $\mathbb{Z}$)

   `requiere` $\{0 \leq i < |s| \ \wedge_L \ a = \sum_{j=0}^{i-1} s[j]\}$

   `asegura` $\{a = \sum_{j=0}^{i} s[j]\}$

b) `proc problema2` (in s: $seq\langle\mathbb{Z}\rangle$, in i: $\mathbb{Z}$, inout a: $\mathbb{Z}$)

   `requiere` $\{0 \leq i < |s| \ \wedge_L \ a = \sum_{j=0}^{i} s[j]\}$

   `asegura` $\{a = \sum_{j=1}^{i} s[j]\}$

c) `proc problema3` (in s: $seq\langle\mathbb{Z}\rangle$, in i: $\mathbb{Z}$) : Bool

   `requiere` $\{0 \leq i < |s| \ \wedge_L \ (\forall j : \mathbb{Z})(0 \leq j < i \rightarrow_L s[j] \geq 0)\}$

   `asegura` $\{res = true \leftrightarrow (\forall j : \mathbb{Z})(0 \leq j \leq i \rightarrow_L s[j] \geq 0)\}$

d) `proc problema4` (in s: $seq\langle\mathbb{Z}\rangle$, in i: $\mathbb{Z}$, inout a: $\mathbb{Z}$)

   `requiere` $\{0 \leq i < |s| \ \wedge_L \ a = \sum_{j=0}^{i-1} \beta(s[j] \neq 0)\}$

   `asegura` $\{a = \sum_{j=0}^{i} \beta(s[j] \neq 0)\}$

e) `proc problema5` (in s: $seq\langle\mathbb{Z}\rangle$, in i: $\mathbb{Z}$, inout a: $\mathbb{Z}$)

   `requiere` $\{0 < i \leq |s| \ \wedge_L \ a = \sum_{j=1}^{i-1} \beta(s[j] \neq 0)\}$

   `asegura` $\{a = \sum_{j=0}^{i-1} \beta(s[j] \neq 0)\}$

2. Demostración de corrección de ciclos en SmallLang

Teorema del invariante: corrección de ciclos

Ejercicio 6. Consideremos el problema de sumar los elementos de un arreglo y la siguiente implementación en SmallLang, con el invariante del ciclo.

Especificación

`proc sumar` (in s: $array < \mathbb{Z} >$) : $\mathbb{Z}$
`requiere` $\{True\}$
`asegura` $\{res = \sum_{j=0}^{|s|-1} s[j]\}$

Implementación en SmallLang

```
res := 0;
i := 0;
while (i < s.size()) do
  res := res + s[i];
  i := i + 1
endwhile
```

Invariante de Ciclo

$$I \equiv 0 \leq i \leq |s| \wedge_L res = \sum_{j=0}^{i-1} s[j]$$

a) Escribir la precondición y la postcondición del ciclo.

b) ¿Qué punto falla en la demostración de corrección si el primer término del invariante se reemplaza por $0 \leq i < |s|$?

c) ¿Qué punto falla en la demostración de corrección si el límite superior de la sumatoria $(i - 1)$ se reemplaza por i?

d) ¿Qué punto falla en la demostración de corrección si se invierte el orden de las dos instrucciones del cuerpo del ciclo?

e) Demostrar formalmente la corrección parcial del ciclo, usando los primeros puntos del teorema del invariante.

f) Proponer una función variante y demostrar formalmente la terminación del ciclo, utilizando la función variante.

Ejercicio 7. Dadas la especificación y la implementación del problema `sumarParesHastaN`, escribir la precondición y la postcondición del ciclo, y demostrar formalmente su corrección a través del teorema del invariante.

Especificación

`proc sumarParesHastaN` (in n: $\mathbb{Z}$) : $\mathbb{Z}$
`requiere` $\{n \geq 0\}$
`asegura` $\{res = \sum_{j=0}^{n-1} (\text{if } j \ mod \ 2 = 0 \text{ then } j \text{ else } 0 \text{ fi})\}$

Implementación en SmallLang

```
res := 0;
i := 0;
while (i < n) do
  res := res + i;
  i := i + 2
endwhile
```

Invariante de ciclo

$$I \equiv 0 \leq i \leq n + 1 \wedge i \ mod \ 2 \ = \ 0 \wedge res = \sum_{j=0}^{i-1} (\text{if } j \ mod \ 2 = 0 \text{ then } j \text{ else } 0 \text{ fi})$$

Ejercicio 8. Considere el problema `sumaDivisores`, dado por la siguiente especificación:

proc sumaDivisores (in n: $\mathbb{Z}$) : $\mathbb{Z}$
  requiere $\{n \geq 1\}$
  asegura $\{res = \sum_{j=1}^{n}(\text{if } n \ mod \ j = 0 \text{ then } j \text{ else } 0 \text{ fi})\}$

a) Escribir un programa en SmallLang que satisfaga la especificación del problema y que contenga exactamente un ciclo.

b) El ciclo del programa propuesto, ¿puede ser demostrado mediante el siguiente invariante?

$$I \equiv 1 \leq i \leq n \wedge res = \sum_{j=1}^{i}(\text{if } n \ mod \ j = 0 \text{ then } j \text{ else } 0 \text{ fi})$$

Si no puede, ¿qué cambios se le deben hacer al invariante para que se corresponda con el ciclo propuesto?

Ejercicio 9. Considere la siguiente especificación e implementación del problema `copiarSecuencia`.

Especificación

proc copiarSecuencia (in s: $array < \mathbb{Z} >$, inout r: $array < \mathbb{Z} >$)
  requiere $\{|s| = |r| \wedge r = r_0\}$
  asegura $\{|s| = |r| \wedge_L (\forall j : \mathbb{Z})(0 \leq j < |s| \rightarrow_L s[j] = r[j])\}$

Implementación en SmallLang

```
i := 0;
while (i < s.size()) do
  r[i]:=s[i];
  i:=i+1
endwhile
```

a) Escribir la precondición y la postcondición del ciclo.

b) Proponer un invariante y demostrar que el ciclo es parcialmente correcto.

c) Proponer una función variante que permita demostrar que el ciclo termina.

Ejercicio 10. Sea el siguiente ciclo con su correspondiente precondición y postcondición:

```
while (i >= length(s) / 2) do
  suma := suma + s[length(s)-1-i];
  i := i - 1
endwhile
```

$$P_c : \{|s| \ mod \ 2 = 0 \wedge i = |s| - 1 \wedge suma = 0\}$$

$$Q_c : \{|s| \ mod \ 2 = 0 \wedge i = |s|/2 - 1 \ \wedge_L \ suma = \sum_{j=0}^{|s|/2-1} s[j]\}$$

a) Especificar un invariante de ciclo que permita demostrar que el ciclo cumple la postcondición.

b) Especificar una función variante que permita demostrar que el ciclo termina.

c) Demostrar formalmente la corrección y terminación del ciclo usando el Teorema del invariante.

Demostración de correctitud: programas completos

Ejercicio 11. Demostrar que el siguiente programa es **correcto** respecto a la especificación dada.

Especificación

proc indice (in s: $array < \mathbb{Z} >$, in e: $\mathbb{Z}$) : $\mathbb{Z}$
  requiere $\{True\}$
  asegura $\{res = -1 \rightarrow$
  $(\forall j : \mathbb{Z})(0 \leq j < |s| \rightarrow_L s[j] \neq e)$
  $\wedge$
  $r \neq -1 \rightarrow$
  $(0 \leq r < |s| \wedge_L s[r] = e)\}$

Implementación en SmallLang

```
i := s.size()-1;
j := -1;
while (i>=0) do
  if (s[i]=e) then
     j:=i
  else
     skip
  endif;
  i:=i-1
endwhile;
r := j;
```

Ejercicio 12. Demostrar que el siguiente programa es correcto respecto a la especificación dada.

Especificación

`proc existeElemento` (in s: $array < \mathbb{Z} >$, in e: $\mathbb{Z}$) : Bool
`requiere` $\{True\}$
`asegura` $\{res = \text{true} \leftrightarrow ((\exists k : \mathbb{Z})(0 \leq k < |s|) \wedge_L s[k] = e)\}$

Implementación en SmallLang

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

Ejercicio 13. Demostrar que el siguiente programa es correcto respecto a la especificación dada.

Especificación

`proc concatenarSecuencias` (in a: $array < \mathbb{Z} >$,
in b: $array < \mathbb{Z} >$,
inout r:$array < \mathbb{Z} >$)
`requiere` $\{|r| = |a| + |b| \wedge r = R_0\}$
`asegura` $\{|r| = |R_0| \wedge (\forall j : \mathbb{Z})(0 \leq j < |a| \rightarrow_L r[j] = a[j]) \wedge (\forall j : \mathbb{Z})(0 \leq j < |b| \rightarrow_L r[j + |a|] = b[j])\}$

Implementación en SmallLang

```
i := 0;
while (i < a.size()) do
  r[i]:=a[i];
  i:=i+1
endwhile;
i:= 0;
while (i < b.size()) do
  r[a.size()+i]:=b[i];
  i:=i+1
endwhile
```