Algoritmos y Estructuras de Datos

Guía Práctica 4
Verificación de programas (Parte 1)
Segundo Cuatrimestre 2025

Departamento de Computación
Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires

4.1. 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 enteros.

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. Para los siguientes pares de programas S y postcondiciones Q

- Escribir la precondición más débil $P = wp(S, Q)$
- Mostrar formalmente que la P elegida es correcta

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. Para las siguientes especificaciones:

- Poner nombre al problema que resuelven
- Escribir un programa S sencillo en SmallLang, sin ciclos, que lo resuelva
- Dar la precondición más débil del programa escrito con respecto a la postcondición de su especificación

a) `proc problema1`(in $s : \mathsf{seq}\langle\mathbb{Z}\rangle$, 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 $s : \mathsf{seq}\langle\mathbb{Z}\rangle$, 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` { $res = true \leftrightarrow (\forall j : \mathbb{Z})\ (0 \leq j \leq i \rightarrow_L s[j] \geq 0)$ }
}

c) `proc problema3`(inout $s : \mathsf{seq}\langle\mathbb{Z}\rangle$, in $i : \mathbb{Z}$) {
`requiere` { $(0 \leq i < |s|) \wedge_L (\forall j : \mathbb{Z})\ (0 \leq j < i \rightarrow s[j] = fibonacci(j))$ }
`asegura` { $(\forall j : \mathbb{Z})\ (0 \leq j \leq i \rightarrow s[j] = fibonacci(j))$ }
}

Ejercicio 6. Dada la poscondición $Q \equiv \{(\forall j : \mathbb{Z})\ (0 \leq j < |s| \rightarrow_L s[j] \ \text{ mód } 2 = 0)\}$ y el siguiente código

```
if (i mod 3 = 0)
  s[i] := s[i] + 6;
else
  s[i] := i;
endif
```

a) Demostrar que las siguientes WPs son incorrectas dando un contraejemplo

i) $P \equiv \{0 \leq i \leq |s| \wedge_L i \ mod\ 3 = 0 \wedge (\forall j : \mathbb{Z})\ (0 \leq j < |s| \rightarrow_L s[j] \ mod\ 2 = 0)\}$
ii) $P \equiv \{0 \leq i < |s| \wedge_L i \ mod\ 3 \neq 0 \wedge (\forall j : \mathbb{Z})\ (0 \leq j < |s| \rightarrow_L s[j] \ mod\ 2 = 0)\}$
iii) $P \equiv \{i \ mod\ 3 = 0 \wedge_L (\forall j : \mathbb{Z})\ (0 \leq j < |s| \rightarrow_L s[j] \ mod\ 2 = 0)\}$
iv) $P \equiv \{0 \leq i < |s|/2 \wedge_L i \ mod\ 3 = 0 \wedge (\forall j : \mathbb{Z})\ (0 \leq j < |s| \rightarrow_L s[j] \ mod\ 2 = 0)\}$

4.1.1. Ejercicios de parcial

Ejercicio 7. Dado el siguiente condicional determinar la precondición más débil que permite hacer valer la poscondición (Q) propuesta. Se pide:

- Describir en palabras la WP esperada
- Derivarla formalmente a partir de los axiomas de precondición más débil. Para obtener el puntaje máximo deberá simplificarla lo más posible.

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

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

b) $Q \equiv \{(\exists j : \mathbb{Z})\ (j \geq 0 \wedge j^2 = a)\}$

```
if (a mod 2 = 0)
  a := a * a
else
  a := - |a|
endif
```

c) $Q \equiv \{(\forall j : \mathbb{Z})\ (0 \leq j < |s| \rightarrow_L s[j] = 2^j)\}$

```
if (s[i] != 2^i)
  s[i] = 2 * s[i-1]
else
  s[0] = 1
endif
```

d) $Q \equiv \{(\forall j : \mathbb{Z})\ (0 \leq j < |s| \rightarrow_L s[j]\ mod\ 3 = 0)\}$

```
if (i mod 3 == 0)
  s[i] = s[i] + 6
else
  s[i] = i
endif
```

e) $Q \equiv \{(\forall j : \mathbb{Z})\ (0 \leq j < |s| \rightarrow_L s[j]\ mod\ 2 = 0)\}$

```
if (i mod 2 = 0)
  s[i] = 2 * s[i]
else
  s[0] = 3
endif
```

Ejercicio 8. Para los siguientes algoritmos **S** con sus post condiciones Q, indique cuál de las precondiciones propuestas es la *Precondición más débil* y justifique muy brevemente en palabras.

a) **S** $\equiv$

```
if (a mod 2 = 0)
  a := |a| + 1
else
  a := |a| * 2
endif
```

$Q \equiv \{a \mod 2 = 0\}$

(i) $P \equiv \{a \mod 2 = 0\}$

(ii) $P \equiv \{a \mod 2 \neq 0\}$

(iii) $P \equiv \{True\}$

b) **S** $\equiv$

```
j := i - 2
s[j] := 2 * i
```

$Q \equiv \{(\forall k : \mathbb{Z})\ (0 \leq k < |s| \rightarrow_L s[k] \mod 2 = 0)\}$

(i) $P \equiv \{(\forall k : \mathbb{Z})\ (0 \leq k < |s| \wedge k \neq i - 2 \rightarrow_L s[k] \mod 2 = 0)\}$

(ii) $P \equiv \{2 \leq i < |s| + 2 \wedge$
$(\forall k : \mathbb{Z})\ (0 \leq k < |s| \wedge k \neq i - 2 \rightarrow_L s[k] \mod 2 = 0)\}$

(iii) $P \equiv \{i \mod 2 = 0 \wedge 2 \leq i < |s| + 2 \wedge$
$(\forall k : \mathbb{Z})\ (0 \leq k < |s| \wedge k \neq i - 2 \rightarrow_L s[k] \mod 2 = 0)\}$

c) **S** $\equiv$

```
if (x > y)
  y := x
else
  y := 3
endif
```

$Q \equiv \{y > 0\}$

(i) $P \equiv \{x > y\}$

(ii) $P \equiv \{x > y \vee x \leq y\}$

(iii) $P \equiv \{(x > 0 \wedge x > y) \vee (x \leq y)\}$

d) **S** $\equiv$

```
if (i mod 2 = 0)
  s[i] := 1
else
  s[i] := 5
endif
```

$Q \equiv \{(\forall i : \mathbb{Z})\ (0 \leq i < |s| \wedge i \mod 2 = 0 \rightarrow_L s[i] = 1)\}$

(i) $P \equiv \{0 \leq i < |s| \wedge i \mod 2 = 0 \wedge$
$(\forall j : \mathbb{Z})\ (0 \leq j < |s| \wedge j \mod 2 = 0 \wedge j \neq i \rightarrow_L s[j] = 1)\}$

(ii) $P \equiv \{0 \leq i < |s| \wedge i \mod 2 = 0 \wedge$
$(\forall j : \mathbb{Z})\ (0 \leq j < |s| \wedge j \neq i \rightarrow_L s[j] = 1)\}$

(iii) $P \equiv \{i \mod 2 = 0 \wedge$
$(\forall j : \mathbb{Z})\ (0 \leq j < |s| \wedge j \mod 2 = 0 \wedge j \neq i \rightarrow_L s[j] = 1)\}$

4.2. Demostración de corrección de ciclos en SmallLang

4.2.1. Teorema del invariante: corrección de ciclos

Ejercicio 9. 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⟨ℤ⟩) : ℤ {
   requiere { True }
   asegura { res = Σ_{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 poscondición 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) Mostrar la corrección parcial del ciclo, usando los primeros puntos del teorema del invariante.

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

Ejercicio 10. Dadas la especificación y la implementación del problema `sumarParesHastaN`

Especificación

```
proc sumarParesHastaN(in n : ℤ) : ℤ {
   requiere { n ≥ 0 }
   asegura {
       res = Σ_{j=0}^{n-1}(IfThenElse(j  mód 2 = 0, j, 0))
   }
}
```

Implementación en SmallLang

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

Invariante de ciclo

$$I \equiv 0 \leq i \leq n+1 \wedge i \quad \text{mód } 2 \;=\; 0 \wedge res = \sum_{j=0}^{i-1} (\mathsf{IfThenElse}(j \quad \text{mód } 2 = 0, j, 0))$$

a) Escribir la precondición y la poscondición del ciclo.

b) Mostrar la corrección parcial del ciclo, usando los primeros puntos del teorema del invariante.

c) Proponer una función variante y mostrar la terminación del ciclo, utilizando la función variante.

Ejercicio 11. Considere el problema `sumaDivisores`, dado por la siguiente especificación:

```
proc sumaDivisores(in n : ℤ) : ℤ {
   requiere { n ≥ 1 }
   asegura { res = Σ_{j=1}^{n}(IfThenElse(n  mód j = 0, j, 0)) }
}
```

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

b) Escribir la pre y post condición del ciclo y su invariante.

c) Considere el siguiente invariante para este problema

$$I \equiv 1 \leq i \leq n/2 \wedge res = \sum_{j=1}^{i} (\mathsf{IfThenElse}(n \quad \text{mód } j = 0, j, 0))$$

Si no coincide con el propuesto en el inciso anterior, ¿qué cambios se le deben hacer al programa para que lo represente este invariante? ¿Deben cambiar la pre y post condición?

Ejercicio 12. Considere la siguiente especificación e implementación del problema `copiarSecuencia`, y la pre y post condiciones del ciclo.

Especificación

```
proc copiarSecuencia(in s : array⟨ℤ⟩, inout r : array⟨ℤ⟩) {
    requiere { |s| = |r| ∧ r = R_0 }
    asegura {
        |s| = |r| ∧_L (∀j : ℤ) (0 ≤ j < |s| →_L s[j] = r[j])
    }
}
```

Implementación en SmallLang

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

$$P_c \equiv |s| = |r| \land i = 0$$

$$Q_c \equiv |s| = |r| \land_L (\forall j : \mathbb{Z})\ (0 \leq j < |r| \rightarrow_L s[j] = r[j])$$

a) ¿Qué variables del programa deben aparecer en el invariante?

b) Proponer un invariante e indicar qué cláusula del mismo es necesario para cada paso de la demostración.

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

Ejercicio 13. Sea el siguiente ciclo con su correspondiente precondición y postcondición:

```
while (i >= s.size() / 2) do
  suma := suma + s[s.size()−1−i];
  i := i − 1
endwhile
```

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

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

a) Proponer un invariante e indicar qué clausula del mismo es necesaria para cada paso de la demostración.

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

c) Demostrar la terminación del ciclo utilizando la función variante.

Ejercicio 14. Dado el siguiente problema

```
proc sumarElementos(in s : array⟨ℤ⟩) : ℤ {
    requiere { |s| ≥ 1 ∧ |s| mod 2 = 0 }
    asegura { res = Σ_{j=0}^{|s|−1} s[j] }
}
```

Dar un invariante y función variante para cada una de estas posibles implementaciones

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

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

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

d)
```
res := 0
i := 0
while (i < s.size() / 2) do
  res := res + s[i] + s[s.size() − 1 − i];
  i := i + 1
endwhile
```

Ejercicio 15. Considerando el siguiente Invariante:

$$I \equiv \{0 \leq i \leq |s| \wedge (\forall j : \mathbb{Z})\ (0 \leq j < i \rightarrow_L (j \mod 2 = 0 \wedge s[j] = 2 \times j \vee j \mod 2 \neq 0 \wedge s[j] = 2 \times j + 1))\}$$

- Escribir un programa en SmallLang que se corresponda al invariante dado.
- Defina las P_c, B y Q_c que correspondan a su programa.
- Dar una función variante para que se pueda completar la demostración.

Ejercicio 16. Considerando el siguiente Invariante:

$$I \equiv \{0 \leq i \leq |s|/2 \wedge (\forall j : \mathbb{Z})\ (0 \leq j < i) \rightarrow_L (s[j] = 0 \wedge s[|s| - j - 1] = 0)\}$$

- Escribir un programa en SmallLang que se corresponda al invariante dado.
- Defina las P_c, B y Q_c que correspondan a su programa.
- Dar una función variante para que se pueda completar la demostración.

Ejercicio 17. Indique si el siguiente enunciado es verdadero o falso; fundamente:
Si dados B y I para un ciclo S existe una función f_v que cumple lo siguiente:

- $\{I \wedge B \wedge f_v = V_0\} S \{f_v > V_0\}$
- $(\exists k : \mathbb{Z})\ (I \wedge f_v \geq k \rightarrow \neg B)$

entonces el ciclo siempre termina.

Ejercicio 18. Considere la especificación de la función `existeElemento` y su implementación

Especificación

```
proc existeElemento(in s : array⟨ℤ⟩, in e : ℤ) : bool {
    requiere { True }
    asegura {
        res = True ↔
        (∃k : ℤ) (0 ≤ k < |s|) ∧_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
```

Escribir los pasos necesarios para demostrar la correctitud de la implementación respecto a la especificación usando WP y el teorema del invariante

4.2.2. Ejercicios de parcial

Ejercicio 19. Dados los siguientes ciclos y sus respectivas precondición (P_c) y poscondición (Q_c).

1. Proponer un invariante (I) y una función variante (f_v) para el ciclo
2. Demostrar los siguientes pasos de la demostración de correctitud del ciclo
   i) $P_c \rightarrow I$
   ii) $(I \wedge \neg B) \rightarrow Q_c$
   iii) $(I \wedge f_v \leq 0) \rightarrow \neg B$

a) $P_c \equiv \{s = S_0 \wedge i = 0 \wedge 0 \leq d < |s|\}$

```
while ( i < d )
  s[ i ] := e
  i := i+1
endwhile
```

$Q_c \equiv \{(\forall j : \mathbb{Z})\ (0 \leq j < d \rightarrow_L s[j] = e) \wedge (\forall j : \mathbb{Z})\ (d \leq j < |s| \rightarrow_L s[j] = S_0[j]))\}$

b) $P_c \equiv \{s = S_0 \wedge 0 \leq d < |s| \wedge i = d\}$

```
while ( i < |s| )
 s[ i ] := e
 i := i+1
endwhile
```

$Q_c \equiv \{(\forall (j : \mathbb{Z})\ (0 \leq j < d \rightarrow_L s[i] = S_0[i])) \wedge (\forall j : \mathbb{Z})\ (d \leq j < |s| \rightarrow_L s[i] = e))\}$

c) $P_c \equiv \{i = |s| - 1 \wedge res = 0\}$

```
while ( i <= 0 )
  res := res + s[ i ] + 1
  i := i - 1
endwhile
```

$Q_c \equiv \{res = |s| + \sum_{j=0}^{|s|-1} s[j]\}$

Ejercicio 20. Dado el siguiente programa con su especificación

$P_c \equiv \{n > 0 \wedge res = 1 \wedge i = 1\}$

```
while( i < n)
     res := res * i
     i := i+1;
endwhile
```

$Q_c \equiv \{res = (n-1)!\}$

a) Escriba el Invariante del ciclo

b) Demuestre formalmente que el invariante propuesto cumple los axiomas del Teorema del Invariante

c) Decida si las siguiente funciones pueden o no usarse como funciones variatnes para demostrar la terminación del ciclo y justifique muy brevemente por qué.

(i) $f_v = n$

(ii) $f_v = n - i$

(iii) $f_v = n - 1 - i$

(iv) $f_v = i - n$

(v) $f_v = (n-1)! - res$

Ejercicio 21. Dado el siguiente programa con su especificación

$P_c \equiv \{n > 0 \wedge res = 1 \wedge i = 1 \wedge j = 2\}$

```
while( i < n)
  res := res + j
  j := j*2;
  i := i+1;
endwhile
```

$Q_c \equiv \{res = \sum_{k=0}^{n-1} 2^k\}$

Contamos con el siguiente invariante, que sabemos que es incorrecto:
$I \equiv \{0 \leq i \leq n \wedge j = 2^{i+1} \wedge res = \sum_{k=0}^{i} 2^k\}$

a) Señale qué axiomas del teorema del invariante no se cumplen, sin usar demostraciones formales.

b) Escriba un invariante que resulte correcto y demuestre formalmente **únicamente** los axiomas que no se cumplían con el invariante anterior.

Ejercicio 22. Dados

• La función $fibonacci : \mathbb{N} \to \mathbb{N}$ definida como:

$$fibonacci(n) = \begin{cases} 1 & \text{si } n = 0 \text{ o } n = 1 \\ fibonacci(n-1) + fibonacci(n-2) & \text{en otro caso} \end{cases}$$

• El siguiente programa a completar, con su especificación:
$P \equiv \{n > 0\}$

```
proc fib(int n)
  s := new array<int>(n+1)
  s[0] := 1
  s[1] := 1

  i := *A COMPLETAR*
  while(*A COMPLETAR*)
    *A COMPLETAR*
  endwhile

  return s[i-1]
end proc
```

$Q_c \equiv \{res = fibonacci(n)\}$

• El siguiente invariante correcto para el ciclo dentro del programa:
$I \equiv \{n > 0 \land |s| = n + 1 \land_L s[0] = 1 \land s[1] = 1 \land$
$2 \leq i \leq |s| \land (\forall j : \mathbb{Z})\ (2 \leq j < i \to_L s[j] = s[j-1] + s[j-2])\}$

Se pide:

a) Completar el código en base al invariante y la especificación del programa

b) Explicar, sin demostrar formalmente, por qué este invariante cumple con los tres puntos del teorema del invariante

Ejercicio 23. Dados la siguiente precondición, postcondición e invariante de un ciclo

$P_c \equiv \{n > 0 \land i = 0 \land j = 2 \times n \land res = 0\}$

$I \equiv \{0 \leq i \leq n \land n \leq j \leq 2 \times n \land res = \sum_{s=0}^{i-1} s + \sum_{t=j+1}^{2n} t\}$

$Q_c \equiv \{res = \sum_{s=0}^{n-1} s + \sum_{t=n+1}^{2n} t\}$

Y el siguiente programa incompleto

```
proc DobleNCuadrado(int n)
  res := 0
  i := 0
  j := 2*n

  while [COMPLETAR]
        [COMPLETAR]
  endwhile

  return res
end proc
```

Se pide

a) Completar el programa para que corresponde al invariante dado

b) Escribir la precondición del programa

c) Explicar sin hacer las demostraciones formales, por qué este invariante cumple con los tres puntos del teorema del invariante

Ejercicio 24. Dadas las siguientes funciones variantes con un n positivo fijo, escriba un ciclo lo más sencillo posible tal que estas funciones resulten correctas para demostrar la terminación del ciclo y explique por qué. De ser necesario defina también el valor inicial de todas las variables. Si la función no se puede usar para ningún ciclo explique por qué.

a) $f_v = n - 2 \times i$

b) $f_v = i - n$

c) $f_v = n - i + 1$

d) $f_v = n + i$

e) $f_v = n + (n - i) + i$

f) $f_v = n - i - j$