Algoritmos y Estructuras de Datos I
Primer Cuatrimestre 2018

Guía Práctica 5
Demostración de corrección de ciclos en SmallLang

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

Teorema del invariante: corrección de ciclos

Ejercicio 1. ★ 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: $seq\langle\mathbb{Z}\rangle$, out result: $\mathbb{Z}$) {
  Pre {true}
  Post $\{result = \sum_{j=0}^{|s|-1} s[j]\}$
}

Implementación en SmallLang

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

Invariante de Ciclo

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

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 (que es "$i - 1$") 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 el teorema del invariante.

f) Proponer una función variante y demostrar formalmente la terminación del ciclo, usando el teorema de la función variante.

Ejercicio 2. ★ Dadas la especificación y la implementación del problema `sumarParesHastaN`, escribir la precondición y la postcondición del ciclo, y demostrar formalmente su corrección.

Especificación

proc sumarParesHastaN (in n: $\mathbb{Z}$, out result: $\mathbb{Z}$) {
  Pre $\{n \geq 0\}$
  Post $\{result = \sum_{j=0}^{n-1}(\text{if } j \ mod\ 2 = 0 \text{ then } j \text{ else } 0 \text{ fi})\}$
}

Implementación en SmallLang

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

Invariante de ciclo

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

Ejercicio 3. Supongamos que se desea implementar la función `exponenciacion`, cuya especificación es la siguiente:

```
proc exponenciacion (in m: ℤ, in n: ℤ, out result: ℤ) {
        Pre {n ≥ 0 ∧ ¬(m = 0 ∧ n = 0)}
        Post {result = m^n}
}
```

Consideremos además el siguiente invariante: $I \equiv 0 \leq i \leq n \land result = m^i$

a) Escribir un programa en SmallLang que resuelva este problema, y que incluya un ciclo que tenga a I como invariante. Demostrar formalmente la corrección de este ciclo.

b) La siguiente implementación en SmallLang es trivialmente errónea. ¿Qué punto del teorema del invariante falla en este caso?[1]

```
i := 0;
result := 0;

while( i < m ) do
  result := result * n;
  i := i + 1
endwhile
```

c) ¿Qué puede decir de la siguiente implementación en SmallLang? En caso de que sea correcta, proporcione una demostración. En caso de que sea incorrecta, explique qué punto del teorema del invariante falla.

```
i := 0;
result := 1;
while( i < n ) do
  i := i + 1;
  result := result * m
endwhile
```

d) ¿Qué puede decir de la siguiente implementación? En caso de que sea incorrecta, ¿se puede reforzar la precondición del problema para que esta especificación pase a ser correcta?

```
i := 2;
result := m*m;

while( i < n ) do
  result := result * m;
  i := i + 1
endwhile
```

Ejercicio 4. ★ Considere el problema `sumaDivisores`, dado por la siguiente especificación:

```
proc sumaDivisores (in n: ℤ, out result: ℤ) {
        Pre {n ≥ 1}
        Post {result = Σ_{j=1}^{n} (if n mod j = 0 then j else 0 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?

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

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

[1] Recordar que para mostrar que una implicación $A \rightarrow B$ no es cierta alcanza con dar valores de las variables libres que hagan que A sea verdadero y que B sea falso. Para mostrar que una tripla de Hoare $\{P\}S\{Q\}$ no es válida, alcanza con dar valores de las variables que satisfacen P, y tales que luego de ejecutar S, el estado final no satisface Q.

Ejercicio 5. Considere la siguiente especificación de la función `sumarPosicionesImpares`.

```
proc sumarPosicionesImpares (in s: seq⟨ℤ⟩, out result: ℤ) {
        Pre {true}
        Post {result = ∑_{i=0}^{|s|-1}(if i mod 2 = 1 then s[i] else 0 fi)}
}
```

a) Implementar un programa en SmallLang para resolver este problema, que incluya exactamente un ciclo con el siguiente invariante:
$I \equiv 0 \leq j \leq |s| \wedge_L result = \sum_{i=0}^{j-1}(\text{if } i \ mod\ 2 = 1 \text{ then } s[i] \text{ else } 0 \text{ fi})$

b) Demostrar formalmente la corrección del ciclo propuesto.

Ejercicio 6. Considere la siguiente especificación e implementación del problema `maximo`.

Especificación

```
proc maximo (in s: seq⟨ℤ⟩, out i: ℤ) {
    Pre {|s| ≥ 1}
    Post {0 ≤ i < |s| ∧_L
          (∀j :ℤ)(0 ≤ j < |s| →_L s[j] ≤ s[i])}
}
```

Implementación en SmallLang

```
i := 0;
j := 1;
while (j < s.size()) do
  if (s[j] > s[i])
    i := j
  else
    skip
  endif;
  j:=j+1
endwhile
```

a) Escribir la precondición y la postcondición del ciclo.

b) Demostrar que el ciclo es parcialmente correcto, utilizando el siguiente invariante:

$$I \equiv (0 \leq i < |s| \wedge 1 \leq j \leq |s|) \wedge_L (\forall k : \mathbb{Z})(0 \leq k < j \longrightarrow_L s[k] \leq s[i])$$

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

Ejercicio 7. ★ Considere la siguiente especificación e implementación del problema `copiarSecuencia`.

Especificación

```
proc copiarSecuencia (in s: seq⟨ℤ⟩, inout r: seq⟨ℤ⟩) {
    Pre {|s| = |r| ∧ r = r_0}
    Post {|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
```

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 8. Considere la siguiente especificación e implementación del problema `llenarSecuencia`.

Especificación

```
proc llenarSecuencia (inout s: seq⟨ℤ⟩, in d: ℤ, in e: ℤ) {
    Pre {d ≥ 0 ∧ d < |s| ∧ s = S_0}
    Post {|s| = |S_0| ∧_L
          (∀j :ℤ)(0 ≤ j < d →_L s[j] = S_0[j]) ∧
          (∀j :ℤ)(d ≤ j < |s| →_L s[j] = e)}
}
```

Implementación en SmallLang

```
i := d;
while (i < s.size()) do
  s[i]:=e;
  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 9. ★ 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
```

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

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

a) 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.

Ejercicio 10. Considere la siguiente especificación del problema `reemplazarTodos`.

proc `reemplazarTodos` (inout s: $seq\langle\mathbb{Z}\rangle$, in a: $\mathbb{Z}$, in b: $\mathbb{Z}$) {
  `Pre` $\{s = S_0\}$
  `Post` $\{|s| = |S_0| \land_L$
    $(\forall j : \mathbb{Z})((0 \leq j < |s| \land_L S_0[j] = a) \rightarrow_L s[j] = b) \land$
    $(\forall j : \mathbb{Z})(d \leq j < |s| \land_L S_0[j] \neq a) \rightarrow_L s[j] = S_0[j])\}$
}

a) Dar un programa en SmallLang que implemente la especificación dada.

b) Escribir la precondición y la postcondición del ciclo.

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

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

Demostración de correctitud: programas completos

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

Especificación

proc `indice` (in s: $seq\langle\mathbb{Z}\rangle$, in e: $\mathbb{Z}$, out r: $\mathbb{Z}$) {
  `Pre` $\{True\}$
  `Post` $\{r = -1 \rightarrow$
    $(\forall j : \mathbb{Z})(0 \leq j < |s| \rightarrow_L s[j] \neq e)$
    $\land$
    $r \neq -1 \rightarrow$
    $(0 \leq r < |s| \land_L s[r] = e)\}$
}

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: seq⟨ℤ⟩, in e: ℤ, out r: Bool) {
    Pre {True}
    Post {r = 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)
  r := true
else
  r := false
endif
```

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

Especificación

```
proc esSimetrico (in s: seq⟨ℤ⟩, out r:Bool) {
    Pre {True}
    Post {r = True ↔ (∀i : ℤ)(0 ≤ i < |s| →_L
          s[i] = s[|s| − (i + 1)])}
}
```

Implementación en SmallLang

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

Ejercicio 14. ★ Demostrar que el siguiente programa es correcto respecto a la especificación dada.

Especificación

```
proc concatenarSecuencias (in a: seq⟨ℤ⟩,
in b: seq⟨ℤ⟩,
inout r:seq⟨ℤ⟩) {
    Pre {|r| = |a| + |b| ∧ r = R0}
    Post {|r| = |R0| ∧ (∀j :ℤ)(0 ≤ j < |a| →L r[j] = a[j]) ∧
          (∀j :ℤ)(0 ≤ j < |b| →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
```

Ejercicio 15. Dar dos programas en SmallLang que satisfagan la siguiente especificación, y demostrar que ambos son correctos.

```
proc buscarPosicionUltimoMaximo (in s: seq⟨ℤ⟩, out r:seq⟨ℤ⟩) {
    Pre {|s| > 0}
    Post {0 ≤ r < |s| ∧L
        (∀j :ℤ)(0 ≤ j < r →L s[r] ≥ s[j]) ∧
        (∀j :ℤ)(r < j < r →L s[r] > s[j])}
}
```