

Algoritmos y Estructuras de Datos

Segundo cuatrimestre de 2024

Departamento de Computación - FCEyN - UBA

Correctitud de ciclos

1

Repaso: Triplas de Hoare

- Consideremos la siguiente tripla de Hoare:

$$\{P\} S \{Q\}.$$

- Esta tripla es **válida** si se cumple que:

1. Si el programa S comienza en un estado que cumple P ...
2. ... entonces termina luego de un número finito de pasos ...
3. ... Y además en un estado que cumple Q .

2

Repaso: Lenguaje SmallLang

- Definimos un lenguaje imperativo basado en **variables** y las siguientes instrucciones:

1. **Nada**: Instrucción **skip** que no hace nada.
2. **Asignación**: Instrucción $x := E$.

- Además, tenemos las siguientes estructuras de control:

1. **Secuencia**: **S1; S2** es un programa, si **S1** y **S2** son dos programas.
2. **Condicional**: **if B then S1 else S2 endif** es un programa, si **B** es una expresión lógica y **S1** y **S2** son dos programas.
3. **Ciclo**: **while B do S endwhile** es un programa, si **B** es una expresión lógica y **S** es un programa.

3

Repaso: Precondición más débil

- **Definición.** La **precondición más débil** de un programa **S** respecto de una postcondición Q es el predicado P más débil posible tal que $\{P\}S\{Q\}$.

- **Notación.** $wp(S, Q)$.

- **Teorema:** Decimos que $\{P\} S \{Q\}$ es válida sii $P \Rightarrow_L wp(S, Q)$

4

Repaso: Axiomas wp

- ▶ **Axioma 1.** $wp(x := E, Q) \equiv \text{def}(E) \wedge_L Q_E^x$.
- ▶ **Axioma 2.** $wp(\text{skip}, Q) \equiv Q$.
- ▶ **Axioma 3.** $wp(S1; S2, Q) \equiv wp(S1, wp(S2, Q))$.
- ▶ **Axioma 4.** $wp(\text{if } B \text{ then } S1 \text{ else } S2 \text{ endif}, Q) \equiv$
$$\text{def}(B) \wedge_L \left((B \wedge wp(S1, Q)) \vee \right.$$
$$\left. (\neg B \wedge wp(S2, Q)) \right)$$
- ▶ **Observación:** $wp(b[i] := E, Q) \equiv wp(b := \text{setAt}(b, i, E), Q)$

5

Ciclos (repaso)

- ▶ Recordemos la **sintaxis** de un ciclo:

```
while (guarda B) {  
    cuerpo del ciclo  $S_c$   
}
```
- ▶ Se repite el cuerpo del ciclo S mientras la **guarda** B se cumpla, cero o más veces. Cada repetición se llama una **iteración**.
- ▶ La ejecución del ciclo **termina** si no se cumple la guarda al comienzo de su ejecución o bien luego de ejecutar una iteración.
- ▶ Si/cuando el ciclo termina, el estado resultante es el estado posterior a la última instrucción del cuerpo del ciclo.

6

¿Es válida la siguiente tripla de Hoare?

$\{n \geq 0 \wedge i = 1 \wedge s = 0\}$

```
while (i <= n) do  
    s := s + i;  
    i := i + 1  
endwhile
```

$\{s = \sum_{k=1}^n k\}$

7

Invariante de un ciclo

- ▶ **Definición.** Un predicado I es un **invariante** de un ciclo si:
 1. I vale antes de comenzar el ciclo, y
 2. si vale $I \wedge B$ al comenzar una iteración arbitraria, entonces sigue valiendo I al finalizar la ejecución del cuerpo del ciclo.
- ▶ Un invariante describe un estado que se satisface cada vez que comienza la ejecución del cuerpo de un ciclo y también se cumple cuando la ejecución del cuerpo del ciclo concluye.
- ▶ Por ejemplo, otros invariantes para este ciclo son:

```
while (i <= n) do  
    s := s + i;  
    i := i + 1  
endwhile
```

- ▶ $I' \equiv i \neq 0$
- ▶ $I'' \equiv s \geq 0$
- ▶ $I''' \equiv i \geq 1$
- ▶ $I''' \equiv i \geq 1 \wedge s = \sum_{k=1}^{i-1} k$
- ▶ ...etc

8

Teorema del invariante

- **Teorema del invariante.** Si existe un predicado I tal que ...

1. $P_C \Rightarrow I$,
2. $\{I \wedge B\} S \{I\}$,
3. $I \wedge \neg B \Rightarrow Q_C$,

entonces el ciclo **while(B) {S}** es parcialmente correcto respecto de la especificación (P_C, Q_C) .

- Este teorema es la **herramienta principal** para argumentar la corrección de ciclos.
- **Nota:** No cualquier predicado que sea invariante del ciclo nos va a servir para comprobar correctitud del mismo.

9

Ejemplo: Sumatoria

Primer punto del teorema del invariante

- Verifiquemos estas tres condiciones con el ejemplo anterior,

```
{n ≥ 0 ∧ i = 1 ∧ s = 0}
while (i ≤ n) do
  s := s + i;
  i := i + 1
endwhile
{s = ∑k=1n k}
```

1. $P_C \equiv n \geq 0 \wedge i = 1 \wedge s = 0$
2. $Q_C \equiv n \geq 0 \wedge s = \sum_{k=1}^n k$
3. $B_C \equiv i \leq n$
4. $I \equiv i \geq 1 \wedge s = \sum_{k=1}^{i-1} k$

Nota: Elegimos uno de los invariantes previos como candidato.

- En primer lugar, debemos verificar que $P_C \Rightarrow I$:

$$(n \geq 0 \wedge i = 1 \wedge s = 0) \Rightarrow i \geq 1 \wedge s = \sum_{k=1}^{i-1} k.$$

- Por lo tanto, podemos concluir que se cumple la condición $P_C \Rightarrow I$

10

Ejemplo: Sumatoria

Segundo punto del teorema del invariante

- En segundo lugar debemos demostrar $\{I \wedge B\} S \{I\}$?

$$\begin{aligned} I \wedge B : \{i \leq n \wedge i \geq 1 \wedge s = \sum_{k=1}^{i-1} k\} \\ s = s + i; \\ i = i + 1; \\ I : \{i \geq 1 \wedge s = \sum_{k=1}^{i-1} k\} \end{aligned}$$

- Dado que queremos demostrar una tripla de Hoare, lo hacemos viendo que $\{I \wedge B\} \Rightarrow wp(S_C, I)$.

$$wp(S_C, I) \equiv i \geq 0 \wedge s = \sum_{k=1}^{i-1} k$$

- Por lo tanto, podemos concluir que $\{I \wedge B\} \Rightarrow wp(S_C, I)$

11

Ejemplo: Sumatoria

Tercer punto del teorema del invariante

- Finalmente, debemos demostrar si $I \wedge \neg B \Rightarrow Q_C$?

$$i \geq 1 \wedge s = \sum_{k=1}^{i-1} k \wedge i > n \Rightarrow s = \sum_{k=1}^n k ?$$

- **¡No!** Contraejemplo: Si $i = n + 2$, entonces ¡la implicación no vale!
- Sin embargo, **sabemos** que esto no puede pasar, puesto que $i \leq n + 1$ a lo largo del ciclo.
- ¿Qué hacemos?
- ⇒ ¡Reforzamos el invariante!

12

Ejemplo: Sumatoria

Nuevo invariante propuesto

- Proponemos el nuevo invariante de ciclo reforzado (i.e. mas restrictivo):

$$I \equiv 1 \leq i \leq n+1 \wedge s = \sum_{k=1}^{i-1} k$$

- Habría que volver a demostrar los dos primeros puntos del teorema (tarea para el hogar):

1. $P_C \Rightarrow I$,
2. $\{I \wedge B\} S \{I\}$,

- ¿Vale ahora que tenemos que $I \wedge \neg B \Rightarrow Q_C$?

$$1 \leq i \leq n+1 \wedge s = \sum_{k=1}^{i-1} k \wedge i > n \Rightarrow i = n+1 \wedge s = \sum_{k=1}^{i-1} k \\ \Rightarrow s = \sum_{k=1}^n k \equiv Q_C$$

13

Ejemplo: Sumatoria

Resultado final

- Finalmente, Sean:

1. $P_C \equiv n \geq 0 \wedge i = 1 \wedge s = 0$
2. $Q_C \equiv n \geq 0 \wedge s = \sum_{k=1}^n k$
3. $B_C \equiv i \leq n$
4. $I \equiv 1 \leq i \leq (n+1) \wedge s = \sum_{k=1}^{i-1} k$

- Ya que demostramos que se cumplen las siguientes condiciones:

1. $P_C \Rightarrow I$
2. $\{I \wedge B\}$ cuerpo del ciclo $\{I\}$
3. $I \wedge \neg B \Rightarrow Q_C$

- Entonces, por el Teorema del Invariante podemos concluir que el ciclo while(B) S es **parcialmente correcto** respecto de la especificación P_C, Q_C .

14

Ejemplo: Sumatoria

Algunas observaciones

- $I \equiv 1 \leq i \leq n+1 \wedge s = \sum_{k=1}^{i-1} k$.

1. El invariante refleja la **hipótesis inductiva** del ciclo.
2. En general, un buen invariante debe incluir el **rango** de la(s) **variable(s) de control** del ciclo.
3. Además, debe incluir alguna afirmación sobre el **acumulador** del ciclo.

- Cuando tenemos un invariante I que permite demostrar la corrección parcial del ciclo, nos referimos a I como **el invariante** del ciclo.

1. El invariante de un ciclo **caracteriza** las acciones del ciclo, y representa al las **asunciones** y **propiedades** que hace nuestro **algoritmo** durante el ciclo.

- En general, es sencillo argumentar **informalmente** la terminación del ciclo (más detalles luego).

15

Ejemplo: Sumatoria

Para concluir... esta parte

- **Ojo:** Para probar esto:

$$\{n \geq 0 \wedge i = 1 \wedge s = 0\}$$

while (i <= n) do

 s := s + i;

 i := i + 1

endwhile

$$\{s = \sum_{k=1}^n k\}$$

- Nos falta demostrar que si vale P_C el ciclo siempre termina.
- Por ahora, solo probamos que es parcialmente correcto¹
- ¿Cómo podemos probar si dada una precondición, un ciclo siempre termina?
- Para eso tenemos el **Teorema de terminación**

¹Cuando termina, cumple Q_C , pero no sabemos si siempre termina

16

Teorema de terminación de un ciclo

- **Teorema.** Sea \mathbb{V} el producto cartesiano de los dominios de las variables del programa y sea I un invariante del ciclo **while B do S endwhile**. Si existe una función $fv : \mathbb{V} \rightarrow \mathbb{Z}$ tal que

1. $\{I \wedge B \wedge v_0 = fv\} \text{ S } \{fv < v_0\}$,
2. $I \wedge fv \leq 0 \Rightarrow \neg B$,

... entonces la ejecución del ciclo **while B do S endwhile** **siempre termina**.

- La función fv se llama **función variante** del ciclo.
- El Teorema de terminación nos permite demostrar que un ciclo termina (i.e. no se cuelga).

17

Ejemplo: Sumatoria

Terminación

- Sea la siguiente tripla de Hoare:

$$\{n \geq 0 \wedge i = 1 \wedge s = 0\}$$

```
while (i <= n) do
  s = s + i;
  i = i + 1;
endwhile
```

$$\{s = \sum_{k=1}^n k\}$$

- Ya probamos que el siguiente predicado es un invariante de este ciclo.

$$I \equiv 1 \leq i \leq n + 1 \wedge s = \sum_{k=1}^{i-1} k$$

- ¿Cuál sería una buena función variante para este ciclo?

18

Ejemplo: Sumatoria

Terminación

- Ejecutemos el ciclo con $n = 6$.

Iteración	i	s	n	n+1-i
0	1	0	6	6
1	2	1	6	5
2	3	3	6	4
3	4	6	6	3
4	5	10	6	2
5	6	15	6	1
6	7	21	6	0

- Una función variante representa una **cantidad que se va reduciendo** a lo largo de las iteraciones. En este caso es la cantidad de índices que falta sumar.
- Proponemos entonces $fv = n + 1 - i$

19

Ejemplo: Sumatoria

Terminación: fv decrece

- Veamos que se cumplen las dos condiciones del teorema.

1. Para verificar que $\{I \wedge B \wedge fv = v_0\} \text{ S } \{fv < v_0\}$ para todo v_0 , calculamos $wp(S, fv < v_0)$.

$$\begin{aligned}
 & wp(s := s + i; i := i + 1, fv < v_0) \\
 & \equiv wp(s := s + i; i := i + 1, (n + 1 - i) < v_0) \\
 & \equiv wp(s := s + i, wp(i := i + 1, (n + 1 - i) < v_0)) \\
 & \equiv wp(s := s + i, def(i + 1) \wedge_L (n + 1 - (i + 1)) < v_0) \\
 & \equiv wp(s := s + i, (n + 1 - (i + 1)) < v_0) \\
 & \equiv def(s + i) \wedge_L n - i < v_0 \\
 & \equiv n - i < v_0 \\
 & \equiv n - i < n + 1 - i \\
 & \equiv n - i < n - i + 1 \checkmark
 \end{aligned}$$

20

Ejemplo: Sumatoria

Terminación: fv llega a 0

► Veamos que se cumplen las dos condiciones del teorema.

2. Verifiquemos que $I \wedge fv \leq 0 \Rightarrow \neg B$

$$\begin{aligned} I \wedge fv \leq 0 &\equiv \overbrace{1 \leq i \leq n+1}^I \wedge s = \sum_{k=1}^{i-1} k \wedge \overbrace{n+1-i \leq 0}^{fv \leq 0} \\ &\Rightarrow i \leq n+1 \wedge n+1-i \leq 0 \\ &\Rightarrow i \leq n+1 \wedge n+1 \leq i \\ &\Rightarrow i = n+1 \\ &\Rightarrow \neg(i \leq n) \\ &\Rightarrow \neg B \checkmark \end{aligned}$$

21

Ejemplo: Sumatoria

Demostración completa

Recapitulando, sean

- $I \equiv 1 \leq i \leq n+1 \wedge s = \sum_{k=1}^{i-1} k$
- $fv = n+1-i$

Ya habíamos probado que el ciclo es **parcialmente** correcto dado que:

1. $P_C \Rightarrow I$
2. $\{I \wedge B\} S \{I\}$
3. $I \wedge \neg B \Rightarrow Q_C$

Ahora acabamos de probar que el ciclo siempre termina ya que:

4. $\{I \wedge B \wedge v_0 = fv\} S \{fv < v_0\}$,
5. $I \wedge fv \leq 0 \Rightarrow \neg B$,

Por lo tanto, por (1)-(5) tenemos (finalmente) que ...

22

Ejemplo: Sumatoria

Demostración completa

► Que la siguiente tripla de Hoare:

$$\{P_C : n \geq 0 \wedge i = 1 \wedge s = 0\}$$

```
while (i <= n) do
  s = s + i;
  i = i + 1;
endwhile
```

$$\{Q_C : s = \sum_{k=1}^n k\}$$

es una tripla de Hoare **válida!**

► Esto significa que:

1. Si el ciclo comienza en un estado que cumple P_C
2. ... entonces termina luego de un número finito de pasos
3. y además en un estado que cumple Q_C

23

Otro ejemplo: Búsqueda lineal

- El problema de búsqueda por valor de un elemento en una secuencia es uno de los problemas fundamentales de la Informática.
- Vamos a aprovecharlo para aplicar el Teorema del Invariante y explorar su relación con el diseño de algoritmos
- Especificado formalmente:

```
proc contiene(in s : seq(Z), in x : Z) : Bool{
  Pre { True }
  Post { result = true ↔ (∃ i : Z)(0 ≤ i < |s| ∧ s[i] = x) }
}
```
- ¿Cómo podemos buscar un elemento en una secuencia?

24

Búsqueda lineal

s[0]	s[1]	s[2]	s[3]	s[4]	...	s[s - 1]
= x? ≠ x	= x? ≠ x	= x? ≠ x	= x? ≠ x			= x? ≠ x
↑	↑	↑	↑			↑
i	i	i	i			i

- ¿Cómo lo podemos implementar en Java?

```
boolean contiene(int[] s, int x) {
    int i = 0;
    while (i < s.length && s[i] != x) {
        i = i + 1;
    }
    return i < s.length;
}
```

25

Búsqueda lineal

```
boolean contiene(int[] s, int x) {
    int i = 0;
    while (i < s.length && s[i] != x) {
        i = i + 1;
    }
    return i < s.length;
}
```

- ¿Qué invariante de ciclo podemos proponer?

$$I \equiv 0 \leq i \leq |s| \wedge_L (\forall j : \mathbb{Z})(0 \leq j < i \rightarrow_L s[j] \neq x)$$

- ¿Qué función variante podemos usar?

$$fv = |s| - i$$

- ¿Es la implementación correcta con respecto a la especificación?

26

Recap: Teorema de corrección de un ciclo

- **Teorema.** Sean un predicado I y una función $fv : \mathbb{V} \rightarrow \mathbb{Z}$ (donde \mathbb{V} es el producto cartesiano de los dominios de las variables del programa), y supongamos que $I \Rightarrow \text{def}(B)$. Si

1. $P_C \Rightarrow I$,
2. $\{I \wedge B\} S \{I\}$,
3. $I \wedge \neg B \Rightarrow Q_C$,
4. $\{I \wedge B \wedge v_0 = fv\} S \{fv < v_0\}$,
5. $I \wedge fv \leq 0 \Rightarrow \neg B$,

... entonces la siguiente tripla de Hoare es válida:

$\{P_C\} \text{while } B \text{ do } S \text{ endwhile } \{Q_C\}$

27

Búsqueda lineal

Especificación

```
proc contiene(in s : seq(Z), in x : Z) : Bool{
    Pre { True }
    Post { result = true ↔
          (∃ i : Z)(0 ≤ i < |s| ∧ s[i] = x) }
}
```

Algoritmo

```
boolean contiene(int[] s, int x) {
    int i = 0;
    while (i < s.length && s[i] != x) {
        i = i + 1;
    }
    bool res = i < s.length;
    return res;
}
```

- Para este ciclo, tenemos:

- $P_C \equiv$ ¿Cómo llegamos a la P_C del ciclo?
 - Partimos de la P de la especificación y vamos agregando la información necesaria hacia abajo.
 - $P_C \equiv i = 0$
- $Q_C \equiv$ ¿Cómo llegamos a la Q_C del ciclo?
 - Partimos de la Q de la especificación y calculamos la Q_C hacia arriba
 - $Q_C \equiv wp(S_{\text{postCiclo}}, Q)$
 - $Q_C \equiv (i < |s|) \leftrightarrow (\exists j : \mathbb{Z})(0 \leq j < |s| \wedge_L s[j] = x)$.
- $B \equiv i < |s| \wedge_L s[i] \neq x$
- $I \equiv 0 \leq i \leq |s| \wedge_L (\forall j : \mathbb{Z})(0 \leq j < i \rightarrow_L s[j] \neq x)$
- $fv = |s| - i$

28

Recap: Teorema de corrección de un ciclo

1. $P_C \Rightarrow I$,
2. $\{I \wedge B\} S \{I\}$,
3. $I \wedge \neg B \Rightarrow Q_C$,
4. $\{I \wedge B \wedge v_0 = f_v\} S \{f_v < v_0\}$,
5. $I \wedge f_v \leq 0 \Rightarrow \neg B$,

En otras palabras, hay que mostrar que:

- ▶ I es un invariante del ciclo (punto 1. y 2.)
- ▶ Se cumple la postcondición del ciclo a la salida del ciclo (punto 3.)
- ▶ La función variante es estrictamente decreciente (punto 4.)
- ▶ Si la función variante alcanza la cota inferior la guarda se deja de cumplir (punto 5.)

29

Corrección de búsqueda lineal

¿ I es un invariante del ciclo?

$$I \equiv 0 \leq i \leq |s| \wedge_L (\forall j : \mathbb{Z})(0 \leq j < i \rightarrow_L s[j] \neq x)$$

- ▶ La variable i toma el primer valor 0 y se incrementa por cada iteración hasta llegar a $|s|$.
- ▶ $\Rightarrow 0 \leq i \leq |s|$
- ▶ En cada iteración, todos los elementos a izquierda de i son distintos de x
- ▶ $\Rightarrow (\forall j : \mathbb{Z})(0 \leq j < i \rightarrow_L s[j] \neq x)$

30

Corrección de búsqueda lineal

¿Se cumple la postcondición del ciclo a la salida del ciclo?

$$I \equiv 0 \leq i \leq |s| \wedge_L (\forall j : \mathbb{Z})(0 \leq j < i \rightarrow_L s[j] \neq x)$$

$$Q_C \equiv (i < |s|) \leftrightarrow (\exists i : \mathbb{Z})(0 \leq i < |s| \wedge_L s[i] = x)$$

- ▶ Al salir del ciclo, no se cumple la guarda. Entonces no se cumple $i < |s|$ o no se cumple $s[i] \neq x$
 - ▶ Si no se cumple $i < |s|$, no existe ninguna posición que contenga x
 - ▶ Si no se cumple $s[i] \neq x$, existe al menos una posición que contiene a x

31

Corrección de búsqueda lineal

¿Es la función variante estrictamente decreciente?

$$f_v = |s| - i$$

- ▶ En cada iteración, se incrementa en 1 el valor de i
- ▶ Por lo tanto, en cada iteración se reduce en 1 la función variante.

32

Corrección de búsqueda lineal

¿Si la función variante alcanza la cota inferior la guarda se deja de cumplir?

$$fv = |s| - i$$

$$B \equiv i < |s| \wedge_L s[i] \neq x$$

- ▶ Si $fv = |s| - i \leq 0$, entonces $i \geq |s|$
- ▶ Como siempre pasa que $i \leq |s|$, entonces es cierto que $i = |s|$
- ▶ Por lo tanto $i < |s|$ es falso.

33

Corrección de búsqueda lineal

▶ Finalmente, ahora que probamos que:

1. $P_C \Rightarrow I$,
2. $\{I \wedge B\} S \{I\}$,
3. $I \wedge \neg B \Rightarrow Q_C$,
4. $\{I \wedge B \wedge v_0 = fv\} S \{fv < v_0\}$,
5. $I \wedge fv \leq 0 \Rightarrow \neg B$,

▶ ...podemos por el teorema concluir que el ciclo termina y es correcto.

34

Bibliografía

- ▶ David Gries - The Science of Programming
 - ▶ Part II - The Semantics of a Small Language
 - ▶ Chapter 11 - The Iterative Command

35