Algoritmos y Estructuras de Datos

Segundo cuatrimestre - 2025

Departamento de Computación - FCEyN - UBA

Correctitud de ciclos

Repaso: Triplas de Hoare

- Consideremos la siguiente tripla de Hoare:

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

- 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.

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.

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

Repaso: Axiomas wp

- **Axioma 1.** $wp(\texttt{x := E}, Q) \equiv \text{def}(E) \wedge_L Q^x_E$.
- **Axioma 2.** $wp(\texttt{skip}, Q) \equiv Q$.
- **Axioma 3.** $wp(\texttt{S1; S2}, Q) \equiv wp(\texttt{S1}, wp(\texttt{S2}, Q))$.
- **Axioma 4.** $wp(\texttt{if B then S1 else S2 endif}, Q) \equiv$

$$def(B) \wedge_L \Big((B \wedge wp(\texttt{S1}, Q)) \vee (\neg B \wedge wp(\texttt{S2}, Q))\Big)$$

- **Observación:** $wp(\texttt{b[i]:=E}, Q) \equiv wp(\texttt{b:=setAt(b,i,E)}, Q)$

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.

¿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
```

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

Precondición más débil de un ciclo

- Supongamos que tenemos el ciclo **while B do S endwhile**.
- Si supieramos que la ejecución del ciclo termina en exactamente k iteraciones satisfaciendo Q, podríamos decirlo ¿cómo?
- Si supiéramos que el ciclo realiza a lo sumo k iteraciones, entonces podríamos decirlo ¿cómo?: con una gran disyunción: se cumple la WP para que termine en 0 pasos O se cumple la wp para que termine en 1 paso O ...
- ¡Pero no lo sabemos!

Ejemplo

$\{???\}$

```
while (0<i && i<3) do
  i := i +1
endwhile
```

$\{i = 3\}$

- A lo sumo, se va a ejecutar 2 veces el cuerpo del ciclo
- ¿Cuál es la precondición más débil?

$$
\begin{aligned}
& wp(\texttt{while } 0 < i < 3 \texttt{ do i:=i+1 endwhile}, i = 3) \\
\equiv \quad & i = 1 \lor i = 2 \lor i = 3
\end{aligned}
$$

Pero...

$\{???\}$

```
while (0<i && i<n) do
  i := i +1
endwhile
```

$\{i \geq 0\}$

- ¿Cuántas veces se va a ejecutar el cuerpo del ciclo?
- ¿Podemos usar la idea anterior para conocer la precondición más débil?
- ¡No! Porque no podemos fijar a priori una cota superior a la cantidad de iteraciones que va a realizar el ciclo.
- Y no podemos escribir una disyunción infinita...

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 $I \land B$ 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, algunos invariantes para este ciclo son:

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

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

Teorema del invariante

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

  1. $P_C \Rightarrow \mathrm{I}$,
  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.

Ejemplo: Sumatoria

Primer punto del teorema del invariante

- Verifiquemos estas tres condiciones con el ejemplo anterior,

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

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

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

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

Nota: Elegimos uno de los invariantes previos como candidato.

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

$$\big(n \geq 0 \land i = 1 \land s = 0\big) \;\Rightarrow\; i \geq 1 \land \mathrm{s} = \sum_{k=1}^{i-1} k.$$

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

Ejemplo: Sumatoria

Segundo punto del teorema del invariante

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

$$I \land B : \{i \leq n \land i \geq 1 \land \mathsf{s} = \sum_{k=1}^{i-1} k\}$$

```
s = s + i;
i = i + 1;
```

$$I : \{i \geq 1 \land \mathsf{s} = \sum_{k=1}^{i-1} k\}$$

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

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

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

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?

$\Rightarrow$ ¡Reforzamos el invariante!

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. $P_C \Rightarrow \mathrm{I}$,
  2. $\{I \wedge B\}\ \mathtt{S}\ \{I\}$,
- ¿Vale ahora que tenemos que $I \wedge \neg B \Rightarrow Q_C$?

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

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.

Ejemplo: Sumatoria

Algunas observaciones

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

  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.

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

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

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 correcto, o lo que es lo mismo, que si termina lo hace cumpliendo Q_C,
- ¡pero no sabemos si siempre termina!
- ¿Cómo podemos probar si, dada una precondición, un ciclo siempre termina?
- Para eso tenemos el Teorema de terminación

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 $fv : \mathbb{V} \to \mathbb{Z}$ tal que

  1. $\{I \wedge B \wedge v_0 = fv\}$ **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).

Ejemplo: Sumatoria

Terminación

- Sea la siguiente tripla de Hoare:

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

  ```
  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.

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

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

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 podría ser la cantidad de índices que falta sumar.

- Proponemos entonces $fv = n + 1 - i$

Ejemplo: Sumatoria

Terminación: fv decrece

- Veamos que se cumplen las dos condiciones del teorema.

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

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

Ejemplo: Sumatoria

Terminación: fv llega a 0

- Veamos que se cumplen las dos condiciones del teorema.

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

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

Ejemplo: Sumatoria

Demostración completa

Recapitulando, sean

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

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

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

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

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

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

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

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:

  $\text{proc } \mathit{contiene}(\text{in } s : \mathit{seq}\langle\mathbb{Z}\rangle, \text{in } x : \mathbb{Z}) : \text{Bool}\{$

  $\quad \text{Pre } \{\mathit{True}\}$

  $\quad \text{Post } \{\mathit{result} = \mathit{true} \leftrightarrow (\exists i : \mathbb{Z})(0 \leq i < |s| \land_L s[i] = x)\}$

  $\}$
- ¿Cómo podemos buscar un elemento en una secuencia?

Búsqueda lineal

s[0]	s[1]	s[2]	s[3]	s[4]	...	s[$\|s\| - 1$]
$= x? \neq x$	$= x? \neq x$	$= x? \neq x$	$= x? \neq x$			$= x? \neq x$
$\uparrow$	$\uparrow$	$\uparrow$	$\uparrow$			$\uparrow$
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;
}
```

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?

Recap: Teorema de corrección de un ciclo

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

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

Búsqueda lineal

Especificación

proc *contiene*(in s : $seq\langle\mathbb{Z}\rangle$, in x : $\mathbb{Z}$) : Bool{
  Pre $\{True\}$
  Post $\{result = true \leftrightarrow$
  $(\exists i : \mathbb{Z})(0 \leq i < |s| \wedge_L s[i] = x)\}$
}

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"
    - $Q_C \equiv wp(S_{postCiclo}, Q)$
    - $Q_C \equiv (i < |s|) \leftrightarrow (\exists j : \mathbb{Z})(0 \leq j < |s| \wedge_L s[j] = x)$.
  - $B \equiv i < |s| \wedge_L s[i] \neq x$
  - $I \equiv 0 \leq i \leq |s| \wedge_L (\forall j : \mathbb{Z})(0 \leq j < i \rightarrow_L s[j] \neq x)$
  - $fv = |s| - i$

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

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.)

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
- $\Rightarrow (\forall j : \mathbb{Z})(0 \leq j < i \rightarrow_L s[j] \neq x)$

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

Corrección de búsqueda lineal

¿Es la función variante estrictamente decreciente?

$$fv = |s| - i$$

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

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.

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.

Bibliografía

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