Departamento de Ingeniería Informática
Grado en Ingeniería Informática
Elisa Guerrero Vázquez
Esther L. Silva Ramírez

Metodología de la Programación

Tema 2 – Teoría

VERIFICACIÓN

Demostrar la corrección de un algoritmo

- Validación Dinámica ➡ Depuración
  - Nunca habrá certeza completa de que el algoritmo funciona correctamente en todos los casos posibles

- Validación Estática ➡ **Verificación**

Un algoritmo es correcto si:

Comportamiento real = Comportamiento deseado

Comportamiento deseado se expresa en la especificación
Verificar = demostrar formalmente algoritmo cumple especificación

Especificación de un algoritmo

- **Estado:** permite describir las condiciones que deben reunir los datos de entrada, los resultados, etc. $y<=0$
- **Estado =** Aplicación de las variables en el conjunto de sus valores $x=9$

  Expresión de tipo lógico: $x>y$

- **Aserto =** { Conjunto de Estados }= $\{x=9 \wedge y<=0 \wedge x>y\}$
  - Permiten especificar el comportamiento que se espera del algoritmo
  - Permiten calcular su comportamiento real
  - Documentan el programa

Notación de Hoare

{P}	Precondiciones
S	Sentencias
{Q}	Postcondiciones

$\{x>0 \wedge y=-1\}$	Precondiciones: x positivo, y= -1
$x \leftarrow x*y$	Multiplicar x por y
$\{x<0 \wedge y=-1\}$	Postcondiciones: x negativo, y=-1

Verificación del Algoritmo

Demostración formal de que el algoritmo cumple su especificación:

- **Corrección Parcial**: si el algoritmo acaba entonces el resultado es correcto.

- **Corrección Total**: para todo dato de entrada válido el algoritmo acaba y el resultado es correcto.

Base de comprobación

- **Corrección Parcial**: si el algoritmo acaba el resultado es correcto.
  - **Demostrar que se cumple la postcondición:**
    1. Anotar al inicio y al final del programa las precondiciones y postcondiciones
    2. Incluir asertos en los puntos intermedios del programa que describan el estado en ese punto
    3. Demostrar que si se cumple un aserto en un punto del programa y se siguen cada una de las líneas de ejecución posible hasta llegar a otro aserto, dicho aserto ha de cumplirse (aplicando las leyes de la lógica y de acuerdo a las acciones realizadas por el programa)

Base de comprobación

- **Corrección Total:** el algoritmo acaba y el resultado es correcto.
  - Corrección Parcial +
  - Demostración de que todos los bucles acaban en un nº finito de pasos, (función monótona con valor acotado)

Propiedades

$A_1 \Rightarrow A_2$ A_1 es un aserto más fuerte que A_2

A_1 es un subconjunto de A_2

Supongamos que esta especificación es correcta:

$\{x \leq 5\}$

S

$\{x \leq 10\}$

- También será correcta si buscamos otro aserto inicial R:{x<=0} que implique la precondición $R \Rightarrow P$
- También será correcta si buscamos otro aserto final T:{x<=45} que sea implicado por la postcondición $Q \Rightarrow T$

Reglas de Consecuencia

Primera Regla de Consecuencia

Si {P} S {Q} **y** $R \Rightarrow P$ **entonces**

{R} S {Q}

Reforzar la Precondición

Segunda Regla de Consecuencia

Si {P} S {Q} **y** $Q \Rightarrow T$ **entonces**

{P} S {T}

Debilitar la Postcondición

$A_1 \Rightarrow A_2$ A_1 es un aserto <u>más fuerte</u> que A_2

A_1 es un <u>subconjunto</u> de A_2

Supongamos que esta especificación es correcta:

P:{x≤5}
S
Q:{x ≤10}

Esta especificación es correcta

R:{x≤0}
S
Q:{x ≤10}

porque $R \Rightarrow P$

REFORZAMIENTO DE LA PRECONDICIÓN

Esta especificación es correcta

P:{x≤5}
S
T:{x ≤45}

porque $Q \Rightarrow T$

DEBILITAMIENTO DE LA POSTCONDICIÓN

$A_1 \Rightarrow A_2$ A_1 es un subconjunto de A_2

Primera Regla de Consecuencia

Si {P} S {Q} **y** R⇒P **entonces**

{R} S {Q}

P:{x≤5}	**R:{x≤0}**
S	S
Q:{x ≤10}	Q:{x ≤10}
Especificación 1 es correcta	¿ Especificación 2 ?

R ⇒ P

por tanto la Especificación 2 también es correcta

Ejemplos

- Demuestra la Corrección de las siguientes especificaciones:

$\{x >= 18\}$
S
$\{x <= 0 \wedge y >= 5\}$

$\{x >= 4\}$
S
$\{x <= 10 \wedge y >= 5\}$

$\{x >= 18 \wedge x <= 25\}$
S
$\{x <= 0 \wedge y >= x\}$

- **Teniendo en cuenta que la siguiente especificación es correcta:**

$\{x >= 4\}$
S
$\{x <= 0 \wedge y >= 5\}$

Ejemplos

- Demuestra la Corrección de las siguientes especificaciones:

$$\{x <= 0 \wedge y >= 10\}$$
S
$$\{x <= 0 \wedge y <= 4\}$$

$$\{x < 0 \wedge y > 10\}$$
S
$$\{x < 0 \wedge y < 4\}$$

$$\{x < -1 \wedge y >= 25\}$$
S
$$\{x <= -1\}$$

- **Teniendo en cuenta que la siguiente especificación es correcta:**

$$\{x <= 0 \wedge y >= 10\}$$
S
$$\{x < 0 \wedge y < 4\}$$

Anotaciones en Sentencias de Asignación

a←36

- Antes de cada sentencia se deben anotar las condiciones que sabemos que se cumplen antes de ejecutarlas.
- Detrás de cada asignación las que podemos demostrar:
  - Condiciones anteriores donde no intervienen las variables asignadas
  - Condición de que la variable tiene el valor asignado

Anotación de la poscondición

$\{A=5\}$

$x \leftarrow A$

____

$\{x<=0 \wedge z>0\}$

$x \leftarrow A$

____

Axioma de la asignación

$x \leftarrow E$

- Q_E^x : Aserto resultante de sustituir toda aparición de x por E

- **Axioma: La siguiente especificación es correcta**

$$\{Q_E^x\}$$
$$x \leftarrow E$$
$$\{Q\}$$

Aplicar el Axioma de la Asignación para demostrar las especificaciones de las siguientes instrucciones de asignación:

$\{\ \}$
$x \leftarrow x+1$
$\{x=7\}$

$\{\ \}$
$x \leftarrow x*2$
$\{x=8\}$

$\{\ \}$
$x \leftarrow x-2$
$\{x=2\}$

$\{\ \}$
$x \leftarrow x/2$
$\{x=5\}$

$\{\ \}$
$x \leftarrow x*2$
$\{x \geq 50\}$

$\{\ \}$
$x \leftarrow x+2$
$\{x \geq 3\}$

$\{\ \}$
$x \leftarrow x-2$
$\{x \geq 12\}$

Aplicar el Axioma de la Asignación para demostrar las especificaciones de las siguientes instrucciones de asignación:

$\{\ \}$
$x \leftarrow x*(-1)$
$\mathbf{\{x<0\}}$

$\{\ \}$
$x \leftarrow x+1$
$\mathbf{\{x+y>0\}}$

$\{\ \}$
$x \leftarrow x*y$
$\mathbf{\{x=2 \wedge k=0 \wedge y=2^k\}}$

Regla de Inferencia de la Asignación

Axioma de la asignación + 1ª Regla de Consecuencia

- Para que sea correcto:

$$\{P\}$$
$$x \leftarrow E$$
$$\{Q\}$$

1. **Se obtiene una especificación correcta** aplicando el axioma de la asignación:

$$\{Q_E^x\}$$
$$x \leftarrow E$$
$$\{Q\}$$

2. **Debe cumplirse que** $P \Rightarrow Q_E^x$ aplicando la 1ª regla de consecuencia

Ejercicios

$\{x=A \land y=B\}$

$x \leftarrow 2*x$

$\{x=2*A \land y=B\}$

Composición Secuencial

La especificación:

$\{P\}$

S_1

S_2

$\{Q\}$

Es correcta si encontramos un aserto R tal que se cumple que:

a) $\mathbf{\{P\}\ S_1\ \{R\}}$

b) $\mathbf{\{R\}\ S_2\ \{Q\}}$

Esquema de Selección: Selectiva simple

- La especificación:

  {P}

  si **B** entonces

  S

  fin_si

  {Q}

  Es correcta si satisface las condiciones de verificación:

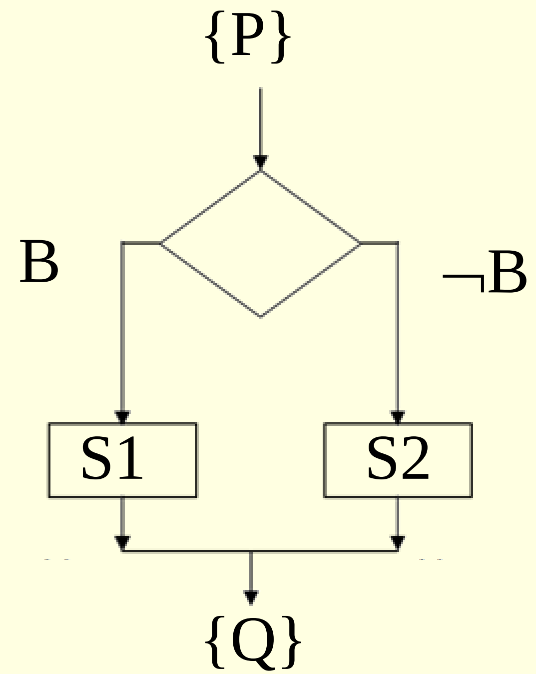
  a.1) $\mathbf{P \wedge \neg B \Rightarrow Q}$

  a.2) y las generadas por $\mathbf{\{P \wedge B\}\ S\ \{Q\}}$

Esquema de Selección: Selectiva doble

- La especificación:

  {P}
  si **B** entonces
  S1
  si_no
  S2
  fin_si
  {Q}



Es correcta si satisface las condiciones de verificación generadas por:

a.1) **{P ∧ B} S1 {Q}**

a.2) **{P ∧¬B} S2 {Q}**

```
{-2<x<2}
Si x>0 entonces
        x ← x-1
si_no
        x ← x+1
fin_si
{x>=0}
```

Esquema de Selección Múltiple

```
{P}
según sea B hacer
        B1: S1
        B2: S2
          .
          .
       Bn: Sn
fin_según
{Q}
```

a.1) La alternativa está bien construida, es decir:

a.1.1) $\mathbf{P \Rightarrow B1 \lor B2 \lor \ldots \lor Bn}$

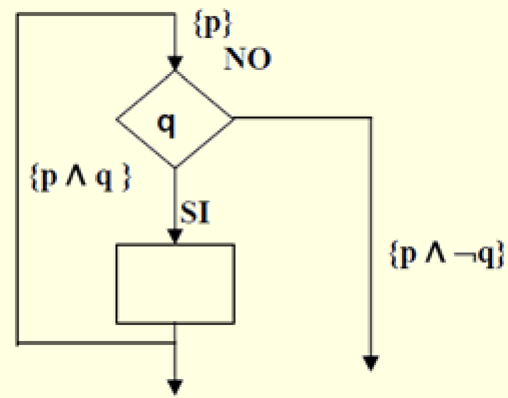
a.1.2) $\mathbf{P \Rightarrow \forall i,j: 1 \leq i, j \leq n \land i \neq j \Rightarrow \neg(Bi \land Bj)}$

a.2) satisface su especificación, o sea,

$\mathbf{\forall i: 1 \leq i \leq n: \{P \land Bi\}\ Si\ \{Q\}}$

Razonamiento sobre bucles

{P}
mientras **B** hacer {I}
S
fin_mientras
{Q}



- INVARIANTE: conjunto de condiciones lógicas que se cumplen antes, durante y después de la ejecución del bucle

La especificación será PARCIALMENTE CORRECTA si:

a.1) $\mathbf{P \Rightarrow I}$
a.2) $\mathbf{I \wedge \neg B \Rightarrow Q}$
a.3) $\mathbf{\{I \wedge B\}\ S\ \{I\}}$

Razonamiento sobre bucles

- Corrección Total
  - Encontrar una función monótona decreciente que garantice que el algoritmo acaba en un nº finito de pasos, alcanzando la cota 0:

    a.4) $\mathbf{I \wedge B \Rightarrow t > 0}$ en cada iteración la función cota se mantiene mayor que la cota

    a.5) $\mathbf{\{I \wedge B \wedge t=T\}\ S\ \{t < T\}}$ en cada iteración la función decrece

Invariante

- El invariante es un predicado que describe todos los estados por los que atraviesa el cómputo realizado por un bucle, observados justo antes de evaluar la condición de terminación.
- El invariante se satisface **antes de la primera iteración**, **después de cada una** de ellas y, **después de la última**.

Búsqueda de Invariantes

1. Elaborar una traza del algoritmo.
2. Debe ser parecido a la postcondición del bucle, ya que el Invariante debe cumplirse justo antes, durante y justo después del bucle.
3. En cada iteración el Invariante debe acercarse a la Postcondición del bucle, por tanto debe contener variables que se modifican dentro del bucle.
4. NUNCA usar expresiones del tipo x=x+b.

Búsqueda de Invariantes

1. Consultar y aprender de los ejercicios resueltos de clase, del material de la asignatura y de la bibliografía recomendada.
2. Comparar con invariantes propuestos por otros compañeros.
3. Resolver ejercicios.
4. Resolver más ejercicios.

Funciones Recursivas

```
tipo₁ función fun_rec (E tipo: x̄)
{P}
var
 (variables locales)
inicio
  si B entonces
         devolver sol(x̄)
  si_no
         devolver comb(fun_rec(suc(x̄)), x̄)
fin_si
{Q}
fin_función
```

```
entero función mult (E entero:a, E entero: b)
{a≥0 ∧ b ≥0}
inicio
  si a=0 entonces
         devolver 0
  si_no
         devolver b+mult(a-1,b)
fin_si
{devuelve v=a*b}
fin_función
```

Funciones Recursivas

- **Condiciones:** todos los casos están cubiertos en las condiciones:

  $P \Rightarrow B \vee \neg B$

- **Caso Base** el problema se resuelve correctamente:

  $P \wedge B \Rightarrow Q^{v}_{sol(\bar{x})}$

- **Caso No trivial:** argumentos deben verificar la precondición de la siguiente llamada

  $P \wedge \neg B \Rightarrow P^{\bar{x}}_{suc(\bar{x})}$

- **Caso No trivial** implica la corrección de la siguiente llamada a través de su postcondición

  $P \wedge \neg B \wedge (Q^{\bar{x}}_{suc(\bar{x})})^{v}_{v'} \Rightarrow Q^{v}_{comb(\bar{x},v')}$

- **Función limitadora:**

  $P \Rightarrow t(\bar{x}) \in N$

  $P \wedge \neg B \Rightarrow t(suc(\bar{x}))<t(\bar{x})$

Funciones Recursivas

- **Condiciones:** todos los casos están cubiertos en las condiciones:

  $P \Rightarrow B \vee \neg B$ $a \geq 0 \wedge b \geq 0 \Rightarrow a=0 \vee \neg(a=0)$

- **Caso Base** el problema se resuelve correctamente:

  $P \wedge B \Rightarrow Q^{v}_{sol(\overline{x})}$ $a \geq 0 \wedge b \geq 0 \wedge a=0 \Rightarrow a*b=0$

- **Caso No trivial:** argumentos deben verificar la precondición de la siguiente llamada

  $P \wedge \neg B \Rightarrow P^{\overline{x}}_{suc(\overline{x})}$ $a \geq 0 \wedge b \geq 0 \wedge a \neq 0 \Rightarrow a-1 \geq 0 \wedge b \geq 0$

- **Caso No trivial** implica la corrección de la siguiente llamada a través de su postcondición

  $P \wedge \neg B \wedge (Q^{\overline{x}}_{suc(\overline{x})})^{v}_{v'} \Rightarrow Q^{v}_{comb(\overline{x},v')}$

  $a \geq 0 \wedge b \geq 0 \wedge a \neq 0 \wedge v'=(a-1)*b \Rightarrow b+v'=a*b$

- **Función limitadora:** $t(x)=a$

  $P \Rightarrow t(\overline{x}) \in N$ $a \geq 0 \wedge b \geq 0 \Rightarrow a \geq 0$

  $P \wedge \neg B \Rightarrow t(suc(\overline{x}))<t(\overline{x})$ $a \geq 0 \wedge b \geq 0 \wedge a \neq 0 \Rightarrow a-1 < a$

Funciones Recursivas

```
tipo1 función fun_rec (E tipo: x̄ )
{P}
var
 (variables locales)
inicio
   si B entonces
          devolver sol( x̄ )
   si_no
          devolver comb(fun_rec(suc( x̄ )), x̄)
fin_si
{Q}
fin_función
```

```
entero función divide(E entero:a, E
    entero: b   )
{a≥0 ∧ b >0}
inicio
  si a<b entonces
         devolver 0
  si_no
         devolver 1+divide(a-b,b)
fin_si
{devuelve v=a/b}
fin_función
```

Funciones Recursivas

1. **Condiciones:** todos los casos están cubiertos en las condiciones:

   $P \Rightarrow B \vee \neg B$ **$a \geq 0 \wedge b > 0 \Rightarrow a<b \vee \neg(a<b)$**

2. **Caso Base** el problema se resuelve correctamente:

   $P \wedge B \Rightarrow Q^{v}{}_{sol(\bar{x})}$ **$a \geq 0 \wedge b > 0 \wedge a<b \Rightarrow 0=a/b$**

3. **Caso No trivial**

   $P \wedge \neg B \Rightarrow P^{\bar{x}}{}_{suc(\bar{x})}$ **$a \geq 0 \wedge b > 0 \wedge a \geq b \Rightarrow a-b \geq 0 \wedge b > 0$**

4. **Caso No trivial**

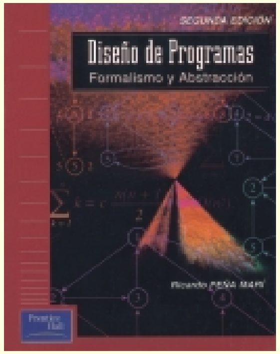
   $P \wedge \neg B \wedge (Q^{\bar{x}}{}_{suc(\bar{x})})^{v}{}_{v'} \Rightarrow Q^{v}{}_{comb(\bar{x},v')}$

   $a \geq 0 \wedge b > 0 \wedge a \geq b \wedge v'=(a-b)/b \Rightarrow 1+v'=a/b$

- **Función limitadora:** **$t(x)=a$**

5. $P \Rightarrow t(\bar{x}) \in N$ **$a \geq 0 \wedge b > 0 \Rightarrow a \geq 0$**
6. $P \wedge \neg B \Rightarrow t(suc(\bar{x}))<t(\bar{x})$ **$a \geq 0 \wedge b>0 \wedge a \geq b \Rightarrow a-b < a$**

Bibliografía

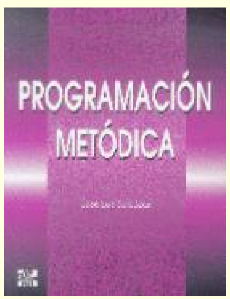


- Peña Marí, Ricardo; (1998) Diseño de Programas. Formalismo y Abstracción. Prentice Hall.

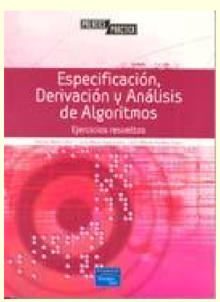
- Silva Ramírez, Esther L., López Coello, Manuel. (2010). Verificación formal de algoritmos: ejercicios resueltos. Servicio de Publicaciones UCA.
- Silva Ramírez, Esther L., López Coello, Manuel. (2010). Corrección de algoritmos complejos: verificación formal. Servicio de Publicaciones UCA.

Otras referencias

- Castro Rabal, Jorge; Cucker Farkas, Felipe (1993). Curso de programación. McGraw-Hill / Interamericana de España, S.A.



- Bálcazar José Luis (2001). Programación Metódica. McGraw-Hill.



- Martí Oliet Narciso (2006), Segura Díaz Clara M., Verdejo López José A. Especificación, derivación y análisis de algoritmos : ejercicios resueltos