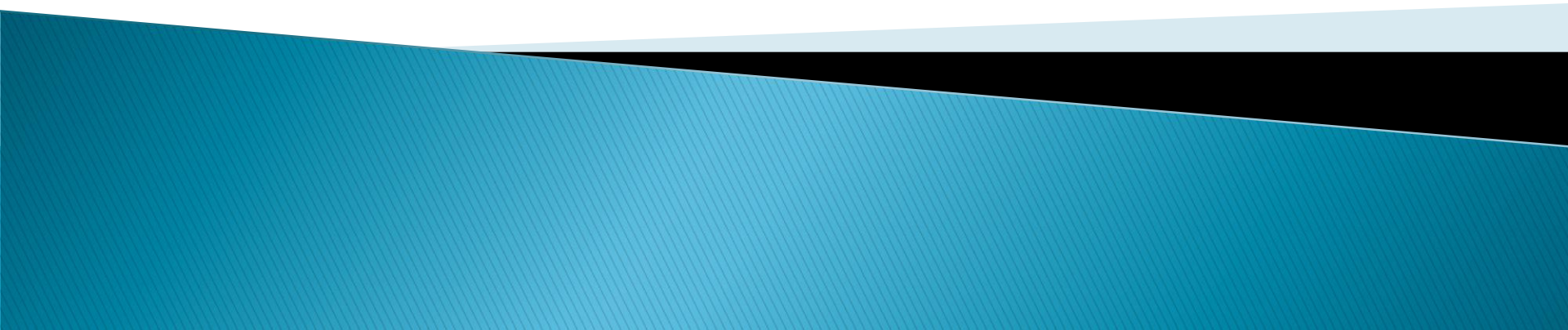


Programación Procedural

Verificación y Derivación

Unidad 1



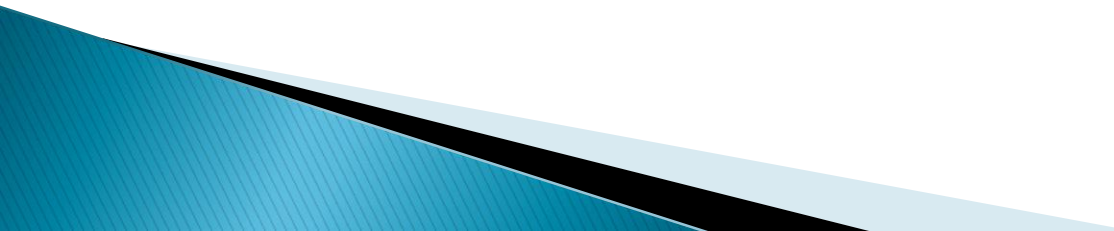
Enfoques para determinar si un programa es correcto

Testing

Para asegurar que un programa es correcto es necesario analizar todos los valores posibles de datos de entrada (imposible cuando la cantidad es infinita).

Verificación formal de programas:

Asegura que el programa es correcto con respecto a una especificación dada.



Verificación Formal

Para la verificación de programas Imperativos se utiliza un sistema de reglas basado en la **tripla de Hoare**

Tripla de Hoare

$$\{ P \} A \{ Q \}$$

A : programa/ sentencia / acción;

P y Q predicados o aserciones

Interpretación: Si el predicado P es cierto antes de la ejecución del programa A y el programa termina , entonces Q será cierto tras la ejecución de A.

- **Precondiciones:** son los asertos que deben cumplirse a la entrada de una tarea. Si la tarea o instrucción se realiza sin que la precondición se cumpla no tendremos garantía de los resultados obtenidos.
- **Postcondiciones** son los asertos acerca de los resultados que se esperan a la salida

Verificación Vs Derivación

Verificar: *demostrar que el programa construido es correcto* respecto de la especificación dada

Derivar: un programa permite *construir un programa* a partir de su especificación, de forma que se obtiene un algoritmo correcto por construcción

Corrección Total y Corrección Parcial

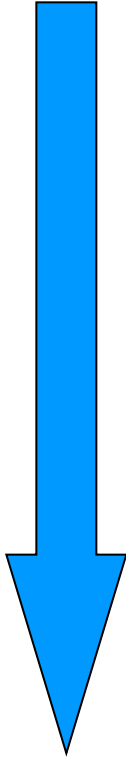
Corrección parcial: $\{P\} A \{Q\}$ es parcialmente correcto si el estado final de A, satisface $\{Q\}$ siempre que el estado inicial satisface $\{P\}$.

Corrección total: Se da cuando un código además de ser correcto parcialmente, termina.

Los algoritmos sin bucles siempre terminan, la corrección parcial implica la corrección total. Esta distinción es esencial sólo en el caso de códigos que incluyan bucles o recursiones

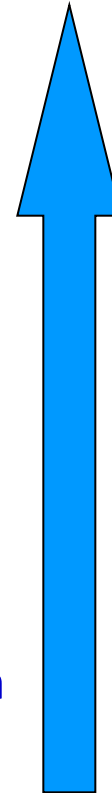
Para la especificación $\{P\} A1; A2; A3; \dots; An \{Q\}$ El código se verifica **en sentido contrario a como se ejecuta**. Se cumple:

E
j
e
c
u
c
i
ó
n



$\{P\}$ Precond. A1
A1;
Poscond. A1= Precond. A2
A2;
Poscond. A2= Precond. A3
A3;
...;
Poscond. An -1= Precond.An
An;
 $\{Q\}$ Poscond. An

V
e
r
i
f
i
c
a
c
i
ó
n



Iteración

La instrucción de **iteración** corresponde a la siguiente sintaxis en pseudocódigo:

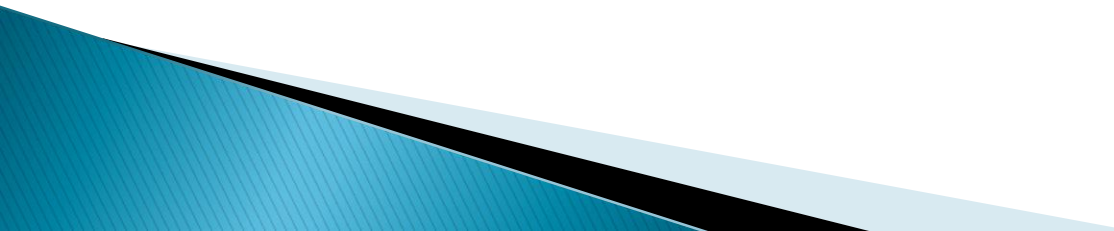
Mientras (B)

A

finMientras

B: expresión booleana, es una condición tal que si B es falsa el ciclo termina, caso contrario se ejecuta la instrucción A o cuerpo del ciclo, y se vuelve a evaluar la expresión B, repitiendo el proceso hasta que el ciclo termina.

A: es una instrucción (simple o compuesta)



Verificación de una iteración: uso del Invariante

Para verificar un bucle se necesita un predicado llamado **invariante** que describa los distintos estados por los que pasa el bucle, estableciendo la relación que existe entre las variables que participan en él.

{P}

{I}

invariante

Mientras(B)

A

//cuerpo del ciclo

finmientras

{Q}

El **invariante** es un predicado lógico que tiene la propiedad de **ser cierto (aserto)** antes de entrar al bucle, tras cada iteración (durante el bucle) y después del bucle.

Verificación de una iteración

PASOS PARA LA VERIFICACION FORMAL DE UNA ITERACIÓN

- (1) $\{P\} A_0 \{I\}$ El invariante se satisface antes de la primer iteración
- (2) $\{I \wedge B\} A \{I\}$ El invariante se mantiene al ejecutar el cuerpo A del bucle
- (3) $\{I \wedge \sim B\} \Rightarrow Q$ El invariante se cumple al salir del bucle (B es falsa) y debe llevarnos a la postcondición
- 4) **Probar que el algoritmo termina:** se debe encontrar una **función de cota C**, entera, construida a partir de una expresión que incluya algunas o todas las variables que intervienen en la condición del bucle y que son modificadas en el cuerpo del bucle, de modo que cumpla:
 - (4-a) $\{I \wedge B\} \Rightarrow C \geq 0$ La cota es ≥ 0 cuando se cumple condición B
 - (4-b) $\{I \wedge B \wedge C=T\} A \{C < T\}$ La cota decrece al ejecutar el cuerpo del bucle

La función cota C debe dar indicio del número de iteraciones que quedan por realizar en una determinada iteración (por eso se llama cota)

Verificación de una iteración

Sintetizando, la regla para la verificación de una instrucción iterativa es:

$$(1) \quad \{P\} A_0 \{ I \}$$

$$(2) \quad \{I \wedge B\} A \{I\}$$

$$(3) \quad \{I \wedge \sim B\} \Rightarrow Q$$

$$(4-a) \quad \{ I \wedge B \} \Rightarrow C \geq 0$$

$$(4-b) \quad \{ I \wedge B \wedge C=T \} A \{ C < T \}$$

$$\{P\} \text{ Mientras } (B) \quad A \quad \text{ FinMientras } \{Q\}$$

Repaso

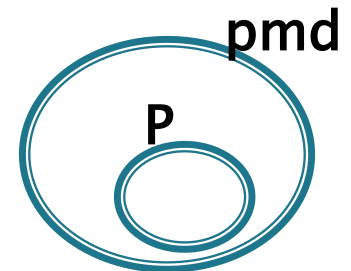
Verificación de una acción

Para verificar una acción es necesario la existencia de una Tripla de Hoare $\{P\} a \text{ (variable = Expresión)} \{Q\}$

Luego se calcula la $\text{pmd} = (Q)_{\text{variable}}^{\text{Expresión}}$

Por ultimo se debe probar la implicación: $P \Rightarrow \text{pmd}$

Si se prueba la implicación, significa que la acción es correcta.



Razonamiento

- En los pasos de la verificación del bucle, se incorpora la invariante para construir triplas de Hoare.
- En los pasos que hay triplas se debe calcular la pmd y luego probar la implicación.
- En los pasos sin tripla, solo se analiza la implicación.

$$(1) \quad \{P\} A_0 \{ I \}$$

$$(2) \quad \{I \wedge B\} A \{I\}$$

$$(3) \quad \{I \wedge \sim B\} \Rightarrow Q$$

$$(4-a) \quad \{ I \wedge B \} \Rightarrow C \geq 0$$

$$(4-b) \quad \{ I \wedge B \wedge C=T \} A \{ C < T \}$$

Ejemplo de Verificación

Verificar que el siguiente algoritmo, que calcula la suma de los elementos de un arreglo con un **recorrido de derecha a izquierda**, cumple la especificación para el siguiente invariante.

$$I = \{x = (\sum i: n \leq i < N: V[i]) \wedge (0 \leq n \leq N)\}$$

Constante N

entero V [N], x, n

{P: N ≥ 0}

n = N

x = 0

Mientras (n ≠ 0)

 x = x + V[n-1]

 n = n - 1

finMientras

{Q : x = (∑ i: 0 ≤ i < N : V[i])}

I establece relación entre las variables del bucle y además describe los distintos estados por los que pasa el bucle

Ejemplo de Verificación – Paso 1

- *El invariante se satisface antes de la primer iteración $\{P\} A_0 \{I\}$*

Esto es probar que la terna $\{P\} n=N; x=0 \{I\}$ es correcta.

$$\{P\} \Rightarrow \text{pmd} (n=N; x=0, I)$$

$$\text{pmd} \equiv \left((I)_x^0 \right)_n^N \equiv ((0 \leq n \leq N) \wedge x = (\sum_{i: n \leq i < N} V[i])_x^0)_n^N$$

$$(I)_x^0 \equiv ((0 \leq n \leq N \wedge 0 = (\sum_{i: n \leq i < N} V[i]))$$

$$(I)_n^N \equiv ((0 \leq N \leq N) \wedge 0 = (\sum_{i: N \leq i < N} V[i])$$



$\{P: N \geq 0\}$
 $n=N$
 $x=0$
Mientras ($n \neq 0$)

$$I = \{x = (\sum_{i: n \leq i < N} V[i]) \wedge (0 \leq n \leq N) \}$$

Por rango vacío, la sumatoria es 0 y el segundo término es True

$$\text{pmd} = \{N \geq 0\} \equiv \{P\}$$

Dado que $\{P\} \Rightarrow \text{pmd}$ las inicializaciones $n=N$ y $x=0$ son correctas

Ejemplo de Verificación – Paso 2

- *El invariante se mantiene al ejecutar el cuerpo A del bucle* $\{I \wedge B\} A \{I\}$

Para demostrar que la terna $\{I \wedge B\} A \{I\}$ es correcta, se debe probar que $\{I \wedge B\} \Rightarrow \text{pmd} (A, I)$

$\text{pmd}(x=x + V[n-1] , n=n-1, I)$

$$\equiv ((0 \leq n \leq N) \wedge (x = (\sum i: n \leq i < N: V[i])) \wedge x^{x+v[n-1]})$$

$$\equiv ((0 \leq n-1 \leq N) \wedge (x = (\sum i: n-1 \leq i < N: V[i])) \wedge x^{x+v[n-1]})$$

$$\equiv ((0 \leq n-1 \leq N) \wedge (x + V[n-1] = (\sum i: n-1 \leq i < N: V[i])))$$

Particion de rango

$$\equiv (0 \leq n-1 \leq N) \wedge (x + V[n-1] = \mathbf{V[n-1]} + (\sum i: n \leq i < N: V[i])) \quad (1)$$

Hemos descompuesto la sumatoria separando un elemento del arreglo.

Ahora debemos probar que $I \wedge B$ es más fuerte que la aserción (1), es decir

$$I \wedge B \Rightarrow (1)$$



Mientras ($n \neq 0$)
 $x = x + V[n-1]$
 $n = n - 1$
finMientras

$$I = \{x = (\sum i: n \leq i < N: V[i]) \wedge (0 \leq n \leq N)\}$$

Ejemplo de Verificación – Paso 2

$$I = \{x = (\sum i: n \leq i < N: V[i]) \wedge (0 \leq n \leq N)\}$$

$$I \wedge B \Rightarrow (1)$$

$$((0 \leq n-1 \leq N) \wedge (x + V[n-1] = V[n-1] + (\sum i: n \leq i < N: V[i]))) \quad (1)$$

Recordemos que $I = ((0 \leq n \leq N) \wedge (x = \langle \sum i: n \leq i < N: V[i] \rangle))$ y $B = n \neq 0$

$$a) (0 \leq n \leq N) \wedge (n \neq 0) \equiv (0 < n \leq N) \equiv (0 \leq n-1 < N) \Rightarrow (0 \leq n-1 \leq N)$$

$$b) x = \langle \sum i: n \leq i < N: V[i] \rangle \equiv x + V[n-1] = V[n-1] + (\sum i: n \leq i < N: V[i])$$

$$\text{entonces} \quad I \wedge B \Rightarrow (0 \leq n-1 \leq N) \wedge (x + V[n-1] = V[n-1] + (\sum i: n \leq i < N: V[i])) \quad (1)$$

Por tanto las asignaciones $x = x + V[n-1]$ y $n = n-1$ son correctas

Ejemplo de Verificación – Paso 3

▪ *El invariante se cumple al salir del bucle (B es falsa) y debe llevarnos a la postcondición $I \wedge \sim B \Rightarrow Q$.*

$$\{I \wedge \sim B\} \equiv I \wedge \sim(n \neq 0) \equiv (0 \leq n \leq N) \wedge (x = (\sum i: n \leq i < N: V[i])) \wedge (n = 0)$$

$$\equiv (N \geq 0) \wedge (x = (\sum i: 0 \leq i < N: V[i])) \Rightarrow x = (\sum i: 0 \leq i < N: V[i]) \equiv Q$$



$$I = \{x = (\sum i: n \leq i < N: V[i]) \wedge (0 \leq n \leq N)\}$$

$$\{Q : x = (\sum i: 0 \leq i < N : V[i])\}$$

Ejemplo de Verificación – Paso 4

► Definamos la función cota

La función cota, debe definirse en función de las variables que aparecen en la condición B, como en B la única variable que aparece es n, entonces podemos tomar:

$$C(n)=n; n \text{ natural } n \geq 0$$



n=N
Mientras (n ≠ 0)
n=n-1

$$I = \{x = (\sum i: n \leq i < N: V[i]) \wedge (0 \leq n \leq N) \}$$

Debemos probar

- La función cota es mayor o igual que 0 cuando se cumple condición B
- La función cota decrece al ejecutar el cuerpo A del bucle

a) La función cota es mayor o igual que 0 cuando se cumple condición B

$$\text{Esto es: } I \wedge B \Rightarrow C \geq 0$$

$$\text{Debemos probar } I \wedge n \neq 0 \Rightarrow n \geq 0$$

$$(I \wedge n \neq 0) \equiv (0 \leq n \leq N) \wedge n \neq 0 \equiv n > 0 \Rightarrow n \geq 0$$

Ejemplo de Verificación – Paso 4

► La función cota decrece al ejecutar el cuerpo A del bucle

Esto es probar que es correcta la siguiente especificación:

$$\{ I \wedge B \wedge C=T \} A \{ C < T \}$$

Para probar que esta terna es correcta se debe demostrar que

$$\{ I \wedge B \wedge C=T \} \Rightarrow \text{pmd}(A, n < T)$$

$$\text{pmd}(A, n < T) \equiv \text{pmd}(x=x+v[n-1] ; n=n-1, n < T)$$

$$\left((n < T)_n^{n-1} \right)_x^{x+v[n-1]} \equiv (n-1 < T)_x^{x+v[n-1]}$$

$$\equiv n-1 < T \quad (1)$$



Mientras (n ≠ 0)
x=x+V[n-1]
n=n-1
finMientras

$$\text{Se debe probar } (I \wedge n \neq 0) \wedge (n=T) \Rightarrow n=T \equiv n-1 < T \quad (1)$$

Hemos verificado que este algoritmo que suma los elementos de un arreglo de N enteros, a partir del último elemento, cumple con la especificación dada, es decir es correcto para ella.

Derivación de algoritmos

Es un proceso que permite **construir las instrucciones a partir de la especificación** preocupándose a lo largo de este proceso de su corrección.

Al terminar la derivación **el algoritmo encontrado será correcto** (cumple su especificación) por construcción.

En la especificación, la postcondición describe el estado que se desea alcanzar, de ahí que el **proceso de construcción está guiado por la postcondición**.

Derivación de algoritmos

A la hora de derivar un bucle, se debe encontrar:

{P}

A0

?

condiciones iniciales

{I, C}

?

I es el invariante y C es la cota

Mientras (B)?

A1

?

Instrucción del cuerpo del bucle que se encarga de mantener cierto el invariante

{R}

precondición más débil de A2 respecto de I, $\text{pmd}(A2, I)$

A2 ?

representa las instrucciones que hacen que las variables que intervienen en la expresión B avancen hacia la condición de salida del bucle ($\neg B$) (hace que decrezca la función cota)

finMientras

{Q}

Derivación de algoritmos

$\{P\}$
A0
Mientras (B)

A1

A2

finMientras
 $\{Q\}$

$$(1) \quad \{P\} A_0 \{ I \}$$

$$(2) \quad \{ I \wedge B \} A \{ I \}$$

$$(3) \quad \{ I \wedge \sim B \} \Rightarrow Q$$

$$(4-a) \quad \{ I \wedge B \} \Rightarrow C \geq 0$$

$$(4-b) \quad \{ I \wedge B \wedge C = T \} A \{ C < T \}$$

$$\{P\} \text{ Mientras (B) } A \text{ FinMientras } \{Q\}$$

Derivación de algoritmos

1. Obtener el invariante **I** y la condición del bucle **B** a partir de la postcondición **Q**, teniendo en cuenta que se debe satisfacer

$$\{I \wedge \neg B\} \Rightarrow Q$$

2. Inicializar las variables (Diseñar A_0) de tal manera que el invariante sea cierto al comienzo del bucle y que cumpla que

$$\{P\} A_0 \{I\} \text{ *Garantiza el cumplimiento del invariante al comenzar el ciclo*}$$

3. Obtener el cuerpo del bucle **A**

$$\{I \wedge B\} A \{I\}$$

4. Diseñar la función cota **C** de la manera que

$$\{I \wedge B\} \Rightarrow C \geq 0$$

5. Asegurarse que la función cota decrezca (*garantiza la finalización del ciclo*)

$$\{I \wedge B \wedge C = T\} A \{C < T\}$$

Derivación de algoritmos - Pasos

- ▶ **Obtener el invariante I y la condición del bucle sabiendo que**

$$I \wedge \neg B \Rightarrow Q$$

Existen 2 posibilidades:

- a) Si la **postcondición está en forma conjuntiva**, se elige una parte como invariante y el resto como negación de la condición del bucle.
- b) Si la postcondición **no está en forma conjuntiva**, se puede introducir una nueva variable que sustituya en la postcondición a una constante. En este caso el invariante será la postcondición generalizada con esa nueva variable y la condición del bucle será la negación de la igualdad entre la variable y la constante sustituida.

Derivación de algoritmos - Pasos

- ▶ b) Si la postcondición **no está en forma conjuntiva**, se puede **introducir una nueva variable que sustituya en la postcondición a una constante**. En este caso el invariante será la postcondición generalizada con esa nueva variable y la condición del bucle será la negación de la igualdad entre la variable y la constante sustituida.

$$I = \{x = (\sum i: 0 \leq i < N: V[i]) \wedge (0 \leq n \leq N)\}$$

Constante N

entero V [N], x, n

{P: $N \geq 0$ }

n=N

x=0

Mientras (n ≠ 0)

 x=x+V[n-1]

 n=n-1

finMientras

{Q : $x = (\sum i: 0 \leq i < N : V[i])$ }

Derivación de algoritmos - Pasos

Para lograr $\{I \wedge B\} A \{I\}$ se sugiere:

- a. Diseñar la instrucción **A2** que avance hacia la condición de salida del bucle, es decir que decrezca el valor de la función cota.

*Construir un aserto o **condición intermedia** R que haga correcta la especificación $\{R\} A2 \{I\}$,*

- b. Diseñar A_1 , de manera que se cumpla $\{I \wedge B\} A_1 \{R\}$

A_1 representa las instrucciones del cuerpo que se encargan de mantener cierto el invariante .



$$\begin{array}{l} \{I \wedge B\} A_1 \{R\} \\ \{R\} A2 \{I\} \\ \hline \{I \wedge B\} A1, A2 \{I\} \end{array}$$

Razonamiento

- EL razonamiento usado en la Verificación se aplica acá, donde hay triplas de Hoare se calcula pmd y luego se prueba la implicación, y sino solo se hace el último paso.
- Donde hay triplas, se desconoce la acción por lo cual, de manera genérica toma la siguiente forma:

$$\{P\} \text{Variable} = \text{Expresión} \{I\}$$

la variable se conoce pero la expresión no.

- Esta expresión es una incógnita que se introduce en el calculo de la pmd:
$$\text{pmd} = (I)$$
- Esta incógnita se resuelve aplicando propiedades matemáticas.

Razonamiento

A partir de Q se construye el invariante, se identifica la guarda B y la función Cota.

{P}

A0

condiciones iniciales. Se calcula pmd con incógnita

Mientras (B)

A1

Instrucción del cuerpo del bucle que se encarga de mantener cierto el invariante Se calcula pmd con incógnita

A2

representa las instrucciones que hacen que las variables que intervienen en la expresión B avancen hacia la condición de salida del bucle ($\neg B$) (hace que decrezca la función cota). Al construir la Invariante se identifica A2

finMientras

{Q}

Razonamiento

1. Obtener el invariante **I** y la condición del bucle **B** a partir de la postcondición **Q**, teniendo en cuenta que se debe satisfacer

$$\{I \wedge \neg B\} \Rightarrow Q$$

En este paso se identifica la acción **A2**

2. Inicializar las variables (Diseñar **A0**) de tal manera que el invariante sea cierto al comienzo del bucle y que cumpla que

$$\{P\} A_0 \{I\} \text{ Garantiza el cumplimiento del invariante al comenzar el ciclo}$$

Acá se calcula la pmd, se introduce la incógnita. Una vez encontrado el valor de inicialización se debe probar la implicación para asegurar que la acción es correcta.

3. Obtener el cuerpo del bucle **A**

$$\{I \wedge B\} A1 \{I\}$$

Se calcula la pmd, se introduce la incógnita. Una vez identificada la expresión, se debe probar la implicación para asegurar que la acción es correcta.

4. Diseñar la función cota **C** de la manera que

$$\{I \wedge B\} \Rightarrow C \geq 0 \text{ Se razona la implicación}$$

5. Asegurarse que la función cota decrezca (*garantiza la finalización del ciclo*)

$$\{I \wedge B \wedge C = T\} A2 \{C < T\} \text{ se calcula la pmd y luego se prueba la implicación}$$

Ejemplo de Derivación

Derivar un programa que calcule la suma de los elementos de un arreglo que contiene N componentes enteras , recorriendo el arreglo de derecha a izquierda

Especificación

const int N

entero x

entero a[0..N)

{P: $N \geq 0$ }

A

{Q: $x = (\sum i: 0 \leq i < N: a[i])$ }

Ejemplo de Derivación – Paso 1

► Obtener invariante y guarda

Como la postcondición **no está en forma conjuntiva**, se utiliza la **sustitución de una constante por una variable**. En la postcondición aparecen las constantes N y 0. Por tanto, utilizando la técnica de reemplazo de constantes por variables

$$\{Q: x = (\sum i: 0 \leq i < N: a[i])\}$$

Se puede introducir una nueva variable que sustituya en la postcondición a una constante. En este caso el invariante será la postcondición generalizada con esa nueva variable y la condición del bucle será la negación de la igualdad entre la variable y la constante sustituida.

Como el recorrido es de derecha a izquierda se reemplaza el cte 0 por la variable n

$$\{I : x = \langle \sum i: n \leq i < N: a[i] \rangle\}$$

$$B : \sim (n=0) \equiv n \neq 0$$

Es necesario reforzar el invariante agregando el rango de variación de la variable n

$$\{I : x = (\sum i: n \leq i < N: a[i]) \wedge (0 \leq n \leq N)\}$$

la guarda es entonces $B: n \neq 0$

{P}

A0

Mientras ($n \neq 0$)

A1

A2

finMientras

{Q}

Ejemplo de Derivación – Paso 2

- *Inicializar las variables, es decir diseñar A_0 de modo que el invariante sea cierto; esto es $\{P\} A_0 \{I\}$*

$$\{I : \mathbf{x} = \sum_{i: \mathbf{n} \leq i < N: a[i]} \wedge (0 \leq \mathbf{n} \leq N)\}$$

Se debe inicializar \mathbf{x} y \mathbf{n} , variables que aparecen en el invariante. Se debe elegir expresiones enteras \mathbf{E} y \mathbf{F} tales que se garantice el cumplimiento del invariante, esto es encontrar la precondition más débil.

$$\text{pmd } (\mathbf{x} = \mathbf{F}, \mathbf{n} = \mathbf{E}, I) \equiv (\mathbf{F} = \sum_{i: \mathbf{E} \leq i < N: a[i]} \wedge (0 \leq \mathbf{E} \leq N))$$

$$\{P: N \geq 0\}$$

como $(0 \leq \mathbf{E} \leq N)$ y se debe recorrer el arreglo de derecha a izq.

elegimos $\mathbf{E} = \mathbf{N}$, esto es $\mathbf{n} = \mathbf{N}$



$$\text{Si } \mathbf{E} = \mathbf{N}; \mathbf{F} = \sum_{i: N \leq i < N: a[i]} \wedge (0 \leq \mathbf{E} \leq N)$$

Por rango vacío, el elemento neutro de la suma es 0

$$\mathbf{F} = 0 ; \mathbf{x} = 0$$

$$\text{pmd } (\mathbf{n} = \mathbf{E}, \mathbf{x} = \mathbf{F}, I) \equiv (\text{true}) \wedge (N \geq 0) \equiv N \geq 0 \quad (1)$$

Para que la inicialización $\mathbf{n} = \mathbf{N}, \mathbf{x} = 0$ sea correcta,

$$\text{se debe probar } \{P\} \Rightarrow N \geq 0 \quad (1)$$

Como $\{P: N \geq 0\} \equiv (1)$ la inicialización es correcta

$\{P\}$

$\mathbf{n} = \mathbf{N},$

$\mathbf{x} = 0$

Mientras $(\mathbf{n} \neq 0)$

$A1$

$A2$

finMientras

$\{Q\}$

Ejemplo de Derivación – Paso 3

► Diseñar el cuerpo del ciclo

$$\{ I: x = (\sum_{i: n \leq i < N: a[i]}) \wedge (0 \leq n \leq N) \}$$

El cuerpo consistirá de una instrucción que permita avanzar hacia $\neg B$ para garantizar la finalización del bucle. Como n se inicializa en N , deberá decrementarse dentro del bucle: **$n = n - 1$** .

Ahora deberá encontrarse la instrucción que asegure el cumplimiento del invariante dentro del bucle, se debe encontrar el valor que tomará x dentro del bucle

$$I \wedge B \Rightarrow \text{pmd} (x = E, n = n - 1, I)$$

$$\text{pmd} (x = E, n = n - 1, I)$$

$$\text{pmd} \equiv (x = (\sum_{i: n \leq i < N: a[i]}) \wedge (0 \leq n \leq N))_{n^{n-1}}^E$$

$$E = (\sum_{i: n-1 \leq i < N: a[i]}) \wedge (0 \leq n - 1 \leq N)$$

por partición de rango

$$E = (\sum_{i: n \leq i < N: a[i]}) + a[n-1] \wedge (0 \leq n-1 \leq N)$$

$$x = x + a[n-1] \wedge (0 \leq n-1 \leq N) \quad (1)$$

$$\text{Se debe probar que } \{ I \wedge B \} \Rightarrow x = x + a[n-1] \wedge (0 \leq n-1 \leq N) \quad (1)$$



$$\frac{\{ I \wedge B \} A_1 \{ R \} \quad \{ R \} A_2 \{ I \}}{\{ I \wedge B \} A_1, A_2 \{ I \}}$$

Ejemplo de Derivación – Paso 3



B: $n \neq 0$

Se debe probar que

$$\{I \text{ y } B\} \Rightarrow x = x + a[n-1] \wedge (0 \leq n-1 \leq N) \quad (1)$$

$$\{I \text{ y } B\} \equiv (x = (\sum i: n \leq i < N: a[i]) \wedge (0 \leq n \leq N)) \wedge (n \neq 0) \quad (2) \quad (3)$$

$$x \Rightarrow x + a[n-1] \quad (2) \Rightarrow (1)$$

$$\text{De (3): } (0 < n \leq N) \equiv (0 \leq n-1 < N) \Rightarrow (0 \leq n-1 \leq N) \quad (3) \Rightarrow (1)$$

$$\text{Como } \{I \text{ y } B\} \Rightarrow (1)$$

$x = x + a[n-1]$; $n = n-1$ son asignaciones corretas

{P}

n = N

x = 0

Mientras (**n ≠ 0**)

x = x + a[n-1]

n = n-1

finMientras

{Q}

Ejemplo de Derivación – Paso 4-a



► **Diseñar la función cota tal que $I \wedge B \Rightarrow C \geq 0$**

B: $n \neq 0$

$\{ I: x = (\sum i: n \leq i < N: a[i]) \wedge (0 \leq n \leq N) \}$

La cota debe definirse teniendo en cuenta las variables que intervienen en la guarda de tal manera de asegurar su decrecimiento con cada ejecución del bucle.

B: $n \neq 0$

como n es la única variable que aparece en el bucle y debe decrecer dentro de él dado que su valor inicial es N , elegimos como cota **$C(n) = n$**

Esto es porque se recorre el arreglo desde la última posición hasta la primera. Si se lo recorre desde el principio al fin, la función Cota sería: $C = N - n$

Se debe probar que esta función se mantiene no negativa durante la ejecución del bucle:

$$\begin{aligned} I \wedge B &\equiv (x = \langle \sum i: n \leq i < N: a[i] \rangle) \wedge (0 \leq n \leq N) \wedge (n \neq 0) \\ &\equiv (x = \langle \sum i: n \leq i < N: a[i] \rangle) \wedge (0 < n \leq N) \\ &\Rightarrow (n > 0) \Rightarrow (n \geq 0) \end{aligned}$$

Ejemplo de Derivación – Paso 4-b

- Probar que el algoritmo termina es decir la cota decrece dentro del ciclo
 $\{I \wedge B \wedge C=T\} A \{C < T\}$

$$C(n) = n$$

$$\{I \wedge B \wedge C=T\} \Rightarrow \text{pmd}(A, n < T) \equiv (x = x + a[n], n = n - 1, n < T)$$

$$\equiv ((n < T) \wedge n^{n-1}) \wedge x^{x+a[n]}$$

$$\text{pmd} \equiv (n-1) < T \quad (1)$$

$$\{I \wedge B \wedge C=T\} \Rightarrow \{C=T\} \equiv n=T \Rightarrow (n-1) < T \quad (1)$$