Precondición más débil

Algoritmos y Estructuras de Datos I

Repaso: Corrección de un programa

- **Definición.** Decimos que un programa S es correcto respecto de una especificación dada por una precondición P y una postcondición Q, si siempre que el programa comienza en un estado que cumple P,
  - el programa termina su ejecución,
  - y en el estado final se cumple Q.
- **Notación.** Cuando S es correcto respecto de la especificación (P, Q), lo denotamos con la siguiente tripla de Hoare:

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

Demostrando que un programa es correcto

- Sabemos razonar sobre la corrección de nuestros programas, anotando el código con predicados que representan los estados.
- Nos interesa formalizar estos razonamientos, para estar seguros de que no cometimos errores en la demostración.
- Una forma de conseguirlo es la siguiente: A partir de la tripla de Hoare $\{P\}\ S\ \{Q\}$, obtener una fórmula lógica α tal que

  α es verdadera si y sólo si $\{P\}\ S\ \{Q\}$ es verdadera.

- Entre otras cosas, esto nos permite automatizar la demostración con un verificador automático (!)

Un lenguaje imperativo simplificado

- Para facilitar nuestro trabajo, definamos un lenguaje imperativo más sencillo que C++ basado al que llamaremos SmallLang. [1]
- SmallLang únicamente soporta las siguientes instrucciones:
  1. Nada: Instrucción **skip** que no hace nada.
  2. Asignación: Instrucción **x := E**.
- Además, soporta 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.

[1] *The Semantics of a Small Language* de David Gries

Demostraciones de corrección

- Buscamos un mecanismo para demostrar “automáticamente” la corrección de un programa respecto de una especificación (es decir, la validez de una tripla de Hoare).
- ¿Es válida esta tripla?

$$\{x \geq 4\}$$
$$\mathbf{x := x + 1}$$
$$\{x \geq 7\}$$

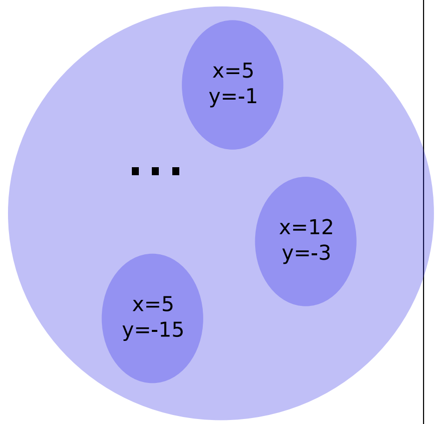
- No. Contrajemplo: con $x = 4$ no se cumple la postcondición.
- ¿Es válida esta tripla?

$$\{x \geq 4\}$$
$$\mathbf{x := x + 1}$$
$$\{x \geq 5\}$$

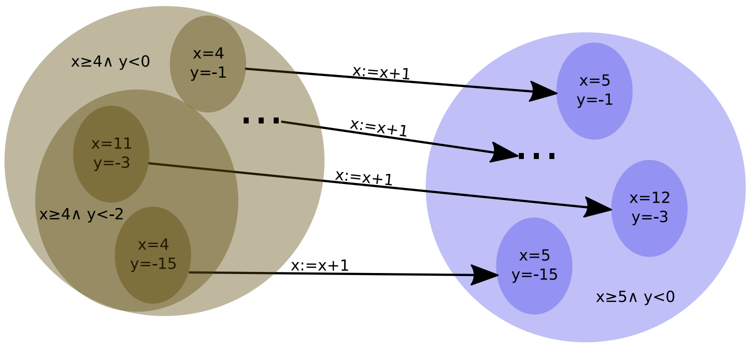
- Sí. Es válida!

La precondición más débil

$$\{\ x \geq 4 \land y < -2\}$$
```
x := x + 1
```
$$\{\ x \geq 5 \land y < 0\}$$



La precondición más débil



- Supongamos que tenemos un predicado que captura la precondición más débil del programa S con la postcondición Q (**Notación:** $wp(\mathbf{S}, Q)$)
- ¿Qué formula podemos usar para probar que la tripla de Hoare es válida?

$$(x \geq 4 \land y < -2) \Rightarrow_L wp(x := x + 1, x \geq 5 \land y < 0)$$

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.** $wp(\mathbf{S}, Q)$.
- **Teorema:** Una tripla de Hoare $\{P\}\mathbf{S}\{Q\}$ es válida si y sólo si:

$$P \Rightarrow_L wp(S, Q)$$

Precondición más débil

- Ejemplo:

$$\{wp(\mathbf{x := x{+}1}, Q)\}$$
$$\mathbf{x := x + 1}$$
$$\{Q : x \geq 7\}$$

- ¿Cuál es la precondición más débil de `x:=x+1` con respecto a la postcondición $x \geq 7$?
- $wp(\mathbf{x := x{+}1}, Q)) \equiv \ x \geq 6$.

Precondición más débil

- Otro ejemplo:

$$\{wp(\mathbf{S2}, Q)\}$$
$$\mathbf{S2: x := 2 * |x| + 1}$$
$$\{Q : x \geq 5\}$$

- $wp(\mathbf{S2}, Q) \ \equiv \ x \geq 2 \ \vee \ x \leq -2$.
- Otro más:

$$\{wp(\mathbf{S3}, Q)\}$$
$$\mathbf{S3: x := y{*}y}$$
$$\{Q : x \geq 0\}$$

- $wp(\mathbf{S3}, Q) \ \equiv \ \mathit{True}$.

Precondición más débil

- Si para demostrar la validez de $\{P\}\mathbf{S}\{Q\}$ nos alcanza con probar la fórmula:

$$P \Rightarrow_L wp(S, Q)$$

- Entonces lo que necesitamos un mecanismo para obtener la *wp* de (S,Q).
- Afortunadamente, existe un conjunto de axiomas que podemos usar para obtener la *wp*
- Antes de empezar a ver estos axiomas, definamos primero dos predicados: $def(E)$ y Q^x_E

Predicado def(E)

- **Definición.** Dada una expresión E, llamamos $\mathrm{def}(E)$ a las condiciones necesarias para que E esté definida.

  1. $\mathrm{def}(x + y) \ \equiv \ \mathrm{def}(x) \wedge \mathrm{def}(y)$.
  2. $\mathrm{def}(x/y) \ \equiv \ \mathrm{def}(x) \wedge (\mathrm{def}(y) \wedge_L y \neq 0)$.
  3. $\mathrm{def}(\sqrt{x}) \ \equiv \ \mathrm{def}(x) \wedge_L x \geq 0$.
  4. $\mathrm{def}(a[i] + 3) \ \equiv \ (\mathrm{def}(a) \wedge \mathrm{def}(i)) \wedge_L 0 \leq i < |a|$.

- Suponemos $\mathrm{def}(x) \equiv \mathit{True}$ para todas las variables, para simplificar la notación.
- Con esta hipótesis extra:

  1. $\mathrm{def}(x + y) \ \equiv \ \mathit{True}$.
  2. $\mathrm{def}(x/y) \ \equiv \ y \neq 0$.
  3. $\mathrm{def}(\sqrt{x}) \ \equiv \ x \geq 0$.
  4. $\mathrm{def}(a[i] + 3) \ \equiv \ 0 \leq i < |a|$.

Predicado Q^x_E

- **Definición.** Dado un predicado Q, el predicado Q^x_E se obtiene reemplazando en Q todas las apariciones libres de la variable x por E.
  1. $Q \equiv 0 \leq i < j < |a| \wedge_L a[i] \leq x < a[j]$.
     $Q^i_k \equiv 0 \leq k < j < |a| \wedge_L a[k] \leq x < a[j]$.
     $Q^i_{i+1} \equiv 0 \leq i+1 < j < |a| \wedge_L a[i+1] \leq x < a[j]$.
  2. $Q \equiv 0 \leq i < |a| \wedge_L (\forall j : \mathbb{Z})(a[j] = x)$.
     $Q^i_k \equiv 0 \leq k < |a| \wedge_L (\forall j : \mathbb{Z})(a[j] = x)$.
     $Q^j_k \equiv 0 \leq i < |a| \wedge_L (\forall j : \mathbb{Z})(a[j] = x)$.

Axioma 1: Asignación

- **Axioma 1.** $wp(\mathbf{x := E}, Q) \equiv \text{def}(E) \wedge_L Q^x_E$.
- Ejemplo:

$$\{??\}$$
$$\mathbf{x := x + 1}$$
$$\{Q : x \geq 7\}$$

- Tenemos que ...

$$\begin{aligned} wp(\mathbf{x := x{+}1}, Q) &\equiv \text{def}(x+1) \wedge_L Q^x_{x+1} \\ &\equiv \mathit{True} \wedge_L (x+1) \geq 7 \\ &\equiv x \geq 6 \end{aligned}$$

Axioma 1: Asignación

- Este axioma está justificado por la siguiente observación. Si buscamos la precondición más débil para el siguiente programa ...

$$\{??\}$$
$$\mathbf{x := E}$$
$$\{Q : x = 25\}$$

- ... entonces tenemos $wp(\mathbf{x := E}, Q) \equiv \text{def}(E) \wedge_L E = 25$.
- Es decir, si luego de **x := E** queremos que $x = 25$, entonces se debe cumplir $E = 25$ antes de la asignación!

Axioma 1: Asignación

- Otro ejemplo:

$$\{??\}$$
$$\mathbf{x := 2 * |x| + 1}$$
$$\{Q : x \geq 5\}$$

- Tenemos que ...

$$\begin{aligned} wp(\mathbf{x := 2 * |x| + 1}, Q) &\equiv \text{def}(2*|x|+1) \wedge_L Q^x_{2*|x|+1} \\ &\equiv \mathit{True} \wedge_L 2*|x|+1 \geq 5 \\ &\equiv |x| \geq 2 \\ &\equiv x \geq 2 \vee x \leq -2 \end{aligned}$$

Axioma 1: Asignación

- Un ejemplo más:

$$\{??\}$$

x := y*y

$$\{Q : x \geq 0\}$$

- Tenemos que ...

$$\begin{aligned} wp(\mathbf{x := y*y}, Q) &\equiv \text{def}(y * y) \ \wedge_L \ Q^x_{y*y} \\ &\equiv \mathit{True} \ \wedge_L \ y * y \geq 0 \\ &\equiv \mathit{True} \end{aligned}$$

Demostraciones de corrección

- Dijimos que $\{P\}$ **S** $\{Q\}$ sii $P \Rightarrow_L wp(\mathbf{S}, Q)$.
- Es decir, queremos que $P \Rightarrow_L wp(\mathbf{S}, Q)$ capture el hecho de que si **S** comienza en un estado que satisface P, entonces termina y lo hace en un estado que satisface Q.
- Por ejemplo, la siguiente tripla de Hoare es válida ...

$$\{P : x \geq 10\}$$

S: x := x+3

$$\{Q : x \neq 4\}$$

- ... puesto que:
  - $wp(\mathbf{S}, Q) \ \equiv \ x \neq 1$ y
  - $x \geq 10 \Rightarrow_L x \neq 1$.

Demostraciones de corrección

- La definición anterior implica que:
  1. Si $P \Rightarrow_L wp(\mathbf{S}, Q)$, entonces $\{P\}$ **S** $\{Q\}$ es válida (i.e., es verdadera).
  2. Si $P \not\Rightarrow_L wp(\mathbf{S}, Q)$, entonces $\{P\}$ **S** $\{Q\}$ no es válida (i.e., es falsa).
- Por ejemplo: $wp(\mathbf{x{:=}x{+}1}, x \geq 7) \equiv x \geq 6$.
- Como $x \geq 4 \not\Rightarrow_L x \geq 6$ (contraejemplo, $x = 5$), entonces se concluye que

$$\{P : x \geq 4\}$$

S: x := x+1

$$\{Q : x \geq 7\}$$

no es válida.

Más axiomas

- **Axioma 2.** $wp(\textbf{skip}, Q) \equiv Q$.
- **Axioma 3.** $wp(\textbf{S1; S2}, Q) \equiv wp(\textbf{S1}, wp(\textbf{S2}, Q))$.
- Ejemplo:

$$\{wp(\textbf{y := 2*x}, R)\} \equiv \{\mathrm{def}(2*x) \wedge_L 2*x \geq 6\} \equiv \{x \geq 3\}$$

y := 2*x;

$$\{wp(\textbf{x := y+1}, Q)\} \equiv \{\mathrm{def}(y+1) \wedge_L y+1 \geq 7\} \equiv \{y \geq 6\}$$

x := y + 1

$$\{Q : x \geq 7\}$$

Intercambiando los valores de dos variables

- **Ejemplo:** Recordemos el programa para intercambiar dos variables numéricas.
-
$$\{wp(\textbf{a := a + b}, E_2)\}$$
$$\equiv \{def(a+b) \wedge_L (b = B_0 \wedge (a+b) - b = A_0)\}$$
$$\equiv \{b = B_0 \wedge a = A_0\} \equiv \{E_3\}$$

```
a := a + b;
```

$$\{wp(\textbf{b := a - b}, E_1)\}$$
$$\equiv \{def(a-b) \wedge_L (a-(a-b) = B_0 \wedge a-b = A_0)\}$$
$$\equiv \{b = B_0 \wedge a-b = A_0\} \equiv \{E_2\}$$

```
b := a - b;
```

$$\{wp(\textbf{a := a - b}, Q)\}$$
$$\equiv \{def(a-b) \wedge_L (a-b = B_0 \wedge b = A_0)\}$$
$$\equiv \{a-b = B_0 \wedge b = A_0\} \equiv \{E_1\}$$

```
a := a - b;
```

$$\{Q\} \equiv \{a = B_0 \wedge b = A_0\}$$

Intercambiando los valores de dos variables

- Como $P \Rightarrow E_3 \equiv wp(S, Q)$, entonces podemos concluir que el algoritmo es correcto respecto de su especificación.
- Observar que los estados intermedios que obtuvimos aplicando *wp* son los mismos que habíamos usado para razonar sobre la corrección de este programa!

$\{a = A_0 \wedge b = B_0\}$

```
a := a + b;
```

$\{a = A_0 + B_0 \wedge b = B_0\}$

```
b := a - b;
```

$\{a = A_0 + B_0 \wedge b = A_0\}$

```
a := a - b;
```

$\{a = B_0 \wedge b = A_0\}$

- En lugar de razonar de manera informal, ahora podemos dar una demostración de que estos estados describen el comportamiento del algoritmo.

Recap: Axiomas wp

- **Axioma 1.** $wp(\textbf{x := E}, Q) \equiv \mathrm{def}(E) \wedge_L Q^x_E$.
- **Axioma 2.** $wp(\textbf{skip}, Q) \equiv Q$.
- **Axioma 3.** $wp(\textbf{S1; S2}, Q) \equiv wp(\textbf{S1}, wp(\textbf{S2}, Q))$.

Alternativas

- **Axioma 4.** Si **S** = **if B then S1 else S2 endif**, entonces

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

- Ejemplo:

$\{??\}$

S: if (x > 0) then y := x else y := -x endif

$\{Q : y \geq 2\}$

- Tenemos que ...

$$\begin{aligned} wp(\mathbf{S}, Q) &\equiv (x > 0 \wedge x \geq 2) \vee (x \leq 0 \wedge -x \geq 2) \\ &\equiv (x \geq 2) \vee (x \leq -2) \\ &\equiv |x| \geq 2 \end{aligned}$$

Alternativas

- La definicion operacional que vimos para demostrar la corrección de una alternativa es ahora un teorema derivado de este axioma!
- **Teorema.** Si $P \Rightarrow def(B)$ y

$$\{P \wedge B\} \ \mathbf{S1} \ \{Q\}$$
$$\{P \wedge \neg B\} \ \mathbf{S2} \ \{Q\}$$

entonces

$$\{P\} \ \textbf{if B then S1 else S2 endif} \ \{Q\}.$$

Alternativas

- **Demostración.**

$$\begin{aligned} & [P \wedge B \Rightarrow wp(\mathbf{S1}, Q)] \wedge [P \wedge \neg B \Rightarrow wp(\mathbf{S2}, Q)] \\ \equiv\ & [\neg(P \wedge B) \vee wp(\mathbf{S1}, Q)] \wedge [\neg(P \wedge \neg B) \vee wp(\mathbf{S2}, Q)] \\ \equiv\ & [\neg P \vee \neg B \vee wp(\mathbf{S1}, Q)] \wedge [\neg P \vee B \vee wp(\mathbf{S2}, Q)] \\ \equiv\ & \neg P \vee ([\neg B \vee wp(\mathbf{S1}, Q)] \wedge [B \vee wp(\mathbf{S2}, Q)]) \\ \equiv\ & P \Rightarrow [B \Rightarrow wp(\mathbf{S1}, Q)] \wedge [\neg B \Rightarrow wp(\mathbf{S2}, Q)] \\ \equiv\ & P \Rightarrow [B \wedge wp(\mathbf{S1}, Q)] \vee [\neg B \wedge wp(\mathbf{S2}, Q)] \\ \equiv\ & P \Rightarrow def(B) \wedge_L ([B \wedge wp(\mathbf{S1}, Q)] \vee [\neg B \wedge wp(\mathbf{S2}, Q)]) \\ \equiv\ & P \Rightarrow wp(\textbf{if B then S1 else S2 endif}, Q) \ \square \end{aligned}$$

Alternativas

- En el ejemplo anterior, vimos que:

$\{P : |x| \geq 2\}$

S: if (x > 0) then y := x else y := -x endif

$\{Q : y \geq 2\}$

- Veamos ahora la validez de esta tripla de Hoare por medio del teorema anterior.

$$\begin{array}{rcl} P \wedge B & \Rightarrow_L & wp(\mathbf{y := x}, Q) \\ |x| \geq 2 \wedge x > 0 & \Rightarrow_L & def(x) \wedge_L x \geq 2 \equiv x \geq 2 \quad \checkmark \end{array}$$

$$\begin{array}{rcl} P \wedge \neg B & \Rightarrow_L & wp(\mathbf{y := -x}, Q) \\ |x| \geq 2 \wedge x \leq 0 & \Rightarrow_L & def(x) \wedge_L -x \geq 2 \equiv x \leq -2 \quad \checkmark \end{array}$$

Asignación a elementos de una secuencia

- ¿Podemos usar el Axioma 1 para el programa `b[i]:=E`?
- El Axioma 1 matchea con `x:=E`, pero `x` es una variable, no una posición de una secuencia?
- Entonces, necesitamos reescribir $b[i] := E$ como $b := setAt(b, i, E)$.
- Donde

$$\begin{aligned} \text{def}(setAt(b,i,E)) \quad &= \quad (\text{def}(E) \ \wedge \text{def}(b) \ \wedge \ \text{def}(i)) \\ &\wedge_L \quad (0 \leq i < |b|). \end{aligned}$$

- **Observación:** En el libro de Gries se usa la notación $(b; i; E)$ en lugar de $setAt(b, i, E)$

Asignación a elementos de una secuencia

- Aplicando el Axioma 1, tenemos:

$$\begin{aligned} & wp(b[i] := E, Q) \\ \equiv \quad & wp(b := setAt(b,i,E), Q) \\ \equiv \quad & def(setAt(b,i,E)) \wedge_L Q^b_{setAt(b,i,E)} \\ \equiv \quad & ((def(b) \wedge def(i)) \wedge_L 0 \leq i < |b|) \wedge \ \text{def}(E)) \wedge_L Q^b_{setAt(b,i,E)} \end{aligned}$$

Además, se cumple que dados $0 \leq i, j < |b|$ sabemos que:

$$setAt(b,i,E)[j] \ = \ \begin{cases} E & \text{si } i = j \\ b[j] & \text{si } i \neq j \end{cases}$$

Asignación a elementos de una secuencia

- **Ejemplo.** Supongamos que i está definida y dentro del rango de la secuencia b.

$$\begin{aligned} & wp(\mathbf{b[i] := 5}, b[i] = 5) \\ \equiv \quad & ((\text{def}(i) \ \wedge_L \ 0 \leq i < |b|) \ \wedge \ \text{def}(5)) \ \wedge_L \ setAt(b,i,5)[i] = 5 \\ \equiv \quad & setAt(b,i,5)[i] = 5 \\ \equiv \quad & 5 = 5 \ \equiv \ True \end{aligned}$$

- **Ejemplo.** Con las mismas hipótesis.

$$\begin{aligned} & wp(\mathbf{b[i] := 5}, b[j] = 2) \\ \equiv \quad & setAt(b,i,5)[j] = 2 \\ \equiv \quad & (i \neq j \ \wedge \ setAt(b,i,5)[j] = 2) \ \vee \ (i = j \ \wedge \ setAt(b,i,5)[j] = 2) \\ \equiv \quad & (i \neq j \ \wedge \ b[j] = 2) \ \vee \ (i = j \ \wedge \ setAt(b,i,5)[i] = 2) \\ \equiv \quad & (i \neq j \ \wedge \ b[j] = 2) \ \vee \ (i = j \ \wedge \ 5 = 2) \\ \equiv \quad & i \neq j \ \wedge \ b[j] = 2 \end{aligned}$$

Propiedades

- Monotonía:
  - Si $Q \Rightarrow R$ entonces $wp(S, Q) \Rightarrow wp(S, R)$.
- Distributividad:
  - $wp(S, Q) \wedge wp(S, R) \Rightarrow wp(S, Q \wedge R)$,
  - $wp(S, Q) \vee wp(S, R) \Rightarrow wp(S, Q \vee R)$.
- *"Excluded Miracle"*:
  - $wp(S, false) \equiv false$.

Corolario de la monotonía

- **Corolario:** Si
  - $P \Rightarrow wp(S1, Q)$,
  - $Q \Rightarrow wp(S2, R)$,

  entonces
  - $P \Rightarrow wp(S1; S2, R)$.
- **Demostración.**

$$
\begin{array}{lll}
P & \Rightarrow & wp(S1, Q) \quad \text{(por hipótesis)} \\
 & \Rightarrow & wp(S1, wp(S2, R)) \quad \text{(monotonía)} \\
 & \equiv & wp(S1; S2, R) \quad \text{(Axioma 2)}
\end{array}
$$

Bibliografía

- David Gries - The Science of Programming
  - Part II - The Semantics of a Small Language
    - Chapter 7 - The Predicate Transformer wp
    - Chapter 8 - The Commands skip, abort and Composition
    - Chapter 9 - The Assignment Command
    - Chapter 10 - The Alternative Command