Demostraciones de corrección Parte I: Precondición más Débil

Algoritmos y Estructuras de Datos

30 de agosto de 2023

¿Qué vamos a ver?

- Estado y Correctitud de los programas
- Lenguaje SmallLang
- Predicados Especiales
- Axiomas
- Ejercicios

- ¿Por qué queremos probar que un programa es correcto?
  - Pasar del qué problema tenemos que resolver, al cómo resolverlo
  - Queremos que nuestros programas cumplan con su respectiva especificación
  - Desarrollar una fuerte intuición sobre cómo son y cómo hacer correctamente programas imperativos

Repaso: Noción de Estado

- **Estado de un programa**: valores que toman todas sus variables en un punto de su ejecución. (Antes de ejecutar la primera instrucción, entre dos instrucciones, y luego de ejecutar la última instrucción.)
  - ¿Qué instrucción permite que una variable pase de un estado a otro? La asignación.
  - ¿Siempre se va a ejecutar la misma sucesión de estados? No! Recordemos lo que ocurre con programas que tienen condicionales.
    Dependiendo del valor que tome su guarda, el flujo de la ejecución del programa puede ser distinto.

Repaso: Noción de Estado

- Mediante la transformación de estados vamos a intentar probar que un programa es correcto.
- Dada una especificación $E = (P, Q)$. El programa S es correcto respecto a ella, si siempre que S comienza en un estado que cumple P, el programa termina, y en el estado final se cumple Q.
- **Tripla de Hoare:** Si S es correcto respecto de E, se denota como $\{P\}$ **S** $\{Q\}$

Repaso: Triplas de Hoare

$\{P\}$ **codigo** $\{Q\}$
¿Es la siguiente tripla válida?

$$\{x \geq 4\}$$
$$\mathbf{x := x + 2}$$
$$\{x \geq 5\}$$

¿Es $\{x \geq 4\}$ la precondición más débil para el programa **x := x + 2** y la postcondición $\{x \geq 5\}$? Intuitivamente no!

Repaso: Precondición más débil (WP)

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.** $wp(\mathbf{S}, Q)$.

Ejemplo S: $\mathbf{x := x + 2}$ y $Q : x \geq 5$
$wp(\mathbf{S}, Q) = x \geq 3$

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

- Dada una expresión E, llamamos def(E) a las condiciones necesarias para que E esté definida.
- Dado un predicado Q, el predicado Q_E^x se obtiene reemplazando en Q todas las apariciones libres de la variable x por E.

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

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

- El programa `b[i]:=E` se reescribe como $b := setAt(b, i, E)$.
- Recordando,

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

- Aplicando el Axioma 1, tenemos que:

  $wp(b[i] := E, Q)$

  $$\equiv \quad ((def(b) \wedge def(i)) \wedge_L 0 \leq i < |b|) \wedge \mathrm{def}(E)) \wedge_L Q^{b}_{setAt(b,i,E)}$$

- Dados $0 \leq i, j < |b|$ sabemos que:

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

proc transformarEnPar (inout n: $\mathbb{Z}$) {
requiere $\{n = N_0 \wedge\ ??\}$
asegura $\{esPar(n) \wedge n > N_0\}$
}

Programa 1

$\{wp(\texttt{n := 2*n}, Q)\}$

$\equiv \{\text{def}(2 * n) \wedge_L (2 * n)\ mod\ 2 = 0 \wedge 2 * n > N_0\}$

$\equiv \{\text{def}(n) \wedge_L True \wedge n > N_0/2\}$

$\equiv \{n > N_0/2\}$

S1: `n := 2*n`

$\{Q : n\ mod\ 2 = 0 \wedge n > N_0\}$

Necesitamos que $P \rightarrow n > N_0/2$

$P : n = N_0 \wedge n > 0$

Programa 1

$\{wp(\texttt{n := n + 1}, Q)\}$

$\equiv \{\text{def}(n+1) \wedge_L (n+1)\ mod\ 2 = 0 \wedge (n+1) > N_0\}$

$\equiv \{\text{def}(n) \wedge_L n\ mod\ 2 = 1 \wedge n \geq N_0\}$

$\equiv \{n\ mod\ 2 = 1 \wedge n \geq N_0\}$

S2: `n := n + 1`

$\{Q : n\ mod\ 2 = 0 \wedge n > N_0\}$

Necesitamos que

$P \rightarrow (n\ mod\ 2 = 1 \wedge n \geq N_0)$

$P : n = N_0 \wedge n\ mod\ 2 = 1$

```
proc swap (inout a: ℤ, inout b: ℤ) {
  requiere {a = A0 ∧ b = B0 ∧ a ≠ 0 ∧ b ≠ 0}
  asegura {a = B0 ∧ b = A0}
}
```

$$\texttt{proc swap (inout a: } \mathbb{Z}\texttt{, inout b: } \mathbb{Z}\texttt{) \{}$$

S1: `a := a*b`
S2: `b := a/b`
S3: `a := a/b`

$\{wp(\mathbf{a := a*b}, E_2)\}$
$\equiv \{def(a*b) \wedge_L b \neq 0 \wedge_L a \neq 0 \wedge_L b = B_0 \wedge (a*b)/b = A_0\}$
$\equiv \{b \neq 0 \wedge_L a \neq 0 \wedge_L b = B_0 \wedge a = A_0\} \equiv \{E_3\}$

S1: `a := a*b`

$\{wp(\mathbf{b := a/b}, E_1)\}$
$\equiv \{def(a/b) \wedge_L a/b \neq 0 \wedge_L a/(a/b) = B_0 \wedge a/b = A_0\}$
$\equiv \{def(a) \wedge def(b) \wedge b \neq 0 \wedge_L a/b \neq 0 \wedge_L a/(a/b) = B_0 \wedge a/b = A_0\}$
$\equiv \{b \neq 0 \wedge_L a \neq 0 \wedge_L b = B_0 \wedge a/b = A_0\} \equiv \{E_2\}$

S2: `b := a/b`

$\{wp(\mathbf{a := a/b}, Q)\}$
$\equiv \{def(a/b) \wedge_L a/b = B_0 \wedge b = A_0\}$
$\equiv \{def(a) \wedge def(b) \wedge b \neq 0 \wedge_L a/b = B_0 \wedge b = A_0\}$
$\equiv \{b \neq 0 \wedge_L a/b = B_0 \wedge b = A_0\} \equiv \{E_1\}$

S3: `a := a/b;`

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

Ejercicio 3

```
proc diferenciaPositiva (in a: ℤ, in b: ℤ, out res: ℤ) {
  requiere {??}
  asegura {res = |a − b|}
}
```

- Programa 1
  S1: `if (a > b) then res := a - b else res := b - a endif`
- Programa 2
  S2: `res := a-b`

- Programa 1

  $\{wp(\mathbf{S1}, Q)\}$
  $\equiv \{def(a > b) \wedge_L ((a > b \wedge wp(res := a - b, Q)) \vee (\neg(a > b) \wedge wp(res := b - a, Q)))\}$
  $\equiv \{((a > b \wedge def(a - b) \wedge_L a - b = |a - b|) \vee (((\neg(a > b) \wedge def(b - a) \wedge_L b - b = |a - b|))\}$
  $\equiv \{((a > b \wedge a-b = |a-b|) \vee (\neg(a > b) \wedge b-a = |a-b|))\}$
  $\equiv \{True\}$

  S1: `if (a > b) then res := a - b else res := b - a endif`

  $\{Q\} \equiv \{res = |a - b|\}$

  Con esta implementación, la precondición puede ser True :)

- Programa 2

  $\{wp(\textbf{res := a-b}, Q)\}$
  $\equiv \{def(a-b) \wedge_L a - b = |a-b|\}$
  $\equiv \{(a-b \geq 0 \wedge a-b = a-b) \vee (a-b < 0 \wedge a-b = b-a)\}$
  $\equiv \{a - b \geq 0 \vee False\}$

  S2: `res := a-b`

  $\{Q\} \equiv \{res = |a-b|\}$

  Luego $P : a - b \geq 0$, la precondición deja de ser True :(

Calcular $wp(\mathbf{A[i+2] := 0}, Q)$, siendo $Q \equiv (\forall j : \mathbb{Z})(0 \leq j < |A| \rightarrow_L A[j] \geq 0)$, i es una variable entera y A es una secuencia de reales.

$\{wp(\textbf{A[i+2] := 0}, Q)\}$
$\equiv \{wp(\textbf{setAt(A, i+2, 0)}, Q)\}$
$\equiv \{0 \leq i+2 < |A| \wedge_L$
$(\forall j : \mathbb{Z})(0 \leq j < |setAt(A, i+2, 0)| \rightarrow_L setAt(A, i+2, 0)[j] \geq 0)\}$
$\equiv \{0 \leq i+2 < |A| \wedge_L$
$(\forall j : \mathbb{Z})((0 \leq j < |A| \wedge i+2 \neq j) \rightarrow_L setAt(A, i+2, 0)[j] \geq 0)\}$.
Donde el último paso se deduce de usar que:

$$setAt(A, i, 0)[j] = \begin{cases} 0 & \text{si } j = i+2 \\ A[j] & \text{si } j \neq i+2 \end{cases}$$

$\equiv \{0 \leq i+2 < |A| \wedge_L$
$(\forall j : \mathbb{Z})((0 \leq j < |A| \wedge i+2 \neq j) \rightarrow_L setAt(A, i+2, 0)[j] \geq 0) \wedge \ setAt(A, i+2, 0)[i+2] \geq 0\}$.
$\equiv \{0 \leq i+2 < |A| \wedge_L$
$(\forall j : \mathbb{Z})((0 \leq j < |A| \wedge i+2 \neq j) \rightarrow_L A[j] \geq 0) \wedge \ 0 \geq 0\}$.

proc sumarTodos (in s: $seq\langle\mathbb{Z}\rangle$, in n: $\mathbb{Z}$, inout suma: $\mathbb{Z}$) {
  requiere
    $\{suma = suma_0 \wedge |s| > 0 \wedge n = |s| \wedge suma = \sum_{i=0}^{|s|-2} s[i]\}$
  asegura $\{suma = \sum_{i=0}^{|s|-1} s[i]\}$
}

S: `suma := suma + s[n-1]`

$\{wp(\textbf{suma:=suma+s[n-1]}, Q)\}$
$\equiv \{def(suma + s[n-1]) \wedge_L suma + s[n-1] = \sum_{i_0}^{|s|-1} s[i]\}$
$\equiv \{0 \leq n-1 < |s| \wedge_L suma + s[n-1] = \sum_{i_0}^{|s|-1} s[i]\}$

S: `suma := suma + s[n-1]`

$\{Q\} \equiv \{suma = \sum_{i_0}^{|s|-1} s[i]\}$

Veamos ahora que
$P \rightarrow (0 \leq n-1 < |s| \wedge_L suma + s[n-1] = \sum_{i_0}^{|s|-1} s[i])$
Dado que $n = |s|$, entonces $n-1 = |s|-1$. Luego, $n-1 < |s|$.
Además $n > 1$, luego $0 \leq n-1$.
Según P, $suma = \sum_{i=0}^{|s|-2} s[i]$. Entonces,
$suma + s[n-1] = \sum_{i=0}^{|s|-2} s[i] + s[n-1] = \sum_{i=0}^{|s|-1} s[i]$, como queríamos.