Lógica de Hoare

Sebastian Mestre

¿Qué es una trípleta de Hoare?

- La **lógica de Hoare** es una forma de razonar formalmente sobre la corrección de programas.
- Una **trípeta de Hoare** se escribe:

$$\{P\}\ C\ \{Q\}$$

donde:

  - P: **precondición**, afirmación lógica sobre el estado antes de ejecutar el código.
  - C: **comando** o fragmento de código.
  - Q: **postcondición**, afirmación lógica sobre el estado después de ejecutar C.

- Lectura: **"Si P es verdadera antes de ejecutar C y se ejecuta C, entonces Q será verdadera al finalizar"**.

Ejemplo simple

- Consideremos el fragmento:

Programa

```
1 x = x * 6;
```

- El mismo programa puede aparecer en **muchas** trípletas válidas, según la propiedad que nos interese:

Algunos ejemplos

$$\{ x \,\%2 == 1 \}\ x = x * 6;\ \{ x \,\%2 == 1 \}$$

$$\{ x \,\%2 == 0 \}\ x = x * 6;\ \{ x \,\%2 == 0 \}$$

$$\{ x \,\%2 == 0 \}\ x = x * 6;\ \{ x \,\%4 == 0 \}$$

$$\{ x > 16 \}\ x = x * 6;\ \{ x > 96 \}$$

Precondición más débil

- Fijada una **postcondición** Q y un programa C, hay muchas precondiciones P tales que:

$$\{P\}\ C\ \{Q\}$$

- La **precondición más débil** es aquella $P_{\text{mín}}$ que es implicada por cualquier otra precondición que garantice Q.
- Intuitivamente: es la condición "mínima" que asegura que, tras ejecutar el programa, la postcondición se cumple.

Ejemplo

Para el programa $x = x * 6$ y la postcondición

$$x\,\%12 == 0,$$

una precondición más débil adecuada es:

$$x\,\%2 == 0.$$

Cómo se usa en la práctica

- En teoría existen reglas para calcular (semi-automáticamente) la precondición más débil de una postcondición dada.
- En la práctica:
  - Se arranca desde la **postcondición deseada** al final del programa.
  - Se van propagando hacia atrás las **condiciones necesarias** para que esa postcondición se cumpla.
  - Si en algún punto podemos garantizar una precondición **más fuerte** que la mínima requerida, no hay problema.
- El concepto dual de **postcondición más fuerte** existe, pero es mucho más difícil de calcular y casi no se usa.

Composición secuencial

- Una regla básica de la lógica de Hoare es la **composición secuencial**.

Regla

Si

$$\{P\}\ C_1\ \{Q\} \quad \text{y} \quad \{Q\}\ C_2\ \{R\},$$

entonces

$$\{P\}\ C_1; C_2\ \{R\}.$$

- Interpretación informal:
  - C_1 transforma estados con P en estados con Q.
  - C_2 transforma estados con Q en estados con R.
  - Por lo tanto, ejecutar primero C_1 y luego C_2 transforma estados con P en estados con R.

Bucle `while` e invariantes

- Para razonar sobre un bucle `while` usamos un **invariante de bucle** I.
- Un invariante es una propiedad lógica que:
  - Es verdadera antes de entrar al bucle.
  - Se mantiene tras cada iteración.
  - Sigue siendo verdadera cuando el bucle termina.
- Además, cuando el bucle termina, la **condición del while es falsa**.

Esquema de razonamiento para `while`

Esquema

```
// {I}
while (condicion) {
  // {I && condicion}
  // ...codigo que debe preservar I...
  // {I}
}
// {I && !condicion}

```

- Para demostrar la corrección:
  1. Encontrar un invariante I que sea cierto antes de entrar al bucle.
  2. Probar que si I es cierto al inicio de una iteración y la condición es verdadera, entonces tras el cuerpo del bucle I sigue siendo cierto.
  3. Al terminar el bucle, usar $I \wedge \neg$condición para implicar la postcondición deseada.

Lógica de Hoare en la vida real

- En programación competitiva **no** escribimos demostraciones formales en notación de Hoare.
- Pero los conceptos fundamentales sí se usan todo el tiempo:
  - **Invariantes** de estructuras de datos y bucles.
  - **Precondiciones** de funciones y rutinas auxiliares.
  - **Postcondiciones** que queremos garantizar al terminar un algoritmo.
- Ejemplos típicos:
  - "Al inicio de cada iteración, los primeros k elementos están ordenados".
  - "Al entrar al bucle, `res` contiene el resultado parcial correcto hasta ahora".

Más allá de los bucles

- El razonamiento con invariantes se usa también en:
  - Recursiones: qué “promesa” cumple cada llamada recursiva.
  - Ecuaciones y desigualdades en problemas matemáticos.
  - Recorridos no triviales sobre arreglos y grafos.
- Muchas veces “sentimos” que un algoritmo es correcto porque identificamos claramente:
  - Qué se mantiene invariante.
  - Qué precondiciones necesita cada paso.
  - Qué postcondición queremos al final.

Ideas clave

- La lógica de Hoare da un **lenguaje formal** para hablar de precondiciones, postcondiciones e invariantes.
- Aunque no se use la notación completa en concursos, la forma de pensar sí está profundamente influenciada por estas ideas.
- Adoptar este estilo de razonamiento es clave para escribir algoritmos correctos y poder justificar por qué funcionan.

Preguntas

Q&A