Trabajo Práctico Nro 13 - Verificación Axiomática de Programas (lógica de Hoare)

Ejercicio 1

Responder breve y claramente los siguientes incisos:

1. ¿En qué se diferencia una prueba semántica de una prueba sintáctica de un programa?

La prueba semántica usa la semántica de las instrucciones del lenguaje para su prueba, mientras que la prueba sintáctica usa axiomas y reglas de un método lógico-deductivo para su prueba.

2. ¿Qué es un estado de un programa? ¿Cuándo un estado satisface un predicado?

Un estado σ es una función que asigna a toda variable un valor. Por ejemplo: $\sigma(x) = 1, \sigma(y) = 2$, etc. Un estado σ satisface un predicado p, si p evaluado con σ es verdadero. Se expresa así: $\sigma \models p$. Por ejemplo: si $\sigma(x) = 1$ y $\sigma(y) = 2$, entonces $\sigma \models x < y$.

3. ¿Cómo se especifica un programa?

Un programa se especifica en base a la precondición y post-condición del mismo. ambas son predicados que se deben cumplir justamente antes de empezar el programa y al terminar el programa, podemos usar un par que contenga a estas para especificar al programa. Por ejemplo, el par $(x = X \wedge y = Y, y = X \wedge x = Y)$ es la especificación de S_{swap}.

4. ¿Cuál es el significado de la fórmula $\{p\}S\{q\}$?

La fórmula $\{p\}S\{q\}$ significa que: Un programa S es correcto con respecto a una especificación (p, q), es decir, para todo estado σ, si $\sigma \models p$ entonces S ejecutado a partir de σ termina en un estado $\sigma´$ tal que $\sigma´ \models q$.

5. ¿Qué son el invariante y el variante de un while?

Un invariante es una propiedad lógica que se mantiene verdadera:

- Antes de entrar al bucle
- Durante cada iteración
- Al salir del bucle (si la condición es falsa)

Es una especie de *"regla"* que el programa *nunca rompe*, sin importar cuántas veces se ejecute el bucle.

Una **variante** es una expresión que decrece en cada iteración del bucle y que sirve para asegurar que el bucle terminará. Nunca se vuelve negativa o llega a un caso base, y representa el número máximo de iteraciones del bucle while.

6. ¿Cuándo un método axiomático es sensato, y cuándo es completo?

Un **método axiomático es sensato** si prueba realmente lo que se quiere probar (no deriva en contradicciones), es decir, si se prueba $\{p\}S\{q\}$, se debe cumplir $\{p\}S\{q\}$. Propiedad **obligatoria**.

Un **método axiomático es completo** cuando todo lo que es verdadero dentro de la ejecución de un programa se puede demostrar usando al método, es decir, si se cumple $\{p\}S\{q\}$, se debe poder probar $\{p\}S\{q\}$. Propiedad **deseable**. La completitud depende de la **expresividad** del lenguaje de especificación.

7. ¿Por qué la lógica de Hoare para los programas secuenciales es composicional?

La lógica de Hoare para los programas secuenciales es **composicional** ya que: Dado un programa S, este está compuesto por subprogramas $S_1, \ldots, S_n$ donde para que valga la fórmula $\{p\}S\{q\}$ se depende **sólo** de que valgan fórmulas $\{p_1\}S_1\{q_1\}, \ldots, \{p_n\}S_n\{q_n\}$, **sin importar el contenido de los S_i** (noción de **caja negra**).

Ejercicio 2

Especificar un programa que a partir de un estado inicial en el que x sea mayor que 0, termine en un estado final en el que y sea el cuadrado del valor inicial de x, y además x tenga su valor inicial.

La **especificación** sería: $(x = X \land X > 0, y = X^2 \land x = X)$

Ejercicio 3

Decir y justificar informalmente (es decir sin usar la lógica de Hoare), si se cumplen o no las siguientes fórmulas. *Comentario: el predicado true representa cualquier estado:*

1. $\{x > 0\}\text{while x} > 0 \text{ do x} := \text{x} - 2 \text{ od}\{true\}$

Esta fórmula cumple, no importa si el valor es un $x > 0$ par o impar, siempre el while va a terminar y el programa quedará en cualquier estado una vez finaliza (true).

2. $\{x > 0\}\text{while x} > 0 \text{ do x} := \text{x} - 2 \text{ od}\{x = 0\}$

Esta fórmula no cumple, si el valor de x es un número impar mayor que cero, el programa no termina en la post-condición que se especifica.

3. $\{x > 0\}\text{while x} \neq 0 \text{ do x} := \text{x} - 2 \text{ od}\{true\}$

Esta fórmula no cumple, si el valor de x es un número impar mayor que cero, el programa no termina ya que se queda loopeando infinitamente.

Ejercicio 4

Probar (usando la lógica de Hoare): $\{true\}\text{x} := 0 \text{ ; x} := \text{x} + 1 \text{ ; x} := \text{x} + 2\{x = 3\}$.

Proceso

1. Empezamos desde la postcondición final.
2. Aplicamos ASI hacia atrás para formar las precondiciones necesarias para después usar SEC.
3. Encadenamos con SEC (si no quedan bien las condiciones intermedias y coinciden exactamente)
4. Usamos CONS en este caso para generalizar la precondición y probar lo que se pide.

1. $\{x + 2 = 3\}x := x + 2\{x = 3\}$ - ASI
2. $\{(x + 1) + 2 = 3\}x := x + 1\{x + 2 = 3\}$ - ASI
3. $\{(0 + 1) + 2 = 3\}x := 0\{(x + 1) + 2 = 3\}$ - ASI
4. $\{(0 + 1) + 2 = 3\}x := 0; x := x + 1\{x + 2 = 3\}$ - SEC 2 Y 3
5. $\{(0 + 1) + 2 = 3\}x := 0; x := x + 1; x := x + 2\{x = 3\}$ - SEC 4 Y 1
6. $\{true\}\text{x} := 0 \text{ ; x} := \text{x} + 1 \text{ ; x} := \text{x} + 2\{x = 3\}$ - CONS ya que: $\{true\} \rightarrow \{(0 + 1) + 2 = 3\}$; es una verdad matemática

---

Ejercicio 5

Se cumple $\{x = 10\}\text{while x} > 0 \text{ do x} := \text{x} - 1 \text{ od}\{x = 0\}$. Se pide probar (usando la lógica de Hoare) que el predicado $p = (x \geq 0)$ es un invariante del while. *Ayuda: hay que probar, por un lado: $x = 10 \longrightarrow p$, y por otro lado: $\{p \wedge x > 0\}\text{x} := \text{x} - 1\{p\}$.*

Prueba de $x = 10 \longrightarrow p$, es decir, $x = 10 \longrightarrow (x \geq 0)$:

1. Trivialmente se ve que $x = 10$ es mayor o igual que 0.

Prueba de $\{p \wedge x > 0\}\text{x} := \text{x} - 1\{p\}$, es decir, $\{x \geq 0 \wedge x > 0\}\text{x} := \text{x} - 1\{x \geq 0\}$:

1. $\{x - 1 \geq 0\}\text{x} := \text{x} - 1\{x \geq 0\}$ - **ASI**
2. $\{x \geq 0 \wedge x > 0\}\text{x} := \text{x} - 1\{x \geq 0\}$ - **CONS ya que $\{x \geq 0 \wedge x > 0\} \rightarrow \{x - 1 \geq 0\}$ es aritméticamente cierto, sabemos que $x > 0$, entonces se cumple siempre que $x - 1 \geq 0$ para cualquier $x > 0$**

Con esto, **queda demostrado que $p = (x \geq 0)$ es invariante del while**.

Ejercicio 6

La instrucción *repeat S until B* consiste en ejecutar S, luego evaluar B, si B es falsa volver a iterar, y en caso contrario terminar. Explicar informalmente por qué la siguiente regla para probar la instrucción es sensata (es decir, si se cumplen las premisas, entonces también se cumple la conclusión):

$$\frac{\{p\}S\{q\}, q \wedge \neg B \longrightarrow p}{\{p\}\text{repeat S until B}\{q \wedge B\}}$$

La instrucción *repeat S until B* ejecuta primero *S*, y luego evalúa *B*.

- La primera premisa $\{p\}S\{q\}$ garantiza que si se parte de un estado que cumple *p*, entonces después de ejecutar *S*, se obtiene un estado que cumple *q*.
- La segunda premisa $q \wedge \neg B \longrightarrow p$ asegura que si al finalizar *S* (terminamos en un estado que cumple *q*) todavía no se cumple *B*, entonces es válido repetir el ciclo, ya que se vuelve a cumplir *p*.
- De este modo, el ciclo puede repetirse varias veces manteniendo la validez de las condiciones.
- Finalmente, si se cumple *B*, el ciclo termina en un estado que cumple $q \wedge B$, como se establece en la postcondición.

Por eso, si ambas premisas se cumplen, la regla es sensata.