Metodología de la Programación

Grado en Ingeniería Informática de Gestión y Sistemas de Información
Escuela de Ingeniería de Bilbao (UPV/EHU)
Departamento de Lenguajes y Sistemas Informáticos
Curso: 1^o
Curso académico: 2016-2017
Tema 3: Verificación formal de programas
2 puntos

31-05-2017

Enunciado

Índice

Lista de figuras

Lista de tablas

****

1 Verificación formal de un programa iterativo (*2 puntos*)

Verificar, utilizando el Cálculo de Hoare, la corrección total del programa que se muestra en la figura 1 (página 3), con respecto a la precondición φ, la postcondición ψ, el invariante INV y la expresión cota E que se muestran en esa misma figura. Según la especificación pre-post formalizada mediante las fórmulas φ y ψ, dados un entero x que es positivo y potencia de 2 y un vector no vacío de enteros $A(1..n)$ cuyos valores son todos positivos y menores o iguales que x, el programa debería decidir en la variable booleana w si en alguna posición (al menos en una) el valor de $A(1..n)$ es una potencia de 2.

En el programa de la figura 1, mod representa el resto de la división entera. Ejemplos: $19\ mod\ 2 = 1$; $19\ mod\ 3 = 1$; $19\ mod\ 4 = 3$; $17\ mod\ 3 = 2$; $8\ mod\ 12 = 8$; $15\ mod\ 5 = 0$.

Las potencias enteras de 2 son $2^0, 2^1, 2^2, 2^3, 2^4, 2^5, \ldots$. Es decir, $1, 2, 4, 8, 16, 32, \ldots$. El conjunto formado por todas las potencias enteras de 2 se puede definir formalmente de la siguiente manera: $\{y \mid y \in \mathbb{N} \land \exists \ell(\ell \geq 0 \ \land \ y = 2^{\ell})\}$, donde $\mathbb{N}$ es el conjunto de los números naturales, es decir, $\mathbb{N} = \{0, 1, 2, 3, 4, \ldots\}$.

Cualquier potencia entera de 2 es un número positivo (≥ 1).

Durante el proceso de verificación, al analizar el punto (IV) de la regla del While, es necesario utilizar la propiedad matemática $(Prop)$ de la figura 2 (página 3), que dice que, si se tienen dos valores enteros u y v donde tanto u como v solo pueden ser positivos (≥ 1) y, además, u es una potencia entera de 2 y es mayor o igual que v, entonces el resto de dividir u por v será 0 si y solo si v es una potencia de 2, es decir, que el que el resto de dividir u por v sea 0 es equivalente a decir que v es una potencia de 2. Por ejemplo, si $u = 32 = 2^5$ y $v = 8$, tenemos que el resto de dividir u por v es 0 y v es una potencia de 2: $v = 2^3$. En cambio, si $u = 32 = 2^5$ y $v = 10$, tenemos que el resto de dividir u por v no es 0 y v no es una potencia de 2.

En la tabla 1 (página 3) se recogen las abreviaciones que se recomienda utilizar durante el proceso de verificación. En la tabla 2 (página 4) se recopilan las denominaciones de las letras griegas utilizadas en este enunciado. Finalmente, en la tabla 3 (página 4) se muestra la puntuación de los distintos pasos o apartados que han de ser considerados en el proceso de verificación.

Teniendo en cuenta las abreviaciones de la tabla 1, la propiedad $(Prop)$ de la figura 2 expresa que cada expresión lógica $\gamma(\ell)$ tiene el mismo valor que la correspondiente expresión lógica $\sigma(\ell)$, siempre que se garantice que x y $A(\ell)$ sean positivos, que x es una potencia de 2 y que x es mayor o igual que $A(\ell)$.

Ejemplo 1.1. (Para el programa de la figura 1) *Sean $x = 32$ y el siguiente vector $A(1..8)$:*

A(1..8)	*30*	*10*	*12*	*16*	*1*	*9*	*8*	*11*
	1	*2*	*3*	*4*	*5*	*6*	*7*	*8*

Para esos valores de x y $A(1..8)$, el programa de la figura 1 devolvería el valor booleano $True$ en w, porque al menos un elemento de $A(1..8)$ es una potencia entera de 2. En concreto, se cumple para $A(4)$, $A(5)$ y $A(7)$, puesto que $A(4) = 16 = 2^4$, $A(5) = 1 = 2^0$ y $A(7) = 8 = 2^3$.

En cambio, para $x = 32$ y el siguiente vector $A(1..8)$:

A(1..8)	*30*	*10*	*12*	*23*	*30*	*9*	*80*	*11*
	1	*2*	*3*	*4*	*5*	*6*	*7*	*8*

el programa de la figura 1 devolvería el valor booleano $False$ en w, porque no hay ningún elemento de $A(1..8)$ que sea potencia entera de 2.

Programa:

$\{\varphi\}$
w := False;
while $\{INV\}$ $\{E\}$ i $\neq$ n **and not** w **loop**
  i := i + 1;
  w := (x **mod** A(i) = 0);
end loop;
$\{\psi\}$

Definición de φ, INV, E y ψ:

$$\varphi \equiv \quad n \geq 1 \ \wedge \ pot\_dos(x) \ \wedge \ positmenorig(x, A(1..n)) \ \wedge \ i = 0$$

$$INV \equiv \quad n \geq 1 \ \wedge \ pot\_dos(x) \ \wedge \ positmenorig(x, A(1..n)) \ \wedge \ (0 \leq i \leq n) \wedge (w \leftrightarrow algun\_divisor(x, A(1..n), i))$$

$$E = \quad n - i$$

$$\psi \equiv \quad w \leftrightarrow \exists k(1 \leq k \leq n \ \wedge \ pot\_dos(A(k)))$$

Definición de los predicados utilizados:

$$pot\_dos(z) \equiv \exists \ell(\ell \geq 0 \ \wedge \ z = 2^{\ell})$$

$$positmenorig(z, G(1..r)) \equiv \forall k(1 \leq k \leq r \ \rightarrow \ (G(k) \geq 1 \ \wedge \ G(k) \leq z))$$

$$algun\_divisor(z, F(1..r), pos) \equiv \exists k(1 \leq k \leq pos \ \wedge \ (z \ mod \ F(k)) = 0)$$

Figura 1: Programa a verificar, definiciones de φ, INV, E y ψ y definiciones de los predicados utilizados.

Propiedad ($Prop$):

$$\forall u, v((u \geq 1 \ \wedge \ v \geq 1 \ \wedge \ pot\_dos(u) \ \wedge \ u \geq v) \rightarrow ((u \ mod \ v = 0) \leftrightarrow pot\_dos(v)))$$

Figura 2: Propiedad que cumplen dos números enteros u y v cualesquiera.

Abreviaciones recomendadas:

$$\lambda \equiv n \geq 1 \ \wedge \ pot\_dos(x) \ \wedge \ positmenorig(x, A(1..n))$$

$$B \equiv i \neq n \ and \ not \ w$$

$$\gamma(\ell) \equiv x \ mod \ A(\ell) = 1$$

$$\sigma(\ell) \equiv pot\_dos(A(\ell))$$

$$\mu(\ell) \equiv algun\_divisor(x, A(1..n), \ell)$$

Tabla 1: Abreviaciones que se recomienda utilizar.

Letras griegas utilizadas en el enunciado:
φ : fi ψ : psi γ : gamma σ : sigma μ : mu λ : lambda

Tabla 2: Denominaciones de las letras griegas utilizadas en el enunciado.

Puntuación:

- *(a)* Partición inicial y esquema: 0,050
- *(b)* Verificación de la asignación inicial: 0,150
  (Cálculo de fórmulas: 0,050. Comprobación de la implicación: 0,100)
- *(c)* Punto (I) de la regla del while: 0,005
- *(d)* Punto (II) de la regla del while: 0,020
- *(e)* Punto (III) de la regla del while: 0,750
  (Cálculo de fórmulas: 0,150. Comprobación de la implicación: 0,600)
- *(f)* Punto (IV) de la regla del while: 0,600
  (Casos $i = n$: 0,250. Caso $i \neq n$: 0,350)
- *(g)* Punto (V) de la regla del while: 0,075
- *(h)* Punto (VI) de la regla del while: 0,200
  (Cálculo de fórmulas: 0,050. Comprobación de la implicación: 0,150)
- *(i)* Demostración formal de la corrección: 0,150

■ Cuando no se explique por qué se cumple una implicación, se contará cero. Es decir, indicar que una implicación sí se cumple sin razonar por qué se cumple cuenta 0.

■ Para aprobar el ejercicio es obligatorio obtener al menos la mitad de la puntuación tanto del apartado *(e)* como del apartado *(f)* (puntos (III) y (IV) de la regla del while).

Tabla 3: Puntuación por apartados.