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

Solución

Índice

Lista de figuras

Lista de tablas

****

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

1.1 Enunciado

Verificar, utilizando el Cálculo de Hoare, la corrección total del programa que se muestra en la figura 1 (página 4), 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: $\{y \mid y \in \mathbb{N} \wedge \exists \ell(\ell \geq 0 \ \wedge \ y = 2^\ell)\}$, 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 4), 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 4) se recogen las abreviaciones que se recomienda utilizar durante el proceso de verificación. En la tabla 2 (página 5) se recopilan las denominaciones de las letras griegas utilizadas en este enunciado. Finalmente, en la tabla 3 (página 5) 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.

1.2 Solución

En la tabla 9 (página 16) se recopilan las denominaciones de las letras griegas adicionales utilizadas en el apartado correspondiente a la solución.

1.2.1 *(a)* Partición inicial y esquema

La información correspondiente a la partición inicial y a los esquemas de los dos programas que surgen tras la partición se muestra en las figuras 3, 4 y 5 (páginas 5, 13 y 14 respectivamente).

En la figura 3 (página 5), se ha colocado el invariante INV entre la inicialización de la variable w y la instrucción $while$. En la figura 4 (página 13) se muestran los dos programas que se obtienen tras la partición en dos del programa original. La tarea es ahora verificar, por separado, la corrección total de esos dos programas (programa 1 y programa 2). Finalmente, en la figura 5 (página 14), se muestran las fórmulas que han de ser calculadas y las implicaciones que han de ser analizadas para decidir si los dos programas son correctos.

1.2.2 *(b)* Verificación de la asignación inicial

- Cálculo de la fórmula φ_1 a partir de la fórmula INV utilizando el axioma de la asignación (AA).

$$\begin{aligned}\varphi_1 &\equiv def(False) \ \wedge \ INV_w^{False} \\ &\equiv True \ \wedge \ \lambda \ \wedge \ (0 \leq i \leq n) \ \wedge \ (False \leftrightarrow \mu(i)) \\ &\equiv \lambda \ \wedge \ (0 \leq i \leq n) \ \wedge \ \neg\mu(i)\end{aligned}$$

  En la espresión booleana $False$, no hay posibilidad de que surja ningún problema a la hora de calcularla o utilizarla: ni división por 0, ni acceso a vector con índice fuera de rango, etc. Para indicar que no es necesario que se cumpla ninguna condición, se ha de poner $True$. Es decir, $def(False)$ es $True$.

  Para simplificar φ_1, por una parte, se ha tenido en cuenta que $(True \ \wedge \ \delta) \equiv \delta$ para cualquier fórmula δ. Por otra parte, se ha tenido en cuenta que $(False \leftrightarrow \delta) \equiv \neg\delta$ para cualquier fórmula δ.

- Comprobación de la implicación: ¿$\varphi \rightarrow \varphi_1$?

$$¿\underbrace{\lambda \ \wedge \ i = 0}_{\varphi} \rightarrow \underbrace{\lambda \ \wedge \ (0 \leq i \leq n) \ \wedge \ \neg\mu(i)}_{\varphi_1}?$$

  En la primera parte (parte izquierda) de la implicación tenemos la información: $\lambda \ \wedge \ i = 0$. En la segunda parte (parte derecha) tenemos cuatro preguntas: ¿λ? ¿$(0 \leq i)$? ¿$(i \leq n)$? ¿$\neg\mu(i)$?.

  - ¿λ? Sí, porque en la primera parte, es decir, en φ, aparece λ.
  - ¿$(0 \leq i)$? Es decir, ¿$(0 = i) \vee (0 < i)$? Sí, porque en la primera parte, es decir, en φ, se tiene que $i = 0$.
  - ¿$(i \leq n)$? Sí, porque en la primera parte, es decir, en φ, dentro de λ se tiene que $n \geq 1$. Además, en φ se tiene que $i = 0$, lo cual significa que $i \leq 1$. Por consiguiente, $i \leq n$. Otra manera de argumentar esto sería transformar la pregunta ¿$(i \leq n)$? en ¿$(0 \leq n)$? porque sabemos que i es 0. Puesto que λ se nos dice que $1 \leq n$, también se cumple $0 \leq n$.

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.

El invariante se cumple también justo antes de empezar el $while$

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

Figura 3: Programa a verificar, con el invariante entre la inicialización de w y la instrucción $while$.

– ¿$\neg\mu(i)$? En φ se tiene que $i = 0$. Por tanto, nos queda la pregunta ¿$\neg\mu(0)$? Si recuperamos el significado de $\mu(0)$, tenemos la siguiente pregunta:

$$¿\neg algun\_divisor(x, A(1..n), 0)?$$

Teniendo en cuenta la definición de $algun\_divisor(x, A(1..n), 0)$, la pregunta es la siguiente:

$$¿\neg\exists k(1 \leq k \leq 0 \ \wedge \ x \ mod \ A(k) = 0)?$$

El dominio de definición de la formula existencial es vacío y, consecuentemente, la fórmula existencial es $False$. La pregunta es, por tanto, la siguiente: ¿$\neg False$? Es decir, ¿$\neg False$ es $True$? La respuesta es afirmativa.

1.2.3 *(c)* Punto (I) de la regla del while

¿Se cumple $INV \rightarrow INV$? Se nos dice, en la primera parte de la implicación, que INV es cierto y en la segunda parte se nos pregunta si INV es cierto. La respuesta es que sí.

Si queremos argumentarlo de manera más técnica, tenemos que $\delta \rightarrow \delta$ es $True$ siempre. Recordemos que $\delta \rightarrow \delta$ es equivalente a $(\neg\delta) \vee \delta$. Puesto que cualquier fórmula lógica δ ha de ser necesariamente o $True$ o $False$, se deduce que $(\neg\delta) \vee \delta$ es siempre $True$ porque alguno de los dos componentes será $True$. Es decir, $\neg\delta$ será $True$ o δ será $True$.

1.2.4 *(d)* Punto (II) de la regla del while

¿$INV \rightarrow def(B)$?
¿$INV \rightarrow def(i \neq n \ and \ not \ w)$?
¿$INV \rightarrow True$?

En la espresión booleana $(i \neq n \ and \ not \ w)$, no hay posibilidad de que surja ningún problema a la hora de calcularla: ni división por 0, ni acceso a vector con índice fuera de rango, etc. Como no se ha de cumplir ninguna condición para poder calcular el valor de B, tenemos que $def(B)$ es $True$. La pregunta final es, por tanto, ¿se cumple $True$?. La información que tenemos es que la fórmula INV es cierta. La respuesta es afirmativa, porque $True$ se cumple siempre, es decir, $True$ es $True$ independientemente de la información que se tenga en INV.

Técnicamente, tenemos que $\delta \rightarrow True$ es siempre $True$. Recordemos que $\delta \rightarrow True$ es equivalente a $(\neg\delta) \vee True$. Por consiguiente, estamos ante una disyunción donde uno de los componentes es $True$. Eso conlleva que toda la fórmula $(\neg\delta) \vee True$ sea $True$.

1.2.5 *(e)* Punto (III) de la regla del while

- Cálculo de la fórmula φ_2 a partir de la fórmula INV utilizando el axioma de la asignación (AA).

$$\begin{aligned}
\varphi_2 \ &\equiv \ def(\gamma(i)) \ \wedge \ INV_w^{\gamma(i)} \\
&\equiv \ def(x \ mod \ A(i) = 0) \ \wedge \ \lambda \ \wedge \ (0 \leq i \leq n) \ \wedge \ (\gamma(i) \leftrightarrow \mu(i)) \\
&\equiv \ \underbrace{(1 \leq i \leq n)}_{A(i)} \wedge \underbrace{(A(i) \neq 0)}_{mod} \ \wedge \ \lambda \ \wedge \ (0 \leq i \leq n) \ \wedge \ (\gamma(i) \leftrightarrow \mu(i)) \\
&\equiv (A(i) \neq 0) \ \wedge \ \lambda \ \wedge \ (1 \leq i \leq n) \ \wedge \ (\gamma(i) \leftrightarrow \mu(i))
\end{aligned}$$

En $def(x \ mod \ A(i) = 0)$, se han de considerar todos los posibles problemas que puedan surgir al calcular el valor de la expresión booleana $(x \ mod \ A(i) = 0)$. Al haber un acceso a vector, hay que exigir que el índice esté en el rango correcto. De ahí que se ponga $(1 \leq i \leq n)$. Además, al aparecer mod (el resto de la división entera), se ha de poner que el divisor no sea 0: $(A(i) \neq 0)$.

Para simplificar φ_2, se ha determinado el intervalo de i a partir de $(1 \leq i \leq n)$ y $(0 \leq i \leq n)$: se han cogido el mayor de los límites inferiores (1) y el menor de los límites superiores (n).

- Cálculo de la fórmula φ_3 a partir de la fórmula φ_2 utilizando el axioma de la asignación (AA).

$$\begin{aligned}\varphi_3 &\equiv \mathit{def}(i+1) \ \wedge \ (\varphi_2)_i^{i+1} \\ &\equiv \mathit{True} \ \wedge \ (A(i+1) \neq 0) \ \wedge \ \lambda \ \wedge \ (1 \leq i+1 \leq n) \ \wedge \ (\gamma(i+1) \leftrightarrow \mu(i+1)) \\ &\equiv (A(i+1) \neq 0) \ \wedge \ \lambda \ \wedge \ (1 \leq i+1 \leq n) \ \wedge \ (\gamma(i+1) \leftrightarrow \mu(i+1)) \\ &\equiv (A(i+1) \neq 0) \ \wedge \ \lambda \ \wedge \ (0 \leq i \leq n-1) \ \wedge \ (\gamma(i+1) \leftrightarrow \mu(i+1))\end{aligned}$$

En la espresión numérica $(i+1)$, no hay posibilidad de que surja ningún problema a la hora de calcularla: ni división por 0, ni acceso a vector con índice fuera de rango, ni ningún otro problema. Por eso, $\mathit{def}(i+1)$ es True.

Para simplificar la fórmula φ_3, se ha tenido en cuenta que $(\mathit{True} \ \wedge \ \delta) \equiv \delta$ para cualquier fórmula δ. Por otra parte, se ha transformado $(1 \leq i+1 \leq n)$ en $(0 \leq i \leq n-1)$, restando 1 en los tres componentes.

- Comprobación de la implicación: ¿$(INV \ \wedge \ B) \rightarrow \varphi_3$?

$$¿\underbrace{\lambda \ \wedge \ (0 \leq i \leq n) \ \wedge \ (w \leftrightarrow \mu(i))}_{INV} \ \wedge \ \underbrace{(i \neq n) \ \wedge \ \neg w}_{B} \rightarrow$$

$$\rightarrow \underbrace{(A(i+1) \neq 0) \ \wedge \ \lambda \ \wedge \ (0 \leq i \leq n-1) \ \wedge \ (\gamma(i+1) \leftrightarrow \mu(i+1))}_{\varphi_3}?$$

En la primera parte (parte izquierda o primera línea) de la implicación tenemos la información:

$$\lambda \ \wedge \ (0 \leq i \leq n) \ \wedge \ (w \leftrightarrow \mu(i)) \ \wedge \ (i \neq n) \ \wedge \ \neg w$$

Si descomponemos $(0 \leq i \leq n)$, nos queda:

$$\underbrace{\lambda}_{\alpha_1} \ \wedge \ \underbrace{(0 \leq i)}_{\alpha_2} \ \wedge \ \underbrace{(i \leq n)}_{\alpha_3} \ \wedge \ \underbrace{(w \leftrightarrow \mu(i))}_{\alpha_4} \ \wedge \ \underbrace{(i \neq n)}_{\alpha_5} \ \wedge \ \underbrace{\neg w}_{\alpha_6}$$

En la segunda parte (parte derecha o segunda línea) tenemos cinco preguntas: ¿$(A(i+1) \neq 0)$? ¿λ? ¿$0 \leq i$? ¿$i \leq n-1$? ¿$\gamma(i+1) \leftrightarrow \mu(i+1)$?

  - ¿$(A(i+1) \neq 0)$? Por α_2, α_3 y α_5 sabemos que $(0 \leq i \leq n-1)$. Consecuentemente, se cumple $(0+1 \leq i+1 \leq n-1+1)$, es decir, $(1 \leq i+1 \leq n)$. Esto significa que el índice $i+1$ está en el rango del vector, que es $1..n$. Por otra parte, por α_1, es decir, por λ sabemos que se cumple $\mathit{positmenorig}(x, A(1..n))$, lo cual significa que todos los elementos de $A(1..n)$ son positivos. Por tanto, sabemos que $A(i+1)$ es positivo, y por consiguiente, no es 0.
  - ¿λ? Sí, por α_1.
  - ¿$0 \leq i$? Sí, por α_2.
  - ¿$i \leq n-1$? Sí, por α_3 y α_5.
  - ¿$\gamma(i+1) \leftrightarrow \mu(i+1)$? Considerando el significado de $\mu(i+1)$, la pregunta queda de la siguiente forma:

$$¿\gamma(i+1) \leftrightarrow \mathit{algun\_divisor}(x, A(1..n), i+1)?$$

    Considerando la definición de $\mathit{algun\_divisor}(x, A(1..n), i+1)$, la pregunta es la siguiente:

$$¿\gamma(i+1) \leftrightarrow \exists k(1 \leq k \leq i+1 \ \wedge \ x \ mod \ A(k) = 0)?$$

    Si utilizamos abreviaturas, la pregunta es la siguiente:

$$¿\gamma(i+1) \leftrightarrow \exists k(1 \leq k \leq i+1 \ \wedge \ \gamma(k))?$$

Las fórmulas existenciales representan una disyunción. En este caso, la pregunta queda de la siguiente manera:

$$¿\gamma(i+1) \leftrightarrow (\gamma(1) \ \vee \ \gamma(2) \ \vee \cdots \vee \ \gamma(i) \vee \ \gamma(i+1))?$$

Esta pregunta no puede ser respondida directamente, porque depende del valor de $\gamma(1) \ \vee \ \gamma(2) \ \vee \cdots \vee \ \gamma(i)$. En α_4 se nos dice que $w \leftrightarrow \mu(i)$, es decir,

$$w \leftrightarrow \exists k(1 \leq k \leq i \ \wedge \ x \ mod \ A(k) = 0)$$

Si utilizamos abreviaturas, tenemos lo siguiente:

$$w \leftrightarrow \exists k(1 \leq k \leq i \ \wedge \ \gamma(k))$$

Utilizando la disyunción, tenemos que $w \leftrightarrow (\gamma(1) \ \vee \ \gamma(2) \ \vee \cdots \vee \ \gamma(i))$. Por su parte, en α_6 se nos dice que w es $False$. De ello deducimos que $\gamma(1) \ \vee \ \gamma(2) \ \vee \cdots \vee \ \gamma(i)$ es $False$. Por tanto, la pregunta queda de la siguiente forma: $¿\gamma(i+1) \leftrightarrow (False \ \vee \ \gamma(i+1))?$ Por equivalencia lógica, sabemos que $(False \ \vee \ \delta) \equiv \delta$ para cualquier fórmula δ. Entonces, la pregunta es ahora la que sigue: $¿\gamma(i+1) \ \leftrightarrow \ \gamma(i+1)?$ La respuesta es afirmativa porque $\delta \ \leftrightarrow \ \delta$ es $True$ para cualquier fórmula δ.

Hemos comprobado que la implicación $(INV \ \wedge \ B) \rightarrow \varphi_3$ se cumple.

1.2.6 *(f)* Punto (IV) de la regla del while

$¿(INV \ \wedge \ \neg B) \rightarrow \psi?$

$$¿\underbrace{\lambda \ \wedge \ (0 \leq i \leq n) \ \wedge \ (w \leftrightarrow \mu(i))}_{INV} \wedge \underbrace{((i = n) \ \vee \ w)}_{\neg B} \rightarrow$$

$$\rightarrow \underbrace{(w \leftrightarrow \exists k(1 \leq k \leq n \ \wedge \ pot\_dos(A(k))))}_{\psi}?$$

En la primera parte (parte izquierda o primera línea) de la implicación tenemos la información:

$$\lambda \ \wedge \ (0 \leq i \leq n) \ \wedge \ (w \leftrightarrow \mu(i)) \ \wedge \ ((i = n) \ \vee \ w)$$

Si descomponemos $(0 \leq i \leq n)$, nos queda:

$$\underbrace{\lambda}_{\alpha_1} \wedge \underbrace{(0 \leq i)}_{\alpha_2} \wedge \underbrace{(i \leq n)}_{\alpha_3} \wedge \underbrace{(w \leftrightarrow \mu(i))}_{\alpha_4} \wedge (\underbrace{(i = n)}_{\alpha_5} \vee \underbrace{w}_{\alpha_6})$$

En la segunda parte (parte derecha o segunda línea) tenemos una pregunta:

$$¿w \leftrightarrow \exists k(1 \leq k \leq n \ \wedge \ pot\_dos(A(k)))?$$

Si utilizamos abreviaturas, la pregunta es la siguiente:

$$¿w \leftrightarrow \exists k(1 \leq k \leq n \ \wedge \ \sigma(k))?$$

Si sustituimos la fórmula existencial por la disyunción, tenemos la siguiente pregunta:

$$¿w \leftrightarrow (\sigma(1) \ \vee \ \sigma(2) \ \vee \cdots \vee \ \sigma(n))? \tag{1}$$

Por α_4 sabemos que se cumple lo siguiente:

$$w \leftrightarrow (\gamma(1) \ \vee \ \gamma(2) \ \vee \cdots \vee \ \gamma(i)) \tag{2}$$

Por α_3 sabemos que $i \leq n$. Por tanto, el número de gammas (γ) es menor o igual que el número de sigmas (σ). Por otra parte, nos encontramos con que en α_4 tenemos información sobre las gammas pero en ψ

necesitamos información sobre las sigmas. En principio, las gammas y las sigmas son expresiones distintas y su valor no tiene porqué coincidir. Pero en α_1, es decir, en λ, se nos dice que x es una potencia entera de 2 y que los valores del vector $A(1..n)$ son todos menores o iguales que x y positivos. Bajo esas circunstancias, se puede aplicar la propiedad $(Prop)$ de la figura 2 (página 4) y tenemos que el valor de cada $\gamma(\ell)$ es igual al valor del correspondiente $\sigma(\ell)$, siendo $1 \leq \ell \leq i$:

$$\begin{array}{l} \gamma(1) = \sigma(1) \\ \gamma(2) = \sigma(2) \\ \cdots \\ \gamma(i) = \sigma(i) \end{array} \tag{3}$$

Y, por consiguiente:

$$(\gamma(1) \ \vee \ \gamma(2) \ \vee \cdots \vee \ \gamma(i)) = (\sigma(1) \ \vee \ \sigma(2) \ \vee \cdots \vee \ \sigma(i)) \tag{4}$$

Aunque sabemos que se cumple $w \leftrightarrow (\gamma(1) \ \vee \ \gamma(2) \ \vee \cdots \vee \ \gamma(i))$, de momento no sabemos el valor de w ni el de $\gamma(1) \ \vee \ \gamma(2) \ \vee \cdots \vee \ \gamma(i)$. Para obtener más información, se ha de recurrir a la disyunción $\alpha_5 \vee \alpha_6$. Al ser $\alpha_5 \vee \alpha_6$ una disyunción, puede ser cierta porque se cumplen tanto α_5 como α_6 o porque se cumple solo α_5 o porque se cumple solo α_6. En total hay tres opciones. Esas opciones se muestran en la tabla 4 (página 9).

	α_5	α_6
	$i = n$	w
1	True	True
2	True	False
3	False	True

Tabla 4: Las tres opciones para que la expresión $((i = n) \ \vee \ w)$ sea cierta.

1.2.6.1 Casos 1 y 2: $i = n$

Si se cumple $i = n$, la ecuación (4) de la página 9 se puede reescribir de la siguiente forma:

$$\underbrace{\gamma(1) \ \wedge \ \gamma(2) \ \wedge \cdots \wedge \ \gamma(n)}_{\delta_2} = \underbrace{\sigma(1) \ \wedge \ \sigma(2) \ \wedge \cdots \wedge \ \sigma(n)}_{\delta_3} \tag{5}$$

La equivalencia (2) de la página 8 se puede reescribir de la siguiente forma:

$$\underbrace{w}_{\delta_1} \leftrightarrow \underbrace{(\gamma(1) \ \wedge \ \gamma(2) \ \wedge \cdots \wedge \ \gamma(n))}_{\delta_2} \tag{6}$$

Recordemos que tenemos que responder a la pregunta (1) planteada en la página 8:

$$¿ \underbrace{w}_{\delta_1} \leftrightarrow \underbrace{\sigma(1) \ \wedge \ \sigma(2) \ \wedge \cdots \wedge \ \sigma(n)}_{\delta_3} ?$$

La respuesta es afirmativa por la igualdad (5) (página 9) y la equivalencia (6) (página 9): w es equivalente a $\gamma(1) \ \vee \ \gamma(2) \ \vee \cdots \vee \ \gamma(n)$ y, a su vez, $\gamma(1) \ \vee \ \gamma(2) \ \vee \cdots \vee \ \gamma(n)$ es equivalente a $\sigma(1) \ \vee \ \sigma(2) \ \vee \cdots \vee \ \sigma(n)$. Por tanto, w es equivalente a $\sigma(1) \ \vee \ \sigma(2) \ \vee \cdots \vee \ \sigma(n)$.

Ahí se ha utilizado la propiedad transitiva de $\leftrightarrow$:

Si $\delta_1 \leftrightarrow \delta_2$ y $\delta_2 \leftrightarrow \delta_3$, entonces $\delta_1 \leftrightarrow \delta_3$

Nótese que en esta deducción no nos hace falta conocer el valor de w. Nos da igual que sea $True$ o $False$. De ahí que se hayan podido analizar juntos los casos 1 y 2 de la tabla 4 de la página 9.

1.2.6.2 Caso 3: $i \neq n$ y $w = True$

Si se cumple $i \neq n$, entonces por α_3 deducimos que $i < n$. Consecuentemente, la pregunta (1) de la página 8 se puede reescribir de la siguiente forma:

$$¿w \leftrightarrow (\sigma(1) \ \vee \ \sigma(2) \ \vee \cdots \vee \ \sigma(i) \ \vee \ \sigma(i+1) \ \vee \cdots \vee \ \sigma(n))? \tag{7}$$

Podemos observar que hay más sigmas que gammas. En concreto, hay n sigmas e i gammas, siendo $i < n$. Sabemos que se cumple la igualdad (4) (página 9) pero no tenemos ninguna información sobre la disyunción $\sigma(i+1) \ \vee \cdots \vee \ \sigma(n)$.

Puesto que estamos considerando el caso en el que $i \neq n$ y $w = True$, por la equivalencia (2) de la página 8 deducimos que $\gamma(1) \ \vee \ \gamma(2) \ \vee \cdots \vee \ \gamma(i)$ es $True$. Por la igualdad (4) de la página 9, deducimos que $\sigma(1) \ \vee \ \sigma(2) \ \vee \cdots \vee \ \sigma(i)$ es $True$. Por tanto, la pregunta (7) de la página 10 se puede reescribir de la siguiente forma:

$$¿True \leftrightarrow (True \ \vee \ \sigma(i+1) \ \vee \cdots \vee \ \sigma(n))? \tag{8}$$

Utilizando la equivalencia lógica $(True \ \vee \ \delta) \equiv True$, la pregunta (8) de la página 10 se puede reescribir de la siguiente forma:

$$¿True \leftrightarrow True?$$

La respuesta a esa última pregunta es afirmativa porque $\delta \leftrightarrow \delta$ es $True$ para cualquier fórmula δ.

Nótese que en la deducción correspondiente al caso $i \neq n$ y $w = True$, sí nos hace falta conocer el valor de w.

1.2.7 *(g)* Punto (V) de la regla del while

$¿(INV \ \wedge \ B) \rightarrow (E > 0)?$

$$¿\underbrace{\lambda \ \wedge \ (0 \leq i \leq n) \ \wedge \ (w \leftrightarrow \mu(i))}_{INV} \wedge \underbrace{(i \neq n) \ \wedge \ \neg w}_{B} \rightarrow \underbrace{(n - i > 0)}_{E > 0}?$$

En la primera parte (parte izquierda) de la implicación tenemos la información:

$$\lambda \ \wedge \ (0 \leq i \leq n) \ \wedge \ (w \leftrightarrow \mu(i)) \ \wedge \ (i \neq n) \ \wedge \ \neg w$$

Si descomponemos $(0 \leq i \leq n)$, nos queda:

$$\underbrace{\lambda}_{\beta_1} \wedge \underbrace{(0 \leq i)}_{\beta_2} \wedge \underbrace{(i \leq n)}_{\beta_3} \wedge \underbrace{(w \leftrightarrow \mu(i))}_{\beta_4} \wedge \underbrace{(i \neq n)}_{\beta_5} \wedge \underbrace{\neg w}_{\beta_6}$$

En la segunda parte (parte derecha) tenemos una pregunta:

$$¿n - i > 0?$$

Por β_3 y β_5 deducimos que $n > i$. Restando i en ambos lados, obtenemos $n - i > i - i$, es decir, $n - i > 0$. Y eso es lo que queríamos obtener.

1.2.8 *(h)* Punto (VI) de la regla del while

- Cálculo de la fórmula φ_4 a partir de la fórmula $E < v$ utilizando el axioma de la asignación (AA).

$$\begin{aligned}
\varphi_4 \ &\equiv \ def(\gamma(i)) \ \wedge \ (E < v)_w^{\gamma(i)} \\
&\equiv \ def(x \ mod \ A(i) = 0) \ \wedge \ (n - i < v)_w^{\gamma(i)} \\
&\equiv \ \underbrace{(1 \leq i \leq n)}_{A(i)} \wedge \underbrace{(A(i) \neq 0)}_{mod} \ \wedge \ (n - i < v)
\end{aligned}$$

Tal como se ha explicado al calcular φ_2, en $def(x \ mod \ A(i) = 0)$, se han de considerar todos los posibles problemas que puedan surgir al calcular el valor de la expresión booleana $(x \ mod \ A(i) = 0)$. Al haber un acceso a vector, hay que exigir que el índice esté en el rango correcto. De ahí que se ponga $(1 \leq i \leq n)$. Además, al aparecer mod (el resto de la división entera), se ha de poner que el divisor no sea 0: $(A(i) \neq 0)$.

φ_4 no se puede simplificar.

- Cálculo de la fórmula φ_5 a partir de la fórmula φ_4 utilizando el axioma de la asignación (AA).

$$\begin{aligned}\varphi_5 &\equiv def(i+1) \ \wedge \ (\varphi_4)_i^{i+1} \\ &\equiv True \ \wedge \ (1 \leq i+1 \leq n) \ \wedge \ A(i+1) \neq 0 \ \wedge \ (n-(i+1) < v) \\ &\equiv (0 \leq i \leq n-1) \ \wedge \ A(i+1) \neq 0 \ \wedge \ (n-i-1 < v) \\ &\equiv (0 \leq i) \ \wedge \ (i \leq n-1) \ \wedge \ A(i+1) \neq 0 \ \wedge \ (n-i-1 < v)\end{aligned}$$

Puesto que al calcular $i+1$ no puede surgir ningún problema de división por cero o acceso a vector con un índice que esté fuera de rango, el valor de $def(i+1)$ es $True$.

Para simplificar φ_5, se ha tenido en cuenta que $True \ \wedge \ \delta \equiv \delta$ para cualquier fórmula δ. Por otra parte, se ha transformado $(1 \leq i+1 \leq n)$ en $(0 \leq i \leq n-1)$, restando 1 a los tres componentes de la expresión. También, se ha transformado $(n-(i+1) < v)$ en $(n-i-1 < v)$, aplicando el signo negativo a $(i+1)$. Finalmente, se ha separado $(0 \leq i \leq n-1)$ en dos trozos: $(0 \leq i)$ y $(i \leq n-1)$.

- Comprobación de la implicación: ¿$(INV \ \wedge \ B \ \wedge \ E = v) \rightarrow \varphi_5$?

$$¿\underbrace{\lambda \ \wedge \ (0 \leq i \leq n) \ \wedge \ (w \leftrightarrow \mu(i))}_{INV} \wedge \underbrace{(i \neq n) \ \wedge \ \neg w}_{B} \wedge \underbrace{(n-i=v)}_{E=v} \rightarrow$$

$$\rightarrow \underbrace{(0 \leq i) \ \wedge \ (i \leq n-1) \ \wedge \ A(i+1) \neq 0 \ \wedge \ (n-i-1 < v)}_{\varphi_5}?$$

En la primera parte (parte izquierda o primera línea) de la implicación tenemos la información:

$$\lambda \ \wedge \ (0 \leq i \leq n) \ \wedge \ (w \leftrightarrow \mu(i)) \ \wedge \ (i \neq n) \ \wedge \ \neg w \ \wedge \ (n-i=v)$$

Si descomponemos $(0 \leq i \leq n)$, nos queda:

$$\underbrace{\lambda}_{\alpha_1} \wedge \underbrace{(0 \leq i)}_{\alpha_2} \wedge \underbrace{(i \leq n)}_{\alpha_3} \wedge \underbrace{(w \leftrightarrow \mu(i))}_{\alpha_4} \wedge \underbrace{(i \neq n)}_{\alpha_5} \wedge \underbrace{\neg w}_{\alpha_6} \wedge \underbrace{(n-i=v)}_{\alpha_7}$$

En la segunda parte (parte derecha o segunda línea) tenemos cuatro preguntas: ¿$0 \leq i$? ¿$i \leq n-1$? ¿$A(i+1) \neq 0$? ¿$n-i-1 < v$?

- ¿$0 \leq i$? Sí, por α_2.
- ¿$i \leq n-1$? Sí, por α_3 y α_5.
- ¿$(A(i+1) \neq 0)$? Por α_2, α_3 y α_5 sabemos que $(0 \leq i \leq n-1)$. Consecuentemente, se cumple $(0+1 \leq i+1 \leq n-1+1)$, es decir, $(1 \leq i+1 \leq n)$. Esto significa que el índice $i+1$ está en el rango del vector, que es $1..n$. Por otra parte, por α_1, es decir, por λ sabemos que se cumple $positmenorig(x, A(1..n))$, lo cual significa que todos los elementos de $A(1..n)$ son positivos. Por tanto, sabemos que $A(i+1)$ es positivo, y por consiguiente, no es 0.
- ¿$n-i-1 < v$? Por α_7 sabemos que $(n-i=v)$. Si restamos 1 a ambos lados de la igualdad, nos queda $(n-i-1=v-1)$. Puesto que para cualquier número entero z se cumple $z-1 < z$, también se cumple $v-1 < v$. Por tanto, deducimos que $(n-i-1 < v)$.

Hemos comprobado que la implicación $(INV \ \wedge \ B \ \wedge \ E = v) \rightarrow \varphi_5$ se cumple.

1.2.9 *(i)* Demostración formal de la corrección

En la tabla 5 (página 13) se muestra la demostración formal de la corrección total del programa de la figura 1 (página 4) . En esa demostración, P1 y P2 se refieren a los dos programas obtenidos en la figura 4 (página 13) tras la partición en dos del programa original (figura 1, página 4).

Los primeros tres pasos (pasos 1-3) de la demostración formal corresponden al programa 1 (P1) de la figura 4 (página 13). En la tabla 6 (página 15), se indica el bloque que corresponde a cada uno de esos pasos. Los pasos del 4 al 18 corresponden al programa 2 (P2) de la figura 4 (página 13). En concreto, cada punto de la regla del while está representado mediante uno o varios pasos de esa demostración formal. Los puntos (I), (II), (IV) y (V) de la regla del while requieren un único paso (los pasos 4, 5, 11 y 12, respectivamente). En cambio, tanto el punto (III) como el (VI) requieren varios pasos en la demostración formal. El punto (III) abarca los pasos 6-10, mientras que, por su parte, el punto (VI) abarca los pasos 13-17. Una manera sistemática de obtener esos pasos, es considerar bloques en los esquemas de los puntos (III) y (VI). En las tablas 7 (página 15) y 8 (página 15) se muestran los bloques a considerar. Puesto que se tienen las pruebas de corrección de los seis puntos de la regla de while, en el paso 18 se afirma que el programa 2 de la figura 4 (página 13) es correcto. Para ello se menciona la regla del while (RWH) y el último paso correspondiente a cada punto de la regla del while: pasos 4, 5, 10, 11, 12 y 17. Finalmente, considerando los pasos 3 y 18, se puede afirmar, en el paso 19, que todo el programa de la figura 1 (página 4) es correcto.

En esa demostración formal, cada vez que se han tenido que juntar una implicación y un programa (pasos 3, 8 y 15), se ha utilizado la regla de la consecuencia (RCN). En cambio, cada vez que se han tenido que juntar dos programas (pasos 10, 17 y 19), se ha utilizado la regla de la composición (RCP).

P1 (Programa 1):

$\{\varphi\}$
w := False;
$\{INV\}$

P2 (Programa 2):

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

Figura 4: Los dos programas a verificar tras la partición.

Demostración formal de la corrección total			
P1		1.	$\varphi \rightarrow \varphi_1$
		2.	$\{\varphi_1\}$ w := False; $\{INV\}$ (AA)
Fin P1		3.	$\{\varphi\}$ w := False; $\{INV\}$ (RCN 1, 2)
P2	(I)	4.	$INV \rightarrow INV$
	(II)	5.	$INV \rightarrow def(B)$
	(III)	6.	$(INV \wedge B) \rightarrow \varphi_3$
		7.	$\{\varphi_3\}$ i := i + 1; $\{\varphi_2\}$ (AA)
		8.	$\{INV \wedge B\}$ i := i + 1; $\{\varphi_2\}$ (RCN 6, 7)
		9.	$\{\varphi_2\}$ w := $\gamma(i)$; $\{INV\}$ (AA)
	Fin (III)	10.	$\{INV \wedge B\}$ i := i + 1; w := $\gamma(i)$; $\{INV\}$ (RCP 8, 9)
	(IV)	11.	$(INV \wedge \neg B) \rightarrow \psi$
	(V)	12.	$(INV \wedge B) \rightarrow (E > 0)$
	(VI)	13.	$(INV \wedge B \wedge E = v) \rightarrow \varphi_5$
		14.	$\{\varphi_5\}$ i := i + 1; $\{\varphi_4\}$ (AA)
		15.	$\{INV \wedge B \wedge E = v\}$ i := i + 1; $\{\varphi_4\}$ (RCN 13, 14)
		16.	$\{\varphi_4\}$ w := $\gamma(i)$; $\{E < v\}$ (AA)
	Fin (VI)	17.	$\{INV \wedge B \wedge E = v\}$ i := i + 1; w := $\gamma(i)$; $\{E < v\}$ (RCP 15, 16)
		18.	$\{INV\}$ while $\{INV\}$ $\{E\}$ B loop i := i + 1; w := $\gamma(i)$; end loop;
Fin P2			$\{\psi\}$ (RWH 4, 5, 10, 11, 12, 17)

P1;P2		19.	$\{\varphi\}$
			w := False;
			while $\{INV\}$ $\{E\}$ B loop
			i := i + 1;
			w := $\gamma(i)$;
			end loop;

			$\{\psi\}$ (RCP 3, 18)

Tabla 5: Demostración formal de la corrección total del programa de la figura 1 (página 4).

Esquema para el programa 1:

$\{\varphi\}$ $\quad ¿\varphi \rightarrow \varphi_1?$

$\{\varphi_1 \equiv def(False) \ \wedge \ INV_w^{False}\}$

(AA) w := False;

$\{INV\}$

Esquema para el programa 2:

(I) $\quad ¿INV \rightarrow INV?$

(II) $\quad ¿INV \rightarrow def(B)?$

(III) $\quad \{INV \ \wedge \ B\}$ $\quad ¿(INV \ \wedge \ B) \rightarrow \varphi_3?$

$\{\varphi_3 \equiv def(i+1) \ \wedge \ (\varphi_2)_i^{i+1}\}$

(AA) i := i + 1;

$\{\varphi_2 \equiv def(\gamma(i)) \ \wedge \ INV_w^{\gamma(i)}\}$

(AA) w := $\gamma(i)$;

$\{INV\}$

(IV) $\quad ¿(INV \ \wedge \ \neg B) \rightarrow \psi?$

(V) $\quad ¿(INV \ \wedge \ B) \rightarrow E > 0?$

(VI) $\quad \{INV \ \wedge \ B \ \wedge \ E = v\}$ $\quad ¿(INV \ \wedge \ B \ \wedge \ E = v) \rightarrow \varphi_5?$

$\{\varphi_5 \equiv def(i+1) \ \wedge \ (\varphi_4)_i^{i+1}\}$

(AA) i := i + 1;

$\{\varphi_4 \equiv def(\gamma(i)) \ \wedge \ (E < v)_w^{\gamma(i)}\}$

(AA) w := $\gamma(i)$;

$\{E < v\}$

donde $\gamma(i)$ representa la expresión booleana $(x \ mod \ A(i) = 0)$.

Figura 5: Esquemas para el programa 1 y para el programa 2 de la figura 4 (página 13).

$P1$

	1.	$\{\varphi\}$ $\{\varphi_1 \equiv def(False) \ \wedge \ INV_w^{False}\}$
3.		
	2.	$\{\varphi_1 \equiv def(False) \ \wedge \ INV_w^{False}\}$ w := $False$; $\{INV\}$

Tabla 6: Bloques a considerar en la demostración formal con respecto al programa 1 (ver figura 5 en la página 14 y tabla 5 en la página 13).

(III)

		6.	$\{INV \ \wedge \ B\}$ $\{\varphi_3 \equiv def(i+1) \ \wedge \ (\varphi_2)_i^{i+1}\}$
	8.		
		7.	$\{\varphi_3 \equiv def(i+1) \ \wedge \ (\varphi_2)_i^{i+1}\}$ i := $i+1$; $\{\varphi_2 \equiv def(\gamma(i)) \ \wedge \ INV_w^{\gamma(i)}\}$
10.			
		9.	$\{\varphi_2 \equiv def(\gamma(i)) \ \wedge \ INV_w^{\gamma(i)}\}$ w := $\gamma(i)$; $\{INV\}$

Tabla 7: Bloques a considerar en la demostración formal con respecto al punto (III) (ver figura 5 en la página 14 y tabla 5 en la página 13).

(VI)

		13.	$\{INV \ \wedge \ B \ \wedge \ E = v\}$ $\{\varphi_5 \equiv def(i+1) \ \wedge \ (\varphi_4)_i^{i+1}\}$
	15.		
		14.	$\{\varphi_5 \equiv def(i+1) \ \wedge \ (\varphi_4)_i^{i+1}\}$ i := i + 1; $\{\varphi_4 \equiv def(\gamma(i)) \ \wedge \ (E < v)_w^{\gamma(i)}\}$
17.			
		16.	$\{\varphi_4 \equiv def(\gamma(i)) \ \wedge \ (E < v)_w^{\gamma(i)}\}$ w := $\gamma(i)$; $\{E < v\}$

Tabla 8: Bloques a considerar en la demostración formal con respecto al punto (VI) (ver figura 5 en la página 14 y tabla 5 en la página 13).

Letras griegas adicionales utilizadas en el apartado correspondiente a la solución:
α : alfa β : beta δ : delta

Tabla 9: Denominaciones de las letras griegas adicionales utilizadas en el apartado correspondiente a la solución.