Introducción a la Computación (Matemática)

Primer Cuatrimestre de 2019

Verificación Formal de Algoritmos

Especificación, algoritmo y programa

Especificación de un problema:

- ¿Qué problema tenemos?
- Lenguaje formal (ej. lógica de primer orden).

Algoritmo: Una solución abstracta del problema (escrita para humanos).

- ¿Cómo resolvemos el problema?
- Pseudocódigo.

Programa: Una solución concreta del problema (escrita para computadoras).

- ¿Cómo resuelve la computadora el problema?
- Lenguaje de programación (ej. C++, Python).

Especificación de problemas

Una **especificación** tiene 3 partes:

1. Encabezado
   Indica el nombre, el tipo y número de los parámetros, y el tipo del valor de retorno (RV).
2. Precondición
   Es una descripción del valor inicial de los parámetros de entrada.
3. Poscondición
   Es una descripción del valor final de los parámetros de entrada y del valor de retorno.

Si la entrada es aceptable, el algoritmo debe terminar exitosamente y la salida debe cumplir las propiedades especificadas.

Ejemplo de especificación

Encabezado: $\mathit{sumarPares} : A \in \mathbb{Z}[] \to \mathbb{Z}$
Precondición: $\{A = A_0\}$
Poscondición: $\{RV = \sum_{0 \leq i < |A_0| \land Par(A_0[i])} A_0[i]\}$

donde $Par(n) \equiv (\exists k)\ n = 2 * k$

Obs: Si no se especifica el tipo de una variable, por defecto es $\mathbb{Z}$.

Lo siguiente es pensar un algoritmo que satisfaga esta especificación. Lo escribimos usando pseudocógido.

Pseudocódigo

C++	**Pseudocódigo**
a = b;	a $\leftarrow$ b
if (cond) {..} else {..}	if (cond) {..} else {..}
while (cond) {..}	while (cond) {..}
`!` `&&` `||`	$\neg$ $\wedge$ $\vee$
`==` `>=` `<=`	$=$ $\geq$ $\leq$
`%` `/`	mod div
return exp;	*RV* $\leftarrow$ *exp* (*)

(*) En C++, "return exp;" termina la ejecución de la función. En nuestro pseudocógido, "*RV* $\leftarrow$ *exp*" es una simple asignación, tras la cual el algoritmo continúa.

Ejemplo de algoritmo en pseudocógido

Encabezado: $sumarPares : A \in \mathbb{Z}[] \to \mathbb{Z}$
Precondición: $\{A = A_0\}$
Poscondición: $\{RV = \sum_{0 \leq i < |A_0| \wedge Par(A_0[i])} A_0[i]\}$

$RV \leftarrow 0$
$i \leftarrow 0$
while $(i < |A|)$ {
  if $(A[i] \bmod 2 = 0)$ {
    $RV \leftarrow RV + A[i]$
  }
  $i \leftarrow i + 1$
}

donde $i \in \mathbb{Z}$.

¿Cómo sabemos si este algoritmo es correcto respecto de la especificación?

Testing vs. Verificación formal de algoritmos

Sean E una especificación de un problema y S un algoritmo.

Testing de S: Experimentación de la corrida de S para un conjunto finito de datos de entrada, donde verificamos si cumple E.

Verificación de S respecto de E: Demostración formal de que el algoritmo S transforma todos los posibles datos de entrada, en salidas de acuerdo con E.

Verificación formal de algoritmos

Terna de Hoare

{precondición}	$\{P\}$
algoritmo	S
{poscondición}	$\{Q\}$

S modifica al estado P, ¿pero lleva a Q?

Una forma de probarlo: Ver que la poscondición más fuerte de ejecutar S a partir de P, implica Q.

$$¿sp(S, P) \Rightarrow Q?$$

$sp(S, P)$ (*strongest postcondition*) es el predicado más fuerte que resulta de ejecutar S a partir del estado P.

Asignación

$\{P\}\ x \leftarrow E\ \{Q\}$

Queremos probar que $sp(x \leftarrow E,\ P) \Rightarrow Q$.

Definición:

$$sp(x \leftarrow E,\ P) \equiv (\exists v)\ x = E[x:v] \wedge P[x:v]$$

donde:

- v es una variable no usada;
- $H[x:E]$ es la sustitución de cada instancia en H de la variable x por la expresión E.
  Ejemplo: $(x+y)[y:z^2+1] = (x+z^2+1)$.

Asignación: Ejemplo

Probar que el siguiente algoritmo es correcto respecto de su especificación:

$$\begin{gathered} P \equiv \{x > 1\} \\ x \leftarrow x + 1 \\ Q \equiv \{x > 2\} \end{gathered}$$

$$\begin{aligned} & sp(x \leftarrow x + 1, x > 1) \equiv \\ & \quad \equiv (\exists v)\ x = (x + 1)[x : v] \wedge (x > 1)[x : v] \\ & \quad \equiv (\exists v)\ x = v + 1 \wedge v > 1 \\ & \quad \Rightarrow x > 2 \end{aligned}$$

Secuencialización

$\{P\}\ S_1;S_2\ \{Q\}$

Queremos probar que $sp(S_1;S_2,\ P) \Rightarrow Q$.

Definición:

$$sp(S_1;S_2,\ P) \equiv sp(S_2, sp(S_1, P))$$

Secuencialización: Ejemplo

$$P \equiv \{x > 1\}$$
$$x \leftarrow x + 1$$
$$y \leftarrow 2 * x$$
$$Q \equiv \{y > 4\}$$

$$\begin{aligned}
& sp(x \leftarrow x+1; y \leftarrow 2*x, P) \equiv \\
& \quad \equiv sp(y \leftarrow 2*x, sp(x \leftarrow x+1, P)) \\
& \quad \equiv sp(y \leftarrow 2*x, (\exists a)\ x = (x+1)[x:a] \wedge (x > 1)[x:a]) \\
& \quad \equiv sp(y \leftarrow 2*x, (\exists a)\ x = a+1 \wedge a > 1) \\
& \quad \equiv (\exists b)\ y = (2*x)[y:b] \wedge ((\exists a)\ x = a+1 \wedge a > 1)[y:b] \\
& \quad \equiv (\exists b)\ y = 2*x \wedge (\exists a)\ x = a+1 \wedge a > 1 \\
& \quad \equiv y = 2*x \wedge (\exists a)\ x = a+1 \wedge a > 1 \\
& \quad \Rightarrow y = 2*x \wedge x > 2 \\
& \quad \Rightarrow y > 4
\end{aligned}$$

Condicional

$\{P\}$ if (B) S_1 else S_2 $\{Q\}$

Queremos probar que $sp(\text{if } (B)\ S_1 \text{ else } S_2,\ P) \Rightarrow Q$.

Definición:

$$sp(\text{if } (B)\ S_1 \text{ else } S_2,\ P) \equiv sp(S_1, B \wedge P) \vee sp(S_2, \neg B \wedge P)$$

¿Qué pasa si no hay "else"? Formalmente, if (B) S_1 equivale a if (B) S_1 else pass, donde pass es una instrucción especial que no tiene efecto:

$$sp(\text{pass},\ P) \equiv P$$

Por lo tanto:

$$sp(\text{if } (B)\ S_1,\ P) \equiv sp(S_1, B \wedge P) \vee (\neg B \wedge P)$$

Condicional: Ejemplo

$P \equiv \{a = a_0 \wedge b = b_0\}$
if $(a \leq b)$ {
  $RV \leftarrow 0$
} else {
  $RV \leftarrow a - b$
}
$Q \equiv \{RV = a_0 \dot{-} b_0\}$

donde $x \dot{-} y = \begin{cases} 0 & \text{si} \quad x \leq y \\ x - y & \text{si} \quad x > y \end{cases}$

Condicional: Ejemplo

$$
\begin{aligned}
&P \equiv \{a = a_0 \wedge b = b_0\} \\
&\text{if } (a \leq b) \ \{\ RV \leftarrow 0\ \} \text{ else } \{\ RV \leftarrow a - b\ \} \\
&Q \equiv \{RV = a_0 \dot{-} b_0\}
\end{aligned}
$$

$$
\begin{aligned}
&sp(\mathit{if}\,(a \leq b)\{RV \leftarrow 0\} \text{ else } \{RV \leftarrow a - b\}, P) \equiv \\
&\quad \equiv sp(RV \leftarrow 0, \qquad a \leq b \wedge P) \vee \\
&\qquad sp(RV \leftarrow a - b,\ a > b \wedge P) \\
&\quad \equiv ((\exists x)\, RV = 0[RV : x] \qquad \wedge(a \leq b \wedge P)[RV : x]) \vee \\
&\qquad ((\exists y)\, RV = (a - b)[RV : y] \ \wedge(a > b \wedge P)[RV : y]) \\
&\quad \equiv ((\exists x)\, RV = 0 \qquad \wedge(a \leq b \wedge P)) \vee \\
&\qquad ((\exists y)\, RV = (a - b) \ \wedge(a > b \wedge P)) \\
&\quad \equiv (RV = 0 \qquad \wedge(a \leq b \wedge a = a_0 \wedge b = b_0)) \vee \\
&\qquad (RV = (a - b) \ \wedge(a > b \wedge a = a_0 \wedge b = b_0)) \\
&\quad \Rightarrow RV = a_0 \dot{-} b_0
\end{aligned}
$$

Repaso de la clase de hoy

- Pseudocódigo
- Testing vs. Verificación formal de algoritmos
- Terna de Hoare; poscondición más fuerte (*sp*).
- Asignación, secuencialización, condicional.

Próximos temas

- Verificación formal de ciclos.