Diseño de algoritmos iterativos

Facultad de Informática - UCM

17 de octubre de 2020

Bibliografía Recomendada

- Algoritmos correctos y eficientes: Diseño razonado ilustrado con ejercicios. *Matí-Oliet, N.; Segura Diaz, C. M., Verdejo Lopez, A.*. Ibergarceta Publicaciones, 2012.
- Especificación, Derivación y Análisis de Algoritmos: ejercicios resueltos. *Narciso Martí Oliet, Clara María Segura Díaz y Jose Alberto Verdejo López*. Colección Prentice Práctica, Pearson Prentice-Hall, 2006

Capítulos 2, 3, y 4 de ambos libros.

- Complejidad de algoritmos iterativos:
  - Ejercicios resueltos: 3.21 a), 3.23 a), 3.25
  - Ejercicios propuestos: 3.12, 3.13
- Verificación de algoritmos iterativos
  - Ejercicios resueltos: 2.7, 2.8, 2.9, 2.13, 2.15
- Derivación de algoritmos iterativos
  - Ejercicios resueltos: 4.3, 4.4, 4.5, 4.6, 4.7, 4.8, 4.9, 4.10, 4.11, 4.12, 4.13, 4.14, 4.15, 4.22, 4.23, 4.24, 4.25, 4.27

Objetivos

- Reglas para comprobar que un programa es correcto respecto a su especificación (verificación).
- Cómo implementar un programa correcto por construcción (derivación).
- Ver soluciones de problemas iterativos típicos para conocer diversas formas de solución.

Verificación

Verificar es demostrar que las instrucciones de un algoritmo son correctas, es decir, que para una entrada válida (precondición) producen el resultado esperado (postcondición).

Ejemplo: Problema: Intercambiar el valor de dos variables

Especificación:

x,y : entero
$P : \{x = X \wedge y = Y\}$
Intercambiar
$Q : \{x = Y \wedge y = X\}$

¿ Cuales de los siguientes algoritmos son correctos?

aux = x;
x = y;
y = aux;
	x = y;
	y = x;
		x = x - y;
		y = x + y;
		x = y - x;
			x = x + y;
			y = x - y;
			x = y - x;

Verificación

- Para verificar un algoritmo se definen predicados intermedios entre cada instrucción elemental, llamados *aserciones* o *asertos*:

$$\{R_0\}A_1\{R_1\};\ldots;\{R_{n-1}\}A_n\{R_n\}$$

- Si para cada instrucción A_k se satisface $\{R_{k-1}\}A_k\{R_k\}$ y $P \Rightarrow R_0$ y $R_n \Rightarrow Q$ entonces se satisface $\{P\}A\{Q\}$.
- La semántica del lenguaje define para cada tipo de instrucción del lenguaje las reglas que determinan si se satisface $\{R_{k-1}\}A_k\{R_k\}$ (reglas de verificación).

Verificación

Axioma de asignación Para toda variable x, toda expresión válida del mismo tipo E y todo predicado R, la especificación:

$$\begin{array}{l} P : \{Dom(E) \wedge \{R_x^E\} \\ \quad \mathrm{x} = \mathrm{E} \\ Q : \{R\} \end{array}$$

es correcta.

P se denomina precondición más débil (*pmd*).

- $Dom(E)$ el conjunto de todos los estados en los que la expresión E está definida.
- R_x^E el predicado resultante de sustituir toda aparición de x por E en el predicado R.

Ejemplo: Suponiendo x entero determina la precondición más débil,P, que safisfaga la especificación:

P :	$Q_x^{x+2} : \{x+2 \geq 0\} \equiv \{x \geq -2\}$
$\mathrm{x} = \mathrm{x} + 2$	$\mathrm{x} = \mathrm{x} + 2 \quad \uparrow$
$Q : \{x \geq 0\}$	$Q : \{x \geq 0\}$

Verificación

Regla de inferencia de la asignación

La especificación $\{P\}\ \mathrm{x} = \mathrm{E}\ \{Q\}$ es correcta si $P \Rightarrow Q_x^E$.

Regla de inferencia de la composición secuencial

La especificación $\{P\}\ A_1; A_2\ \{Q\}$ es correcta si existe un predicado R tal que las especificaciones $\{P\}\ A_1\ \{R\}$ y $\{R\}\ A_2\ \{Q\}$ son correctas.

Ejemplo: Suponiendo x, y enteros, calcula el predicado P más débil que satisfaga la especificación:

P :	$R_2 \equiv R_{1_x}^{x*x} : \{x * x + 1 > 0\} \equiv \{true\}$
$\mathrm{x} = \mathrm{x} * \mathrm{x}$	$\mathrm{x} = \mathrm{x} * \mathrm{x} \quad \uparrow$
	$R_1 \equiv Q_x^{x+1} : \{x + 1 > 0\}$
$\mathrm{x} = \mathrm{x} + 1$	$\mathrm{x} = \mathrm{x} + 1 \quad \uparrow$
$Q : \{x > 0\}$	$Q : \{x > 0\}$

Regla de inferencia de la composición alternativa (I)

La especificación $\boxed{\begin{array}{l}\{P\}\\ \quad \texttt{if}\ (B)\ A_1\ \texttt{else}\ A_2;\\ \{Q\}\end{array}}$ es correcta si

las especificaciones $\boxed{\begin{array}{l}\{P \wedge B\}\\ \quad A_1\\ \{Q\}\end{array}}$ y $\boxed{\begin{array}{l}\{P \wedge \neg B\}\\ \quad A_2\\ \{Q\}\end{array}}$

son correctas. La *pmd* es $B \wedge pmd(A_1, Q)) \vee (\neg B \wedge pmd(A_2, Q))$

Ejemplo:

P :	$P \wedge x \geq 0 \Rightarrow R_1$	$P \wedge \neg(x \geq 0) \Rightarrow R_2$
if (x <= 0) y = x;	$R_1 \equiv Q_y^x : \{x = Y\}$	$R_2 \equiv Q_y^{-x} : \{-x = Y\}$
else y = -x;	y = x $\uparrow$	y = -x $\uparrow$
$Q : \{y = Y\}$	$Q : \{y = Y\}$	$Q : \{y = Y\}$

$pmd \equiv (x \geq 0 \wedge x = Y) \vee (\neg(x \geq 0) \wedge -x = Y)$

Regla de inferencia de la composición iterativa

La especificación $\boxed{\{P\}\ \texttt{while}\ (B)\ \texttt{do}\ A\ \{Q\}}$ es correcta si existe un predicado I que llamaremos invariante y una función t dependiente de las variables que intervienen en el proceso y que toma valores enteros, que llamaremos función cota, de forma que:

1. $P \Rightarrow I$
2. $\{I \wedge B\}\ A\ \{I\}$
3. $I \wedge \neg B \Rightarrow Q$
4. $I \wedge B \Rightarrow t > 0$
5. $\{I \wedge B \wedge t = T\}\ A\ \{t < T\}$

Existen muchos predicados invariantes, se ha de elegir uno que nos permita verificar las 5 condiciones de corrección del bucle, esto es

- Lo suficientemente fuerte para $I \wedge \neg B \Rightarrow Q$
- Lo suficientemente débil para $P \Rightarrow I$.

Verificación

El invariante es un predicado que describe todos los estados por los que atraviesa el cómputo realizado por el bucle, observados justo antes de evaluar la condición B de terminación.

Ejemplo:

$i = 0;\ q = 0;\ p = 1;$
$P : \{i = 0 \wedge q = 0 \wedge p = 1\}$
while $(i < n)\{$
    $q = q + p;\ p = p + 2;\ i = i + 1;$
$\}$
$Q : \{i = n \wedge q = ? \wedge p = ?\}$

valores de las variables antes de eval. cond. $i < n$.

estado	i	q	p
P_0	0	0	1
P_1	1	1	3
P_2	2	4	5
P_3	3	9	7
...			

¿ Que relaciones entre las variables se mantienen durante la ejecución del bucle?
Un invariante del bucle es: $\boxed{I : \{0 \leq i \leq n \wedge q = i^2 \wedge p = 2i + 1\}}$

Verificación

La *función cota* o *función limitadora* es una función $t : estado \to \mathcal{Z}$ positiva que decrece cada vez que se ejecuta el cuerpo del bucle.

La función cota garantiza la terminación del bucle

Para encontrar una función cota se observan las variables que son modificadas por el cuerpo A del bucle, y se construye con ellas una expresión entera t que decrezca

Ejemplo: cálculo del cuadrado de un número.
Las variables i, p y q crecen, n se mantiene inalterable, por lo tanto $n - i$ decrece.
Además, la condición del bucle $i \leq n$ garantiza que la función es positiva.

La función cota es : $t = n - i$.

Ejemplos

Especifica una función que calcule la suma de las componentes de un vector.

Dada la siguiente implementación indica un invariante del bucle.

```
1        int suma(vector<int> const& v){
2          int n = v.size();  x = 0;
3          while (n != 0)
4          {
5            x = x+v[n-1];
6            n = n-1;
7          }
8          return x;
9        }
```

Ejemplos

Especificación.

$\{\mathbf{v.size}() \geq \mathbf{0}\}$
int *suma* (**vector<int>** *v*)
$\{\mathbf{return}\ (\mathbf{\Sigma} i : \mathbf{0} \leq i < \mathbf{v.size}() : v[i])\}$

```
1       int suma(vector<int> const& v){
2         int n = v.size();  x = 0;
3         while (n != 0)
4         {
5           x = x+v[n-1];
6           n = n-1;
7         }
8         return x;
9       }
```

Invariante:
$I \equiv \{0 \leq n \leq \mathbf{v.size}() \wedge x = (\mathbf{\Sigma} i : n \leq i < \mathbf{v.size}() : v[i])\}$.

Ejemplos

- Indicar un invariante para el siguiente bucle suponiendo x, y, n: enteros.

  $\{n \geq 0\}$

```
1       int x, y;
2       x = 0;  y = 1;
3       while (x != n)
4       {
5          x = x+1;
6          y = y+y;
7       }
```

  $\{y = 2^n\}$

- Invariante: $I \equiv \{0 \leq x \leq n \wedge y = 2^x\}$

Derivación

Derivar: construir las instrucciones de un algoritmo a partir de su especificación asegurando su corrección.

Los algoritmos se ajustan al esquema:

$\{P\}$
A_0 `(Inicializacion)`
$\{I, Cota\}$
`while` `(B) {`
  $\{I \wedge B\}$
  A_1 `(Restablecer)`
  $\{R\}$
  A_2 `(Avanzar)`
  $\{I\}$
`}`
$\{Q\}$

- A_0 Hace que el invariante se cumpla inicialmente A_2 hace que la cota decrezca.
- A_1 mantiene el invariante a cierto.

- **Pasos** para construir un algoritmo con bucle:
  1. Diseñar el invariante y la condición del bucle sabiendo que se tiene que cumplir:
     $I \wedge \neg B \Rightarrow Q$
  2. Diseñar A_0 para hacer el invariante cierto: $\{P\}A_0\{I\}$
  3. Diseñar la función cota, C, de tal forma que: $I \wedge B \Rightarrow C \geq 0$.
  4. Diseñar A_2 y el predicado $R \equiv pmd(A_2, I)$.
  5. Diseñar A_1 para que se cumpla: $\{I \wedge B\}A_1\{R\}$.
  6. Comprobar que la cota realmente decrece:
     $$\{I \wedge B \wedge C = T\}A_1, A_2\{C < T\}$$