Notas en Algo 1

Sergio Esteban Muñoz

11 de junio de 2022

Capítulo 1

Desarrollo de loop's a partir de invariantes y f_v

Este capítulo analiza dos métodos para desarrollar un bucle cuando se dan la condición previa P, la condición posterior Q, el invariante I y la función ligada f_v.

1. El primer método conduce naturalmente a **un** ciclo con una sola guarda, **do** B $\rightarrow$ S **od**.
2. El segundo aprovecha la flexibilidad de la construcción iterativa y generalmente da como resultado bucles **con más** de una guarda.

Se usará mucho el teorema del invariante vista en la teórica y puede ser conveniente revisarla antes de continuar. Como es nuestra práctica en todo momento, las partes del desarrollo que ilustran los principios que se cubrirán se discuten de manera formal y detallada, mientras que otras partes se tratan de manera más informal.

1.0.1. Desarrollo de la guarda primero

Sumar los elementos de una secuencia

Considere el siguiente problema. Escriba un programa que, dado un $n \geq 0 \wedge$

b[0]	b[1]	$\cdots$	b[i]	$\cdots$	b[n-2]	b[n-1]

, almacene en la variable s la suma de los elementos de b. La precondición P es simplemente $n \geq 0$; la postcondición Q es:

$$Q : s = \sum_{j=0}^{n-1} b[j]$$

Se desea un ciclo con el siguiente invariante y función variante (f_v).

$I : 0 \leq i \leq n \wedge_L s = \sum_{j=0}^{i-1} b[j]$

$f_v = n - i$

Se ha introducido la variable i . El invariante establece que en cualquier punto del cálculo s contiene la suma de los primeros valores de i y b .

La asignación $i,\ s := 0,\ 0$ obviamente establece I, por lo que será suficiente como inicialización. (Tenga en cuenta que i , s:= 1, b[0] no es suficiente porque, si n = 0, no se puede ejecutar. Si n = 0, la ejecución del programa debe establecer s en la identidad de suma, 0).

El siguiente paso es determinar la guarda B para el bucle **do** B $\rightarrow$ S **od**. El teorema del invariante requiere que $I \wedge \neg B \rightarrow Q$, por lo que se elige B para satisfacerla. Comparando I y Q , concluimos que $i = n$ servirá. La guarda deseada B del ciclo es por lo tanto su complemento ($i \neq n$), el programa se ve como:

$$i, s := 0, 0 \textbf{ do } i \neq n \rightarrow ?? \textbf{ do}$$

Ahora para el cuerpo del ciclo. El propósito del comando es avanzar hacia la terminación. Es decir, disminuir la función f_v y una primera opción obvia para disminurla es $i := i + 1$. Pero, esto destruiría el invariante, y para restablecerlo $b[i]$ debe agregarse simultáneamente a s. Así, el programa es:

$$i, s := 0, 0 \textbf{ do } i \neq n \rightarrow \text{i, s = i+1, s + b[i] } \textbf{do}$$

Observación: Para aquellos que no se sientan cómodos con la asignación múltiple, la prueba formal de que I se mantiene es la siguiente. Tenemos:

$$wp(\text{"}i, s = i + 1, s + b[i]\text{"}) = 0 \leq i + 1 \leq n \wedge s + b[i] = \sum_{j=0}^{i} b[j]$$

y esto es mas débil que $I \wedge i \neq n$

Discución
En primer lugar, analicemos el equilibrio entre formalidad e intuición observado aquí. Las condiciones previas y posteriores, el invariante y la función ligada se dieron de manera formal y precisa. El desarrollo de las partes del programa se proporcionó de manera menos formal, pero por los teoremas de clase, que se basan en el teorema formal de la construcción iterativa, proporcionó la mayor parte de la motivación y la comprensión. Para verificar el desarrollo informal, nos basamos en la teoría (al verificar que el cuerpo del bucle mantuvo el invariante).

Una estrategia importante en el desarrollo fue encontrar la guarda antes que el comando. Y la principal consideración al encontrar la guarda B fue que tenía que satisfacer $I \wedge \neg B \rightarrow Q$. Entonces, $\neg B$ se desarrolló y luego se uso su complemento para producir B.

Al principio, algunos se oponen a encontrar la guarda de esta manera, porque tradicionalmente usarían la guarda $i < n$ en lugar de $i \neq n$. Sin embargo, $i \neq n$ es mejor, porque un error de software o hardware que hiciera $i > n$ daría como resultado una ejecución sin terminación. Es mejor usar el tiempo pensando una buena guarda antes que sufrir las consecuencias de que no se detecte un error, lo que sucedería si se usara la guarda $i < n$. Este análisis conduce a lo siguiente:

Principio:
En igualdad de condiciones, haga que las guardas de un ciclo sean lo más débiles posible de modo que un error pueda causar un loop infinito.

Un punto importante sobre el desarrollo fue el énfasis en la terminación. La necesidad de avanzar hacia la terminación motivó el desarrollo del cuerpo de bucle; restablecer el invariante fue la segunda consideración. En realidad, cada bucle con una guarda tiene una interpretación de alto nivel

{ invariante : I}
{ funcion variante : f_v}
do $B \rightarrow$ decrece f_v, manteniendo I `true` **od**
{ $I \wedge \neg B$ }

Estrategia para desarrollar un loop:
Primero desarrollar la guarda B de modo que $P \wedge \neg B \rightarrow Q$; luego desarrolle el cuerpo para que disminuya la función ligada mientras restablece el invariante.

Buscando un elemento en una matriz

Considere el siguiente problema. Escriba un algoritmo que, dada una matriz $b[0 : m - 1][0 : n - 1]$, donde $m > 0 \wedge n > 0$, busque en b un valor fijo x.
Si x ocurre en varios lugares en b, no importa en qué lugar se encuentre. Para este problema, usaremos la notación bidimensional convencional del libro de Gries, escribiendo b como $b[0 : m - 1, 0 : n - 1]$.
Usando las variables i y j, al terminar $x = b[i, j]$ o, si esto no es posible, $i = m$. Para ser más precisos, la ejecución del programa debe establecer:

$$Q : (0 \leq i < m \wedge 0 \leq j < n \wedge x = b[i, j]) \vee (i = m \wedge x \notin b)$$

El invariante I, dado a continuación usando un diagrama, $R \equiv$ establece que x no está en las filas ya buscadas $b[0 : i - 1]$ y no en las columnas ya buscadas $b[i, 0 : 0 : j - 1]$ de la fila actual i.(Pag 182 - Gries)

$$I : 0 \leq i \leq m \wedge 0 \leq j < n \wedge R$$

La función ligada f_v es el número de valores en la sección no probada: $(m - i) * n - j$. Como primer paso en el desarrollo, antes de seguir leyendo, determine la inicialización del bucle.

La elección obvia es $i, j := 0, 0$ para la sección en la que *"x no esta aquí"* está vacío. A continuación, ¿cuál debería ser la guarda B del ciclo?

Expresión , $\neg B$ debe satisfacer $I \wedge \neg B \rightarrow Q$. Debe ser lo suficientemente fuerte para que se pueda establecer cada una de las dos disyuntivas de Q. Para proporcionar la primera disyunción, elija $i < m \wedge_L x = b[i, j]$; para proporcionar el segundo, elija $i = m$. Se necesita el operador $\wedge_L$ para garantizar que la expresión esté bien definida, ya que $b[i, j]$ puede no estar definido si $i \geq m$. Por lo tanto, $\neg B$ debe ser

$$\neg B : i = m \vee (i < m \wedge_L x = b[i, j])$$

Usando *De Morgan's*

$$B : i \neq m \wedge (i \geq m \vee_L x \neq b[i, j])$$

Dado que la guarda B debe evaluarse solo cuando el invariante I es verdadero, lo que significa que $i < m$ es verdadero, se puede simplificar a

$$B : i \neq m \wedge (i = m \vee_L x \neq b[i, j])$$

$$B : i \neq m \wedge_L x \neq b[i, j])$$

La línea final es, por lo tanto, la guarda del bucle. El siguiente paso es determinar el cuerpo del bucle. Intentalo, antes de seguir leyendo.

El proposito del cuerpo del cliclo es disminr la función ligada f_v, que es el número de elementos en la sección no probada: $(m-i)*n-j$. $I \wedge B$, es la condición bajo la cual se ejecuta el cuerpo, implica que $i < m, j < n \wedge x \neq b[i, j]$, por lo que el elemento $b[i, j]$ que está en la sección no probada, se puede mover en la sección probada. Un comando posible para hacer esto es $j = j + 1$ pero mantiene el invariante I si $j < n - 1$. Entonces tenemos el comando:

$j < n - 1 \rightarrow j := j + 1$

¿Qué hacemos si $j \geq n-1$? En este caso como el invriante I es verdadero, tenemos $j = n-1$. Por lo tanto, debemos determinar que hacer si $j = n - 1$, es decir si $b[i, j]$ es elemento más a la derecha de su fila. Pra mover $b[i, j]$ a la sección probada se requiere moverse al principio de la siguiente fila, es decir, ejecutar $i, j := i + 1, 0$.

El cuerpo del loop es:

if $j < n - 1 \rightarrow j := j + 1$
$| \; j = n - 1 \rightarrow i, j := i + 1, 0$

El programa queda

$i, j := 0, 0$
do $i \neq m \wedge x \neq b[i, j] \rightarrow$
if $j < n - 1 \rightarrow j := j + 1 | \; j = n - 1 \rightarrow i, j := i + 1, 0$ **fi**
od