Programación Avanzada

25/Septiembre/2014

Ixchel S. Zazueta García.

1. a) subroutine comp(x, y; z)

```
z ← x
w ← y
while (w > 0)
    z ← z + y
    w ← w - 1
return z
```

análisis, z_n, donde n son iteraciones:

$z_0 = x$

$w_0 = y$

$z_1 = z_0 + y = x + y$

$w_1 = w_0 - 1 = y - 1$

$z_2 = z_1 + y = x + y + y = x + 2y$

$w_2 = w_1 - 1 = y - 1 - 1 = y - 2$

$z_3 = z_2 + y = x + 2y + y = x + 3y$

$w_3 = w_2 - 1 = y - 2 - 1 = y - 3$

Invariante:

$P(n)$: $z_n = x + (y)n$,

$w_n = y - n$ $\forall n$ ciclos

demostración por el método de inducción:

paso base para $n = 0$

$P(0) \Rightarrow z_0 = x + (y)0 = x$

$w_0 = y - 0 = y$, y esto es verdad por programa, antes de entrar al ciclo.

hipótesis de inducción: $z_k = x + ky$ y $w_k = y - k$, $k > 0$ es verdad.

por demostrar para $(k+1)$: $z_{k+1} = x + (k+1)y$ y $w_{k+1} = y - (k+1)$

por el programa, sabemos: $z_{k+1} = \underbrace{z_k}_{\text{por hipótesis}} + y$

$= \underbrace{(x + ky)}_{\text{por hipótesis}} + y = x + ky + y = x + (k+1)y$

por el programa sabemos: $w_{k+1} = w_k - 1$

$= \underbrace{(y - k)}_{\text{por hipótesis}} - 1 = y - (k+1)$

y queda demostrado.

b) subroutine Diff (x, y; z)

```
z ← x
w ← y
while (w > 0)
    z ← z - 1
    w ← w - 1
return z
```

análisis

$z_0 = x$; $w_0 = y$

$z_1 = z_0 - 1 = x - 1$

$w_1 = w_0 - 1 = y - 1$

$z_2 = z_1 - 1 = x - 1 - 1 = x - 2$

$w_2 = w_1 - 1 = y - 1 - 1 = y - 2$

$z_3 = z_2 - 1 = x - 2 - 1 = x - 3$

$w_3 = w_2 - 1 = y - 2 - 1 = y - 3$

Invariante:

$P(n)$: $z_n = x - n$

$w_n = y - n$, donde n es el número de ciclos.

Inducción

caso base para $n = 0$

$P(0)$: $z_0 = x - 0 = x$

$w_0 = y - 0 = y$, y esto es verdad para 0 ciclos.

hipótesis de inducción. $z_k = x - k$ y $w_k = y - k$, $k > 0$, donde k son los ciclos.

por demostrar para $(k+1)$: $z_{k+1} = x - (k+1)$ y $w_{k+1} = y - (k+1)$

por el programa sabemos: $z_{k+1} = z_k - 1$; $w_{k+1} = w_k - 1$

por hipótesis de inducción: $= x - k - 1$; $w_{k+1} = y - k - 1$

$= x - (k+1)$ $= y - (k+1)$,

y así queda demostrada la invariante.

2. $\{K>n\}$ if $(K<n)$ $K:=n$ else $n:=K$ $\{n=K\}$
P B C_1 C_2 Q

ⓐ $\{P \wedge B\}\ C_1\ \{Q\}$
ⓑ $\{P \wedge \neg B\}\ C_2\ \{Q\}$
$\{P\}$ if B C_1 else C_2 $\{Q\}$

ⓐ por demostrar:

$\{P\} = Q^{v}_{\epsilon} = \{K:=n, n=K\} = \{K=K\} = \{\}$

$\{(K>n) \wedge (K<n)\} \Rightarrow \{\}$

$\{\}\ K:=n\ \{n=K\}$

$\{(K>n) \wedge (K<n)\}\ K:=n\ \{n=K\}$

por demostrar ⓑ.

$\{P\} = Q^{v}_{\epsilon} = \{n:=K, n=K\} = \{K=K\} = \{\}$

$\{(K>n) \wedge \neg(K<n)\} \Rightarrow \{\}$

$\{\}\ n:=K\ \{n=K\}$

$\{(K>n) \wedge \neg(K<n)\}\ n:=K\ \{n=K\}$

por lo tanto, ahora podemos demostrar:

$\{(K>n) \wedge (K<n)\}\ K:=n\ \{n=K\}$

$\{(K>n) \wedge \neg(K<n)\}\ n:=K\ \{n=K\}$

$\{K>n\}$ If $(K<n)$ then $K:=n$ else $n:=K$ $\{n=K\}$ ||

3. Demostrar q la tema es válida $\{\}w:=\text{calcula}(n)\{w \geq 2^{n-1}\}$, para el programa

```
public static int calcula (int n){
    int c=1;
    if(n==0 || n==1)
        return 1;
    else
        for (i=2; i<=n; i++)
            c=c*i;
        return c;
}
```

análisis del código

$C_0 = 1 =$
$i_1 = 2$, $C_1 = C_0 \cdot i_1 = 1 \cdot 2$
$i_1+1 = i_2 = 3$, $C_2 = C_1 \cdot i_2 = 1 \cdot 2 \cdot 3$
$i_2+1 = i_3 = 4$, $C_3 = C_2 \cdot i_3 = 1 \cdot 2 \cdot 3 \cdot 4$, $\rightarrow i_n = n+1$

invariante:
$P(n)$: $C_n = (i_n)!$ donde n son ciclos,
esto significa que calcula(z) $= z!$

vamos a demostrar la invariante por inducción:

caso base, para $n=0$ $P(0)$: $C_0 = (0+1)! = 1$, y vemos que esto es verdad en el código.

hipótesis de inducción: $P(k)$: $C_k = (i_k)! = (k+1)!$ para k ciclos y $K>0$

por demostrar para $(k+1)$: $C_{k+1} = (i_{k+1})!$

por el programa sabemos: $C_{k+1} = C_k * i_{k+1}$

$= \underbrace{i_k!}_{\text{por hipótesis}} \cdot i_{k+1} = i_{k+1}!$ ha quedado demostrada la invariante. ||

ahora podemos decir que calcula(n) $= n!$ para demostrar la tema

$\{\}\ w := calcula(n)\{\ w \geq 2^{n-1}\}$

$\{Q\} = C_p^t = \{w := calcula(n), \{\}\} = \{w := calcula(n)\}$, pero tenemos que $w \geq 2^{n-1}$,

entonces debemos demostrar que: $calcula(n) \geq 2^{n-1}$, pero por demostración anterior queda:

$n! \geq 2^{n-1}$.

Inducción

Paso base, $n=0$: $0! \geq 2^{0-1}$

$1 \geq 1/2$ y esto es verdad.

hipótesis de inducción: $k! \geq 2^{k-1}$, $k \geq 0$.

por demostrar para $(k+1)$: $(k+1)! \geq 2^{(k+1)-1}$, $k \geq 0$

sabemos por hipótesis: $k! \geq 2^{k-1}$

$(k+1)k! \geq 2^{k-1}(k+1)$

pero tenemos que: $k+1 \geq 0+1 \Rightarrow (k+1) \geq 1$,

entonces podemos hacer $\{((k+1)=2)\} \Rightarrow \{(k+1) \geq 1\}$ y con esto tenemos

$(k+1)! \geq 2^{k-1}(2)$

$(k+1)! \geq 2^k$, y así queda demostrado que

$n! \geq 2^{n-1}$, más específicamente que

$calcula(n) \geq 2^{n-1}$