DEPARTAMENTO DE COMPUTACIÓN
UNIVERSIDADE DA CORUÑA

Tecnología de la programación
Ingeniería Técnica en Informática de Sistemas. Curso 2006/07
Convocatoria Ordinaria de Septiembre (17 de Septiembre de 2007)

Pregunta 1: Sea α el predicado $(\forall I \in [0,n\rangle \wedge b[I] \neq 0 : (\forall J \in [0,m\rangle : c[J] \neq b[I]))$. Entonces:

×(a) α es equivalente a $\neg(\exists I \in [0,n\rangle, J \in [0,m\rangle : b[I] \neq 0 \wedge c[J] = b[I])$

×(b) $\alpha_K^I = \alpha$

(c) $\alpha_I^m = (\forall I \in [0,n\rangle \wedge b[I] \neq 0 : (\forall J \in [0,I\rangle : c[J] \neq b[I]))$

×(d) $\alpha_I^m = (\forall K \in [0,n\rangle \wedge b[K] \neq 0 : (\forall J \in [0,I\rangle : c[J] \neq b[K]))$

(a) **Sí**. Podemos verlo haciendo las siguientes transformaciones:

$$
\begin{array}{l}
(\forall I \in [0,n\rangle \wedge b[I] \neq 0 : (\forall J \in [0,m\rangle : c[J] \neq b[I])) \\
\equiv (\forall I \in [0,n\rangle : b[I] \neq 0 \Rightarrow (\forall J \in [0,m\rangle : c[J] \neq b[I])) \\
\equiv (\forall I \in [0,n\rangle : b[I] \neq 0 \Rightarrow \neg(\exists J \in [0,m\rangle : c[J] = b[I])) \\
\equiv \neg(\exists I \in [0,n\rangle : \neg(b[I] \neq 0 \Rightarrow \neg(\exists J \in [0,m\rangle : c[J] = b[I]))) \\
\equiv \neg(\exists I \in [0,n\rangle : b[I] \neq 0 \wedge (\exists J \in [0,m\rangle : c[J] = b[I])) \\
\equiv \neg(\exists I \in [0,n\rangle, J \in [0,m\rangle : b[I] \neq 0 \wedge c[J] = b[I])
\end{array}
$$

(b) **Sí**. Dado que no hay apariciones libres de I en α.

(c) **No**. Puesto que en el cambio de m por una I libre hemos ligado accidentalmente esa I al cuantificador $\forall$ de la izquierda. Para poder hacer ese cambio, deberíamos primero cambiar el nombre de la variable ligada en $\forall I$ ...

(d) **Sí**. Por lo dicho en la respuesta anterior, el cambio es correcto, ya que hemos cambiado el nombre de la variable ligada en $\forall K$ en lugar de $\forall I$. Es necesario destacar que una fórmula es idéntica a otra si sólo hemos cambiado los nombres de las variables ligadas por otros nombres nuevos.

Pregunta 2: Demostrar que *P* es una *invariante* para **DO**: **do** $B_1 \rightarrow S_1 | B_2 \rightarrow S_2$ **od**, **equivale** a probar:

(a) Para cualquier postcondición R, $P \wedge \neg(B_1 \vee B_2) \Rightarrow R$

(b) $B_1 \wedge P \Rightarrow wp(S_1, P)$

(c) $\{B_i\}S_i\{P\}$ para las dos ramas $i \in \{1,2\}$

×(d) $\{P \wedge B_i\}S_i\{P\}$ para las dos ramas $i \in \{1,2\}$

(a) **No**. La definición de que una fórmula P es invariante nunca depende de que escojamos una u otra postcondición R determinada. En esta respuesta estaríamos exigiendo que P nos permita probar cierta cualquier postcondición R que se nos ocurra.

(b) **No**. Esta condición es necesaria, pero no es suficiente para que P sea invariante, dado que no estamos garantizando si P se mantiene cierta cuando el bucle entra en la rama S_2.

(c) **No**. Aquí estaríamos probando que P es cierto al terminar cualquier rama del bucle, aunque no lo fuese antes de ejecutarla. Esta condición sería suficiente para que P fuese invariante, siempre que P se cumpliese al comenzar (algo que no se especifica). Sin embargo, no es una condición necesaria: sólo tenemos que probar que P se mantiene cierta si ya era cierta antes.

(d) **Sí**. Es exactamente la definición.

Pregunta 3: Supongamos que $\{Q\}S\{A \wedge B\}$ entonces:

(a) $\{Q\}S; S\{A \wedge B\}$

(b) $\{Q \vee A\}S\{A \wedge B\}$

×(c) $\{Q\}S\{A\}$ y $\{Q\}S\{B\}$

×(d) $\{Q\}S\{A \vee B\}$

(a) **No**. si ejecutamos S dos veces seguidas puede producirse un cambio de estado no contemplado en el programa original. Para verlo mejor basta tomar, por ejemplo, S como la instrucción $i := i + 1$, Q como $i = 0$ y A y B iguales a $i = 1$.

(b) **No**. Si debilitamos la precondición, puede dejar de cumplirse. El contraejemplo de antes también nos vale aquí. Si usamos $i = 0 \vee i = 1$ de precondición, después de ejecutar $i := i + 1$ ya no podemos garantizar $i = 1$ al acabar.

(c) **Sí**. Se deriva de la propiedad $wp(S, A \wedge B) \equiv wp(S, A) \wedge wp(S, B)$

(d) **Sí**. Siempre es posible debilitar la postcondición, y la fórmula $A \vee B$ es más débil que $A \wedge B$.

Pregunta 4: Sea la función **func** $f(X)$ **ret(r)** $\{X \geq 0\}i, r := 0, 1;$ **do** $r < X \rightarrow i := i + 1; r := r + f(i)$ **od** $\{T\}$

×(a) $f(2)$ devuelve 2

(b) $f(2)$ devuelve 1

×(c) f es recursiva múltiple

(d) f es recursiva simple, lineal y no final

(a) **Sí**. La llamada a $f(1)$ devuelve 1, dado que no entra en el bucle y el resultado que se devuelve es el valor inicial de r. En ese momento r toma el valor 2 (valor inicial más el resultado de $f(1)$) y finaliza el bucle ya que el guardián deja de ser cierto. El valor final acumulado en r es $1 + f(1) = 1 + 1 = 2$.

(b) **No**. Por lo dicho en el apartado anterior.

(c) **Sí**. Ya que la recursividad es múltiple al poder llamar a f varias veces, cuando el bucle hace más de una iteración. Es además recursiva final por que la llamada recursiva es la última instrucción de la función.

(d) **No**. Por lo dicho en la respuesta anterior.

Pregunta 5: (2 puntos) Calcula y simplifica: $wp(\text{"}b[i], b[0] := b[b[i]], b[i]; b[b[i]] := b[0]\text{"}, b[0] = b[i])$

$wp(\text{"}b[i], b[0] := b[b[i]], b[i]; b[b[i]] := b[0]\text{"}, b[0] = b[i])$

$\equiv wp(\text{"}b[i], b[0] := b[b[i]], b[i]\text{"}, \underbrace{wp(\text{"}b[b[i]] := b[0]\text{"}, b[0] = b[i])})$

Calculamos primero el wp para la última instrucción:

$wp(\text{"}b[b[i]] := b[0]\text{"}, b[0] = b[i])) \equiv enrango(b, b[i])$ cand $enrango(b, 0)$ cand $(b[0] = b[i])^{b}_{(b;b[i]:b[0])}$

Desarrollando el último término obtenemos:

$\equiv (b; b[i] : b[0])[0] = (b; b[i] : b[0])[i]$

$\equiv (b[i] = 0 \wedge b[0] = (b; b[i] : b[0])[i]) \vee (b[i] \neq 0 \wedge (b[0] = (b; b[i] : b[0])[i])$

Sacando factor común:

$\equiv (b[i] = 0 \vee b[i] \neq 0) \wedge b[0] = (b; b[i] : b[0])[i])$

$\equiv T \wedge (b[0] = (b; b[i] : b[0])[i])$

$\equiv b[0] = (b; b[i] : b[0])[i]$

$\equiv (b[i] = i \wedge b[0] = b[0]) \vee (b[i] \neq i \wedge b[0] = b[i])$

$\equiv b[i] = i \vee (b[i] \neq i \wedge b[0] = b[i])$

$\equiv (b[i] = i \vee b[0] = b[i])$

Sustituyendo ahora en la expresión inicial:

$wp(\text{"}b[i], b[0] := b[b[i]], b[i]\text{"}, enrango(b, b[i])$ cand $enrango(b, 0)$ cand $b[i] = i \vee b[0] = b[i])$

$\equiv enrango(b, i)$ cand $enrango(b, 0)$ cand $enrango(b, b[i])$ cand $enrango((b; i : b[b[i]]; 0 : b[i]), (b; i : b[b[i]]; 0 : b[i])[i])$
cand $enrango((b; i : b[b[i]]; 0 : b[i]), 0)$ cand $(b[i] = i \vee b[0] = b[i])^b_{(b;i:b[b[i]];0:b[i])}$

$\equiv enrango(b, i)$ cand $enrango(b, 0)$ cand $enrango(b, b[i])$ cand $enrango(b, i; 0)$
cand $enrango(b, 0)$ cand $(b[i] = i \vee b[0] = b[i])^b_{(b;i:b[b[i]];0:b[i])}$

$\equiv enrango(b, i)$ cand $enrango(b, 0)$ cand $enrango(b, b[i])$ cand $enrango(b, i; 0)$
cand $(b[i] = i \vee b[0] = b[i])^b_{(b;i:b[b[i]];0:b[i])}$

Resolviendo el último término:

$(b[i] = i \vee b[0] = b[i])^b_{(b;i:b[b[i]];0:b[i])}$

$\equiv (b; i : b[b[i]]; 0 : b[i])[i] = i \vee (b; i : b[b[i]]; 0 : b[i])[0] = (b; i : b[b[i]]; 0 : b[i])[i]$

$\equiv (0 = i \wedge (b[i] = i \vee b[i] = b[i])) \vee (0 \neq i \wedge ((b; i : b[b[i]])[i] = i \vee b[i] = (b; i : b[b[i]])[i]))$

$\equiv (0 = i \wedge (b[i] = i \vee T)) \vee (0 \neq i \wedge (b[b[i]] = i \vee b[i] = b[b[i]]))$

$\equiv 0 = i \vee (0 \neq i \wedge (b[b[i]] = i \vee b[i] = b[b[i]]))$

$\equiv 0 = i \vee b[b[i]] = i \vee b[i] = b[b[i]]$

Con lo que la solución final quedaría:

$\equiv enrango(b, i)$ cand $enrango(b, 0)$ cand $enrango(b, b[i])$ cand $enrango(b, i; 0)$ cand $(0 = i \vee b[b[i]] = i \vee b[i] = b[b[i]])$

Pregunta 6: (4,5 puntos) Dado un vector $b[0 \ldots n-1] : integer$ con $n \geq 4$, escribe un programa que devuelva un booleano indicando si se puede hallar dentro de dicho vector dos parejas **consecutivas** de elementos tales que sus sumas sean idénticas (es decir, que la suma de los valores de la primera pareja sea la misma que la suma de los valores de la segunda pareja). Por ejemplo, si $b = \{0, -1, -3, 3, -4, 0, -1, -7, 1\}$ el valor que debe devolver el programa es T. Se pide: **NOTA: Esta es una de las soluciones posibles.**

6.1 Establecer una precondición y una postcondición adecuadas

$\{Q: \ n \geq 4 \wedge b[0 : n-1] : integer \wedge b = B\}$
$\{R: \ p = (\exists K : 2 < K < n : b[K-3] + b[K-2] = b[K-1] + b[K]) \wedge b = B\}$

6.2 Fijar una invariante y una función cota

$\{P:\ 2 \leq i < n \wedge p = (\exists K : i < K < n : b[K-3] + b[K-2] = b[K-1] + b[K]) \wedge b = B\}$
$\{t:\ i - 2\}$

6.3 Escribir el programa

$\{Q\}$
$S_0 : i, p := n - 1, F;$
$\{P\}$
$\{t\}$
do $B_1 : i \neq 2 \wedge (b[i-3] + b[i-2] = b[i-1] + b[i]) \rightarrow S_1 : p, i := T, i - 1;$
| $B_2 : i \neq 2 \wedge (b[i-3] + b[i-2] \neq b[i-1] + b[i]) \rightarrow S_2 : i := i - 1;$
od
$\{P \wedge \neg BB\}$
$\{R\}$

6.4 Demostrar su corrección *total*

a) $Q \Rightarrow wp(S_0, P)$

$n \geq 4 \wedge b[0 : n-1] : integer \wedge b = B \Rightarrow wp(\text{"}i, p := n - 1, F\text{"}, P)$

$\equiv n \geq 4 \wedge b = B \Rightarrow 2 \leq n - 1 < n \wedge F = (\exists K : n - 1 < K < n : b[K-3] + b[K-2] = b[K-1] + b[K]) \wedge b = B\}$

El rango del cuantificador existencial es vacío y su evaluación es F.

$\equiv n \geq 4 \wedge b = B \Rightarrow 2 \leq n - 1 < n \wedge (F = F) \wedge b = B$

$\equiv n \geq 4 \wedge b = B \Rightarrow 3 \leq n < n \wedge b = B$ que es cierto.

b) $\{P \wedge B_i\} S_i \{P\}$ para $1 \leq i \leq n$.

1) $P \wedge B_1 \Rightarrow wp(S_1, P) \equiv P \wedge B_1 \Rightarrow wp(\text{"}p, i := T, i - 1\text{"}, P)$

$2 \leq i < n \wedge p = (\exists K : i < K < n : b[K-3] + b[K-2] = b[K-1] + b[K]) \wedge b = B$

$\wedge i \neq 2 \wedge (b[i-3] + b[i-2] = b[i-1] + b[i]) \Rightarrow$

$$\underbrace{2 \leq i - 1 < n}_{\beta} \wedge T = \overbrace{(\exists K : i - 1 < K < n : b[K-3] + b[K-2] = b[K-1] + b[K])}^{\alpha} \wedge b = B$$

$b = B$ lo tenemos directamente en P y la primera parte de la conjunción (β) se obtiene a partir de P y B_1 de la siguiente forma:

$$\left.\begin{array}{l} P \Rightarrow 2 \leq i < n \\ B_1 \Rightarrow i \neq 2 \end{array}\right\} 2 < i < n \Rightarrow 2 \leq i - 1 < n$$

En α aislamos el caso $K = i$:

$(\exists K : i < K < n : b[K-3] + b[K-2] = b[K-1] + b[K]) \vee (b[i-3] + b[i-2] = b[i-1] + b[i])$

Por B_1 sabemos que la segunda parte de la disyunción es cierta, por lo que sustituyendo en la parte derecha de la implicación: $T = T$ y α es cierto.

2) $P \wedge B_2 \Rightarrow wp(S_2, P) \equiv P \wedge B_2 \Rightarrow wp(\text{"}i := i - 1\text{"}, P)$

$2 \leq i < n \wedge p = (\exists K : i < K < n : b[K-3] + b[K-2] = b[K-1] + b[K]) \wedge b = B$

$\wedge i \neq 2 \wedge (b[i-3] + b[i-2] \neq b[i-1] + b[i]) \Rightarrow$

$$\underbrace{2 \leq i - 1 < n}_{\beta} \wedge p = \overbrace{(\exists K : i - 1 < K < n : b[K-3] + b[K-2] = b[K-1] + b[K])}^{\alpha} \wedge b = B$$

$b = B$ lo tenemos directamente en P y la primera parte de la conjunción (β) se obtiene a partir de P y B_2:

$$\left.\begin{array}{l} P \Rightarrow 2 \leq i < n \\ B_2 \Rightarrow i \neq 2 \end{array}\right\} 2 < i < n \Rightarrow 2 \leq i - 1 < n$$

En α aislamos el caso $K = i$:

$(\exists K : i < K < n : b[K-3] + b[K-2] = b[K-1] + b[K]) \vee (b[i-3] + b[i-2] = b[i-1] + b[i])$

La segunda parte de la disyunción sabemos por B_2 que es falsa,

y por P que $p = (\exists K : i < K < n : b[K-3] + b[K-2] = b[K-1] + b[K])$. Si sustituimos, se obtiene que:

$p = p \vee F \equiv p = p$ que es cierto.

c) $P \wedge \neg BB \Rightarrow R$

Primero calculamos $\neg BB \equiv \neg((i \neq 2 \wedge (b[i-3] + b[i-2] = b[i-1] + b[i])) \vee (i \neq 2 \wedge (b[i-3] + b[i-2] \neq b[i-1] + b[i])))$

$\equiv \neg(i \neq 2 \wedge ((b[i-3] + b[i-2] = b[i-1] + b[i]) \vee (b[i-3] + b[i-2] \neq b[i-1] + b[i])))$

$\equiv \neg((i \neq 2) \wedge T) \equiv i = 2$

En la expresión de partida tenemos entonces:

$2 \leq i < n \wedge p = (\exists K : i < K < n : b[K-3] + b[K-2] = b[K-1] + b[K]) \wedge b = B) \wedge (i = 2) \Rightarrow R$

Sustituyendo el valor de i en P obtenemos directamente R.

d) $P \wedge BB \Rightarrow (t > 0)$

En P tenemos que $2 \leq i$ que junto con $BB \equiv i \neq 2$ conlleva $2 < i$, que a su vez equivale a decir que $0 < i - 2$ es cierto.

e) $\{P \wedge B_i\} t_1 := t; S_i \{t < t_1\}$ para $1 \leq i \leq n$. En este último paso tenemos que distinguir dos casos:

a) $P \wedge B_1 \Rightarrow wp(\text{"}t_1 := t; p, i := T, i-1\text{"}, t < t_1)$

$$\begin{array}{l} P \wedge B_1 \Rightarrow ((i-2 < t_1)^{p,i}_{T,i-1})^{t_1}_{t} \\ P \wedge B_1 \Rightarrow ((i-1)-2 < t_1)^{t_1}_{i-1} \\ P \wedge B_1 \Rightarrow i-1-2 < i-2 \\ P \wedge B_1 \Rightarrow -1 < 0 \\ \text{Que es cierto.} \end{array}$$

b) $P \wedge B_2 \Rightarrow wp(\text{"}t_1 := t; i := i-1\text{"}, t < t_1)$

$$\begin{array}{l} P \wedge B_2 \Rightarrow ((i-2 < t_1)^{i}_{i-1})^{t_1}_{t} \\ P \wedge B_2 \Rightarrow ((i-2)-1 < t_1)^{t_1}_{i-2} \\ P \wedge B_2 \Rightarrow i-2-1 < i-2 \\ P \wedge B_2 \Rightarrow -1 < 0 \\ \text{Que es cierto.} \end{array}$$

Pregunta 7: (1,5 punto) Dado un array $b[0..N-1] : integer$ con $n > 0$, se desea averiguar si hay en él una posición cuyo valor coincide con su índice (el programa devolverá una variable booleana). Establecer la precondición y la postcondición, la invariante, el programa y demostrar la corrección *parcial*.

La especificación formal sería:

$$\{Q: \ n > 0 \wedge b[0:n-1] : integer \wedge b = B\}$$
$$\{R: \ hay = (\exists J : 0 \leq J < n : b[J] = J) \wedge b = B\}$$

$$\{P: \ 0 \leq i \leq n \wedge hay = (\exists J : 0 \leq J < i : b[J] = J) \wedge b = B\}$$

$$S_0 : i, hay := 0, F;$$
$$\text{do } i \neq n \text{ cand } b[i] = i \rightarrow hay, i := T, i+1;$$
$$\ \mid \ i \neq n \text{ cand } b[i] \neq i \rightarrow i := i+1;$$
$$\text{od}$$

La demostración de corrección parcial:

1. $Q \Rightarrow wp(" i, hay := 0, F", P)$

   Calculamos el wp: $\equiv 0 \leq 0 \leq n \wedge F = \overbrace{(\exists J : 0 \leq J < 0 : b[J] = J)}^{F} \wedge b = B$

   $\equiv 0 \leq n \wedge T \wedge b = B$ que se deduce directamente de $Q : n > 0 \wedge b = B$.

2. $P \wedge B_1 \Rightarrow wp(" hay, i := T, i+1", P)$.

   Calculamos el wp: $\equiv \overbrace{0 \leq i+1 \leq n}^{\alpha} \wedge \underbrace{T = (\exists J : 0 \leq J < i+1 : b[J] = J)}_{\beta} \wedge b = B$

   α se puede concluir de $P : 0 \leq i \leq n$ y de $B_1 : i \neq n$.

   En β separamos el caso $J = i$. $T = (\exists J : 0 \leq J < i : b[J] = J) \vee (b[i] = i)$

   y como B_1 implica la segunda parte de la disyunción, toda la disyunción es cierta y β también es cierto.

   De esta forma tenemos demostrado que la implicación es cierta.

3. $P \wedge B_2 \Rightarrow wp(" i := i+1", P)$.

   Calculamos el wp: $\equiv \overbrace{0 \leq i+1 \leq n}^{\alpha} \wedge \underbrace{(\exists J : 0 \leq J < i+1 : b[J] = J)}_{\beta} \wedge b = B$

   La verdad de α se demuestra de la misma forma que en el apartado anterior.

   En β separamos el caso $J = i$. $hay = (\exists J : 0 \leq J < i : b[J] = J) \vee (b[i] = i)$

   y como la segunda parte de la disyunción es falsa por B_2, β queda como $hay = (\exists J : 0 \leq J < i : b[J] = J)$

   que es cierto por P.

4. $P \wedge \neg BB \Rightarrow R$

Calculando BB obtenemos: $BB \equiv i \neq n$ cand $\underbrace{(b[i] = i \vee b[i] \neq i)}_{T} \equiv i \neq n$

Por lo que $\neg BB$ es $i = n$. Por como hemos construido P, cuando $i = n$ se obtiene R de forma directa.