VERIFICACIÓN DE FUNCIONES RECURSIVAS

```
func f(x: T1) dev r: T2
{Pre: Q(x); Dec: t(x)}

var v: T2

    [ d(x)→ r:= h(x)
     ~d(x)→ v:= f(s(x));
            {Q(x) ∧ ~d(x) ∧ R(s(x),v)}
            r:= c(x,v)
    ]
{Post: R(x,r)}
```

Caso directo:

$Q(x) \wedge d(x) \Rightarrow R(x,h(x))$

Caso recursivo:

$s(x)$ se puede evaluar.

$Q(x) \wedge \sim d(x) \Rightarrow Q(s(x))$

$Q(x) \wedge \sim d(x) \wedge R(s(x),v) \Rightarrow R(x,c(x,v))$

Función de cota:

$Q(x) \Rightarrow t(x) \in N$

$Q(x) \wedge \sim d(x) \Rightarrow t(s(x)) < t(x)$

INMERSIÓN DE EFICIENCIA (RESULTADOS)

```
func fr(x: T1) dev r: T2, w: T3
{Pre: Q(x)}

var v: T2; u: T3

    [ d(x)→ <r,w>:= h'(x)
     ~d(x)→ <v,u>:= fr(s(x));
            {Q(x) ∧ ~d(x) ∧ R(s(x),v) ∧ u = ϕ(s(x))}
            <r,w>:= c'(x,v,u)
    ]
{Post: R(x,r) ∧ w = ϕ(x)}
```

INMERSIÓN DE EFICIENCIA (PARÁMETROS)

```
func fp(x: T1; w: T3) dev r: T2
{Pre: Q(x) ∧ w = ϕ(x)}

var v: T2

    [ d(x)→ r:=h'(x,w)
     ~d(x)→ v:= fp(s(x),ϕ(s(x)));
             {Q(x) ∧ ~d(x) ∧ R(s(x),v)}
             r:= c'(x,v,w)
    ]
{Post: R(x,r)}
```

INMERSIÓN Y RECURSIVIDAD FINAL: DESPLEGADO Y PLEGADO

```
g(y,w) = c(f(y),w)
```

Si la expresión *c* es asociativa:

```
func g(y: T1; w: T2) dev r: T2
{Pre: Q(y) ∧ D(w)}

    [ d(y)→ r:= c(h(y),w)
     ~d(y)→ r:= g(s(y), c(y,w))
    ]
{Post: r = c(f(y),w)}
```

OBTENCIÓN DE POSTCONDICIÓN CONSTANTE

```
func g(x: T1) dev r: T2
{Pre: Q(x)}

    [ d(x)→ r:= e(x)
     ~d(x)→ r:= g(s(x))
    ]
{Post: R(x,r)}
```

A partir de la función con recursividad final g, se obtiene la función con post. cte. g':

```
func g'(x,X: T1) dev r: T2
{Pre: Q(x) ∧ Q(X) ∧ (∀α: R(x,α) ⇒ R(X,α) )}

    [ d(x)→ r:= e(x)
     ~d(x)→ r:= g'(s(x),X)
    ]
{Post: R(X,r)}
```

Si R(x, r) es de la forma r = h(x):

```
(∀α: R(x,α) ⇒ R(X,α) ) ≡ (h(x) = h(X))
```