Teorema del invariante

Diego Bendersky, 6/9/2023

Momento, antes de empezar…

Pred?

Proc?

Aux?

Momento, antes de empezar…

Proc?

- Lo usamos para especificar un problema que después vamos a programar
- Indicamos cómo son los parámetros, los requiere y los asegura
- Puede tener “efectos secundarios” (cambiar los parámetros)

```
proc todosPares(inout s: seq<int>): bool
    requiere  true
    asegura   res = true ↔ (∀i : ℤ)(0 ≤ i < |s| → s[i] % 2 = 0)
```

$$\texttt{requiere} \quad true$$

$$\texttt{asegura} \quad res = true \leftrightarrow (\forall i : \mathbb{Z})(0 \leq i < |s| \rightarrow s[i] \,\%\, 2 = 0)$$

Momento, antes de empezar…

Pred?

- Los usamos para hacer más claros los predicados (requiere y asegura)
- Es **lo mismo** que reemplazar el predicado por su definición

```
proc tableroValido(inout t: seq<seq<tupla<int, bool>>): bool
    requiere
    asegura
```

requiere $true$

asegura $res = true \leftrightarrow esCuadrado(c)$

```
pred esCuadrado(t: seq<seq<tupla<int, bool>>) {
```

$(\forall i : \mathbb{Z})(0 \leq i < |t| \rightarrow |s[i]| = |s|)$

```
}
```

Momento, antes de empezar…

Aux?

- También los usamos para hacer más claros los predicados (requiere y asegura)
- Pueden devolver cualquier tipo
- Es **lo mismo** que reemplazar el predicado por su definición

```
proc tableroValido(inout t: seq<seq<tupla<int, bool>>): bool
```

requiere $true$

asegura $res = true \leftrightarrow cantBombas(t) < 10$

```
aux cantBombas(t: seq<seq<tupla<int, bool>>): int {
```

$$\sum_{i=0}^{|s|} \sum_{j=0}^{|s[i]|} if\ s[i][j] = true\ then\ 1\ else\ 0\ fi$$

```
}
```

Momento, antes de empezar…

No se puede meter un proc en un predicado

```
proc tableroValido(inout t: seq<seq<tupla<int, bool>>): bool
```

requiere $\cancel{tableroCuadrado(t)}$

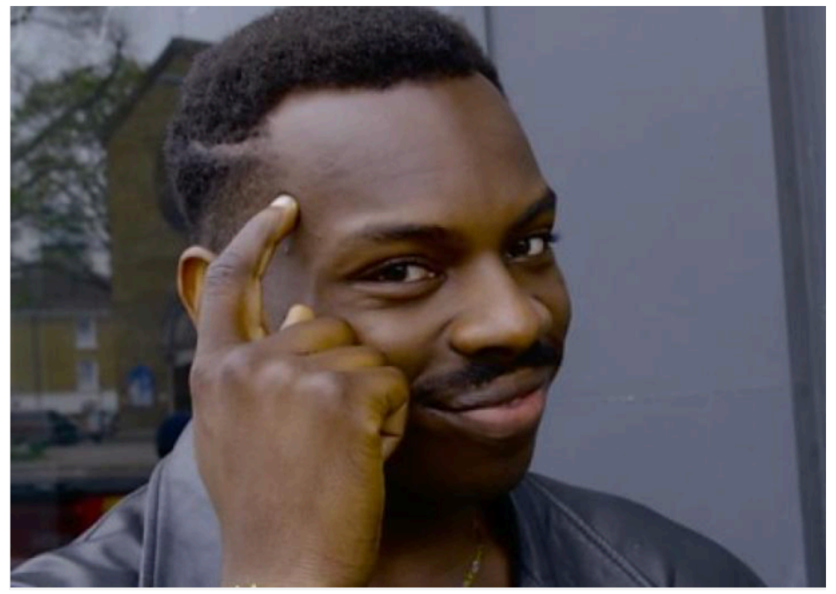
asegura $res = true \leftrightarrow cantBombas(t) < 10$

```
proc tableroCuadrado(inout t: seq<seq<tupla<int, bool>>): bool
```

asegura $res = true \leftrightarrow \ldots$

Momento, antes de empezar…

No se puede meter un proc en un predicado



```
proc tableroValido(inout t: seq<seq<tupla<int, bool>>): bool
    requiere    esCuadrado(t)
    asegura     res = true ↔ cantBombas(t) < 10

proc tableroCuadrado(inout t: seq<seq<tupla<int, bool>>): bool
    asegura     res = true ↔ esCuadrado(t)

pred esCuadrado(t: seq<seq<tupla<int, bool>>) {
    (∀i : ℤ)(0 ≤ i < |t| → |s[i]| = |s|)
}
```

Requiere: $esCuadrado(t)$

Asegura: $res = true \leftrightarrow cantBombas(t) < 10$

Asegura: $res = true \leftrightarrow esCuadrado(t)$

$(\forall i : \mathbb{Z})(0 \leq i < |t| \rightarrow |s[i]| = |s|)$

Teorema del invariante

Diego Bendersky, 6/9/2023

Repaso

- Aprendimos a verificar la correctitud de programas en SmallLang calculando wp

P

$E_3 = wp(S1, E_2)$

```
S1
```

$E_2 = wp(S2, E_1)$

```
S2
```

$E_1 = wp(S3, Q)$

```
S3
```

Q

De abajo hacia arriba

😳

Tenemos que demostrar que

$P \to E_3$

Repaso

- Si hay ciclos, todo se complica

P

$wp(ciclo, Q)$

```
while B do
      S1
      S2
      S3
endwhile
```

Q

No se puede!! 😭

- Tenemos que usar el “teorema del invariante"

Repaso

- Teorema del invariante
    - Hay que inventar un predicado, llamado invariante de ciclo
    - Hay que inventar una función variante

P

```
while B do
       S1
       S2
       S3
endwhile
```

Q

Repaso

- Si el invariante que inventamos cumple…

P

```
while B do
    S1
    S2
    S3
endwhile
```

Q

1. Vale al entrar $P \rightarrow I$
2. Vale al salir $I \wedge \neg B \rightarrow Q$
3. Vale al iniciar cada iteración $\{I \wedge B\} S \{I\}$

Entonces la tripla P {…ciclo…} Q es válida

OJO, no encontramos la wp! Sólo sabemos que el programa es correcto respecto de P y Q

Repaso

- Si la función variante que inventamos…

P

```
while B do
      S1
      S2
      S3
endwhile
```

Q

1. Siempre se achica $\{I \wedge B \wedge fv = v_0\} S \{fv < v_0\}$
2. Si vale cero (o menos) se termina $I \wedge fv \leq 0 \rightarrow \neg B$

Entonces el ciclo **siempre termina**

Ejercicio

- Indique si el siguiente código es correcto respecto de la especificación. Demuéstrelo.

$P_c : \{i = 0 \wedge res = true\}$

```
while (i < s.size()) do
   res := res && s[i] != 7
   i := i+1
end while
```

$Qc : \{res = true \leftrightarrow (\forall k : \mathbb{Z})(0 \leq k < |s| \rightarrow_L s[k] \neq 7$

1. Qué dice la especificación?
2. Qué hace el programa?
3. Les parece que cumple o no cumple?

Ejercicio

$P_c : \{i = 0 \wedge res = true\}$

```
while (i < s.size()) do
   res := res && s[i] != 7
   i := i+1
end while
```

$Qc : \{res = true \leftrightarrow (\forall k : \mathbb{Z})(0 \leq k < |s| \rightarrow_L s[k] \neq 7)$

Proponemos un invariante…

1. Vale al entrar $P_c \rightarrow I$
2. Vale al salir $I \wedge \neg B \rightarrow Q_c$
3. Vale al iniciar cada iteración $\{I \wedge \neg B\}S\{I\}$
4. OJO: no tiene por qué valer en el medio de una iteración

$I : \{res = true \leftrightarrow (\forall k : \mathbb{Z})(0 \leq k < i \rightarrow_L s[k] \neq 7)$

Falta algo!

$I : \{(0 \leq i \leq |s|) \wedge_L res = true \leftrightarrow (\forall k : \mathbb{Z})(0 \leq k < i \rightarrow_L s[k] \neq 7)$

Vale al entrar?

Tenemos que probar que:

$P_c : \{i = 0 \land res = true\} \quad \rightarrow \quad I : \{(0 \leq i \leq |s|) \land_L res = true \leftrightarrow (\forall k : \mathbb{Z})(0 \leq k < i \rightarrow_L s[k] \neq 7)$

Asumo que el precedente es verdadero y tengo que llegar a que el consecuente es verdadero

Si res=true, entonces

$$(0 \leq i \leq |s|) \land_L \boxed{res} = true \leftrightarrow (\forall k : \mathbb{Z})(0 \leq k < i \rightarrow_L s[k] \neq 7)$$

$$(0 \leq i \leq |s|) \land_L \boxed{true = true} \leftrightarrow (\forall k : \mathbb{Z})(0 \leq k < i \rightarrow_L s[k] \neq 7)$$

$$(0 \leq i \leq |s|) \land_L \boxed{true \leftrightarrow} (\forall k : \mathbb{Z})(0 \leq k < i \rightarrow_L s[k] \neq 7)$$

$$(0 \leq \boxed{i} \leq |s|) \land_L (\forall k : \mathbb{Z})(0 \leq k < \boxed{i} \rightarrow_L s[k] \neq 7)$$

Si i=0, entonces

$$(\boxed{0 \leq 0} \leq |s|) \land_L (\forall k : \mathbb{Z})(\boxed{0 \leq k < 0} \rightarrow_L s[k] \neq 7)$$

$$\boxed{(0 \leq |s|)} \land_L (\forall k : \mathbb{Z})(0 \leq k < 0 \rightarrow_L s[k] \neq 7)$$

$$\boxed{true \land_L} (\forall k : \mathbb{Z})(0 \leq k < 0 \rightarrow_L s[k] \neq 7)$$

$$(\forall k : \mathbb{Z})(\boxed{0 \leq k < 0} \rightarrow_L s[k] \neq 7)$$

$$(\forall k : \mathbb{Z})\boxed{(false \rightarrow_L s[k] \neq 7)}$$

$$\boxed{(\forall k : \mathbb{Z})(true)}$$

$$true$$

Asumimos que vale el precedente y llegamos a que vale el consecuente. Queda demostrado. ✓

Vale al salir?

$$I \wedge \neg B \quad \rightarrow \quad Q_c$$

$$(0 \leq i \leq |s|) \wedge_L res = true \leftrightarrow (\forall k : \mathbb{Z})(0 \leq k < i \rightarrow_L s[k] \neq 7) \wedge i \geq |s| \quad \rightarrow \quad res = true \leftrightarrow (\forall k : \mathbb{Z})(0 \leq k < |s| \rightarrow_L s[k] \neq 7)$$

Como tenemos que $i \leq |s|$ y que $i \geq |s|$ entonces vale $i = |s|$

$\boxed{(0 \leq i \leq |s|)} \wedge_L res = true \leftrightarrow (\forall k : \mathbb{Z})(0 \leq k < i \rightarrow_L s[k] \neq 7) \boxed{\wedge\, i \geq |s|}$

$\boxed{(0 \leq |s|)} \wedge_L res = true \leftrightarrow (\forall k : \mathbb{Z})(0 \leq k < |s| \rightarrow_L s[k] \neq 7) \wedge i = |s|$

$\boxed{true \wedge_L} res = true \leftrightarrow (\forall k : \mathbb{Z})(0 \leq k < |s| \rightarrow_L s[k] \neq 7) \wedge i = |s|$

$res = true \leftrightarrow (\forall k : \mathbb{Z})(0 \leq k < |s| \rightarrow_L s[k] \neq 7) \wedge i = |s|$

Me quedó igual que el consecuente (con algo de más)! Queda demostrado ✓

Vale el código del ciclo?

$$(0 \leq i \leq |s|) \wedge_L res = true \leftrightarrow (\forall k : \mathbb{Z})(0 \leq k < i \rightarrow_L s[k] \neq 7) \wedge i < |s|$$

```
res := res && s[i] != 7
i := i+1
```

$$(0 \leq i \leq |s|) \wedge_L res = true \leftrightarrow (\forall k : \mathbb{Z})(0 \leq k < i \rightarrow_L s[k] \neq 7)$$

Vale el código del ciclo?

$$\boxed{0 \leq i \leq |s|} \wedge_L res = true \leftrightarrow (\forall k : \mathbb{Z})(0 \leq k < i \rightarrow_L s[k] \neq 7) \wedge \boxed{i < |s|}$$

$$0 \leq i < |s| \wedge_L res = true \leftrightarrow (\forall k : \mathbb{Z})(0 \leq k < i \rightarrow_L s[k] \neq 7)$$

$$0 \leq i < |s| \wedge_L res \wedge s[i] \neq 7 = true \leftrightarrow (\forall k : \mathbb{Z})(0 \leq k \leq i \rightarrow_L s[k] \neq 7)$$

$$\boxed{0 \leq i < |s| \wedge_L (-1 \leq i < |s|)} \wedge_L res \wedge s[i] \neq 7 = true \leftrightarrow (\forall k : \mathbb{Z})(0 \leq k \leq i \rightarrow_L s[k] \neq 7)$$

$$\boxed{def(res \wedge s[i] \neq 7)} \wedge_L (-1 \leq i < |s|) \wedge_L res \wedge s[i] \neq 7 = true \leftrightarrow (\forall k : \mathbb{Z})(0 \leq k \leq i \rightarrow_L s[k] \neq 7)$$

```
res := res && s[i] != 7
```

$$(-1 \leq i < |s|) \wedge_L \boxed{res} = true \leftrightarrow (\forall k : \mathbb{Z})(0 \leq k \leq i \rightarrow_L s[k] \neq 7)$$

$$(-1 \leq i \boxed{\leq |s| - 1)} \wedge_L res = true \leftrightarrow (\forall k : \mathbb{Z})(0 \leq k \leq i \rightarrow_L s[k] \neq 7)$$

$$\boxed{def(i+1) \wedge} \boxed{(0 \leq i+1 \leq |s|)} \wedge_L res = true \leftrightarrow (\forall k : \mathbb{Z})(0 \leq k \boxed{< i+1} \rightarrow_L s[k] \neq 7)$$

```
i := i+1
```

$$(0 \leq \boxed{i} \leq |s|) \wedge_L res = true \leftrightarrow (\forall k : \mathbb{Z})(0 \leq k < \boxed{i} \rightarrow_L s[k] \neq 7)$$

- Vemos que el paratodo incluye el caso de s[i]
- Vamos a “sacar un caso del paratodo”

Disgresión: sacar un caso del paratodo

$$(\forall i : \mathbb{Z})P(i) \quad \equiv \quad \ldots \wedge P(-2) \wedge P(-1) \wedge P(0) \wedge P(1) \wedge P(2) \wedge \ldots$$

Quiero escribir el paratodo sin el caso P(0) y el caso P(0) aparte

$$\ldots \wedge P(-2) \wedge P(-1) \wedge P(1) \wedge P(2) \wedge \ldots \qquad \wedge P(0)$$

Cómo hacemos cuando necesitamos condiciones en el paratodo?

$$(\forall i : \mathbb{Z})(i \neq 0 \wedge P(i)) \quad ✗$$

$$(\forall i : \mathbb{Z})(i \neq 0 \rightarrow P(i)) \quad ✓$$

Entonces nos queda:

$$(\forall i : \mathbb{Z})P(i) \quad \equiv \quad (\forall i : \mathbb{Z})(i \neq 0 \rightarrow P(i)) \wedge P(0)$$

Vale el código del ciclo?

$$0 \leq i < |s| \wedge_L res = true \leftrightarrow (\forall k : \mathbb{Z})(0 \leq k < i \rightarrow_L s[k] \neq 7)$$

OJO! No se puede eliminar de los dos lados

$(P \wedge Q) \leftrightarrow (R \wedge Q)$ No es equivalente a $P \leftrightarrow R$

$$0 \leq i < |s| \wedge_L res \boxed{\wedge\, s[i] \neq 7} = true \leftrightarrow (\forall k : \mathbb{Z})(0 \leq k < i \rightarrow_L s[k] \neq 7) \boxed{\wedge\, s[i] \neq 7}$$

$$0 \leq i < |s| \wedge_L res \wedge s[i] \neq 7 = true \leftrightarrow (\forall k : \mathbb{Z})(0 \leq \boxed{k \leq i \wedge k \neq i} \rightarrow_L s[k] \neq 7) \wedge s[i] \neq 7$$

$$0 \leq i < |s| \wedge_L res \wedge s[i] \neq 7 = true \leftrightarrow \boxed{(\forall k : \mathbb{Z})(0 \leq k \leq i \rightarrow_L s[k] \neq 7)}$$

$$0 \leq i < |s| \wedge_L (-1 \leq i < |s|) \wedge_L res \wedge s[i] \neq 7 = true \leftrightarrow (\forall k : \mathbb{Z})(0 \leq k \leq i \rightarrow_L s[k] \neq 7)$$

$$def(res \wedge s[i] \neq 7) \wedge_L (-1 \leq i < |s|) \wedge_L res \wedge s[i] \neq 7 = true \leftrightarrow (\forall k : \mathbb{Z})(0 \leq k \leq i \rightarrow_L s[k] \neq 7)$$

```
res := res && s[i] != 7
```

$$(-1 \leq i < |s|) \wedge_L res = true \leftrightarrow (\forall k : \mathbb{Z})(0 \leq k \leq i \rightarrow_L s[k] \neq 7)$$

$$(-1 \leq i \leq |s| - 1) \wedge_L res = true \leftrightarrow (\forall k : \mathbb{Z})(0 \leq k \leq i \rightarrow_L s[k] \neq 7)$$

$$(0 \leq i + 1 \leq |s|) \wedge_L res = true \leftrightarrow (\forall k : \mathbb{Z})(0 \leq k < i + 1 \rightarrow_L s[k] \neq 7)$$

```
i := i+1
```

$$(0 \leq i \leq |s|) \wedge_L res = true \leftrightarrow (\forall k : \mathbb{Z})(0 \leq k < i \rightarrow_L s[k] \neq 7)$$

P	Q	R	P&Q	R&Q	P<->R	P&Q<->R*Q
T	T	T	T	T	T	T
T	T	F	T	F	F	F
T	F	T	F	F	T	T
T	F	F	F	F	F	T
F	T	T	F	T	F	F
F	T	F	F	F	T	T
F	F	T	F	F	F	T
F	F	F	F	F	T	T

Vale el código del ciclo?

$0 \leq i < |s| \wedge_L res = true \leftrightarrow (\forall k : \mathbb{Z})(0 \leq k < i \rightarrow_L s[k] \neq 7) \quad \rightarrow \quad 0 \leq i < |s| \wedge_L res \wedge s[i] \neq 7 = true \leftrightarrow (\forall k : \mathbb{Z})(0 \leq k < i \rightarrow_L s[k] \neq 7) \wedge s[i] \neq 7$

Asumo que el precedente es verdadero y tengo que llegar a que el consecuente es verdadero

Como $0 \leq i < |s|$ es verdadero:

$\boxed{0 \leq i < |s|} \wedge_L res \wedge s[i] \neq 7 = true \leftrightarrow (\forall k : \mathbb{Z})(0 \leq k < i \rightarrow_L s[k] \neq 7) \wedge s[i] \neq 7$

$\boxed{true} \wedge_L res \wedge s[i] \neq 7 = true \leftrightarrow (\forall k : \mathbb{Z})(0 \leq k < i \rightarrow_L s[k] \neq 7) \wedge s[i] \neq 7$

$res \wedge s[i] \neq 7 = true \leftrightarrow (\forall k : \mathbb{Z})(0 \leq k < i \rightarrow_L s[k] \neq 7) \wedge s[i] \neq 7$

Voy a probar dos casos: $s[i] \neq 7$ y $s[i] = 7$

Caso 1: $s[i] \neq 7$

$res \wedge \boxed{s[i] \neq 7} = true \leftrightarrow (\forall k : \mathbb{Z})(0 \leq k < i \rightarrow_L s[k] \neq 7) \wedge \boxed{s[i] \neq 7}$

$res \boxed{\wedge\, true} = true \leftrightarrow (\forall k : \mathbb{Z})(0 \leq k < i \rightarrow_L s[k] \neq 7) \wedge true$

$res = true \leftrightarrow (\forall k : \mathbb{Z})(0 \leq k < i \rightarrow_L s[k] \neq 7)$

Como $res = true \leftrightarrow (\forall k : \mathbb{Z})(0 \leq k < i \rightarrow_L s[k] \neq 7)$ es verdadero:

$\boxed{res = true \leftrightarrow (\forall k : \mathbb{Z})(0 \leq k < i \rightarrow_L s[k] \neq 7)}$

$true$

Caso 2: $s[i] = 7$

$res \wedge \boxed{s[i] \neq 7} = true \leftrightarrow (\forall k : \mathbb{Z})(0 \leq k < i \rightarrow_L s[k] \neq 7) \wedge \boxed{s[i] \neq 7}$

$res \wedge false = true \leftrightarrow \boxed{(\forall k : \mathbb{Z})(0 \leq k < i \rightarrow_L s[k] \neq 7) \wedge false}$

$res \boxed{\wedge\, false} \leftrightarrow false$

$\boxed{false \leftrightarrow false}$

$true$

Resumiendo

$P_c : \{i = 0 \land res = true\}$

```
while (i < s.size()) do
   res := res && s[i] != 7
   i := i+1
end while
```

$Qc : \{res = true \leftrightarrow (\forall k : \mathbb{Z})(0 \leq k < |s| \rightarrow_L s[k] \neq 7)$

$I : \{(0 \leq i \leq |s|) \land_L res = true \leftrightarrow (\forall k : \mathbb{Z})(0 \leq k < i \rightarrow_L s[k] \neq 7)$

Proponemos un invariante…

1. Vale al entrar $P_c \rightarrow I$ ✓
2. Vale al salir $I \land \neg B \rightarrow Q_c$ ✓
3. Vale al iniciar cada iteración $\{I \land \neg B\}S\{I\}$ ✓

Quedó demostrada la correctitud parcial

Faltaría la terminación…

Terminación

Proponemos una función variante…

1. Siempre se achica $\{I \wedge B \wedge fv = v_0\} S \{fv < v_0\}$
2. Si vale cero (o menos) se termina $I \wedge fv \leq 0 \rightarrow \neg B$
3. OJO. No depende de Pc ni de Qc

$P_c : \{i = 0 \wedge res = true\}$

```
while (i < s.size()) do
   res := res && s[i] != 7
   i := i+1
end while
```

$Qc : \{res = true \leftrightarrow (\forall k : \mathbb{Z})(0 \leq k < |s| \rightarrow_L s[k] \neq 7)$

$I : \{(0 \leq i \leq |s|) \wedge_L res = true \leftrightarrow (\forall k : \mathbb{Z})(0 \leq k < i \rightarrow_L s[k] \neq 7)$

$fv = |s| - i$

Siempre se achica?

$$(0 \leq i \leq |s|) \wedge_L res = true \leftrightarrow (\forall k : \mathbb{Z})(0 \leq k < i \rightarrow_L s[k] \neq 7) \wedge i < |s| \wedge fv = v_0$$

```
res := res && s[i] != 7
i := i+1
```

$$fv < v_0$$

Siempre se achica?

$$(0 \leq i \leq |s|) \wedge_L res = true \leftrightarrow (\forall k : \mathbb{Z})(0 \leq k < i \rightarrow_L s[k] \neq 7) \wedge i < |s| \wedge \boxed{fv} = v_0$$

$$(0 \leq i \leq |s|) \wedge_L res = true \leftrightarrow (\forall k : \mathbb{Z})(0 \leq k < i \rightarrow_L s[k] \neq 7) \wedge i < |s| \wedge |s| - i = v_0$$

$0 \leq i < |s| \wedge_L |s| - i \leq v_0$

$\boxed{def(res \wedge s[i] \neq 7)} \wedge_L |s| - i \leq v_0$

```
res := res && s[i] != 7
```

$|s| - i \leq v_0$

$|s| - i \boxed{< v_0 + 1}$

$\boxed{def(i+1) \wedge_L} |s| - \boxed{(i+1) < v_0}$

```
i := i+1
```

$|s| - \boxed{i} < v_0$

$\boxed{fv} < v_0$

Siempre se achica?

Quiero probar que:

$$(0 \leq i \leq |s|) \wedge_L res = true \leftrightarrow (\forall k : \mathbb{Z})(0 \leq k < i \rightarrow_L s[k] \neq 7) \wedge i < |s| \wedge |s| - i = v_0 \qquad \rightarrow \qquad 0 \leq i < |s| \wedge_L |s| - i \leq v_0$$

Asumo que el precedente es verdadero y llego a que el consecuente es verdadero

Como $0 \leq i \leq |s|$ es verdadero:

$\boxed{0 \leq i < |s|} \wedge_L |s| - i \leq v_0$

$\boxed{true \wedge_L} |s| - i \leq v_0$

$|s| - i \leq v_0$

Como $|s| - i = v_0$ es verdadero:

$\boxed{|s| - i \leq v_0}$

$true$

Si vale cero (o menos) termina

Quiero probar que:

$$(0 \leq i \leq |s|) \wedge_L res = true \leftrightarrow (\forall k : \mathbb{Z})(0 \leq k < i \rightarrow_L s[k] \neq 7) \wedge \boxed{|s| - i \leq 0} \quad \rightarrow \quad i \geq |s|$$

$$(0 \leq \boxed{i \leq |s|}) \wedge_L res = true \leftrightarrow (\forall k : \mathbb{Z})(0 \leq k < i \rightarrow_L s[k] \neq 7) \wedge \boxed{|s| \leq i}$$

$$(0 \leq i = |s|) \wedge_L res = true \leftrightarrow (\forall k : \mathbb{Z})(0 \leq k < i \rightarrow_L s[k] \neq 7) \wedge |s| \leq i$$

Asumo que el precedente es verdadero y llego a que el consecuente es verdadero

Según el precedente, $|s| = i$ Por lo que vale $i \geq |s|$

Resumiendo

Propusimos una función variante…

1. Siempre se achica $\{I \wedge B \wedge fv = v_0\} S \{fv < v_0\}$ ✓
2. Si vale cero (o menos) se termina $I \wedge fv \leq 0 \rightarrow \neg B$ ✓

✓

$P_c : \{i = 0 \wedge res = true\}$

```
while (i < s.size()) do
   res := res && s[i] != 7
   i := i+1
end while
```

$Qc : \{res = true \leftrightarrow (\forall k : \mathbb{Z})(0 \leq k < |s| \rightarrow_L s[k] \neq 7)$

$I : \{(0 \leq i \leq |s|) \wedge_L res = true \leftrightarrow (\forall k : \mathbb{Z})(0 \leq k < i \rightarrow_L s[k] \neq 7)$

$fv = |s| - i$

Quedó demostrada la terminación

Trabajo terminado!

Programas completos

- Qué pasa con esto?

```
S0
while B do
      S1
      S2
      S3
endwhile
S4
S5
```

P

```
i := 0
while i < s.size() do
      s[i] := 2 * s[i]
      i := i+1
endwhile
ret := a[0] = 0
```

Q

Programas completos

- Qué pasa con esto?

P

$P \rightarrow P_1$

P_1

```
i := 0
```

Q_1

$Q_1 \rightarrow P_2$

P_2

```
while i < s.size() do
       s[i] := 2 * s[i]
       i := i+1
endwhile
```

Q_2

$Q_2 \rightarrow P_3$

P_3

```
ret := a[0] = 0
```

Q_3

$Q_3 \rightarrow Q$

Q