20) *inversa(inversa(s)) = s (abril 2009 #1) -- #*

a) ++:: ([t], [t]) → [t]

[] ++ s = s (#1)
(x:r) ++ s = x:(r ++ s) (#2)

b) inversa:: ([t]) → [t]

inversa([]) = [] (#3)
inversa(x:r) = inversa(r) ++ (x:[]) (#4)

c) Ahora se probará por inducción sobre s la siguiente propiedad:
inversa(inversa(s)) = s

Caso básico: s = []

¿inversa(inversa([])) = []?

✓ inversa(inversa([])) =(#3)
= inversa([]) =(#3)
= []

Se cumple. Los dos lados de la igualdad tienen el mismo valor.

Caso inductivo: s = x:r

¿inversa(inversa(x:r)) = x:r?

Hipótesis de la inducción: (r cumple la propiedad) inversa(inversa(r)) = r

Volviendo a lo que se quiere probar: ¿inversa(inversa(x:r)) = x:r?

✓ inversa(inversa(x:r)) =(#4)
= inversa(inversa(r) ++ (x:[])) =(Prop)
= inversa(x:[]) ++ inversa(inversa(r)) =(#4)
= (inversa([]) ++ (x:[]))++ inversa(inversa(r)) =(#3)
= ([] ++ (x:[]))++ inversa(inversa(r)) =(#1)
= (x:[]) ++ inversa(inversa(r)) =(#2)
= x:([] ++ inversa(inversa(r))) =(#1)
= x:(inversa(inversa(r))) =(hi)
= x:r

Por tanto, sí se cumple la propiedad.

En cada paso se ha coloreado la parte de la fórmula a la que se le aplica la ecuación indicada en la esquina derecha de la línea. Se han utilizado dos colores para mejorar la legibilidad, por lo demás los colores distintos no tienen ningún significado.

21) *longitud(resto(s)) = longitud(sin_ultimo(s)) (abril 2009 #2) -- #*

a) longitud :: ([t]) → Int

longitud([]) = 0 (#1)
longitud(x:r) = 1 + longitud(r) (#2)

b) sin_ultimo:: ([t]) → [t]

sin_ultimo([]) = error "Lista vacía" (#3)
sin_ultimo(x:r)
| es_vacia(r) = [] (#4)
| otherwise = x: sin_ultimo(r) (#5)

c) resto:: ([t]) → [t]

resto([]) = error "Lista vacía" (#6)
resto(x:r) = r (#7)

d) Ahora se probará por inducción sobre s la siguiente propiedad:

longitud(resto(s)) = longitud(sin_ultimo(s))

Caso básico: s = z:[]

¿longitud(resto(z:[])) = longitud(sin_ultimo(z:[]))?

✓ longitud(resto(z:[])) =(#7)
= longitud([]) =(#1)
= 0

✓ longitud(sin_ultimo(z:[])) =(#4)
= longitud([]) =(#1)
= 0

Se cumple. Los dos lados de la igualdad tienen el mismo valor.

Caso inductivo: s = z:r   donde r no es vacía

¿longitud(resto(z:r)) = longitud(sin_ultimo(z:r))?

Hipótesis de la inducción: (r cumple la propiedad)

longitud(resto(r)) = longitud(sin_ultimo(r))

Volviendo a lo que se quiere probar:
¿longitud(resto(z:r)) = longitud(sin_ultimo(z:r))?

- ✓ longitud(resto(z:r)) =(#6)
  = longitud(r) =(Prop)
  = 1 + longitud(resto(r)) =(hi)
  = 1 + longitud(sin_ultimo(r))

- ✓ longitud(sin_ultimo(z:r)) =(#5)
  = longitud(z:sin_ultimo(r)) =(#2)
  = 1 + longitud(sin_ultimo(r))

Por tanto, sí se cumple la propiedad.

En cada paso se ha coloreado la parte de la fórmula a la que se le aplica la ecuación indicada en la esquina derecha de la línea. Se han utilizado dos colores para mejorar la legibilidad, por lo demás los colores distintos no tienen ningún significado.

22) *nveces(x, s) ≥ nveces(x, sin_ultimo(s)) (junio 2009) -- #*

a) nveces:: (t, [t]) → Int

nveces(x, []) = 0 (#1)
nveces(x, y:s)
|x == y = 1 + nveces(x, s) (#2)
|otherwise = nveces(x, s) (#3)

b) sin_ultimo:: ([t]) → [t]

sin_ultimo([]) = error "Lista vacía" (#4)
sin_ultimo(x:r)
| es_vacia(r) = [] (#5)
| otherwise = x: sin_ultimo(r) (#6)

c) Ahora se probará por inducción sobre s la siguiente propiedad (sabiendo que s no es vacía):

nveces(x, s) ≥ nveces(x, sin_ultimo(s))

Caso básico: s = z:[]

¿nveces(x, z:[]) ≥ nveces(x, sin_ultimo(z:[]))?

Hay dos posibilidades: x = z o x ≠ z

- **x = z**
  - ✓ nveces(x, z:[]) $=^{(\#2)}$
    = 1 + nveces(x, []) $=^{(\#1)}$
    = 1 + 0 = 1
  - ✓ nveces(x, sin_ultimo(z:[])) $=^{(\#5)}$
    = nveces(x, []) $=^{(\#1)}$
    = 0

    Se cumple porque 1 ≥ 0.
- **x ≠ z**
  - ✓ nveces(x, z:[]) $=^{(\#3)}$
    = nveces(x, []) $=^{(\#1)}$
    = 0
  - ✓ nveces(x, sin_ultimo(z:[])) $=^{(\#5)}$
    = nveces(x, []) $=^{(\#1)}$
    = 0

Se cumple porque 0 ≥ 0.
Por tanto, en el caso básico se cumple la propiedad.

Caso inductivo: s = z:r donde r es una lista no vacía

¿nveces(x, z:r) ≥ nveces(x, sin_ultimo(z:r))?

Hipótesis de la inducción: (x y r cumplen la propiedad)

nveces(x, r) ≥ nveces(x, sin_ultimo(r))

Volviendo a lo que se quiere probar:

¿nveces(x, z:r) ≥ nveces(x, sin_ultimo(z:r))?

Hay dos posibilidades: x = z o x ≠ z

- **x = z**

  - ✓ nveces(x, z:r) =$^{(\#2)}$
    = 1 + nveces(x, r)

  - ✓ nveces(x, sin_ultimo(z:r)) =$^{(\#6)}$
    = nveces(x, z:sin_ultimo(r)) =$^{(\#2)}$
    = 1 + nveces(x, sin_ultimo(r))

    Por hipótesis de la inducción se cumple

    nveces(x, r) ≥ nveces(x, sin_ultimo(r))

    y por tanto también se cumple
    1 + nveces(x, r) ≥ 1 + nveces(x, sin_ultimo(r))

  Por tanto, en este caso sí se cumple la propiedad.

- **x ≠ z**

  - ✓ nveces(x, z:r) =$^{(\#3)}$
    = nveces(x, r)

  - ✓ nveces(x, sin_ultimo(z:r)) =$^{(\#6)}$
    = nveces(x, z:sin_ultimo(r)) =$^{(\#3)}$
    = nveces(x, sin_ultimo(r))

    Por hipótesis de la inducción se cumple

    nveces(x, r) ≥ nveces(x, sin_ultimo(r))

  Por tanto, en este caso también se cumple la propiedad.

  La propiedad se cumple en el caso general.

23) *resto(inversa(s)) = inversa(sin_ultimo(s)) (septiembre 2009) -- #*

a) ++:: ([t], [t]) → [t]

[] ++ s = s (#1)
(x:r) ++ s = x:(r ++ s) (#2)

b) inversa:: ([t]) → [t]

inversa([]) = [] (#3)
inversa(x:r) = inversa(r) ++ (x:[]) (#4)

c) resto:: ([t]) → [t]

resto([]) = error "Lista vacía" (#5)
resto(x:r) = r (#6)

d) sin_ultimo:: ([t]) → [t]

sin_ultimo([]) = error "Lista vacía" (#7)
sin_ultimo(x:r)
| es_vacia(r) = [] (#8)
| otherwise = x: sin_ultimo(r) (#9)

e) Ahora se probará por inducción sobre s la siguiente propiedad (sabiendo que s no es vacía):

resto(inversa(s)) = inversa(sin_ultimo(s))

Caso básico: s = z:[]

¿resto(inversa(z:[])) = inversa(sin_ultimo(z:[]))?

✓ resto(inversa(z:[])) =(#4)
= resto(inversa([]) ++ (z:[])) =(#3)
= resto([] ++ (z:[])) =(#1)
= resto(z:[]) =(#6)
= []

✓ inversa(sin_ultimo(z:[])) =(#8)
= inversa([]) =(#3)
= []

En los dos lados se obtiene lo mismo.

Por tanto, en el caso básico se cumple la propiedad.

Caso inductivo: s = z:r donde r es una lista no vacía

¿resto(inversa(z:r)) = inversa(sin_ultimo(z:r))?

Hipótesis de la inducción (h.i.): (la lista no vacía r cumple la propiedad)

resto(inversa(r)) = inversa(sin_ultimo(r))

Volviendo a lo que se quiere probar:

¿resto(inversa(z:r)) = inversa(sin_ultimo(z:r))?

- ✓ resto(inversa(z:r)) =(#4)
  = resto(inversa(r) ++ (z:[])) =(como r no es vacía, por Prop2 sabemos que inversa(r) no es vacía ya que inversa(r) tiene la misma longitud que r y por tanto podemos aplicar Prop1 a la lista inversa(r) ++ (z:[]))
  = resto(inversa(r)) ++ (z:[])

- ✓ inversa(sin_ultimo(z:r)) =(#9)
  = inversa(z:sin_ultimo(r)) =(#4)
  = inversa(sin_ultimo(r)) ++ (z:[]) =(h.i.)
  = resto(inversa(r)) ++ (z:[])

Por tanto, se obtiene lo mismo en los dos lados.

La propiedad se cumple en el caso general.

24) *sumar(s) = sumar(inversa(s)) (Abril 2010 #1) -- #*

a) ++:: ([t], [t]) → [t]

[] ++ ℓ = ℓ (#1)
(*a*:q) ++ ℓ = *a*:(q ++ ℓ) (#2)

b) sumar:: ([Int]) → Int

sumar([]) = 0 (#3)
sumar(*a*:q) = *a* + sumar(q) (#4)

c) inversa:: ([t]) → [t]

inversa([]) = [] (#5)
inversa(*a*:q) = inversa(q) ++ (*a*:[]) (#6)

d) Se probará la siguiente propiedad aplicando inducción sobre s

sumar(s) = sumar(inversa(s))

Caso básico o caso simple: s = []

¿sumar([]) = sumar(inversa([]))?

✓ sumar([]) =(#3)
= 0

✓ sumar(inversa([])) =(#5)
= sumar([]) =(#3)
= 0

Se cumple. En los dos lados de la igualdad se obtiene el mismo valor.

Caso general o inductivo: s = x:r

sumar(x:r) = sumar(inversa(x:r))?

Hipótesis de la inducción: (la lista r cumple la propiedad)

sumar(r) = sumar(inversa(r))

Volviendo a lo que se quiere probar: sumar(x:r) = sumar(inversa(x:r))?

- ✓ sumar(x:r) $=^{(\#4)}$
  = x + sumar(r)

- ✓ sumar(inversa(x:r)) $=^{(\#4)}$
  = sumar(inversa(r) ++ (x:[])) $=^{(\text{Prop})}$
  = sumar(inversa(r)) + sumar(x:[]) $=^{(\#4)}$
  = sumar(inversa(r)) + (x + sumar([])) $=^{(\#3)}$
  = sumar(inversa(r)) + (x + 0) $=^{(\text{0 es el elemento neutro para la suma})}$
  = sumar(inversa(r)) + x $=^{(\text{h.i.})}$
  = sumar(r) + x $=^{(\text{la suma es conmutativa})}$
  = x + sumar(r)

Como se ha obtenido lo mismo en ambos lados de la igualdad, la propiedad se cumple también en el caso inductivo.

En cada paso se ha coloreado la parte de la fórmula a la que se le aplica la ecuación indicada en la esquina derecha de la línea. Se han utilizado dos colores para mejorar la legibilidad, por lo demás los colores distintos no tienen ningún significado.

25) *incr(inversa(s)) = inversa(incr(s)) (Abril 2010 #2) -- #*

a) ++:: ([t], [t]) → [t]

[] ++ h = h (#1)
(*a*:q) ++ h = *a*:(q ++ h) (#2)

b) incr:: ([Int]) → [Int]

incr([]) = 0 (#3)
incr(*a*:q) = (*a* + 1) : incr(q) (#4)

c) inversa:: ([t]) → [t]

inversa([]) = [] (#5)
inversa(*a*:q) = inversa(q) ++ (*a*:[]) (#6)

d) Hay que probar la siguiente propiedad aplicando inducción sobre la lista de enteros s:

¿incr(inversa(s)) = inversa(incr(s))

Caso básico o caso simple: s = []

incr(inversa([])) = inversa(incr([]))?

✓ incr(inversa([])) $=^{(\#5)}$
= incr([]) $=^{(\#3)}$
= []

✓ inversa(incr([])) $=^{(\#3)}$
= inversa([]) $=^{(\#5)}$
= []

Se cumple ya que en los dos lados de la igualdad se obtiene lo mismo.

Caso general o inductivo: s = x:r

¿incr(inversa(x:r)) = inversa(incr(x:r))?

Hipótesis de la inducción: (la lista r cumple la propiedad)

incr(inversa(r)) = inversa(incr(r))

Volviendo a lo que se quiere probar:

¿incr(inversa(x:r)) = inversa(incr(x:r))?

✓ incr(inversa(x:r)) =(#6)
= incr(inversa(r) ++ (x:[])) =(Prop)
= incr(inversa(r)) ++ incr(x:[]) =(#4)
= incr(inversa(r)) ++ ((x + 1): incr([])) =(#3)
= incr(inversa(r)) ++ ((x + 1): [])

✓ inversa(incr(x:r)) =(#4)
= inversa((x + 1):incr(r)) =(#6)
= inversa(incr(r)) ++ ((x + 1):[]) =(h.i.)
= incr(inversa(r)) ++ ((x + 1): [])

Como se ha obtenido lo mismo en ambos lados de la igualdad, la propiedad se cumple también en el caso inductivo.

En cada paso se ha coloreado la parte de la fórmula a la que se le aplica la ecuación indicada en la esquina derecha de la línea. Se han utilizado dos colores para mejorar la legibilidad, por lo demás los colores distintos no tienen ningún significado.

26) *nveces(x, inversa(s)) = nveces(x, s) (Junio 2010) -- #*

a) ++:: ([t], [t]) → [t]

[] ++ h = h (#1)
(*a*:q) ++ h = *a*:(q ++ h) (#2)

b) nveces:: (t, [t]) → Int

nveces(b, []) = 0 (#3)
nveces(b, a:q)
|b == a = 1 + nveces(b, q) (#4)
|otherwise = nveces(b, q) (#5)

c) inversa:: ([t]) → [t]

inversa([]) = [] (#6)
inversa(*a*:q) = inversa(q) ++ (*a*:[]) (#7)

d) Ahora se probará por inducción sobre s la siguiente propiedad:

nveces(x, inversa(s)) = nveces(x, s)

Caso básico: s = []

¿nveces(x, inversa([])) = nveces(x, [])?

✓ nveces(x, inversa([])) =(#6)
= nveces(x, []) =(#3)
= 0

✓ nveces(x, []) =(#3)
= 0

Se cumple porque 0 = 0.

Por tanto en el caso básico se cumple la propiedad.

Caso inductivo: s = z:r

¿nveces(x, inversa(z:r)) = nveces(x, z:r)?

Hipótesis de la inducción: (x y r cumplen la propiedad)

nveces(x, inversa(r)) = nveces(x, r)

Volviendo a lo que se quiere probar:

¿nveces(x, inversa(z:r)) = nveces(x, z:r)?

Hay dos posibilidades: x = z ó x ≠ z

- **x = z**

  - ✓ nveces(x, inversa(z:r)) $=^{(\#7)}$
    = nveces(x, inversa(r) ++ (z:[])) $=^{(Prop)}$
    = nveces(x, inversa(r)) + nveces(x, z:[]) $=^{(\#4)}$
    = nveces(x, inversa(r)) + 1 + nveces(x, []) $=^{(\#3)}$
    = nveces(x, inversa(r)) + 1 + 0 $=^{(0 \text{ es neutro para } +)}$
    = nveces(x, inversa(r)) + 1 $=^{(+ \text{ es conmutativa})}$
    = 1 + nveces(x, inversa(r))

  - ✓ nveces(x, z:r) $=^{(\#4)}$
    = 1 + nveces(x, r) $=^{(h.i.)}$
    = 1 + nveces(x, inversa(r))

  Por tanto en este caso sí se cumple la propiedad.

- **x ≠ z**

  - ✓ nveces(x, inversa(z:r)) $=^{(\#7)}$
    = nveces(x, inversa(r) ++ (z:[])) $=^{(Prop)}$
    = nveces(x, inversa(r)) + nveces(x, z:[]) $=^{(\#5)}$
    = nveces(x, inversa(r)) + nveces(x, []) $=^{(\#3)}$
    = nveces(x, inversa(r)) + 0 $=^{(0 \text{ es neutro para } +)}$
    = nveces(x, inversa(r))

  - ✓ nveces(x, z:r) $=^{(\#5)}$
    = nveces(x, r) $=^{(h.i.)}$
    = nveces(x, inversa(r))

  Por tanto, en este caso también se cumple la propiedad.

La propiedad se cumple en el caso general.

Nota: Al desarrollar la demostración, en cada paso se ha coloreado la parte de la fórmula a la que se le aplica la ecuación indicada en la esquina derecha de la línea. Se han utilizado dos colores para mejorar la legibilidad, por lo demás los colores distintos no tienen ningún significado.

27) *incr(s ++ r) = incr(s) ++ incr(r) (Septiembre 2010) -- #*

a) ++:: ([t], [t]) → [t]

[] ++ s = s (#1)
(x:r) ++ s = x:(r ++ s) (#2)

b) incr:: ([Int]) → [Int]

incr([]) = [] (#3)
incr(*a*:q) = (*a* + 1) : incr(q) (#4)

c) Ahora se probará por inducción sobre s la siguiente propiedad (sabiendo que s no es vacía):

incr(s ++ r) = incr(s) ++ incr(r)

Caso básico: s = []

¿incr([] ++ r) = incr([]) ++ incr(r)?

✓ incr([] ++ r) =(#1)
= incr(r)

✓ incr([]) ++ incr(r) =(#3)
= [] ++ incr(r) =(#1)
= incr(r)

En los dos lados se obtiene lo mismo: incr(r)

Por tanto, en el caso básico se cumple la propiedad.

Caso inductivo: s = z:w

¿incr((z:w) ++ r) = incr(z:w) ++ incr(r)?

Hipótesis de la inducción (h.i.): (las listas w y r cumplen la propiedad)

incr(w ++ r) = incr(w) ++ incr(r)

Volviendo a lo que se quiere probar:

¿incr((z:w) ++ r) = incr(z:w) ++ incr(r)?

✓ incr((z:w) ++ r) =$^{(\#2)}$
= incr(z:(w ++ r)) =$^{(\#4)}$
= (z + 1) : incr(w ++ r)

✓ incr(z:w) ++ incr(r) =$^{(\#4)}$
= ((z + 1) : incr(w)) ++ incr(r) =$^{(\#2)}$
= (z + 1) : (incr(w) ++ incr(r)) =$^{(h.i.)}$
= (z + 1) : incr(w ++ r)

Se obtiene lo mismo en los dos lados: (z + 1) : incr(w ++ r)

Por tanto, la propiedad se cumple también en el caso general.

Al desarrollar las expresiones de las igualdades planteadas, en cada paso se ha coloreado la parte de la fórmula a la que se le aplica la ecuación indicada en la esquina derecha de la línea. Se han utilizado distintos colores para mejorar la legibilidad.