R12.thy

De EjerciciosLMF2014

```
header {* R12: Cuantificadores sobre listas *}

theory R12
imports Main
begin

text {*
  ----------------------------------------------------------------------
  Ejercicio 1. Definir la función
     todos :: ('a ⇒ bool) ⇒ 'a list ⇒ bool
  tal que (todos p xs) se verifica si todos los elementos de la lista
  xs cumplen la propiedad p. Por ejemplo, se verifica
     todos (λx. 1<length x) [[2,1,4],[1,3]]
     ¬todos (λx. 1<length x) [[2,1,4],[3]]

  Nota: La función todos es equivalente a la predefinida list_all.
  ----------------------------------------------------------------------
*}

fun todos :: "('a ⇒ bool) ⇒ 'a list ⇒ bool" where
  "todos p xs = undefined"

text {*
  ----------------------------------------------------------------------
  Ejercicio 2. Definir la función
     algunos :: ('a ⇒ bool) ⇒ 'a list ⇒ bool
  tal que (algunos p xs) se verifica si algunos elementos de la lista
  xs cumplen la propiedad p. Por ejemplo, se verifica
     algunos (λx. 1<length x) [[2,1,4],[3]]
     ¬algunos (λx. 1<length x) [[],[3]]"

  Nota: La función algunos es equivalente a la predefinida list_ex.
  ----------------------------------------------------------------------
*}

fun algunos  :: "('a ⇒ bool) ⇒ 'a list ⇒ bool" where
  "algunos p xs = undefined"

text {*
  ----------------------------------------------------------------------
  Ejercicio 3.1. Demostrar o refutar automáticamente
     todos (λx. P x ∧ Q x) xs = (todos P xs ∧ todos Q xs)
  ----------------------------------------------------------------------
*}

lemma "todos (λx. P x ∧ Q x) xs = (todos P xs ∧ todos Q xs)"
oops

text {*
  ----------------------------------------------------------------------
  Ejercicio 3.2. Demostrar o refutar detalladamente
     todos (λx. P x ∧ Q x) xs = (todos P xs ∧ todos Q xs)
  ----------------------------------------------------------------------
*}

lemma "todos (λx. P x ∧ Q x) xs = (todos P xs ∧ todos Q xs)"
oops
```

```
text {*
  ---------------------------------------------------------------------
  Ejercicio 4.1. Demostrar o refutar automáticamente
     todos P (x @ y) = (todos P x ∧ todos P y)
  ---------------------------------------------------------------------
*}

lemma "todos P (x @ y) = (todos P x ∧ todos P y)"
oops

text {*
  ---------------------------------------------------------------------
  Ejercicio 4.2. Demostrar o refutar detalladamente
     todos P (x @ y) = (todos P x ∧ todos P y)
  ---------------------------------------------------------------------
*}

lemma todos_append:
  "todos P (x @ y) = (todos P x ∧ todos P y)"
oops

text {*
  ---------------------------------------------------------------------
  Ejercicio 5.1. Demostrar o refutar automáticamente
     todos P (rev xs) = todos P xs
  ---------------------------------------------------------------------
*}

lemma "todos P (rev xs) = todos P xs"
oops

text {*
  ---------------------------------------------------------------------
  Ejercicio 5.2. Demostrar o refutar detalladamente
     todos P (rev xs) = todos P xs
  ---------------------------------------------------------------------
*}

lemma "todos P (rev xs) = todos P xs"
oops

text {*
  ---------------------------------------------------------------------
  Ejercicio 6. Demostrar o refutar:
    algunos (λx. P x ∧ Q x) xs = (algunos P xs ∧ algunos Q xs)
  ---------------------------------------------------------------------
*}

lemma "algunos (λx. P x ∧ Q x) xs = (algunos P xs ∧ algunos Q xs)"
oops

text {*
  ---------------------------------------------------------------------
  Ejercicio 7.1. Demostrar o refutar automáticamente
     algunos P (map f xs) = algunos (P ∘ f) xs
  ---------------------------------------------------------------------
*}

lemma "algunos P (map f xs) = algunos (P o f) xs"
oops

text {*
  ---------------------------------------------------------------------
  Ejercicio 7.2. Demostrar o refutar datalladamente
     algunos P (map f xs) = algunos (P ∘ f) xs
```

```
  ------------------------------------------------------------------
*}

lemma "algunos P (map f xs) = algunos (P o f) xs"
oops

text {*
  ------------------------------------------------------------------
  Ejercicio 8.1. Demostrar o refutar automáticamente
     algunos P (xs @ ys) = (algunos P xs ∨ algunos P ys)
  ------------------------------------------------------------------
*}

lemma "algunos P (xs @ ys) = (algunos P xs ∨ algunos P ys)"
oops

text {*
  ------------------------------------------------------------------
  Ejercicio 8.2. Demostrar o refutar detalladamente
     algunos P (xs @ ys) = (algunos P xs ∨ algunos P ys)
  ------------------------------------------------------------------
*}

lemma algunos_append:
  "algunos P (xs @ ys) = (algunos P xs ∨ algunos P ys)"
oops

text {*
  ------------------------------------------------------------------
  Ejercicio 9.1. Demostrar o refutar automáticamente
     algunos P (rev xs) = algunos P xs
  ------------------------------------------------------------------
*}

lemma "algunos P (rev xs) = algunos P xs"
oops

text {*
  ------------------------------------------------------------------
  Ejercicio 9.2. Demostrar o refutar detalladamente
     algunos P (rev xs) = algunos P xs
  ------------------------------------------------------------------
*}

lemma "algunos P (rev xs) = algunos P xs"
oops

text {*
  ------------------------------------------------------------------
  Ejercicio 10. Encontrar un término no trivial Z tal que sea cierta la
  siguiente ecuación:
     algunos (λx. P x ∨ Q x) xs = Z
  y demostrar la equivalencia de forma automática y detallada.
  ------------------------------------------------------------------
*}

text {*
  ------------------------------------------------------------------
  Ejercicio 11.1. Demostrar o refutar automáticamente
     algunos P xs = (¬ todos (λx. (¬ P x)) xs)
  ------------------------------------------------------------------
*}

lemma "algunos P xs = (¬ todos (λx. (¬ P x)) xs)"
oops
```

```
text {*
  ------------------------------------------------------------------
  Ejercicio 11.2. Demostrar o refutar datalladamente
     algunos P xs = (¬ todos (λx. (¬ P x)) xs)
  ------------------------------------------------------------------
*}

lemma "algunos P xs = (¬ todos (λx. (¬ P x)) xs)"
oops

text {*
  ------------------------------------------------------------------
  Ejercicio 12. Definir la funcion primitiva recursiva
     estaEn :: 'a ⇒ 'a list ⇒ bool
  tal que (estaEn x xs) se verifica si el elemento x está en la lista
  xs. Por ejemplo,
     estaEn (2::nat) [3,2,4] = True
     estaEn (1::nat) [3,2,4] = False
  ------------------------------------------------------------------
*}

fun estaEn :: "'a ⇒ 'a list ⇒ bool" where
  "estaEn x xs = undefined"

text {*
  ------------------------------------------------------------------
  Ejercicio 13. Expresar la relación existente entre estaEn y algunos.
  Demostrar dicha relación de forma automática y detallada.
  ------------------------------------------------------------------
*}
```

```
text {*
  ------------------------------------------------------------------
  Ejercicio 14. Definir la función primitiva recursiva
     sinDuplicados :: 'a list ⇒ bool
  tal que (sinDuplicados xs) se verifica si la lista xs no contiene
  duplicados. Por ejemplo,
     sinDuplicados [1::nat,4,2]   = True
     sinDuplicados [1::nat,4,2,4] = False
  ------------------------------------------------------------------
*}

fun sinDuplicados :: "'a list ⇒ bool" where
  "sinDuplicados xs = undefined"

text {*
  ------------------------------------------------------------------
  Ejercicio 15. Definir la función primitiva recursiva
     borraDuplicados :: 'a list ⇒ bool
  tal que (borraDuplicados xs) es la lista obtenida eliminando los
  elementos duplicados de la lista xs. Por ejemplo,
     borraDuplicados [1::nat,2,4,2,3] = [1,4,2,3]

  Nota: La función borraDuplicados es equivalente a la predefinida
  remdups.
  ------------------------------------------------------------------
*}

fun borraDuplicados :: "'a list ⇒ 'a list" where
  "borraDuplicados xs = undefined"

text {*
  ------------------------------------------------------------------
  Ejercicio 16.1. Demostrar o refutar automáticamente
```

```
     length (borraDuplicados xs) ≤ length xs
  ----------------------------------------------------------------------
*}

lemma length_borraDuplicados:
  "length (borraDuplicados xs) ≤ length xs"
oops

text {*
  ----------------------------------------------------------------------
  Ejercicio 16.2. Demostrar o refutar detalladamente
     length (borraDuplicados xs) ≤ length xs
  ----------------------------------------------------------------------
*}

lemma length_borraDuplicados:
  "length (borraDuplicados xs) ≤ length xs"
oops

text {*
  ----------------------------------------------------------------------
  Ejercicio 17.1. Demostrar o refutar automáticamente
     estaEn a (borraDuplicados xs) = estaEn a xs
  ----------------------------------------------------------------------
*}

lemma "estaEn a (borraDuplicados xs) = estaEn a xs"
oops

text {*
  ----------------------------------------------------------------------
  Ejercicio 17.2. Demostrar o refutar detalladamente
     estaEn a (borraDuplicados xs) = estaEn a xs
  ----------------------------------------------------------------------
*}

lemma estaEn_borraDuplicados:
  "estaEn a (borraDuplicados xs) = estaEn a xs"
oops

text {*
  ----------------------------------------------------------------------
  Ejercicio 18.1. Demostrar o refutar automáticamente
     sinDuplicados (borraDuplicados xs)
  ----------------------------------------------------------------------
*}

lemma "sinDuplicados (borraDuplicados xs)"
oops

text {*
  ----------------------------------------------------------------------
  Ejercicio 18.2. Demostrar o refutar detalladamente
     sinDuplicados (borraDuplicados xs)
  ----------------------------------------------------------------------
*}

lemma sinDuplicados_borraDuplicados:
  "sinDuplicados (borraDuplicados xs)"
oops

text {*
  ----------------------------------------------------------------------
  Ejercicio 19. Demostrar o refutar:
    borraDuplicados (rev xs) = rev (borraDuplicados xs)
```

```
  ----------------------------------------------------------------------
*}

lemma "borraDuplicados (rev xs) = rev (borraDuplicados xs)"
oops

end
```

Obtenido de "http://www.glc.us.es/~jalonso/ejerciciosLMF2014/index.php5/R12.thy"