# Razonamiento automático
# Práctica 4

**Dpto. de Ciencias de la Computación e Inteligencia Artificial**

Universidad de Sevilla

# Verificando `mergesort`

- **Definicion del algoritmo**
  - **Función `merge`.**
  - **Función `alternos`.**
  - **Función `mergesort`.**
  - **Indicaciones para la admisión.**
- **Verificación, paso 1:**
  - **El resultado que devuelve `mergesort` es una lista ordenada.**
- **Verificación, paso 2:**
  - **El resultado que devuelve `mergesort` es una lista con los mismos elementos que la original.**

# La función merge

- **Definición:**

```
(defn merge (l m)
  (if (not (listp l))
      m
    (if (not (listp m))
        l
      (if (lessp (car l) (car m))
          (cons (car l) (merge (cdr l) m))
        (cons (car m) (merge l (cdr m))))))
  ............)
```

- **Inicialmente no admitida:**

```
ERROR:  The admissibility of this definition
has not been established.  The theorem prover's
heuristics found no plausible measure to justify the
recursion.  In particular, no single argument of
the function is both tested in each branch and changed
in each recursive call.  The definition is rejected.
...
```

# Admisión de merge y alternos.

- **Definición de** longitud**:**

```
(defn longitud (l)
  (if (nlistp l)
      0
    (add1 (longitud (cdr l)))))
```

- **Indicación para la admisión de** merge**:**

```
(defn merge (l m)
  (if (not (listp l))
      m
    (if (not (listp m))
        l
      (if (lessp (car l) (car m))
          (cons (car l) (merge (cdr l) m))
        (cons (car m) (merge l (cdr m))))))
  ((lessp (plus (longitud l) (longitud m)))))

...Linear arithmetic, the lemma SUB1-ADD1,
and the definitions of LESSP, PLUS, and LONGITUD
inform us that the measure
(PLUS (LONGITUD L) (LONGITUD M))
decreases according to the well-founded relation LESSP
in each recursive call. Hence, MERGE is accepted under
the principle of definition....
```

- **La función** alternos

```
(defn alternos (l)
  (if (not (listp l))
      nil
    (cons (car l) (alternos (cddr l)))))
```

# La función mergesort.

- **La función** mergesort**:**

```
(defn mergesort (l)
  (if (or (nlistp l) (nlistp (cdr l)))
        l
      (merge (mergesort (alternos (cdr l)))
             (mergesort (alternos l)))))
```

- **No admitida:**

```
ERROR:  The admissibility of this definition
has not been established.  The simplifier could
not prove that the measure(s) tried decrease in each
recursive call.  The definition is rejected.
Below are listed the relations and measures
tried and some of the unproved goals for each.

Relation:  LESSP
Measure:   (COUNT L)
Unproved goals:
      (IMPLIES (AND (LISTP L) (LISTP (CDR L)))
               (LESSP (COUNT (ALTERNOS L))
                      (COUNT L)))
...
```

# Admisión de la función mergesort.

- **Indicación:**

```
(defn mergesort (l)
  (if (or (nlistp l) (nlistp (cdr l)))
        l
      (merge (mergesort (alternos (cdr l)))
             (mergesort (alternos l))))
  ((lessp (longitud l))))
```

- **Aún no es admitida:**

```
ERROR:  The admissibility of this definition
has not been established.  The simplifier could
not prove that the measure(s) tried decrease in each
recursive call.  The definition is rejected.
Below are listed the relations and measures tried
and some of the unproved goals for each.

Relation:  LESSP
Measure:   (LONGITUD L)
Unproved goals:
 (IMPLIES (AND(LISTP L)
               (LISTP (CDR L))
               (NOT (EQUAL (LONGITUD (ALTERNOS L))0)))
          (LESSP (SUB1 (LONGITUD (ALTERNOS L)))
                 (LONGITUD (CDR L))))
```

# Admisión de la función mergesort.

- **Necesitamos probar:**

```
1)
  (implies (and (listp l)
                (listp (cdr l)))
           (lessp (longitud (alternos l))
                  (longitud l)))
2)
  (implies (and (listp l)
                (listp (cdr l)))
           (lessp (longitud (alternos (cdr l)))
                  (longitud l)))
```

- **El primer lema se prueba:**

```
(prove-lemma mergesort-ayuda-1 (rewrite)
  (implies (and (listp l)
                (listp (cdr l)))
           (lessp (longitud (alternos l))
                  (longitud l))))
...
MERGESORT-AYUDA-1
```

- **El segundo lema NO se prueba:**

```
(prove-lemma mergesort-ayuda-2 (rewrite)
  (implies (and (listp l)
                (listp (cdr l)))
           (lessp (longitud (alternos (cdr l)))
                  (longitud l))))
...
************** F  A  I  L  E  D **************
```

# Admisión de la función mergesort.

- **Un segundo intento:**

  Si sabemos que, bajo ciertas condiciones,
  (LESSP (LONGITUD (ALTERNOS (CDDR L)))
         (LONGITUD (CDR L)))  (*)

  ya que esto es un caso particular del
  lema mergesort-ayuda-1.

  Entonces, como bajo ciertas condiciones,
  (LESSP (LONGITUD (CDR L)) (LONGITUD L)),
  se tendrá mergesort-ayuda-2

  Pero (*) es la hipótesis de inducción
  que se usa en la prueba de mergesort-ayuda-1

  IDEA: Probar los dos resultados a la vez.

- **Deshacemos** mergesort-ayuda-1

```
>(ubt)
MERGESORT-AYUDA-1
```

# Admisión de la función mergesort.

- **Ahora sí:**

```
(prove-lemma mergesort-ayuda (rewrite)
  (implies (and (listp l)
                (listp (cdr l)))
           (and (lessp (longitud (alternos l))
                       (longitud l))
                (lessp (longitud (alternos (cdr l)))
                       (longitud l)))))
```

- **Se obtienen dos lemas de reescritura.**
- **Nuevo intento de definición:**

```
(defn mergesort (l)
  (if (or (nlistp l) (nlistp (cdr l)))
      l
      (merge (mergesort (alternos (cdr l)))
             (mergesort (alternos l))))
  ((lessp (longitud l))))
```

- **Gracias al lema anterior, la función se admite.**

```
...
     Linear arithmetic, the lemmas SUB1-ADD1 and
MERGESORT-AYUDA, and the definitions of LESSP,
LONGITUD, OR, and NLISTP inform us that the measure
(LONGITUD L) decreases according to the well-founded
relation LESSP in each recursive call.
Hence, MERGESORT is accepted under the definitional
principle.
...
MERGESORT
```

# La función mergesort ordena.

- **Definición de** ordenada**:**

```
(defn ordenada (l)
  (if (or (nlistp l) (nlistp (cdr l)))
      t
    (and (not (lessp (cadr l) (car l)))
         (ordenada (cdr l)))))
```

- mergesort **ordena:**

```
(prove-lemma ordenada-mergesort (rewrite)
  (ordenada (mergesort x)))
```

- **Demostración con más de una inducción:**

```
...
    (IMPLIES (AND (ORDENADA B) (ORDENADA U))
             (ORDENADA (MERGE U B))),
  which we will finally name *1.1.
...

;;; Se puede mejorar la demostración, hacemos UBT
;;; y probamos este resultado previamente.
```

- **Mejorando la demostración:**

```
(prove-lemma ordenada-merge (rewrite)
  (implies (and (ordenada x) (ordenada y))
           (ordenada (merge x y))))

(prove-lemma ordenada-mergesort (rewrite)
  (ordenada (mergesort x)))
```

# `mergesort` conserva los elementos.

- **Número de veces que `x` aparece en `l`:**

```
(defn apariciones (x l)
  (cond ((nlistp l) 0)
        ((equal x (car l))
         (add1 (apariciones x (cdr l))))
        (t (apariciones x (cdr l)))))
```

- **Hay que probar:**

```
(prove-lemma mismos-elementos-mergesort (rewrite)
  (equal (apariciones x (mergesort l))
         (apariciones x l)))
```

- **En el primer intento, no se prueba**
- **Cortamos la ejecución:**

```
Correctable error: Console interrupt.
Signalled by SIMPLIFY-SENT.
If continued:
Type :r to resume execution, or :q to quit to top level.
Broken at APPLY.  Type :H for Help.
>>
```

# Buscando lemas de ayuda.

- **Paso peligroso:**

```
....
(IMPLIES
   (AND (LISTP Z)
        (EQUAL (APARICIONES X
                            (MERGESORT (ALTERNOS (CONS V Z))))
               (APARICIONES X (ALTERNOS (CONS V Z))))
        (EQUAL (APARICIONES X
                            (MERGESORT (ALTERNOS Z)))
               (APARICIONES X (ALTERNOS Z)))
        (NOT (EQUAL X V)))
   (EQUAL (APARICIONES X
                       (MERGE (MERGESORT (ALTERNOS Z))
                              (MERGESORT (ALTERNOS (CONS V Z)))))
          (APARICIONES X Z))).

    We will try to prove the above formula by generalizing it, replacing
    (ALTERNOS Z) by Y and (ALTERNOS (CONS V Z)) by A.  We must thus prove:
...
```

- **Hay que simplificar la fórmula con este lema:**

```
(prove-lemma apariciones-merge (rewrite)
  (equal (apariciones x (merge a b))
         (plus (apariciones x a)
               (apariciones x b))))
```

# Buscando lemas de ayuda.

- **Segundo intento, vuelve a fallar:**
- **Paso peligroso:**

```
(IMPLIES (AND (LISTP Z)
              (EQUAL (APARICIONES X
                                  (MERGESORT (ALTERNOS (CONS V Z))))
                     (APARICIONES X (ALTERNOS (CONS V Z))))
              (EQUAL (APARICIONES X
                                  (MERGESORT (ALTERNOS Z)))
                     (APARICIONES X (ALTERNOS Z)))
              (NOT (EQUAL X V)))
         (EQUAL (PLUS (APARICIONES X (ALTERNOS Z))
                      (APARICIONES X (ALTERNOS (CONS V Z))))
                (APARICIONES X Z))).

   We use the first equality hypothesis by substituting:
         (APARICIONES X
                      (MERGESORT (ALTERNOS (CONS V Z))))
   for (APARICIONES X (ALTERNOS (CONS V Z))) and throwing away the equality.
...
```

- **Hay que simplificar la fórmula con este lema:**

```
(prove-lemma plus-numero-de-ocurrencias-alterno (rewrite)
  (equal (plus (apariciones x (alternos (cdr l)))
               (apariciones x (alternos l)))
         (apariciones x l)))
```

# Fin de la prueba.

- **Ya se prueba:**

```
(prove-lemma mismos-elementos-mergesort (rewrite)
  (equal (apariciones x (mergesort l))
         (apariciones x l)))
```

- **Efectivamente, se usan los lemas anteriores:**

```
....
Case 1. (IMPLIES (AND (NOT (OR (NLISTP L) (NLISTP (CDR L))))
                      (EQUAL (APARICIONES X
                                          (MERGESORT (ALTERNOS L)))
                             (APARICIONES X (ALTERNOS L)))
                      (EQUAL (APARICIONES X
                                          (MERGESORT (ALTERNOS (CDR L))))
                             (APARICIONES X (ALTERNOS (CDR L)))))
                 (EQUAL (APARICIONES X (MERGESORT L))
                        (APARICIONES X L))).

  This simplifies, rewriting with PLUS-NUMERO-DE-OCURRENCIAS-ALTERNO and
  APARICIONES-MERGE, and unfolding the definitions of NLISTP, OR, MERGESORT,
  and APARICIONES, to:

        T.
....
```

# La prueba completa (I).

```
(boot-strap nqthm)

(defn longitud (l)
  (if (nlistp l)
      0
    (add1 (longitud (cdr l)))))

(defn merge (l m)
  (if (not (listp l))
      m
    (if (not (listp m))
        l
      (if (lessp (car l) (car m))
          (cons (car l) (merge (cdr l) m))
        (cons (car m) (merge l (cdr m))))))
  ((lessp (plus (longitud l) (longitud m)))))

(defn alternos (l)
  (if (not (listp l))
      nil
    (cons (car l) (alternos (cddr l)))))

(prove-lemma mergesort-ayuda (rewrite)
  (implies (and (listp l)
                (listp (cdr l)))
           (and (lessp (longitud (alternos l))
                       (longitud l))
                (lessp (longitud (alternos (cdr l)))
                       (longitud l)))))
```

# La prueba completa (II).

```
(defn mergesort (l)
  (if (or (nlistp l) (nlistp (cdr l)))
        l
      (merge (mergesort (alternos (cdr l)))
             (mergesort (alternos l))))
  ((lessp (longitud l))))

(defn ordenada (l)
  (if (or (nlistp l) (nlistp (cdr l)))
      t
    (and (not (lessp (cadr l) (car l)))
         (ordenada (cdr l)))))

(prove-lemma ordenada-merge (rewrite)
  (implies (and (ordenada x) (ordenada y))
           (ordenada (merge x y))))

(prove-lemma ordenada-mergesort (rewrite)
  (ordenada (mergesort x)))
```

# La prueba completa (y III).

```
(defn apariciones (x l)
  (cond ((nlistp l) 0)
        ((equal x (car l)) (add1 (apariciones x (cdr l))))
        (t (apariciones x (cdr l)))))

(prove-lemma apariciones-merge (rewrite)
  (equal (apariciones x (merge a b))
         (plus (apariciones x a)
               (apariciones x b))))

(prove-lemma plus-numero-de-ocurrencias-alterno (rewrite)
  (equal (plus (apariciones x (alternos (cdr l)))
               (apariciones x (alternos l)))
         (apariciones x l)))

(prove-lemma mismos-elementos-mergesort (rewrite)
  (equal (apariciones x (mergesort l))
         (apariciones x l)))
```