ACL2: Recursión e inducción

Francisco J. Martín Mateos
José L. Ruiz Reina

Dpto. Ciencias de la Computación e Inteligencia Artificial
Universidad de Sevilla

Principio de definición

- La lógica se extiende dinámicamente, añadiendo nuevas definiciones de funciones mediante *el principio de definición*, usando **`defun`**
  - Cada definición añade nuevos axiomas a la lógica
  - La consistencia de la lógica se debe mantener
- Sólo se admiten funciones bajo ciertas restricciones
  - Sólo se puede hacer referencia a funciones previamente definidas
  - Las únicas variables de la definición han de ser las que aparecen como sus argumentos o las introducidas mediante entornos locales
  - Sólo se admiten funciones que terminan para cualquier dato de entrada
- Esto asegura extensiones consistentes y conservativas de la lógica

Principio de definición

- Dada una historia $\mathcal{H}$ (secuencia de definiciones previas), la definición **`(defun f (x1 ... xn) cuerpo)`** es admisible respecto de $\mathcal{H}$ si
  - **`f`** es un símbolo de función nuevo (no aparece en $\mathcal{H}$),
  - cada **`xi`** es un símbolo de variable distinto,
  - **`cuerpo`** es un término en el lenguaje de $\mathcal{H}$ ampliado con el símbolo **`f`** de aridad **`n`**, cuyas variables libres están entre las **`xi`**,
  - se pueden demostrar las *conjeturas de terminación* de **`f`**
- Si la definición es aceptada, se introduce el axioma

$$\texttt{(f x1 ... xn)} = \texttt{cuerpo}$$

Principio de definición

- No se permiten definiciones por recursión doble

```
(defun par (n)
  (if (zp n)
      t
      (impar (- n 1))))

(defun impar (n)
  (if (zp n)
      nil
      (par (- n 1))))
```

- La primera definición incluiría un axioma que hace referencia a un símbolo de función desconocido !!

Principio de definición

- No se permite el uso de variables globales

```
(defun g (x)
  y)
```

- Esta definición añadiría el siguiente axioma a la lógica

$$\texttt{(g x)} = \texttt{y}$$

donde `y` es una variable global, cuyo valor puede cambiar después de la definición de `g`. Este cambio de valor puede hacer que el axioma anterior produzca una inconsistencia en el sistema

Principio de definición

- El propio lenguaje de programación fuerza a que las funciones tomen un valor para cualquier dato de entrada (funciones totales)
  - Todas las funciones predefinidas en el sistema son totales
  - Sólo se permite la versión completa del condicional **`if`**

```
(if <condicion>
    <instruccion>
    <alternativa>)
```

  - El caso final del condicional amplio **`cond`** ha de ser el caso por defecto **`t`**

```
(cond (<condicion-1> <cuerpo-1>)
      ...
      (<condicion-n> <cuerpo-n>)
      (t <alternativa>))
```

Principio de definición

- No se permiten las recursiones infinitas

```
(defun f (x)
  (+ 1 (f x)))
```

- Esta definición añadiría el siguiente axioma a la lógica

$$\texttt{(f x)} = \texttt{(+ 1 (f x))}$$

- El uso de esta igualdad como regla de simplificación introduciría el sistema en un bucle infinito

Conjeturas de terminación

- Dada una historia $\mathcal{H}$ (secuencia de definiciones previas), las conjeturas de terminación de la definición

  (defun f (x1 ... xn) cuerpo)

  aseguran que existe una *medida de terminación* **m** en el lenguaje de $\mathcal{H}$, respecto de la cual se tienen

  - **m** es la representación en ACL2 de un ordinal

    ```
    (o-p m)
    ```

  - Por cada ocurrencia en **cuerpo** de un subtérmino **(f u1 ... un)** se puede demostrar la fórmula

    ```
    (implies (and t1 ... tk)
             (o< σ(m) m))
    ```

    donde **t1**,...,**tk** son los términos que influyen en dicha ocurrencia y σ es la sustitución $\{$**x1**/**u1**,...,**xn**/**un**$\}$

Admisión de funciones recursivas

- La función factorial

```
(defun factorial (n)
  (if (zp n)
      1
      (* n (factorial (- n 1)))))
```

- Medida de terminación `m` = `(if (natp n) n 0)`
  - Hay una única llamada recursiva `(factorial (- n 1))` y el término que influye en ella es `(not (zp n))`
  - La conjetura de terminación es

```
(implies (not (zp n))
         (o< (if (natp (- n 1)) (- n 1) 0)
             (if (natp n) n 0)))
```

Admisión de funciones recursivas

- Concatenación de listas

```
(defun concatena (l1 l2)
  (if (endp l1)
      l2
      (cons (car l1) (concatena (cdr l1) l2))))
```

- Medida de terminación `m` = `(len l1)`
  - Hay una única llamada recursiva `(concatena (cdr l1) l2)` y el término que influye en ella es `(not (endp l1))`
  - La conjetura de terminación es

```
(implies (not (endp l1))
         (o< (len (cdr l1)) (len l1)))
```

Admisión de funciones recursivas

- Listas ordenadas

```
(defun ordenada (l)
  (cond ((endp l) (equal l nil))
        (t (or (endp (cdr l))
               (and (<= (car l) (cadr l))
                    (ordenada (cdr l)))))))
```

- Medida de terminación `m` = `(len l)`
  - Hay una única llamada recursiva `(ordenada (cdr l))` y los términos que influyen en ella son `(not (endp l))`, `(not (endp (cdr l)))` y `(<= (car l) (cadr l))`
  - La conjetura de terminación es

```
(implies (and (not (endp l))
              (not (endp (cdr l)))
              (<= (car l) (cadr l)))
         (o< (len (cdr l)) (len l)))
```

Admisión de funciones recursivas

- Mezcla ordenada de dos listas

```
(defun mezcla (l1 l2)
  (cond ((endp l1) l2)
        ((endp l2) l1)
        ((< (car l1) (car l2))
         (cons (car l1) (mezcla (cdr l1) l2)))
        (t (cons (car l2) (mezcla l1 (cdr l2))))))
```

- Medida de terminación `m` = `(+ (len l1) (len l2))`
- Hay dos llamadas recursivas
  - `(mezcla (cdr l1) l2)`
  - `(mezcla l1 (cdr l2))`

Admisión de funciones recursivas

- La primera llamada recursiva es `(mezcla (cdr l1) l2)` y los términos que influyen en ella son `(not (endp l1))`, `(not (endp l2))` y `(< (car l1) (car l2))`
- La conjetura de terminación es

```
(implies (and (not (endp l1))
              (not (endp l2))
              (< (car l1) (car l2)))
         (o< (+ (len (cdr l1)) (len l2))
             (+ (len l1) (len l2))))
```

Admisión de funciones recursivas

- La segunda llamada recursiva es `(mezcla l1 (cdr l2))` y los términos que influyen en ella son `(not (endp l1))`, `(not (endp l2))` y `(not (< (car l1) (car l2)))`
- La conjetura de terminación es

```
(implies (and (not (endp l1))
              (not (endp l2))
              (not (< (car l1) (car l2))))
         (o< (+ (len l1) (len (cdr l2)))
             (+ (len l1) (len l2))))
```

Admisión de funciones recursivas

- Usualmente el sistema es capaz de deducir una medida de terminación adecuada a partir de los casos recursivos de la definición
- Para proporcionar explícitamente la medida de terminación utilizamos la directiva **`(declare (xargs :measure m))`**

```
(defun mezcla (l1 l2)
  (declare (xargs :measure (+ (len l1) (len l2))))
  (cond ((endp l1) l2)
        ((endp l2) l1)
        ((< (car l1) (car l2))
         (cons (car l1) (mezcla (cdr l1) l2)))
        (t (cons (car l2) (mezcla l1 (cdr l2))))))
```

El principio de inducción estructural en ACL2

- Una prueba por inducción de $\phi \equiv \sum_{k=1}^{n}(2k-1) = n^2$
- En ACL2: $\phi \equiv$ `(equal (sum-2k-1 n) (* n n))`
  - Caso base: $n \notin \mathbb{N}^+ \rightarrow \phi(n)$

```
(implies (zp n)
         (equal (sum-2k-1 n) (* n n)))
```

  - Caso de inducción: $((n \in \mathbb{N}^+) \wedge \phi(n-1)) \rightarrow \phi(n)$

```
(implies (and (not (zp n))
              (equal (sum-2k-1 (- n 1))
                     (* (- n 1) (- n 1))))
         (equal (sum-2k-1 n) (* n n)))
```

  - La inducción es válida pues la hipótesis de inducción se establece para el valor `(- n 1)`, menor que `n`, y el conjunto de los números naturales está bien ordenado con respecto al orden usual

El principio de inducción estructural en ACL2

- Una fórmula ϕ se deduce a partir de las siguientes fórmulas
  - *Casos de inducción*: Para cada $\mathbf{1 \leq i \leq k}$

    ```
    (implies (and qi σi,1(φ) ... σi,hi(φ)) φ)
    ```

  - *Caso base*

    ```
    (implies (and (not q1) ... (not qk)) φ)
    ```

  donde $\mathbf{q_1}, \ldots, \mathbf{q_k}$ son términos, $\boldsymbol{\sigma_{i,j}}$, $\mathbf{(1 \leq i \leq k, 1 \leq j \leq h_i)}$ son sustituciones y, para cierto término `m`, se tienen

  - `m` es la representación en ACL2 de un ordinal

    ```
    (o-p m)
    ```

  - Para cada $\mathbf{i, j}$ tales que $\mathbf{1 \leq i \leq k}$ y $\mathbf{1 \leq j \leq h_i}$

    ```
    (implies qi (o< σi,j(m) m))
    ```

- Decimos que ϕ se demuestra por *inducción en las variables* del término `m`, denominado *medida*

El principio de inducción estructural en ACL2

- Una prueba por inducción de $\phi \equiv \sum_{k=1}^{n}(2k-1) = n^2$
  - Caso de inducción: $q_1 \equiv$ `(not (zp n))`, $\sigma_{1,1} = \{$`n/(- n 1)`$\}$

```
(implies (and (not (zp n))
              (equal (sum-2k-1 (- n 1))
                     (* (- n 1) (- n 1))))
         (equal (sum-2k-1 n) (* n n)))
```

  - Caso base

```
(implies (zp n)
         (equal (sum-2k-1 n) (* n n)))
```

  - Medida de inducción: `m = (if (natp n) n 0)`

```
(o-p (if (natp n) n 0))

(implies (not (zp n))
         (o< (if (natp (- n 1)) (- n 1) 0)
             (if (natp n) n 0)))
```

Inducción sugerida por una definición

- Una prueba por inducción de $\phi \equiv \sum_{k=1}^{n}(2k-1) = n^2$
  - Definición en ACL2

```
(defun sum-2k-1 (n)
  (cond ((zp n) 0)
        (t (+ (- (* 2 n) 1)
              (sum-2k-1 (- n 1))))))
```

  - Medida de terminación `m = (if (natp n) n 0)`
  - Hay una única llamada recursiva `(sum-2k-1 (- n 1))` y el término que influye en ella es `(not (zp n))`
  - La conjetura de terminación es

```
(implies (not (zp n))
         (o< (if (natp (- n 1)) (- n 1) 0)
             (if (natp n) n 0)))
```

Inducción sugerida por una definición

- Una prueba por inducción de $\phi \equiv \sum_{k=1}^{n}(2k-1) = n^2$
  - Caso de inducción: $t_{1,1} \equiv$ `(not (zp n))`, $\sigma_1 = \{n/$ `(- n 1)`$\}$

```
(implies (and (not (zp n))
              (equal (sum-2k-1 (- n 1))
                     (* (- n 1) (- n 1))))
         (equal (sum-2k-1 n) (* n n)))
```

  - Caso base:

```
(implies (zp n)
         (equal (sum-2k-1 n) (* n n)))
```

  - Medida de inducción `m` = `(if (natp n) n 0)`
  - Las propiedades de la medida de inducción coinciden con las conjeturas de terminación

Inducción sugerida por una definición

- Consideremos la definición **(defun f (x1 ... xn) cuerpo)**, con **m** casos finales en **cuerpo** en los que intervienen $\mathbf{n_m}$ llamadas recursivas de la forma **(f u1$_i$ ... un$_i$)** y siendo $\mathbf{t_{i,1}}, \ldots, \mathbf{t_{i,k_i}}$ los términos que influyen en dicho caso final
- Una fórmula ϕ se deriva a partir de las siguientes fórmulas
  - *Caso base*:

$$\mathbf{(implies\ (and\ (not\ (and\ t_{1,1}\ \ldots\ t_{1,k_1}))\ \ldots\ (not\ (and\ t_{m,1}\ \ldots\ t_{m,k_m})))\ \phi)}$$

  - *Casos de inducción*: Por cada llamada recursiva en **cuerpo**

$$\mathbf{(implies\ (and\ t_{i,1}\ \ldots\ t_{i,k_i}\ \sigma_{i,1}(\phi)\ \ldots\ \sigma_{i,n_i}(\phi))\ \phi)}$$

    donde $\sigma_{i,j}$ es la sustitución $\{\mathbf{x1}/\mathbf{u1_i}, \ldots, \mathbf{xn}/\mathbf{un_i}\}$ correspondiente a la **j**-ésima llamada recursiva en el **i**-ésimo caso

Inducción sugerida por una definición

- Las conjeturas de terminación de la definición

  (defun f (x1 ... xn) cuerpo)

  aseguran que existe una *medida de terminación* **m** que cumple
  - **m** es la representación en ACL2 de un ordinal:

    (o-p m)

  - por cada llamada recursiva en **cuerpo**

    (implies (and $t_{i,1}$ ... t_{i,k_i})
    (o< $\sigma_{i,j}$(m) m))

- No es necesario que la función **f** aparezca en la conjetura a demostrar

Ejemplo de inducción sugerida por una definición

- Conjetura

```
(equal (len (concatena l1 l2))
       (+ (len l1) (len l2)))
```

- Posibles inducciones sugeridas por las funciones implicadas en la conjetura
  - Inducción sugerida por **`(concatena l1 l2)`**
  - Inducción sugerida por **`(len l1)`**
  - Inducción sugerida por **`(len l2)`**

Ejemplo de inducción sugerida por una definición

- Definiciones

```
(defun len (l)
  (if (consp l)
      (+ 1 (len (cdr l)))
      0))

(defun concatena (l1 l2)
  (if (endp l1)
      l2
      (cons (car l1) (concatena (cdr l1) l2))))
```

Ejemplo de inducción sugerida por una definición

- Inducción sugerida por `(concatena l1 l2)`
  - Caso base: $\mathbf{t}_{1,1} \equiv$ `(not (endp l1))`

```
(implies (endp l1)
         (equal (len (concatena l1 l2))
                (+ (len l1) (len l2))))
```

  - Caso de inducción: $\sigma_1 = \{$`l1/(cdr l1)`$\}$

```
(implies (and (not (endp l1))
              (equal (len (concatena (cdr l1) l2))
                     (+ (len (cdr l1)) (len l2))))
         (equal (len (concatena l1 l2))
                (+ (len l1) (len l2))))
```

  - Medida de inducción: `m` $=$ `(len l1)`

Ejemplo de inducción sugerida por una definición

- Inducción sugerida por `(len l1)`
  - Caso base: $\mathbf{t}_{1,1} \equiv$ `(consp l1)`

```
(implies (not (consp l1))
         (equal (len (concatena l1 l2))
                (+ (len l1) (len l2))))
```

  - Caso de inducción: $\sigma_1 = \{$`l1/(cdr l1)`$\}$

```
(implies (and (consp l1)
              (equal (len (concatena (cdr l1) l2))
                     (+ (len (cdr l1)) (len l2))))
         (equal (len (concatena l1 l2))
                (+ (len l1) (len l2))))
```

  - Medida de inducción: `m` $=$ `(len l1)`
- Equivalente a la inducción sugerida por `(concatena l1 l2)`

Ejemplo de inducción sugerida por una definición

- Inducción sugerida por `(len l2)`
  - Caso base: $\mathbf{t}_{1,1} \equiv$ `(consp l2)`

```
(implies (not (consp l2))
         (equal (len (concatena l1 l2))
                (+ (len l1) (len l2))))
```

  - Caso de inducción: $\sigma_1 = \{$`l2`$/$`(cdr l2)`$\}$

```
(implies (and (consp l2)
              (equal (len (concatena l1 (cdr l2)))
                     (+ (len l1) (len (cdr l2)))))
         (equal (len (concatena l1 l2))
                (+ (len l1) (len l2))))
```

  - Medida de inducción: `m` $=$ `(len l2)`

Ejemplo de inducción sugerida por una definición

- Conjetura

```
(implies (and (ordenada l1)
              (ordenada l2))
         (ordenada (mezcla l1 l2)))
```

- Posibles inducciones sugeridas por las funciones implicadas en la conjetura
  - Inducción sugerida por **`(ordenada l1)`**
  - Inducción sugerida por **`(ordenada l2)`**
  - Inducción sugerida por **`(mezcla l1 l2)`**

Ejemplo de inducción sugerida por una definición

- Definiciones

```
(defun ordenada (l)
  (cond ((endp l) (equal l nil))
        (t (or (endp (cdr l))
               (and (<= (car l) (cadr l))
                    (ordenada (cdr l)))))))

(defun mezcla (l1 l2)
  (cond ((endp l1) l2)
        ((endp l2) l1)
        ((< (car l1) (car l2))
         (cons (car l1) (mezcla (cdr l1) l2)))
        (t (cons (car l2) (mezcla l1 (cdr l2))))))
```

Ejemplo de inducción sugerida por una definición

- Inducción sugerida por `(ordenada l1)`
  - Medida de inducción: `m` = `(len l1)`
  - Caso de inducción: $\sigma_1 = \{$`l1`$/$`(cdr l1)`$\}$

```
(implies
  (and (not (endp l1))                                  ; t1,1
       (not (endp (cdr l1)))                            ; t1,2
       (<= (car l1) (cadr l1))                          ; t1,3
       (implies (and (ordenada (cdr l1))
                     (ordenada l2))
                (ordenada (mezcla (cdr l1) l2))))
  (implies (and (ordenada l1)
                (ordenada l2))
           (ordenada (mezcla l1 l2))))
```

Ejemplo de inducción sugerida por una definición

- Inducción sugerida por `(ordenada l1)`
  - Caso base

```
(implies
  (not (and (not (endp l1))                          ; t1,1
            (not (endp (cdr l1)))                    ; t1,2
            (<= (car l1) (cadr l1))))                ; t1,3
  (implies (and (ordenada l1)
                (ordenada l2))
           (ordenada (mezcla l1 l2))))
```

  - Comportamiento del `or` computacional

$$\texttt{(not (and } q_1 \ \ldots \ q_m\texttt{))} \equiv \texttt{(or (not } q_1\texttt{) } \ldots \ \texttt{(not } q_m\texttt{))} \equiv \begin{cases} \texttt{(not } q_1\texttt{)} \\ \texttt{(and } q_1 \ \texttt{(not } q_2\texttt{))} \\ \ldots \\ \texttt{(and } q_1 \ \ldots \ q_{m-1} \ \texttt{(not } q_m\texttt{)))} \end{cases}$$

Ejemplo de inducción sugerida por una definición

- Inducción sugerida por `(ordenada l1)`
  - Casos bases

```
(implies
  (endp l1)                                          ; ¬t1,1
  (implies (and (ordenada l1)
                (ordenada l2))
           (ordenada (mezcla l1 l2))))

(implies
  (and (not (endp l1))                               ; t1,1
       (endp (cdr l1)))                              ; ¬t1,2
  (implies (and (ordenada l1)
                (ordenada l2))
           (ordenada (mezcla l1 l2))))

(implies
  (and (not (endp l1))                               ; t1,1
       (not (endp (cdr l1)))                         ; t1,2
       (not (<= (car l1) (cadr l1))))                ; ¬t1,3
  (implies (and (ordenada l1)
                (ordenada l2))
           (ordenada (mezcla l1 l2))))
```

Ejemplo de inducción sugerida por una definición

- Inducción sugerida por `(ordenada l2)`
  - Medida de inducción: `m` = `(len l2)`
  - Caso de inducción: $\sigma_1 = \{$`l2`$/$`(cdr l2)`$\}$

```
(implies
  (and (not (endp l2))                                  ; t1,1
       (not (endp (cdr l2)))                            ; t1,2
       (<= (car l2) (cadr l2))                          ; t1,3
       (implies (and (ordenada l1)
                     (ordenada (cdr l2)))
                (ordenada (mezcla l1 (cdr l2)))))
  (implies (and (ordenada l1)
                (ordenada l2))
           (ordenada (mezcla l1 l2))))
```

Ejemplo de inducción sugerida por una definición

- Inducción sugerida por `(ordenada l2)`
  - Casos bases

```
(implies
  (endp l2)                                              ; ¬t1,1
  (implies (and (ordenada l1)
                (ordenada l2))
           (ordenada (mezcla l1 l2))))

(implies
  (and (not (endp l2))                                   ; t1,1
       (endp (cdr l2)))                                  ; ¬t1,2
  (implies (and (ordenada l1)
                (ordenada l2))
           (ordenada (mezcla l1 l2))))

(implies
  (and (not (endp l2))                                   ; t1,1
       (not (endp (cdr l2)))                             ; t1,2
       (not (<= (car l2) (cadr l2))))                    ; ¬t1,3
  (implies (and (ordenada l1)
                (ordenada l2))
           (ordenada (mezcla l1 l2))))
```

Ejemplo de inducción sugerida por una definición

- Inducción sugerida por `(mezcla l1 l2)`
  - Medida de inducción: `m` = `(+ (len l1) (len l2))`
  - Caso de inducción: $\sigma_1 = \{$`l1`$/$`(cdr l1)`$\}$

```
(implies
  (and (not (endp l1))                                  ; t1,1
       (not (endp l2))                                  ; t1,2
       (< (car l1) (car l2))                            ; t1,3
       (implies (and (ordenada (cdr l1))
                     (ordenada l2))
                (ordenada (mezcla (cdr l1) l2))))
  (implies (and (ordenada l1)
                (ordenada l2))
           (ordenada (mezcla l1 l2))))
```

Ejemplo de inducción sugerida por una definición

- Inducción sugerida por `(mezcla l1 l2)`
  - Caso de inducción: $\sigma_2 = \{\texttt{l2}/\texttt{(cdr l2)}\}$

```
(implies
  (and (not (endp l1))                                    ; t2,1
       (not (endp l2))                                    ; t2,2
       (not (< (car l1) (car l2)))                        ; t2,3
       (implies (and (ordenada l1)
                     (ordenada (cdr l2)))
                (ordenada (mezcla l1 (cdr l2)))))
  (implies (and (ordenada l1)
                (ordenada l2))
           (ordenada (mezcla l1 l2))))
```

Ejemplo de inducción sugerida por una definición

- Inducción sugerida por `(mezcla l1 l2)`
  - Caso base

```
(implies
  (and (not (and (not (endp l1))                    ; t1,1
                 (not (endp l2))                    ; t1,2
                 (< (car l1) (car l2))))            ; t1,3
       (not (and (not (endp l1))                    ; t2,1
                 (not (endp l2))                    ; t2,2
                 (<= (car l2) (car l1)))))          ; t2,3
  (implies (and (ordenada l1)
                (ordenada l2))
           (ordenada (mezcla l1 l2))))
```

Ejemplo de inducción sugerida por una definición

- Inducción sugerida por `(mezcla l1 l2)`
  - Caso base

```
(implies (endp l1)                                    ; ¬t1,1
         (implies (and (ordenada l1)
                       (ordenada l2))
                  (ordenada (mezcla l1 l2))))

(implies (and (not (endp l1))                         ; t1,1
              (endp l2))                              ; ¬t2,2
         (implies (and (ordenada l1)
                       (ordenada l2))
                  (ordenada (mezcla l1 l2))))
```

- Correspondencia con los casos de la definición de `mezcla`

Ejemplo de inducción sugerida por una definición

- Definiciones

```
(defun len (l)
  (if (consp l)
      (+ 1 (len (cdr l)))
  0))

(defun hojas (tree)
  (if (consp tree)
      (if (consp (car tree))
          (+ (hojas (car tree)) (hojas (cdr tree)))
          (+ 1 (hojas (cdr tree))))
      0))

(defun flat (tree)
  (if (consp tree)
      (if (consp (car tree))
          (append (flat (car tree)) (flat (cdr tree)))
          (cons (car tree) (flat (cdr tree))))
      nil))
```

Ejemplo de inducción sugerida por una definición

- Conjetura

```
(equal (hojas tree)
       (len (flat tree)))
```

- Posibles inducciones sugeridas por las funciones implicadas en la conjetura
  - Inducción sugerida por **`(hojas tree)`**
  - Inducción sugerida por **`(flat tree)`**
- Ambos esquemas de inducción son iguales

Ejemplo de inducción sugerida por una definición

- Inducción sugerida por `(hojas tree)`
  - Primer caso de inducción: $\sigma_{1,1} = \{\texttt{tree}/\texttt{(car tree)}\}$
    $\sigma_{1,2} = \{\texttt{tree}/\texttt{(cdr tree)}\}$

```
(implies
  (and (consp tree)
       (consp (car tree))
       (equal (hojas (car tree))
              (len (flat (car tree))))        ; σ1,1(φ)
       (equal (hojas (cdr tree))
              (len (flat (cdr tree)))))       ; σ1,2(φ)
  (equal (hojas tree)
         (len (flat tree))))
```

Ejemplo de inducción sugerida por una definición

- Inducción sugerida por `(hojas tree)`
  - Segundo caso de inducción: $\sigma_{2,1} = \{\texttt{tree}/\texttt{(cdr tree)}\}$

```
(implies
  (and (consp tree)
       (not (consp (car tree)))
       (equal (hojas (cdr tree))
              (len (flat (cdr tree)))))        ; σ2,1(φ)
  (equal (hojas tree)
         (len (flat tree))))
```

Ejemplo de inducción sugerida por una definición

- Inducción sugerida por **`(hojas tree)`**
  - Caso base:

```
(implies (not (consp tree))
         (equal (hojas tree)
                (len (flat tree))))
```

Bibliografía

- Kaufmann, M. and Manolios, P. and Moore, J S. *Computer–Aided Reasoning: An Approach*. (Kluwer Academic Publishers, 2000)
- Kaufmann, M. and Moore, J S. ACL2 *Version 8.5*. (`http://www.cs.utexas.edu/users/moore/acl2/`)