Inducción y recursión en NQTHM

Dpto. de Ciencias de la Computación e Inteligencia Artificial

Universidad de Sevilla

Funciones inadmisibles en NQTHM

- **Ejemplo:**

```
(defn long (l)
  (if (equal l nil)
      0
    (add1 (long (cdr l)))))

********* FUNCION NO ADMITIDA ***********
```

- **La función no termina para ciertos valores:**

```
(long 0) = (add1 (long 0)) =
(add1 (add1 (long 0))) =
(add1 (add1 (add1 (long 0)))) = ...
```

- **La admisión de tal función daría lugar a inconsistencias en la lógica:**

```
(long 0) = (add1 (long 0))
y
(long 0) /= (add1 (long 0))
```

- **Solución: sólo admitir funciones que terminen para cualquier valor.**
  - **funciones no recursivas, sin problema.**
  - **¿cómo admitir funciones recursivas?**

Relaciones bien fundamentadas

• Definición 1:

Dado un conjunto A y una relación $\prec$ definida sobre A, decimos que $\prec$ es *noetheriana* sobre A si no existe una secuencia infinita $x_1, x_2, x_3 \ldots$ de elementos de A tales que $\ldots x_3 \prec x_2 \prec x_1$

• Definición 2:

La relación $\prec$ se dice *bien fundamentada* (b. f.) sobre A si es irreflexiva, transitiva y noetheriana en A

• Ejemplos:

- En el conjunto N de los números naturales, la relación $<$ es bien fundamentada.
- Tanto en el conjunto Z de los números enteros como en el conjunto R^+ de los reales positivos, la relación $<$ no es bien fundamentada.
- La relación de inclusión estricta es bien fundamentada en el conjunto de los subconjuntos finitos de un conjunto dado.
- La relación definida en Z por

  $$n|m \iff n \text{ divide a y } n \neq mm$$

  es bien fundamentada.
- (*Orden lexicográfico*) en el conjunto $N \times N$ de pares de números naturales, la relación $<_{lex}$ definida por:

  $$(n_1, n_2) <_{lex} (m_1, m_2) \iff n_1 < m_1 \vee (n1 = m1 \wedge n_2 < m_2)$$

  es bien fundamentada.

Contextos

- **Definición 3.**

Si un término `b` ocurre en un término `p` que a su vez ocurre en un término `(if t p q)` decimos que `b` ocurre en el *contexto* `t`. Si, por el contrario, `b` ocurre en `q`, decimos que `b` ocurre en el *contexto* `(not t)`

- **Nota: recuérdese que `cond` es una abreviatura de varios `if`.**

- **Ejemplos:**

```
1) En el término

 (if (vaciop a) a
     (arbol-b (raiz a) (simetrico (hd a))
                       (simetrico (hi a)))))

los términos (simetrico (hd a)) y (simetrico (hi a))
ocurren en el contexto (not (vaciop a))

2) En el término

 (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))))))

el término (merge (cdr l) m) ocurre en el contexto
(listp l), (listp m), (lessp (car l) (car m))
```

Funciones admisibles

• Definicion 4.

La definición `(DEFN FN (x1 ... xn) CUERPO)` es *admisible* si:

- `FN` es un símbolo no definido previamente.
- `x1, ..., xn` son n variables distintas.
- no aparecen en `CUERPO` variables que no sean `x1, ..., xn`
- existe un término `m` (llamado *medida*) tal que `(NUMBERP m)` y tal que para cada ocurrencia `(FN s1 .. sn)` en `CUERPO`, con contextos `t1,...,tk`, se tiene que

  ```
  (IMPLIES (AND t1 ... tk) (LESSP m' m))
  ```

  donde `m'` es el término obtenido sustituyendo en `m` las variables `x1, ..., xn` respectivamente por `s1,...,sn`

Ejemplo de función admisible (I)

- **La función `rev1`:**

```
(defn rev1 (x acc)
  (if (nlistp x) acc
      (rev1 (cdr x) (cons (car x) acc))))
```

- **La función es admisible:**

```
Si tomamos como medida m el término

 (COUNT X)

la única llamada recursiva es

(REV1 (CDR X) (CONS (CAR X) ACC))

que se produce en el contexto

 (LISTP X)

en este caso el término m' es

(COUNT (CDR X))

y se prueba sin problemas que

 (IMPLIES (LISTP X)
          (LESSP (COUNT (CDR X)) (COUNT X)))
```

Ejemplo de función admisible (II)

- **La función** numeros-en-primeros-n:

```
(defn numeros-en-primeros-n (n l)
   (cond  ((zerop n) nil)
          ((numberp (car l))
           (cons (car l)
                 (numeros-en-primeros-n (sub1 n) (cdr l))))
          (t (numeros-en-primeros-n (sub1 n) (cdr l)))))
```

Si tomamos como medida m el término

```
 (COUNT N)
```

1) La primera llamada recursiva es
(NUMEROS-EN-PRIMEROS-N (SUB1 N) (CDR L))
que se produce en el contexto
 (NOT (ZEROP N)), (NUMBERP (CAR L))
y se prueba que
(IMPLIES (AND (NOT (ZEROP N)) (NUMBERP (CAR L)))
         (LESSP (COUNT (SUB1 N)) (COUNT N)))

2) La segunda llamada recursiva es
(NUMEROS-EN-PRIMEROS-N (SUB1 N) (CDR L))
que se produce en el contexto
 (NOT (ZEROP N)), (NOT (NUMBERP (CAR L)))
y se prueba que
(IMPLIES (AND (NOT (ZEROP N)) (NOT (NUMBERP (CAR L))))
         (LESSP (COUNT (SUB1 N)) (COUNT N)))

en ambos casos, el término m' es (COUNT (SUB1 N))

Buscando la función de medida

- **Problema: encontrar una fórmula `m` que prueba la admisibilidad.**
- **Por defecto, el sistema intenta `(COUNT xi)`, para cada argumento `xi`.**
- **Salida del sistema:**

```
>(defn numeros-en-primeros-n (n l)
   (cond  ((zerop n) nil)
          ((numberp (car l))
           (cons (car l)
                 (numeros-en-primeros-n (sub1 n) (cdr l))))
          (t (numeros-en-primeros-n (sub1 n) (cdr l)))))
....
....
inform us that the measure (COUNT N) decreases according
to the well-founded relation LESSP in each recursive call.
hence, NUMEROS-EN-PRIMEROS-N is accepted under the principle
of definition.....
```

Sugiriendo la función de medida

- **Ejemplo:**

```
(defn anade-x-l-hasta-n (n x l)
   (if (geq (longitud l) n)
       l
     (anade-x-l-hasta-n n x (cons x l))))
****** LA FUNCION NO ES ADMITIDA **********
```

- **Problema:**

El sistema intenta como términos de medida, resp., (COUNT n), (COUNT x) y (COUNT l)
y ninguno decrece en la llamada recursiva.

- **Sin embargo, la función es admisible.**

En cada llamada recursiva, la longitud de l se ‘‘acerca’’ al número n, hasta llegar al caso base.

- **La función de medida adecuada es**
  `(difference n (longitud l))`

- **¿Cómo indicárselo al sistema?**

Sugerencia en DEFN

- **Sintaxis:**

```
(defn FN (x1 ... xn)
      CUERPO
   ((relacion m)))
```

donde:
- "relación" indica la relación de decrecimiento, usualmente LESSP.
- m es la medida usada para la admisión de la función.

- **Ejemplo:**

```
>(defn anade-x-l-hasta-n (n x l)
   (if (geq (longitud l) n)
       l
     (anade-x-l-hasta-n n x (cons x l)))
   ((LESSP (difference n (longitud l)))))
...
     linear arithmetic, the lemmas SUB1-ADD1 AND CDR-CONS,
and the definitions of DIFFERENCE and LONGITUD can be used
to show that the measure:
      (DIFFERENCE N (LONGITUD L))
decreases according to the well-founded relation LESSP
in each recursive call. hence, ANADE-X-L-HASTA-N
is accepted under the definitional principle.
....
...
 ANADE-X-L-HASTA-N
```

Ordenes lexicográficos

- **Ejemplo:**

```
(defn suma-elementos (l acc)
   (cond ((nlistp l) acc)
         ((zerop (car l))
          (suma-elementos (cdr l) acc))
         (t (suma-elementos
               (cons (sub1 (car l)) (cdr l))
               (add1 acc)))))
***** LA FUNCION NO ES ADMITIDA ******
```

- **Problema:**

Ni (COUNT l) ni (COUNT acc) sirven como funciones de medida,
y éstas son las medidas que se intentan por defecto

- **La función de medida debe ser la misma para todas las llamadas recursivas.**
- **La función es admisible:**

En la primera llamada recursiva decrece (LONGITUD l)
y en la segunda llamada recursiva la longitud
es la misma pero decrece (CAR l)

- **En lugar de usar LESSP hay que usar un orden lexicográfico:** ORD-LESSP

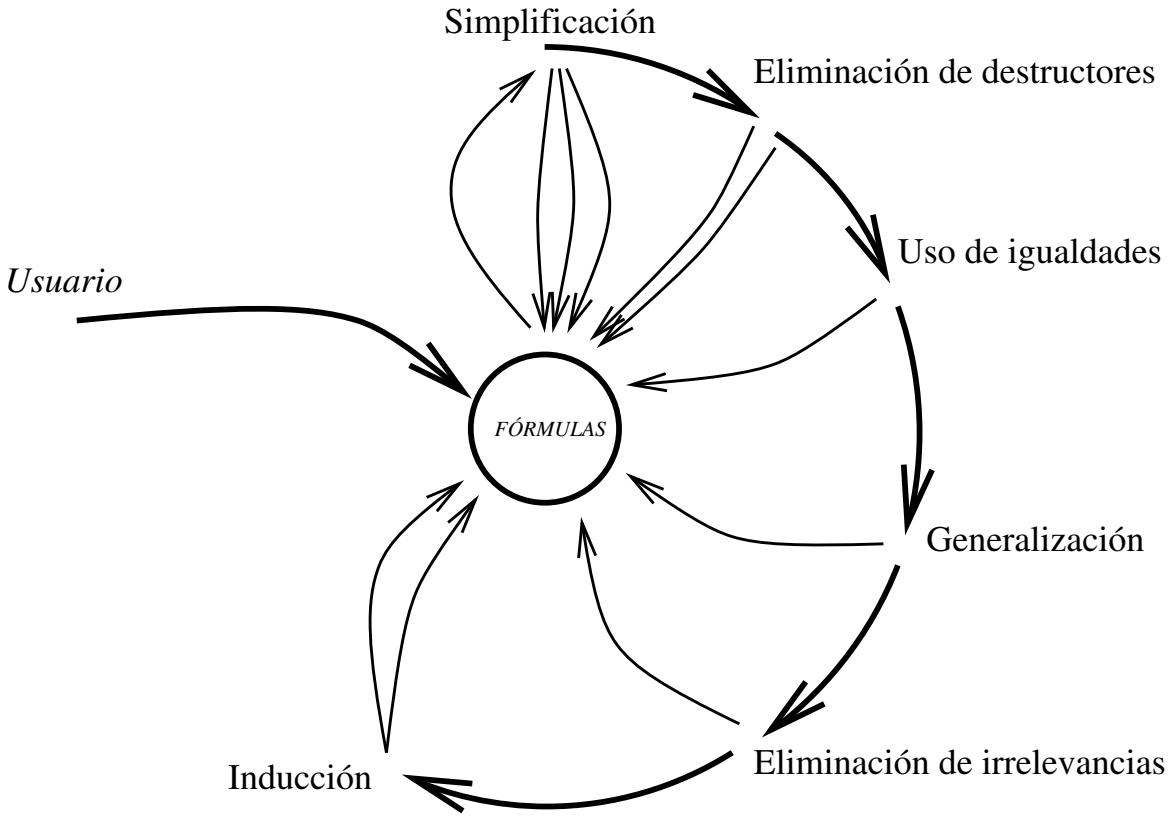
Ordenes lexicográficos con DEFN

- **Ejemplo:**

```
(defn suma-elementos (l acc)
   (cond ((nlistp l) acc)
         ((zerop (car l))
          (suma-elementos (cdr l) acc))
         (t (suma-elementos
               (cons (sub1 (car l)) (cdr l))
               (add1 acc))))
   ((ORD-LESSP (cons (add1 (longitud l)) (add1 (car l))))))
....
....can be used to establish that the
measure (CONS (ADD1 (LONGITUD L)) (ADD1 (CAR L)))
decreases according to the well-founded relation
ORD-LESSP in each recursive call.  Hence, SUMA-ELEMENTOS
is accepted under the definitional principle.
....
SUMA-ELEMENTOS
```

- **Nótese el uso de ADD1.**
- **ORD-LESPP: orden entre ordinales.**
- **Puede que sea necesario probar lemas previos que ayuden al sistema a probar la admisibilidad de una función.**

Pruebas por inducción



Inducción noetheriana

• Principio de inducción noetheriana:

Sea A un conjunto, $\prec$ una relación noetheriana definida sobre A, y $P(x)$ una propiedad definida para los elementos de A. Si se tiene que:

$$\forall a \in A([\forall b \in A(b \prec a \rightarrow P(b))] \rightarrow P(a)),$$

entonces

$$\forall a \in A(P(a)).$$

• Es decir:

Para probar que una propiedad es cierta en todos los elementos de un conjunto, en el que está definida una relación noetheriana, basta con probar que la propiedad es cierta para cada elemento, *suponiendo* que es cierta para cualquier elemento *menor* (**hipótesis de inducción**).

• Definición 5:

Sea A un conjunto, $\prec$ una relación noetheriana definida sobre A, y $x \in A$. Decimos que x es un *elemento minimal* (respecto a $\prec$), si
$\nexists a \in A(a \prec x)$.

• Nota:

En una prueba por inducción los elementos minimales *no pueden* tener hipótesis de inducción, ya que no tienen elementos "anteriores" (**casos base**).

Inducción noetheriana (II)

• Ejemplos:

- En el conjunto N de los números naturales, con la relación $<$, se pueden hacer pruebas por inducción.
- La relación $\prec_{abs}$ definida en Z por

$$a \prec_{abs} b \iff |a| < |b|$$

es una relación bien fundamentada y es válido hacer pruebas por inducción de una propiedad suponiendo como h. de i. que la propiedad es cierta para todo elemento con menor valor absoluto.

- Sea A un alfabeto y A^* el conjunto de sus palabras. Entonces la relación $\prec_{long}$ definida por

$$a \prec_{long} b \iff long(a) < long(b)$$

es bien fundamentada y por tanto es válido hacer pruebas por inducción de una propiedad suponiendo como h. de i. que la propiedad es cierta para todo elemento con menor longitud.

• Relación trasladada:

Si $f : A \to B$, y $\prec$ es noetheriana en A, entonces la relación $\prec_f$ definida en B como

$$a \prec_f b \iff f(a) \prec f(b)$$

es noetheriana en B.

Mecanizando el principio de inducción.

- **Todos los esquemas de inducción son casos particulares del principio de inducción noetheriana.**
- **En NQTHM:**

Las relaciones noetherianas que existen son `LESSP` y `ORD-LESSP`, definidas sobre `NUMBERP` y n-uplas de números, respectivamente.

- **Es posible considerar otros órdenes b.f.:**

Se define una medida `m` sobre los objetos que maneja NQTHM y se considera la relación `(LESSP m(X) m(Y))` (relación trasladada). Así, para probar una propiedad `p` por inducción, es posible suponer que `p` es cierto para objetos de menor medida.

- **Esquemas de inducción en NQTHM:**

```
>(prove-lemma longitud-concatena (rewrite)
  (equal (longitud (concatena l m))
         (plus (longitud l) (longitud m))))
.... We will induct
according to the following scheme:
      (AND (IMPLIES (NLISTP L) (p L M))
           (IMPLIES (AND (NOT (NLISTP L)) (p (CDR L) M))
                    (p L M))).
....can be used to prove that the measure (COUNT L)
decreases according to the well-founded relation LESSP...
```

Analizando el ejemplo.

- **Salida del sistema::**

```
....
... We will induct according to the following scheme:
      (AND (IMPLIES (NLISTP L) (p L M))
           (IMPLIES (AND (NOT (NLISTP L)) (p (CDR L) M))
                    (p L M))).
....can be used to prove that the measure (COUNT L)
decreases according to the well-founded relation LESSP...
```

- **Prueba por casos:**

En el anterior esquema:
- Caso 1: (NLISTP L)
- Caso 2: (NOT (NLISTP L))

- **Hipótesis de inducción:**

En el anterior esquema:
- Caso 1: Caso base (sin h. de i.)
- Caso 2: Caso inductivo, con h. de i. (p (CDR L) M)

- **Relación noetheriana que justifica la validez del esquema:**

- Relación trasladada,
- a partir de LESSP,
- usando la medida (COUNT L).

Otro ejemplo (I).

```
>(prove-lemma todos-numeros ()
   (implies (member x (numeros-en-primeros-n n l))
            (numberp x)))

     Name the conjecture *1.

     Perhaps we can prove it by induction.  There is only one
plausible induction.
We will induct according to the following scheme:
      (AND (IMPLIES (ZEROP N) (p X N L))
           (IMPLIES (AND (NOT (ZEROP N))
                         (NUMBERP (CAR L))
                         (p X (SUB1 N) (CDR L)))
                    (p X N L))
           (IMPLIES (AND (NOT (ZEROP N))
                         (NOT (NUMBERP (CAR L)))
                         (p X (SUB1 N) (CDR L)))
                    (p X N L))).
Linear arithmetic, the lemma COUNT-NUMBERP, and the
definition of ZEROP establish that the measure (COUNT N)
decreases according to the well-founded relation LESSP
in each induction step of the scheme.  Note, however, the
inductive instances chosen for L.  The above induction
scheme produces the following three new goals:
....
```

Otro ejemplo (II).

- **Casos:**

```
Caso 1: (ZEROP N)

Caso 2: (AND (NOT (ZEROP N)) (NUMBERP (CAR L)))

Caso 3: (AND (NOT (ZEROP N)) (NOT (NUMBERP (CAR L))))
```

- **Hipótesis de inducción:**

```
Caso 1: Caso base (sin h. de i.)

Caso 2: Caso inductivo, con h. de i.
        (p X (SUB1 N) (CDR L))

Caso 3: Caso inductivo, con h. de i.
        (p X (SUB1 N) (CDR L))
```

- **Relación noetheriana que justifica la validez del esquema:**

```
- Relación trasladada,
- a partir de LESSP,
- usando la medida (COUNT N).
```

- **Problema: ¿cómo encontrar el esquema de inducción adecuado?**

Principio de inducción en NQTHM.

• Inducción respecto a `LESSP`.

Supongamos que:

- `p` es un término (propiedad a probar) con variables `x1,...,xn`.
- `m` es un término (función de medida) cuyas variables están en `{x1,...,xn}`.
- `q1, ..., qk` son términos (casos inductivos).
- `(NUMBERP m)` es demostrable.
- Por cada caso `qi`, se tienen una serie de hipótesis de inducción, que son instancias de `p`, obtenidas sustituyendo las variables de `p` por otros datos. Debe ser demostrable que, suponiendo cierto `qi`, tales datos tienen menor medida que los sustituidos.

Entonces `p` es teorema si:

- Se prueba que `p` es cierto, cuando todos los `qi` son falsos (caso base).
- Se prueba que `p` es cierto, suponiendo cierto `qi` y las hipótesis de inducción correspondientes a `qi`, para `1<=i<=k`.

• EN NQTHM, además, se puede usar un principio de inducción análogo para `ORD-LESSP`.

Ejemplo.

```
* Término p a probar:
(implies (member x (numeros-en-primeros-n n l))
         (numberp x)))

* Variables de p: {x,n,l}

* Casos inductivos:
  - q1: (and (not (zerop n)) (numberp (car l)))
    con una h. de i.: (p x (sub1 n) (cdr l))
  - q2: (and (not (zerop n)) (not (numberp (car l))))
    con una h. de i.: (p x (sub1 n) (cdr l))

* Medida: (count n)

* La medida decrece en cada h. de i., ya que es
  posible probar:
  (implies (and (not (zerop n)) (numberp (car l)))
           (lessp (count (sub1 n)) (count n)))
  y
  (implies (and (not (zerop n)) (not (numberp (car l))))
           (lessp (count (sub1 n)) (count n)))

* Por tanto, para probar p, bastará probar que:
  - Caso base: (implies (zerop n) (p x n l))
  - Casos inductivos:
    (implies (and (not (zerop n)) (numberp (car l))
                  (p x (sub1 n) (cdr l)))
             (p x n l))
    (implies (and (not (zerop n)) (not (numberp (car l)))
                  (p x (sub1 n) (cdr l)))
             (p x n l)))
```

Inducción y recursión.

- **Problema: elegir los casos y las hipótesis de inducción**
- **Relación entre recursión e inducción:**
  - **El sistema analiza las funciones recursivas que aparecen en el término `p` a probar.**
  - **Las funciones recursivas *sugieren* esquemas de inducción.**
- **Esquemas de inducción sugerido por una función recursiva:**
  - **Por cada contexto en el que se produce una llamada recursiva, se tiene un caso de inducción distinto.**
  - **Por cada una de las llamadas recursivas que se producen en un contexto dado, se tiene una hipótesis de inducción distinta.**
  - **La medida es la que se usó en la admisión de la función.**
  - **El decrecimiento de la medida en las hipótesis de inducción se tiene asegurado, ya que la función fue admitida.**

Ejemplos.

- **Esquema de inducción sugerido por** `numeros-en-primeros-n`**:**

```
1) La primera llamada recursiva es
(NUMEROS-EN-PRIMEROS-N (SUB1 N) (CDR L))
y se produce en el contexto
 (NOT (ZEROP N)), (NUMBERP (CAR L))

Esto produce un caso inductivo, y la hipótesis
de inducción correspondiente:
(p X (SUB1 N) (CDR L))

2) La segunda llamada recursiva es
(NUMEROS-EN-PRIMEROS-N (SUB1 N) (CDR L))
y se produce en el contexto
 (NOT (ZEROP N)), (NOT (NUMBERP (CAR L)))

Esto produce otro caso inductivo, y la hipótesis
de inducción correspondiente:
(p X (SUB1 N) (CDR L))
```

- **Esquema de inducción sugerido por** `longitud`**:**

```
La única llamada recursiva  es
(LONGITUD (CDR L))
y se produce en el contexto
 (NOT (NLISTP L))

Esto produce un caso inductivo, y la hipótesis
de inducción correspondiente:
(p (CDR L) M)
```

Transparencia en blanco.

Sugiriendo al sistema la inducción.

- **A veces, el esquema de inducción que usa el sistema no es el más adecuado.**
- **El usuario puede obligar a que el sistema use un determinado esquema de inducción:**
  - **En realidad, obliga a que se use el esquema de inducción sugerido por una determinada función recursiva.**
- **Consejo** `INDUCT` **a** `PROVE-LEMMA`**.**
- **Sintaxis:**

```
(prove-lemma  NOMBRE ([rewrite])
      CUERPO
   ((induct (FN x1 ... xn))))

donde FN es una función de aridad n
y xi son variables que (usualmente)
aparecen en CUERPO
```

- **El sistema prueba** `CUERPO` **por inducción, usando como esquema el sugerido por la definición de** `(FN x1 ... xn)`**.**
- **Usualmente** `FN` **se define con el único propósito de proporcionar el esquema de inducción.**

Ejemplo.

- **Esquema de inducción:**

Supongamos que queremos probar un determinado resultado (p L N) con el siguiente esquema de inducción:

* Caso base: L tiene menos de dos elementos.
* Caso inductivo 1: L tiene al menos dos elementos y N>0.
  H. de i. en este caso: (p (CDDR L) (SUB1 N))
* Caso inductivo 2: L tiene al menos dos elementos y N=0.
  H. de i. en este caso: (p (CDDR L) N)

SUPONGAMOS QUE EL SISTEMA NO USA TAL ESQUEMA POR SI MISMO.

- **Definiendo la función que proporciona el esquema de inducción deseado:**

```
>(defn consejo-induccion (l n)
   (cond ((or (nlistp l) (nlistp (cdr l)))
          t)               ; El valor es lo de menos
         ((not (zerop n)) ; Lo importante es el esquema
          (consejo-induccion (cddr l) (sub1 n)))
         (t
          (consejo-induccion (cddr l) n))))
```

- **Sugiriendo el esquema:**

```
>(prove-lemma NOMBRE ([rewrite])
    (p l n)
   ((INDUCT (consejo-induccion l n))))
```