Práctica 1
Razonamiento Automático
Introducción a NQTHM

Introducción

- **NQTHM: demostrador de Boyer y Moore.**
- **Demostrador de teoremas.**
- **Probar propiedades sobre programas LISP:**
  - **formalmente.**
  - **automáticamente.**
- **Métodos formales aplicados a la IS.**
- **Inicio (en entorno emacs):**

```
Teclear: ESC-x run-nqthm
Buffer de emacs *inferior-lisp*:
```

- **Evaluación y respuesta:**

```
GCL (GNU Common Lisp)  Version(2.2.2) lunes, .....
Licensed under GNU Public Library License
Contains Enhancements by W. Schelter

Pc-Nqthm-1992.
Initialized with (BOOT-STRAP NQTHM) on ....

>
```

NQTHM Y LISP

- **NQTHM está programado en LISP:**
  - **mismo entorno que LISP,**
  - **LISP + funciones propias del demostrador.**
- **Algunas variaciones de sintaxis:**

LISP	NQTHM
`====`	`=====`
`(defun ...)`	`(defn ...)`
`(not (listp X))`	`(nlistp X)`
`(+ X Y)`	`(plus X Y)`
`(+ 1 X)`	`(ADD1 X)`
`...`	`...`

- **Muchas similitudes:**

  `EQUAL, AND, OR, NOT, CAR, CDR, IF, COND, MEMBER, NIL...`

- **Funciones nuevas:**

  `PROVE-LEMMA, IMPLIES,...`

Listas en NQTHM Y LISP

- **Ejemplos:**

```
>(cons 3 4)
(3 . 4)
;;; Internamente: (3 . 4)

>(cons 4 nil)
(4)
;;; Internamente: (4 . nil)

>(cons 3 (cons 4 (cons 5 nil)))
(3 4 5)
;;; Internamente: (3 . (4 . (5 . nil)))

>(cons 3 (cons 4 (cons 5 6)))
(3 4 5 . 6)
;;; Internamente: (3 . (4 . (5 . 6)))
```

- **En general:**

```
1) (a1 a2 ... aN) es la representación externa de:
   (cons a1 (cons a2 (cons ...(cons aN nil))))
   que, internamente, se representa como:
   (a1 . (a2 . ( ...(aN . nil))))

2) (a1 a2 ... aN . aN+1) es la representación externa de:
   (cons a1 (cons a2 (cons ...(cons aN aN+1))))
   que, internamente, se representa como:
   (a1 . (a2 . ( ...(aN . aN+1))))
   siempre que aN+1 no es NIL.
```

Listas en NQTHM

- **Predicado** LISTP**:**

```
- Un dato es de tipo LISTA (LISTP) si es el resultado
  de hacer una operaci\'on de la forma (CONS . .)
Ejemplos:

(listp '(1 2 3)) = T
(listp '(1 2 3 . 4)) = T
(listp nil) = F
(listp 3) = F
```

- **Funciones** CAR **y** CDR**:**

```
- Si l es LISTP (i.e., l = (CONS a b)), entonces
  (CAR l) = a y (CDR l) = b.
- Si l no es LISTP, (CAR l) = (CDR l) = 0.
Ejemplos:

(car '(3 4)) = 3
(cdr '(3 4)) = '(4)
(car '(3 . 4)) = 3
(cdr '(3 . 4)) = 4
(car nil) = 0
(cdr nil) = 0
(car 3) = 0
(cdr 3) = 0
```

Listas propias en NQTHM

• Definición de listas propias.

```
Un dato l es una LISTA PROPIA:
* Si (LISTP l) (i.e., l=(CONS a b)) y b es lista propia.
* Si (NLISTP l), sólo si l = NIL.
```

• Ejemplos.

```
*1*
'(1 2 3 . 4) NO es propia
ya que sería propia si
'(2 3 . 4) lo fuera,
y ésta sería propia si
'(3 . 4) lo fuera
y ésta sería propia si
4 lo fuera
pero 4 no es NIL.

*2*
'(1 2 3) SI es propia
ya que
'(2 3) lo es,
ya que
'(3) lo es,
ya que
NIL lo es.
```

Definición de funciones en NQTHM

- **Usando `defn`.**
- **NQTHM sólo admite definición de funciones que es capaz de probar que terminan para cualquier entrada:**

```
La función:
(defn longitud (l)
   (if (equal l nil) 0 (add1 (longitud (cdr l)))))
NO ES ADMITIDA, ya que:
(longitud '(1 2 . 3)) = (add1 (longitud '(2 . 3))) =
= (add1 (add1 (longitud 3))) =
= (add1 (add1 (add1 (longitud 0)))) =
= (add1 (add1 (add1 (add1 (longitud 0))))) = ...

Tal y como está definida, longitud sólo termina si recibe
como entrada una lista propia.

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

SI ES ADMITIDA. Por ejemplo:
(longitud '(1 2 . 3)) = (add1 (longitud '(2 . 3))) =
= (add1 (add1 (longitud 3))) =
= (add1 (add1 0)) = 2
```

Entornos de evaluación y demostración

- **Entorno de evaluación (de funciones `defn`):**

```
>(defn longitud (l)
   (if (nlistp l)
       0
     (add1 (longitud (cdr l)))))
....
[ 0.0 0.1 0.0 ]
LONGITUD

>(r-loop)
Trace Mode: Off   Abbreviated Output Mode:  On
Type ? for help.
*(longitud '(1 2 3))
 3
*ok
Exiting R-LOOP.
NIL
>
```

- **Entorno de demostración:**

```
>(prove-lemma long-concatena ()
   (equal (longitud (concatena x y))
          (plus (longitud x) (longitud y))))
.....
[ 0.0 0.0 0.1 ]
LONG-CONCATENA
>
```

Demostraciones (I)

- **Demostración a mano de**
  LONG-CONCATENA:

```
Demostración por inducción:
- CASO BASE:
Demostrar el resultado para x, cuando (nlistp x).
- PASO INDUCTIVO:
Supuesto que x es listp, y se tiene el resultado
para (cdr x), probarlo para x.
-------------------------------------------------------
********** CASO BASE ***********
En este caso, (concatena x y)=y, y (longitud x)=0, luego
(longitud (concatena x y))=(longitud y)
=(plus 0 (longitud y))=(plus (longitud x) (longitud y))

******* PASO INDUCTIVO *********
Supongamos (hipótesis de inducción) que:
(longitud (concatena (cdr x)  y)) =
= (plus (longitud (cdr x)) (longitud y))
Entonces:
(longitud (concatena x y)) = (*1*)
(longitud (cons (car x) (concatena (cdr x) y))) = (*2*)
(add1 (longitud (concatena (cdr x) y))) = (H.I.)
(add1 (plus (longitud (cdr x)) (longitud y))) = (*3*)
(plus (add1 (longitud (cdr x))) (longitud y)) = (*2*)
(plus (longitud x) (longitud y))

(*1*) Definición de concatena.
(*2*) Definición de longitud.
(*3*) Definición de plus.
```

Demostraciones (II)

- **El usuario teclea:**

```
>(prove-lemma long-concatena ()
   (equal (longitud (concatena x y))
          (plus (longitud x) (longitud y))))
```

- **El sistema responde:**

```
     Call the conjecture *1.

     Perhaps we can prove it by induction.
Three inductions are suggested by terms
in the conjecture.  They merge into two likely
candidate inductions. However, only one is unflawed.
We will induct according to the following
scheme:
  (AND (IMPLIES (AND (LISTP X) (p (CDR X) Y))
                (p X Y))
       (IMPLIES (NOT (LISTP X)) (p X Y))).
Linear arithmetic and the lemma CDR-LESSP
can be used to prove that the
measure (COUNT X) decreases according
to the well-founded relation LESSP in
each induction step of the scheme.
The above induction scheme leads to two
new goals:
      ..... (continúa)
```

Demostraciones (III)

```
Case 2. (IMPLIES (AND (LISTP X)
                      (EQUAL (LONGITUD (CONCATENA (CDR X) Y))
                             (PLUS (LONGITUD (CDR X))
                                   (LONGITUD Y))))
                 (EQUAL (LONGITUD (CONCATENA X Y))
                        (PLUS (LONGITUD X) (LONGITUD Y)))),

  which simplifies, applying the lemmas CDR-CONS and SUB1-ADD1, and opening
  up the definitions of CONCATENA, LONGITUD, and PLUS, to:

        T.

Case 1. (IMPLIES (NOT (LISTP X))
                 (EQUAL (LONGITUD (CONCATENA X Y))
                        (PLUS (LONGITUD X) (LONGITUD Y)))),

  which simplifies, unfolding the functions CONCATENA, LONGITUD, EQUAL, and
  PLUS, to:
        T.

     That finishes the proof of *1.  Q.E.D.

[ 0.0 0.0 0.1 ]
LONG-CONCATENA
```

- **Pruebas por:**
  - **Inducción**
  - **Simplificación**
  - **Lógica proposicional.**

Base de datos (conocimiento)

- **Inicialmente, definiciones y axiomas.**
- **Todos los `defn`, si es que son admitidos.**
- **Todos los `prove-lemma`:**

```
>(prove-lemma long-concatena (rewrite)
   (equal (longitud (concatena x y))
          (plus (longitud x) (longitud y))))
...
[ 0.0 0.0 0.1 ]
LONG-CONCATENA
************ Almacenado como regla de reescritura.
```

- **Al intentar probar un resultado, el sistema hace uso de todos los *eventos* de la base de datos:**

```
>(prove-lemma longitud-reverse ()
   (equal (longitud (reverse l))
          (longitud l)))
....
  This simplifies, rewriting with the lemmas CDR-CONS and
  LONGITUD-CONCATENA, and opening up NLISTP, REVERSE, ADD1,
  and LONGITUD, to:
...
[ 0.0 0.1 0.1 ]
LONGITUD-REVERSE
```

La lógica de Boyer y Moore (I)

- **Sentencias que expresan propiedades de funciones LISP.**
- **Sentencias universalmente cuantificadas (implícitamente).**

```
(prove-lemma contenida-member ()
  (implies (and (member x l)
                (contenida l m))
           (member x m)))

;;; "PARA CUALESQUIERA x, l y m, SI x es miembro de l
;;; y l esta contenido en m, ENTONCES x es miembro de m".
```

- **La equivalencia lógica se puede formular con `equal` o con `iff`.**

```
(prove-lemma contenida-concatena (rewrite)
        (equal (contenida (concatena l m) n)
               (and (contenida l n)
                    (contenida m n))))

;;; "PARA CUALESQUIERA l, m, n, la concatenacion de l e m
;;; esta contenida de n SI Y SOLO SI l esta contenida de n
;;; Y m esta contenida de n".
```

La lógica de Boyer y Moore (II)

- **Contiene a la lógica proposicional:**

```
Por ejemplo, los siguientes lemas se prueban
inmediatamente:

(prove-lemma de-morgan ()
  (equal (not (and p q))
         (or (not p) (not q))))

(prove-lemma modus-ponens ()
  (implies (and p (implies p q)) q))

(prove-lemma implica-transitivo ()
  (implies (and (implies p q) (implies q r))
           (implies p r)))
```

- **Conectivas proposicionales:**

```
T, F, IF, AND, OR, NOT, IMPLIES, IFF
```

- **Predicado `equal` para indicar igualdad.**

```
(prove-lemma longitud-reverse (rewrite)
  (equal (longitud (reverse l))
         (longitud l)))
```

Cooperando con el demostrador (I)

- **Intento fallido de demostración:**

```
> (prove-lemma concatena-nil (rewrite)
          (equal (concatena l nil) l))
...
************** F  A  I  L  E  D **************
[ 0.0 0.0 0.1 ]
NIL
```

- **Inspección de la prueba fallida:**

```
...
  which simplifies, expanding NLISTP and CONCATENA, ...
        (IMPLIES (NOT (LISTP L))
                 (EQUAL NIL L)).
...
```

- **El lema es falso:**

```
>(r-loop)
*(concatena '(a b . c) nil)
 '(A B)
```

- **Ahora sí:**

```
(prove-lemma concatena-nil (rewrite)
  (implies (es-lista-propia l)
           (equal (concatena l nil)
                  l)))
```

Las reglas de reescritura

- **El predicado de igualdad `equal` como regla `rewrite`**
- **Ejemplos:**
  - **regla incondicional:**

```
(prove-lemma long-concatena (rewrite)
   (equal (longitud (concatena x y))
          (plus (longitud x) (longitud y))))
;; Reemplaza (longitud (concatena A B))
;; por (plus (longitud A) (longitud B))
```

  - **regla condicional:**

```
(prove-lemma concatena-nil (rewrite)
  (implies (es-lista-propia l)
           (equal (concatena l nil)
                  l)))
;; Reemplaza (concatena A nil) por A
;; siempre que A sea propia
```

  - **`equal` implícito:**

```
(prove-lemma contenida-member (rewrite)
  (implies (and (member x l)
                (contenida l m))
           (member x m)))
;; Reemplaza (member A M) por T siempre que
;; exista L tal que A sea miembro de L y L
;; este contenida en M
```

Cooperando con el demostrador (II)

- **Otro ejemplo fallido (C-c C-c):**

```
(prove-lemma contenida-reflexivo (rewrite)
  (contenida a a))
```

- **Inspeccionando la prueba fallida (C-t 1):**

```
  ...
  This simplifies, opening up ...
       (IMPLIES (CONTENIDA X X)
                (CONTENIDA X (CONS Z X))),
  ...
```

- **Hay que "ayudar" al sistema, probando una regla:**

```
(prove-lemma contenida-cons (rewrite)
  (implies (contenida a b)
           (contenida a (cons x b))))
```

- **Ahora sí:**

```
  ...
  which further simplifies, applying CONTENIDA-CONS, to:
       T.
  ...
[ 0.0 0.0 0.0 ]
CONTENIDA-REFLEXIVO
```

Conclusiones

- **NQTHM: un demostrador de propiedades formales sobre funciones LISP.**
- **Ademas de probar propiedades, el sistema permite evaluar llamadas a las funciones.**
- **La lógica: proposicional, con símbolos de funciones e igualdad.**
- **Las variables están universalmente cuantificadas.**
- **Base de datos de eventos: `defn` y `prove-lemma`.**
- **Las reglas de reescritura.**
- **Frecuentemente, la prueba no tiene éxito.**
- **Inspección de la prueba fallida.**
- **Cooperación con el demostrador:**
  - **imponiendo restricciones**
  - **probando lemas de reescritura**