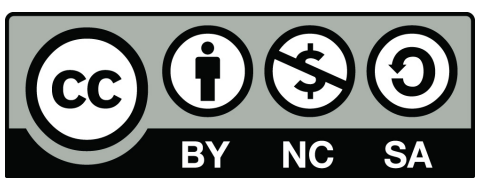
Inteligencia Artificial: Prolog

Recursión

Universidad de La Laguna

Christopher Expósito-Izquierdo[1],
Belén Melián-Batista[2]

{cexposit[1], mbmelian[2]}@ull.es
Universidad de La Laguna (España)



Contenidos

Concepto de Recursión

La recursión es la manera en la que la especificación de un determinado proceso del mundo real se basa en sí misma

Muchos problemas tienen definiciones intrínsecamente recursivas. Algunos ejemplos son los siguientes:

- Torres de Hanoi
- Secuencia de Fibonacci
- Máximo común divisor
- Número factorial
- ...

Recursión en Prolog

La recursión es una herramienta de suma importancia en Prolog para la definición de predicados. En Prolog podemos encontrar, en general, dos tipos de recursión:

- *Recursión directa*. Un cierto predicado `pred` está definido en base a sí mismo
- *Recursión indirecta*. Un determinado predicado `pred1` está definido en base al predicado `pred2`, el cual a su vez está definido en base al predicado `pred3`,... el cual está definido en base al predicado `pred1`

Ejemplo de Recursión

A continuación veamos cómo definir relaciones de parentesco entre personas de una forma recursiva. Comenzamos definiendo un conjunto de hechos simples:

- `padre(marcos, antonio).`
- `padre(antonio, sandra).`
- `padre(luis, marcos).`
- `padre(luis, andrea).`
- `padre(luis, nuria).`

Ahora podemos definir las siguientes reglas:

- `ascendiente(X, Y) :- padre(X, Y).`
- `ascendiente(X, Y) :- padre(X, Z), ascendiente(Z, Y).`

Ejemplo de Recursión

La base de conocimiento la guardamos en un fichero de texto denominado `myKnowledgeBase1.pl`. Este fichero debe ser cargado en el intérprete antes de poder emplear sus predicados. A continuación podemos ver ejemplos del uso:

```
hiper@hiper-System-Product-Name: ~/Dropbox/OCW2015/3
Welcome to SWI-Prolog (Multi-threaded, 64 bits, Version 6.6.6)
Copyright (c) 1990-2013 University of Amsterdam, VU Amsterdam
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software,
and you are welcome to redistribute it under certain conditions.
Please visit http://www.swi-prolog.org for details.

For help, use ?- help(Topic). or ?- apropos(Word).

?- consult(myKnowledgeBase1).
% myKnowledgeBase1 compiled 0.00 sec, 8 clauses
true.

?- ascendiente(X, nuria).
X = luis ;
false.

?- ascendiente(X, sandra).
X = antonio ;
X = marcos ;
X = luis ;
false.

?-
```

El predicado `fibonacci`

Objetivo: calcular la sucesión de Fibonacci. Esta sucesión viene definida por la siguiente expresión matemática:

$$fib(n) = \begin{cases} 0 & \text{if } n = 0 \\ 1 & \text{if } n = 1 \\ fib(n-2) + fib(n-1) & \text{if } n > 1 \end{cases}$$

El predicado propuesto debe tener dos argumentos:

1. El número n
2. El resultado de la sucesión para el número n establecido como primer argumento

El predicado `fibonacci`

El elemento de la sucesión de Fibonacci cuando $n = 0$ es 0.
Se puede expresar en Prolog con el siguiente hecho:

```
fib(0, 0).
```

El predicado `fibonacci`

Al igual que en el caso anterior, el elemento de la sucesión de Fibonacci cuando $n = 1$ es 1.
Se puede expresar en Prolog con el siguiente hecho:

```
fib(1, 1).
```

El predicado `fibonacci`

El elemento de la sucesión de Fibonacci cuando $n > 1$ se calcula como la suma de los elementos de la sucesión de Fibonacci para $n-2$ y $n-1$. Por tanto, en primer lugar hay que comprobar que n toma un valor mayor que 1, calcular los elementos de la sucesión de Fibonacci para $n-2$ y $n-1$ y finalmente sumarlos.
Se puede expresar en Prolog con la siguiente regla:

```
fib(X, Y) :- X > 1, X1 is X - 1, X2 is X - 2, fib(X1,
             Y1), fib(X2, Y2), Y is Y1 + Y2.
```

El predicado `fibonacci`

La definición del predicado `fibonacci` la guardamos en un fichero de texto denominado `myPredicateFib.pl`. Este fichero debe ser cargado en el intérprete antes de poder emplear el predicado. A continuación podemos ver ejemplos del uso:

```
hiper@hiper-System-Product-Name: ~/Dropbox/OCW2015/3
Welcome to SWI-Prolog (Multi-threaded, 64 bits, Version 6.6.6)
Copyright (c) 1990-2013 University of Amsterdam, VU Amsterdam
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software,
and you are welcome to redistribute it under certain conditions.
Please visit http://www.swi-prolog.org for details.

For help, use ?- help(Topic). or ?- apropos(Word).

?- consult(myPredicateFib).
% myPredicateFib compiled 0.00 sec, 4 clauses
true.

?- fib(1, X).
X = 1 .

?- fib(5, X).
X = 5 .

?- fib(10, X).
X = 55 .

?-
```

El predicado `maximum`

Objetivo: calcular el máximo común divisor de dos números. El máximo común divisor de dos números x e y viene definido por la siguiente expresión matemática:

$$mcd(x, y) = \begin{cases} x & \text{if } y = 0 \\ mcd(y, mod(x, y)) & \text{if } y > 0 \wedge x \geq y \end{cases}$$

El predicado propuesto debe tener tres argumentos:

1. El primero de los números cuyo máximo común divisor se desea calcular
2. El segundo de los números cuyo máximo común divisor se desea calcular
3. El máximo común divisor de los números anteriormente definidos

El predicado `maximum`

El máximo común divisor de dos números iguales es ese mismo número.
Se puede expresar en Prolog con el siguiente hecho:

```
maximum(X, X, X).
```

El predicado `maximum`

Si el número x es menor que y se calcula el máximo común divisor de x y la resta $y - x$.
Se puede expresar en Prolog con la siguiente regla:

```
maximum(X, Y, Z) :- X < Y, Y1 is Y - X, maximum(X, Y1,
                                                Z).
```

El predicado `maximum`

Finalmente, si el número x es mayor que y se calcula el máximo común divisor de y y x.
Se puede expresar en Prolog con la siguiente regla:

```
maximum(X, Y, Z) :- X > Y, maximum(Y, X, Z).
```

El predicado `maximum`

La definición del predicado `maximum` la guardamos en un fichero de texto denominado `myPredicateMaximum.pl`. Este fichero debe ser cargado en el intérprete antes de poder emplear el predicado. A continuación podemos ver ejemplos del uso:

```
hiper@hiper-System-Product-Name: ~/Dropbox/OCW2015/3
Welcome to SWI-Prolog (Multi-threaded, 64 bits, Version 6.6.6)
Copyright (c) 1990-2013 University of Amsterdam, VU Amsterdam
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software,
and you are welcome to redistribute it under certain conditions.
Please visit http://www.swi-prolog.org for details.

For help, use ?- help(Topic). or ?- apropos(Word).

?- consult(myPredicateMaximum).
% myPredicateMaximum compiled 0.00 sec, 4 clauses
true.

?- maximum(10, 15, Z).
Z = 5 .

?- maximum(10, 10, Z).
Z = 10 .

?- maximum(82, 127, Z).
Z = 1 .

?-
```

Inteligencia Artificial: Prolog

Recursión

Christopher Expósito-Izquierdo[1],
Belén Melián-Batista[2]

{cexposit[1], mbmelian[2]}@ull.es
Universidad de La Laguna (España)