Práctico 8: Gramáticas. Metaintérpretes.

Ejercicio 1

Describa la sintaxis para definir gramáticas en Prolog [llamadas DCG: *Definite Clause Grammar*] usando una gramática en Prolog.

Ejercicio 2

a) Escriba una gramática en Prolog, usando la notación DCG, que permita reconocer frases como:

Los osos polares comen peces.
Los osos polares viven en la Antártida.
El investigador da alimento a los osos polares.
Los investigadores estudian el comportamiento de las aves.

b) Explique cómo haría para no permitir frases como:

Las investigador estudian el comportamiento de los aves.

c) Explique cómo haría para no permitir frases como:

Los osos polares comen la Antártida.

Ejercicio 3

Defina programas Prolog para reconocer los siguientes lenguajes:

- $L = \{a^n b^m c^{n+m} / n,m>0\}$
- Sentencias *while*, *assign,* expresiones aritméticas y lógicas de un lenguaje imperativo [suponga que la entrada es una lista de tokens]

Ejercicio 4

Considere el siguiente programa Prolog:

```
indice(Patron,Lista,Indice):-
indice(Patron,1,Indice,Lista,_).

indice(Patron,I,I) --> Patron.

indice(Patron,I0,I) -->
      [_],
      {I1 is I0+1},
      indice(Patron,I1,I).
```

Indique su comportamiento frente a cada una de las siguientes consultas:

a) ? indice([a,b,a], [x,y,a,b,a,b,a], I).
   a.1 – La respuesta es NO
   a.2 – La respuesta es I = [a,b]
   a.3 – La respuesta es I = 3 ; I = 5
   a.4 – La respuesta es I =1 ; I = 2 ; I = 3
   a.5 – Ninguna de las anteriores

b) ? indice([a,b,a], [x,y,a,b,b], I).
   b.1 – La respuesta es NO
   b.2 – La respuesta es I = [a,b,b]
   b.3 – La respuesta es I = 3
   b.4 – La respuesta es I =1 ; I = 2 ; I = 3
   b.5 – Ninguna de las anteriores

c) ? indice([], [a,b,x,a], I).
   c.1 – La respuesta es NO
   c.2 – La respuesta es I = [a,b]
   c.3 – La respuesta es I = [a,b,x,a]
   c.4 – La respuesta es I =1 ; I = 4
   c.5 – Ninguna de las anteriores

d) ? indice ([X,Y,Z], [a,b,c,d], _).
   d.1 – La respuesta es NO
   d.2 – La respuesta es X = a , Y = b, Z =c ; X=b, Y=c, Z=d
   d.3 – La respuesta es X = [a,b,c,d], Y = [], Z=[]
   d.4 – La respuesta es I =1 ; I = 2 ; I = 3; I = 4
   d.5 – Ninguna de las anteriores

Ejercicio 5

Implemente los siguientes metaintérpretes en Prolog:

resDerecha(+G) ← resuelve *G* con la siguiente regla de computación: siempre se selecciona el átomo de más a la derecha para la sustitución.

resAviso(+G,+N,+Max) ← resuelve *G* siempre que el árbol de la prueba no supere la profundidad *Max*. Cada vez que se llegue a un árbol de profundidad *N*, el intérprete debe imprimir el siguiente mensaje en la salida estándar: «Cuidado, árbol muy profundo».

resArbol(+G, -R) ← resuelve *G*, siendo *R* la correspondiente rama de prueba del árbol SLD.

Ejercicio 6

En Prolog, el significado de un argumento en un predicado está dado por su posición. Por ejemplo, en el predicado *member* de dos argumentos definido a continuación, el primer argumento es un elemento de la lista y el segundo una lista.

```
member(X,[X|_]).
member(X,[_|Ys])   :- member(X,Ys).
```

Se considera una variante de Prolog en la que:
- TODOS los predicados del usuario sólo tienen un argumento.
- Este único argumento es una lista de pares de la forma *atributo=valor* [*lista AV*].
- Dos listas AV unifican si para todo atributo que aparezca en **ambas** listas los respectivos valores unifican.

En esta variante, el predicado *member* podría escribirse de la siguiente forma:

```
member([elem=X, lista=[X|_]]).
member([lista=[_|Xs], elem=X]) :- member([elem=X, lista=Xs]).
```

Observar que:
- No importa el orden de aparición los pares atributo-valor en una lista AV.
- Si el predicado es de aridad cero, tiene como único argumento a la lista vacía.
- La lista vacía unifica con cualquier lista AV.

Es posible también que en algún uso del predicado aparezca sólo un subconjunto de los atributos. Por ejemplo, la consulta *?member([elem=X])* tiene infinitas soluciones [es equivalente, en un intérprete «normal», a la consulta *?member(X,_)*]

Se pide: escriba un metaintérprete para esta variante de Prolog [puro]

Ejercicio 7 [prueba 03]

a) Construya una gramática DCG para el lenguaje $L=\{a^p\ b^m\ c^{p*m} / m,p>0\}$

b) Escriba un metaintérprete para Prolog puro con negación que difiera la prueba de átomos de la forma **not(X)**, cuando **X** tiene variables sin instanciar, hasta que no queden otros átomos por probar.

Ejercicio 8

Implemente los siguientes metaintérpretes en Prolog:

resOccurCheck(+G) ← resuelve *G*, realizando occur_check en la unificación.

resAmplitud(+G) ← resuelve *G* realizando una búsqueda en amplitud en el árbol SLD.