

Extensiones: Gramáticas Definidas y Restricciones

Curso 2002/03, Fecha:24/01/2002

Enunciado 1 (Aprende) *Modificar el programa Eliza de la práctica anterior para que aprenda nuevos conceptos de forma dinámica. Para ello, se añade un argumento extra a cada uno de los patrones indicando la acción a realizar (en la mayoría de los casos, "nada"), y se incluyen patrones que indican acciones para definir conceptos y eliminar conceptos.*

```
patron ([ adios ], [ 'Hasta ', la , proxima ], nada ).
patron ([ si ], [ puedes , ser , mas , expresivo , ? ], nada ).
patron ([ X , es , Y ], [ vale ], asocia ( X , Y ) ).
patron ([ X , no , es , Y ], [ vale ], elimina ( X , Y ) ).
patron ([ habla , de , X ], [ X , es , Y ], nada) : - asocia ( X , Y ) .
patron ([ 1 ], [ ' Por ', favor , continua ], nada ) .
```

Además, se modifica el predicado *trata* para que ejecute la acción correspondiente

```
trata ( Ps ) : - once ( ( patron ( Estimulo , Respta , Accion ) ,
                      encaja ( Estimulo , Dicc , Ps ) ,
                      encaja ( Respta , Dicc , Salida ) ,
                      ejecuta ( Accion ) ,
                      responde ( Salida )
                    ) ) .
```

```
ejecuta ( nada ) .
ejecuta ( asocia ( X , Y ) ) : - asserta ( asocia ( X , Y ) ) .
ejecuta ( elimina ( X , Y ) ) : - retract ( asocia ( X , Y ) ) .
```

Enunciado 2 (Personalizar(Opcional)) *Modificar el programa Eliza para que al iniciarse lea de un fichero información sobre diferentes personas. Luego, durante la ejecución, el sistema puede aprender nuevas cosas sobre esas personas (por ejemplo, gustos, edad, etc.) y al finalizar, la información aprendida se vuelve a almacenar en el fichero.*

De esa forma, la conversación será diferente en cada ejecución y se adaptará al conocimiento adquirido en las diferentes ejecuciones

Enunciado 3 (Gramatica) *El siguiente programa define una sencilla gramática de castellano.*

$frase(N, C) \longrightarrow$
 $suje(N, C),$
 $predicado(N, C).$

$suje(N, C) \longrightarrow$
 $det(N, C), nombre(N, C).$

$predicado(N, C) \longrightarrow$
 $verbo(N), atributo(N, C).$

$det(singular, masculino) \longrightarrow [el].$
 $det(singular, femenino) \longrightarrow [la].$
 $det(plural, masculino) \longrightarrow [los].$
 $det(plural, femenino) \longrightarrow [las].$

$nombre(singular, masculino) \longrightarrow [perro].$
 $nombre(singular, femenino) \longrightarrow [perra].$
 $nombre(singular, masculino) \longrightarrow [hombre].$
 $nombre(singular, femenino) \longrightarrow [mujer].$
 $nombre(plural, masculino) \longrightarrow [hombres].$
 $nombre(plural, femenino) \longrightarrow [mujeres].$

$verbo(singular) \longrightarrow [esta].$
 $verbo(plural) \longrightarrow [están].$

$atributo(singular, masculino) \longrightarrow [cansado].$
 $atributo(singular, femenino) \longrightarrow [cansada].$
 $atributo(plural, masculino) \longrightarrow [cansados].$
 $atributo(plural, femenino) \longrightarrow [cansadas].$

Probar el programa y ejecutar las siguientes preguntas

- ?-frase(V, W, [el, perro, esta, cansado], []).
- ?-frase(N, femenino, X, []).

Enunciado 4 (Reconocedor(Opcional)) *Incorporar al programa Eliza un reconocedor de frases basado en gramáticas definidas*

Enunciado 5 (Send) *El siguiente predicado resuelve el problema de asignar dígitos a cada una de las letras de forma que se cumpla la siguiente suma*

$S E N D$
 $+ M O R E$

M O N E Y

Compilar y ejecutar el programa

```
send(LD) :-
    LD = [S, E, N, D, M, O, R, Y],
    fd_all_different(LD),
    fd_domain(LD, 0, 9),
    S #\= 0,
    M #\= 0,
    1000*S + 100*E + 10*N + D
    + 1000*M + 100*O + 10*R + E
    #= 10000*M + 1000*O + 100*N + 10*E + Y,
    fd_labeling(LD).
```

Nota: El programa utiliza el sistema de restricciones del GNU-Prolog, por lo que es necesario utilizar dicho compilador.

Enunciado 6 (colorC) El siguiente programa resuelve el problema de colorear quadrees utilizando restricciones.

Compilar y ejecutar el programa (comparar tiempos de ejecución con el algoritmo de coloreado de la práctica 4)

```
% Incluir código de práctica 4
:- include('pl4.pl').

colorC(Q1, Q2) :- coloreaC(Q1, Q),
                  noJuntosC(Q),
                  elemsQ(Q, Xs),
                  fd_labeling(Xs),
                  mapC(Q, Q2).

coloreaC(vacio, vacio).
coloreaC(rect(_), rect(X)) :- fd_domain(X, 0, 2).
coloreaC(d(A, B, C, D), d(A1, B1, C1, D1)) :-
    coloreaC(A, A1), coloreaC(B, B1),
    coloreaC(C, C1), coloreaC(D, D1).

noJuntosC(vacio).
noJuntosC(rect(_)).
noJuntosC(d(A, B, C, D)) :-
    noJuntosC(A), noJuntosC(B), noJuntosC(C), noJuntosC(D),
    hazArbol(der, A, Ar), hazArbol(izq, B, Bl), noEqC(Ar, Bl),
    hazArbol(der, C, Cr), hazArbol(izq, D, Dl), noEqC(Cr, Dl),
```

```

hazArbol(inf, A, Ad), hazArbol(sup, C, Cu), noEqC(Ad, Cu),
hazArbol(inf, B, Bd), hazArbol(sup, D, Du), noEqC(Bd, Du).

```

```

noEqC(vacio, _).
noEqC(_, vacio).
noEqC(rect(X), rect(Y)):-X#\=Y.
noEqC(rect(X), f(A, B)):-noEstaC(X, A), noEstaC(X, B).
noEqC(f(A, B), rect(X)):-noEstaC(X, A), noEstaC(X, B).
noEqC(f(A, B), f(C, D)):-noEqC(A, C), noEqC(B, D).

```

```

noEstaC(_, vacio).
noEstaC(X, rect(Y)):-X#\=Y.
noEstaC(X, f(A, B)):-noEstaC(X, A), noEstaC(X, B).

```

```

% elemsQ(QT, Ls):-Ls contiene la lista de colores
%                   del quadtree QT
elemsQ(vacio, []).
elemsQ(rect(X), [X]).
elemsQ(d(A, B, C, D), Xs):-
    elemsQ(A, As), elemsQ(B, Bs), elemsQ(C, Cs), elemsQ(D, Ds),
    concatls([As, Bs, Cs, Ds], Xs).

```

```

% concatLs(Lss, Ls):-Ls es la lista resultante de concatenar
%                   las listas de Lss
concatls([], []).
concatls([Xs|Xss], Zs):-concatls(Xss, Ys), append(Xs, Ys, Zs).

```

```

mapC(vacio, vacio).
mapC(rect(0), rect(rgb(1, 0, 0))).
mapC(rect(1), rect(rgb(0, 1, 0))).
mapC(rect(2), rect(rgb(0, 0, 1))).
mapC(d(A, B, C, D), d(A1, B1, C1, D1)):-
    mapC(A, A1), mapC(B, B1), mapC(C, C1), mapC(D, D1).

```

Enunciado 7 (reinas) El siguiente ejercicio resuelve el problema de las n reinas utilizando restricciones.

Compilar y ejecutar el programa comparando el tiempo de respuesta con el obtenido en la práctica 2

```

reinas(N, L):-length(L, N),
               fd_domain(L, 1, N),
               segura(L),
               fd_labeling(L).

```

```

segura([]).

```

segura (*[R|Rs]*):- *segura* (*Rs*), *noAtaque* (*R, Rs, 1*).

noAtaque (*- , [] , -*).

noAtaque (*Y, [Y1|Ys], D*):-

Y #\= *Y1*,

Y1 - Y #\= *D*,

Y - Y1 #\= *D*,

D1 is *D + 1*,

noAtaque (*Y, Ys, D1*).

Enunciado 8 (ExamenTest) Resolver el siguiente examen de tipo Test que se auto-referencia

1 : la primer pregunta cuya respuesta es A es:

(A) 4 (B) 3 (C) 2 (D) 1 (E) ninguna de las anteriores

2 : las dos únicas preguntas consecutivas cuyas respuestas son idénticas son:

(A) 3 y 4 (B) 4 y 5 (C) 5 y 6 (D) 6 y 7 (E) 7 y 8

3 : la siguiente pregunta cuya respuesta es A es:

(A) 4 (B) 5 (C) 6 (D) 7 (E) 8

4 : la primer pregunta de número par cuya respuesta es B es:

(A) 2 (B) 4 (C) 6 (D) 8 (E) 10

5 : la única pregunta de número impar cuya respuesta es C es:

(A) 1 (B) 3 (C) 5 (D) 7 (E) 9

6 : Un pregunta cuya respuesta es D...

(A) aparece antes de ésta pero no despues de ésta

(B) aparece después de ésta , pero no antes de ésta

(C) aparece antes y después de ésta

(D) no aparece

(E) ninguna de las anteriores

7 : la última pregunta cuya respuesta es E es:

(A) 5 (B) 6 (C) 7 (D) 8 (E) 9

8 : el número de preguntas cuyas respuestas son consonantes es:

(A) 7 (B) 6 (C) 5 (D) 4 (E) 3

9 : el número de preguntas cuyas respuestas son vocales es:

(A) 0 (B) 1 (C) 2 (D) 3 (E) 4

10: la respuesta a esta pregunta es:

(A) A (B) B (C) C (D) D (E) E

Nota: Traducido de la versión de GNU-Prolog (Fuente original: M. Henz)

El siguiente programa Prolog resuelve dicho ejercicio enunciando las restricciones correspondientes

% Código adaptado de D. Díaz (GNU-Prolog)

examen (*Resp*):-

L=[*Q1, Q2, Q3, Q4, Q5, Q6, Q7, Q8, Q9, Q10*],

$fd_domain(L, 1, 5),$

$Q1\#=1 \#<=> Q4\#=1 \#/\ Q1\#\=1 \#/\ Q2\#\=1 \#/\ Q3\#\=1,$
 $Q1\#=2 \#<=> Q3\#=1 \#/\ Q1\#\=1 \#/\ Q2\#\=1,$
 $Q1\#=3 \#<=> Q2\#=1 \#/\ Q1\#\=1,$
 $Q1\#=4 \#<=> Q1\#=1,$
 $Q1\#=5 \#<=> Q1\#\=1 \#/\ Q2\#\=1 \#/\ Q3\#\=1 \#/\ Q4\#\=1,$

%

$Q2\#=1 \#<=> Q3\#=Q4,$
 $Q2\#=2 \#<=> Q4\#=Q5,$
 $Q2\#=3 \#<=> Q5\#=Q6,$
 $Q2\#=4 \#<=> Q6\#=Q7,$
 $Q2\#=5 \#<=> Q7\#=Q8,$

$Q3\#=1 \#<=> Q4\#=1,$
 $Q3\#=2 \#<=> Q5\#=1 \#/\ Q4\#\=1,$
 $Q3\#=3 \#<=> Q6\#=1 \#/\ Q4\#\=1 \#/\ Q5\#\=1,$
 $Q3\#=4 \#<=> Q7\#=1 \#/\ Q4\#\=1 \#/\ Q5\#\=1 \#/\ Q6\#\=1,$
 $Q3\#=5 \#<=> Q8\#=1 \#/\ Q4\#\=1 \#/\ Q5\#\=1 \#/\ Q6\#\=1 \#/\ Q7\#\=1,$

$Q4\#=1 \#<=> Q2\#=2,$
 $Q4\#=2 \#<=> Q4\#=2 \#/\ Q2\#\=2,$
 $Q4\#=3 \#<=> Q6\#=2 \#/\ Q2\#\=2 \#/\ Q4\#\=2,$
 $Q4\#=4 \#<=> Q8\#=2 \#/\ Q2\#\=2 \#/\ Q4\#\=2 \#/\ Q6\#\=2,$
 $Q4\#=5 \#<=> Q10\#=2 \#/\ Q2\#\=2 \#/\ Q4\#\=2 \#/\ Q6\#\=2 \#/\ Q8\#\=2,$

$Q5\#=1 \#<=> Q1\#=3 \#/\ Q3\#\=3 \#/\ Q5\#\=3 \#/\ Q7\#\=3 \#/\ Q9\#\=3,$
 $Q5\#=2 \#<=> Q3\#=3 \#/\ Q1\#\=3 \#/\ Q5\#\=3 \#/\ Q7\#\=3 \#/\ Q9\#\=3,$
 $Q5\#=3 \#<=> Q5\#=3 \#/\ Q1\#\=3 \#/\ Q3\#\=3 \#/\ Q7\#\=3 \#/\ Q9\#\=3,$
 $Q5\#=4 \#<=> Q7\#=3 \#/\ Q1\#\=3 \#/\ Q2\#\=3 \#/\ Q5\#\=3 \#/\ Q9\#\=3,$
 $Q5\#=5 \#<=> Q9\#=3 \#/\ Q1\#\=3 \#/\ Q3\#\=3 \#/\ Q5\#\=3 \#/\ Q7\#\=3,$

$BeforeQ4 \#<=> Q1\#=4 \#/\ Q2\#=4 \#/\ Q3\#=4 \#/\ Q4\#=4 \#/\ Q5\#=4,$
 $AfterQ4 \#<=> Q7\#=4 \#/\ Q8\#=4 \#/\ Q9\#=4 \#/\ Q10\#=4,$

$Q6\#=1 \#<=> BeforeQ4 \#/\ \#\ AfterQ4,$
 $Q6\#=2 \#<=> \#\ BeforeQ4 \#/\ AfterQ4,$
 $Q6\#=3 \#<=> BeforeQ4 \#/\ AfterQ4,$
 $Q6\#=4 \#<=> Q1\#\=4 \#/\ Q2\#\=4 \#/\ Q3\#\=4 \#/\ Q4\#\=4 \#/\ Q5\#\=4 \#/\$
 $Q6\#\=4 \#/\ Q7\#\=4 \#/\ Q8\#\=4 \#/\ Q9\#\=4 \#/\ Q10\#\=4,$

%

$Q6\#=5 \#<=> Q6\#=4,$

$$\begin{aligned}
Q7\#=1 \#<=> Q5\#=5 \# \wedge Q6\#\=5 \# \wedge Q7\#\=5 \# \wedge Q8\#\=5 \# \wedge Q9\#\=5 \# \wedge \\
Q10\#\=5, \\
Q7\#=2 \#<=> Q6\#=5 \# \wedge Q7\#\=5 \# \wedge Q8\#\=5 \# \wedge Q9\#\=5 \# \wedge Q10\#\=5, \\
Q7\#=3 \#<=> Q7\#=5 \# \wedge Q8\#\=5 \# \wedge Q9\#\=5 \# \wedge Q10\#\=5, \\
Q7\#=4 \#<=> Q8\#=5 \# \wedge Q9\#\=5 \# \wedge Q10\#\=5, \\
Q7\#=5 \#<=> Q9\#=5 \# \wedge Q10\#\=5,
\end{aligned}$$

$$\begin{aligned}
BCD1 \#<=> Q1 \#>= 2 \# \wedge Q1 \#<= 4, AE1 \#<=> \# \setminus BCD1, \\
BCD2 \#<=> Q2 \#>= 2 \# \wedge Q2 \#<= 4, AE2 \#<=> \# \setminus BCD2, \\
BCD3 \#<=> Q3 \#>= 2 \# \wedge Q3 \#<= 4, AE3 \#<=> \# \setminus BCD3, \\
BCD4 \#<=> Q4 \#>= 2 \# \wedge Q4 \#<= 4, AE4 \#<=> \# \setminus BCD4, \\
BCD5 \#<=> Q5 \#>= 2 \# \wedge Q5 \#<= 4, AE5 \#<=> \# \setminus BCD5, \\
BCD6 \#<=> Q6 \#>= 2 \# \wedge Q6 \#<= 4, AE6 \#<=> \# \setminus BCD6, \\
BCD7 \#<=> Q7 \#>= 2 \# \wedge Q7 \#<= 4, AE7 \#<=> \# \setminus BCD7, \\
BCD8 \#<=> Q8 \#>= 2 \# \wedge Q8 \#<= 4, AE8 \#<=> \# \setminus BCD8, \\
BCD9 \#<=> Q9 \#>= 2 \# \wedge Q9 \#<= 4, AE9 \#<=> \# \setminus BCD9, \\
BCD10\#<=> Q10 \#>= 2 \# \wedge Q10 \#<= 4, AE10 \#<=> \# \setminus BCD10,
\end{aligned}$$

$$BCD\#=BCD1+BCD2+BCD3+BCD4+BCD5+BCD6+BCD7+BCD8+BCD9+BCD10,$$

$$AE\#=AE1+AE2+AE3+AE4+AE5+AE6+AE7+AE8+AE9+AE10,$$

$$\begin{aligned}
Q8\#=1 \#<=> BCD\#=7, \\
Q8\#=2 \#<=> BCD\#=6, \\
Q8\#=3 \#<=> BCD\#=5, \\
Q8\#=4 \#<=> BCD\#=4, \\
Q8\#=5 \#<=> BCD\#=3,
\end{aligned}$$

$$\begin{aligned}
Q9\#=1 \#<=> AE\#=0, \\
Q9\#=2 \#<=> AE\#=1, \\
Q9\#=3 \#<=> AE\#=2, \\
Q9\#=4 \#<=> AE\#=3, \\
Q9\#=5 \#<=> AE\#=4,
\end{aligned}$$

$$\begin{aligned}
&fd_labeling(L), \\
&mapResp(L, Resp).
\end{aligned}$$

$$\begin{aligned}
mapResp([X|L], [Y|M]):-cnv(X,Y), mapResp(L,M). \\
mapResp([], []).
\end{aligned}$$

$$cnv(1,a). cnv(2,b). cnv(3,c). cnv(4,d). cnv(5,e).$$

Enunciado 9 (Pasatiempos(Opcional)) Plantear y resolver mediante programación con restricciones pasatiempos lógicos