


UNIVERSIDAD DE GUADALAJARA

Centro Universitario de Ciencias Exactas e Ingenierías

Departamento de Ciencias Computacionales

Seminario de Solución de Problemas de Sistemas Basados en Conocimiento

Practica 7

Profesor:	**Sección**:	**Fecha:**
Valdes Lopez Julio Esteban	D05	02/10/2022
Alumno:	**Código:**	**Carrera:**
Sandoval Padilla Fernando Cesar	215685409	Ingeniería informática

Veamos un ejemplo simple donde el backtracking es innecesario (e ineficiente).

Ejemplo 1
R1: si X<3 entonces Y=0
R2: si 3=<X y X<6 entonces Y=2
R3: si 6=<X entonces Y=4

Solución 1:

```
f(X,0):- X<3.
f(X,2):- 3=<X,X<6.
f(X,4):- 6=<X.
```

El backtracking es innecesario ya que las funciones son mutuamente excluyentes, los posibles valores de X solo se unificaran a una de las tres reglas.

Debemos intentar encontrar una solución para esta ineficiencia: el corte.

1. Comportamiento del corte en PROLOG

El corte es un predicado predefinido, cuya sintaxis es “!” y cuyo comportamiento es el siguiente.
Suponemos definido el predicado “a” con las siguientes cláusulas

```
a :-b,!,c.
a :-d.
```

Para comprobar o demostrar que el predicado a es cierto, pasamos a comprobar si es cierto el predicado b. Pueden ocurrir dos cosas:

1. Si b no es cierto, mediante backtracking, se comprobará si d es cierto. Si d es cierto, quedará demostrado que a es cierto.

Hasta aquí el predicado a tiene un comportamiento normal en PROLOG.

2. Si b es cierto, se sobrepasa la "barrera" del corte que significa:
    - Que no se podrá hacer backtracking para tomar otras posibilidades alternativas en el predicado b.
    - Que no se pueden considerar otras posibles definiciones del predicado a (es decir, no podemos acceder a a :- d.).

Si c es cierto, entonces se ha demostrado a, pero si c no se cumple, entonces el predicado a falla.
Intentemos, utilizando el corte, ver si hay alguna solución buena para el problema anterior.

Solución 2:

f(X,0):- X<3,!.

f(X,2):- 3=<X, X<6,!.

f(X,4):- 6=<X.

1:-Comprueba y analiza con trace los resultados en PROLOG para las tres situaciones, usando el comando "trace", antes y después de agregar la función corte (Solución 1 y solución 2).

Solución 1:

```
[trace]  ?- f(2,Y).
   Call: (10) f(2, _2238) ? creep
   Call: (11) 2<3 ? creep
   Exit: (11) 2<3 ? creep
   Exit: (10) f(2, 0) ? creep
Y = 0 .

[trace]  ?- f(3,Y).
   Call: (10) f(3, _7784) ? creep
   Call: (11) 3<3 ? creep
   Fail: (11) 3<3 ? creep
   Redo: (10) f(3, _7784) ? creep
   Call: (11) 3=<3 ? creep
   Exit: (11) 3=<3 ? creep
   Call: (11) 3<6 ? creep
   Exit: (11) 3<6 ? creep
   Exit: (10) f(3, 2) ? creep
Y = 2 .

[trace]  ?- f(6,Y).
   Call: (10) f(6, _17100) ? creep
   Call: (11) 6<3 ? creep
   Fail: (11) 6<3 ? creep
   Redo: (10) f(6, _17100) ? creep
   Call: (11) 3=<6 ? creep
   Exit: (11) 3=<6 ? creep
   Call: (11) 6<6 ? creep
   Fail: (11) 6<6 ? creep
   Redo: (10) f(6, _17100) ? creep
   Call: (11) 6=<6 ? creep
   Exit: (11) 6=<6 ? creep
   Exit: (10) f(6, 4) ? creep
Y = 4.

[trace]  ?-
```

Solución 2:

```
[trace]  ?- f(2,Y).
   Call: (10) f(2, _28084) ? creep
   Call: (11) 2<3 ? creep
   Exit: (11) 2<3 ? creep
   Exit: (10) f(2, 0) ? creep
Y = 0.

[trace]  ?- f(3,Y).
   Call: (10) f(3, _33036) ? creep
   Call: (11) 3<3 ? creep
   Fail: (11) 3<3 ? creep
   Redo: (10) f(3, _33036) ? creep
   Call: (11) 3=<3 ? creep
   Exit: (11) 3=<3 ? creep
   Call: (11) 3<6 ? creep
   Exit: (11) 3<6 ? creep
   Exit: (10) f(3, 2) ? creep
Y = 2.

[trace]  ?- f(6,Y).
   Call: (10) f(6, _41758) ? creep
   Call: (11) 6<3 ? creep
   Fail: (11) 6<3 ? creep
   Redo: (10) f(6, _41758) ? creep
   Call: (11) 3=<6 ? creep
   Exit: (11) 3=<6 ? creep
   Call: (11) 6<6 ? creep
   Fail: (11) 6<6 ? creep
   Redo: (10) f(6, _41758) ? creep
   Call: (11) 6=<6 ? creep
   Exit: (11) 6=<6 ? creep
   Exit: (10) f(6, 4) ? creep
Y = 4.

[trace]  ?-
```

Ejemplo 2

Dado el programa:

hola(3):- !.

hola(6).

hola(X):- Y=X-1, hola(Y), write(Y).

Contesta a los siguientes objetivos sin ejecutar el programa,

¿cual crees que te dará recursividad infinita?.

El hola(4) y hola(9) porque no están incluidos en la función después del corte.

2:- hola(4)

3:- hola(9)

4:- hola(X)

Comprueba si tus respuestas son correctas realizando un trace para cada opción.

Hola(4).

```
[trace]  ?- hola(4).
   Call: (10) hola(4) ? creep
   Call: (11) _24872=4-1 ? creep
   Exit: (11) 4-1=4-1 ? creep
   Call: (11) hola(4-1) ? creep
   Call: (12) _27138=4-1-1 ? creep
   Exit: (12) 4-1-1=4-1-1 ? creep
   Call: (12) hola(4-1-1) ? creep
   Call: (13) _29404=4-1-1-1 ? creep
   Exit: (13) 4-1-1-1=4-1-1-1 ? creep
   Call: (13) hola(4-1-1-1) ? creep
   Call: (14) _31670=4-1-1-1-1 ? creep
   Exit: (14) 4-1-1-1-1=4-1-1-1-1 ? creep
   Call: (14) hola(4-1-1-1-1) ? creep
   Call: (15) _33936=4-1-1-1-1-1 ? creep
   Exit: (15) 4-1-1-1-1-1=4-1-1-1-1-1 ? creep
   Call: (15) hola(4-1-1-1-1-1) ? creep
   Call: (16) _36202=4-1-1-1-1-1-1 ? creep
   Exit: (16) 4-1-1-1-1-1-1=4-1-1-1-1-1-1 ? creep
   Call: (16) hola(4-1-1-1-1-1-1) ? creep
   Call: (17) _38468=4-1-1-1-1-1-1-1 ?
```

Hola(9)

```
[trace]  ?- hola(9).
   Call: (10) hola(9) ? creep
   Call: (11) _21618=9-1 ? creep
   Exit: (11) 9-1=9-1 ? creep
   Call: (11) hola(9-1) ? creep
   Call: (12) _23884=9-1-1 ? creep
   Exit: (12) 9-1-1=9-1-1 ? creep
   Call: (12) hola(9-1-1) ? creep
   Call: (13) _26150=9-1-1-1 ? creep
   Exit: (13) 9-1-1-1=9-1-1-1 ? creep
   Call: (13) hola(9-1-1-1) ? creep
   Call: (14) _28416=9-1-1-1-1 ? creep
   Exit: (14) 9-1-1-1-1=9-1-1-1-1 ? creep
   Call: (14) hola(9-1-1-1-1) ? creep
   Call: (15) _30682=9-1-1-1-1-1 ? creep
   Exit: (15) 9-1-1-1-1-1=9-1-1-1-1-1 ? creep
   Call: (15) hola(9-1-1-1-1-1) ? creep
   Call: (16) _32948=9-1-1-1-1-1-1 ? creep
   Exit: (16) 9-1-1-1-1-1-1=9-1-1-1-1-1-1 ? creep
   Call: (16) hola(9-1-1-1-1-1-1) ? creep
   Call: (17) _35214=9-1-1-1-1-1-1-1 ? creep
   Exit: (17) 9-1-1-1-1-1-1-1=9-1-1-1-1-1-1-1 ? creep
   Call: (17) hola(9-1-1-1-1-1-1-1) ? creep
   Call: (18) _37480=9-1-1-1-1-1-1-1-1 ? creep
   Exit: (18) 9-1-1-1-1-1-1-1-1=9-1-1-1-1-1-1-1-1 ? creep
   Call: (18) hola(9-1-1-1-1-1-1-1-1) ?
```

Hola(X).

```
% c:/Users/ferna/OneDrive/Documentos/Prolog/practica 7.pl compiled 0.00 sec, 3 clauses
?- trace.
true.

[trace]  ?- hola(X).
   Call: (10) hola(_20466) ? creep
   Exit: (10) hola(3) ? creep
X = 3.

[trace]  ?-
```

Ejemplo 3

Considera el siguiente programa:

a(X):- b(X).

a(X):- f(X).

b(X):- g(X), !, v(X).

b(X):- X=4, v(X).

g(1).

g(2).

g(3).

v(_).

f(5).

5:-Ejecuta con usando el comando "trace" a(X), y analiza los resultados también

al ejecutar los objetivos a(2), a(3) y a(4).

A(X)

```
[trace]  ?- a(X).
   Call: (10) a(_92786) ? creep
   Call: (11) b(_92786) ? creep
   Call: (12) g(_92786) ? creep
   Exit: (12) g(1) ? creep
   Call: (12) v(1) ? creep
   Exit: (12) v(1) ? creep
   Exit: (11) b(1) ? creep
   Exit: (10) a(1) ? creep
X = 1 ;
   Redo: (10) a(_92786) ? creep
   Call: (11) f(_92786) ? creep
   Exit: (11) f(5) ? creep
   Exit: (10) a(5) ? creep
X = 5.
```

A(2)

```
[trace]  ?- a(2).
   Call: (10) a(2) ? creep
   Call: (11) b(2) ? creep
   Call: (12) g(2) ? creep
   Exit: (12) g(2) ? creep
   Call: (12) v(2) ? creep
   Exit: (12) v(2) ? creep
   Exit: (11) b(2) ? creep
   Exit: (10) a(2) ? creep
true ;
   Redo: (10) a(2) ? creep
   Call: (11) f(2) ? creep
   Fail: (11) f(2) ? creep
   Fail: (10) a(2) ? creep
false.
```

A(3)

```
[trace]  ?- a(3).
   Call: (10) a(3) ? creep
   Call: (11) b(3) ? creep
   Call: (12) g(3) ? creep
   Exit: (12) g(3) ? creep
   Call: (12) v(3) ? creep
   Exit: (12) v(3) ? creep
   Exit: (11) b(3) ? creep
   Exit: (10) a(3) ? creep
true ;
   Redo: (10) a(3) ? creep
   Call: (11) f(3) ? creep
   Fail: (11) f(3) ? creep
   Fail: (10) a(3) ? creep
false.
```

A(4)

```
[trace]  ?- a(4).
   Call: (10) a(4) ? creep
   Call: (11) b(4) ? creep
   Call: (12) g(4) ? creep
   Fail: (12) g(4) ? creep
   Redo: (11) b(4) ? creep
   Call: (12) v(4) ? creep
   Exit: (12) v(4) ? creep
   Exit: (11) b(4) ? creep
   Exit: (10) a(4) ? creep
true ;
   Redo: (10) a(4) ? creep
   Call: (11) f(4) ? creep
   Fail: (11) f(4) ? creep
   Fail: (10) a(4) ? creep
false.
```

6:-¿Cuál sería la respuesta si quitásemos el corte que aparece en la tercera línea?

X=1 x=2 x=3 x=4 x=5

Ejemplo 4

Se considera el siguiente programa:

p(X):- q(X),r(X).

q(a).

r(a).

q(b).

r(X):-s(X).

s(b).

Si se ejecuta el objetivo p(X) se obtiene como respuesta X=a

y X=b.

```
% c:/users/ferna/onedrive/documentos/prol
[trace]  ?- p(X).
   Call: (10) p(_211826) ? creep
   Call: (11) q(_211826) ? creep
   Exit: (11) q(a) ? creep
   Call: (11) r(a) ? creep
   Exit: (11) r(a) ? creep
   Exit: (10) p(a) ? creep
X = a ;
   Redo: (11) r(a) ? creep
   Call: (12) s(a) ? creep
   Fail: (12) s(a) ? creep
   Fail: (11) r(a) ? creep
   Redo: (11) q(_211826) ? creep
   Exit: (11) q(b) ? creep
   Call: (11) r(b) ? creep
   Call: (12) s(b) ? creep
   Exit: (12) s(b) ? creep
   Exit: (11) r(b) ? creep
   Exit: (10) p(b) ? creep
X = b.
```

7:-Modificar el programa insertando el predicado corte en el lugar adecuado para que el sistema conteste sólo X=a.

p(X):- q(X),!,r(X).

```
% c:/users/ferna/onedrive/documentos/prolog/practica 7 compiled 0.00 sec, 0 clauses
[trace]  ?- p(X).
   Call: (10) p(_243866) ? creep
   Call: (11) q(_243866) ? creep
   Exit: (11) q(a) ? creep
   Call: (11) r(a) ? creep
   Exit: (11) r(a) ? creep
   Exit: (10) p(a) ? creep
X = a ;
   Redo: (11) r(a) ? creep
   Call: (12) s(a) ? creep
   Fail: (12) s(a) ? creep
   Fail: (11) r(a) ? creep
   Fail: (10) p(_243866) ? creep
false.

[trace]  ?-
```