# Solución Evaluación (1ª instancia)

**Duración:** 3 horas

## Ejercicio 0: Pregunta individual sobre el laboratorio

Describa la estrategia utilizada para el jugador `ia_det`

## Ejercicio 1 [30 puntos]

Considere un tablero de tamaño N, donde cada casilla tiene un color, rojo o negro, como se muestra en la figura:

RNRN
NRRN
RRRR
NNRR

Las filas del tablero están numeradas de arriba hacia abajo y las columnas de izquierda a derecha, ambas comenzando en 1.

Se quiere obtener, para una cuadrícula dada, el camino más largo partiendo de una casilla que solamente incluya celdas de color rojo, y donde no se repitan celdas, moviéndose solamente en dirección horizontal o vertical. Por ejemplo, la secuencia de celdas `[(1,3),(2,3),(3,3),(3,4)]` es un camino válido, donde cada elemento corresponde a una posición denotada por `(Fila,Columna)`.

Cada tablero está representado por un término de Prolog con functor `tablero`, y con tantos argumentos como filas tenga el tablero. Cada fila, a su vez, está representada por el functor `f`, con tantos argumentos como celdas tenga la fila. Cada fila tendrá el valor `rojo` o `negro`, según corresponda. Por lo tanto, el tablero de ejemplo presentado anteriormente se representará como:

```
tablero(f(rojo,negro,rojo,negro),f(negro,rojo,rojo,negro),f(rojo,rojo,rojo,rojo),f(negro,negro,rojo,rojo))
```

Implemente en Prolog los siguientes predicados:

**a)** `color(+Tablero,?Nfila,?NColumna,?Color)`: la celda `(NFila,NColumna)` tiene color `Color`.

**b)** `paso_valido(+Tablero,+NFilaO,+NColumnaO,?NFilaD,?NColumnaD)`: se puede pasar de la celda `(NFilaO,NColumnaO)` a la celda `(NFilaD, NColumnaD)`. Es decir que ambas celdas son adyacentes vertical u horizontalmente, y tienen el mismo color.

**c)** `camino_rojo(+Tablero,+NFilaO,+NColumnaO,?NFilaD,?NColumnaD,?CaminoRojo)`: existe un camino `CaminoRojo` entre la celda `(NFilaO,NColumnaO)` y `(NFilaD, NColumnaD)`, compuesto completamente de celdas de color rojo.

**d)** `camino_rojo_mas_largo(+Tablero,+NFilaO,+NColumnaO,?CaminoMasLargo)`: `CaminoMasLargo` es el camino rojo más largo que parte de la celda `(NFilaO,NColumnaO)`.
Observación: Puede existir más de un camino con largo maximal, se pide que este predicado obtenenga alguno de ellos.

### Solución

**a)**

```
%%% Tarea: buscar el camino más largo de color rojo en una cuadrícula, donde cada punto de la cuadrícula
%%% es de color rojo o negro. Podemos movernos en horizontal o vertical, empezar por cualquier lugar
%%% pero no podemos pasar dos veces por el mismo punto

%%% Predicado que obtiene el el color de la posición (NFila,NColumna). El tablero está numerado de arriba hacia bajo y de izquierda
%%% a derecha
```

```
color(Tablero,NFila,NColumna,Color):-
    %% Obtengo la n-esima fila
    %% Esto también podría resolverse usando los predicados univ y nth1
    arg(NFila,Tablero,Fila),
    %% Obtengo el color de la posición NColumna
    arg(NColumna,Fila,Color).
```

**b)**

```
%%% Predicado que obtiene los adycentes a una posición dada
%%% Para ello, tienen que ser una columna antes o después en la misma línea
%%% O una línea antes o después en la misma columna
%%% Y ambos del mismo color


%%% Estrictamente, no es necesario chequear el tamaño del tablero
%%% ya que, si la posición no es válida, el predicado color va a fallar

paso_valido(Tablero,NFilaO,NColumnaO,NFilaD ,NColumnaD):-
    adyacente(N,NFilaO,NColumnaO,NFilaD,NColumnaD),
    color(Tablero,NFilaO,NColumnaO,Color),
    color(Tablero,NFilaD,NColumnaD,Color).

%%% Tamaño del tablero
tam(Tablero,N):-
    Tablero =.. [tablero|Filas],
    length(Filas,N).



%%% Misma fila, siguiente columna
adyacente(N,X,Y,X,Y1) :- Y1 is Y+1,Y1 =< N.

%%% Misma fila, anterior columna
adyacente(_,X,Y,X,Y1) :- Y1 is Y-1, Y1 >= 0.

%%% Misma columna, siguiente fila
adyacente(N,X,Y,X1,Y) :- X1 is X+1, X1=<N.

%%% Misma columna, anterior fila
adyacente(_,X,Y,X1,Y) :- X1 is X-1, X1 >=1.
```

**c)**

```
%%%% Un nodo es un camino, si es el del color que buscamos
camino_rojo(Tablero,FilaO,ColumnaO,FilaD,ColumnaD,Camino)                                    :-
camino_rojo(Tablero,FilaO,ColumnaO,FilaD,ColumnaD,[],Camino).

camino_rojo(Tablero,FilaO,ColumnaO,FilaO,ColumnaO,Visitados,[(FilaO,ColumnaO)]):-
    not(member((FilaO,ColumnaO),Visitados)),
    color(Tablero,FilaO,ColumnaO,rojo).

%%% Definición recursiva de camino
          camino_rojo(Tablero,FilaO,ColumnaO,FilaD,ColumnaD,Visitados,[(FilaO,ColumnaO)|
Camino]):-
        not(member((FilaO,ColumnaO),Visitados)),
        paso_valido(Tablero,FilaO,ColumnaO,FilaY,ColumnaY),
                    camino_rojo(Tablero,FilaY,ColumnaY,FilaD,ColumnaD,[(FilaO,ColumnaO)|
Visitados],Camino).
```

**d)**

```
%%% Camino más largo que parte de un punto
camino_rojo_mas_largo(Tablero,FilaO,ColumnaO,CaminoMasLargo):-
    findall(Camino,camino_rojo(Tablero,FilaO,ColumnaO,_,_,Camino),Caminos),
    mas_largo(Caminos,CaminoMasLargo).


%%%% Obtengo el camino más largo de una lista de caminos
mas_largo(Camino,CaminoMasLargo):- mas_largo(Camino,[],CaminoMasLargo).

mas_largo([],CaminoMasLargo,CaminoMasLargo).

mas_largo([Camino|Y],CaminoAct,CaminoMasLargo):-
    length(Camino,N),
    length(CaminoAct,NAct),
    N>NAct,
    mas_largo(Y,Camino,CaminoMasLargo).

mas_largo([Camino|Y],CaminoAct,CaminoMasLargo):-
    length(Camino,N),
```

```
    length(CaminoAct,NAct),
    N=< NAct,
    mas_largo(Y,CaminoAct,CaminoMasLargo).
```

## Ejercicio 2 [15 puntos]

Indique si son verdaderas o falsas las siguientes afirmaciones. Fundamente.

i) Todos los literales de una cláusula definida deben ser positivos.
**Falso:** Por definición las cláusulas definidas deben tener exactamente un literal positivo, pero además pueden tener cualquier cantidad de literales negativos.

ii) La resolvente entre un objetivo definido y una cláusula definida es un objetivo definido.
**Verdadero:** La cláusula definida tiene un literal positivo que se cancela con uno negativo del objetivo definido, dando como resultado otro objetivo definido (todos los literales negativos).

iii) Un conjunto de fórmulas cerradas es satisfactible si existe al menos una interpretación que sea modelo del conjunto.
**Verdadero:** Por definición de satisfactible.

iv) Si no existe ninguna respuesta computada para una consulta, entonces no existe ninguna respuesta correcta.
**Verdadero:** Toda respuesta correcta se puede construir como la composición de una respuesta computada y una sustitución. Si no hay respuestas computadas, tampoco se puede encontrar ninguna respuesta correcta.

v) La sustitución θ={X/a, Y/a} es un m.g.u. para las expresiones p(X) y p(Y).
**Falso:** Esta sustitución θ es un unificador, pero no es el más general porque unifica con un átomo particular. Un m.g.u. para estas expresiones podría ser μ={X/Y}, y a partir de este se puede construir θ componiendo θ=μ∘{Y/a}.

## Ejercicio 3 [25 puntos]

Considere el siguiente programa Prolog:

```
f(a,b).
f(c,c).

g(X) :- f(X,X).
g(X) :- f(Y,X), g(Y).
g(X) :- f(X,Y), X\=Y.
```

a) Dibuje el árbol SLD correspondiente a la consulta **`?- g(Z)`** suponiendo que la regla de computación toma el átomo de más a la izquierda para la resolución.
b) ¿Qué respuestas dará el intérprete Prolog para el objetivo anterior?
c) ¿Cuáles son las respuestas si cambiamos la segunda cláusula del predicado `g/1` por la siguiente? Justifique.

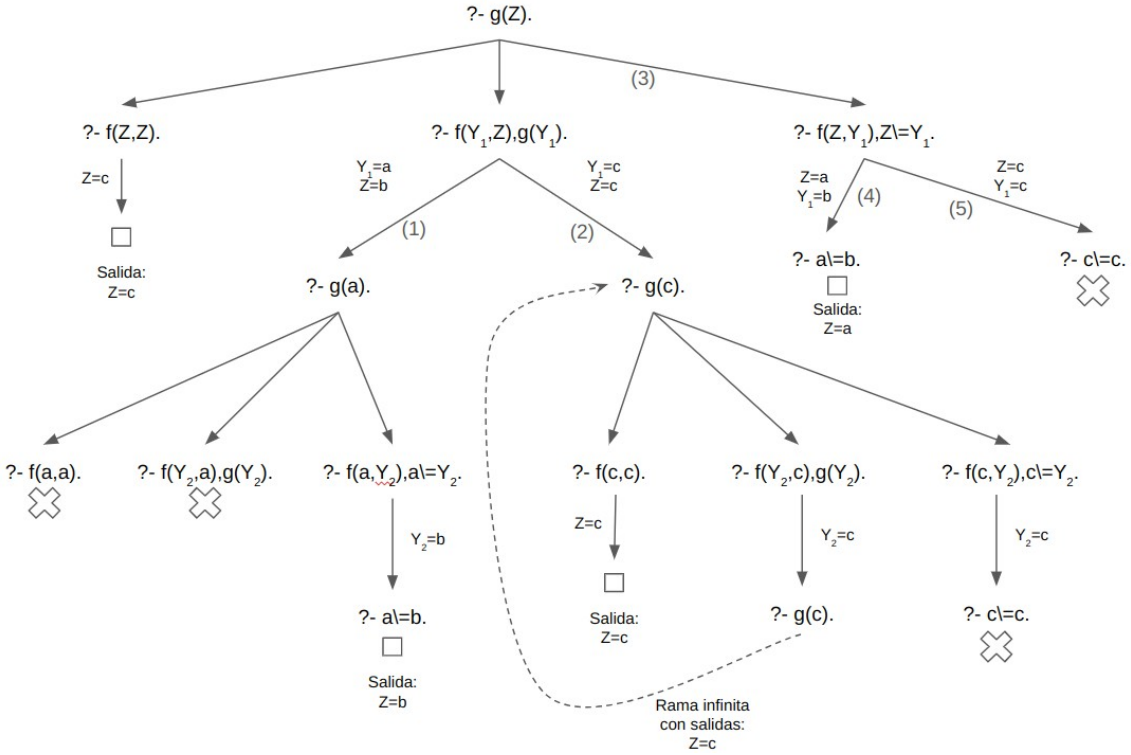
```
g(X) :- f(Y,X), !, g(Y).
```

d) ¿Cuáles son las respuestas si cambiamos la primera cláusula del predicado `f/2` por la siguiente? Justifique.

```
f(a,b) :- !.
```

**Solución**

a)




b) Las salidas son: c, b, c, c, c... cae por una rama infinita devolviendo siempre c.

c) Este cut se activa luego de pasar por el arco marcado con (1), y corta las ramas (2) y (3). Por lo tanto las salidas que quedan son: c, b.

d) Este cut se activa luego de pasar por el arco marcado con (1), pero en este caso corta solo la rama (2). Luego se activa otra vez en el arco (4), y corta la (5). Por lo tanto las salidas que quedan son: c, b, a.

## Ejercicio 4 [15 puntos]

Considere un escenario donde hay dos dados balanceados y tres urnas con bolas de colores: la primera tiene solamente bolas rojas, la segunda solamente bolas azules, y la tercera un tercio de bolas blancas, un tercio de bolas azules y un tercio de bolas rojas. Se lanzan los dados, y si la suma de ambos es 10, se elige una bola al azar de la primera urna; si la suma es menor que 5, se elige una bola al azar de la segunda urna; en otro caso, se elige una bola al azar de la tercera urna.

i) Construya un program en Problog que permita calcular las probabilidades de sacar una bola roja, una bola azul o una bola blanca.

**Solución**

```
%% Dados equilibrados
1/6::dado(X,1) ; 1/6:: dado(X,2) ; 1/6::dado(X,3)  ;1/6:: dado(X,4); 1/6::
dado(X,5) ;1/6:: dado(X,6).

%% Urnas
urna1(rojo).
urna2(azul).
1/3::urna3(rojo) ; 1/3::urna3(azul) ; 1/3::urna3(blanco).

suma_dados(S) :-
    dado(1,X),
    dado(2,Y),
    S is X+Y.
```

```
bola_urna(Color) :-
    suma_dados(S),
    S =:= 10,
    urna1(Color).

bola_urna(Color) :-
    suma_dados(S),
    S < 5,
    urna2(Color).

bola_urna(Color) :-
    suma_dados(S),
    S >= 5, S =\= 10,
    urna3(Color).

query(bola_urna(Color)).
```

ii) ¿Cuál es la probabilidad de obtener una bola blanca si sabemos que el primer dado vale 4? ¿Cómo se expresa esto en Problog?

**Solución**

La probabilidad es 5/6 * 1/3 (son las probabilidades de obtener un 6 en el segundo dado y de elegir luego una bola blanca). Para expresarlo en Problog, basta agregar:

```
evidence(dado(1,4)).
```

al programa anterior.