

Evaluación (1ª instancia)

Duración: 3 horas

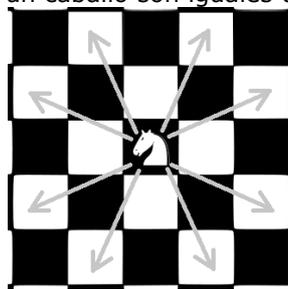
Ejercicio 1 [evaluación individual del laboratorio]

- Describa las funciones objetivo que implementaron en su solución del laboratorio, tanto para el zorro como para los gansos.
- Describa la estrategia minimax.

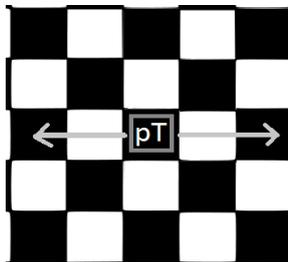
Ejercicio 2 [40 puntos]

Considere un tablero de $m \times n$ celdas y un caballo de ajedrez que busca recorrer la mayor cantidad posible de celdas, sin pasar más de una vez por una misma celda, y con la restricción de que existen otras piezas en el tablero. A las otras piezas se les llama *pseudotorres*, las cuales únicamente pueden atacar celdas de su misma fila.

Los movimientos que puede hacer un caballo son iguales que en el ajedrez:



Las pseudotorres solo tienen movimiento horizontal:



Implemente en Prolog los predicados que se describen a continuación:

a) `celdas_atacadas(+Filas, +Columnas, +ListaTorres, ?Tablero) <-`

Las celdas de Tablero tienen el valor *atacada* si la celda es atacada por alguna torre de ListaTorres, o el valor *libre* en caso contrario. ListaTorres es una lista de naturales entre 1 y Filas inclusive, que indica en qué filas hay pseudotorres (puede suponer que hay como máximo 1 torre por fila).

b) `recorrida_caballo(+Filas, +Columnas, +X, +Y, +Tablero, -LargoRecorrida) <-`

LargoRecorrida es la cantidad de celdas que recorre un caballo comenzando en la celda (X, Y), sin ocupar en ningún momento una celda que contenga el valor *atacada* en Tablero y sin pasar más de una vez por ninguna celda, para un tablero de tamaño Filas X Columnas.

Suponga que ya cuenta con el predicado:

`movimiento_caballo(+Filas, +Columnas, +X, +Y, -X1, -Y1) <-`
(X1, Y1) es un movimiento válido de un caballo que se encuentra en la posición (X, Y) para un tablero de tamaño Filas X Columnas.

c) `todas_recorridas(+Filas, +Columnas, +X, +Y, +Tablero, -ListaLargos) <-`

ListaLargos es una lista con los largos correspondientes a todas las recorridas posibles de un caballo a partir de la posición inicial (X,Y), manteniendo las mismas restricciones que en la parte b.

NOTA: Asuma que cuenta con los siguientes predicados ya implementados para el manejo de matrices. La implementación interna de la matriz no es relevante, debido a que solamente utilizará los predicados para manipularla.

matrix(+M,+N,+Cell,?Matrix) ← Matrix es una matriz de **M** filas y **N** columnas donde cada celda tiene el valor **Cell**.

get_cell(+Matrix,+X,+Y,?Cell) ← Cell es el valor de la celda en la fila **X**, columna **Y** de la matriz **M**.

set_cell(+Matrix,+X,+Y,+Cell,?Matrix2) ← Matrix2 es una matriz igual a **Matrix**, sustituyendo la celda (X,Y) por el valor **Cell**.

Ejercicio 3 [30 puntos]

Dado el siguiente conjunto de cláusulas para los predicados **p, q, r**:

```
Cp1: p(X,X).
Cp2: p(X,Y) :- q(X,Y), p(Y,X).
Cp3: p(X,Y) :- r(X,Y).
Cq1: q(a,b).
Cq2: q(a,a).
Cq3: q(b,c).
Cr1: r(e,e).
Cr2: r(a,b).
```

a) Dibuje el árbol SLD correspondiente a la consulta **?- p(a,V)**, suponiendo que la regla de computación toma el átomo de más a la izquierda para la resolución. Indique para cada rama de éxito cuál es la respuesta computada correspondiente.

b) ¿Qué respuestas dará un intérprete prolog que aplique la regla de computación de la parte a), tomando las cláusulas en su orden de aparición y haciendo recorrida en profundidad del árbol?

c) Diga cuáles serían las respuestas de la parte b) si las cláusulas de **p** estuvieran en el siguiente orden:

i. Cp1	ii. Cp3
Cp3	Cp2
Cp2	Cp1

Justifique su respuesta.

d) Indique qué ramas del árbol de la parte a) son podadas por los cuts de las siguientes variantes del conjunto de cláusulas de **p**. Diga qué respuestas daría en cada caso un intérprete con las características de la parte b).

```
i. Cp1: p(X,X) :- !.
Cp2: p(X,Y) :- q(X,Y), p(Y,X).
Cp3: p(X,Y) :- r(X,Y).

ii. Cp1: p(X,X).
Cp2: p(X,Y) :- q(X,Y), !, p(Y,X).
Cp3: p(X,Y) :- r(X,Y).

iii. Cp1: p(X,X).
Cp2: p(X,Y) :- q(X,Y), p(Y,X), !.
Cp3: p(X,Y) :- r(X,Y).
```

Ejercicio 4 [15 puntos]

Implemente en Prolog el metaintérprete **resNoInstanciado(+G)** que funciona de la siguiente manera:

Si durante la resolución se encuentra una invocación al predicado **is/2**, se debe verificar que el lado derecho de la invocación esté completamente instanciado. Si no lo está, en vez de finalizar la ejecución con un error, se debe imprimir un mensaje de advertencia "El lado derecho no está instanciado" y fallar, permitiendo que Prolog continúe con el backtracking. Asuma que el único predicado built-in que puede utilizar **G** es el predicado **is/2**.