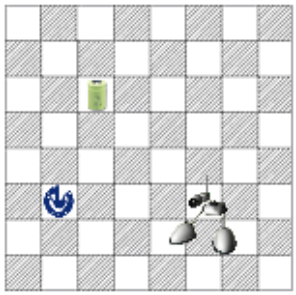
Evaluación

Ejercicio 1 [evaluación individual del laboratorio]

a) Describa la estructura que utilizó para representar internamente el estado del tablero.

b) Explique cómo se actualiza dicha estructura cada vez que se realiza una jugada completa (todos los movimientos correspondientes a una tirada de dados). Especifique si la actualización se hace paso a paso, representando cada movimiento, o se hace al final de la jugada; qué elementos de la estructura se modifican; cuáles son los diferentes casos que se consideran. Se asume que todos los movimientos son válidos y no se pide deshacer.

Ejercicio 2 [25 puntos]



Un robot tiene como objetivo llegar a la base para recargar sus baterías de repuesto. Primero debe encontrarlas, dado que las perdió en algún lugar de su campo de acción. Llegado a la base, que tampoco sabe dónde está, tendrá que comenzar el programa de recarga.

El campo de trabajo es un área rectangular, rodeada de paredes electrificadas. El robot debe tener cuidado en no intentar atravesarlas, ya que la electricidad quemaría sus circuitos.

El campo puede ser visto como una cuadrícula (de tamaño desconocido), en donde el robot se mueve de un casillero a otro, en cuatro posibles direcciones: sur, norte, este y oeste. Además, el robot cuenta con un visor que le permite determinar qué objetos se encuentran en su mismo casillero (la base, las baterías, las paredes electrificadas, etc.).

Se cuenta con los siguientes predicados para manejar el robot:

- **visible**(-Objetos) ← *Objetos* es una lista de elementos captados por el visor del robot. Los elementos de esta lista podrían ser *batería*, *cargador*, *pared_norte*, *pared_sur, pared_este* y *pared_oeste*, entre otros.
- **mover**(+Direcc) ← El robot se mueve un casillero en la dirección especificada (*Direcc*). Esta puede ser uno de los siguientes valores: *norte*, *sur*, *este* u *oeste.*
- **obtener**(+Objeto) ← Obtiene el *Objeto*, si se encuentra en el casillero.
- **recargar** ← El robot comienza el programa de recarga.

Se pide implementar el siguiente predicado, junto con todos los predicados auxiliares necesarios:

recarga(?Baterias, ?Cargador, ?Movs) ←

Baterias y *Cargador* son las coordenadas relativas de las baterías y del cargador respectivamente. *Movs* es la secuencia de comandos que permiten al robot recuperar sus baterías y recargarlas.

Ejercicio 3 [15 puntos]

Sea el siguiente programa Prolog:

```
multi([L|Ls])          :- length([L|Ls],N), N mod 3 =:=0.

tira([a|Xs], Ys)       :- tira(Xs,Ys).
tira([b|Xs], Ys)       :- tira(Xs,Ys,[]).
tira([b|Xs], Ys)       :- tira(Xs,Ys)

tira([X|Xs], Ys, Zs)   :- tira(Xs,Ys,[X|Zs]).
tira([b|_], Ys, Ys)    :- multi(Ys).
```

a) Dé los valores de *Xs* que son solución de la consulta: *? tira([b,b,b,a,a,b,b], Xs)*.

b) Para las siguientes variantes con *cut* del programa anterior, indique los valores de *Xs* que son solución de la consulta de la parte (a). Justifique sus respuestas.

i.

```
multi([L|Ls])          :- length([L|Ls],N), N mod 3 =:=0.

tira([a|Xs],Ys)        :- tira(Xs,Ys).
tira([b|Xs], Ys)       :- !, tira(Xs,Ys,[]).
tira([b|Xs], Ys)       :- tira(Xs,Ys).

tira([X|Xs], Ys, Zs)   :- tira(Xs,Ys,[X|Zs]).
tira([b|_], Ys, Ys)    :- multi(Ys).
```

ii.

```
multi([L|Ls])          :- length([L|Ls],N), N mod 3 =:=0.

tira([a|Xs],Ys)        :- tira(Xs,Ys).
tira([b|Xs], Ys)       :- tira(Xs,Ys,[]), !.
tira([b|Xs], Ys)       :- tira(Xs,Ys).

tira([X|Xs], Ys, Zs)   :- tira(Xs,Ys,[X|Zs]).
tira([b|_], Ys, Ys)    :- multi(Ys).
```

Ejercicio 4 [20 puntos]

a) Implemente el siguiente metaintérprete para Prolog puro:

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

b) El siguiente pseudocódigo representa el esquema básico de un algoritmo de ILP:

Entrada: B, E^+ y E^-
Salida: Una teoría Σ, tal que $\Sigma \cup B$ sea correcta respecto a E^+ y E^-.

Inicializar $\Sigma = \emptyset$.
Repetir
1. Si $\Sigma \cup B$ es fuerte, especializar Σ.
2. Si $\Sigma \cup B$ es débil, generalizar Σ.

hasta que $\Sigma \cup B$ sea correcta respecto a E^+ y E^-.

Describa los elementos de la entrada y de la salida y explique brevemente en qué consisten los dos pasos de la iteración.

Ejercicio 5 [25 puntos]

Sea el siguiente programa lógico:

```
p(f(X),f(X))
p(d, f(e))
p(f(a), d)
p(f(b), c)
p(a, f(b))
q(X, Y)      ← p(f(Z), Y), q(X, Z).
q(X, Y)      ← p(X, f(Y))
```

y los siguientes objetivos: $G_1 \equiv \leftarrow$ **q(X,a)**, $G_2 \equiv \leftarrow$ **q(a,X)** y $G_3 \equiv \leftarrow$ **q(X,X).** Suponga que la regla de computación elige el átomo de más a la izquierda para la resolución.

a) Dibuje los árboles de derivación SLD asociados a los tres objetivos.
b) Sea $\theta_1=\{X/b\}$ y $\theta_3=\varepsilon$. Para cada objetivo, indique si estas son respuestas correctas o computadas. Justifique.
c) Suponga que el intérprete construye el árbol asociando las ramas a las reglas en orden inverso a la aparición de éstas en el programa. Indique los valores para **X** (en el orden correspondiente) que devolvería un intérprete para cada objetivo, según la regla de computación dada y una estrategia de recorrida DFS.
d) ¿Cuál sería su respuesta en el punto anterior si el intérprete construye el árbol asociando las ramas en el orden de aparición de éstas en el programa?
e) ¿Es posible probar la negación del objetivo G_1? ¿Y la negación de G_2?. Justifique.