

Solución Evaluación (2ª instancia)**Duración:** 3 horas**Pregunta sobre el laboratorio**

- Describa brevemente (no más de 10 líneas) el algoritmo de minimax: para qué se usa y cómo funciona.
- ¿Qué heurística utilizó en su implementación del laboratorio para el cálculo del valor de un tablero?

Ejercicio 1 [25 puntos]

Implemente en Prolog los siguientes predicados:

a) `adyacente_valido(+T,+I1,+I2,?J1,?J2)` $\leftarrow$ `(I1,J1)` e `(I2,J2)` son celdas adyacentes y la segunda tiene un valor igual a la primera más 1.

% Valido posiciones adyacentes

`adyacente(I1,J1,I2,J1):- (I2 is I1+1 ; I2 is I1-1).`

`adyacente(I1,J1,I1,J2):- (J2 is J1+1 ; J2 is J1-1).`

% Valido posiciones adyacentes válidas: tienen que ser adyacentes y la segunda ser uno más que la original

`adyacente_valido(T,I1,J1,I2,J2):-`

```
    adyacente(I1,J1,I2,J2),
    member(celda(I1,J1,V1),T),
    member(celda(I2,J2,V2),T),
    V2 is V1 + 1.
```

b) `viaje(+T,+I1,+J1,?I2,?J2,?C)` $\leftarrow$ C es un viaje que sale de la celda `(I1,J1)` y llega a la celda `(I2,J2)`.

% Busco un viaje entre `(I1,J2)` e `(I2,J2)`.

`viaje(T,I1,J1,I2,J2,C):-`

```
    % La primera celda debe ser un uno
    member(celda(I1,J1,1),T),
    % Busco un camino entre la celda origen y la destino
    % Al comienzo solamente tengo visitada la celda inicial
    camino(T,I1,J1,I2,J2,[(I1,J1)],C).
```

% Para cada celda hay un camino de ella a sí misma. No importan los visitados.

`camino(_,I1,J1,I1,J1,_,[celda(I1,J1)]).`

% A partir de una celda, busco los posibles adyacentes, y recursivamente veo si desde allí hay un camino válido

% Tengo que chequear que los adyacentes no hayan sido ya visitados

`camino(T,I1,J1,I2,J2,Visitados, [celda(I1,J1)|C]):-`

```
    adyacente_valido(T,I1,J1,I3,J3),
    \+ member(celda(I3,J3),Visitados),
    camino(T,I3,J3,I2,J2,[celda(I3,J3)|Visitados],C).
```

c) `ciclo(+T,+I1,+J1,?I2,?J2,?C)` $\leftarrow$ C es un ciclo que sale de la celda `(I1,J1)` y llega a la celda `(I2,J2)`.

% Busco un ciclo entre (I1,J2) e (I2,J2).

ciclo(T,I1,J1,I2,J2,C):-

% Primero verifico que haya un viaje

viaje(T,I1,J1,I2,J2,C),

% Chequeo que la primera y la última celda sean adyacentes.

adyacente(I1,J1,I2,J2).

d) maximo_largo_de_viaje(+T,?N) ← N es el largo del viaje o los viajes más largos que aparecen en el tablero T.

% Máximo largo de viaje

% Busco todos los viajes que arrancan de una celda que vale 1

maximo_largo_de_viaje(T,N):-

findall(C,(member(celda(I,J,1),T),viaje(T,I,J,_,_,C)),L1),

largo_maximo(L1,N).

largo_maximo([],0).

largo_maximo([H|T],N1):- length(H,N), largo_maximo(T,N1), N1 > N.

largo_maximo([H|T],N):- length(H,N), largo_maximo(T,N1), N1 <= N.

Ejercicio 2 [20 puntos]

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

i) Una cláusula definida contiene exactamente un literal positivo.

ii) Si una fórmula f es consecuencia lógica de un conjunto de fórmulas S, entonces se cumple que $S \cup \{f\}$ es insatisfactible.

iii) Un árbol SLD contiene todas las respuestas correctas para una consulta.

iv) Todos los nodos de un árbol SLD, salvo la raíz, son cláusulas definidas.

b) Deduzca, utilizando resolución:

Si p y $p \rightarrow q$, entonces q

Si $p \rightarrow q$ y $\neg q$, entonces $\neg p$

Solución

a)

i) Verdadero: Por definición de cláusula definida.

ii) Falso: Lo que se cumple es que $S \cup \{\neg f\}$ es insatisfactible.

iii) Falso: Un árbol SLD contiene todas las respuestas computadas, pero puede haber respuestas correctas que no son computadas.

iv) Falso: Todos los nodos de un árbol SLD son objetivos definidos (no tienen ningún literal positivo).

b)

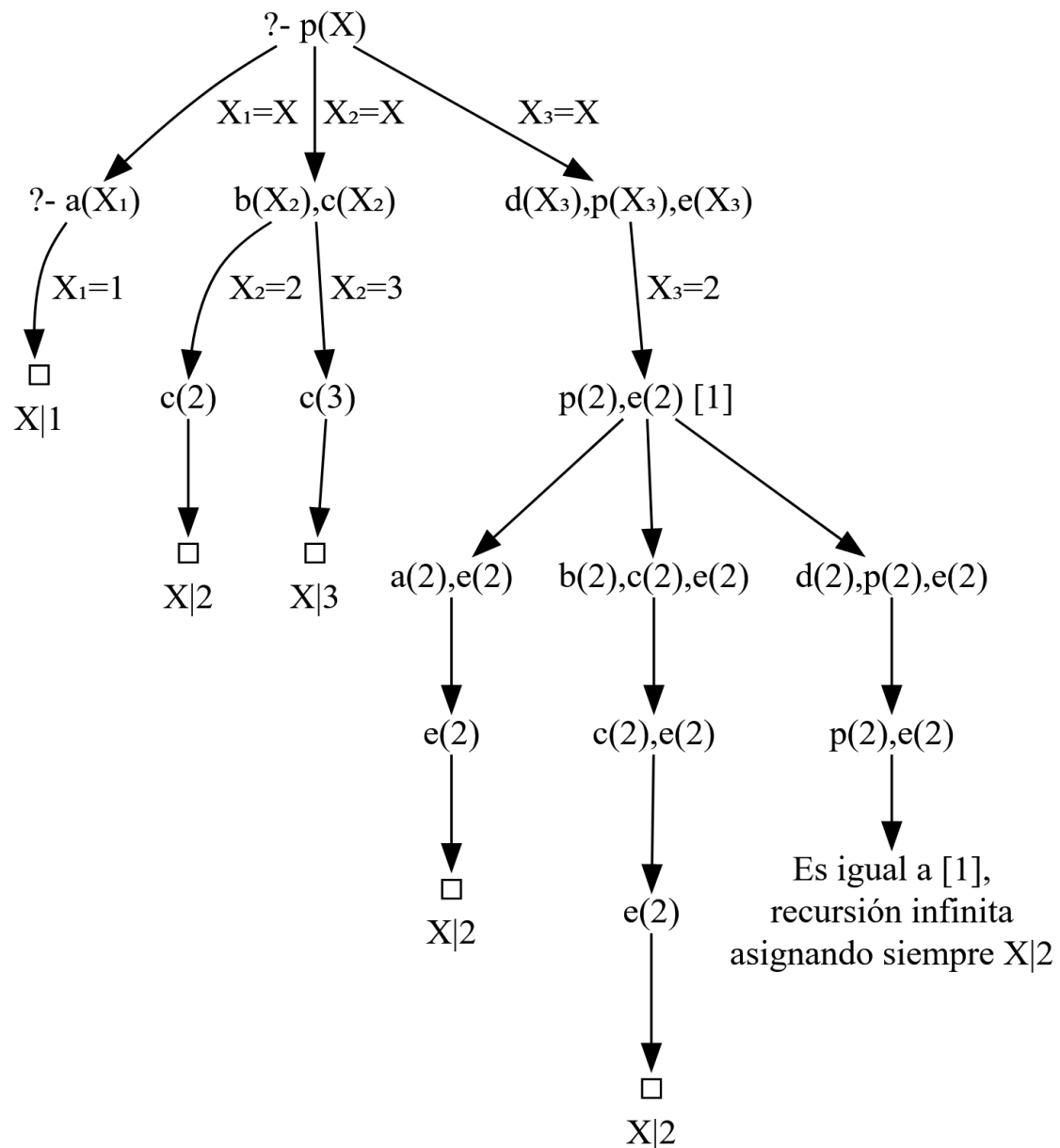
$$\text{Regla de resolución: } \frac{\alpha \vee \beta, \quad \gamma \vee \neg\beta}{\alpha \vee \gamma}$$

$$\begin{array}{lcl} & p, & p \rightarrow q \\ \hline & p \vee \perp, & \neg p \vee q \\ \hline \text{i)} & \perp \vee q & \\ \hline & q & \end{array} \quad \begin{array}{l} \text{equivalencia} \\ \text{resolución} \\ \text{equivalencia} \end{array}$$

$$\begin{array}{lcl} & p \rightarrow q, & \neg q \\ \hline & \neg p \vee q, & \neg q \vee \perp \\ \hline \text{ii)} & \neg p \vee \perp & \\ \hline & \neg p & \end{array} \quad \begin{array}{l} \text{equivalencia} \\ \text{resolución} \\ \text{equivalencia} \end{array}$$

Ejercicio 3 [25 puntos]

- a) Dibuje el árbol SLD correspondiente a la consulta $?- p(X)$ 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?

$X = 1$;
 $X = 2$;
 $X = 3$;
 $X = 2$;
 $X = 2$;
 ... (sigue en recursión infinita devolviendo $X = 2$).

- c) ¿Cuáles son las respuestas si sustituimos la segunda cláusula por la siguiente? Justifique.

$p(X) :- b(X), !, c(X).$

$X = 1$;
 $X = 2.$

(Se poda la rama de la derecha de la cláusula donde aparece el cut, y la tercera rama de la raíz, porque es la cláusula "padre")

- d) ¿Cuáles son las respuestas si sustituimos en el programa original la tercera cláusula por la siguiente? Justifique.

$p(X) :- d(X), !, p(X), e(X).$

Queda igual que en la solución original, porque se entra en recursión y nunca se llega al cut.

- e) ¿Cuáles son las respuestas si sustituimos en el programa original la tercera cláusula por la siguiente? Justifique.

$p(X) :- d(X), p(X), !, e(X).$

$X = 1 ;$
 $X = 2 ;$
 $X = 3 ;$
 $X = 2.$

Se satisface p por primera vez, y no se evalúan soluciones alternativas. Además, se corta la recursión porque la tercera cláusula de p no se evalúa.

- f) ¿Cuáles son las respuestas si sustituimos en el programa original el hecho d(2) por d(1)? Justifique

$X = 1 ;$
 $X = 2 ;$
 $X = 3 ;$

[entra en recursión infinita sin dar soluciones]

Es igual al original, pero como d se satisface con 1, y los otros predicados no, no hay asignaciones para X que devuelvan soluciones.

Ejercicio 4 [15 puntos]

a) Implemente el siguientes predicado sobre listas de diferencias en Prolog:

$\text{largo_ld}(+X, ?N) \leftarrow N$ es el largo de los elementos de la lista de diferencias X , sin contar el resto variable.

Por ejemplo:

```
largo_ld([a,b,c,d|Xr]-Xr,4).
largo_ld([c,d|Yr]-Yr,2).
```

b) Utilizando DCG defina una gramática para el lenguaje sobre el alfabeto $\{0,1\}$ cuyas tiras son de la forma ww , $w \in \{0,1\}^*$

Solución

a)

```
largo_ld(X-Xr,N):-
    largo_ld(X-Xr,0,N).
```

```
largo_ld(X-X,Ac,Ac):-
    var(X),!.
```

```
largo_ld(X-X,_,_):-
    var(X),!,
    false.
```

```
largo_ld([_|X]-Xr,Ac,N):-
    Ac1 is Ac + 1,
    largo_ld(X-Xr,Ac1,N).
```

b)

```
s3 --> w(X),X.
```

```
w([]) --> [].
```

```
w([a|X]) --> [a],w(X).
```

```
w([b|X]) --> [b],w(X).
```