Evaluación (1ª instancia)

Duración: 3 horas

Ejercicio 1 [40 puntos]

Considere un tablero de N x N celdas en donde se intentarán ubicar palabras de modo que el tablero quede completo -todas sus celdas están ocupadas por una letra- y válido -todas las palabras formadas pertenecen a un diccionario-. La figura que sigue muestra un tablero completo y válido de tamaño 3 X 3, se asume que el diccionario incluye las palabras: *los, asa, sol, las, oso, sal*.

l	o	s
a	s	a
s	o	l

Un tablero N X N se representa mediante una lista de N filas. Las filas, a su vez, son listas de largo N.

Las palabras del diccionario están representadas por medio del predicado `palabra\1` que tiene como argumento una lista de letras:

```
palabra([l,o,s]).
palabra([s,o,l]).
palabra([a,r,e,n,a]).
palabra([p,l,a,n]).
...
```

Implemente en Prolog los predicados que se describen a continuación:

a) `pal_valida_en_fila(+T)<-`
T tiene al menos una palabra válida en alguna de sus filas.

b) `col_j(?T, +J, -Col) <-`
Col es la columna j-ésima del tablero T (representado como lista de filas).

c) `tablero_completo_valido(?T, +N) <-`
El tablero T (representado como lista de filas), de dimensión N X N, es completo y válido según lo definido anteriormente.

d) `todos_tableros_sin_rep(?T, +N, -TT) <-`
TT es la lista de todos los tableros de dimensión N X N completos y válidos, compatibles con el tablero T, que no contienen palabras repetidas.

Observación importante
En las partes b), c) y d) se asume que:
El argumento T es una lista que contiene N listas de largo N.
El argumento T puede estar:

- `completamente instanciado: [[l, o, s],[a, s, a],[s, o, l]]`
- `completamente sin instanciar: [[A, B, C],[D, E, F],[G, H, I]]`
- `semi-instanciado: [[A, B, s],[o, E, F],[G, H, I]]`

Soluciones

a)

```
% pal_valida_en_fila([F|Fs])  --- hay una palabra válida en alguna fila

pal_valida_en_fila([F|_]) :-
   palabra(F),
   !.
pal_valida_en_fila([_|Fs]) :-
   pal_valida_en_fila(Fs).
```

b)

```
% col_j(T, J, Col) --- col es la columna j-esima del tablero T

col_j([], _J, []).
col_j([F|Fs], J, [Elem|Cs]) :-
   elem_j(J, F, Elem),
   col_j(Fs, J, Cs).
elem_j(1, [E|_], E) :- !.
elem_j(J, [_|R], E) :-
   J1 is J-1,
   elem_j(J1, R, E).
```

c)

```
% tablero_completo_valido(?T) --- T es un tablero completo y válido

tablero_completo_valido(T,N) :-
   tablero_valido_fils(T),
   tablero_valido_cols(T,N).

tablero_valido_fils([]).
tablero_valido_fils([F|Fs]) :-
   palabra(F),
   tablero_valido_fils(Fs).

tablero_valido_cols(_,0) :- !.
tablero_valido_cols(T,J) :-
   col_j(T,J,C),
   palabra(C),
   J1 is J-1,
   tablero_valido_cols(T,J1).
```

d)

```
% todos_tableros_sin_rep(?T,+N,-TT) --- TT es la lista de los tableros T válidos sin
palabras repetidas

todos_tableros_sin_rep(T,N,TT) :-
   findall(T,(tablero_valido(T,N),sin_rep_tab(T,N)),TT).

sin_rep_tab(T,N) :-
   sin_rep_tab_fil(T),
   sin_rep_tab_col(T,[],N).

sin_rep_tab_fil([]).
sin_rep_tab_fil([F|Fs]) :-
   \+ member(F,Fs),
   sin_rep_tab_fil(Fs).

sin_rep_tab_col(_,_,0) :- !.
sin_rep_tab_col(T, Pals, N) :-
   col_j(T,N,C),
   \+ member(C,Pals),  % chequeo no repetidos entre columnas
   \+ member(C,T),     % chequeo no repetidas entre filas y columnas
   N1 is N-1,
   sin_rep_tab_col(T,[C|Pals], N1).
```

Ejercicio 2 [5 puntos]

Indique si las siguientes afirmaciones son verdaderas o falsas:

Soluciones

a) Si una fórmula **f** es consecuencia lógica de un conjunto de fórmulas **S**, entonces se cumple que **S U {f}** es insatisfactible.
Falso: En ese caso, lo que se cumple es: **S U {not f}** es insatisfactible

b) Un objetivo definido no puede contener ningún literal positivo.
Verdadero: Por definición, los objetivos definidos tienen todos sus literales negativos.

c) Las cláusulas de Horn contienen al menos un literal positivo.
Falso: Las cláusulas de Horn incluyen cláusulas definidas y objetivos definidos, y estos últimos no tienen ningún literal positivo.

d) Un m.g.u. es una sustitución con un solo elemento.
Falso: No existe ninguna restricción sobre la cantidad de elementos de un m.g.u.

e) Un árbol SLD contiene todas las respuestas correctas para una consulta.
Falso: El árbol SLD contiene todas las respuestas **computadas** para una consulta.

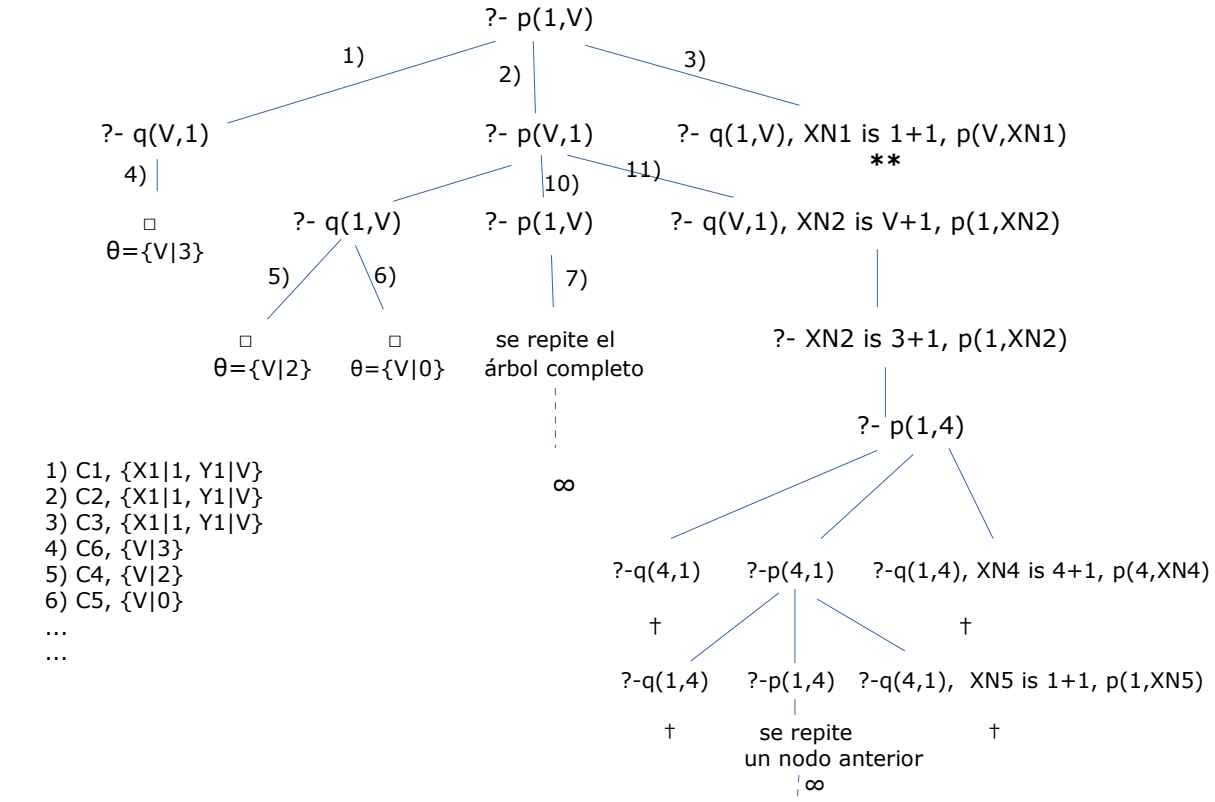
Ejercicio 3 [30 puntos]

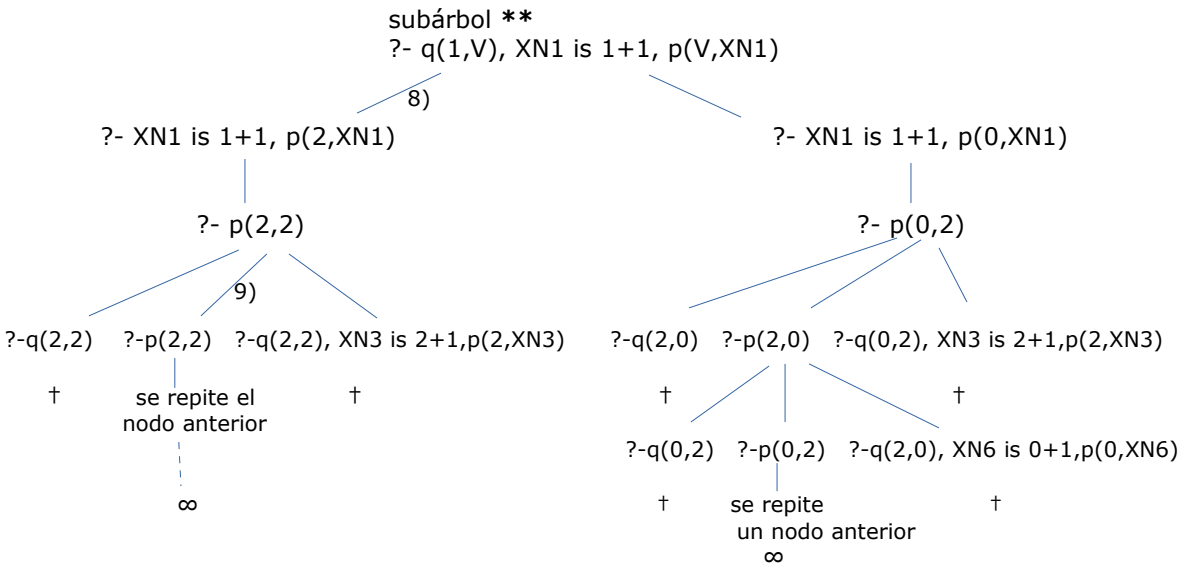
Dado el siguiente conjunto de cláusulas para los predicados **p** y **q**:

```
C1: p(X,Y) :- q(Y,X).
C2: p(X,Y) :- p(Y,X).
C3: p(X,Y) :- q(X,Y), XN is X+1, p(Y,XN).
C4: q(1,2).
C5: q(1,0).
C6: q(3,1).
```

a) Dibuje el árbol SLD correspondiente a la consulta **?- p(1,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.

Solución





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?

Solución

Un intérprete como el que se describe dará como primera repsuesta V=3, luego si se le pide otra respuesta dará V=2 y luego V=0. Si se siguen pidiendo repuestas repetirá estas tres infinitamente, ya que el subárbol infinito de la rama 7) llega a esas tres respuestas infinitas veces.

c) Diga cuáles serían las respuestas de la parte b) si las cláusulas de **p** estuvieran en el siguiente orden: `C3, C2, C1.` Justifique su respuesta.

Solución

En este caso, el intérprete entraría en primer lugar en la rama 3, luego en la 8 y en la 9, de la cual no saldría nunca (o se quedaría sin stack) ya que es infinitay no tiene respuestas para dar.

d) ¿Qué respuestas daría un intérprete prolog como el de la parte b) si se eliminara la clásula C2? Justifique su respuesta.

Solución

Daría la respuesta V=3, correspondiente a la rama 1. Si se le pidieran más respuestas daría false, ya que el resto del árbol sería finitamente fallado (las ramas infinitas eran generadas por C2, que ya no está).

e) Considere las siguientes variantes para la cláusula C1:

i. `C1: p(X,Y) :- !, q(Y,X).`

ii. `C1: p(X,Y) :- q(Y,X), !.`

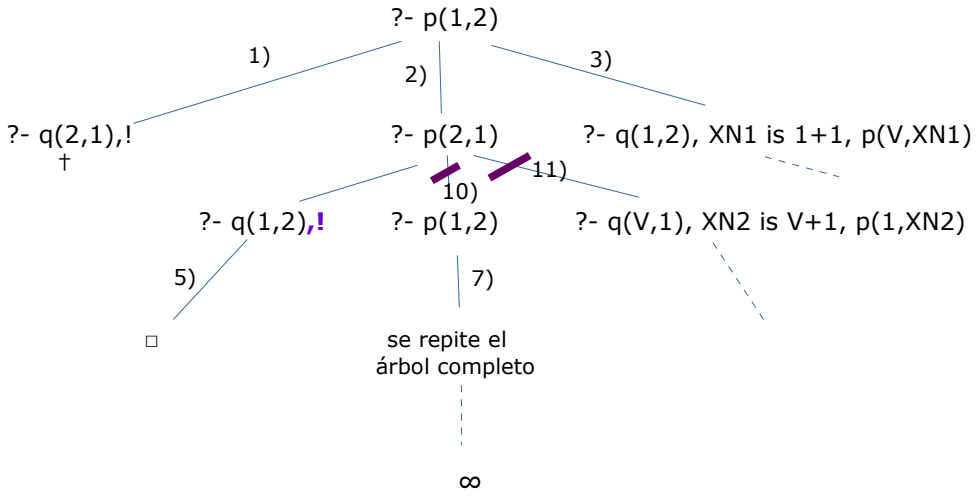
¿Qué respuestas daría un intérprete prolog como el de la parte b) para la consulta **?- p(1,2)**, para cada una de las variantes de C1?

Solución

i.

El cut de esta variante corta las ramas correspondientes a C2 y C3 del árbol SLD de la consulta ?- p(1,2) (que es parecido al árbol de la parte a). La rama correspondiente a C1, que es la única, falla, ya que no puede satisfacerse el objetivo q(2,1). Por lo tanto la consulta da false.

ii.



El intérprete da true, al probar q(1,2), entrando en la rama 5, y luego el cut resaltado en violeta poda las ramas 10 y 11. Si se pide otra respuesta, entra en una rama infinita al intentar buscar respuestas por la rama 3.

Ejercicio 4 [10 puntos]

Escribir el siguiente predicado:

`cumple_nocumple(+Prop, +L, ?L1, ?L2) <-`
La lista L1 contiene todos los elementos de L que cumplen la propiedad Prop y la lista L2 contiene todos los elementos de L que no la cumplen.

Observaciones

- No se debe probar dos veces la propiedad `Prop` p ara un mismo elemento de la lista.
- `Prop` es el nombre de algún predicado que se asume definido previamente. Por ejemplo, si se cuenta con el predicado `par\1`, que verifica si un número es par, la siguiente consulta es verdadera:
  `?- cumple_nocumple(par, [2,3,5,6,1,11,10], [2,6,10], [3,5,1,11]).`

Solución

```
cumple_nocumple(_,[],[],[]).
cumple_nocumple(Prop,[E|L],[E|L1],L2) :-
   Predicado =.. [Prop,E],
   Predicado,
   !,
   cumple_nocumple(Prop,L,L1,L2).
cumple_nocumple(Prop,[E|L],L1,[E|L2]) :-
   cumple_nocumple(Prop,L,L1,L2).
```