Evaluación (1ª instancia)

Duración: 3 horas

Solución

Ejercicio 1 [40 puntos]

Dado un diccionario de palabras representadas como listas de letras, escriba los siguientes predicados, orientados a simplificar la escritura de textos, proponiendo predicciones y correcciones.

a) `prediccion(?Texto, ?PalDic)<-` PalDic es una palabra del diccionario con prefijo Texto.

b) `distanciaSimplificada(+Pal1, +Pal2, ?DistSimp)<-` La distancia simplificada se calcula como sigue: se asume que la dos palabras tienen el mismo largo y se cuenta la cantidad de caracteres diferentes posición a posición (ver ejemplos más abajo).

c) `correccionFrase(+Frase, ?FraseCorrecta, ?DistTotal)<-` FraseCorrecta es una frase formada por palabras del diccionario y DistTotal es la suma de las distancias simplificadas entre cada palabra de Frase y su correspondiente en FraseCorrecta. Falla si alguna palabra no tiene el mismo largo que su correspondiente.

d) `correccionesFrase(+Frase, -Correcciones)<-` Correcciones es una lista de frases que corresponden a correcciones de Frase, ordenada por distancias.

Ejemplos:

```
diccionario([c,a,s,a]).
diccionario([c,a,o,s]).
diccionario([c,a,s,a,m,i,e,n,t,o]).
diccionario([d,e]).
diccionario([p,a,p,e,l]).
diccionario([t,r,a,p,o]).
diccionario([a,y,e,r]).

prediccion([c,a],[c,a,s,a]).
prediccion([c,a],[c,a,o,s]).
prediccion([c,a],[c,a,s,a,m,i,e,n,t,o]).

distanciaSimplificada([c,a,s,a],[c,a,s,a],0).
distanciaSimplificada([c,a,s,a],[c,a,o,s],2).
distanciaSimplificada([c,a,s,a],[a,y,e,r],4).
distanciaSimplificada([c,a,s,a],[c,a,s,a,d,a],D) falla

correccionFrase([[c,a,d,a],[d,e],[p,a,o,e,l]],
      [[c,a,s,a],[d,e],[p,a,p,e,l]], 2).
correccionFrase([[c,a,d,a],[d,e],[p,a,o,e,l]],
      [[c,a,o,s],[d,e],[p,a,p,e,l]], 4).

correccionesFrase([[c,a,d,a],[d,e],[p,a,o,e,l]], Corr).
Corr = [ 2-[[c,a,s,a],[d,e],[p,a,p,e,l]],3-[[c,a,o,s],[d,e],[p,a,p,e,l]], 4-
[[c,a,s,a],[d,e],[t,r,a,p,o]]], 5-[[c,a,o,s],[d,e],[t,r,a,p,o]],
5-[[a,y,e,r],[d,e],[p,a,p,e,l]],7-[[a,y,e,r],[d,e],[t,r,a,p,o]] ]).
```

Solución

a)

```
prediccion(Pre,Pal) :-
   append(Pre,_,Pal),
   diccionario(Pal).
```

b)

```
distanciaSimplificada(P1,P2,D) :- distanciaSimplificada(P1,P2,0,D).
distanciaSimplificada([],[],D,D).
%letras iguales, mantengo el acumulador como está
distanciaSimplificada([L|R1],[L|R2],Sum,D) :-
   !,
   distanciaSimplificada(R1,R2,Sum,D).
%letras diferentes, incremento el acumulador
distanciaSimplificada([_|R1],[_|R2],Sum,D) :-
   Sum1 is Sum + 1,
   distanciaSimplificada(R1,R2,Sum1,D).
```

c)

```
correccionFrase(Frase,FraseCorrecta,Dist) :-
   correccionFrase(Frase,FraseCorrecta,0,Dist).
correccionFrase([],[],Ac,Ac).
correccionFrase([P|Frase],[PC|FraseCorrecta],Ac,Dist) :-
   correccionPal(P,PC,D1),
   Ac1 is Ac + D1,
   correccionFrase(Frase,FraseCorrecta,Ac1,Dist).

correccionPal(P1,P2,D):-
   diccionario(P2),
   distanciaSimple(P1,P2,D).
```

d)

```
correccionesFrase(Frase,Corrs) :-
   setof(D-FC,correccionFrase(Frase,FC,D),Corrs).
```

Ejercicio 2 [10 puntos]

a) Sean C1 y C2 dos cláusulas de primer orden:
C1 : {L1, . . . , Ln}
C2 : {M1, . . . , Mk}
y θ un mgu de Li y ¬Mj
Escriba la resolvente binaria de C1 y C2 y diga qué restricción deben cumplir las variables de las cláusulas.

b) Para cada afirmación, diga si es verdadera o falsa.
En un árbol SLD:

i. La raíz siempre es un objetivo definido.
ii. Todos los nodos del árbol, salvo la raíz, son cláusulas definidas.
iii. Cada nodo distinto de la raíz es una resolvente entre un objetivo definido y una cláusula definida.
iv. Todas las hojas de las ramas finitas del árbol corresponden a la cláusula vacía.

c) Aplique el algoritmo para hallar el m.g.u. de dos expresiones a las expresiones siguientes: `p(y,f(a,g(x)))` y `p(f(w),f(z,w))`. Muestre paso a paso la aplicación del algoritmo.

Solución

a)
La resolvente es: C : {L1, . Li-1,Li+1, . . , Ln, M1, ..., Mj-1, Mj+1, ..., Mk}θ

Las cláusulas C1 y C2 no pueden compartir variables (lo cual se resuelve con un renombre de variables).

b)

i. V
ii. F
iii. V
iv. F

c)

```
inicialización:
       stack =      p(y,f(a,g(x))) = p(f(w),f(z,w))
       mgu = {}

1er paso de iteración:
       pop p(y,f(a,g(x))) = p(f(w),f(z,w))
       estoy en caso 4
       mgu = {}
       stack =      f(a,g(x))    =     f(z,w)
                    y            =     f(w)

2º paso de iteración:
       pop f(a,g(x)) = f(z,w)
       estoy en caso 4
       mgu = {}
       stack =      g(x)   =     w
                    a      =     z
                    y      =     f(w)

3er paso de iteración:
       pop g(x) = w
       estoy en caso 2
       mgu = {w|g(x)}
       stack =      a      =     z
                    y      =     f(g(x))

4º paso de iteración:
       pop a = z
       estoy en caso 2
       mgu = {w|g(x)} o {z|a} = {w|g(x), z|a}
       stack =      y      =     f(g(x))

5º paso de iteracíon:
       pop y = f(g(x))
       estoy en caso 1
       mgu = {w|g(x), z|a} o {y|f(g(x))} = {w|g(x), z|a, y|f(g(x))}
       stack =      vacío

termino

salida: mgu = {w|g(x), z|a, y|f(g(x))}
```

Ejercicio 3 [25 puntos]

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

```
C1: p(X) :- q(X,X).
C2: p(X) :- q(X,Y), p(Y).
C3: p(X) :- q(Y,X), q(Y,Y).
C4: q(0,0).
C5: q(0,1).
```

a) Dibuje el árbol SLD correspondiente a la consulta **`?- p(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: `C2, C1, C3`. Justifique su respuesta.

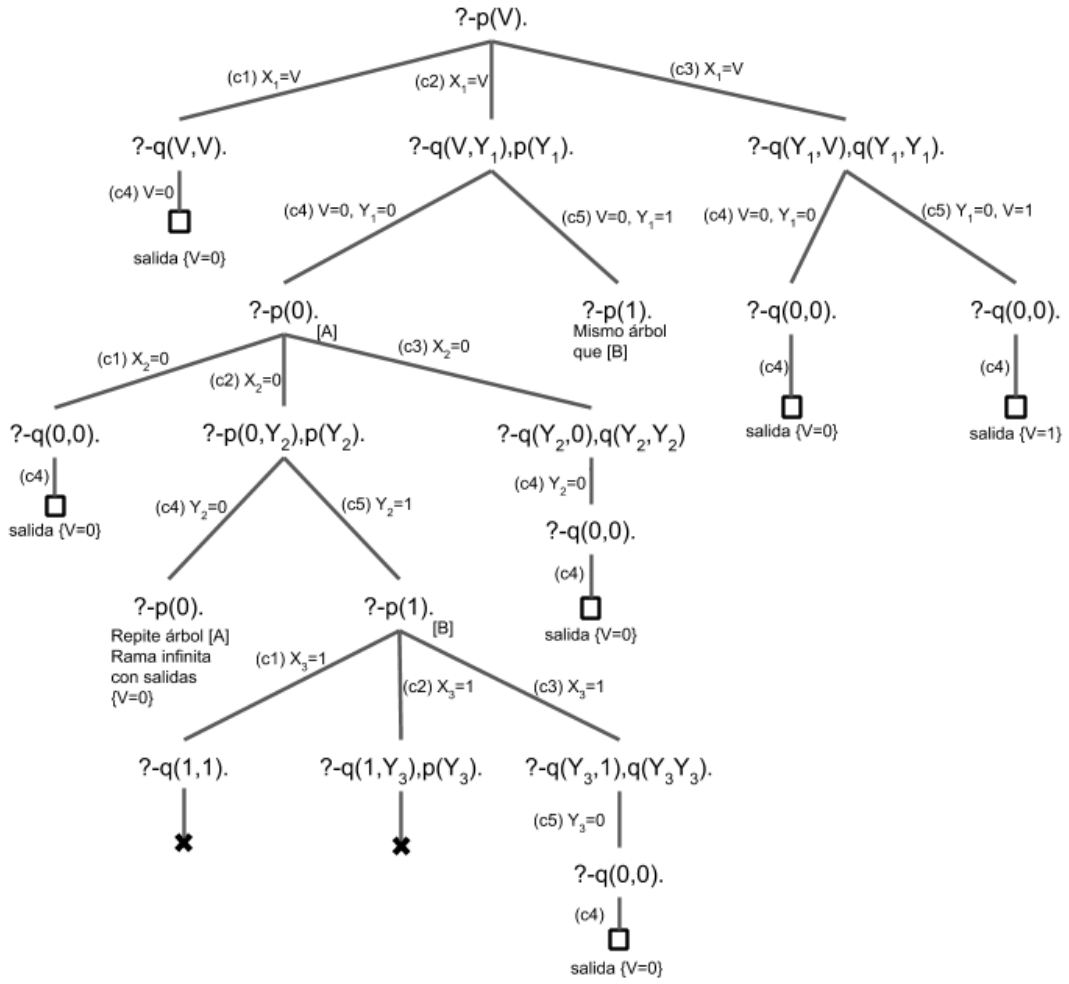
d) Considere las siguientes variantes para la cláusula C1:

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

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

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

Solución
a)



b) Las salidas son:
{V=0}
{V=0}
Entra por la rama infinita del árbol [A], devolviendo infinitas veces {V=0}

c) Entraría directamente por la rama infinita y no llegaría a devolver ninguna salida.

d)
i. Devuelve *false*: al ejecutar la rama C1 poda las ramas C2 y C3, pero C1 no puede satisfacerse.
ii. Devuelve *true*: como la rama C1 no puede satisfacerse, no llega a ejecutar el cut y por lo tanto no poda las ramas C2 y C3. C3 puede satisfacerse.

Ejercicio 4 [10 puntos]

a) Dé una solución alternativa para el predicado `prediccion(?Texto, ?PalDic)` del ejercicio 1, asumiendo que Texto es una lista incompleta. Ejemplo (para el predicado `diccionario` del ejercicio 1):

```
prediccion([c,a|_], Pal).
Pal = [c,a,s,a];
Pal = [c,a,o,s];
Pal = [c,a,s,a,m,i,e,n,t,o]
```

b) Escriba el siguiente predicado:

```
esLI(L)  <-  L es una lista incompleta.
esLI([a,b,c|_]) da true.
esLI(L) da true.
esLI([a,b,c]) falla.
esLI([]) falla.
```

Solución

a)

```
prediccion(Texto,Texto):-
   diccionario(Texto).
```

b)

```
esLI(L) :-
   var(L),
   !.
esLI([_|L]) :-
   esLI(L).
```