Departamento de Cs. e Ingeniería de la Computación
Universidad Nacional del Sur

Lógica para Ciencias de la Computación

Trabajo Práctico "D"

Semántica Operacional

Primer Cuatrimestre de 2008

Observación importante: Las consultas relacionadas con los temas desarrollados en los trabajos prácticos serán únicamente respondidas durante las propias clases prácticas a lo largo del cursado.

Ejercicios

1. Encontrar una sustitución unificadora para cada uno de los siguientes pares de fórmulas.

   a) hermano(X, Y) y hermano(beto, dario).

   b) suma(0, X, X) y suma(X, s(0), Z).

   c) nn(s(s(0))) y nn(s(X)).

   d) fbf(a -> (b or c)) y fbf(X -> Y).

   e) eslista([1,2,3]) y eslista([X|Xs]).

2. Encontrar el unificador más general para cada uno de los siguientes pares de fórmulas atómicas, fundamentando la validez del resultado obtenido.

   a) append([b],[c,d],L) y append([X|Xs],Ys,[X|Zs]).

   b) suma(X,s(s(Y)),s(X)) y suma(s(0),s(s(s(s(Z)))),Z).

   c) btree(tree(X,tree(X,tree(Y,void,Z),Z),tree(c,Z,void))) y
   btree(tree(a,X,tree(Y,tree(b,tree(Y,W,void),void),void))).

   d) sort([X,c,[Z]|[[a],Z]],W) y sort([[3,4,L|Y],M,N,L,3],[3,5|N]).

   e) hanoi(s(N),A,B,C,Ms) y hanoi(s(s(0)),a,b,c,Xs).

3. Considerando el siguiente programa lógico,

```
q(a).                    p(X) :- q(X), r(X), s(Y).
q(b).                    r(X) :- s(X).
s(b).

nn(0).
nn(s(X)) :- nn(X).
```

   obtener los árboles SLD asociados a las consultas indicadas a continuación, señalando claramente las sustituciones que justifican las respuestas obtenidas.

   a) ?- r(a).

   b) ?- r(X).

c) ?- p(X).

d) ?- nn(s(0)).

e) ?- nn(X).

Considerando el siguiente programa lógico,

```
concat([], Xs, Xs).
concat([X|Xs], Ys, [X|Zs]) :- concat(Xs, Ys, Zs).

sublist([],Xs).
sublist([X|Xs], [X|Ys]) :- sublist(Xs, Ys).
sublist([X|Xs], [Y|Ys]) :- X \= Y, sublist([X|Xs], Ys).

sort([],[]).
sort([X|Xs], Zs) :- sort(Xs, Ys), insert(X, Ys, Zs).
insert(X, [], [X]).
insert(X, [Y|Ys], [X,Y|Ys]) :- X =< Y.
insert(X, [Y|Ys], [Y|Zs]) :- X > Y, insert(X, Ys, Zs).
```

obtener los árboles SLD asociados a las consultas indicadas a continuación, señalando claramente las sustituciones que justifican las respuestas obtenidas.

a) ?- concat([1,2], [3,4], C).

b) ?- concat([1,2,3], B, C).

c) ?- sublist([a,b,c], [a,b,d,a,b,c,d|X]).

d) ?- sublist([a,b,c], [a,b,d,e]).

e) ?- sort([3,1,2], B).

4. ¿En qué consiste el *occurs check*? ¿Por que razón esta comprobación es omitida por la mayoría de las implementaciones de Prolog?

5. Teniendo en cuenta que la mayoría de la implementaciones de Prolog no comprueban el occurs check, al interpretar a un programa prolog como teoría formal, ¿esta omisión hace que presente un comportamiento no sensato, o más bien un comportamiento no completo? Fundamentar la respuesta suministrada a través de un ejemplo en concreto.

6. En base al siguiente programa lógico,

```
letra(a).   letra(b).   letra(c).   letra(d).

lista([]).
lista([X|Xs]) :- letra(X), lista(Xs).

append([], Xs, Xs) :- lista(Xs).
append([X|Xs], Ys, [X|Zs]) :- letra(X), append(Xs, Ys, Zs).
```

generar los árboles SLD para las siguientes consultas:

a) ?- append([], tree(a, void, void), L3).

b) ?- append(L1, L2, [a,B,c,D]).

c) ?- append([a,A], [A|B], [a,C,D,D]).