# Prolog, Alberi SLD e SLDNF

- *Scopo:*
  1. Semplici esercizi sul Prolog
  2. Esercizi su alberi di risoluzione SLD e SLDNF

# Esercizio 1 - Prolog

Si definisca un predicato in PROLOG chiamato averStud che applicato a un numero di matricola di uno studente Matr e a una lista di esami LE dia come risultato la media AV dei suoi voti. Ogni esame sia rappresentato da un termine della lista LE della forma asame(Matr,Esame,Voto).

Si definisca prima la versione ricorsiva e poi quella ricorsiva-tail.

Esempio:

?- averStud(s1,[esame(s2,f1,30), esame(s1,f1,27),esame(s3,f1,25), esame(s1,f2,30)], AV).

*yes, AV = 28.5*

# Esercizio 1 – Prolog - Soluzione

```
% versione ricorsiva
averStud(S,L,AV) :-
    totStud(S,L,N,T),
    N > 0,
    AV is T/N.
totStud(_,[],0,0) :- !.
totStud(S,[esame(S,_,V)|R],N,T) :- !,
    totStud(S,R,NN,TT),
    N is NN + 1, T is TT + V.
totStud(S,[_|R],N,T) :-
    totStud(S,R,N,T).
```

# Esercizio 1 – Prolog - Soluzione

```
% versione tail-ricorsiva
averStud(S,L,AV) :-
    totStud(S,L,0,N,0,T),
    N > 0,
    AV is T/N.
totStud(_,[],N,N,T,T) :- !.
totStud(S,[esame(S,_,V)|R],NI,NO,TI,TO):- !,
    N is NI + 1, T is TI + V,
    totStud(S,R,N,NO,T,TO).
totStud(S,[_|R],NI,NO,TI,TO) :-
    totStud(S,R,NI,NO,TI,TO).
```

# Esercizio 2 - Prolog

Un associazione di volontariato tiene traccia delle spese di cancelleria tramite un insieme di fatti Prolog del tipo:

spesa(carta, 13).

spesa(matite, 12).

spesa(graffette, 5.99).

spesa(carta, 12).

Dove il primo parametro rappresenta l'oggetto acquistato, ed il secondo l'importo.

# Esercizio 2 - Prolog

Definire i seguenti predicati Prolog:

elenca(L), che deve restituire la lista degli item acquistati, senza ripetizione. Si usi a tal scopo il predicato setof.

subTotale(X, S), che ricevuto in ingresso un oggetto X, restituisce l'ammontare della spesa per tale oggetto

totale(S), che restituisce quanto è stato speso in cancelleria.

Ad esempio, i predicati (invocati sulla base di conoscenza presentata all'inizio di questo esercizio) devono restituire:

```
| ?- elenca(L).
L = [carta,graffette,matite] ?

| ?- subtotale(carta, S).
S = 25 ?

| ?- totale(S).
S = 42.99 ?
```

# Esercizio 2 – Prolog - Soluzione

Se si usa il predicato setof è scorretto a meno di utilizzare la quantificazione sulla variabile Y.

```
elenca(L) :-
   setof( X, Y^spesa(X, Y), L).
```

Oppure :

```
elenca1(L1) :-
    findall( X, spesa(X, _), L),  eliminarip(L,L1).
eliminarip([],[]):-!.
eliminarip([A|Rest1], Rest3):-
    member(A,Rest1),!,eliminarip(Rest1,Rest3).
eliminarip([A|Rest1], [A|Rest3]):-
    eliminarip(Rest1,Rest3).
```

# Esercizio 2 – Prolog - Soluzione

```
subtotale(X, S) :-
    bagof(V, spesa(X, V), L),
    somma(L, S).

totale(S) :-
    findall(V, spesa(_, V), L),
    somma(L, S).

somma([], 0).
somma([H|T], S) :-
    somma(T, S1),
    S is S1 + H.
```

# Esercizio 3 - Prolog

Si scriva un programma Prolog che, prendendo in ingresso due liste L1 e L2, restituisca in uscita due liste L3 e L4 tali che L3 contenga gli elementi di L1 che appartengono anche a L2, mentre L4 contenga gli elementi di L1 che non appartengono a L2. Si supponga disponibile il predicato member. Si dica inoltre se il predicato così definito è ricorsivo tail.

Esempio:

?- list_mem([a,r,t],[t,s,m,n,a],L3,L4).

restituirà L3=[a,t] e L4=[r].

# Esercizio 3 – Prolog - Soluzione

```
list_m([],L2,[],[]).
list_m([A|Rest1],L2,[A|Rest3],L4):-
    member(A,L2),!,list_m(Rest1,L2,Rest3,L4).
list_m([A|Rest1],L2,L3,[A|Rest4]):-
    list_m(Rest1,L2,L3,Rest4).
```

Il predicato è ricorsivo tail.

# Esercizio 4 - Prolog

Si scriva un predicato Prolog list_to_set a due argomenti che data una lista di liste come primo argomento leghi il secondo argomento a una lista nella quale sono state eliminate le liste ripetute o le loro permutazioni.

Per esempio dato il goal:

?-list_to_set([[1,2,3],[3,1,2],[1]], Y).

si vuole ottenere:

yes Y=[[3,1,2],[1]]

Per esempio dato il goal:

?-list_to_set([[1,2],[1,2],[1,2]], Y).

si vuole ottenere:

yes Y=[[1,2]]

Si supponga dato il predicato permutation(X,Y) che verifica se una lista X è una permutazione della lista Y.

# Esercizio 4 – Prolog - Soluzione

```
list_to_set([],[]) :-  !.
list_to_set([I|R],[I|R1]) :-
    not member_list(I,R),
    !,
    list_to_set(R,R1).
list_to_set([I|R],R1) :-
    list_to_set(R,R1).

member_list(X,[Y|_]) :- permutation(X,Y),  !.
member_list(X,[_|R]) :- member_list(X,R).
```

# Esercizio 5 - Prolog

Si scriva un programma Prolog che data in ingresso una lista di liste con 2 elementi ciascuna ed una costante c1 restituisca in uscita due liste DX ed SX, la prima contenente gli elementi che nelle coppie compaiono a destra di c1, la seconda a sinistra.

**Soluzione:**

```
coppie([],_,[],[]).
coppie([[X,X]|T],X,[X|Td],[X|Ts]) :- !,
    coppie(T,X,Td,Ts).
coppie([[X,Y]|T],X,[Y|Td],Ts) :- !,
    coppie(T,X,Td,Ts).
coppie([[Y,X]|T],X,Td,[Y|Ts]) :- !,
    coppie(T,X,Td,Ts).
coppie([_|T],X,Td,Ts) :-
    coppie(T,X,Td,Ts).
```

# Esercizi vari - Prolog

1. Scrivere un predicato Prolog che fornisce l'ultimo elemento di una lista.
2. Dare un programma in Prolog che definisca la relazione tra due liste di avere l'una lunghezza doppia dell'altra.
3. Scrivere un predicato Prolog per verificare se una lista è palindroma
4. Dato un albero binario, si scriva un predicato che calcola la profondità massima dell'albero.
5. Dare un programma in Prolog che definisca la relazione nodiInterni tra un albero binario e un naturale, tale che il numero naturale indichi il numero di nodi interni (non foglie) dell'albero
6. Dati due alberi A1 ed A2 si scriva un predicato che verifica se A1 è un sottoalbero di A2.
7. Una matrice $M$ di bit $n \times m$, si dice matrice C-Grey se, $\forall j$=1,…,$m$-1, la $j$-esima colonna della matrice differisce dalla colonna ($j$+1)-esima esattamente in un bit. Si realizzi un programma Prolog che, data $M$ rappresentata come lista di colonne, risponda sì se $M$ è C-Grey.

# Esercizio 6 - SLD

Si consideri il seguente programma Prolog:

```
superclasse(X,Y):- classe(X,Y).
superclasse(X,Z):- classe(W,Z),
                   superclasse(X,W).
```

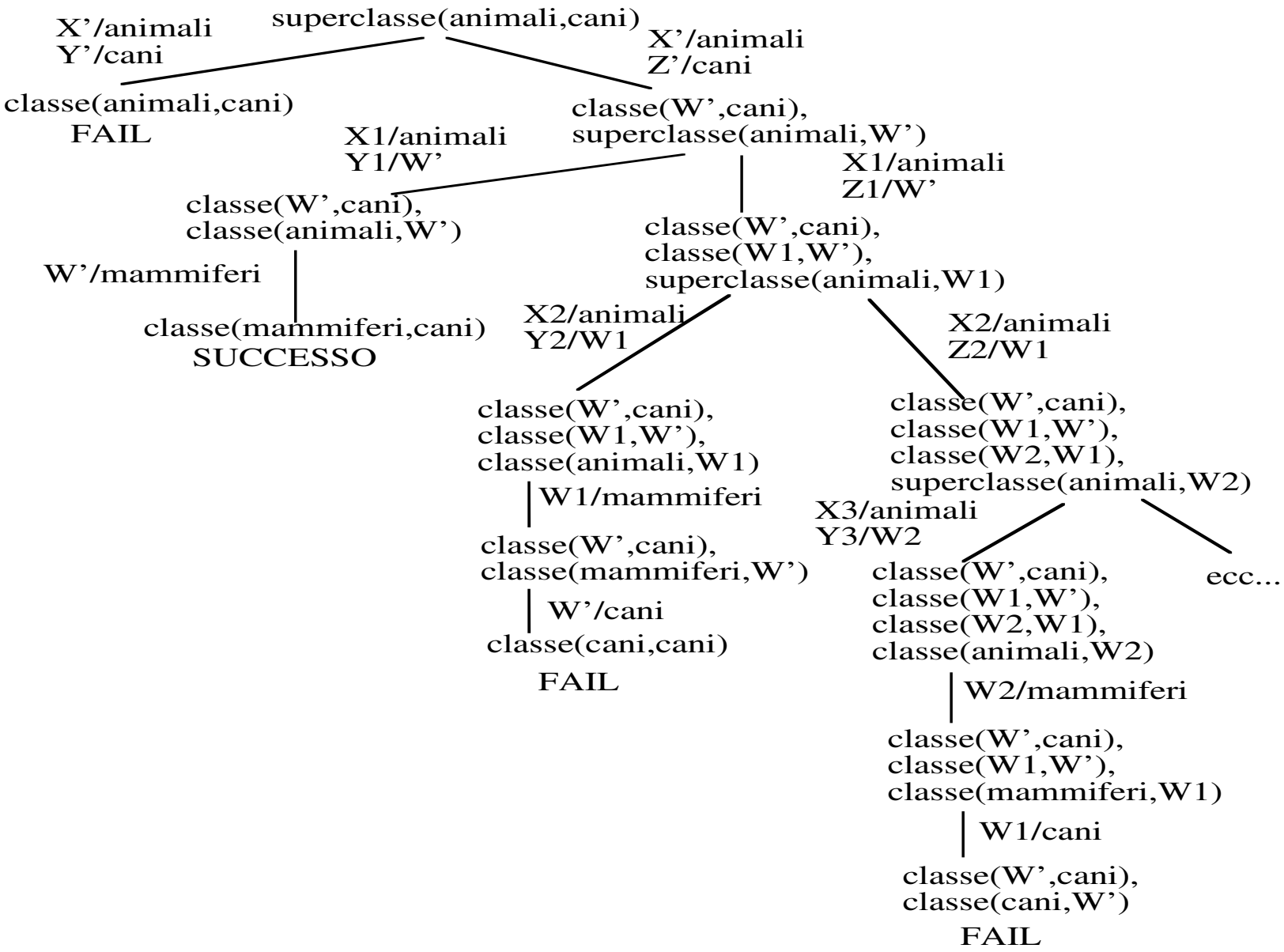
Si rappresenti l'albero SLD con regola di selezione right-most relativo al goal:

```
:- superclasse(animali, cani)
```

assumendo la presenza, all'inizio del database, dei fatti:

```
classe(mammiferi,cani).
classe(animali,mammiferi).
```

# Soluzione Esercizio 6 - SLD



# Esercizio 7 – SLD & cut

Si consideri il seguente programma Prolog:

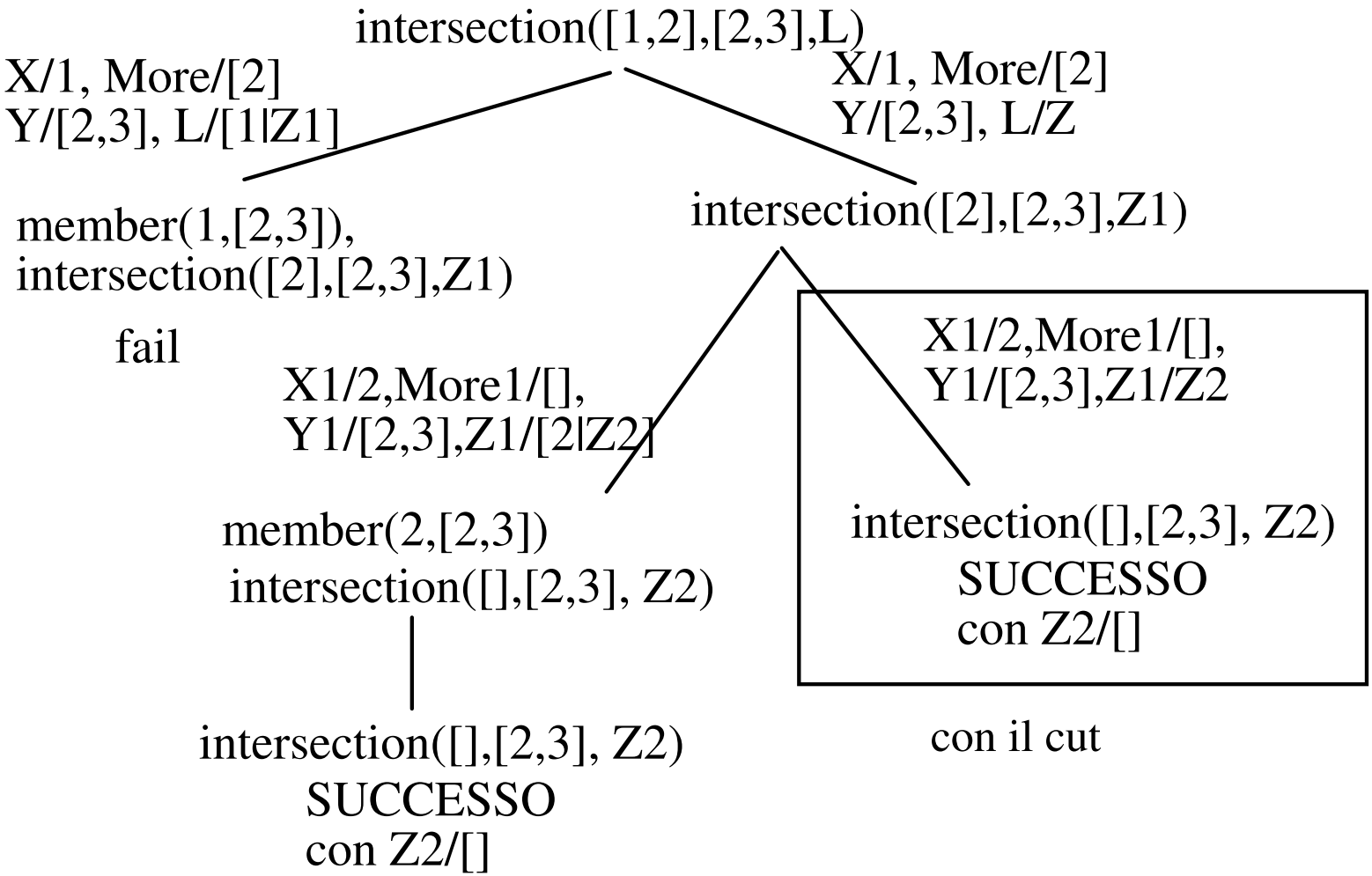
```
intersection([],Y,[]).
intersection([X|More],Y,[X|Z]):-
   member(X,Y),
   intersection(More,Y,Z).
intersection([X|More],Y,Z):-  intersection(More,Y,Z).
```

Si rappresenti l'albero SLD relativo al goal

```
:- intersection([1,2],[2,3],L).
```

e si indichino i rami di successo. Si indichi come l'utilizzo del cut (!) possa portare alla definizione corretta del predicato intersezione.

# Soluzione Esercizio 7 – SLD & cut



# Esercizio 8 – SLDNF

Si consideri il seguente programma Prolog:

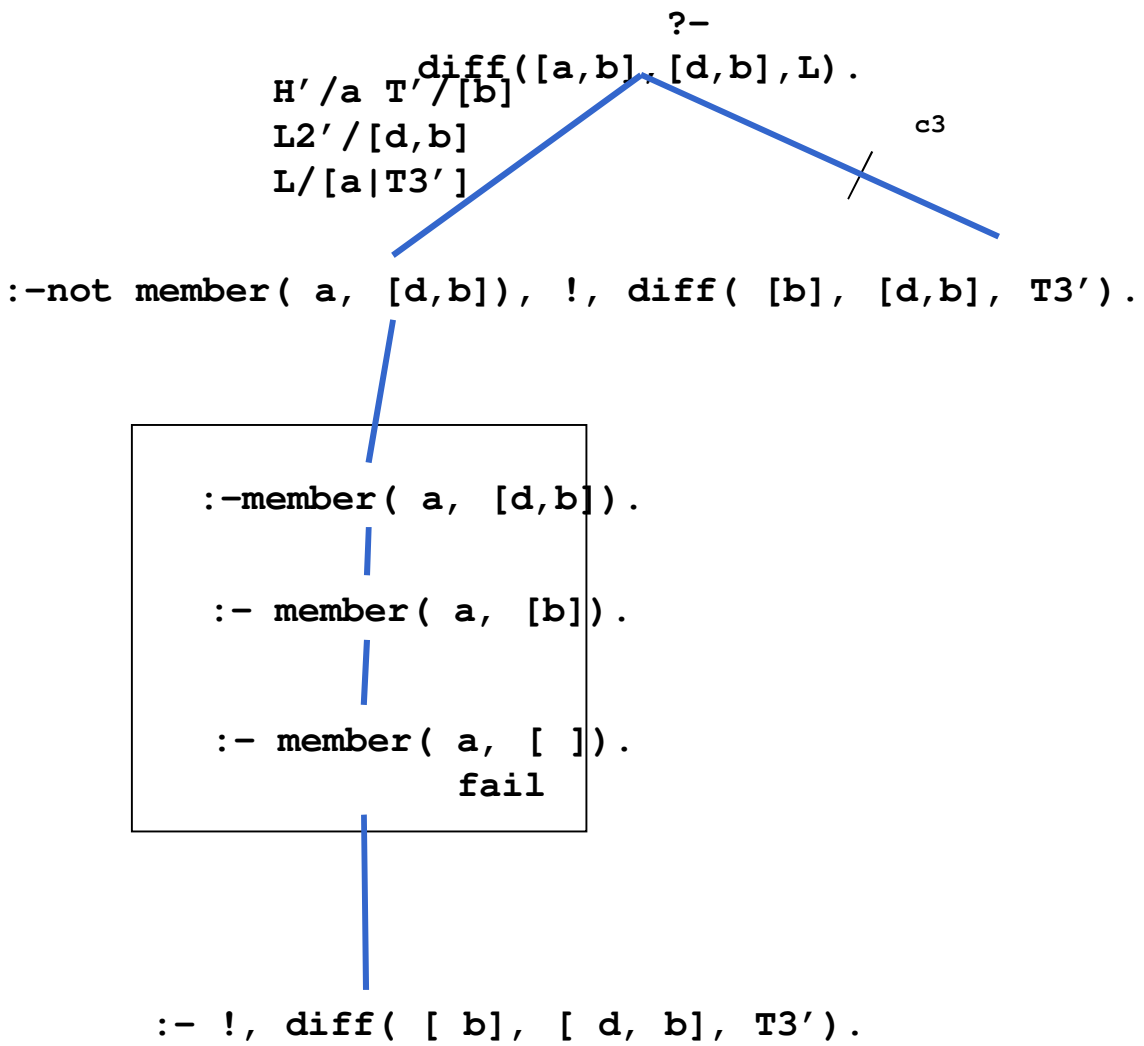
```
diff([],_,[]):- !.
diff( [H|T], L2, [H|T3]):- not member( H, L2),!,
                                    diff( T, L2, T3).
diff( [_|T], L2, T3):- diff( T, L2, T3).

member( X, [X|_]):-  !.
member( X, [_|T]):-  member(X,T).
```

Si rappresenti l'albero SLD relativo al goal

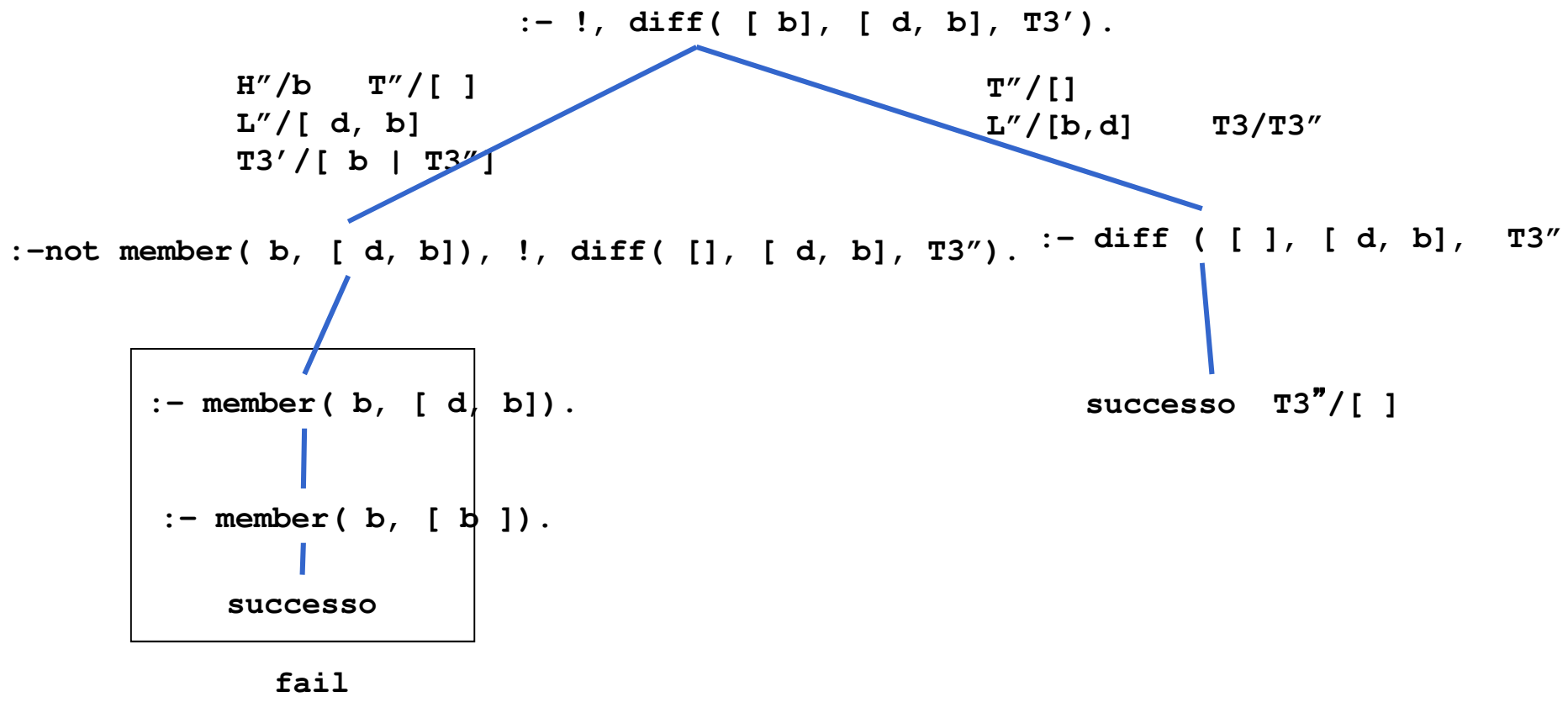
```
:- diff( [a,b], [d,b], L).
```

e si indichino i rami di successo.

# Soluzione Esercizio 8 – SLDNF



# Soluzione Esercizio 8 – SLDNF



# Esercizio 9 – Compito del 5 / 11 / 2003

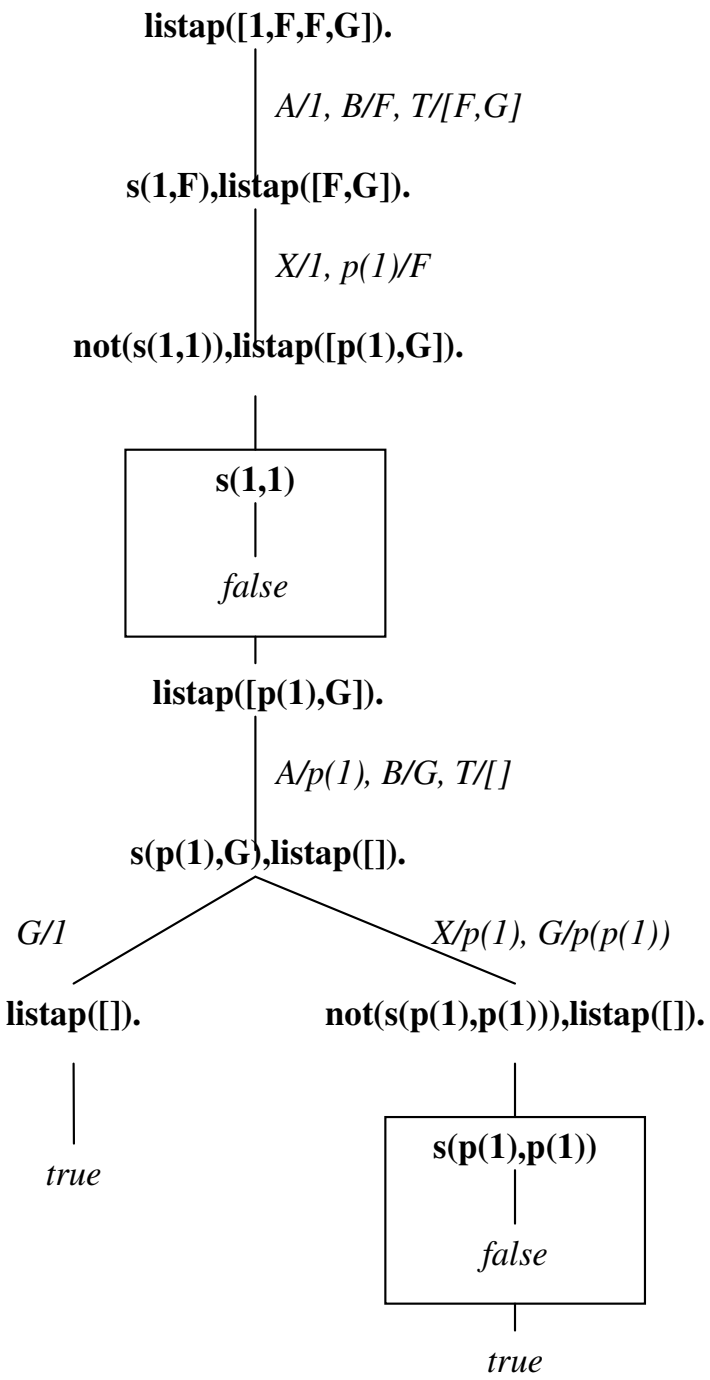
Si consideri il seguente programma Prolog:

```
listap([]).
listap([ A, B | T]):-
  s( A, B),
  listap(T).

s( p(X), X).
s( X,p(X)):-
  not(s(X,X)).
```

Si rappresenti l'albero SLDNF corrispondente al seguente goal:

```
listap([1,F,F,G]).
```

# Soluzione Esercizio 9
# Compito del 5/11/2003



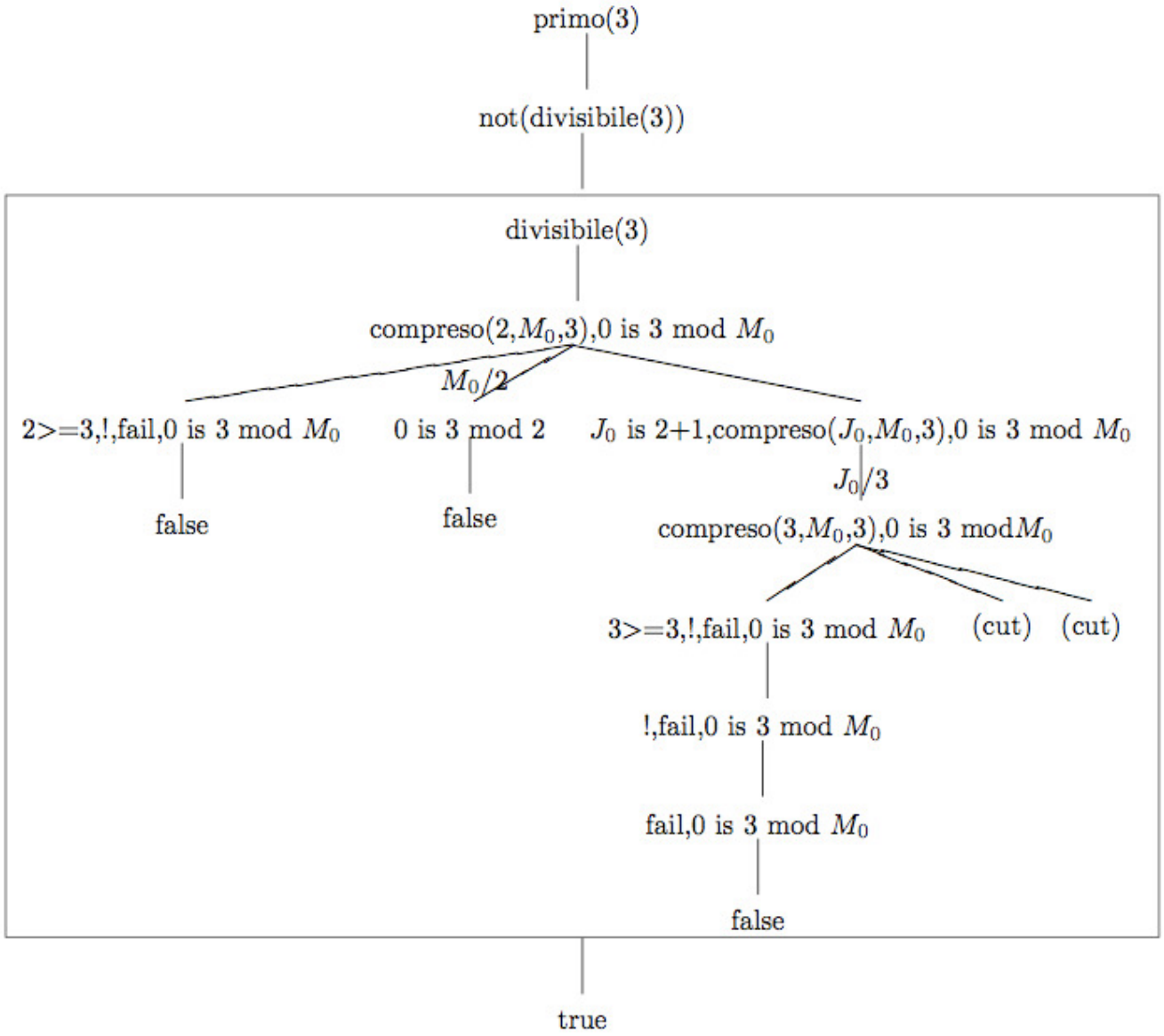
# Esercizio 10 – Compito del 20 / 12 / 2004

- Si consideri il seguente programma Prolog che calcola se un numero è primo (dove *mod* calcola il modulo, cioè il resto della divisione intera):

```
primo(N):- not(divisibile(N)).
divisibile(N):- compreso(2,M,N),0 is N mod M.

compreso(I,X,S):- I>=S, !, fail.
compreso(I,I,S).
compreso(I,X,S):- J is I+1, compreso(J,X,S).
```

- Si disegni l'albero SLDNF relativo al goal:

```
?- primo(3).
```


primo(3)
not(divisibile(3))
divisibile(3)
compreso(2,$M_0$,3),0 is 3 mod $M_0$
$M_0/2$
2>=3,!,fail,0 is 3 mod $M_0$
0 is 3 mod 2
$J_0$ is 2+1,compreso($J_0$,$M_0$,3),0 is 3 mod $M_0$
$J_0/3$
false
false
compreso(3,$M_0$,3),0 is 3 mod$M_0$
3>=3,!,fail,0 is 3 mod $M_0$
(cut)
(cut)
!,fail,0 is 3 mod $M_0$
fail,0 is 3 mod $M_0$
false
true

# Esercizio 11 - compito del 16 dicembre 2005

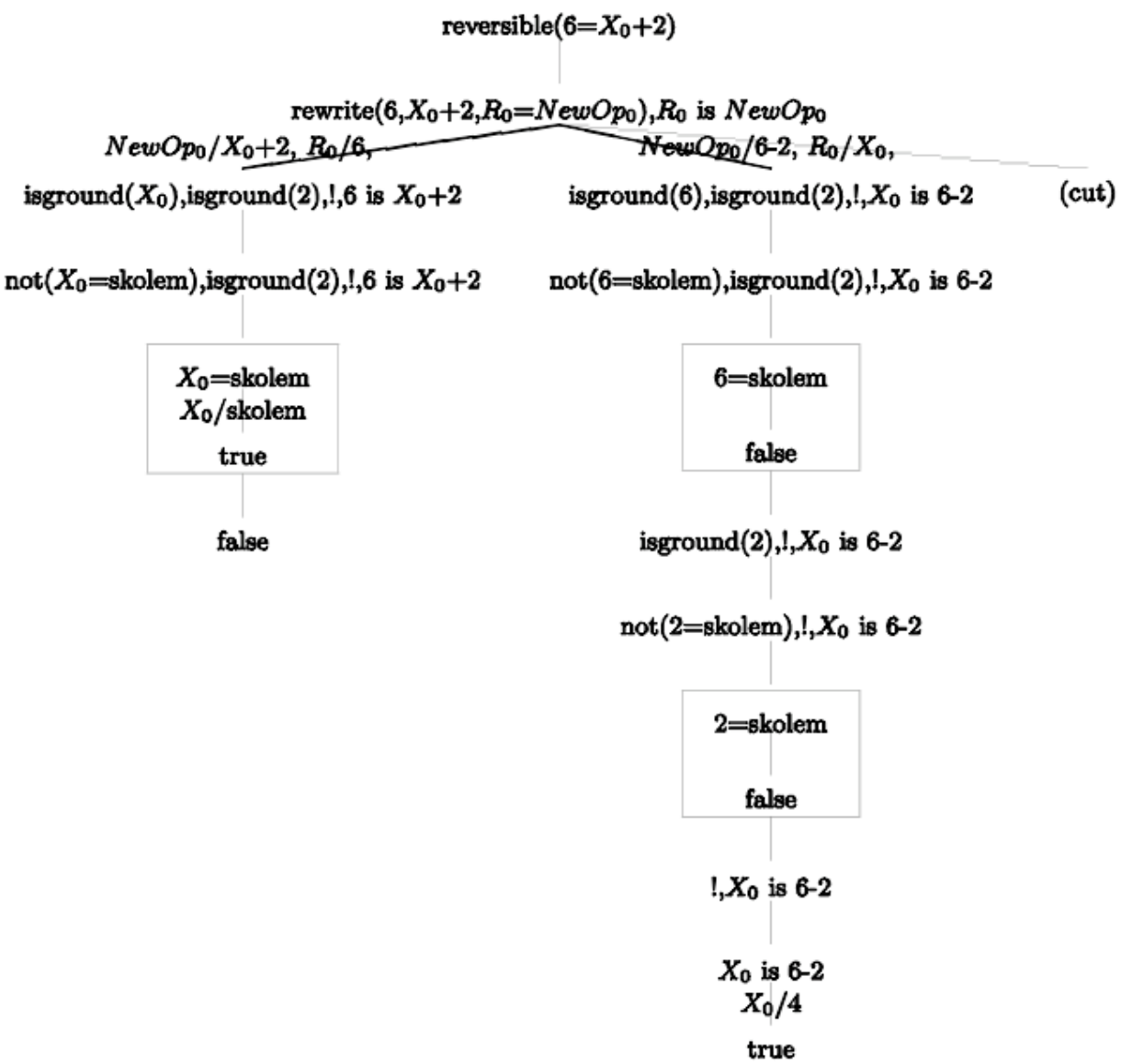
- Si consideri il seguente programma Prolog:

```
isground(X):-
        not(X=skolem).

reversible(S=Op):-
        rewrite(S,Op,R=NewOp),
        R is NewOp.

rewrite(S,A+B,S=A+B):- isground(A), isground(B), !.
rewrite(S,A+B,A=S-B):- isground(S), isground(B),!.
rewrite(S,A+B,B=S-A):- isground(S), isground(A).
```

Si rappresenti l'albero di derivazione SLD relativo al goal:

```
?- reversible(6=X+2).
```

e si dica qual'è la risposta calcolata.


reversible(6=X0+2)
rewrite(6,X0+2,R0=NewOp0),R0 is NewOp0
NewOp0/X0+2, R0/6,
NewOp0/6-2, R0/X0,
isground(X0),isground(2),!,6 is X0+2
isground(6),isground(2),!,X0 is 6-2
(cut)
not(X0=skolem),isground(2),!,6 is X0+2
not(6=skolem),isground(2),!,X0 is 6-2
X0=skolem
X0/skolem
true
6=skolem
false
false
isground(2),!,X0 is 6-2
not(2=skolem),!,X0 is 6-2
2=skolem
false
!,X0 is 6-2
X0 is 6-2
X0/4
true