Logica, alberi SLD e SLDNF

Esercizi – martedì 7 giugno 2004

- *Scopo:*
  1. Esercizi sulla logica dei predicati del primo ordine
  2. Esercizi su alberi di risoluzione SLD e SLDNF

Logica dei Predicati e Linguaggio Naturale

Non sempre e' facile rappresentare frasi del linguaggio naturale in logica.

"Esiste un cane nero"

Se il dominio dell'interpretazione (universo del discorso) e' solo di cani:

$\exists$ X nero (X).

Se il dominio ha anche altri oggetti che non sono cani, devo aggiungere la proprieta` di essere cani:

$\exists$ X (nero (X) $\wedge$ cane(X)).

Errore !:

$\exists$ X (cane(X) $\rightarrow$ nero (X)) e' equivalente a $\exists$ X (nero (X) $\vee$ $\neg$ cane(X)).

Tale formula e' vera in ogni dominio per cui c'e' un oggetto nero o c'e' un oggetto che non e' un cane.

Logica dei Predicati e Linguaggio Naturale (cont.)

"Tutti i corvi sono neri"

Se il dominio dell'interpretazione (universo del discorso) e' solo di corvi:

$$\forall X \text{ nero } (X).$$

Se il dominio ha anche altri oggetti che non sono corvi devo aggiungere la proprieta` di essere corvi:

$$\forall X (\text{corvo } (X) \rightarrow \text{nero}(X)).$$

Diverso significato:

$\forall X (\text{corvo}(X) \wedge \text{nero } (X))$ è equivalente a:

$\forall X (\text{nero } (X)) \wedge \forall X (\text{corvo}(X)).$

Tutti gli oggetti del dominio sono corvi e sono neri

Logica dei Predicati e Linguaggio Naturale (cont.)

"Tutte le scimmie sono fuggite su un albero"

Il dominio contiene differenti oggetti:
(scimmie, alberi + il predicato fuggire)

Procedimento Top-down:

$$\forall X (\text{scimmia } (X) \rightarrow A(X)).$$

Dove A(X) e` una formula logica non atomica che rappresenta "X e' fuggito su un albero", ovvero esiste un albero su cui X e' fuggito:

$$\exists Y (\text{albero}(Y) \wedge \text{fugge}(X,Y)).$$

Dunque:

$$\forall X \exists Y (\text{scimmia } (X) \rightarrow \text{fugge}(X,Y) \wedge \text{albero}(Y)).$$

Significato, alberi possibilmente diversi per scimmie diverse (l'albero dipende da X)

Logica dei Predicati e Linguaggio Naturale (cont.)

Altro significato:

"Tutte le scimmie sono fuggite sullo stesso albero"

In altro modo:

Esiste un albero su cui sono fuggite tutte le scimmie

Procedimento Top-down:

$$\exists Y (albero(Y) \wedge \forall X (scimmia(X) \rightarrow fugge(X,Y)))$$

Errore!

$$\forall X \ \exists Y (scimmia(X) \wedge fugge(X,Y) \wedge albero(Y)).$$

Ovvero:

$$\forall X \ scimmia(X) \wedge \exists Y (fugge(X,Y) \wedge albero(Y)).$$

Afferma che tutti gli oggetti sono scimmie e tutti gli oggetti sono fuggiti sull'albero.

Logica dei Predicati e Linguaggio Naturale (cont.)

"esiste una tarataruga che e' piu' veccchia di ogni essere umano"

$$\exists X (tartaruga(X) \wedge C(X)).$$

Dove C(X) e` una formula logica non atomica che rappresenta "X e' piu' vecchio di tutti gli esseri umani":

$$\forall Y \ umano(Y) \rightarrow piu\text{-}vecchio(X,Y).$$

Dunque:

$$\exists X (tartaruga(X) \wedge \forall Y \ umano(Y) \rightarrow piu\text{-}vecchio(X,Y)).$$

Sbagliata:

$$\exists X (tartaruga(X) \rightarrow \forall Y \ umano(Y) \rightarrow piu\text{-}vecchio(X,Y)).$$

(Significato vero se non esistono tartarughe mentre la frase originale lo afferma)

Esercizio 1

- Si trasformi la seguente frase della logica dei predicati del primo ordine nella forma a clausole:

- *"Le case grandi richiedono un grosso lavoro a meno che non abbiano una persona addetta alle pulizie e non abbiano il giardino".*

  $\forall$H big(H) $\wedge$ house(H) $\rightarrow$ work(H) $\vee$ {$\exists$M cleans(M,H) and not $\exists$ G garden(G,H)}

- Si discuta inoltre se sarebbe possibile trasformarla in clausole di Horn e si motivi la risposta.

Soluzione Esercizio 1

1. **Trasformazione in fbf chiuse**
2. **Elimino le implicazioni: A $\rightarrow$ B equivale a not A $\vee$ B**

   $\forall$ H not big(H) $\vee$ not house(H) $\vee$ work(H) $\vee$ {$\exists$ M cleans(M,H) $\wedge$ not $\exists$ G garden(G,H)}
3. **Riduzione del connettivo not a soli atomi e non più a formule composte**

   $\forall$ H not big(H) $\vee$ not house(H) $\vee$ work(H) $\vee$ {$\exists$ M cleans(M,H) $\wedge$ $\forall$ G not garden(G,H)}
4. **Cambiamento di nomi delle variabili (in caso di conflitti).**

Soluzione Esercizio 1

5. **Spostamento dei quantificatori in testa alla formula**

   $\forall$ H $\exists$ M $\forall$ G not big(H) $\vee$ not house(H) $\vee$ work(H) $\vee$ {cleans(M,H) $\wedge$ not garden(G,H)}

6. **Forma prenessa congiuntiva (congiunzione di disgiunzioni)**

   $\forall$ H $\exists$ M $\forall$ G ((not big(H) $\vee$ not house(H) $\vee$ work(H) $\vee$ cleans(M,H)) $\wedge$ (not big(H) $\vee$ not house(H) $\vee$ work(H) $\vee$ not garden(G,H)))

7. **Skolemizzazione**

   $\forall$ H $\forall$ G ((not big(H) $\vee$ not house(H) $\vee$ work(H) $\vee$ cleans(f(H),H)) $\wedge$ (not big(H) $\vee$ not house(H) $\vee$ work(H) $\vee$ not garden(G,H)))

8. **Eliminazione dei quantificatori universali**

Soluzione Esercizio 1

Forma a clausole:

not big(H) $\vee$ not house(H) $\vee$ work(H) $\vee$ cleans(f(H),H)

not big(H) $\vee$ not house(H) $\vee$ work(H) $\vee$ not garden(G,H)

La frase non può essere trasformata in clausole di Horn a causa dei letterali positivi: infatti la prima clausola contiene due letterali positivi, mentre le clausole di Horn ne contengono al più uno.

Esercizio 2 - risoluzione

Si assumano i seguenti fatti:

- A Simone piacciono soltanto i corsi facili;
- I corsi di scienze sono difficili;
- Tutti i corsi del dipartimento di Intelligenza Artificiale sono facili;
- BK301 è un corso di Intelligenza Artificiale.

Si usi la risoluzione per rispondere alla domanda: Quale corso piace a Simone?

Soluzione Esercizio 2 - risoluzione

- A Simone piacciono soltanto i corsi facili;

  $\forall$**X,** $\forall$**Y** **corso(Y,X), facile(X)** $\rightarrow$ **piace(simone,X)**
- I corsi di scienze sono difficili;

  $\forall$**X** **corso(scienze, X)** $\rightarrow$ **not facile(X)**
- Tutti i corsi del dipartimento di Intelligenza Artificiale sono facili;

  $\forall$**X** **corso(ai, X)** $\rightarrow$ **facile(X)**
- BK301 è un corso di Intelligenza Artificiale.

  corso(ai,bk301).

Goal G: $\exists$ **X piace(simone,X), corso(Y,X)**

Soluzione Esercizio 2 - risoluzione

Forma a clausole:

- Gneg: not piace(simone,X) $\vee$ not corso(Y,X)
- C1: piace(simone,X) $\vee$ not corso(Y,X) $\vee$ not facile(X)
- C2: not facile(X) $\vee$ not corso(scienze,X)
- C3: facile(X) $\vee$ not corso(ai,X)
- C4: corso(ai,bk301)

Soluzione Esercizio 2 - risoluzione

Risoluzione

C5 = G e C4 : not piace(simone, bk301) X/bk301 Y/ai

C6 = C3 e C4: facile(bk301)

C7 = C1 e C5: not corso(Y, bk301) $\vee$ not facile(bk301)

C8 = C6 e C7: not corso(Y,bk301)

C9 = C8 e C4: contraddizione

Esercizio 3 – compito del 5/11/2003

Si formalizzino le seguenti frasi in logica dei predicati:

- Esiste almeno studente di Ingegneria che conosce la logica booleana.
- Chi conosce la logica booleana ha capacità logiche.
- Chi non ha capacità logiche, si contraddice.
- Chi si contraddice, non ha capacità logiche.
- Piero studia ad ingegneria e conosce la logica booleana.

Le si trasformi in clausole e si usi poi il principio di risoluzione per dimostrare che c'è uno studente di Ingegneria che non si contraddice.

Soluzione Esercizio 3

- Esiste almeno studente di Ingegneria che conosce la logica booleana.

  ∃Y (studIng(Y) and conosce(Y,boole))
- Chi conosce la logica booleana ha capacità logiche.

  ∀ X (conosce(X,boole) => haLogica(X))
- Chi non ha capacità logiche, si contraddice.

  ∀ X (not haLogica(X) => contraddice(X))
- Chi si contraddice, non ha capacità logiche.

  ∀ X (contraddice(X) => not haLogica(X))
- Piero studia ad ingegneria e conosce la logica booleana.

  studIng(piero) and conosce(piero,boole)

Goal: **∃Y studIng(Y) and not contraddice(Y)**

Soluzione Esercizio 3

Clausole:

- **C1 studIng(c)**
- **C2 conosce(c,boole)**
- **C3 not conosce(X,boole) or haLogica(X)**
- **C4 haLogica(X) or contraddice(X)**
- **C5 not contraddice(X) or not haLogica(X)**
- **C6 studIng(piero)**
- **C7 conosce(piero,boole)**

- **C8 not studIng(Y) or contraddice(Y)** (goal negato)

Soluzione Esercizio 3

Risoluzione:

- **C9 not haLogica(Y) or not studIng(Y)** (da C5 e C8)
- **C10 not conosce(Y,boole) or not studIng(Y)** (da C9 e C3)
- **C11 not conosce(piero,boole)** (da C10 e C6)
- **C12 Clausola vuota** (da C11 e C7)

Esercizio 4

Si consideri la seguente conoscenza:

Antonio, Michele e Giovanni sono iscritti al CAI (Club Alpino Italiano). Ogni appartenente al Club che non è sciatore è uno scalatore. Gli scalatori non amano la pioggia. Ogni persona che non ama la neve non è uno sciatore. Antonio non ama ciò che Michele ama. Antonio ama la pioggia e la neve.

Si rappresenti tale conoscenza come un insieme di predicati del primo ordine appropriati per un sistema di refutazione che lavori mediante risoluzione.

Si mostri come tale sistema risolverebbe la domanda: "**C'è un membro del CAI che è uno scalatore, ma non uno sciatore?**"

Soluzione Esercizio 4

Formule logiche:

1. ∀ X iscritto(X), not sciatore(X) → scalatore(X)

2. ∀ X scalatore (X) → not ama(X,pioggia)

3. ∀ X not ama(X,neve) → not sciatore(X)

4. ∀ X ama(michele,X) → not ama(antonio,X)

5. ama(antonio,neve)

6. ama(antonio, pioggia)

7. iscritto(antonio)

8. iscritto(michele)

9. iscritto(giovanni)

Goal: ∃ **X iscritto(X), scalatore(X), not sciatore(X)**

Soluzione Esercizio 4

Forma a clausole:

C1. not iscritto(X) $\vee$ sciatore(X) $\vee$ scalatore(X)
C2. not scalatore(X) $\vee$ not ama(X,pioggia)
C3. ama(X,neve) $\vee$ not sciatore(X)
C4. not ama(michele,X) $\vee$ not ama(antonio,X)
C5. ama(antonio,neve)
C6. ama(antonio, pioggia)
C7. iscritto(antonio)
C8. iscritto(michele)
C9. iscritto(giovanni)

Gneg: not iscritto(X) $\vee$ not scalatore(X) $\vee$ sciatore(X)

Soluzione Esercizio 4

Risoluzione

C10=Gneg - C8
not scalatore(michele) Ú sciatore(michele) {X/michele}

C11=C10 - C3
not scalatore(michele) Ú ama(michele,neve)

C12=C11-C4
not scalatore(michele) Ú not ama(antonio,neve)

C13=C12 e C5 not scalatore(michele)
C14=C13 e C1 not iscritto(michele) Ú sciatore(michele)
C15=C13 e C8 sciatore(michele)
C16=C15 e C3 ama(michele,neve)
C17=C16 e C4 not ama(antonio,neve)
C18=C17 e C5 clausola vuota

Esercizio 5 - 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:

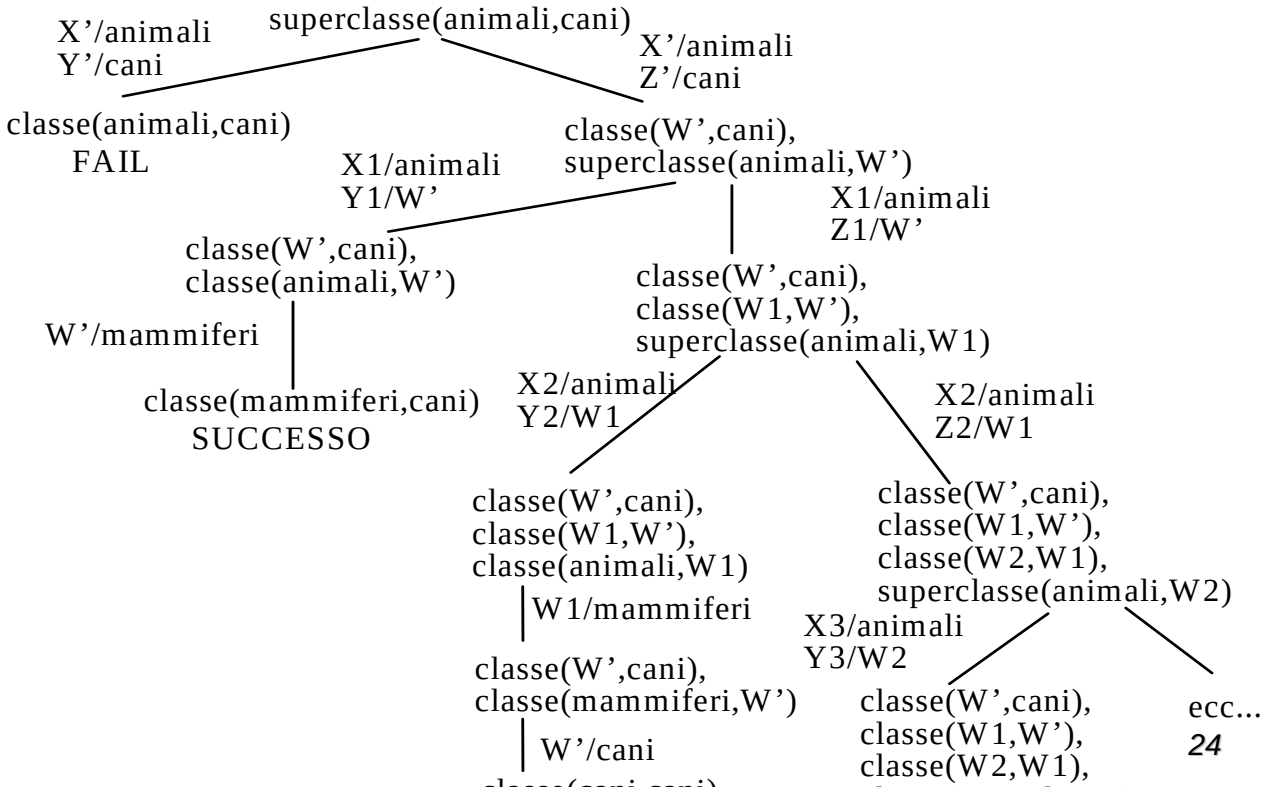
:- superclasse(animali, cani)

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

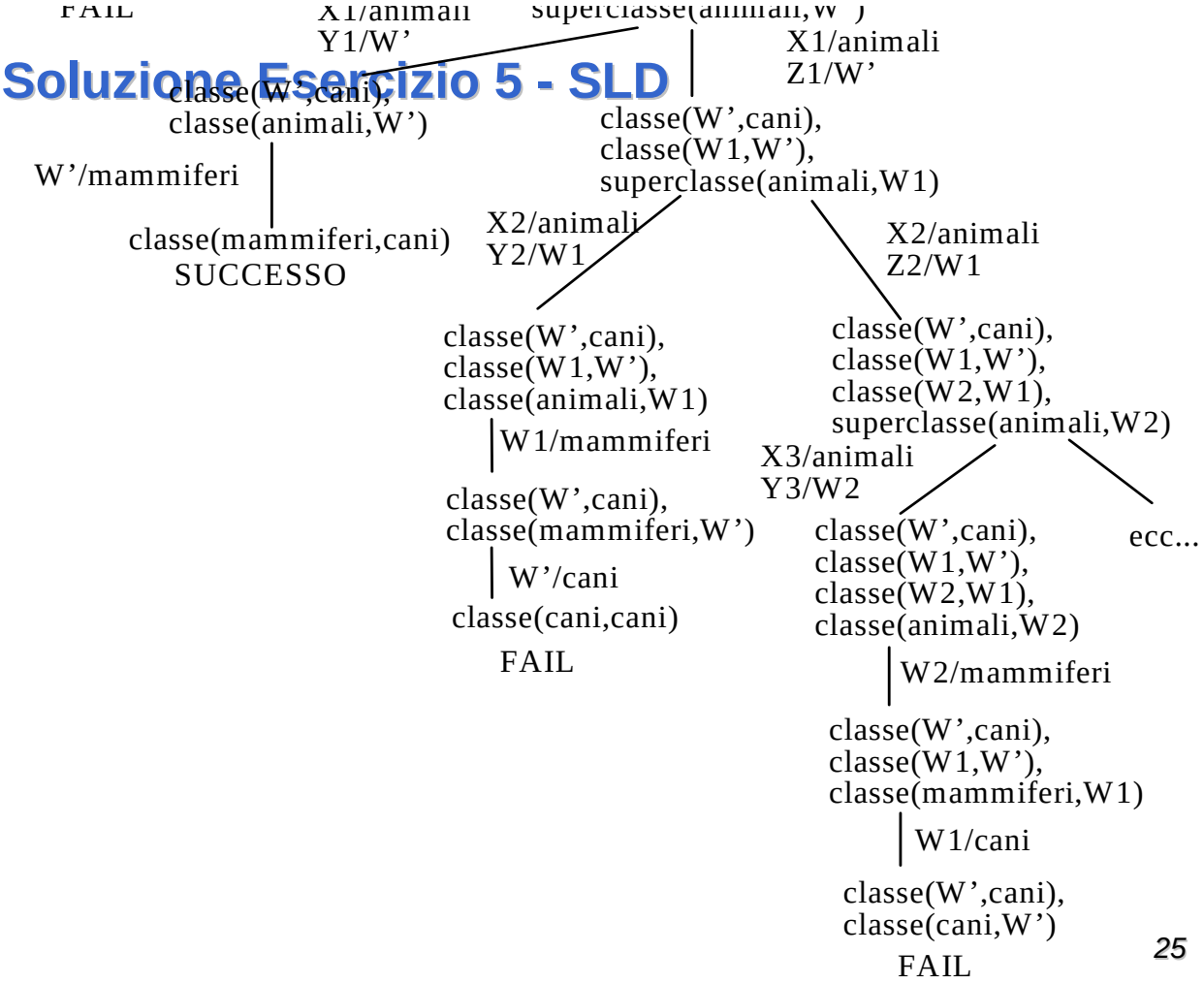
classe(mammiferi,cani).

classe(animali,mammiferi).

Soluzione Esercizio 5 - SLD



Soluzione Esercizio 5 - SLD



Esercizio 6 – SLD & cut

Si consideri il seguente programma Prolog:

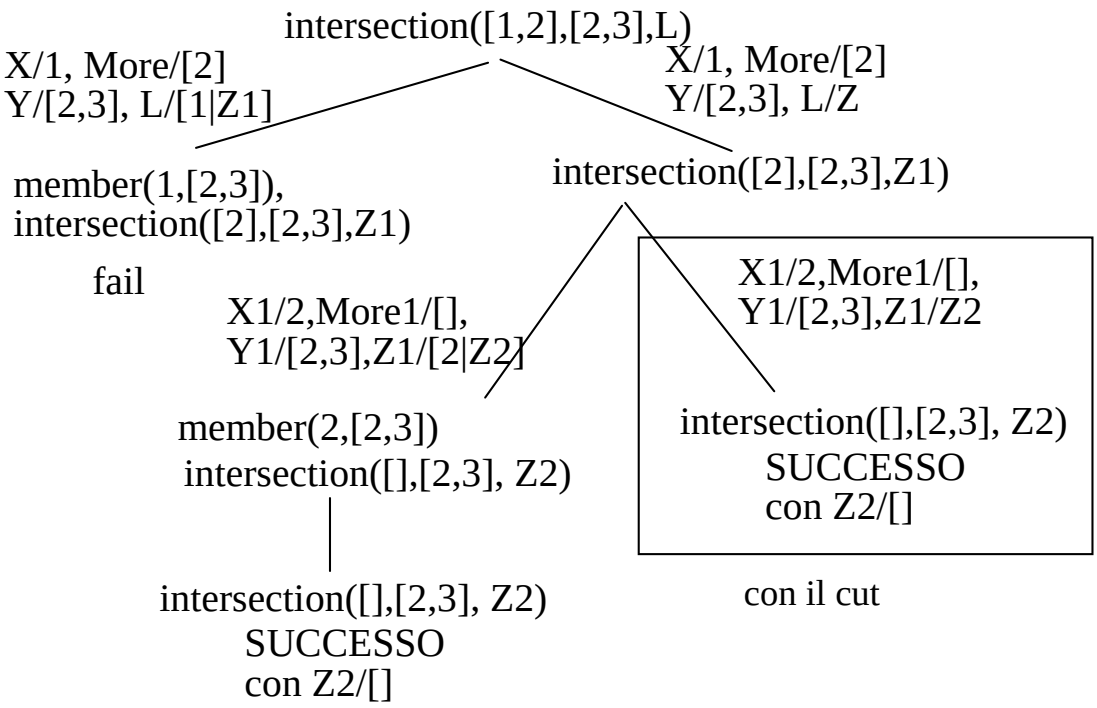
```
intersection([],X,[]).
intersection([X|More],Y,[X|Z]):-
    member(X,Y),
    intersection(More,Y,Z).
intersection([X|More],Y,Z):-      intersection(More,Y,Z).
```

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 6 – SLD & cut



Esercizio 7 – SLDNF

Si consideri il seguente programma Prolog:

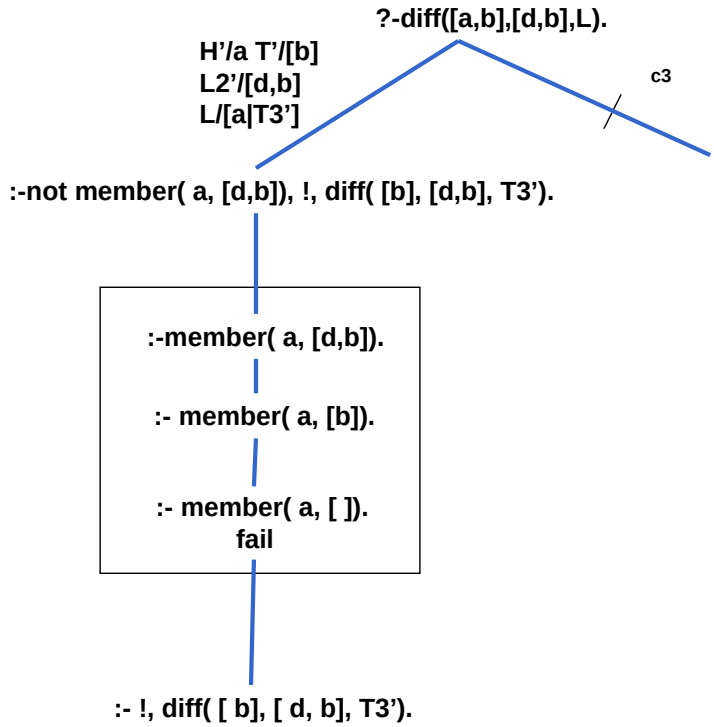
```
diff([],_,[]):- !.
diff( [H|T], L2, [H|T3]):- not member( H, L2),!,
                            diff( T, L2, T3).
diff( [_|T], L2, T3):- diff( T, L2, T3).

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

Si rappresenti l'albero SLD relativo al goal

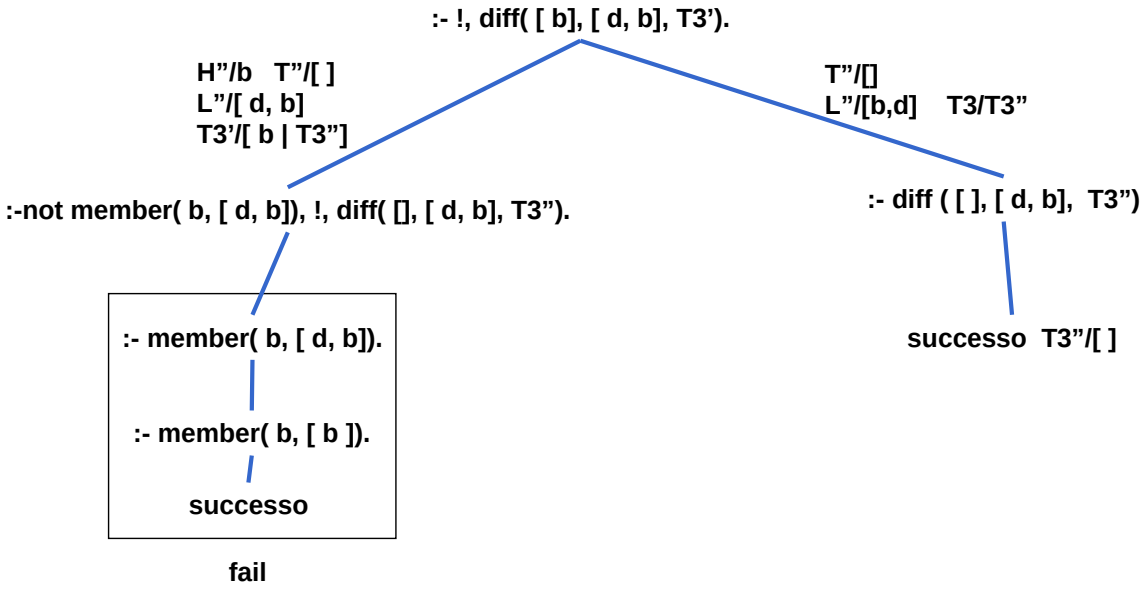
:- diff([a,b], [d,b], L).

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

Soluzione Esercizio 7 – SLDNF



Soluzione Esercizio 7 – SLDNF



Esercizio 8 – Compito del 5 / 11 / 2003

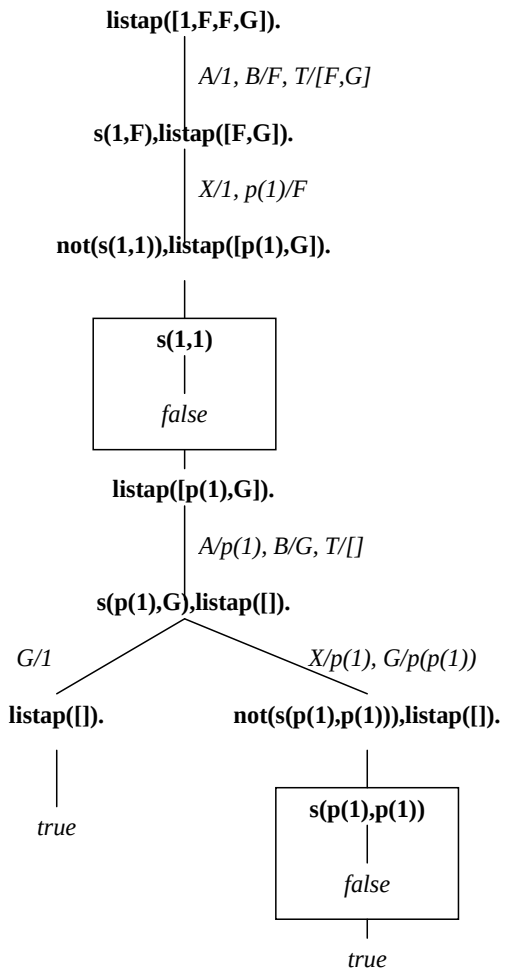
Si consideri il seguente programma Prolog:

```
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 8 – Compito del 5/11/2003



Esercizio 9 – 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).
```