

Logica, alberi SLD e SLDNF

Esercizi – martedì 10 giugno 2008/ mercoledì 11 giugno 2008

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

1

CLAUSOLE

- Una **clausola** è una disgiunzione di letterali (cioè formule atomiche negate e non negate), in cui tutte le variabili sono quantificate universalmente in modo implicito.
- Una clausola generica può essere rappresentata come la disgiunzione:
$$A_1 \vee A_2 \vee \dots \vee A_n \vee \sim B_1 \vee \dots \vee \sim B_m$$
dove A_i ($i=1, \dots, n$) e B_j ($j=1, \dots, m$) sono atomi.
- Una clausola nella quale non compare alcun letterale, sia positivo sia negativo, è detta **clausola vuota** e verrà indicata con \square , interpretato come contraddizione: disgiunzione falso $\vee \sim$ vero
- Un sottoinsieme delle clausole è costituito dalle **clausole definite**, nelle quali si ha sempre un solo letterale positivo:

$$A_1 \vee \sim B_1 \vee \dots \vee \sim B_m$$

2

TRASFORMAZIONE IN CLAUSOLE (1)

- Passi per trasformare una qualunque fbf della logica dei predicati del primo ordine in un insieme di clausole

- 1) **Trasformazione in fbf chiusa**

Esempio la formula:

$$- \forall X (p(Y) \rightarrow \sim(\forall Y (q(X,Y) \rightarrow p(Y)))) \quad (1)$$

è trasformata in:

$$- \forall X \forall Y (p(Y) \rightarrow \sim(\forall Y (q(X,Y) \rightarrow p(Y)))) \quad (2)$$

- 2) **Applicazione delle equivalenze per i connettivi logici** (ad esempio $A \rightarrow B$ è sostituito da $\sim A \vee B$) e la si riduce in forma and-or.

La formula (2) diventa:

$$- \forall X \forall Y (\sim p(Y) \vee \sim(\forall Y (\sim q(X,Y) \vee p(Y)))) \quad (3)$$

3

TRASFORMAZIONE IN CLAUSOLE (2)

- 3) **Applicazione della negazione ad atomi e non a formule composte**, tenendo presente che:

$$\forall X \sim A \quad \text{equivale a} \quad \sim \exists X A$$

$$\exists X \sim A \quad \text{equivale a} \quad \sim \forall X A$$

$$\sim(A_1 \vee A_2 \vee \dots \vee A_n) \quad \text{equivale a} \quad \sim A_1 \wedge \sim A_2 \wedge \dots \wedge \sim A_n$$

$$\sim(A_1 \wedge A_2 \wedge \dots \wedge A_n) \quad \text{equivale a} \quad \sim A_1 \vee \sim A_2 \vee \dots \vee \sim A_n$$

(leggi di De Morgan).

(3) si modifica in:

$$\forall X \forall Y (\sim p(Y) \vee (\exists Y (q(X,Y) \wedge \sim p(Y)))) \quad (4)$$

- 4) **Cambiamento di nomi delle variabili**, nel caso di conflitti.

in (4) la seconda variabile Y viene rinominata Z:

$$\forall X \forall Y (\sim p(Y) \vee (\exists Z (q(X,Z) \wedge \sim p(Z)))) \quad (5)$$

4

TRASFORMAZIONE IN CLAUSOLE (3)

- 5) **Spostamento dei quantificatori** in testa alla formula (forma prenessa).

$$\forall X \forall Y \exists Z (\sim p(Y) \vee (q(X,Z) \wedge \sim p(Z))) \quad (6)$$

- 6) **Forma normale congiuntiva** cioè come congiunzione di disgiunzioni, con quantificazione in testa.

$$\forall X \forall Y \exists Z ((\sim p(Y) \vee q(X,Z)) \wedge (\sim p(Y) \vee \sim p(Z))) \quad (7)$$

- 7) **Skolemizzazione**: ogni variabile quantificata esistenzialmente viene sostituita da una funzione delle variabili quantificate universalmente che la precedono. Tale funzione è detta funzione di Skolem.

Ad esempio una formula del tipo: $\forall X \exists Y p(X,Y)$ può essere espressa in modo equivalente come: $\forall X p(X,g(X))$

In (7) Z è sostituita da $f(X,Y)$, perché Z si trova nel campo di azione delle quantificazioni $\forall X$ e $\forall Y$:

$$\forall X \forall Y ((\sim p(Y) \vee q(X,f(X,Y))) \wedge (\sim p(Y) \vee \sim p(f(X,Y)))) \quad (8)$$

5

TRASFORMAZIONE IN CLAUSOLE (4)

- **Perdita in espressività**. Non è la stessa cosa asserire: $F: \exists X p(X)$ oppure $F': p(f)$.
- Vale comunque la proprietà che F è inconsistente se e solo se F' è inconsistente.
- 8) **Eliminazione dei quantificatori universali**: si ottiene una formula detta universale (tutte le sue variabili sono quantificate universalmente) in forma normale congiuntiva.

$$((\sim p(Y) \vee q(X,f(X,Y))) \wedge (\sim p(Y) \vee \sim p(f(X,Y)))) \quad (9)$$

- Una formula di questo tipo rappresenta **un insieme di clausole** (ciascuna data da un congiunto nella formula). La forma normale a clausole che si ottiene: $\{\sim p(Y) \vee q(X,f(X,Y)), \sim p(Y) \vee \sim p(f(X,Y))\}$ (10)

- La seconda clausola può essere riscritta rinominando le variabili (sostituendo cioè la formula con una sua variante).

$$\{\sim p(Y) \vee q(X,f(X,Y)), \sim p(Z) \vee \sim p(f(W,Z))\} \quad (11)$$

6

TRASFORMAZIONE IN CLAUSOLE (5)

- Qualunque teoria del primo ordine T può essere trasformata in una teoria T' in forma a clausole.
- Anche se T non è logicamente equivalente a T' (a causa dell'introduzione delle funzioni di Skolem), vale comunque la seguente proprietà:

Proprietà

- Sia T una teoria del primo ordine e T' una sua trasformazione in clausole. Allora T è insoddisfacibile se e solo se T' è insoddisfacibile.
- Il principio di risoluzione è una procedura di dimostrazione che opera per contraddizione e si basa sul concetto di insoddisfacibilità.

7

IL PRINCIPIO DI RISOLUZIONE

- Il principio di risoluzione, che si applica a formule in forma a clausole, è molto più efficiente del metodo assiomatico-deduttivo ed è utilizzato dalla maggior parte dei risolutori automatici di teoremi.
- **Logica Proporzionale:** clausole prive di variabili.
- Siano C_1 e C_2 due clausole prive di variabili:

$$C_1 = A_1 \vee \dots \vee A_n \qquad C_2 = B_1 \vee \dots \vee B_m$$

- Se esistono in C_1 e C_2 due letterali **opposti**, A_i e B_j , ossia tali che $A_i = \sim B_j$, allora da C_1 e C_2 , (clausole **parent**) si può derivare una nuova clausola C_3 , denominata **risolvente**, della forma:

$$C_3 = A_1 \vee \dots \vee A_{i-1} \vee A_{i+1} \vee \dots \vee A_n \vee B_1 \vee \dots \vee B_{j-1} \vee B_{j+1} \vee \dots \vee B_m$$

- C_3 è conseguenza logica di $C_1 \cup C_2$.

$$\begin{array}{ccc} C1: L \vee C1' & & C2: \neg L \vee C2' \\ & \searrow & \swarrow \\ & C: C1' \vee C2' & \end{array}$$

8

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:

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

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

$\exists X (\text{nero}(X) \wedge \text{cane}(X))$.

Errore !:

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

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

9

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:

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

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

Tutti gli oggetti del dominio sono corvi e sono neri

10

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 $A(X)$ è una formula logica non atomica che rappresenta “X è fuggito su un albero”, ovvero esiste un albero su cui X è 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)

11

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:

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

Errore!

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

Ovvero:

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

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

12

Logica dei Predicati e Linguaggio Naturale (cont.)

“esiste una tartaruga che e' piu' vecchia di ogni essere umano”

$$\exists X (\text{tartaruga}(X) \wedge C(X)).$$

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

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

Dunque:

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

Sbagliata:

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

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

13

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".*
 $\forall H \text{ big}(H) \wedge \text{house}(H) \rightarrow \text{work}(H) \vee \{\exists M \text{ cleans}(M,H) \text{ and not } \exists G \text{ garden}(G,H)\}$
- Si discuta inoltre se sarebbe possibile trasformarla in clausole di Horn e si motivi la risposta.

14

Soluzione Esercizio 1

1. Trasformazione in fbf chiuse

2. Elimino le implicazioni: $A \rightarrow B$ equivale a $\text{not } A \vee B$

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

3. Riduzione del connettivo not a soli atomi e non più a formule composte

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

4. Cambiamento di nomi delle variabili (in caso di conflitti).

15

Soluzione Esercizio 1

5. Spostamento dei quantificatori in testa alla formula

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

6. Forma prenessa congiuntiva (congiunzione di disgiunzioni)

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

7. Skolemizzazione

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

8. Eliminazione dei quantificatori universali

16

Soluzione Esercizio 1

Forma a clausole:

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

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

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.

17

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?

18

Soluzione Esercizio 2 - risoluzione

- A Simone piacciono soltanto i corsi facili;
 $\forall X, \forall Y \quad \text{corso}(Y,X), \text{facile}(X) \rightarrow \text{piace}(\text{simone},X)$
- I corsi di scienze sono difficili;
 $\forall X \quad \text{corso}(\text{scienze}, X) \rightarrow \text{not facile}(X)$
- Tutti i corsi del dipartimento di Intelligenza Artificiale sono facili;
 $\forall X \quad \text{corso}(\text{ai}, X) \rightarrow \text{facile}(X)$
- BK301 è un corso di Intelligenza Artificiale.
 $\text{corso}(\text{ai}, \text{bk301})$.

Goal G: $\exists X \text{ piace}(\text{simone},X), \text{corso}(Y,X)$

19

Soluzione Esercizio 2 - risoluzione

Forma a clausole:

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

20

Soluzione Esercizio 2 - risoluzione

Risoluzione

$C5 = G$ e $C4$: $\text{not piace}(\text{simone}, \text{bk301}) \wedge \text{X/bk301} \wedge \text{Y/ai}$

$C6 = C3$ e $C4$: $\text{facile}(\text{bk301})$

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

$C8 = C6$ e $C7$: $\text{not corso}(\text{Y}, \text{bk301})$

$C9 = C8$ e $C4$: contraddizione

21

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.

22

Soluzione Esercizio 3

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

$\exists Y$ (**studIng(Y) and conosce(Y,boole)**)

- Chi conosce la logica booleana ha capacità logiche.

$\forall X$ (**conosce(X,boole) => haLogica(X)**)

- Chi non ha capacità logiche, si contraddice.

$\forall X$ (**not haLogica(X) => contraddice(X)**)

- Chi si contraddice, non ha capacità logiche.

$\forall X$ (**contraddice(X) => not haLogica(X)**)

- Piero studia ad ingegneria e conosce la logica booleana.

studIng(piero) and conosce(piero,boole)

Goal: $\exists Y$ **studIng(Y) and not contraddice(Y)**

23

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)

24

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)

25

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?"

26

Soluzione Esercizio 4

Formule logiche:

1. $\forall X \text{ iscritto}(X), \text{ not sciatore}(X) \rightarrow \text{scalatore}(X)$
2. $\forall X \text{ scalatore}(X) \rightarrow \text{not ama}(X, \text{pioggia})$
3. $\forall X \text{ not ama}(X, \text{neve}) \rightarrow \text{not sciatore}(X)$
4. $\forall X \text{ ama}(\text{michele}, X) \rightarrow \text{not ama}(\text{antonio}, X)$
5. $\text{ama}(\text{antonio}, \text{neve})$
6. $\text{ama}(\text{antonio}, \text{pioggia})$
7. $\text{iscritto}(\text{antonio})$
8. $\text{iscritto}(\text{michele})$
9. $\text{iscritto}(\text{giovanni})$

Goal: $\exists X \text{ iscritto}(X), \text{ scalatore}(X), \text{ not sciatore}(X)$

27

Soluzione Esercizio 4

Forma a clausole:

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

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

28

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

29

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)
```

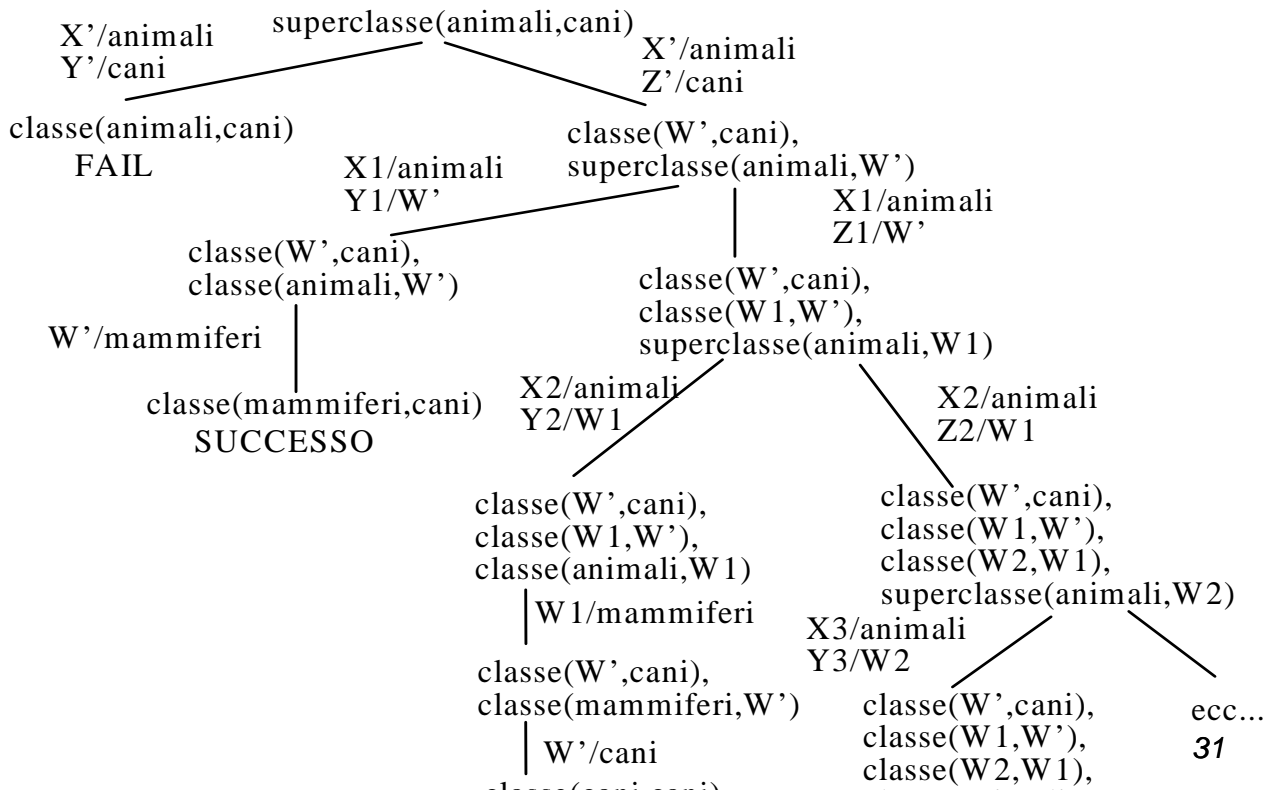
assumendo la presenza, all'inizio del database, dei fatti:

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

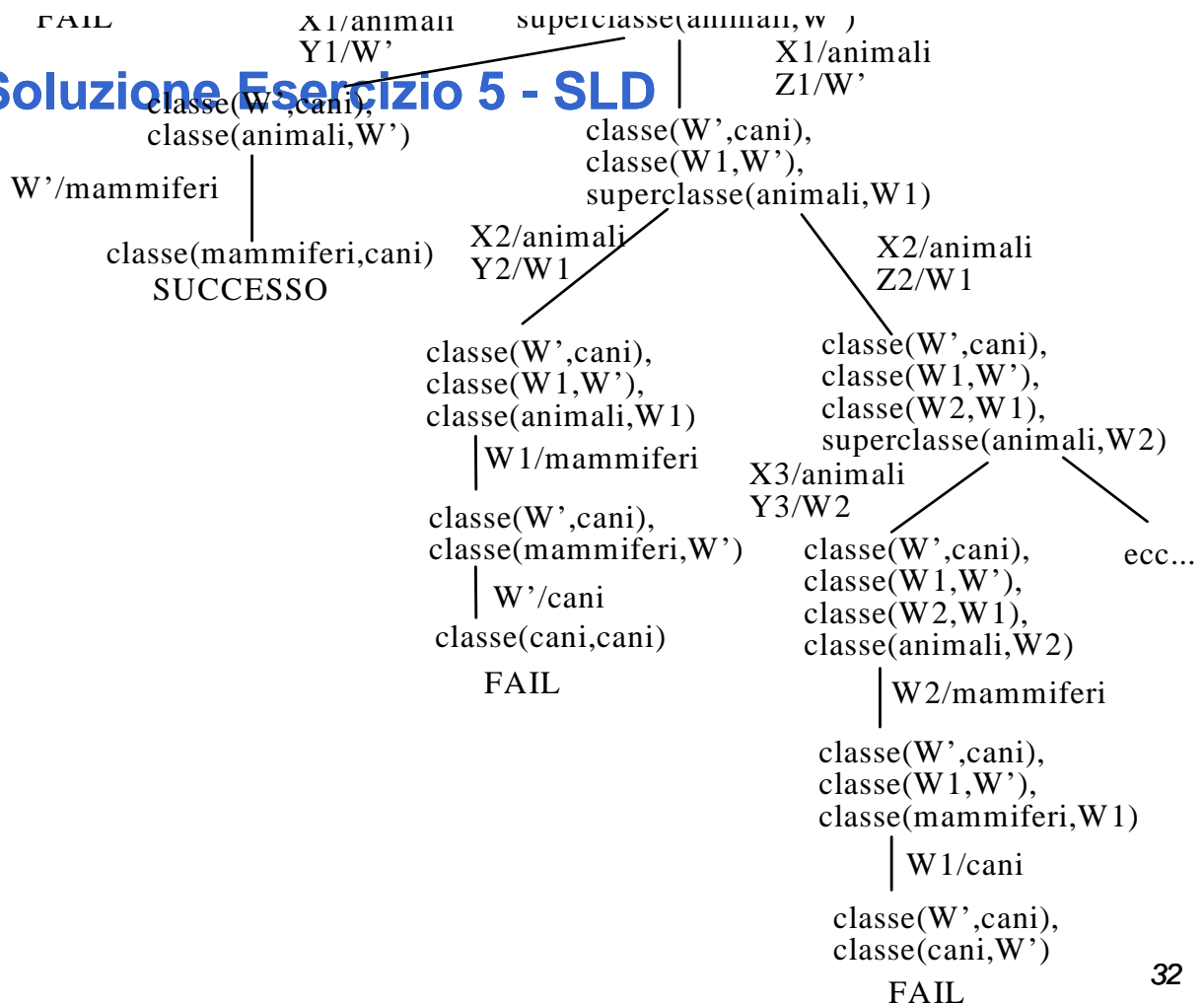
```
classe(animali, mammiferi) .
```

30

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).
```

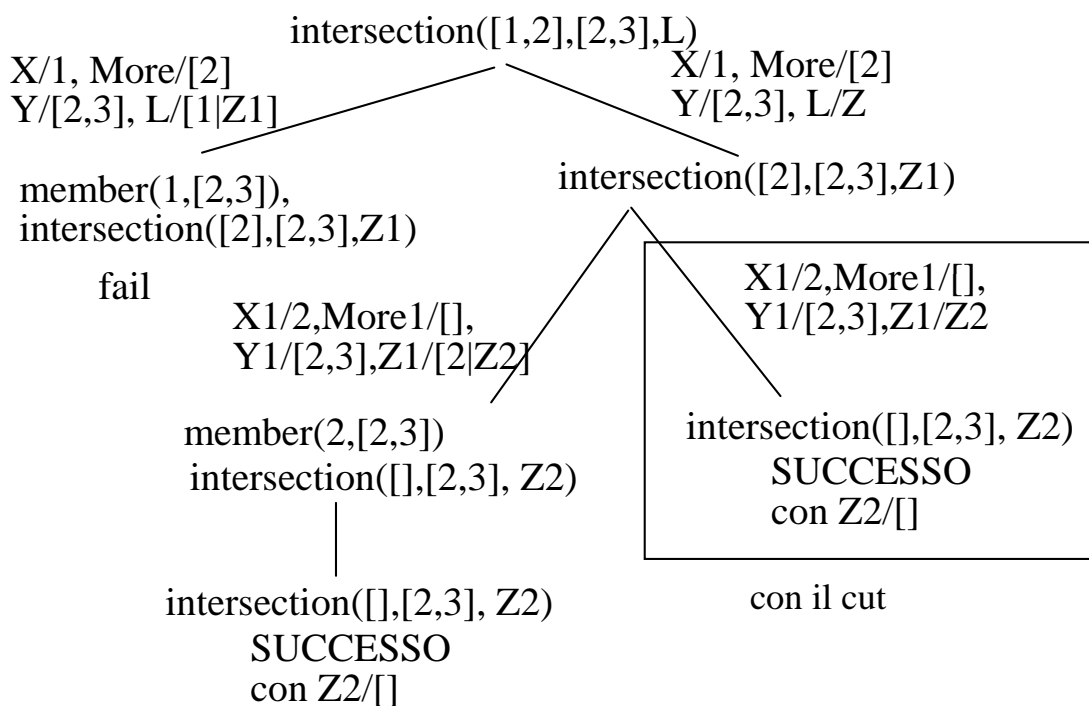
Si rappresenti l'albero SLD relativo al goal

```
:- 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.

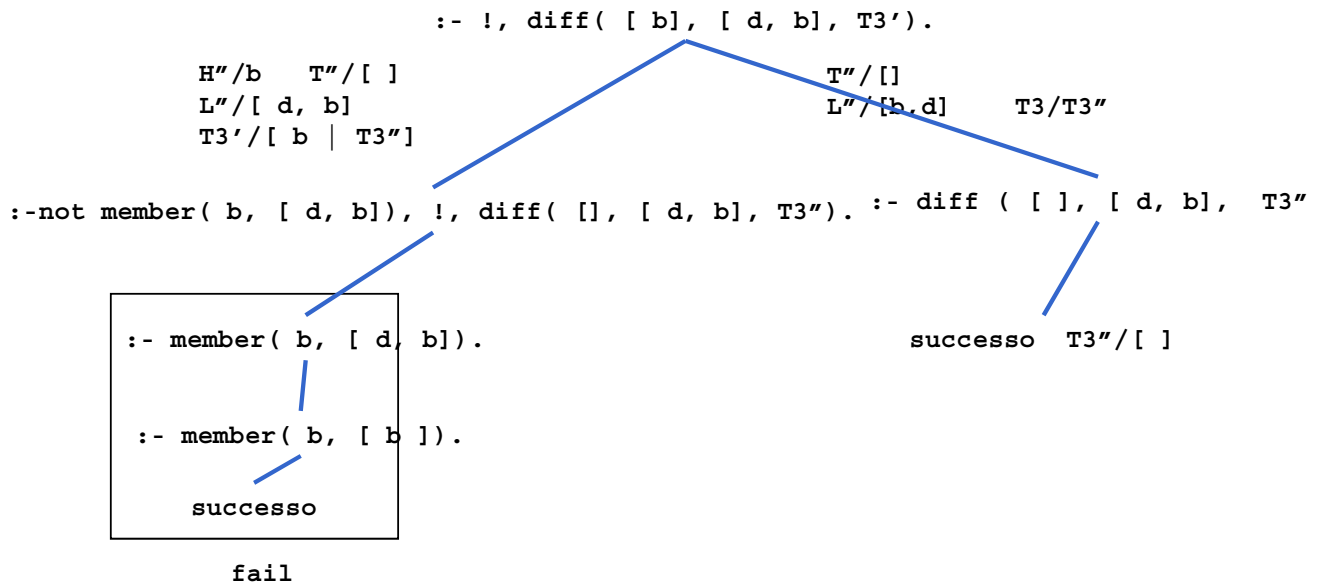
33

Soluzione Esercizio 6 – SLD & cut



34

Soluzione Esercizio 7 – SLDNF



37

Esercizio 8 – Compito del 5 / 11 / 2003

Si consideri il seguente programma Prolog:

```

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

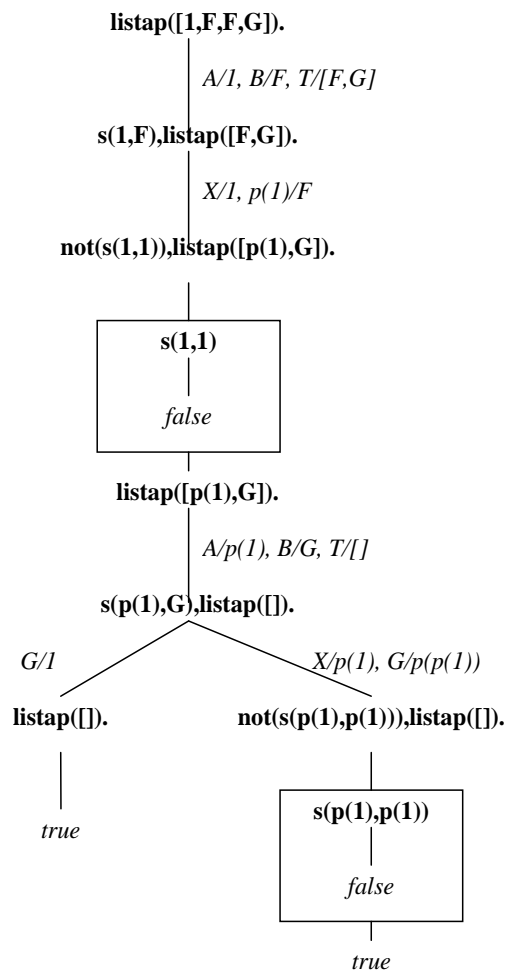
s( p( X ), X ).
s( X, p( X ) ) :-
    not( s( X, X ) ).
  
```

Si rappresenti l'albero SLDNF corrispondente al seguente goal:

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

38

Soluzione Esercizio 8 – Compito del 5/11/2003



39

Esercizio 9 – Compito del 20 / 12 / 2004

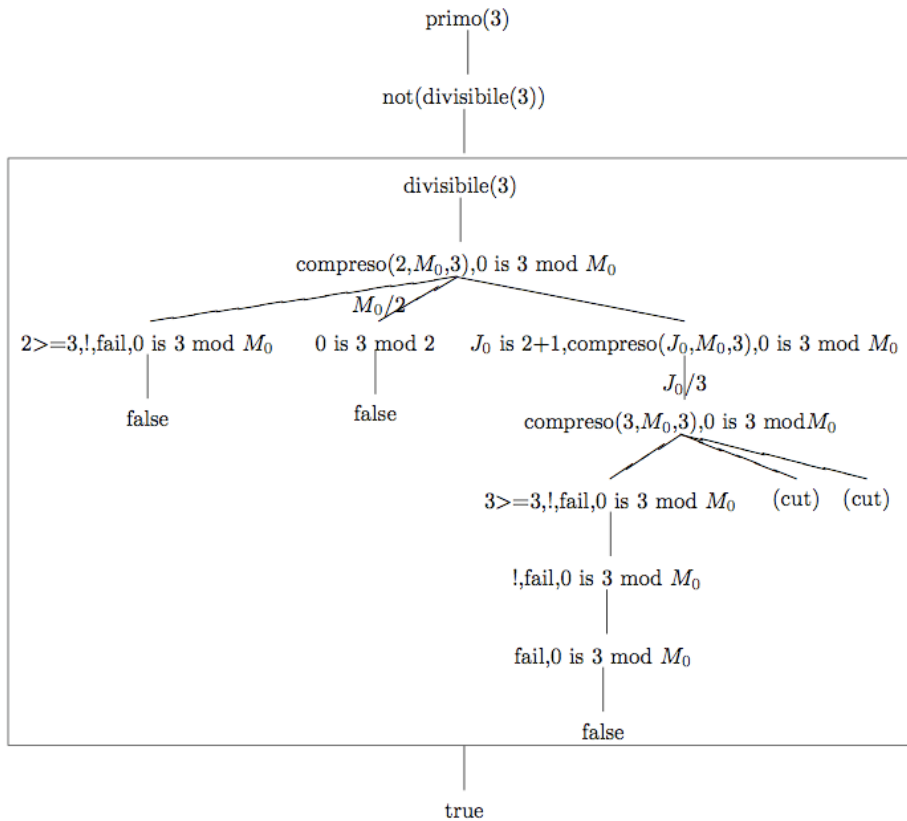
- Si consideri il seguente programma Prolog che calcola se un numero è primo (dove *mod* calcola il modulo, cioè il resto della divisione intera):

```
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).`

40



Esercizio 10 - 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.

