Lezione 12

PROLOG

Soluzioni esercizi

Sommatoria con ricorsione di coda

sumL(L,S) :- s3(L,0,S).

s3([], A, A).
s3([X|L], A, R) :- A1 is A+X, s3(L,A1,R).

Significato inteso di s3:
s3(L, A, R): R - A = sommatoria(L).

Esercizio: dimostare la verità delle clausole del programma, usando il significato inteso


```
minore([],L).
minore([X|A],[Y|B]) :- elementoMinore(X,Y).
minore([X|A],[Y|B]) :- X=Y, minore(A,B).
```

Il programma è aperto; a seconda degli elementi, si avrà un'implementazione del predicato aperto elementoMinore/2

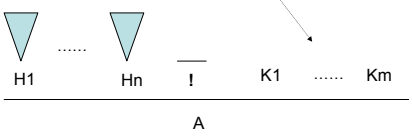
Il taglio

- Riconsiderando il programma precedente, si vede che è inutile fare backtraking se la clausola:
  minore([X,A],[Y|B]) :- elementoMinore(X,Y)
  ha avuto successo.
- Si evita backtraking con un cut:

```
minore([],L).
minore([X|A],[Y|B]) :- elementoMinore(X,Y), !.
minore([X|A],[Y|B]) :- X=Y, minore(A,B).
```

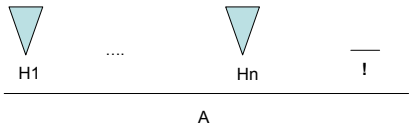

Semantica del cut con gli alberi di prova

Backtraking solo qui; la parte a sinistra di ! viene congelata e alla fine l'albero è staccato e nessun'altra clausola per A è considerata



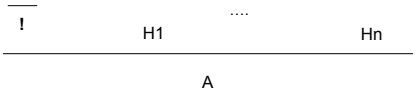
Caso particolare

Blocca l'albero.
L'intero albero viene staccato e nessun'altra clausola considerata



Caso particolare

Blocca l'atomo.
Terminato il backtraking, l'atomo A viene staccato e nessun'altra clausola per A considerata




```
minore([],L, si).
minore([X|A],[Y|B], si) :- X < Y.
minore([X|A],[Y|B], no) :- Y < X.
minore([X|A],[Y|B],R) :- X=Y, minore(A,B,R).
```

minore([1,1,3],[1,2,3,5],R)


```
minore([],L, si).
minore([X|A],[Y|B], si) :- X < Y.
minore([X|A],[Y|B], no) :- Y < X.
minore([X|A],[Y|B],R) :- X=Y, minore(A,B,R).
```

FAIL,
backtrack

1 < 1

minore([1,1,3],[1,2,3,5],si)


```
minore([],L, si).
minore([X|A],[Y|B], si) :- X < Y.
minore([X|A],[Y|B], no) :- Y < X.
minore([X|A],[Y|B],R) :- X=Y, minore(A,B,R).
```

minore([1,1,3],[1,2,3,5],R)


```
minore([],L, si).
minore([X|A],[Y|B], si) :- X < Y.
minore([X|A],[Y|B], no) :- Y < X.
minore([X|A],[Y|B],R) :- X=Y, minore(A,B,R).
```

FAIL, backtrack

1 < 1

minore([1,1,3],[1,2,3,5],no)


```
minore([],L, si).
minore([X|A],[Y|B], si) :- X < Y.
minore([X|A],[Y|B], no) :- Y < X.
minore([X|A],[Y|B],R) :- X=Y, minore(A,B,R).
```

minore([1,1,3],[1,2,3,5],R)

```
minore([],L, si).
minore([X|A],[Y|B], si) :- X < Y.
minore([X|A],[Y|B], no) :- Y < X.
minore([X|A],[Y|B],R) :- X=Y, minore(A,B,R).
```

1=1 minore([1,3],[2,3,5],R)

minore([1,1,3],[1,2,3,5],R)


```
minore([],L, si).
minore([X|A],[Y|B], si) :- X < Y.
minore([X|A],[Y|B], no) :- Y < X.
minore([X|A],[Y|B],R) :- X=Y, minore(A,B,R).
```

1=1 minore([1,3],[2,3,5],R)

minore([1,1,3],[1,2,3,5],R)


```
minore([],L, si).
minore([X|A],[Y|B], si) :- X < Y.
minore([X|A],[Y|B], no) :- Y < X.
minore([X|A],[Y|B],R) :- X=Y, minore(A,B,R).
```

Ricerca altra soluzione, Backtrack facendo finta che 1<2 sia “fallito”

1 < 2

1=1 minore([1,3],[2,3,5],si)

minore([1,1,3],[1,2,3,5],si)


```
minore([],L, si).
minore([X|A],[Y|B], si) :- X < Y.
minore([X|A],[Y|B], no) :- Y < X.
minore([X|A],[Y|B],R) :- X=Y, minore(A,B,R).
```

1=1 minore([1,3],[2,3,5],R)

minore([1,1,3],[1,2,3,5],R)


```
minore([],L, si).
minore([X|A],[Y|B], si) :- X < Y.
minore([X|A],[Y|B], no) :- Y < X.
minore([X|A],[Y|B],R) :- X=Y, minore(A,B,R).
```

fail

2 < 1

1=1 minore([1,3],[2,3,5],no)

minore([1,1,3],[1,2,3,5],no)


```
minore([],L, si).
minore([X|A],[Y|B], si) :- X < Y.
minore([X|A],[Y|B], no) :- Y < X.
minore([X|A],[Y|B],R) :- X=Y, minore(A,B,R).
```

Fail, nessuna altra clausola applicabile ai vari livelli di backtraking, NO altre soluzioni

1 = 2 minore([3],[3,5],R)

1=1 minore([1,3],[2,3,5],R)

minore([1,1,3],[1,2,3,5],R)

CON CUT


```
minore([],L, si).
minore([X|A],[Y|B], si) :- X < Y, !.
minore([X|A],[Y|B], no) :- Y < X, !.
minore([X|A],[Y|B],R) :- X=Y, minore(A,B,R).
```

minore([1,1,3],[1,2,3,5],R)


```
minore([],L, si).
minore([X|A],[Y|B], si) :- X < Y, !.
minore([X|A],[Y|B], no) :- Y < X, !.
minore([X|A],[Y|B],R) :- X=Y, minore(A,B,R).
```

FAIL,
backtrack

1 < 1

minore([1,1,3],[1,2,3,5],si)


```
minore([],L, si).
minore([X|A],[Y|B], si) :- X < Y, !.
minore([X|A],[Y|B], no) :- Y < X, !.
minore([X|A],[Y|B],R) :- X=Y, minore(A,B,R).
```

minore([1,1,3],[1,2,3,5],R)


```
minore([],L, si).
minore([X|A],[Y|B], si) :- X < Y, !.
minore([X|A],[Y|B], no) :- Y < X, !.
minore([X|A],[Y|B],R) :- X=Y, minore(A,B,R).
```

FAIL, backtrack

1 < 1

minore([1,1,3],[1,2,3,5],no)


```
minore([],L, si).
minore([X|A],[Y|B], si) :- X < Y, !.
minore([X|A],[Y|B], no) :- Y < X, !.
minore([X|A],[Y|B],R) :- X=Y, minore(A,B,R).
```

minore([1,1,3],[1,2,3,5],R)

```
minore([],L, si).
minore([X|A],[Y|B], si) :- X < Y, !.
minore([X|A],[Y|B], no) :- Y < X, !.
minore([X|A],[Y|B],R) :- X=Y, minore(A,B,R).
```

1=1 minore([1,3],[2,3,5],R)

minore([1,1,3],[1,2,3,5],R)


```
minore([],L, si).
minore([X|A],[Y|B], si) :- X < Y, !.
minore([X|A],[Y|B], no) :- Y < X, !.
minore([X|A],[Y|B],R) :- X=Y, minore(A,B,R).
```

1=1 minore([1,3],[2,3,5],R)

minore([1,1,3],[1,2,3,5],R)


```
minore([],L, si).
minore([X|A],[Y|B], si) :- X < Y, !.
minore([X|A],[Y|B], no) :- Y < X, !.
minore([X|A],[Y|B],R) :- X=Y, minore(A,B,R).
```

Ricerca altra soluzione, Backtrack facendo finta che 1<2 sia "fallito"; per ! l'albero intero con radice sotto al ! Fallisce e nessuna clausola per esso è più considerata

1 < 2 !

1=1 minore([1,3],[2,3,5],si)

minore([1,1,3],[1,2,3,5],si)


```
minore([],L, si).
minore([X|A],[Y|B], si) :- X < Y, !.
minore([X|A],[Y|B], no) :- Y < X, !.
minore([X|A],[Y|B],R) :- X=Y, minore(A,B,R).
```

Finite le clausole, fallito STOP

minore([1,1,3],[1,2,3,5],R)