FONDAMENTI DI INTELLIGENZA ARTIFICIALE – 1° parte (6 CFU)
14 Settembre 2011 – Tempo a disposizione: 2 h – Risultato: 32/32 punti

Esercizio 1 (6 punti)
In logica dei predicati del primo ordine, vale la seguente conseguenza logica:

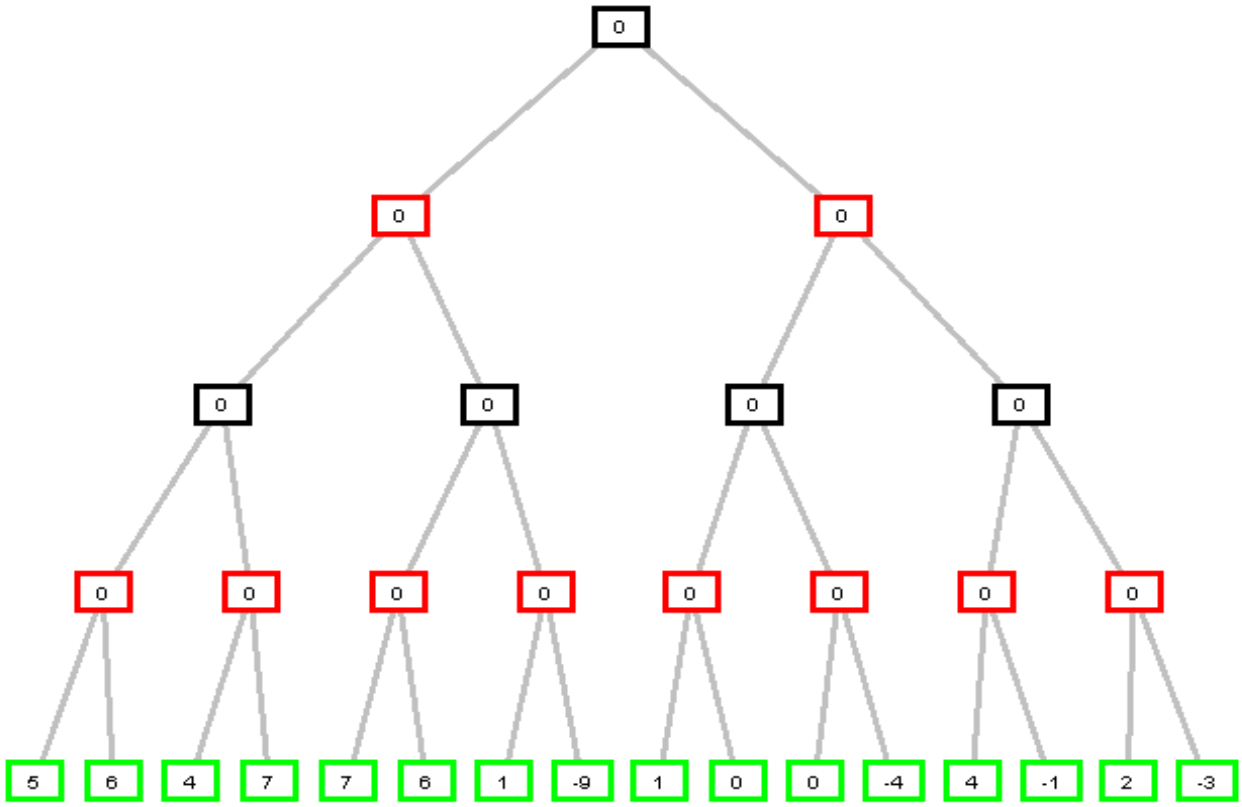
$$\exists x\big(b(x) \wedge e(x)\big), \exists x\left(b(x) \wedge \forall y\big(e(y) \rightarrow \neg a(x,y)\big)\right) \models \exists x\left(b(x) \wedge \neg\forall y\big(b(y) \rightarrow a(y,x)\big)\right)$$

Ricordando che vale:

$$T \models F \quad \text{se e solo se} \quad (T \cup \neg F)^{clausal\ form} \vdash_{RES} \square$$

si rappresentino la teoria e la formula negata come clausole e si applichi la risoluzione in refutazione per derivare la clausola vuota.

Esercizio 2 (6 punti)
Si consideri il seguente albero di gioco in cui la valutazione dei nodi terminali è dal punto di vista del primo giocatore (che è *Max*). Si mostri come gli algoritmi *min-max* e *alfa-beta* risolvono il problema.



Esercizio 3 (7 punti)
Dato il seguente programma Prolog:

```
lds(L1, L2, 0) :- !, L1 = L2.
lds([X|T], [X|TR], D) :- lds(T, TR, D).
lds([X|T], [Rif|TR], D) :- assegna(X), not(X=Rif), E is D-1, lds(T, TR, E).
assegna(1).
assegna(2).
```

Si mostri l'albero di derivazione SLD (fino alla 1ª soluzione, indicando eventuali cut) per la query:

```
?- lds([A, 2, B], [2, 1, 1], 1).
```

Esercizio 4 (5 punti)

Si scriva un programma Prolog `palindroma(X,Y)` che è vero se X è una lista palindroma di elementi (ground) e Y è l'insieme ordinato privo di ripetizioni dei suoi elementi.

Es.:
```
?- palindroma([1,2,3,2,1], [1,2,3]).
yes
?- palindroma([1,2,3,2,1], Y).
yes Y=[1,2,3]
?- palindroma([1,2,3,2,1], [3,1,2]).
no
?- palindroma([1,2,2,1], [1,2,4]).
no
?- palindroma([1,2,2,1], [1,1,2]).
no
?- palindroma([1,2,3,5,4], Y).
no
```

Esercizio 5 (8 punti)

Un gruppo di studenti ha terminato lo svolgimento di una serie di sei demo assegnate dal docente ma non c'è stato abbastanza tempo per pianificarne adeguatamente la presentazione in modo da ottenere un buon voto. Gli studenti pensano quindi di utilizzare un CSP per modellare il problema e pianificare la presentazione. Si indichino i sei slot temporali in cui presentare le demo con i numeri {1,2,3,4,5,6}), e le demo con le lettere {A,B,C,D,E,F}. Si hanno i seguenti vincoli:

- In ogni slot può essere presentata una sola demo (ogni demo occupa un solo slot),
- Nel primo e nell'ultimo slot deve essere presentata una demo con effetti speciali (le demo con effetti speciali sono tre ed indicate dalle lettere {A,C,D}),
- Demo con studenti in comune non devono essere successive onde evitare che uno studente debba presentare più demo consecutivamente (le demo con studenti in comune sono rappresentate nella tabella che segue).

	A	B	C	D	E	F
A	–		shares			
B		–	shares		shares	
C	shares	shares	–	shares		
D			shares	–		shares
E		shares			–	shares
F				shares	shares	–

Nel rappresentare il CSP, si usino le variabili $X_1, \dots, X_6$ per rappresentare i sei slot temporali e sia $[A, \dots, F]$ il dominio di ciascuna di esse. Pertanto i vincoli risultano essere:

$$X_1 \neq X_2 \neq X_3 \neq X_4 \neq X_5 \neq X_6 \ (binari)$$
$$X_1 = A \vee X_1 = C \vee X_1 = D \ (unari)$$
$$X_6 = A \vee X_6 = C \vee X_6 = D \ (unari)$$

e la tabella precedente si traduce in:

$\forall i, 1 \leq i \leq 6: X_i = A \rightarrow X_{i+1} \neq C$ $\quad$ $\forall i, 1 \leq i \leq 6: X_i = B \rightarrow X_{i+1} \neq E$

$\forall i, 1 \leq i \leq 6: X_i = C \rightarrow X_{i+1} \neq B \wedge X_{i+1} \neq D$ $\quad$ $\forall i, 1 \leq i \leq 6: X_i = D \rightarrow X_{i+1} \neq C \wedge X_{i+1} \neq F$

$\forall i, 1 \leq i \leq 6: X_i = E \rightarrow X_{i+1} \neq B \wedge X_{i+1} \neq F$ $\quad$ $\forall i, 1 \leq i \leq 6: X_i = F \rightarrow X_{i+1} \neq D \wedge X_{i+1} \neq E$

Trovare la prima soluzione applicando *forward checking*. Si considerino le variabili da istanziare (gli slot temporali) in ordine crescente di indice e i valori da provare (le demo) in ordine alfabetico.

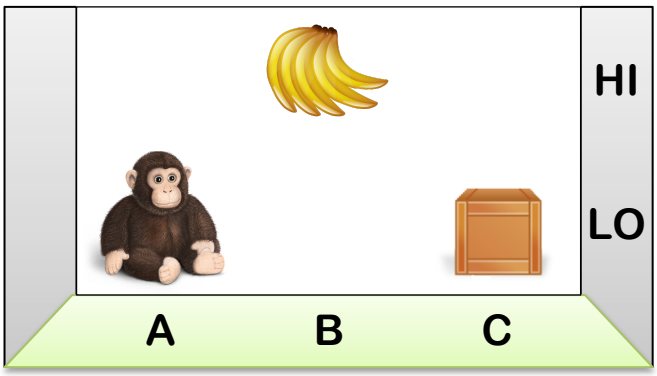
VOTO:

- *Esame da 6 CFU, il voto è determinato da questa I parte*
- *Esame da 9 CFU, è la media pesata della I parte (che vale 2/3) e della II (che vale 1/3) ovvero il voto finale è dato da:* $\frac{2 \times voto\ I\ parte + voto\ II\ parte}{3}$ *e varia quindi da 0 ad un massimo di 32 (equivalente alla lode).*

FONDAMENTI DI INTELLIGENZA ARTIFICIALE – 2° parte (3 CFU)
14 Settembre 2011 – Tempo a disposizione: 45 min – Risultato: 32/32 punti

Esercizio 6 (10 punti)
Facendo riferimento al frammento di logica descrittiva visto a lezione, si descrivano gli *operatori di costruzione dei concetti* introdotti (ALL, EXISTS, FILLS e AND) fornendo un breve esempio d'uso.

Esercizio 7 (10 punti)



Una scimmia si trova in una stanza nella posizione A (vedi figura). Nella posizione C c'è una cassa. La scimmia vuole alcune banane che penzolano dal soffitto in posizione B, ma le occorre spostare la cassa e salirci sopra.
Supponendo di usare i seguenti predicati per descrivere lo *stato iniziale*:

```
monkey_at(a), monkey_level(lo),
bananas_at(b), crate_at(c)
```

e il *goal*:

```
have(bananas)
```

Si completi la descrizione in linguaggio STRIPS delle seguenti azioni:

```
MOVE(X,Y)
// La scimmia si sposta a terra da X ad Y
Preconditions:  ...
Postconditions: ...

DRAG(X,Y)
// La scimmia trascina la cassa da X ad Y
Preconditions:  ...
Postconditions: ...

CLIMBUP(Loc)
// La scimmia sale sulla cassa in posizione Loc e si trova in alto
Preconditions:  ...
Postconditions: ...

CLIMBDOWN(Loc)
// La scimmia scende dalla cassa in posizione Loc e si trova in basso
Preconditions:  ...
Postconditions: ...

GRAB(Loc)
// La scimmia prende le banane in posizione Loc
Preconditions:  ...
Postconditions: ...
```

Si descrivano poi brevemente a parole i passi con cui l'algoritmo STRIPS trova la soluzione a tale problema di pianificazione.

Esercizio 8 (12 punti)

Si scriva un meta-interprete che considera già risolte (senza dimostrazione) le chiamate ai predicati appartenenti a una lista fornita in ingresso mediate coppie *[funtore, arità]*. In uscita il meta-interprete dovrà restituire la lista di tali chiamate (assumendole vere e non dimostrandole).
Ad esempio, date le clausole e la chiamata che seguono:

```
p(X) :- q(X), r(X).                    ?- solve(p(X), [[r,1], [g,3]], L).
q(a).
```

si otterrà come risultato (essendo `r/1` e `g/3` da considerarsi veri):

```
yes X=a, L=[r(a)]
```

VOTO:

- *Esame da 3 CFU, il voto è determinato da questa 2° parte*
- *Esame da 9 CFU, è la media pesata della 1° parte (che vale 2/3) e della 2° (che vale 1/3) ovvero il voto finale è dato da:* $\frac{2\times voto\ I\ parte + voto\ II\ parte}{3}$ *e varia quindi da 0 ad un massimo di 32 (equivalente alla lode).*

FONDAMENTI DI INTELLIGENZA ARTIFICIALE – 1° parte
14 Settembre 2011 – Soluzioni

Esercizio 1

$$\exists x(b(x) \wedge e(x)),\ \exists x(b(x) \wedge \forall y(e(y) \rightarrow \neg a(x,y))) \models \exists x(b(x) \wedge \neg \forall y(b(y) \rightarrow a(y,x)))$$

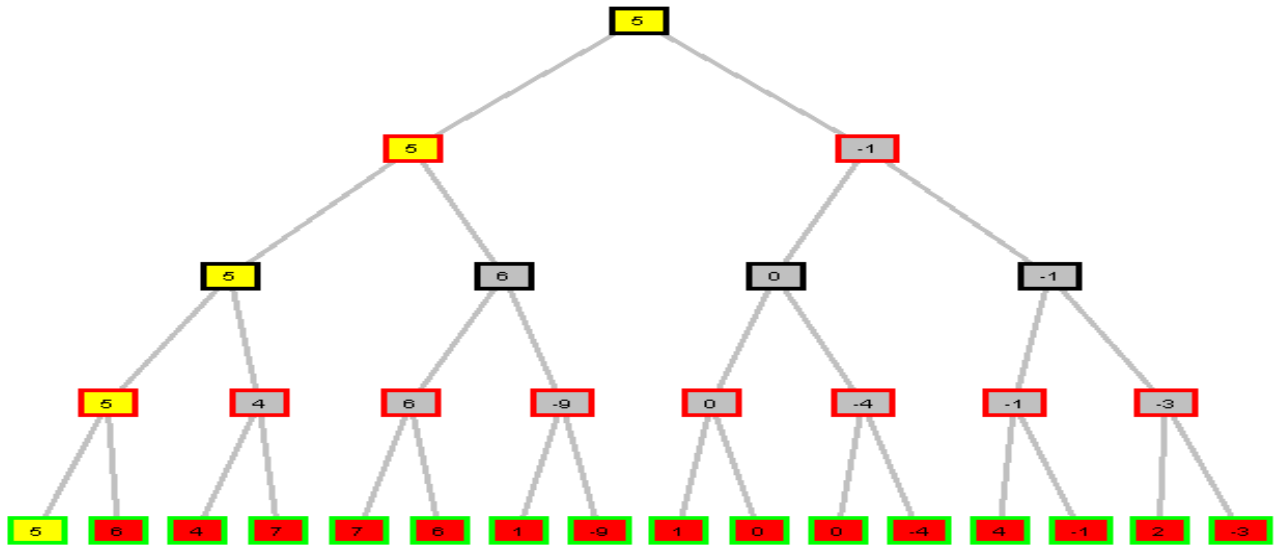
se e solo se

$$b(c_0),\ e(c_0),\ b(c_1),\ \neg e(y) \vee \neg a(c_1,y),\ \neg b(x) \vee \neg b(y) \vee a(y,x) \vdash_{RES} \square$$

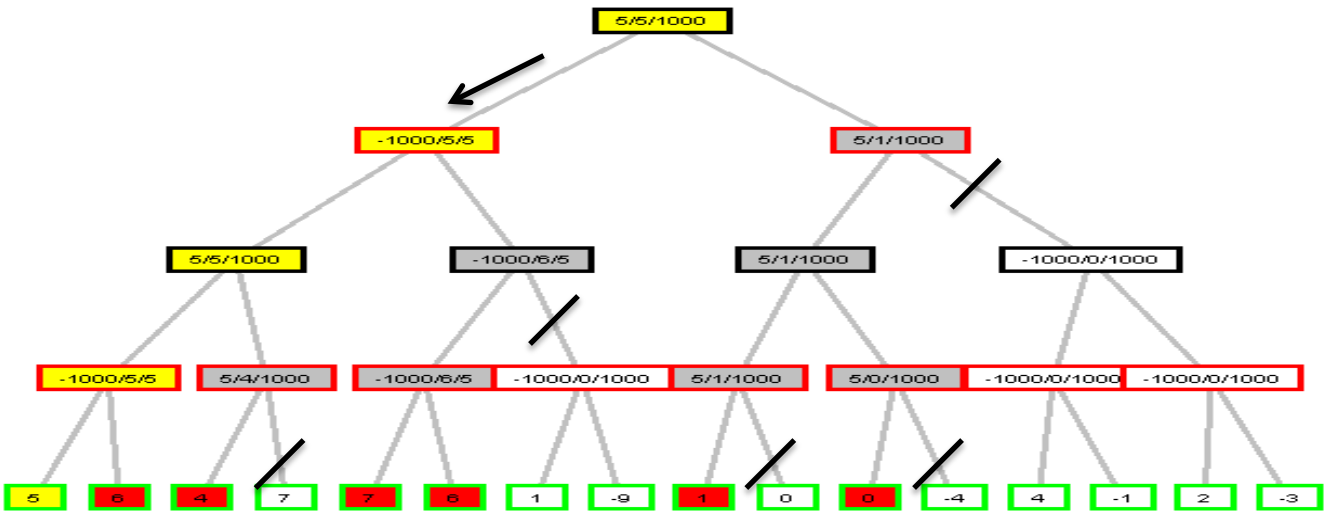
1.	$b(c_0)$	*in* S
2.	$e(c_0)$	*in* S
3.	$b(c_1)$	*in* S
4.	$\neg e(y) \vee \neg a(c_1,y)$	*in* S
5.	$\neg b(x) \vee \neg b(y) \vee a(y,x)$	*in* S
6.	$\neg b(y) \vee a(y,c_0)$	$RES(1,5), \{c_0/x\}$
7.	$a(c_1,c_0)$	$RES(3,6), \{c_1/y\}$
8.	$\neg e(c_0)$	$RES(4,7), \{c_0/y\}$
9.	$\square$	$RES(2,8)$

Esercizio 2

min-max:

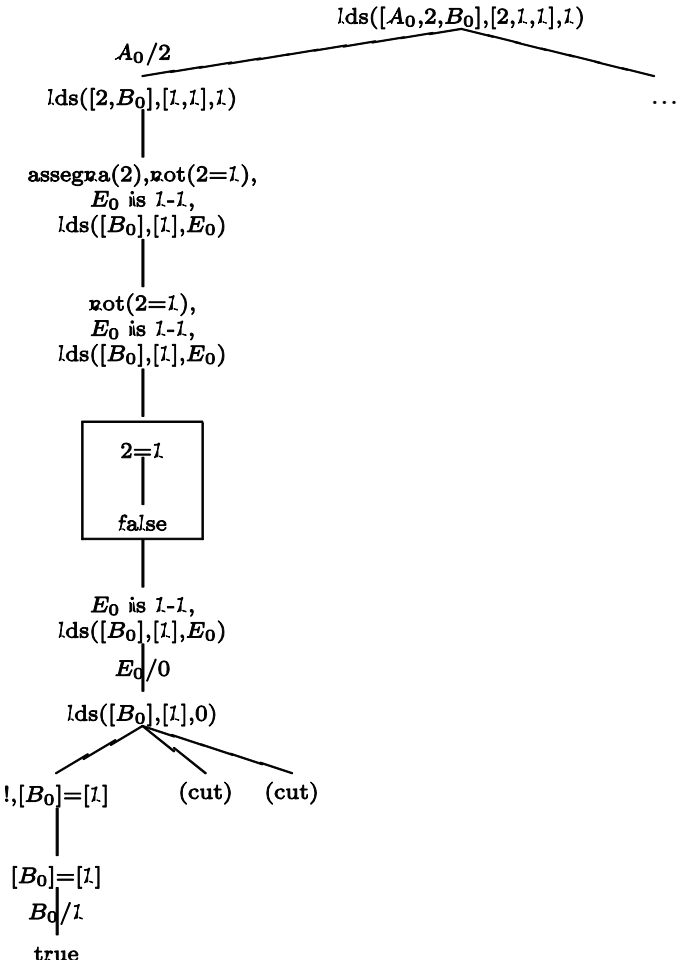


alfa-beta:



I nodi non colorati sono quelli che vengono tagliati nell'algoritmo alfa-beta.

Esercizio 3

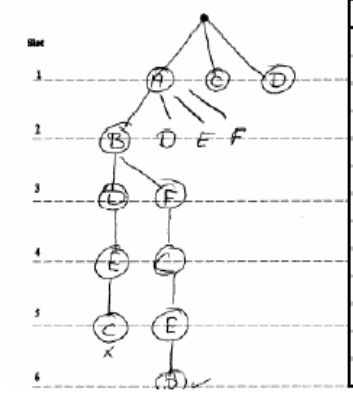


Esercizio 4

```
palindroma([],[]) :- !.
palindroma([X],[X]) :- ground(X), !.
palindroma([H|T],[H|T1]) :- ground(H), reverse(T,[H|L]), !,
                            reverse(L,L1), palindroma(L1,T1).
```

Esercizio 5

Si mostra in figura l'albero di ricerca e in tabella l'applicazione del FC dopo ogni singola istanziazione di variabile:



SLOT	1						2						3						4						5						6					
INITIAL DOMAIN	A	B	C	D	E	F	A	B	C	D	E	F	A	B	C	D	E	F	A	B	C	D	E	F	A	B	C	D	E	F	A	B	C	D	E	F
show stoppers constrain slots 1 and 6	A		C	D			A	B	C	D	E	F	A	B	C	D	E	F	A	B	C	D	E	F	A	B	C	D	E	F	A		C	D		
1=A	**A**							B		D	E	F		B	C	D	E	F		B	C	D	E	F		B	C	D	E	F			C	D		
2=B	**A**						**B**				D		F			C	D	E	F			C	D	E	F			C	D	E	F		C	D		
3=D	**A**						**B**				**D**							E				C			E	F			C						
4=E	**A**						**B**				**D**							**E**				C							C						
5=C	**A**						**B**				**D**							**E**				**C**														
5≠C	**A**						**B**				**D**							**E**											C						
4≠E	**A**						**B**				**D**											C		E	F			C						
3≠D	**A**						**B**						D		F		C	D	E	F		C	D	E	F			C	D						
3=F	**A**						**B**						**F**			C							D	E			C	D							
4=C	**A**						**B**						**F**			**C**								E				D							
5=E	**A**						**B**						**F**			**C**								**E**				D							
6=D	**A**						**B**						**F**			**C**								**E**				**D**							

FONDAMENTI DI INTELLIGENZA ARTIFICIALE – 2° parte
14 Settembre 2011 – Soluzioni

Esercizio 6

- [ALL r d], dove r è una relazione e d un concetto. Indica tutti gli individui che, se sono in relazione r, allora sono in relazione r con individui appartenenti al concetto d.
- [EXISTS n r], dove n è un intero positivo, e d un concetto. Indica tutti gli individui che sono in almeno n relazioni r con n individui (distinti)
- [FILLS r c], dove r è una relazione e c una costante (un individuo). Indica tutti gli individui che sono in relazione r con c.
- [AND d_1 ... d_n], dove d_1 ... d_n sono concetti. Indica tutti gli individui che sono istanze di tutti i concetti d_1 ... d_n (intersezione).

Esercizio 7

Nota: le variabili `X` ed `Y` dei predicati `MOVE` e `DRAG` devono essere diverse tra di loro.

```
MOVE(X,Y)
// La scimmia si sposta a terra da X ad Y
Preconditions:  monkey_at(X), X≠Y, monkey_level(lo)
Postconditions: ¬monkey_at(X), monkey_at(Y)

DRAG(X,Y)
// La scimmia trascina la cassa da X ad Y
Preconditions:  crate_at(X), monkey_at(X), X≠Y, monkey_level(lo)
Postconditions: ¬crate_at(X), ¬monkey_at(X), crate_at(Y), monkey_at(Y)

CLIMBUP(Loc)
// La scimmia sale sulla cassa in posizione Loc
Preconditions:  monkey_at(Loc), crate_at(Loc), monkey_level(lo)
Postconditions: ¬monkey_level(lo), monkey_level(hi)

CLIMBDOWN(Loc)
// La scimmia scende dalla cassa in posizione Loc
Preconditions:  monkey_at(Loc), crate_at(Loc), monkey_level(hi)
Postconditions: ¬monkey_level(hi), monkey_level(lo)

GRAB(Loc)
// La scimmia prende le banane in posizione Loc
Preconditions:  monkey_at(Loc), monkey_level(hi), bananas_at(Loc)
Postconditions: have(bananas)
```

Esercizio 8

```
solve(true, _, []).
solve((A, B), Lin, Lout) :-
      solve (A,Lin,L1),
      solve (B, Lin, L2),
      append (L1,L2, Lout).
solve(A, Lin, [A]) :-
      functor(A, F, N),
      member([F,N], Lin), !.
solve(A, Lin, Lout) :-
      clause(A,Body),
      solve(Body, Lin, Lout).
```