FONDAMENTI DI INTELLIGENZA ARTIFICIALE

Prof. Paola Mello- 12 Luglio 2018 – Tempo a disposizione: 2 h – Risultato: 32/32 punti

Esercizio 1 (punti 6)

Modellare in logica del I ordine le seguenti frasi in linguaggio naturale.

1. Mariani&Figli è uno stabilimento;
2. Fred è il cane da guardia della Mariani&Figli;
3. Esiste un ladro alla Mariani&Figli, e Fred non ha abbaiato a quel ladro;
4. Se un cane da guardia di un luogo non abbaia a un ladro di quel luogo, allora quella persona è il padrone del cane.

Si dimostri mediante il principio di risoluzione che esiste un ladro alla Mariani&Figli, e che questi è il padrone di Fred.

Si utilizzi un linguaggio logico con i predicati seguenti:

`cane(X,Y)`: X è un cane da guardia di Y;

`stab(X)`: X è uno stabilimento;

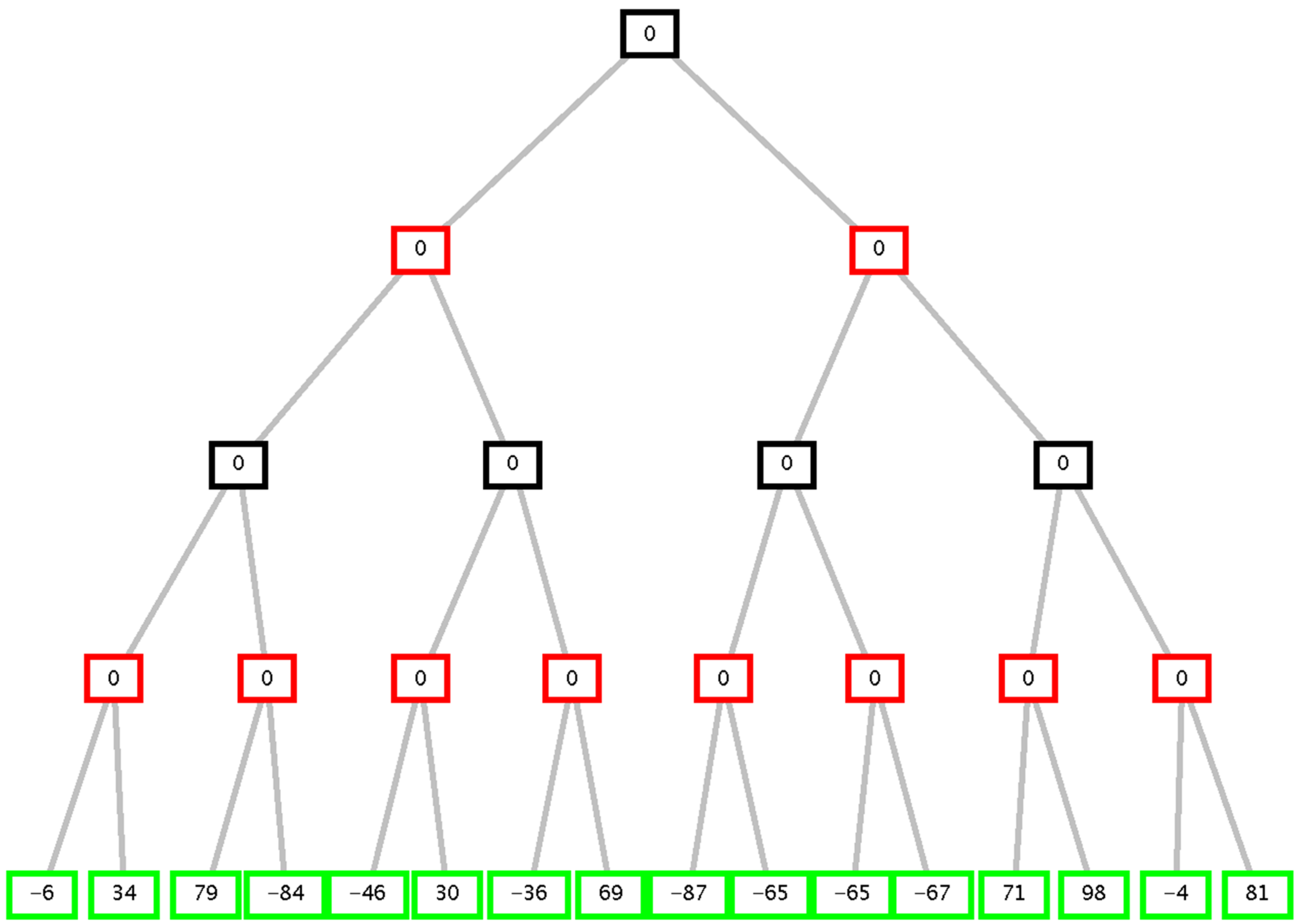
`ladro(X,Y)`: X è un ladro nel luogo Y;

`abbaia(X,Y)`: X abbaia a Y;

`padrone(X,Y)`: X è padrone di Y.

Esercizio 2 (punti 4)

Si consideri il seguente albero di gioco in cui la valutazione dei nodi terminali è dal punto di vista del primo giocatore (*MAX*). Si mostri come l'algoritmo *min-max* e l'algoritmo *alfa-beta* risolvono il problema e la mossa selezionata dal giocatore.



Esercizio 3 (punti 6)

Dato il seguente programma Prolog:

```
last([El], [], El) :- !.
last([El|Tail], [El|Rest], Last) :-
        last(Tail, Rest, Last).
```

dove il predicato `last(List,Rest,LastEl)` restituisce in `LastEl` il suo ultimo elemento e in `Rest` il resto della lista `List` senza `LastEl`, si disegni l'albero SLD-NF del goal:

```
?- last([4, a, 2, 2], R, El),
   last(R, Rr, El).
```

Esercizio 4 (punti 5)

Si scriva un predicato Prolog `intersection(L1, L2, El, L3)` che date le liste di atomi `L1` e `L2` produce la lista `L3` degli elementi di L1 uguali all'atomo `El` e che compaiono anche in `L2`.

Esempio:

```
?- intersection([a,b,c,a],[1,2,a,b,a,a], a, L3).
Yes L3=[a,a]
?- intersection([a,b,c],[1,2,a,b,a,a], a, L3).
Yes L3=[a]
?- intersection([a,b,c],[1,2,a,b,a,a], 2, L3).
Yes L3=[]
?- intersection([a,b,c],[1,2,a,b,a,a], w, L3).
Yes L3=[]
```

Si supponga di avere a disposizione il predicato `member(E,L)` per verificare se `E` compare in `L`.

Esercizio 5 (punti 7)

Si supponga di avere un robot che deve svolgere alcune attività: A, B, C, D ed E. Ogni attività può essere eseguita al tempo 1, 2, 3, o 4. Si supponga che le attività debbano rispettare i seguenti vincoli di tempo:

$\{(B\neq3), (C\neq2), (A\neq B), (B\neq C), (C<D), (A=D), (E<A), (E<B), (E<C), (E<D), (B\neq D)\}$

Si modelli il problema come un CSP, determinando variabili e domini.
Si disegni poi il corrispondente Constraint Graph e si applichi la NODE-consistency.
Successivamente si mostrino i domini ridotti dopo l'applicazione dell'algoritmo di ARC-Consistency al Constraint graph in esame.
Si determini poi la soluzione, se esiste.

Esercizio 6 (punti 4)

Si descriva il trattamento della negazione in Prolog, si spieghi la differenza rispetto alla negazione classica nonché i limiti e problemi di utilizzo in alcuni casi. Se ne mostri poi l'implementazione nel linguaggio Prolog stesso.

FONDAMENTI DI INTELLIGENZA ARTIFICIALE

Luglio 2018 – Soluzioni

Esercizio 1

Modellare in logica del I ordine le seguenti frasi in linguaggio naturale.

1. Mariani&Figli è uno stabilimento;
   stab(mariani&figli)
2. Fred è il cane da guardia della Mariani&Figli;
   cane(fred, mariani&figli)
3. Esiste un ladro alla Mariani&Figli, e Fred non ha abbaiato a quel ladro;
   ∃X ladro(X, mariani&figli) ∧ ¬abbaia(fred, X).
4. Se un cane da guardia di un luogo non abbaia a un ladro di quel luogo, allora quella persona è il padrone del cane.
   ∀X ∀Y ∀Z cane(X,Y) ∧ ladro(Z,Y) ∧ ¬abbaia(X,Z) ⇒ padrone(Z,X)

Goal: esiste un ladro alla Mariani&Figli, e questi è il padrone di Fred
∃X padrone(X,fred) ∧ ladro(X,mariani&figli).

Trasformazione in Clausole:

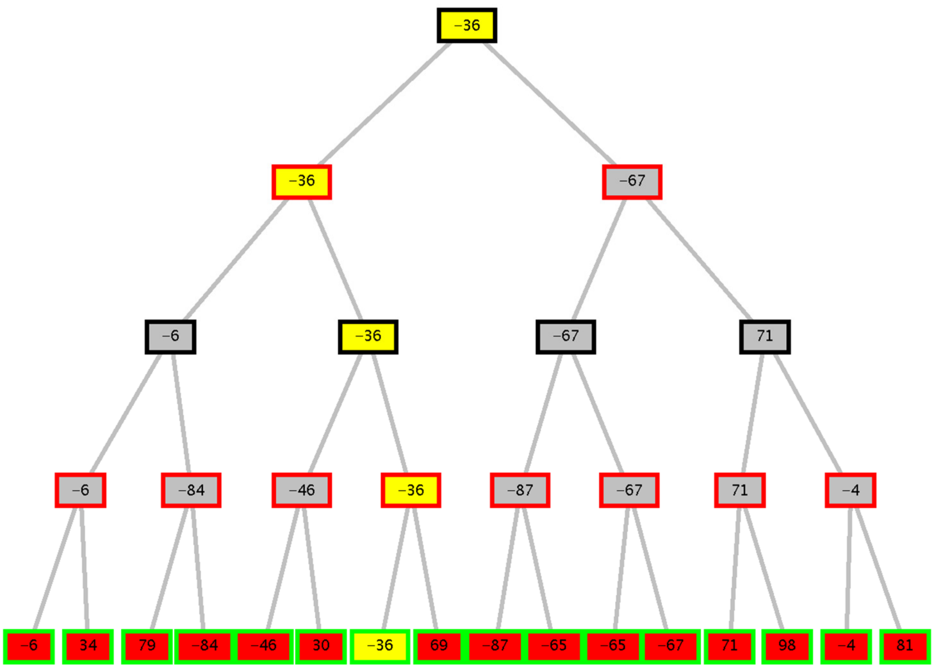
C1:	stab(mariani&figli)
C2:	cane(fred, mariani&figli)
C3a:	ladro(l1, mariani&figli) **l1 costante di Skolem**
C3b:	¬abbaia(fred, l1).
C4:	¬cane(X,Y) ∨ ¬ladro(Z,Y) ∨ abbaia(X,Z) ∨ padrone(Z,X)
GNeg:	¬ladro(X,mariani&figli) ∨ ¬padrone(X,fred)

Risoluzione:

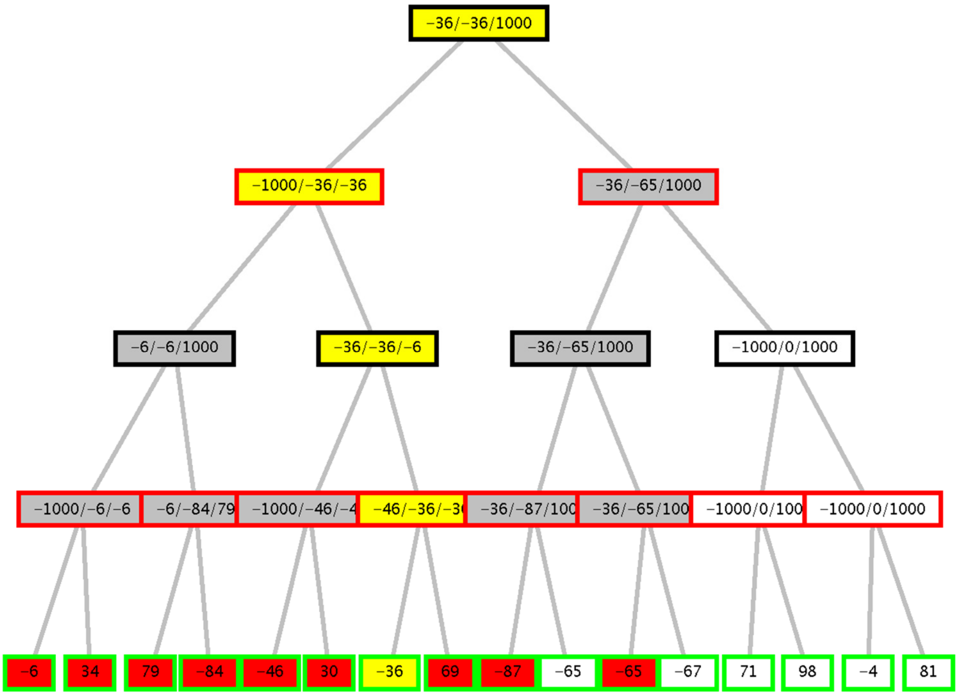
C5: GNeg + C3a:	¬padrone(l1, fred)
C6: C5 + C4:	¬cane(fred,Y) ∨ ¬ladro(l1,Y) ∨ abbaia(fred,l1)
C7: C6 + C2:	¬ladro(l1,mariani) ∨ abbaia(fred,l1)
C8: C7 + C3a:	abbaia(fred,l1)
C9: C8 + C3b:	contraddizione.

Esercizio 2

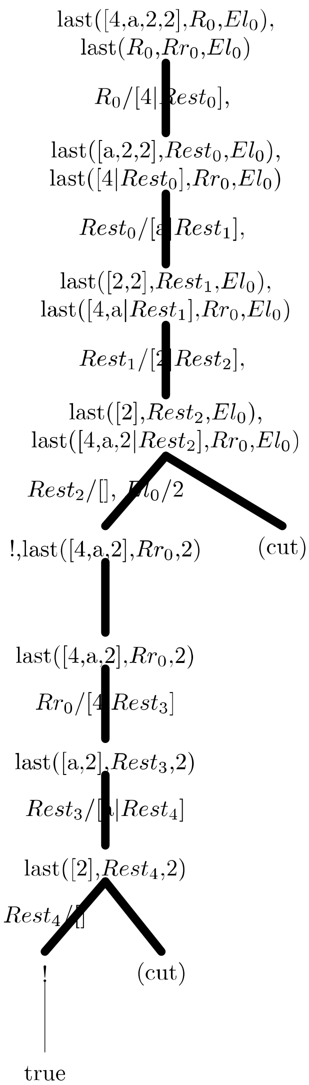
Min-Max:



Alfa-beta: I nodi che portano alla soluzione sono in giallo, quelli tagliati in bianco.



Esercizio 3



Esercizio 4

```
intersection([], _, _, []):-!.
intersection(_, [] , _, []):-!.  NOTA: questo fatto può essere omesso
intersection([H|T], L2, H, [H|T2]) :-
     member(H, L2),
     !,
     intersection(T, L2, H, T2).
intersection([_|T], L2, E, T2) :-
     intersection(T, L2, E, T2).
```

Esercizio 5

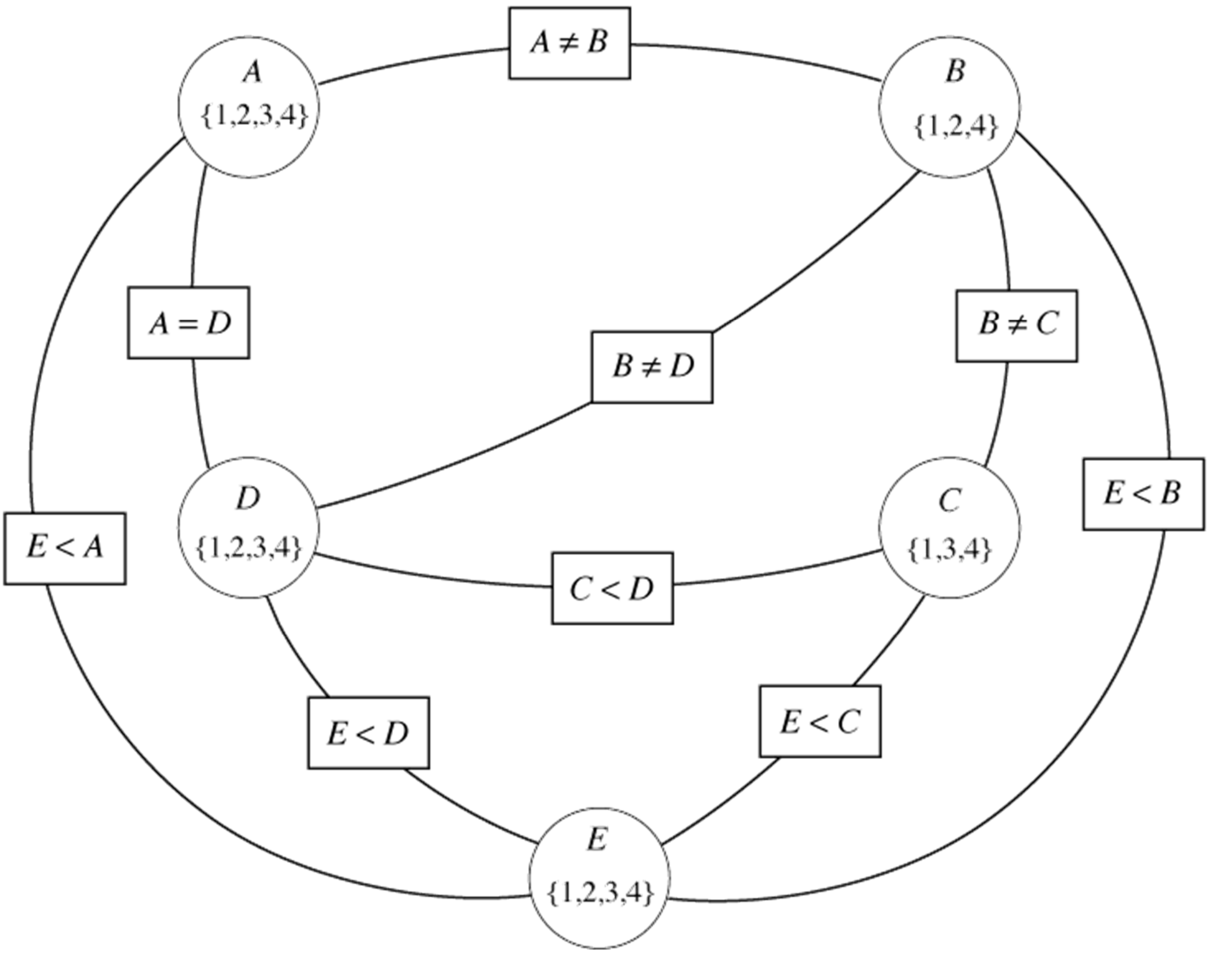
Domini:
dom(A)={1,2,3,4}
dom(B)={1,2,3,4}
dom(C)={1,2,3,4}
dom(D)={1,2,3,4}
dom(E)={1,2,3,4}

Constraints:

{(B≠3), (C≠2), (A≠B), (B≠C), (C<D), (A=D),

(E<A), (E<B), (E<C), (E<D), (B≠D)}

Grafo, già node consistent: 3 è stato rimosso dal dominio di B and 2 dal dominio di C.



Algoritmo di ARC-Consistency AC3 (si vedano le slides del corso).

Supponiamo che l'arco ⟨D, C <D⟩ sia considerato per primo. L'arco non è coerente con l'arco perché D = 1 non è coerente con alcun valore nel dominio di C, quindi 1 viene eliminato dal dominio di D che diventa {2,3,4}. Supponiamo ora di considerare l’arco ⟨C, E <C⟩ Il dominio di C viene ridotto a {3,4} e l'arco ⟨D, C <D⟩ dovrà essere riconsiderato.
Supponiamo che l'arco ⟨D, C <D⟩ sia il prossimo; quindi il dominio di D viene ulteriormente ridotto al singleton {4}. L’ arco ⟨C, C <D⟩riduce il dominio di C a {3}. L’ arco ⟨A, A = D⟩ riduce il dominio di A a {4}. L'elaborazione di ⟨B, B ≠ D⟩ riduce il dominio di B a {1,2}. Quindi l'arco ⟨B, E <B⟩ riduce il dominio di B a {2}. Infine, l'arco ⟨E, E <B⟩ riduce il dominio di E a {1}. Tutti gli archi rimasti nella coda sono soddisfatti e quindi l'algoritmo arriva a uno stato di quiescenza e termina. Vengono quindi restituiti i domini delle variabili opportunamente ridotti. In questo caso, i domini hanno tutti la dimensione 1 e esiste un'unica soluzione: A = 4, B = 2, C = 3, D = 4, E = 1 che si può verificare rispetta tutti i vincoli.

Esercizio 6

Si vedano le slide del Corso