

FONDAMENTI DI INTELLIGENZA ARTIFICIALE

10 Gennaio 2019 – Tempo a disposizione: 2 h – Risultato: 32/32 punti

Esercizio 1 (punti 6)

Modellare in logica del I ordine le seguenti frasi:

1. CTA è la sigla di un aeroporto.
2. BLQ è la sigla di un aeroporto.
3. FunnyAir è una compagnia aerea
4. FU1088 è la sigla di un volo da CTA a BLQ
5. Esiste un volo da CTA a BLQ che è in ritardo
6. Ogni volo da CTA a BLQ è della FunnyAir;

Si mettano tutte le formule in forma a clausole e si dimostri poi, mediante il principio di risoluzione, che esiste un volo da CTA a BLQ, in ritardo e della FunnyAir.

Si utilizzi un linguaggio logico con i predicati seguenti:

aerop(X): X è la sigla di un aeroporto;

ritardo(X): X è in ritardo;

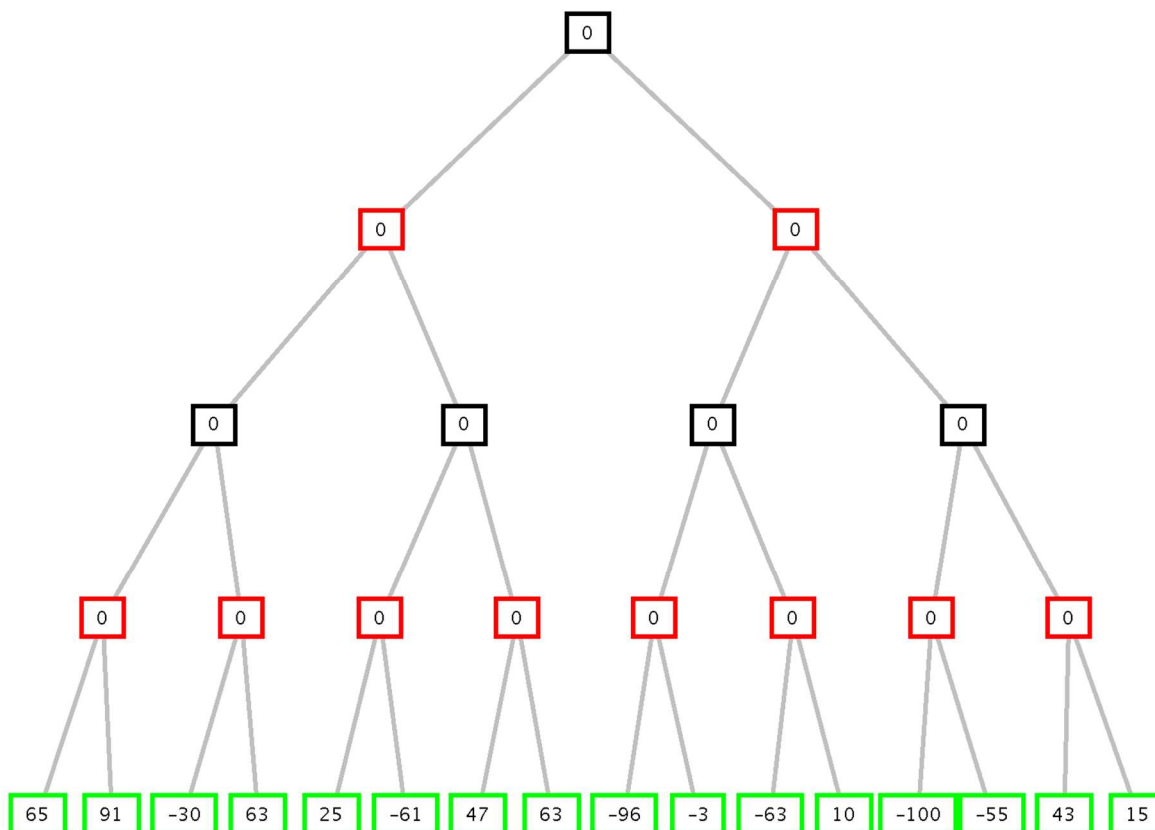
compagnia(X): X è il nome di una compagnia aerea;

volo(X, A, B): X è un volo da A a B ;

volo-di(X, Y): X è un volo della compagnia Y .

Esercizio 2 (punti 5)

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 5)

Dato il seguente programma Prolog che definisce la relazione di antenato e alcuni fatti su relazioni di parentela (padre/2 e madre/2):

```
antenato(X,Y):-padre(X,Y),!.
antenato(X,Y):-madre(X,Y),!.
antenato(X,Y):-antenato(X,Z),antenato(Z,Y).
padre(b,c).
padre(c,d).
madre(d,e).
```

si disegni l'albero SLD originato dal goal seguente (si indichino i tagli effettuati dal *cut* e non si espandano i rami tagliati):

```
?- antenato(b,d).
```

Esercizio 4 (punti 5)

Dato il programma Prolog dell'esercizio 3, si definisca un predicato `grado/3` tale che se X è antenato di Y , allora il predicato `grado(X,Y,G)` restituisce il grado di parentela G (numero di livelli di ascendenza) tra l'antenato X e Y , fallisce se X non è antenato di Y .

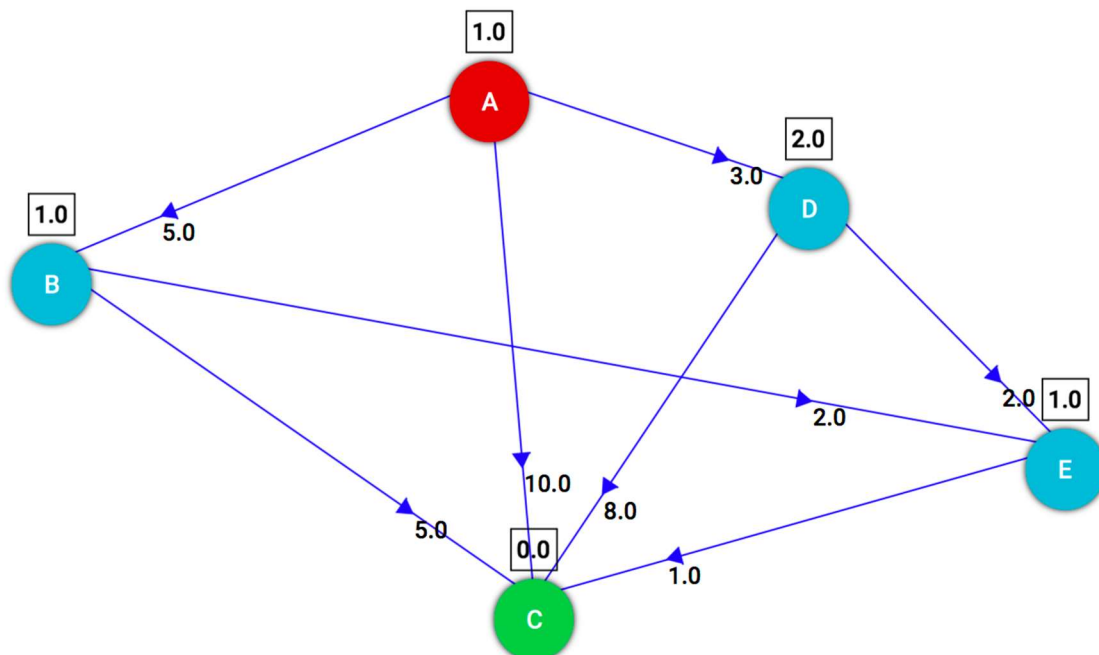
Il padre (o la madre) di una persona ha grado di parentela uguale a 1 con la persona.

Esempio:

```
?- grado(b,c,G).
Yes G=1
/* b è padre di c */
?- grado(b,e,G).
Yes G=3
?- grado(e,b,G).
No
```

Esercizio 5 (punti 7)

Si consideri il seguente grafo, dove A è il nodo iniziale e C il nodo goal, e il numero associato agli archi è il costo dell'operatore per andare dal nodo di partenza al nodo di arrivo dell'arco. A fianco di ogni nodo, in un quadrato, è indicata inoltre la stima euristica della sua distanza dal nodo goal.



- a) Si applichi la ricerca depth-first, e si disegni l'albero di ricerca sviluppato indicando per ogni nodo n il costo $g(n)$ e l'ordine di espansione; in caso di non-determinismo, si scelgano i nodi da espandere in base all'ordine alfabetico.
- b) Si applichi la ricerca A^* , e si disegni l'albero di ricerca sviluppato indicando per ogni nodo n la funzione $f(n)$ e l'ordine di espansione. In caso di non-determinismo, si scelgano i nodi da espandere in base all'ordine alfabetico. Si consideri come euristica $h(n)$ quella indicata nel quadrato a fianco di ogni nodo in figura, ovvero:
- $h(A) = 1$
 $h(B) = 1$
 $h(D) = 2$
 $h(E) = 1$
 $h(C) = 0$
- L'euristica h così definita è ammissibile?
Che vantaggio si ottiene applicando A^* , rispetto all'esito della ricerca depth-first?

Esercizio 6 (punti 4)

Si descriva brevemente l'algoritmo AC-3 (arc-consistency) visto a lezione, e se ne mostri l'applicazione al seguente CSP:

$X1, X2, X3 :: [1, 2, 3, 4]$

$X1 = 1$

$X1 < X2 < X3$

Si considerino le variabili in ordine crescente secondo il loro pedice (e conseguentemente i vincoli che le legano).

10 Gennaio 2019 - Soluzioni

Esercizio 1

1. `aerop(cta)`
2. `aerop(blq)`
3. `compagnia(funnyAir)`
4. `volo(fu1088,cta,blq)`
5. $\exists Y \text{ volo}(X,cta,blq) \text{ and ritardo}(X)$
6. $\forall X \text{ volo}(X,cta,blq) \Rightarrow \text{volo-di}(X,funnyAir)$

Query:

$\exists Y (\text{volo}(X,cta,blq) \text{ and ritardo}(X) \text{ and volo-di}(X,funnyAir))$

Goal (Query negata):

$\forall X (\text{not volo}(X,cta,blq) \text{ or not ritardo}(X) \text{ or not volo-di}(X,funnyAir))$

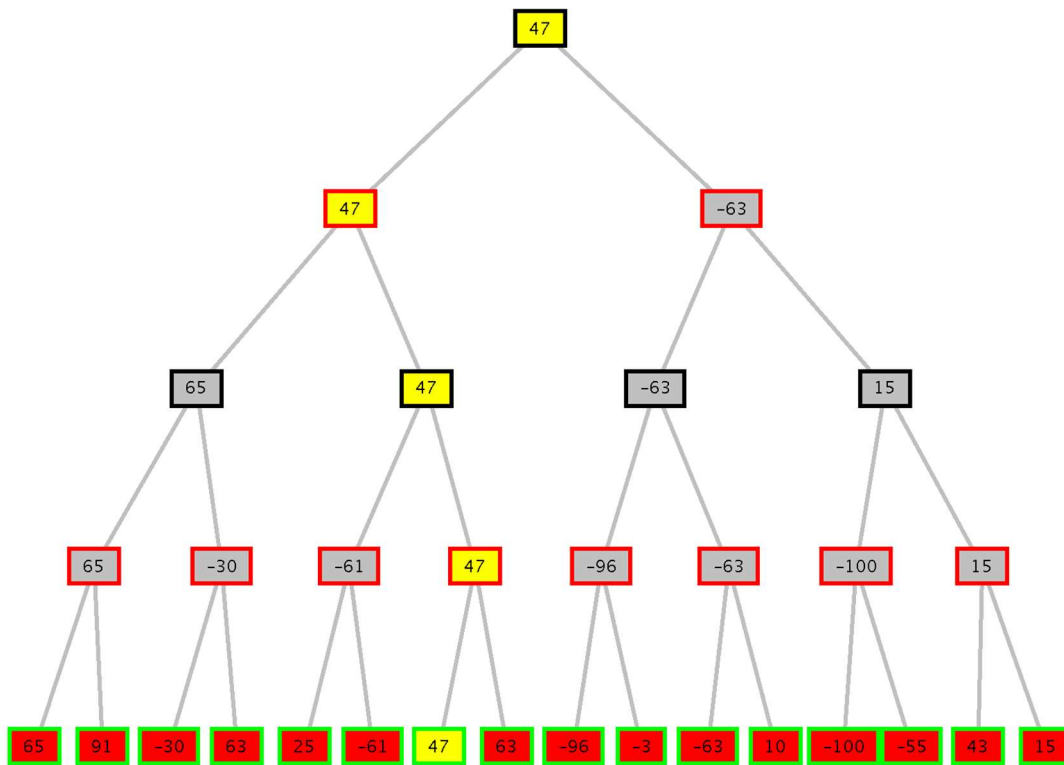
Trasformazione in clausole:

1. `aerop(cta)`
2. `aerop(blq)`
3. `compagnia(funnyAir)`
4. `volo(fu1088,cta,blq)`
- 5a. `volo(c,cta,blq)`
- 5b. `ritardo(c)`
- 6 `not volo(X,cta,blq) or volo-di(X,funnyAir)`
- GNeg: `not volo(X,cta,blq) or not ritardo(X) or not volo-di(X,funnyAir)`

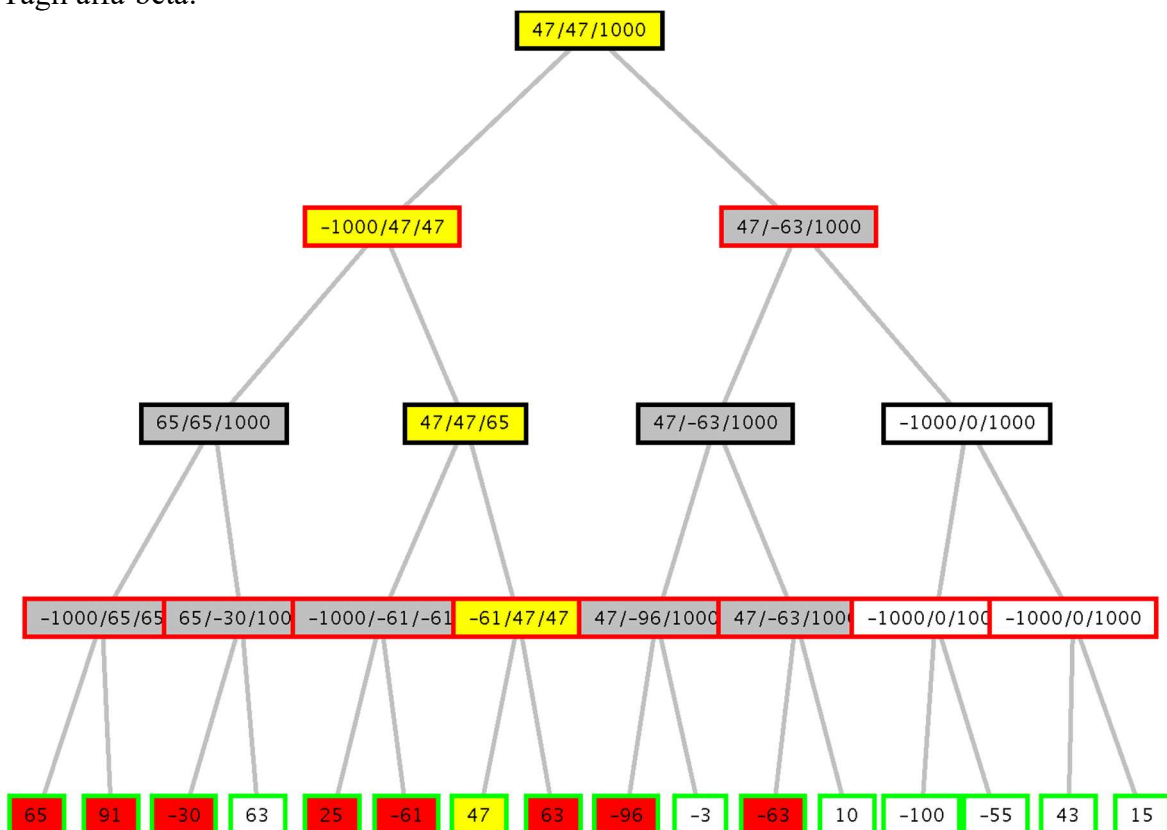
- 7: Gneg+ 5a: `not ritardo (c) or not volo-di(c, funny)`
- 8: 7+5b: `not volo-di(c, funny)`
- 9: 8+6: `not volo(c, cta, blq)`
- 10: 9+5a: `clausola vuota, contraddizione`

Esercizio 2

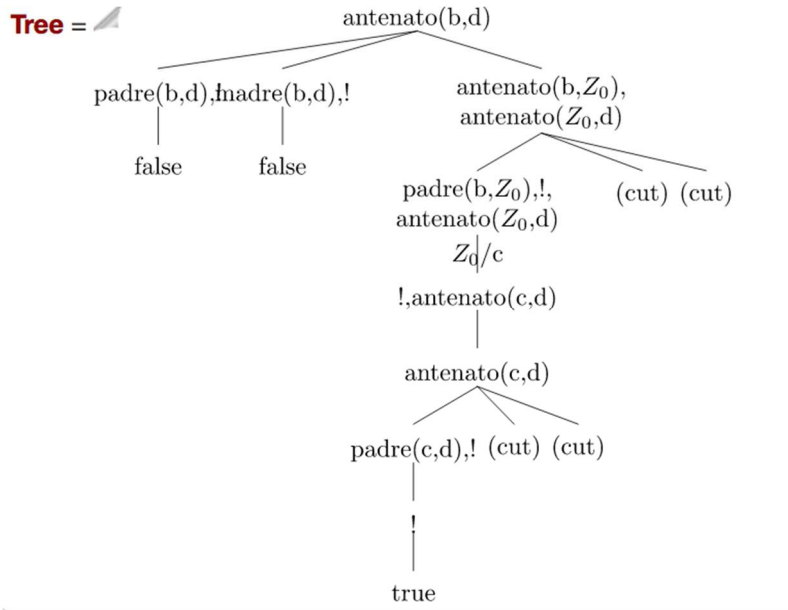
Min-Max:



Tagli alfa-beta:



Esercizio 3



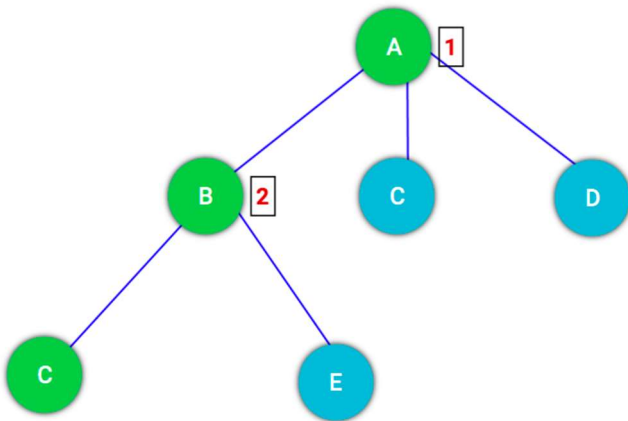
Esercizio 4

```

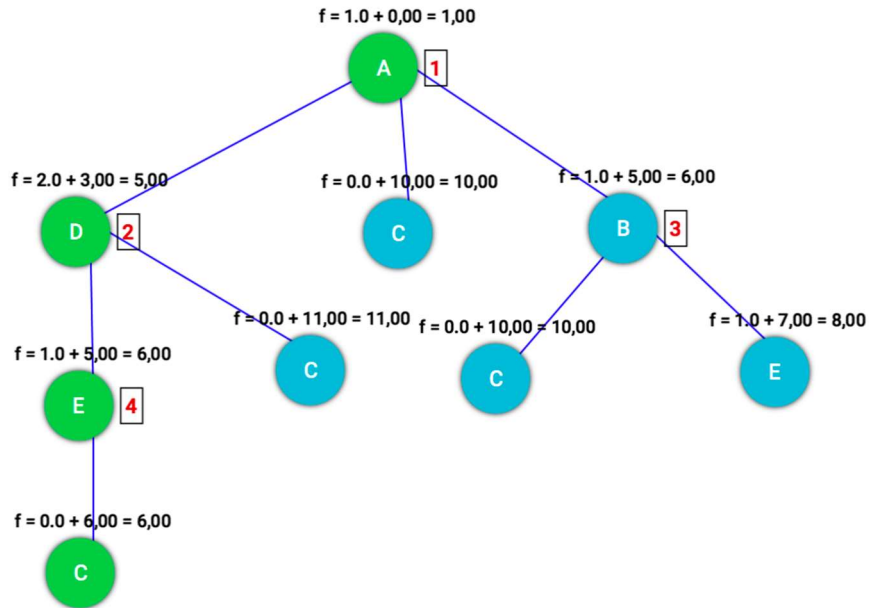
grado(X,Y,1):-padre(X,Y),!.
grado(X,Y,1):-madre(X,Y),!.
grado(X,Y,G):-grado(X,Z,G1),grado(Z,Y,G2),G is G1+G2.
  
```

Esercizio 5

Depth-first



Costo cammino trovato (in verde), strada ABC pari a 10.



Con A*, costo cammino trovato (in verde) – ADEC pari a 6 (cammino ottimo)
L'euristica è ammissibile.

Esercizio 6

Per la descrizione dell'algoritmo AC-3 si vedano le slides.

$X1, X2, X3 :: [1, 2, 3, 4]$

$X1=1$

$X1 < X2 < X3$

	X1	X2	X3
	1..4	1..4	1..4
X1=1	1		
X1<X2		1..4	
X1<X3			1..4
X2>X1		2..4	
X2<X3		2..3	
X3>X1			2..4
X3>X2			3..4

Ci sono state cancellazioni, si ricontrollano (tutti) gli archi che terminano su variabili i cui domini sono stati ridotti

	X1	X2	X3
	1	2..3	3..4
X1<X2	<i>idem</i>		
X1<X3	<i>idem</i>		
X2>X1		<i>idem</i>	
X2<X3		<i>idem</i>	
X3>X1			<i>idem</i>
X3>X2			<i>idem</i>

Quiescenza, termine di AC-3.