

TDE LP 10/11/2021 (primo compito)

Queste sono soltanto le mie 6 risposte dell'esame. Non assicuro la correttezza, perciò invito chiunque trovi un errore di suggerirmi la correzione. Potete contattarmi su Telegram @Kappo_99 (https://t.me/Kappo_99).

Domanda **1**

Risposta non ancora data

Punteggio max.: 5,00

Contrassegna domanda

Sia data la seguente descrizione di uno stato delle cose:

Se giochi e studi superi gli esami, ma se giochi e non studi non superi gli esami. Giochi. Non superi gli esami.

Dimostrare che non studi.

Si ricordino le tautologie viste a lezione.

◀
▶

↶ ↷

A ▾ B I

☰ ☱

📊 📝 📋

U ⊖ x₂ x² √

☰ ☱ ☲ ☳ ☴

✂ ↺ 🖼

⚙ 🔍 🌐 ⚙

</>

Legenda

G: giochi

S: studi

E: esame superato

Svolgimento

$S = \{ (G \text{ AND } S) \Rightarrow E, (G \text{ AND } \underline{\text{not}}(S)) \Rightarrow \underline{\text{not}}(E), G, \underline{\text{not}}(E) \}$

1. G	da S
2. <u>not</u> (E)	da S
3. (G AND <u>not</u> (S)) \Rightarrow <u>not</u> (E)	da S
4. <u>not</u> (G AND <u>not</u> (S)) OR <u>not</u> (E)	da conoscenza pregressa (conversione implica in OR)
5. <u>not</u> (G) OR S OR <u>not</u> (E)	da conoscenza pregressa (De Morgan)
6. <u>not</u> (G) OR S	da 2, 5 e Eliminazione OR
7. <u>not</u> (S)	da 1, 6 e Unit Resolution

not(S) = NON studi

Domanda **2**

Risposta non ancora data

Punteggio max.: 5,00

⚑

Contrassegna domanda

Tempo rimasto: 1:20

Considerate il seguente automa a stati finiti scritto in Prolog.

Quali espressioni riconosce questo automa?

NB. Il predicato `alphanumeric/1` è vero quando il suo argomento è una lettera o una cifra.

```

accept([I | Is], S) :- delta(S, I, N), accept(Is, N).
accept([], Q)       :- final(Q).

initial(start).
final(type).

delta(start, 'A', dev).
delta(start, 'B', dev).
delta(start, 'C', dev).
...
delta(start, 'Z', dev).
delta(dev, ':', n1).

delta(n1, '\', name).
delta(n1, L, name) :- alphanumeric(L).
delta(name, L, name) :- alphanumeric(L).
delta(name, '\', name).
delta(name, '.', type).
delta(name, L, type) :- alphanumeric(L).
```

Attenzione! **NON** scrivete "il programma riconosce un 'due punti', uno 'slash' e poi delle lettere e poi se la lista è vuota...". Dite solo che tipo di stringhe sono riconosciute dall'automato.

⏮

⏪

A ▾

B

I

≡

≡

📄

📝

📄

U

S

x₂

x²

I

≡

≡

≡

≡

≡

🔗

🔄

🖼️

⚙️

🔍

🌐

📄

🔗

Il programma riconosce le stringhe così composte:

- almeno una lettera maiuscola
- carattere :
- carattere \ oppure lettera maiuscola
- una serie di caratteri che possono essere \ o lettera maiuscola (almeno uno)
- carattere .
- una serie di lettere maiuscole (anche nessuna)

L'automato serve a riconoscere una serie di indirizzi URL, i quali potrebbero anche non essere ben formati secondo lo standard.

Di seguito alcuni esempi:

- `HTTPS:\\SITO1\\PAGINA3.HTML`

Domanda **3**

Risposta non ancora data

Punteggio max.: 5,00



Contrassegna domanda

Definite un predicato `deepest_node/3` che è vero quando il primo argomento è la radice di un (sotto)albero binario ed il secondo argomento è un nodo foglia che si trova alla profondità massima. La profondità massima è riportata nel terzo argomento. Se ci sono più foglie alla stessa profondità massima, non si specifica l'ordine in cui queste sono recuperabili via backtracking.

Si assume che i nodi siano definiti come `node(Key, Value, Left, Right)` come al solito.

Esempi

```
:- deepest_node(node(42, the_answer, void, void), DN, D).
```

```
DN = node(42, the_answer, nil, nil)
```

```
D = 0
```

```
:- deepest_node(node(42, the_answer, void, node(666, the_beast, void, void)), DN, D).
```

```
DN = node(666, the_beast, nil, nil)
```

```
D = 1
```

```
deepest_node(node(Key, Value, void, void), DN, D) :- deepest_node(_, node(Key, Value, nil, nil), 0) .
deepest_node(node(Key, Value, Left, void), DN, D) :- deepest_node(Left, DN, D+1) .
deepest_node(node(Key, Value, void, Right), DN, D) :- deepest_node(Right, DN, D+1) .
deepest_node(node(Key, Value, Left, Right), DN, D) :- deepest_node(Left, DN, D+1), deepest_node(Right, DN, D+1) .
```

Domanda **4**

Risposta non ancora data

Punteggio max.: 5,00



Contrassegna domanda

Dare una definizione di "regola di inferenza" e definire almeno tre regole di inferenza presentate a lezione.

<

Una regola di inferenza serve, a partire da alcune formule date nella forma FBF, a ricavare una nuova formula.

F1, F2, ..., Fk

[nome regola]

R

Ogni Fi rappresenta una formula (vera) in FBF e R è la formula generata da "Inserire" in FBF. [nome regola] serve per indicare quale regola di inferenza è stata usata.

Alcuni esempi di regole di inferenza viste a lezione sono:

A, A => B

Modus Ponens

B

A, not(A) OR B

Unit Resolution

B

not(B), A => B

Modus Tollens

not(A)

Domanda **5**

Risposta non ancora data

Punteggio max.: 5,00



Contrassegna domanda

Spiegare come avviene la dimostrazione per assurdo mediante uso del principio di risoluzione.

<

Il principio di risoluzione è l'unica regola di inferenza utilizzata da Prolog (è facile da usare e da implementare):

- opera su FBF trasformate in forma normale congiunta
- ognuno dei congiunti viene detto clausola

Per effettuare una dimostrazione per assurdo, il principio di risoluzione viene applicato nel seguente modo:

Supponiamo di avere a disposizione un insieme di FBF, e supponiamo di voler dimostrare che una formula atomica (ovvero una supposizione che facciamo) sia vera (ad esempio p).

Per poter procedere utilizzando il metodo della riduzione ad assurdo (reuction ad absurdum), assumiamo che not(p) sia vera.

Se combinandola con le mie FBF ottengo una contraddizione (ad esempio A AND not(A)), allora posso concludere che la mia supposizione p sia necessariamente vera.

Contrassegna
domanda

Oververo, qualunque sia la regola di calcolo, il numero di cammini di successo (se in numero finito) è lo stesso in tutti gli alberi SLD costruibili partendo dal nostro programma Prolog e il goal da dimostrare.