Inteligenta Artificiala - Proiectul 2
- Puzzles -

Malaescu Bianca

6/12/2020

1 The case of Lucy and Minna

1.1 Descrierea problemei

Investigator Craig was sent to Transylvania to solve a case. Trnasylvania is inhabited by both vampires and humans; the vampires always lie and the humans always tel the truth. However, half the inhabitants, both human and vampire, are insane and totally deluded in their beliefs - all true propositions they belive false and all false propositions they belive true. Sane humans and insane vampires both make only true statements; insane humans and sane vampires make only false statements.

This case involved two sisters named Lucy and Minna, and Craig had to determine which one of them was a vampire. It is known that one of them was a vampire and the other was human. As indicated above, nothing known about the sanity of either. Here is the transcript of the investigation:

Craig (to Lucy): Tell me something about yourselves.
Lucy: We are both insane.
Craig (to Minna): Is that true?
Minna: Of course not!

From this, Craig was able to prove to everyone's satisfaction which of the sisters was the vapire. Which one was it?

1.2 Rezolvare in Prover9 / Mace4

Cod:

```
% Saved by Prover9-Mace4 Version 0.5, December 2007.

set(ignore_option_dependencies). % GUI handles dependencies

if(Prover9). % Options for Prover9
  assign(max_seconds, 60).
end_if.

if(Mace4).   % Options for Mace4
  assign(domain_size, 2).
  assign(end_size, 2).
  assign(max_models, -1).
  assign(max_seconds, 60).
end_if.

formulas(assumptions).

%Lucy and Minna

%Lucy e vampir sau om
Lucy_vampire | Lucy_human.

%Doar vampir sau doar om.
Lucy_vampire -> -Lucy_human.
```

```
25 Lucy_human -> -Lucy_vampire.
26
27 %Mina e vampir sau om
28 Minna_vampire | Minna_human.
29
30 %Doar vampir sau doar om
31 Minna_vampire -> -Minna_human.
32 Minna_human -> -Minna_vampire.
33
34 %una e vampire si una om
35 Minna_vampire <-> Lucy_human.
36 Lucy_vampire <-> Minna_human.
37
38 %oamenii sau vampirii pot fi sane sau insane
39 Lucy_human <-> Lucy_sane_human | Lucy_insane_human.
40 Minna_human <-> Minna_sane_human | Minna_insane_human.
41 Lucy_vampire <-> Lucy_sane_vampire | Lucy_insane_vampire.
42 Minna_vampire <-> Minna_sane_vampire | Minna_insane_vampire.
43
44 %Lucy si Mina pot fi vampire_sane, vampire_insane, human_sane sau human_insane
45 Lucy_sane_human | Lucy_insane_human | Lucy_sane_vampire | Lucy_insane_vampire.
46 Minna_sane_human | Minna_insane_human | Minna_sane_vampire | Minna_insane_vampire.
47
48 %Pot fi doar una din variante
49 Lucy_sane_human ->  -Lucy_insane_human & -Lucy_sane_vampire & -Lucy_insane_vampire.
50 Lucy_insane_human ->  -Lucy_sane_human & -Lucy_sane_vampire & -Lucy_insane_vampire.
51 Lucy_sane_vampire ->  -Lucy_insane_human & -Lucy_sane_human & -Lucy_insane_vampire.
52 Lucy_insane_vampire ->  -Lucy_insane_human & -Lucy_sane_vampire & -Lucy_sane_human.
53
54 Minna_sane_human ->  -Minna_insane_human & -Minna_sane_vampire & -Minna_insane_vampire.
55 Minna_insane_human ->  -Minna_sane_human & -Minna_sane_vampire & -Minna_insane_vampire.
56 Minna_sane_vampire ->  -Minna_insane_human & -Minna_sane_human & -Minna_insane_vampire.
57 Minna_insane_vampire ->  -Minna_insane_human & -Minna_sane_vampire & -Minna_sane_human.
58
59 Lucy_sane_human | Lucy_insane_vampire -> L1.
60 Lucy_insane_human  | Lucy_sane_vampire -> -L1.
61
62 Minna_sane_human | Minna_insane_vampire -> M1.
63 Minna_insane_human  | Minna_sane_vampire -> -M1.
64
65 L1 <-> (Lucy_insane_human & Minna_insane_vampire) | (Minna_insane_human & Lucy_insane_vampire).
66 M1 <->  -((Lucy_insane_human & Minna_insane_vampire) | (Minna_insane_human & Lucy_insane_vampire)).
67
68 end_of_list.
69
70 formulas(goals).
71
72 end_of_list.
```

1.3 Explicarea codului

Minna si Lucy sunt vampiri sau oameni, dar stim sigur ca una dintre ele este vampir si cealalta este om. Acest fapt l-am transcris in logica propozitionala intre liniile 20 si 36.

Stim ca oamenii si vampirii din Transilvania pot fi sanatosi sau nebuni, deci Lucy si Minna se pot regasi in una din categoriile: om sanatos, om nebun, vampir sanatos si vampir nebun. Fiecare din aceste variante le exclude pe celelalte trei, fapt descris intre liniile 48 si 57.

In functie de categoria in care se incadreaza (om sanatos, om nebun, vampir sanatos, vampir nebun), intre liniile 59 si 63 am stabilit valoarea de adevar a afirmatiilor fiecarei surori. Daca acestea sunt oameni sanatosi sau vampiri nebun, afirmatiile lor sunt adevarate, iar daca sunt vampiri sanatosi sau oameni nebuni afirmatiile acestora vor fi fase.

In continuare am scris in logica propozitionala afirmatiile celor doua surori: Lucy afirma ca amandoua sunt nebune, dar Minna infirma acest lucru (negarea propozitiei lui Lucy).

1.4 Rezultate

```
interpretation( 2, [number = 1,seconds = 0], [
    relation(L1, [0]),
    relation(Lucy_human, [0]),
    relation(Lucy_insane_human, [0]),
    relation(Lucy_insane_vampire, [0]),
    relation(Lucy_sane_human, [0]),
    relation(Lucy_sane_vampire, [1]),
    relation(Lucy_vampire, [1]),
    relation(M1, [1]),
    relation(Minna_human, [1]),
    relation(Minna_insane_human, [0]),
    relation(Minna_insane_vampire, [0]),
    relation(Minna_sane_human, [1]),
    relation(Minna_sane_vampire, [0]),
    relation(Minna_vampire, [0])]).
interpretation( 2, [number = 2,seconds = 0], [
    relation(L1, [1]),
    relation(Lucy_human, [0]),
    relation(Lucy_insane_human, [0]),
    relation(Lucy_insane_vampire, [1]),
    relation(Lucy_sane_human, [0]),
    relation(Lucy_sane_vampire, [0]),
    relation(Lucy_vampire, [1]),
    relation(M1, [0]),
    relation(Minna_human, [1]),
    relation(Minna_insane_human, [1]),
    relation(Minna_insane_vampire, [0]),
    relation(Minna_sane_human, [0]),
    relation(Minna_sane_vampire, [0]),
    relation(Minna_vampire, [0])]).
```

Ruland Mace4 observam ca obtinem doua modele: in primul, Lucy este vampir nebun iar Minna este om nebun, iar in al doilea Lucy este vampir sanatos si Minna este om nebun. Din aceste modele reiese ca solutia problemei este: Lucy este vampir si Minna este om, fapt ce se poate demonstra si ruland Proover9.

2 The Lady and the tigers: the fourth day.

2.1 Descrierea problemei

A person accused of a crime is brought into a public arena and must choose one of 9 doors with messages on them. Behind one door is a lady whom the king has deemed an appropriate match for the accused; behind the others can be a fierce, hungry tiger or an empty room. All doors are heavily soundproofed to prevent the accused from hearing what is behind each one. If he chooses the door with the lady behind it, he is innocent and must immediately marry her, but if he chooses the door with the tiger behind it, he is deemed guilty and is immediately devoured by the animal.

The king added: the sign on the door of the room containing the lady is true; the signs on doors of all rooms containing the tigers are false; and the signs on doors of empty rooms can be either true or false. Here are the signs:

- I = The lady is in an odd-numbered room.
- II = This room is empty.
- III = Either sign V or sign VII is wrong.
- IV = Sign I is wrong.
- V = Either sign II or sign IV is right.
- VI = Sign III is wrong.
- VII = The lady is not in room I.
- VIII = This room contains a tiger and room IX is empty.
- IX = This room contains a tiger and VI is wrong.

The king was decent enough to tell him whether Room VIII was empty or not, and the prisoner was then able to deduce where the lady was.

2.2 Rezolvare in Prover9 / Mace4

Cod:

```
% Saved by Prover9-Mace4 Version 0.5, December 2007.

set(ignore_option_dependencies). % GUI handles dependencies

if(Prover9). % Options for Prover9
  assign(max_seconds, 60).
end_if.

if(Mace4).   % Options for Mace4
  assign(domain_size, 2).
  assign(end_size, 2).
  assign(max_models, -1).
  assign(max_seconds, 60).
end_if.
```

```

formulas(assumptions).

%Lady, tigers and empty rooms - 9 rooms

%cel putin o lady in camere
L1 | L2 | L3 |L4 | L5 | L6 | L7 | L8 | L9.

%cel putin un tigru in camere
T1 | T2 | T3 |T4 | T5 | T6 | T7 | T8 | T9.

%este cel putin o camera goala
E1 | E2 | E3 |E4 | E5 | E6 | E7 | E8 | E9.

%o singura lady in camere
L1 -> -L2 & -L3 & -L4 & -L5 & -L6 & -L7 & -L8 & -L9.
L2 -> -L1 & -L3 & -L4 & -L5 & -L6 & -L7 & -L8 & -L9.
L3 -> -L2 & -L1 & -L4 & -L5 & -L6 & -L7 & -L8 & -L9.
L4 -> -L2 & -L3 & -L1 & -L5 & -L6 & -L7 & -L8 & -L9.
L5 -> -L2 & -L3 & -L4 & -L1 & -L6 & -L7 & -L8 & -L9.
L6 -> -L2 & -L3 & -L4 & -L5 & -L1 & -L7 & -L8 & -L9.
L7 -> -L2 & -L3 & -L4 & -L5 & -L6 & -L1 & -L8 & -L9.
L8 -> -L2 & -L3 & -L4 & -L5 & -L6 & -L7 & -L1 & -L9.
L9 -> -L2 & -L3 & -L4 & -L5 & -L6 & -L7 & -L8 & -L1.

%in fiecare camera poate fi lady, tigru sau empty
L1 | T1 | E1.
L2 | T2 | E2.
L3 | T3 | E3.
L4 | T4 | E4.
L5 | T5 | E5.
L6 | T6 | E6.
L7 | T7 | E7.
L8 | T8 | E8.
L9 | T9 | E9.

%daca e lady, nu e tigru sau empty
L1 -> -T1 & -E1.
L2 -> -T2 & -E2.
L3 -> -T3 & -E3.
L4 -> -T4 & -E4.
L5 -> -T5 & -E5.
L6 -> -T6 & -E6.
L7 -> -T7 & -E7.
L8 -> -T8 & -E8.
L9 -> -T9 & -E9.

%daca e tigru, nu e lady sau empty
T1 -> -L1 & -E1.
T2 -> -L2 & -E2.
T3 -> -L3 & -E3.
T4 -> -L4 & -E4.
T5 -> -L5 & -E5.
T6 -> -L6 & -E6.
```

```
T7 -> -L7 & -E7.
T8 -> -L8 & -E8.
T9 -> -L9 & -E9.

%daca e empty, nu e lady sau tigru
E1 -> -L1 & -T1.
E2 -> -L2 & -T2.
E3 -> -L3 & -T3.
E4 -> -L4 & -T4.
E5 -> -L5 & -T5.
E6 -> -L6 & -T6.
E7 -> -L7 & -T7.
E8 -> -L8 & -T8.
E9 -> -L9 & -T9.

%pe usile unde e lady afirmatiile sunt adevarate
L1 -> R1.
L2 -> R2.
L3 -> R3.
L4 -> R4.
L5 -> R5.
L6 -> R6.
L7 -> R7.
L8 -> R8.
L9 -> R9.

%pe usile camerelor cu tigru, afirmatiile scrise sunt false
T1 -> -R1.
T2 -> -R2.
T3 -> -R3.
T4 -> -R4.
T5 -> -R5.
T6 -> -R6.
T7 -> -R7.
T8 -> -R8.
T9 -> -R9.

%indiciile de pe usi
R1 <-> L1 | L3 | L5 | L7 | L9.
R2 <-> E2.
R3 <-> R5 | -R7.
R4 <-> -R1.
R5 <-> R2 | R4.
R6 <-> -R3.
R7 <-> -L1.
R8 <-> T8 & E9.
R9 <-> T9 & -R6.

%indiciul in plus de la rege
-E8. %sau E8.

end_of_list.

formulas(goals).
```

```
123
124 end_of_list.
```

2.3 Explicarea codului

In fiecare camera se poate afla printesa, un tigru sau o camera goala. Prezenta printesei in camera "x" am reprezentat-o prin "Lx" (ex: L7 inseamna ca printesa se afla in camera 7), prezenta unui tigru intr-o camera am reprezentat-o prin "Tx", unde "x" este numarul camerei (ex: T2 inseamna ca in camera 2 se afla un tigru), iar faptul ca o camera este goala l-am reprezentat prin "Ex", unde "x" este, la fel, numarul camerei. "Rx" reprezinta mesajul scris pe usa camerei "x", care poate fi adevarat ("Rx.") sau fals ("-Rx.").

Intre liniile 21 si 27 am exprimat in logica propozitionala faptul ca exista printesa, tigru si camera goala in cel putin una din camere. Stim ca este o singura lady in povestea noastra, deci prezenta acesteia intr-o camera exclude prezenta acesteia in toate celelalte, fapt descris intre liniile 29 si 38. Urmatoarele linii subliniaza ca in fiecare camera poate fi doar una dintre cele 3 variante posibile (lady, tigru sau camera goala), si fiecare dintre aceste variante le exclude pe celelalte doua, atfel ca nicio camera nu poate contine mai mult de un element.

Intre liniile 84 si 104 am stabilit valorile de adevar pentru mesajele de pe fiecare usa, in functie de ce se afla in camerele respective. Daca in camera "x" se afla printesa, atunci mesajul de pe usa camerei este adevarat (Lx -> Rx). Daca in camera "x" se afla un tigru, mesajul de pe usa camerei e fals (Tx -> -Rx). Mesajele scris pe usile camerelor goale pot fi adevarate sau false, asa ca nu am mai tratat acest caz in logica propozitionala, nefiind necesar.

Liniile 106-115 prezinta traducerea mesajelor de pe usi in logica propozitionala, linia 118 fiind indiciul in plus oferit prizonierului de catre rege. Aici avem 2 variante: daca regele i-a confirmat prizonierului ca nu se alfa nimic in camera 8 ("E8.") sau i-a infirmat acest lucru ("-E8."), fiind nevoiti sa rulam Mace4 pentru ambele variante.

2.4 Rezultate

1. Cazul in care camera 8 este goala.

Ruland Mace4 observam ca obtinem 55 de modele: in 24 dintre ele printesa se afla in camera 1, in 4 modele se afla in camera 3, in 8 modele se afla in camera 4, in 4 modele se afla in camera 5 si in 20 de modele printesa este in camera7. De aici putem deduce ca indiciul oferit de rege ales (camera 8 este goala) nu este unul concludent, deci in cazul in care nu se alfa nici tigru si nici printesa in camera 8, prizonierului ii este imposibil sa gaseasca unde se afla printesa. Incercam sa rulam Mace4 luan in considerare indiciul ca camera 8 nu este goala.

2. Cazul in care camera 8 nu este goala.

Ruland Mace4 observam ca obtinem opt modele, deci 8 solutii posibile pentru problema noastra. In toate cele 8 modele printesa se afla in camera 7, asadar solutia ramane aceeasi pentru oricare dintre cele 8 scenarii posibile. Prizonierul poate afla ca printesa se afla in camera 7.

3 Friends

3.1 Descrierea problemei

Suppose we know that:

- “if Paolo is thin, then Carlo is not blonde or Roberta is not tall”
- “if Roberta is tall then Sandra is lovely”
- “if Sandra is lovely and Carlo is blonde then Paolo is thin”
- “Carlo is blonde”

Can we deduce that “Roberta is not tall” ?

3.2 Rezolvare in Prover9 / Mace4

Cod:

```
% Saved by Prover9-Mace4 Version 0.5, December 2007.

set(ignore_option_dependencies). % GUI handles dependencies

if(Prover9). % Options for Prover9
  assign(max_seconds, 60).
end_if.

if(Mace4).   % Options for Mace4
  assign(domain_size, 9).
  assign(end_size, 9).
  assign(max_models, -1).
  assign(max_seconds, 1200).
end_if.

formulas(assumptions).

%constante: Paolo, Carlo, Roberta, Sandra
%predicate: thin, blonde, tall, lovely

thin(Paolo) -> -blonde(Carlo) | -tall(Roberta).
tall(Roberta) -> lovely(Sandra).
lovely(Sandra) & blonde(Carlo) -> thin(Paolo).
blonde(Carlo).

end_of_list.

formulas(goals).

-tall(Roberta).

end_of_list.
```

3.3 Explicarea codului

Am stabilit constantele si predicatele pentru aceasta problema.

Constante: Paolo, Carlo, Roberta, Sandra.

Predicate: thin, tall, blonde, lovely.

In continuare, intre liniile 21 si 24 am descris in first order logic indiciile care se dau. Linia 30 reprezinta faptul ce trebuie demonstrat folosind Prover9(Roberta nu este inalta).

3.4 Rezultate

```
============================== prooftrans ============================
Prover9 (32) version Dec-2007, Dec 2007.
Process 12816 was started by Bianca on DESKTOP-2ATM775,
Sun Dec  5 15:04:10 2021
The command was "/cygdrive/c/Program Files (x86)/Prover9-Mace4/bin-win32/prover9".
============================== end of head ===========================

============================== end of input ==========================

============================== PROOF =================================

% -------- Comments from original proof --------
% Proof 1 at 0.00 (+ 0.01) seconds.
% Length of proof is 14.
% Level of proof is 5.
% Maximum clause weight is 2.
% Given clauses 0.

1 thin(Paolo) -> -blonde(Carlo) | -tall(Roberta) # label(non_clause).  [assumption].
2 tall(Roberta) -> lovely(Sandra) # label(non_clause).  [assumption].
3 lovely(Sandra) & blonde(Carlo) -> thin(Paolo) # label(non_clause).  [assumption].
4 -tall(Roberta) # label(non_clause) # label(goal).  [goal].
5 -lovely(Sandra) | -blonde(Carlo) | thin(Paolo).  [clausify(3)].
6 -thin(Paolo) | -blonde(Carlo) | -tall(Roberta).  [clausify(1)].
7 tall(Roberta).  [deny(4)].
8 -tall(Roberta) | lovely(Sandra).  [clausify(2)].
9 -lovely(Sandra) | -blonde(Carlo) | -blonde(Carlo) | -tall(Roberta).  [resolve(5,c,6,a)].
10 -lovely(Sandra) | -blonde(Carlo) | -blonde(Carlo).  [resolve(9,d,7,a)].
11 lovely(Sandra).  [resolve(7,a,8,a)].
12 blonde(Carlo).  [assumption].
13 -blonde(Carlo) | -blonde(Carlo).  [resolve(10,a,11,a)].
14 $F.  [copy(13),merge(b),unit_del(a,12)].

============================== end of proof ==========================
```

Cerinta a fost demonstrata.

4 Date night

4.1 Descrierea problemei

Five couples are next to each other on a date night. Each couple is at a table that has a flower and they are eating pasta and drinking wine. Follow the clues to find out which wine they are drinking and which pasta they are eating.

- The couple drinking French wine is exactly to the left of the couple who has been dating for 3 years.
- At one of the ends is the table that has the Red flower.
- Laura is exactly to the left of the couple who is eating Rigatoni.
- Daniel is exactly to the right of the couple drinking Italian wine.
- Donna is sitting somewhere between the couple that has been dating for 6 years and Ann, in that order.
- Laura is exactly to the right of the couple eating Spaghetti.
- The White flower's table is somewhere to the left of the couple that is eating Fettuccine.
- Barry is somewhere between the table that has the Pink flower and Andrew, in that order.
- At one of the ends is the couple that has been dating for 2 years.
- Rose and his boyfriend are eating Rigatoni.
- Andrew is next to the couple who has been dating for 5 years.
- The table that has the White flower is somewhere to the left of the couple drinking Spanish wine.
- The couples eating Spaghetti and Fettuccine are next to each other.
- At the second position is the couple eating Lasagne.
- The couple that has been dating for 6 years is at the table that has the Pink flower.
- Scott's table is somewhere to the right of the table that has the Purple flower.
- The couple eating Rigatoni is at one of the ends.
- The couple that has been dating for 4 years is next to the table that has the Pink flower. couple drinking Spanish wine is somewhere between Kristen and the couple drinking Chilean wine, in that order.
- The couple drinking Italian wine is somewhere to the left of the couple that has been dating for 3 years.
- Laura is exactly to the left of the couple drinking Argentine wine.

4.2 Rezolvare in Prover9 / Mace4

Cod:

```
% Saved by Prover9-Mace4 Version 0.5, December 2007.

set(ignore_option_dependencies). % GUI handles dependencies

if(Prover9). % Options for Prover9
  assign(max_seconds, 60).
end_if.

if(Mace4).   % Options for Mace4
  assign(domain_size, 5).
  assign(start_size, 5).
  assign(end_size, 5).
  assign(max_models, -1).
  assign(max_seconds, 60).
end_if.

formulas(assumptions).

%fiecare masa (a, b, c, d, e) e diferita de cealalta
diferit(a, b).
diferit(a, c).
diferit(a, d).
diferit(a, e).
diferit(b, c).
diferit(b, d).
diferit(b, e).
diferit(c, d).
diferit(c, e).
diferit(d, e).

diferit(x, y) -> diferit(y, x).

vecinDreapta(a,b).
vecinDreapta(b,c).
vecinDreapta(c,d).
vecinDreapta(d,e).

-vecinDreapta(a,a).
-vecinDreapta(a,c).
-vecinDreapta(a,d).
-vecinDreapta(a,e).

-vecinDreapta(b,a).
-vecinDreapta(b,b).
-vecinDreapta(b,d).
-vecinDreapta(b,e).

-vecinDreapta(c,b).
-vecinDreapta(c,a).
-vecinDreapta(c,c).
-vecinDreapta(c,e).
```

```

-vecinDreapta(d,d).
-vecinDreapta(d,c).
-vecinDreapta(d,b).
-vecinDreapta(d,a).

-vecinDreapta(e,e).
-vecinDreapta(e,d).
-vecinDreapta(e,c).
-vecinDreapta(e,b).
-vecinDreapta(e,a).

vecin(x,y) <-> vecinDreapta(x,y) | vecinDreapta(y,x).

undevaDreapta(a,b).
undevaDreapta(a,c).
undevaDreapta(a,d).
undevaDreapta(a,e).

undevaDreapta(b,c).
undevaDreapta(b,d).
undevaDreapta(b,e).

undevaDreapta(c,d).
undevaDreapta(c,e).

undevaDreapta(d,e).

-undevaDreapta(a, a).

-undevaDreapta(b,a).
-undevaDreapta(b,b).

-undevaDreapta(c,a).
-undevaDreapta(c,b).
-undevaDreapta(c,c).

-undevaDreapta(d,c).
-undevaDreapta(d,b).
-undevaDreapta(d,a).
-undevaDreapta(d,d).

-undevaDreapta(e,a).
-undevaDreapta(e,b).
-undevaDreapta(e,c).
-undevaDreapta(e,d).
-undevaDreapta(e,e).

Andrew(x) | Barry(x) | Daniel(x) | Jerry(x) | Scott(x).
Ann(x) | Donna(x) | Kristen(x) | Rose(x) | Laura(x).
blue(x) | pink(x) | purple(x) | red(x) | white(x).
2years(x) | 3years(x) | 4years(x) | 5years(x) | 6years(x).
Fettuccine(x) | Lasagne(x) | Ravioli(x) | Spaghetti(x) | Rigatoni(x).
Argentine(x) | Chilean(x) | French(x) | Italian(x) | Spanish(x).
```

```

Andrew(x) & Andrew(y) -> -diferit(x,y).
Barry(x) & Barry(y) -> -diferit(x,y).
Daniel(x) & Daniel(y) -> -diferit(x,y).
Jerry(x) & Jerry(y) -> -diferit(x,y).
Scott(x) & Scott(y) -> -diferit(x,y).

Ann(x) & Ann(y) -> -diferit(x,y).
Donna(x) & Donna(y) -> -diferit(x,y).
Kristen(x) & Kristen(y) -> -diferit(x,y).
Rose(x) & Rose(y) -> -diferit(x,y).
Laura(x) & Laura(y) -> -diferit(x,y).

blue(x) & blue(y) -> -diferit(x,y).
pink(x) & pink(y) -> -diferit(x,y).
purple(x) & purple(y) -> -diferit(x,y).
red(x) & red(y) -> -diferit(x,y).
white(x) & white(y) -> -diferit(x,y).

2years(x) & 2years(y) -> -diferit(x,y).
3years(x) & 3years(y) -> -diferit(x,y).
4years(x) & 4years(y) -> -diferit(x,y).
5years(x) & 5years(y) -> -diferit(x,y).
6years(x) & 6years(y) -> -diferit(x,y).

Fettuccine(x) & Fettuccine(y) -> -diferit(x,y).
Lasagne(x) & Lasagne(y) -> -diferit(x,y).
Ravioli(x) & Ravioli(y) -> -diferit(x,y).
Spaghetti(x) & Spaghetti(y) -> -diferit(x,y).
Rigatoni(x) & Rigatoni(y) -> -diferit(x,y).
Argentine(x) & Argentine(y) -> -diferit(x,y).
Chilean(x) & Chilean(y) -> -diferit(x,y).
French(x) & French(y) -> -diferit(x,y).
Italian(x) & Italian(y) -> -diferit(x,y).
Spanish(x) & Spanish(y) -> -diferit(x,y).

%Indiciile

3years(x) & French(y) -> vecinDreapta(y,x).

red(e) | red(a).

Rigatoni(x) & Laura(y) -> vecinDreapta(y,x).

Daniel(x) & Italian(y) -> vecinDreapta(y,x).

6years(x)&Donna(y) -> undevaDreapta(x,y).
Ann(x)&Donna(y) -> undevaDreapta(y,x).

Laura(x)&Spaghetti(y)->vecinDreapta(y,x).

Fettuccine(x) & white(y) -> undevaDreapta(y,x).

pink(x) & Barry(y) -> undevaDreapta(x,y).
```

```
160 Andrew(x) & Barry(y) -> undevaDreapta(y,x).
161
162 2years(a) | 2years(e).
163
164 Rose(x) <-> Rigatoni(x).
165
166 Andrew(y) & 5years(x) -> vecin(x,y).
167
168 Spanish(x) & white(y) -> undevaDreapta(y,x).
169
170 Spaghetti(y) & Fettuccine(x) -> vecin(x,y).
171
172 Lasagne(b).
173
174 6years(x) <->pink(x).
175
176 purple(x) & Scott(y) -> undevaDreapta(x,y).
177
178 Rigatoni(a) | Rigatoni(e).
179
180 4years(y) & pink(x) -> vecin(x,y).
181
182 Kristen(x) & Spanish(y) -> undevaDreapta(x,y).
183 Chilean(x) & Spanish(y) -> undevaDreapta(y,x).
184
185 3years(x) & Italian(y) -> undevaDreapta(y,x).
186
187 Argentine(x) & Laura(y) -> vecinDreapta(y,x).
188
189 end_of_list.
190
191 formulas(goals).
192
193 end_of_list.
```

4.3 Explicarea codului

Pentru a sublinia ca fiecare masa (a, b, c, d, e) este diferita de cealalta, am folosit diferit(x, y). De asemenea, pentru a traduce in first order logic indiciile, am avut nevoie de conceptul de vecin, vecin din dreapta, undeva in dreapta.

Pentru a ma putea folosi de vecinul din dreapta ca sa exprim si vecinul din stanga, vecinDreapta(y,x) ar fi echivalentul vecinului din stanga pentru "x" si "y". Asemena acestui exemplu pot exprima si conceptul de "undeva in stanga" folosing undevaDreapta(y,x).

Liniile 33-98 descriu aceste concepte, precizand valoarea de adevar pentru fiecare pereche de mese (a, b, c, d, e). In continuare precizez ca pentru fiecare masa "x" se potriveste cate o valoare din fiecare set de caracteristici(ex: din setul de caracteristici ale pastelor pe care le mananca fiecare pereche la masa x, pot fi: fettucine, lasagna, ravioli, spaghetti sau rigatoni). Penru a sublinia factul ca oricare din aceste valori ale caracteristicilor se potriveste doar unei mese, m-am folosit de conceptul de "diferit" implementat. Daca se intampla ca doua mese sa aiba acceasi valoare a unei caracteristici, inseamna ca de fapt facem referire la una si aceeasi masa.(ex: blue(x) blue(y) -> -diferit(x,y). Am precizat acest lucru pentru fiecare valoare.

Incepand cu linia 142 am transcris in FOL indiciile, folosindu-ma si de conceptele descrise de "vecin", "vecinDreapta", "undevaDreapta".

4.4 Rezultate

```
interpretation( 5, [number = 1,seconds = 0], [
    function(a, [0]),
    function(b, [1]),
    function(c, [2]),
    function(d, [3]),
    function(e, [4]),
    relation(2years(_), [0,0,0,0,1]),
    relation(3years(_), [0,0,1,0,0]),
    relation(4years(_), [0,1,0,0,0]),
    relation(5years(_), [0,0,0,1,0]),
    relation(6years(_), [1,0,0,0,0]),
    relation(Andrew(_), [0,0,0,0,1]),
    relation(Ann(_), [0,0,1,0,0]),
    relation(Argentine(_), [0,0,0,0,1]),
    relation(Barry(_), [0,0,1,0,0]),
    relation(Chilean(_), [0,0,0,1,0]),
    relation(Daniel(_), [0,1,0,0,0]),
    relation(Donna(_), [0,1,0,0,0]),
    relation(Fettuccine(_), [0,0,0,1,0]),
    relation(French(_), [0,1,0,0,0]),
    relation(Italian(_), [1,0,0,0,0]),
    relation(Jerry(_), [1,0,0,0,0]),
    relation(Kristen(_), [1,0,0,0,0]),
    relation(Lasagne(_), [0,1,0,0,0]),
    relation(Laura(_), [0,0,0,1,0]),
    relation(Ravioli(_), [1,0,0,0,0]),
    relation(Rigatoni(_), [0,0,0,0,1]),
    relation(Rose(_), [0,0,0,0,1]),
    relation(Scott(_), [0,0,0,1,0]),
    relation(Spaghetti(_), [0,0,1,0,0]),
    relation(Spanish(_), [0,0,1,0,0]),
    relation(blue(_), [0,0,0,1,0]),
    relation(pink(_), [1,0,0,0,0]),
    relation(purple(_), [0,0,1,0,0]),
    relation(red(_), [0,0,0,0,1]),
    relation(white(_), [0,1,0,0,0]),
    relation(diferit(_,_), [
        0,1,1,1,1,
        1,0,1,1,1,
        1,1,0,1,1,
        1,1,1,0,1,
        1,1,1,1,0]),
    relation(undevaDreapta(_,_), [
        0,1,1,1,1,
        0,0,1,1,1,
        0,0,0,1,1,
        0,0,0,0,1,
        0,0,0,0,0]),
    relation(vecin(_,_), [
        0,1,0,0,0,
        1,0,1,0,0,
        0,1,0,1,0,
        0,0,1,0,1,
        0,0,0,1,0]),
    relation(vecinDreapta(_,_), [
        0,1,0,0,0,
        0,0,1,0,0,
        0,0,0,1,0,
        0,0,0,0,1,
        0,0,0,0,0])]).
```

Ruland Mace4 observam ca obtinem un singur model care respecta toate indiciile puzzle-ului. Pentru a putea fi observat mai usor, l-am descris prin urmatoarul tabel:

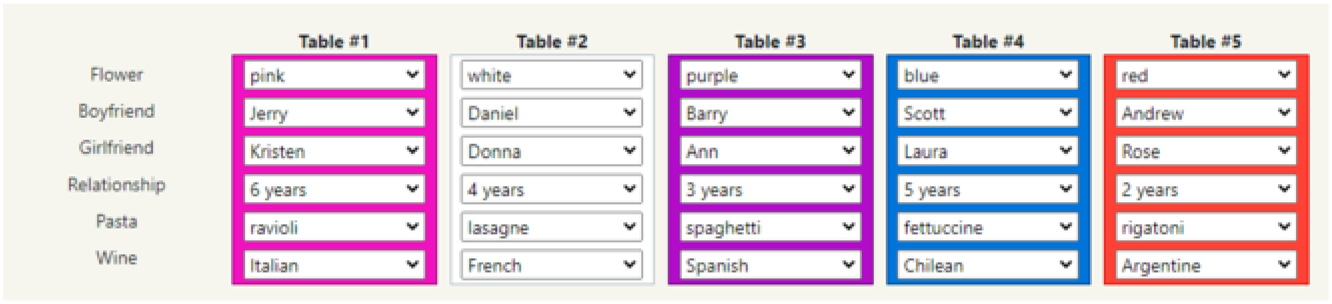
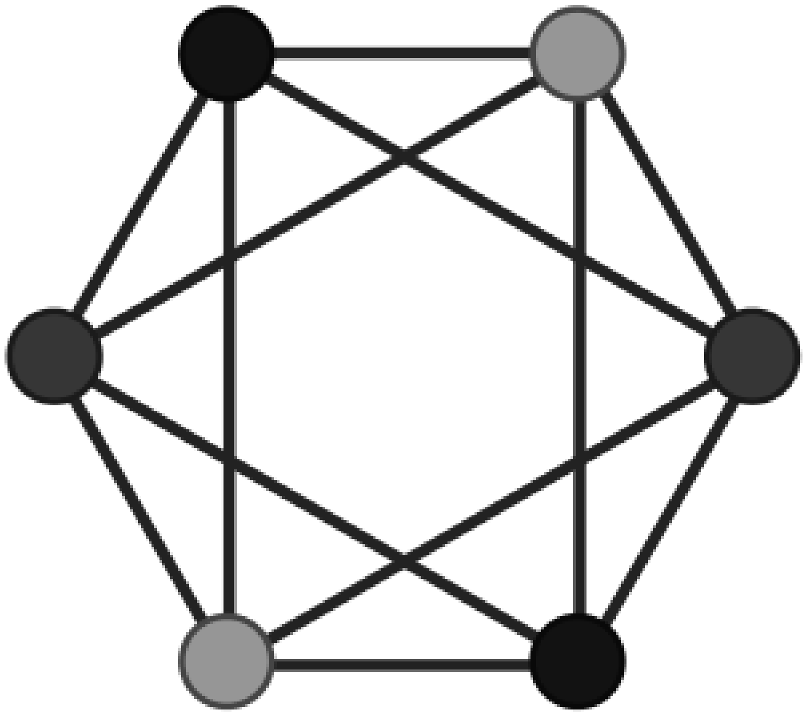


	Table #1	Table #2	Table #3	Table #4	Table #5
Flower	pink	white	purple	blue	red
Boyfriend	Jerry	Daniel	Barry	Scott	Andrew
Girlfriend	Kristen	Donna	Ann	Laura	Rose
Relationship	6 years	4 years	3 years	5 years	2 years
Pasta	ravioli	lasagne	spaghetti	fettuccine	rigatoni
Wine	Italian	French	Spanish	Chilean	Argentine

5 Graph coloring

5.1 Descrierea problemei

Consider the following graph and the following 3 colors: red, blue and green. Using propositional logic, formalize the problem of coloring the vertices of this graph so that each node has exactly one color associated with it and any two adjacent vertices are colored differently. Using Mace4, determine the existing models.



5.2 Rezolvare in Mace4

Cod:

```
% Saved by Prover9-Mace4 Version 0.5, December 2007.

set(ignore_option_dependencies). % GUI handles dependencies

if(Prover9). % Options for Prover9
  assign(max_seconds, 60).
end_if.

if(Mace4).   % Options for Mace4
  assign(domain_size, 2).
  assign(end_size, 2).
  assign(max_models, -1).
  assign(max_seconds, 60).
end_if.
```

```

formulas(assumptions).

%Colored Graph
%6 noduri, 3 culori disponibile: rosu, verde, albastru

%fiecare nod are o culoare
A_Red | A_Green | A_Blue.
B_Red | B_Green | B_Blue.
C_Red | C_Green | C_Blue.
D_Red | D_Green | D_Blue.
E_Red | E_Green | E_Blue.
F_Red | F_Green | F_Blue.

%Fiecare nod are o singura culoare
A_Red -> -A_Blue.
A_Red -> -A_Green.
A_Green -> -A_Blue.

B_Red -> -B_Blue.
B_Red -> -B_Green.
B_Green -> -B_Blue.

C_Red -> -C_Blue.
C_Red -> -C_Green.
C_Green -> -C_Blue.

D_Red -> -D_Blue.
D_Red -> -D_Green.
D_Green -> -D_Blue.

E_Red -> -E_Blue.
E_Red -> -E_Green.
E_Green -> -E_Blue.

F_Red -> -F_Blue.
F_Red -> -F_Green.
F_Green -> -F_Blue.

%Vecinii nu pot avea aceeasi culoare

A_Red -> -B_Red & -C_Red & -E_Red & -F_Red.
A_Green -> -B_Green & -C_Green & -E_Green & -F_Green.
A_Blue -> -B_Blue & -C_Blue & -E_Blue & -F_Blue.

B_Red ->  -C_Red & -D_Red & -F_Red.
B_Green ->  -C_Green & -D_Green & -F_Green.
B_Blue ->  -C_Blue & -D_Blue & -F_Blue.

C_Red ->  -D_Red & -E_Red.
C_Green ->  -D_Green & -E_Green.
C_Blue -> -D_Blue & -E_Blue.

D_Red -> -E_Red & -F_Red.
```

```
69  D_Green ->  -E_Green & -F_Green.
70  D_Blue ->  -E_Blue & -F_Blue.
71
72  E_Red ->  -F_Red.
73  E_Green ->  -F_Green.
74  E_Blue ->  -F_Blue.
75
76  end_of_list.
77
78  formulas(goals).
79
80  end_of_list.
```

5.3 Explicarea codului

Nodurile le-am notat cu A, B, C, D, E, incepand din coltul din dreapta sus si mergand in sensul acelor de ceasornic.

Primele linii din cod (22 - 52) sugereaza ca fiecare nod are una din culorile rosu, verde sau abastru, si doar una dintre aceastea. Faptul ca un nod are anumita culoare exclude faptul ca ar avea si celelalte culori.

Se cere ca nodurile adiacente sa nu aiba aceeasi culoare, asa ca pentru fiecare nod de anumita culoare am scris conditia ca nodurile adiacente nu pot avea aceeasi culoare (liniile 56-74).

5.4 Rezultate

```
interpretation( 3, [number = 1,seconds = 0], [
    relation(A_Blue, [0]),
    relation(A_Green, [0]),
    relation(A_Red, [1]),
    relation(B_Blue, [0]),
    relation(B_Green, [1]),
    relation(B_Red, [0]),
    relation(C_Blue, [1]),
    relation(C_Green, [0]),
    relation(C_Red, [0]),
    relation(D_Blue, [0]),
    relation(D_Green, [0]),
    relation(D_Red, [1]),
    relation(E_Blue, [0]),
    relation(E_Green, [1]),
    relation(E_Red, [0]),
    relation(F_Blue, [1]),
    relation(F_Green, [0]),
    relation(F_Red, [0])]).
interpretation( 3, [number = 2,seconds = 0], [
    relation(A_Blue, [0]),
    relation(A_Green, [0]),
    relation(A_Red, [1]),
    relation(B_Blue, [1]),
    relation(B_Green, [0]),
    relation(B_Red, [0]),
    relation(C_Blue, [0]),
    relation(C_Green, [1]),
    relation(C_Red, [0]),
    relation(D_Blue, [0]),
    relation(D_Green, [0]),
    relation(D_Red, [1]),
    relation(E_Blue, [1]),
    relation(E_Green, [0]),
    relation(E_Red, [0]),
    relation(F_Blue, [0]),
    relation(F_Green, [1]),
    relation(F_Red, [0])]).
interpretation( 3, [number = 3,seconds = 0], [
    relation(A_Blue, [0]),
    relation(A_Green, [1]),
    relation(A_Red, [0]),
    relation(B_Blue, [0]),
    relation(B_Green, [0]),
    relation(B_Red, [1]),
    relation(C_Blue, [1]),
    relation(C_Green, [0]),
    relation(C_Red, [0]),
    relation(D_Blue, [0]),
    relation(D_Green, [1]),
    relation(D_Red, [0]),
    relation(E_Blue, [0]),
    relation(E_Green, [0]),
    relation(E_Red, [1]),
    relation(F_Blue, [1]),
    relation(F_Green, [0]),
    relation(F_Red, [0])]).
interpretation( 3, [number = 4,seconds = 0], [
    relation(A_Blue, [0]),
    relation(A_Green, [1]),
    relation(A_Red, [0]),
    relation(B_Blue, [1]),
    relation(B_Green, [0]),
    relation(B_Red, [0]),
    relation(C_Blue, [0]),
    relation(C_Green, [0]),
    relation(C_Red, [1]),
    relation(D_Blue, [0]),
    relation(D_Green, [1]),
    relation(D_Red, [0]),
    relation(E_Blue, [1]),
    relation(E_Green, [0]),
    relation(E_Red, [0]),
    relation(F_Blue, [0]),
    relation(F_Green, [0]),
    relation(F_Red, [1])]).
interpretation( 3, [number = 5,seconds = 0], [
    relation(A_Blue, [1]),
    relation(A_Green, [0]),
    relation(A_Red, [0]),
    relation(B_Blue, [0]),
    relation(B_Green, [0]),
    relation(B_Red, [1]),
    relation(C_Blue, [0]),
    relation(C_Green, [1]),
    relation(C_Red, [0]),
    relation(D_Blue, [1]),
    relation(D_Green, [0]),
    relation(D_Red, [0]),
    relation(E_Blue, [0]),
    relation(E_Green, [0]),
    relation(E_Red, [1]),
    relation(F_Blue, [0]),
    relation(F_Green, [1]),
    relation(F_Red, [0])]).
interpretation( 3, [number = 6,seconds = 0], [
    relation(A_Blue, [1]),
    relation(A_Green, [0]),
    relation(A_Red, [0]),
    relation(B_Blue, [0]),
    relation(B_Green, [1]),
    relation(B_Red, [0]),
    relation(C_Blue, [0]),
    relation(C_Green, [0]),
    relation(C_Red, [1]),
    relation(D_Blue, [1]),
    relation(D_Green, [0]),
    relation(D_Red, [0]),
    relation(E_Blue, [0]),
    relation(E_Green, [1]),
    relation(E_Red, [0]),
    relation(F_Blue, [0]),
    relation(F_Green, [0]),
    relation(F_Red, [1])]).
```

Ruland Mace4 observam ca obtinem 6 modele care respecta cerintele.

6 Three goddesses

6.1 Descrierea problemei

Three goddesses were sitting in an old Indian temple. Their names were Truth (alwaystelling the truth), Lie (always lying), and Wisdom (sometimes lying). A visitor askedthe one on the left: "Who is sitting next to you?" "Truth", she answered. Then he askedthe one in the middle: "Who are you?" "Wisdom". Last, he asked the one on the right:"Who is your neighbour?" "Lie", she replied. And then it became clear who was who.

6.2 Rezolvare in Mace4

Cod:

```
% Saved by Prover9-Mace4 Version 0.5, December 2007.

set(ignore_option_dependencies). % GUI handles dependencies

if(Prover9). % Options for Prover9
  assign(max_seconds, 60).
end_if.

if(Mace4).   % Options for Mace4
  assign(domain_size, 2).
  assign(end_size, 2).
  assign(max_models, -1).
  assign(max_seconds, 60).
end_if.

formulas(assumptions).

%Sunt trei zeite pe care le vom numerota cu 1, 2, 3
%T - truth ; W- wisdom ; L- lie

%cel putin o zeita care spune adevarul, una care minte si o "wisdom"
T1 | T2| T3.
W1 | W2 | W3.
L1 | L2 | L3.

%doar o singura true
T1 -> -T2 & -T3.
T2 -> -T1 & -T3.
T3 -> -T1 & -T2.

%doar o singura wisdom
W1 -> -W2 & -W3.
W2 -> -W1 & -W3.
W3 -> -W1 & -W2.

%doar o singura lie
L1 -> -L2 & -L3.
```

```
38  L2 -> -L1 & -L3.
39  L3 -> -L1 & -L2.
40
41  %fiecare are doar o caracteristica
42  T1 -> -W1 & -L1.
43  T2 -> -W2 & -L2.
44  T3 -> -W3 & -L3.
45
46  W1 -> -T1 & -L1.
47  W2 -> -L2 & -T2.
48  W3 -> -T3 & -L3.
49
50  L1 -> -T1 & -W1.
51  L2 -> -T2 & -W2.
52  L3 -> -T3 & -W3.
53
54  L1 -> -A1.
55  L2 -> -A2.
56  L3 -> -A3.
57
58  T1 -> A1.
59  T2 -> A2.
60  T3 -> A3.
61
62  A1 <-> T2.
63  A2 <-> W2.
64  A3 <-> L2.
65
66  end_of_list.
67
68  formulas(goals).
69
70  end_of_list.
```

6.3 Explicarea codului

Sunt 3 zeite: "Lie", "Wisdom" si "Truth" pe care le-am numerotat de la 1 la 3 si le am prescurtat cu initiala numelui fiecareia. Ca exemplu, "L2" reprezinta, conform notatiilor alese, ca zeita 2 este Lie.

Liniile 22-24 sugereaza ca exista cel putin o zeita Lie, una Wisdom si una Truth. In continuare am transcris in logica propozitionala faptul ca exista doar cate una din cele 3 zeite, astfel faptul ca o zeita este, spre exemplu, Truth, exclude posibilitatea ca si celelalte 2 zeite sa fie tot Truth.

Intre liniile 42 si 52 am exprimat faptul ca o zeita poate avea doar una din cele trei caracteristici (Lie, Truth, Wisdom), certitudinea uneia excluzandu-le pe celelalte. In functie de caracteristica zeitelor, am stabilit valorile de adevar pentru afrmatiile lor, notate sub forma "Ax", unde "x" este numarul zeitei. Zeita Lie spune mereu afirmatii false, zeita Truth face mereu afirmatii adevarate, iar zeita Wisdom spune uneori adevarul dar alteori minte. Cazul afirmatiilor zeitei Wisdom nu a mai fost necesar sa fie tratat in cod.

Ultimele 3 randuri din cod reprezinta afirmatiile facute de cele 3 zeite.

6.4 Rezultate

```
interpretation( 2, [number = 1,seconds = 0], [
    relation(A1, [0]),
    relation(A2, [0]),
    relation(A3, [1]),
    relation(L1, [0]),
    relation(L2, [1]),
    relation(L3, [0]),
    relation(T1, [0]),
    relation(T2, [0]),
    relation(T3, [1]),
    relation(W1, [1]),
    relation(W2, [0]),
    relation(W3, [0])]).
```

Ruland Mace4 observam ca obtinem un singur model. Solutia problemei este: Zeita 1 spune uneori adevarul(Wisdom), zeita 2 este mincinoasa(Lie) si zeita 3 spune mereu adevarul(Truth).