Artificial intelligence - Project 2
- Prover9/Mace4 Puzzles -

Andriescu Antoino

04/12/2021

1 A Logical Labyrinth

1.1 Descriere problema

A prisoner is given by the King holding him the opportunity to improve his situation if he solves a puzzle. He is told there are nine rooms in the castle: one room contains a lady and the other eight contain a tiger or is empty. If the prisoner opens the door to the room containing the lady, he will marry her and get a pardon. If he opens a door to a tiger room though, he will be eaten alive. If he opens the door to an empty room, he will go on with staying in prison. The sign on the door containing a lady is true. The signs on the doors containing tigers are false. The signs on the empty rooms can be either true or false.

You studied the situation for a long while: “The problem is unsolvable!” “I know”laughed the king. “Now, at least give me a decent clue: is Room 8 empty or not?”. The king was decent enough to tell whether Room 8 was empty or not, and you are then able to deduce where the lady is. Which door to open in order to find the lady, marry her, and get half of the kingdom as compensation?

1.2 Indicii

Room 1: Lady is in an odd numbered room.

Room 2: This room is empty.

Room 3: Either sign 5 is right or sign 7 is wrong.

Room 4: Sign 1 is wrong.

Room 5: Either sign 2 or sign 4 is right.

Room 6: Sign 3 is wrong.

Room 7: Lady is not in room 1.

Room 8: This room contains a tiger and room 9 is empty.

Room 9: This room contains a tiger and sign 6 is wrong.

1.3 Implementare cod

Code:

```
% 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).

L1 | L2 | L3 |L4 | L5 | L6 | L7 | L8 | L9.
T1 | T2 | T3 |T4 | T5 | T6 | T7 | T8 | T9.
E1 | E2 | E3 |E4 | E5 | E6 | E7 | E8 | E9.

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).

L1 -> R1.
L2 -> R2.
L3 -> R3.
L4 -> R4.
L5 -> R5.
L6 -> R6.
L7 -> R7.
L8 -> R8.
L9 -> R9.

T1 -> -R1.
T2 -> -R2.
T3 -> -R3.
T4 -> -R4.
T5 -> -R5.
T6 -> -R6.
T7 -> -R7.
T8 -> -R8.
T9 -> -R9.

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.
```

```
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.

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.

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.

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).

-E8. %sau E8.

end_of_list.

formulas(goals).

end_of_list.
```

1.4 Explicarea codului

Prin notatiile L1-L9, T1-T9 si E1-E9 am stabilit starile posibilie ale celor noua camere. Prin L1 se intelege ca in camera 1 este o "lady", prin T1 se intelege ca in camera 1 este un tigru, si prin E1 se intelege ca prima camera este goala si tot asa pana la camera 9. De la linia 18 la 20 se afirma ca este cel putin o "lady", un tigru sau camera goala pentru fiecare dintre cele 9 camere.

De la linia 22 la 30 se afirma ca poate fi o singura "lady" in toate cele 9 camere. Spre exemplu, linia 22 descrie ca daca o "lady" este in camera 1 atunci ea nu poate fi si in celelalte camere de la 2 la 8.

In urmatoarea etapa descriem valorile de adevar ale mesajelor de pe fiecare usa, in functie de ce se afla in camera ("lady", tigru sau camera e goala). Daca intr-o camera este "lady", atunci mesajul de pe usa este adevarat, iar daca inauntru este tigru mesajul de pe usa este fals. Pentru camerele goale nu am notat nimic, deoarece nu stim valoarea de adevar a mesajelor de pe usile camerelor respective. Mesajele de pe usile camerelor sunt notate cu R1, R2, ..., R9.

Mai departe se noteaza faptul ca o camera nu poate avea mai mult de o caracteristica, adica daca in camera 1 este o "lady" atunci nu poate sa mai fie si un tigru, sau sa fie si goala in acelasi timp.

Mai departe vom scrie mesajele care apar pe fiecare usa. In plus ni se mai da un indiciu in text in legatura cu camera 8: aceasta este goala sau nu (E8 sau -E8).

1.5 Rezultate

Pentru rularea codului avem doua variante posibile: una in care camera 8 este goala si un in care camera 8 nu este goala.

Luand in considerare prima varianta, la rularea programului Mace4 vom avea 55 de modele posibile. Din aceste modele observam ca in 24 "lady" este in camera 1, in 4 modele este in camera 3, in 8 modele este in camera 4, in 4 modele in camera 5 si in 20 de modele in camera 7. Deoarece "lady" poate fi in mai multe camere atuci agentul nostru nu poate gasi camera in care se afla aceasta.

In situatia in care camera 8 nu este goala, dupa rularea programului Mace4 vom aveam 8 modele posibile. In toate cele 8, "lady" se afla in camera 7, in plus Prover9 poate demonstra ca L7 este adevarat.

In concluzie camera 8 nu poate fi goala (-E8) pentru ca agentul nostru sa poate gasi camera in care se afla "lady", deci rezultatul final este ca sunt 8 modele posibile, iar in toate acestea L7 este adevarat ("lady" se afla in camera 7).

2 The case of Boris and Dorothy Vampyre

2.1 Descriere problema

Transylvania is inhabited by both vampires and human, the vampires always lie and the human always tell the truth. However, half the inhabitants, both human and vampire, are insane and totally deluded in their beliefs - all true propositions they believe false and and all false propositions they believe true. The other half of the inhabitants are completely sane and totally accurate in their judgements - all true statements they know to be true and all false statements they know to be false. Sane humans and insane vampires both make only true statements: insane humans and sane vampires make only false statements.

In Transylvania it is illegal for humans and vampires to intermarry, hence any married couple there are either both human or both vampires.

2.2 Indicii

"It is important", said the Transylvanian chief of police to Inspector Craig, "not to let the last name of the suspects prejudice the issue".

Here are the answers they gave:

Boris Vampyre: We are both vampires.
Dorothy Vampyre: Yes, we are.
Boris Vampyre: We are alike, as far as our sanity goes.

What kind of couple are we dealing with?

2.3 Implementare cod

Code:

```
% 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, 3).
  assign(start_size, 3).
  assign(end_size, 3).
  assign(max_models, -1).
  assign(max_seconds, 60).
end_if.

```

```
if(Mace4).   % Additional input for Mace4
assign(max_models, -1).

   % Additional input for Mace4
assign(max_models, -1).
end_if.

formulas(assumptions).

%B = Boris.
%D = Dorothy.
%H = Human.
%V = Vampire.
%S = Sane.
%I = Insane.
%exemplu: BHI = Boris Human Insane(adica Boris este human si este insane)

BHS | BHI | BVS | BVI.
DHS | DHI | DVS | DVI.

BHS -> -BHI & -BVS & -BVI.
BHI -> -BHS & -BVS & -BVI.
BVS -> -BHI & -BHS & -BVI.
BVI -> -BHI & -BVS & -BHS.

DHS -> -DHI & -DVS & -DVI.
DHI -> -DHS & -DVS & -DVI.
DVS -> -DHI & -DHS & -DVI.
DVI -> -DHI & -DVS & -DHS.

BH | BV.
DH | DV.
BS | BI.
DS | DI.

BH <-> DH.
BV <-> DV.

BH -> BHS | BHI.
BV -> BVS | BVI.

DH -> DHS | DHI.
DV -> DVS | DVI.

BS -> BHS | BVS.
BI -> BHI | BVI.

DS -> DHS | DVS.
DI -> DHI | DVI.

BH & BS -> BHS.
BH & BI -> BHI.
BV & BS -> BVS.
BV & BI -> BVI.
```

```
DH & DS -> DHS.
DH & DI -> DHI.
DV & DS -> DVS.
DV & DI -> DVI.

BHS | BVI -> B1.
BHI | BVS -> -B1.

BHS | BVI -> B2.
BHI | BVS -> -B2.

DHS | DVI -> D1.
DHI | DVS -> -D1.

B1 <-> BV & DV.
D1 <-> B1.
B2 <-> (BS & DS) | (BI & DI).

end_of_list.

formulas(goals).

end_of_list.
```

2.4 Explicarea codului

In primul rand, pentru a intelege codul, trebuie intelese notatiile folosite: "B" si "D" sunt prescurtarile de la Boris, respectiv Dorothy, "H" si "V" sunt prescurtarile de la human, respectiv vampire, "S" si "I" sunt prescurtarile de la sane , respectiv insane. Cele 6 prescurtari pot fi combinate pentru a avea o anumita insemnatate dupa cum urmeaza:

- "BH" = Boris este human
- "BV" = Boris este vampire
- "BS" = Boris este sane
- "BI" = Boris este insane
- "DH" = Dorothy este human
- "DV" = Dorothy este vampire
- "DS" = Dorothy este sane
- "DI" = Dorothy este insane
- "BHS" = Boris este human si este sane
- "BHI" = Boris este human si este insane
- "BVS" = Boris este vampire si este sane
- "BVI" = Boris este vampire si este insane

- "DHS" = Dorothy este human si este sane
- "DHI" = Dorothy este human si este insane
- "DVS" = Dorothy este vampire si este sane
- "DVI" = Dorothy este vampire si este insane

In prima parte trebuie sa definim faptul ca cele doua personaje pot avea cel putin una dintre cele patru combinatii posibile pentru fiecare. Urmeaza, la liniile 37-45, sa impunem conditia ca un personaj sa aiba cel mult o combinatie.

Mai departe vom conditiona ca fiecare personaj sa aiba doar una dintre cele 4 caracteristici posibile (human, vampire, sane, insane), cu conditia ca nu pot fi in acelasi timp human si vampire, sau sane si insane. La liniile 52 si 53 exprimam faptul ca o familie este consituita doar din oameni sau vampiri.

In continuare vom, la liniile 67-75, vom scrie la ce rezultat duc combinatiile de caracteristici ale celor 2 personaje.

Tot ce mai ramane de facut este sa scriem raspunsurile celor doua personaje si sa descriem valorile de adevar ale raspunsurilor celor doua personaje (liniile 77-84). Daca Boris/Dorothy este human si sane sau vampire si insane, atunci raspunsurile sale sunt adevarate, dar daca este human si insane sau vampire si sane, atunci raspunsurile sale sunt false.

2.5 Rezultate

Ruland Mace4 obtinem un singur model posibil in care Boris si Dorothy sunt ambii vampire si in plus descoperim ca ambii sunt insane.

Presupunand ca ambele personaje sunt human, atunci primele doua raspunsuri sunt false, deci ar trebuie sa fie amandoi insane, ceea ce contrazice ultimul raspuns al lui Boris.

In concluzie singura solutie posibila este cea obtinuta de Mace4, care poate fi demonstrata si ruland Prover9.

3 Who is the spy?

3.1 Descriere problema

This case involves a trial of three defendants: A, B, and C. It was known at the outset of the trial that one of the three was a knight (he always told the truth), one a knave (he always lied), and the other was a spy who was normal (he sometimes lied and sometimes told the truth). The purpose of the trial was to find the spy.

First, A was asked to make a statement. He said either that C is a knave or that C is the spy, but we are not told which. The B said either that A is a knight, or that A is a knave, or that A was the spy, but we are not told which. Then C made a statement about B, and he said either B was a knight, or that B was a knave, or that B was the spy, but we are not told which. The judge then knew who the spy was and convicted him.

Which one is the spy - A, B, or C?

3.2 Implementare cod

Code:

```
% 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, 3).
  assign(start_size, 3).
  assign(end_size, 3).
  assign(max_seconds, 60).
end_if.

if(Mace4).   % Additional input for Mace4
assign(max_models, -1).
end_if.

formulas(assumptions).

%K = Knight.
%KN = Knave.
%S = Spy.

AK | AKN | AS.
BK | BKN | BS.
CK | CKN | CS.

AK -> -BK & -CK.
```

```
BK -> -AK & -CK.
CK -> -BK & -AK.

AKN -> -BKN & -CKN.
BKN -> -AKN & -CKN.
CKN -> -BKN & -AKN.

AS -> -BS & -CS.
BS -> -AS & -CS.
CS -> -BS & -AS.

AK -> A.
AKN -> -A.
AS -> A | -A.

BK -> B.
BKN -> -B.
BS -> B | -B.

CK -> C.
CKN -> -C.
CS -> C | -C.

A <-> CKN | CS.
B <-> AK | AKN | AS.
C <-> BK | BKN | BS.

end_of_list.

formulas(goals).

end_of_list.
```

3.3 Explicarea codului

Ca si la problema cu oamenii si vampirii pentru a intelege codul mai intai trebuie explicate notatiile folosite:

- "AK" = A is a knight
- "AKN" = A is a knave
- "AS" = A is a spy
- "BK" = B is a knight
- "BKN" = B is a knave
- "BS" = B is a spy
- "CK" = C is a knight
- "CKN" = C is a knave
- "CS" = C is a spy

Pentru a fi usor de explicat si inteles voi imparti codul in trei parti importante.

In prima parte a codului descriem ca cele 3 personaje (A, B, C) pot avea cel putin una din cele trei caracteristici (knight, knave, spy) (liniile 26-28) si ca pot avea cel mult una din cele trei caracteristici (liniile 30-40).

In a doua parte a codului descriem valorile de adevar ale afirmatiilor celor trei inculpati in functie de caracteristica pe care ar putea-o avea (knight - spune mereu adevarul; knave - nu spune niciodata adevarul; spy - nu stim daca spune adevarul sau nu).

In ultima parte ne ramane de scris fiecarea afirmatie prezentata de A, B si C.

3.4 Rezultate

Dupa rularea programului Mace4 obtinem un sigur model posibil, in care A este knave, B este spy si C este knight.

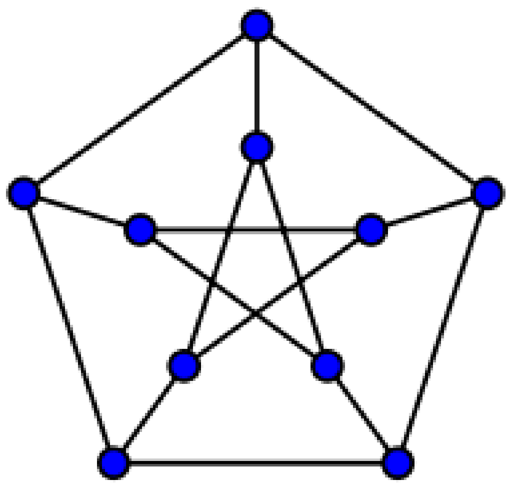
Rezultat:

```
interpretation( 3, [number = 1,seconds = 0], [
    relation(A, [0]),
    relation(AK, [0]),
    relation(AKN, [1]),
    relation(AS, [0]),
    relation(B, [1]),
    relation(BK, [0]),
    relation(BKN, [0]),
    relation(BS, [1]),
    relation(C, [1]),
    relation(CK, [1]),
    relation(CKN, [0]),
    relation(CS, [0])]).
```

4 Graf colorat

4.1 Descriere problema

Se consideră graful următor și următoarele 3 culori: roșu, albastru și galben. Folosind logica propozițională, formalizați problema de colorare a nodurilor acestui graf astfel încât fiecare nod să aibă asociată exact o singură culoare și oricare două noduri adiacente să fie colorate diferit. Utilizând Mace4, determinați modelele existente.



4.2 Implementare cod

Code:

```
% 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, 3).
  assign(start_size, 3).
  assign(end_size, 3).
  assign(max_models, -1).
  assign(max_seconds, 60).
end_if.

if(Mace4).   % Additional input for Mace4
assign(max_models, -1).
end_if.
```

```
formulas(assumptions).

%fiecare nod are cel putin o culoare.
N1(rosu) | N1(albastru) | N1(galben).
N2(rosu) | N2(albastru) | N2(galben).
N3(rosu) | N3(albastru) | N3(galben).
N4(rosu) | N4(albastru) | N4(galben).
N5(rosu) | N5(albastru) | N5(galben).
N6(rosu) | N6(albastru) | N6(galben).
N7(rosu) | N7(albastru) | N7(galben).
N8(rosu) | N8(albastru) | N8(galben).
N9(rosu) | N9(albastru) | N9(galben).
N10(rosu) | N10(albastru) | N10(galben).

%fiecare nod nu poate avea mai mult de o culoare.
N1(rosu) <-> -N1(albastru) & -N1(galben).
N1(albastru) <-> -N1(rosu) & -N1(galben).
N1(galben) <-> -N1(albastru) & -N1(rosu).

N2(rosu) <-> -N2(albastru) & -N2(galben).
N2(albastru) <-> -N2(rosu) & -N2(galben).
N2(galben) <-> -N2(albastru) & -N2(rosu).

N3(rosu) <-> -N3(albastru) & -N3(galben).
N3(albastru) <-> -N3(rosu) & -N3(galben).
N3(galben) <-> -N3(albastru) & -N3(rosu).

N4(rosu) <-> -N4(albastru) & -N4(galben).
N4(albastru) <-> -N4(rosu) & -N4(galben).
N4(galben) <-> -N4(albastru) & -N4(rosu).

N5(rosu) <-> -N5(albastru) & -N5(galben).
N5(albastru) <-> -N5(rosu) & -N5(galben).
N5(galben) <-> -N5(albastru) & -N5(rosu).

N6(rosu) <-> -N6(albastru) & -N6(galben).
N6(albastru) <-> -N6(rosu) & -N6(galben).
N6(galben) <-> -N6(albastru) & -N6(rosu).

N7(rosu) <-> -N7(albastru) & -N7(galben).
N7(albastru) <-> -N7(rosu) & -N7(galben).
N7(galben) <-> -N7(albastru) & -N7(rosu).

N8(rosu) <-> -N8(albastru) & -N8(galben).
N8(albastru) <-> -N8(rosu) & -N8(galben).
N8(galben) <-> -N8(albastru) & -N8(rosu).

N9(rosu) <-> -N9(albastru) & -N9(galben).
N9(albastru) <-> -N9(rosu) & -N9(galben).
N9(galben) <-> -N9(albastru) & -N9(rosu).

N10(rosu) <-> -N10(albastru) & -N10(galben).
N10(albastru) <-> -N10(rosu) & -N10(galben).
```

```
N10(galben) <-> -N10(albastru) & -N10(rosu).

%vecinii nodurilor nu pot avea aceeasi culoare.
N1(rosu) -> -N2(rosu) & -N6(rosu) & -N5(rosu).
N1(albastru) -> -N2(albastru) & -N6(albastru) & -N5(albastru).
N1(galben) -> -N2(galben) & -N6(galben) & -N5(galben).

N2(rosu) -> -N1(rosu) & -N3(rosu) & -N7(rosu).
N2(albastru) -> -N1(albastru) & -N3(albastru) & -N7(albastru).
N2(galben) -> -N1(galben) & -N3(galben) & -N7(galben).

N3(rosu) -> -N2(rosu) & -N4(rosu) & -N8(rosu).
N3(albastru) -> -N2(albastru) & -N4(albastru) & -N8(albastru).
N3(galben) -> -N2(galben) & -N4(galben) & -N8(galben).

N4(rosu) -> -N3(rosu) & -N5(rosu) & -N9(rosu).
N4(albastru) -> -N3(albastru) & -N5(albastru) & -N9(albastru).
N4(galben) -> -N3(galben) & -N5(galben) & -N9(galben).

N5(rosu) -> -N4(rosu) & -N1(rosu) & -N10(rosu).
N5(albastru) -> -N4(albastru) & -N1(albastru) & -N10(albastru).
N5(galben) -> -N4(galben) & -N1(galben) & -N10(galben).

N6(rosu) -> -N1(rosu) & -N8(rosu) & -N9(rosu).
N6(albastru) -> -N1(albastru) & -N8(albastru) & -N9(albastru).
N6(galben) -> -N1(galben) & -N8(galben) & -N9(galben).

N7(rosu) -> -N2(rosu) & -N9(rosu) & -N10(rosu).
N7(albastru) -> -N2(albastru) & -N9(albastru) & -N10(albastru).
N7(galben) -> -N2(galben) & -N9(galben) & -N10(galben).

N8(rosu) -> -N3(rosu) & -N6(rosu) & -N10(rosu).
N8(albastru) -> -N3(albastru) & -N6(albastru) & -N10(albastru).
N8(galben) -> -N3(galben) & -N6(galben) & -N10(galben).

N9(rosu) -> -N4(rosu) & -N6(rosu) & -N7(rosu).
N9(albastru) -> -N4(albastru) & -N6(albastru) & -N7(albastru).
N9(galben) -> -N4(galben) & -N6(galben) & -N7(galben).

N10(rosu) -> -N7(rosu) & -N8(rosu) & -N5(rosu).
N10(albastru) -> -N7(albastru) & -N8(albastru) & -N5(albastru).
N10(galben) -> -N7(galben) & -N8(galben) & -N5(galben).

end_of_list.

formulas(goals).

end_of_list.
```

4.3 Explicarea codului

Pnetru rezolvarea problemei am notat nodurile cu N1, N2, N3, N4, N5, N6, N7, N8, N9 si N10. Fiecare nod poate fi colorat cu rosu, albastru sau galben, dar nu poate avea mai mult de o culoarea in acelasi timp (liniile 23-74). In continuarea am scris ca nodurile adiacente de o anumita culoarea nu pot avea aceeasi culaore ca si nodul respectiv. Am repetat acest proces pentru fiecare nod in parte.

4.4 Rezultate

Dupa rularea programului Mace4 vor rezulta 120 de modele posibile pentru a colora graful nostru.

Tinand cont de faptul ca graful nostru este un graf Peterson, formula de calcul pentru numarul de colorari posibile in functie de numarul de culori utilizate este:

$$t * (t-1) * (t-2) * (t^7 - 12 * t^6 + 67 * t^5 - 230 * t^4 + 529 * t^3 - 814 * t^2 + 775 * t - 352)$$

***"t" reprezinta numarul de culori folosite pentru a colora ficare nod**

Avand in vedere ca in problema se cere sa folosim 3 culori, atunci t = 3. Dupa efectuarea calculelor rezultatul este 120, deci, in concluzie metoda de rezolvare propusa este corecta.

5 Secret Santa

5.1 Descriere problema

Five employees are side by side at their company secret Santa. Find out what each one is drinking, which department they work at and what was the gift they got.

Shirt: black, blue, green, red, white
Name: Cody, Jason, Riley, Steven, Tyler
Gift: book, chocolate, mug, notepad, tie
Department: HR, IT, marketing, RD, sales
Age: 23 years, 28 years, 35 years, 41 years, 50 years
Drink: coffee, juice, soft drink, tea, water

5.2 Indicii

Cody is the youngest employee.

The person gifted with a Book is exactly to the left of the one who works at the HR department.

In the fifth position is the person drinking Juice.

Riley is next to the 41-year-old employee.

The 35-year-old employee is at one of the ends.

The man wearing the Red shirt is somewhere between the person who received a Mug and the one drinking Soft drink, in that order.

The employee drinking Coffee is exactly to the left of the employee who got a Notepad as a gift.

The man drinking Tea is exactly to the right of the man wearing the Blue shirt.

The employee wearing the Green shirt is next to the 28-year-old.

Steven is exactly to the right of Cody.

In the second position is the person drinking Water.

The employee that works at the RD department is at the third position.

Tyler's gift was a Mug.

The oldest employee is at the fifth position.

The person drinking Soft drink is at the third position.

Riley is next to the person who got a Tie as a gift.

The youngest employee is somewhere between the person drinking Water and the oldest person, in that order.

Jason is exactly to the right of the man wearing the Black shirt.

Cody is next to the one drinking Soft drink.

The man wearing the Blue shirt is somewhere to the left of the employee that works at the Sales department.

The employee that works at the IT department was gifted a Notepad.

At the fourth position is the one drinking Tea.

5.3 Implementare cod

Code:

```
% 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).

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

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

differentFrom(c, d).
differentFrom(c, e).

differentFrom(d, e).

differentFrom(x, y) <-> differentFrom(y, x).

```

```
rightNeighbor(a,b).
rightNeighbor(b,c).
rightNeighbor(c,d).
rightNeighbor(d,e).

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

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

-rightNeighbor(c,a).
-rightNeighbor(c,b).
-rightNeighbor(c,c).
-rightNeighbor(c,e).

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

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

neighbor(x,y) <-> rightNeighbor(x,y) | rightNeighbor(y,x).

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

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

somewhereRight(c,d).
somewhereRight(c,e).

somewhereRight(d,e).

-somewhereRight(a,a).

-somewhereRight(b,a).
-somewhereRight(b,b).

-somewhereRight(c,a).
-somewhereRight(c,b).
```

```
-somewhereRight(c,c).

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

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

black(x) | blue(x) | green(x) | white(x) | red(x).
Cody(x) | Jason(x) | Riley(x) | Steven(x) | Tyler(x).
book(x) | chocolate(x) | mug(x) | notepad(x) | tie(x).
HR(x) | IT(x) | marketing(x) | RD(x) | sales(x).
23years(x) | 28years(x) | 35years(x) | 41years(x) | 50years(x).
Coffee(x) | Juice(x) | Softdrink(x) | Tea(x) | Water(x).

black(x) & black(y) -> -differentFrom(x,y).
blue(x) & blue(y) -> -differentFrom(x,y).
green(x) & green(y) -> -differentFrom(x,y).
red(x) & red(y) -> -differentFrom(x,y).
white(x) & white(y) -> -differentFrom(x,y).

Cody(x) & Cody(y) -> -differentFrom(x,y).
Jason(x) & Jason(y) -> -differentFrom(x,y).
Riley(x) & Riley(y) -> -differentFrom(x,y).
Steven(x) & Steven(y) -> -differentFrom(x,y).
Tyler(x) & Tyler(y) -> -differentFrom(x,y).

book(x) & book(y) -> -differentFrom(x,y).
chocolate(x) & chocolate(y) -> -differentFrom(x,y).
mug(x) & mug(y) -> -differentFrom(x,y).
notepad(x) & notepad(y) -> -differentFrom(x,y).
tie(x) & tie(y) -> -differentFrom(x,y).

HR(x) & HR(y) -> -differentFrom(x,y).
IT(x) & IT(y) -> -differentFrom(x,y).
marketing(x) & marketing(y) -> -differentFrom(x,y).
RD(x) & RD(y) -> -differentFrom(x,y).
sales(x) & sales(y) -> -differentFrom(x,y).

23years(x) & 23years(y) -> -differentFrom(x,y).
28years(x) & 28years(y) -> -differentFrom(x,y).
35years(x) & 35years(y) -> -differentFrom(x,y).
41years(x) & 41years(y) -> -differentFrom(x,y).
50years(x) & 50years(y) -> -differentFrom(x,y).

Coffee(x) & Coffee(y) -> -differentFrom(x,y).
Juice(x) & Juice(y) -> -differentFrom(x,y).
Softdrink(x) & Softdrink(y) -> -differentFrom(x,y).
Tea(x) & Tea(y) -> -differentFrom(x,y).
```

```
Water(x) & Water(y) -> -differentFrom(x,y).

%Cody is the youngest employee.
Cody(x) <-> 23years(x).

%The person gifted with a Book is exactly to the left of who works at the HR department.
book(x)&HR(y)->rightNeighbor(x,y).

%In the fifth position is the person drinking Juice.
Juice(e).

%Riley is next to the 41 years old employee.
Riley(x)&41years(y)->neighbor(x,y).

%The 35 years old employee is at one of the ends.
35years(a) | 35years(e).

%The man wearing the Red shirt is somewhere between the one who received a Mug and the one drinking Soft
mug(x)&red(y) -> somewhereRight(x,y).
Softdrink(x)&red(y)->somewhereRight(y,x).

%The employee drinking Coffee is exactly to the left of who got a Notepad as a gift.
Coffee(x) & notepad(y) -> rightNeighbor(x,y).

%The man drinking Tea is exactly to the right of the man wearing the Blue shirt.
Tea(x)&blue(y)->rightNeighbor(y,x).

%The employee wearing the Green shirt is next to the 28 years old.
green(x)&28years(y)->neighbor(x,y).

%Steven is exactly to the right of Cody.
Steven(x) & Cody(y) -> rightNeighbor(y,x).

%In the second position is the person drinking Water.
Water(b).

%The employee that works at the R&D department is at the third position.
RD(c).

%Tyler's gift was a Mug.
Tyler(x)<->mug(x).

%The oldest employee is at the fifth position.
50years(e).

%The person drinking Soft drink is at the third position.
Softdrink(c).

%Riley is next to the person who got a Tie as a gift.
Riley(x)&tie(y)->neighbor(x,y).

%The youngest employee is somewhere between the person drinking Water and the oldest person, in that ord
Water(x)&23years(y) -> somewhereRight(x,y).
50years(x)&23years(y)->somewhereRight(y,x).
```

```
197
198 %Jason is exactly to the right of the man wearing the Black shirt.
199 Jason(x) & black(y) -> rightNeighbor(y,x).
200
201 %Cody is next to the one drinking Soft drink.
202 Cody(x)&Softdrink(y)->neighbor(x,y).
203
204 %The man wearing the Blue shirt is somewhere to the left of the employee that works at the Sales departm
205 blue(x)&sales(y) -> somewhereRight(x,y).
206
207 %The employee that works at the IT department was gifted a Notepad.
208 IT(x) <->notepad(x).
209
210 %At the fourth position is the one drinking Tea.
211 Tea(d).
212
213 end_of_list.
214
215 formulas(goals).
216
217 end_of_list.
```

5.4 Explicarea codului

Am notat cei cinci angajati ai companiei in ordine crescatoare folosind literele a, b, c, d, e. Fiecarui angajat ii corespunde cate una din cele cinci litere. Pentru a arata ca fie fiecare angajat este diferit de celalalt am folosit "differentFrom(x, y)".

Pentru a putea scrie unele din cele 22 de indicii oferite de problema a trebuit sa creez "rightNeighbor(x, y)", "neighbor(x,y)" si "somewhereRight(x, y)".

- "rightNeighbor(x, y)" sugereaza faptul ca y este in dreapta lui x. Nu a fost nevoie sa creez ceva asemanator si pentru vecinul din stanga deoarece "rightNeighbor(y, x)" sugereaza faptul ca y este in stanga lui x.
- "neighbor(x,y)" sugereaza faptul ca x si y sunt vecini, fara sa stim daca in stanga sau in dreapta
- "somewhereRight(x, y)" indica faptul ca y este undeva in dreapta lui x, dar nu stim exact unde. Folosind aceeasi metoda ca si la "rightNeighbor(x, y)", nu a fost nevoie sa creez pentru persoana din stanga deoarece "somewhereRight(y, x)" indica prezenta lui y undeva in stanga lui x, fara sa stim exact unde.

Liniile 102-143 descriu ca fiecare angajat "x" nu poate avea nici mai mult, nici mai putin de o singura o valoare din fiecare set de caracteristici (shirt, name, gift, department, age, drink). Aici am folosit "differentFrom(x, y)" pentru a arata aceasta conditie.

De la linia 145 la 211 sunt scrise indiciile problemei folosind First Order Logic.

5.5 Rezultate

Dupa rularea programului Mace4 va rezulta un singur model. Rezultatul l-am descris printr-un tabel pentru a putea fi observat mai usor.

Model:

```
interpretation( 5, [number = 1,seconds = 0], [
    function(a, [0]),
    function(b, [1]),
    function(c, [2]),
    function(d, [3]),
    function(e, [4]),
    relation(23years(_), [0,0,0,1,0]),
    relation(28years(_), [0,0,1,0,0]),
    relation(35years(_), [1,0,0,0,0]),
    relation(41years(_), [0,1,0,0,0]),
    relation(50years(_), [0,0,0,0,1]),
    relation(Cody(_), [0,0,0,1,0]),
    relation(Coffee(_), [1,0,0,0,0]),
    relation(HR(_), [0,0,0,1,0]),
    relation(IT(_), [0,1,0,0,0]),
    relation(Jason(_), [0,1,0,0,0]),
    relation(Juice(_), [0,0,0,0,1]),
    relation(RD(_), [0,0,1,0,0]),
    relation(Riley(_), [0,0,1,0,0]),
    relation(Softdrink(_), [0,0,1,0,0]),
    relation(Steven(_), [0,0,0,0,1]),
    relation(Tea(_), [0,0,0,1,0]),
    relation(Tyler(_), [1,0,0,0,0]),
    relation(Water(_), [0,1,0,0,0]),
    relation(black(_), [1,0,0,0,0]),
    relation(blue(_), [0,0,1,0,0]),
    relation(book(_), [0,0,1,0,0]),
    relation(chocolate(_), [0,0,0,0,1]),
    relation(green(_), [0,0,0,1,0]),
    relation(marketing(_), [1,0,0,0,0]),
    relation(mug(_), [1,0,0,0,0]),
    relation(notepad(_), [0,1,0,0,0]),
    relation(red(_), [0,1,0,0,0]),
    relation(sales(_), [0,0,0,0,1]),
    relation(tie(_), [0,0,0,1,0]),
    relation(white(_), [0,0,0,0,1]),
    relation(differentFrom(_,_), [
        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(neighbor(_,_), [
        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(rightNeighbor(_,_), [
        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]),
    relation(somewhereRight(_,_), [
        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])]).
```

	Employee #1	Employee #2	Employee #3	Employee #4	Employee #5
Shirt	black	red	blue	green	white
Name	Tyler	Jason	Riley	Cody	Steven
Gift	mug	notepad	book	tie	chocolate
Department	marketing	IT	R&D	HR	sales
Age	35years	41years	28years	23years	50years
Drink	Coffee	Water	Soft drink	Tea	juice