Artificial Intelligence *Laboratory activity*

Name: Muresan Andreea, Rus Tudor
Group: 30231
Email: amuresan99@gmail.com, rustudor70@gmail.com

Teaching Assistant: Alexandru Ghiurutan alexandru.ghiurutan95@gmail.com

Contents

Chapter 1

A1: Search

Am ales sa studiem A* pentru fiecare dintre euristicile compatibile, pentru a putea afla care dintre ele este mai eficienta din punctul de vedere al timpului, al costului, respectiv al nodurilor expandate la fiecare mutare. Pentru aceasta am ales cateva layout-uri din biblioteca de maze-uri si am testat pentru fiecare heuristica in parte.

Compararea a 4 eurisitici diferite pentru algoritmul A*

Algoritmul A*

A* evaluează nodurile combinând distanta deja parcursa până la nod cu distanta estimata până la cea mai apropiata stare finala sau scop. Cu alte cuvinte, pentru un nod n oarecare, f(n) reprezintă costul estimat al celei mai bune soluții care trece prin n. Aceasta strategie are la baza o strategie simpla surprinsa de relatia: **f(n)=h(n)+g(n)** unde h(n) reprezinta euristica folosita iar g(n) reprezinta pasul la care ne aflam fata de nodul n. La o extremă, dacă h(n) este 0, atunci g(n) joaca rolul de f(n) , iar A * se transformă în algoritmul lui Dijkstra, care este garantat că va găsi o cale mai scurtă. La cealaltă extremă, dacă h(n)este foarte mare în raport cu g(n), atunci h(n) ia rolul de f(n) , iar A * se transformă în Greedy Best-First-Search.

Euclidean Heuristic

Deoarece distanța euclidiană este mai mică decât distanta Manhattan sau distanța diagonală, se vor obține în continuare cele mai scurte căi catre destinatie, dar A* va dura mai mult pentru a rula.

Euclidean Squared Heuristic

Când A* calculează f(n) = g(n) + h(n), pătratul distanței va fi mult mai mare decât costul și se va ajunge la o euristică supraestimată. Pentru distanțe mai mari, acest lucru se va apropia de extrema cand g(n)nu contribuie la f(n), iar A* se va degrada în Greedy Best-First-Search.

Octile Distance Heuristic

Aceasta este derivata din euristica Diagonal Distance, si este utilizata cand harta nu ne permite miscari pe diagonala. Se utilizeaza doua ponderi, D si D2 astfel: ponderea D este inmultita cu distanta Manhattan, iar din ponderea D2 se scande dublul ponderii D si se inmulteste cu minimul distantei pana la destinatie, pe axa Ox sau Oy. Ponderea D2 are valoarea sqrt(2) iar D2, valoarea 1.

Tie Break Heuristic

Aceasta euristica tine cont si de zidurile intalnite pe parcurs, folosindu-se de produsul vectorial dintre vectorul punctPlecare, punctDestinatie si vectorul punctCurent, punctDestinatie.

Analiza Tie Break vs Octile Distance

In urma testelor efectuate, am ajuns la concluzia ca, desi ambele euristici determina de cele mai multe ori, drumuri similare, euristica Tie Break expandeaza mai putine noduri ale arborelui de cautare.Iar ca si timp de executie, euristicile octileDistanceHeuristic si tieBreakHeuristic sunt echivalente.

```
(venv) D:\AN3sem1\IA\Proiect\search>python pacman.py -l tinyMaze -p SearchAgent -a fn=astar,heuristic=octileDistanceHeuristic
[SearchAgent] using function astar and heuristic octileDistanceHeuristic
[SearchAgent] using problem type PositionSearchProblem
Path found with total cost of 8 in 0.0 seconds
Search nodes expanded: 13
Pacman emerges victorious! Score: 502
Average Score: 502.0
Scores:        502.0
Win Rate:      1/1 (1.00)
Record:        Win
```

```
(venv) D:\AN3sem1\IA\Proiect\search>python pacman.py -l tinyMaze -p SearchAgent -a fn=astar,heuristic=tieBreakHeuristic
[SearchAgent] using function astar and heuristic tieBreakHeuristic
[SearchAgent] using problem type PositionSearchProblem
Path found with total cost of 8 in 0.0 seconds
Search nodes expanded: 15
Pacman emerges victorious! Score: 502
Average Score: 502.0
Scores:        502.0
Win Rate:      1/1 (1.00)
Record:        Win
```

Figure 1.1: tinyMaze

```
(venv) D:\AN3sem1\IA\Proiect\search>python pacman.py -l smallMaze -p SearchAgent -a fn=astar,heuristic=octileDistanceHeuristic
[SearchAgent] using function astar and heuristic octileDistanceHeuristic
[SearchAgent] using problem type PositionSearchProblem
Path found with total cost of 19 in 0.0 seconds
Search nodes expanded: 55
Pacman emerges victorious! Score: 491
Average Score: 491.0
Scores:        491.0
Win Rate:      1/1 (1.00)
Record:        Win
```

```
(venv) D:\AN3sem1\IA\Proiect\search>python pacman.py -l smallMaze -p SearchAgent -a fn=astar,heuristic=tieBreakHeuristic
[SearchAgent] using function astar and heuristic tieBreakHeuristic
[SearchAgent] using problem type PositionSearchProblem
Path found with total cost of 19 in 0.0 seconds
Search nodes expanded: 59
Pacman emerges victorious! Score: 491
Average Score: 491.0
Scores:        491.0
Win Rate:      1/1 (1.00)
Record:        Win
```

Figure 1.2: smallMaze

```
(venv) D:\AN3sem1\IA\Proiect\search>python pacman.py -l mediumMaze -p SearchAgent -a fn=astar,heuristic=octileDistanceHeuristic
[SearchAgent] using function astar and heuristic octileDistanceHeuristic
[SearchAgent] using problem type PositionSearchProblem
Path found with total cost of 68 in 0.0 seconds
Search nodes expanded: 223
Pacman emerges victorious! Score: 442
Average Score: 442.0
Scores:        442.0
Win Rate:      1/1 (1.00)
Record:        Win
```

```
(venv) D:\AN3sem1\IA\Proiect\search>python pacman.py -l mediumMaze -p SearchAgent -a fn=astar,heuristic=tieBreakHeuristic
[SearchAgent] using function astar and heuristic tieBreakHeuristic
[SearchAgent] using problem type PositionSearchProblem
Path found with total cost of 152 in 0.0 seconds
Search nodes expanded: 176
Pacman emerges victorious! Score: 358
Average Score: 358.0
Scores:        358.0
Win Rate:      1/1 (1.00)
Record:        Win
```

Figure 1.3: mediumMaze

```
(venv) D:\AN3sem1\IA\Proiect\search>python pacman.py -l bigMaze -p SearchAgent -a fn=astar,heuristic=octileDistanceHeuristic
[SearchAgent] using function astar and heuristic octileDistanceHeuristic
[SearchAgent] using problem type PositionSearchProblem
Path found with total cost of 210 in 0.1 seconds
Search nodes expanded: 554
Pacman emerges victorious! Score: 300
Average Score: 300.0
Scores:        300.0
Win Rate:      1/1 (1.00)
Record:        Win
(venv) D:\AN3sem1\IA\Proiect\search>python pacman.py -l bigMaze -p SearchAgent -a fn=astar,heuristic=tieBreakHeuristic
[SearchAgent] using function astar and heuristic tieBreakHeuristic
[SearchAgent] using problem type PositionSearchProblem
Path found with total cost of 210 in 0.1 seconds
Search nodes expanded: 460
Pacman emerges victorious! Score: 300
Average Score: 300.0
Scores:        300.0
Win Rate:      1/1 (1.00)
Record:        Win
```

Figure 1.4: bigMaze

```
(venv) D:\AN3sem1\IA\Proiect\search>python pacman.py -l openMaze -p SearchAgent -a fn=astar,heuristic=octileDistanceHeuristic
[SearchAgent] using function astar and heuristic octileDistanceHeuristic
[SearchAgent] using problem type PositionSearchProblem
Path found with total cost of 54 in 0.1 seconds
Search nodes expanded: 546
Pacman emerges victorious! Score: 456
Average Score: 456.0
Scores:        456.0
Win Rate:      1/1 (1.00)
Record:        Win
(venv) D:\AN3sem1\IA\Proiect\search>python pacman.py -l openMaze -p SearchAgent -a fn=astar,heuristic=tieBreakHeuristic
[SearchAgent] using function astar and heuristic tieBreakHeuristic
[SearchAgent] using problem type PositionSearchProblem
Path found with total cost of 54 in 0.0 seconds
Search nodes expanded: 218
Pacman emerges victorious! Score: 456
Average Score: 456.0
Scores:        456.0
Win Rate:      1/1 (1.00)
Record:        Win
```

Figure 1.5: openMaze

```
C:\Windows\System32\cmd.exe
C:\Users\rustu\OneDrive\Desktop\laburi\IA\1st\search>python pacman.py -l tinyMaze -z .5 -p SearchAgent -a fn=astar,heuristic=euclideanSquaredHeuris
tic
[SearchAgent] using function astar and heuristic euclideanSquaredHeuristic
[SearchAgent] using problem type PositionSearchProblem
Path found with total cost of 8 in 0.0 seconds
Search nodes expanded: 8
Pacman emerges victorious! Score: 502
Average Score: 502.0
Scores:        502.0
Win Rate:      1/1 (1.00)
Record:        Win

C:\Users\rustu\OneDrive\Desktop\laburi\IA\1st\search>python pacman.py -l tinyMaze -z .5 -p SearchAgent -a fn=astar,heuristic=euclideanHeuristic
[SearchAgent] using function astar and heuristic euclideanHeuristic
[SearchAgent] using problem type PositionSearchProblem
Path found with total cost of 8 in 0.0 seconds
Search nodes expanded: 13
Pacman emerges victorious! Score: 502
Average Score: 502.0
Scores:        502.0
Win Rate:      1/1 (1.00)
Record:        Win

C:\Users\rustu\OneDrive\Desktop\laburi\IA\1st\search>
```

Figure 1.6: tinyEuclidian

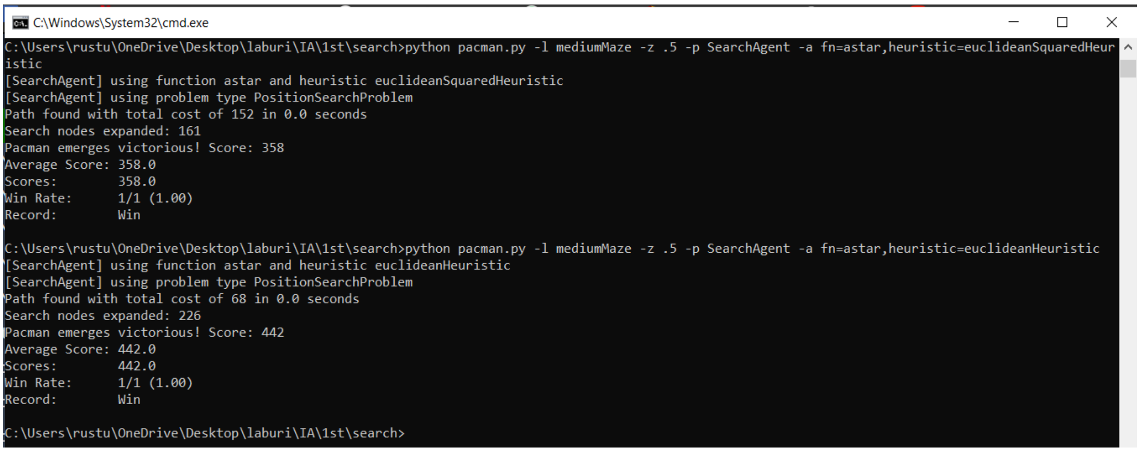

Figure 1.7: mediumEuclidian

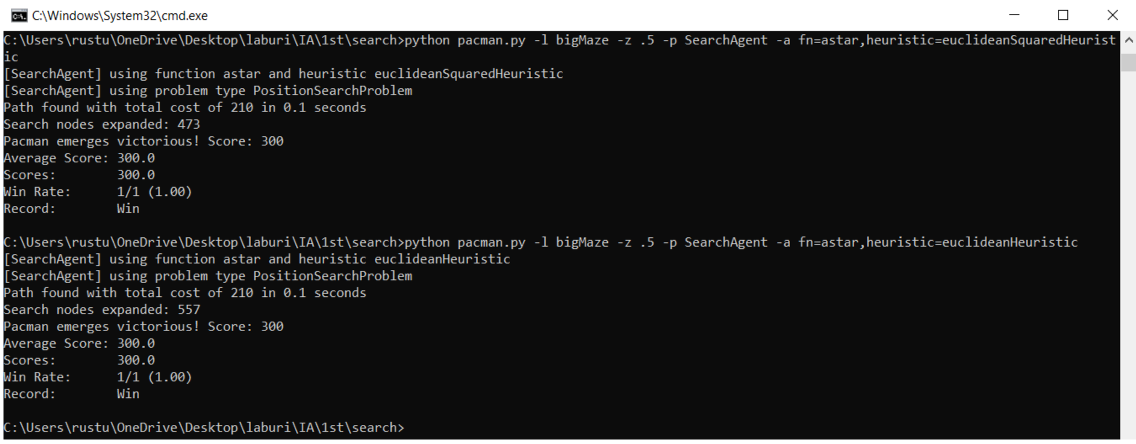

Figure 1.8: bigEuclidian

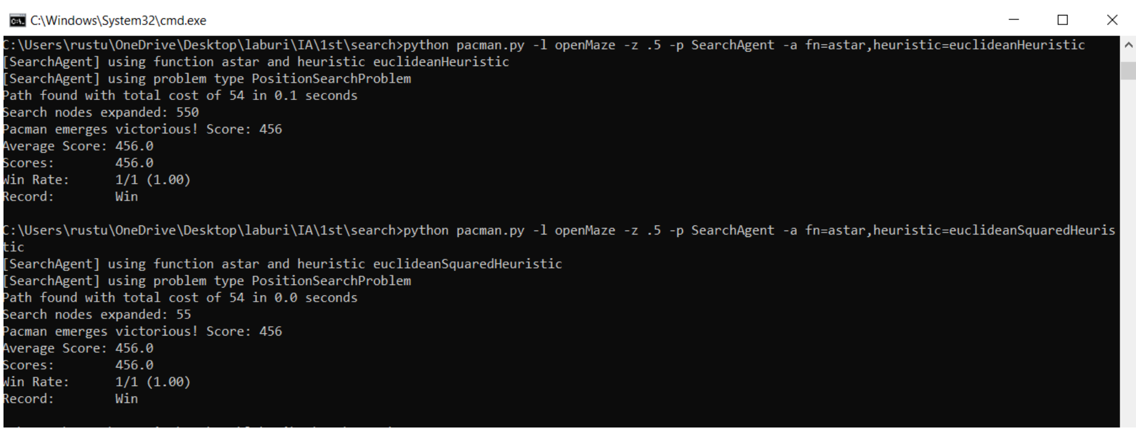

Figure 1.9: openEuclidian

Chapter 2

A2: Logics

1. The Zebra Puzzle - Einstein's riddle

Pe o strada se afla 5 case de diferite culori, fiecare casa are cate un locatar de nationalitate diferita. Fiecare locatar are o culoare preferata, o bautura preferata, fumeaza un anumit tip de tigari si au un animal de companie. Fiecare casa are cel putin una din urmatoarele: o nationalitate, un animal de companie, o culoare preferata, o bautura preferata, si un fel de tigari.

In care casa se prefera ginul si in care casa exista o zebra ca si animal de companie?

Indicii:

1. Englezul locuieste in casa rosie.
2. Spaniolul are un caine.
3. Danezul traieste in prima casa de pe stanga.
4. In casa galbena se fumeaza Marlboro.
5. Persoana care fumeaza Chesterfield locuieste langa individul care detine o pisica.
6. Norvegianul locuieste langa casa albastra.
7. Persoana care fumeaza Lucky Strike prefera sucul de portocale.
8. Persoana care fumeaza Winston detine ornitorinci.
9. Japonezul fumeaza Parliament.
10. Canadianul prefera ceaiul.
11. Casa in care se fumeaza Marlboro se afla langa cea in care exista un sarpe ca animal de companie.
12. Cafeaua este bautura preferata in casa verde.
13. Casa verde se afla in dreapta casei de culoare roz.
14. In casa din mijloc bautura preferata este laptele.

Rezolvarea problemei

Dupa traducerea conditiilor in logica propozitionala, am impus conditia ca fiecare casa sa aiba cel putin una dintre proprietati si fiecare proprietate sa se aplice unei singure case.

Dupa rularea folosind comanda mace4 -f zebra.in si interpretarea rezultatelor, se observa ca bautura preferata in prima casa este ginul iar zebra este animalul de companie care traieste in ultima casa.

```
============================== PROCESS NON-CLAUSAL FORMULAS ==========

% Formulas that are not ordinary clauses:
1 England(x) <-> Rosu(x) # label(non_clause).  [assumption].
2 Spain(x) <-> Caine(x) # label(non_clause).  [assumption].
3 Marlboro(x) <-> Galben(x) # label(non_clause).  [assumption].
4 Chesterfield(x) & Pisica(y) -> neighbors(x,y) # label(non_clause).  [assumption].
5 Denmark(x) & Albastru(y) -> neighbors(x,y) # label(non_clause).  [assumption].
6 Winston(x) <-> Ornitorinc(x) # label(non_clause).  [assumption].
7 LuckyStrike(x) <-> Suc(x) # label(non_clause).  [assumption].
8 Canada(x) <-> Ceai(x) # label(non_clause).  [assumption].
9 Japan(x) <-> Parliament(x) # label(non_clause).  [assumption].
10 Marlboro(x) & Sarpe(y) -> neighbors(x,y) # label(non_clause).  [assumption].
11 Cafea(x) <-> Verde(x) # label(non_clause).  [assumption].
12 Verde(x) & Roz(y) -> successor(y,x) # label(non_clause).  [assumption].
13 successor(x,y) <-> x + 1 = y & x < y # label(non_clause).  [assumption].
14 neighbors(x,y) <-> successor(x,y) | successor(y,x) # label(non_clause).  [assumption].
15 England(x) & England(y) -> x = y # label(non_clause).  [assumption].
16 Spain(x) & Spain(y) -> x = y # label(non_clause).  [assumption].
17 Canada(x) & Canada(y) -> x = y # label(non_clause).  [assumption].
18 Japan(x) & Japan(y) -> x = y # label(non_clause).  [assumption].
19 Denmark(x) & Denmark(y) -> x = y # label(non_clause).  [assumption].
20 Caine(x) & Caine(y) -> x = y # label(non_clause).  [assumption].
21 Ornitorinc(x) & Ornitorinc(y) -> x = y # label(non_clause).  [assumption].
22 Sarpe(x) & Sarpe(y) -> x = y # label(non_clause).  [assumption].
23 Zebra(x) & Zebra(y) -> x = y # label(non_clause).  [assumption].
24 Pisica(x) & Pisica(y) -> x = y # label(non_clause).  [assumption].
25 Gin(x) & Gin(y) -> x = y # label(non_clause).  [assumption].
26 Lapte(x) & Lapte(y) -> x = y # label(non_clause).  [assumption].
27 Suc(x) & Suc(y) -> x = y # label(non_clause).  [assumption].
28 Ceai(x) & Ceai(y) -> x = y # label(non_clause).  [assumption].
29 Cafea(x) & Cafea(y) -> x = y # label(non_clause).  [assumption].
30 Rosu(x) & Rosu(y) -> x = y # label(non_clause).  [assumption].
31 Albastru(x) & Albastru(y) -> x = y # label(non_clause).  [assumption].
32 Galben(x) & Galben(y) -> x = y # label(non_clause).  [assumption].
33 Roz(x) & Roz(y) -> x = y # label(non_clause).  [assumption].
34 Verde(x) & Verde(y) -> x = y # label(non_clause).  [assumption].
35 LuckyStrike(x) & LuckyStrike(y) -> x = y # label(non_clause).  [assumption].
36 Winston(x) & Winston(y) -> x = y # label(non_clause).  [assumption].
37 Marlboro(x) & Marlboro(y) -> x = y # label(non_clause).  [assumption].
38 Chesterfield(x) & Chesterfield(y) -> x = y # label(non_clause).  [assumption].
39 Parliament(x) & Parliament(y) -> x = y # label(non_clause).  [assumption].

============================== end of process non-clausal formulas ===
```

Figure 2.1: non-clausal formulas

```
interpretation( 5, [number=1, seconds=0], [

        relation(Albastru(_), [ 0, 1, 0, 0, 0 ]),

        relation(Cafea(_), [ 0, 0, 0, 0, 1 ]),

        relation(Caine(_), [ 0, 0, 0, 1, 0 ]),

        relation(Canada(_), [ 0, 1, 0, 0, 0 ]),

        relation(Ceai(_), [ 0, 1, 0, 0, 0 ]),

        relation(Chesterfield(_), [ 0, 1, 0, 0, 0 ]),

        relation(Denmark(_), [ 1, 0, 0, 0, 0 ]),

        relation(England(_), [ 0, 0, 1, 0, 0 ]),

        relation(Galben(_), [ 1, 0, 0, 0, 0 ]),

        relation(Gin(_), [ 1, 0, 0, 0, 0 ]),

        relation(Japan(_), [ 0, 0, 0, 0, 1 ]),

        relation(Lapte(_), [ 0, 0, 1, 0, 0 ]),

        relation(LuckyStrike(_), [ 0, 0, 0, 1, 0 ]),

        relation(Marlboro(_), [ 1, 0, 0, 0, 0 ]),

        relation(Ornitorinc(_), [ 0, 0, 1, 0, 0 ]),

        relation(Parliament(_), [ 0, 0, 0, 0, 1 ]),

        relation(Pisica(_), [ 1, 0, 0, 0, 0 ]),

        relation(Rosu(_), [ 0, 0, 1, 0, 0 ]),

        relation(Roz(_), [ 0, 0, 0, 1, 0 ]),

        relation(Sarpe(_), [ 0, 1, 0, 0, 0 ]),

        relation(Spain(_), [ 0, 0, 0, 1, 0 ]),

        relation(Suc(_), [ 0, 0, 0, 1, 0 ]),

        relation(Verde(_), [ 0, 0, 0, 0, 1 ]),
```

Figure 2.2: interpretation

```
        relation(Roz(_), [ 0, 0, 0, 1, 0 ]),

        relation(Sarpe(_), [ 0, 1, 0, 0, 0 ]),

        relation(Spain(_), [ 0, 0, 0, 1, 0 ]),

        relation(Suc(_), [ 0, 0, 0, 1, 0 ]),

        relation(Verde(_), [ 0, 0, 0, 0, 1 ]),

        relation(Winston(_), [ 0, 0, 1, 0, 0 ]),

        relation(Zebra(_), [ 0, 0, 0, 0, 1 ]),

        relation(neighbors(_,_), [
                           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(successor(_,_), [
                           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 ])
]).
```

Figure 2.3: interpretation

2. The murder mystery

Într-o seară a avut loc o crimă în casa unei familii formata din parinti si 2 copii,un baiat si o fata. Una dintre aceste patru persoane a ucis-o pe una dintre celelalte. Unul dintre membrii familiei a asistat la crimă. Celălalt l-a ajutat pe criminal. Acestea sunt lucrurile pe care le știm sigur:

Facts:

1. Martorul și cel care il ajuta pe criminal nu sunt de același sex.

2. Cea mai în vârstă și martorul nu sunt de același sex.

3. Aproape cea mai tânără persoană și victima nu sunt de același sex.

4. Cel care il ajuta pe criminal este mai în vârstă decât victima.

5. Tatăl este cel mai în vârstă membru al familiei.

6. Criminalul nu este cel mai tânăr membru al familiei.

Cine este criminalul?

Tratarea problemei incepe de la a spune ca totii membri familiei locuiesc intr o casa.Am stabilit o relatie de ordine intre persoane pe baza varstei si pe baza sexului pe care il au. Fiecare dintre ei poate sa fie victima. Fiecare persoana participa mai mult sau mai putin la crima.Pentru a descoperi cine a ucis pe cine am folosit mace4.

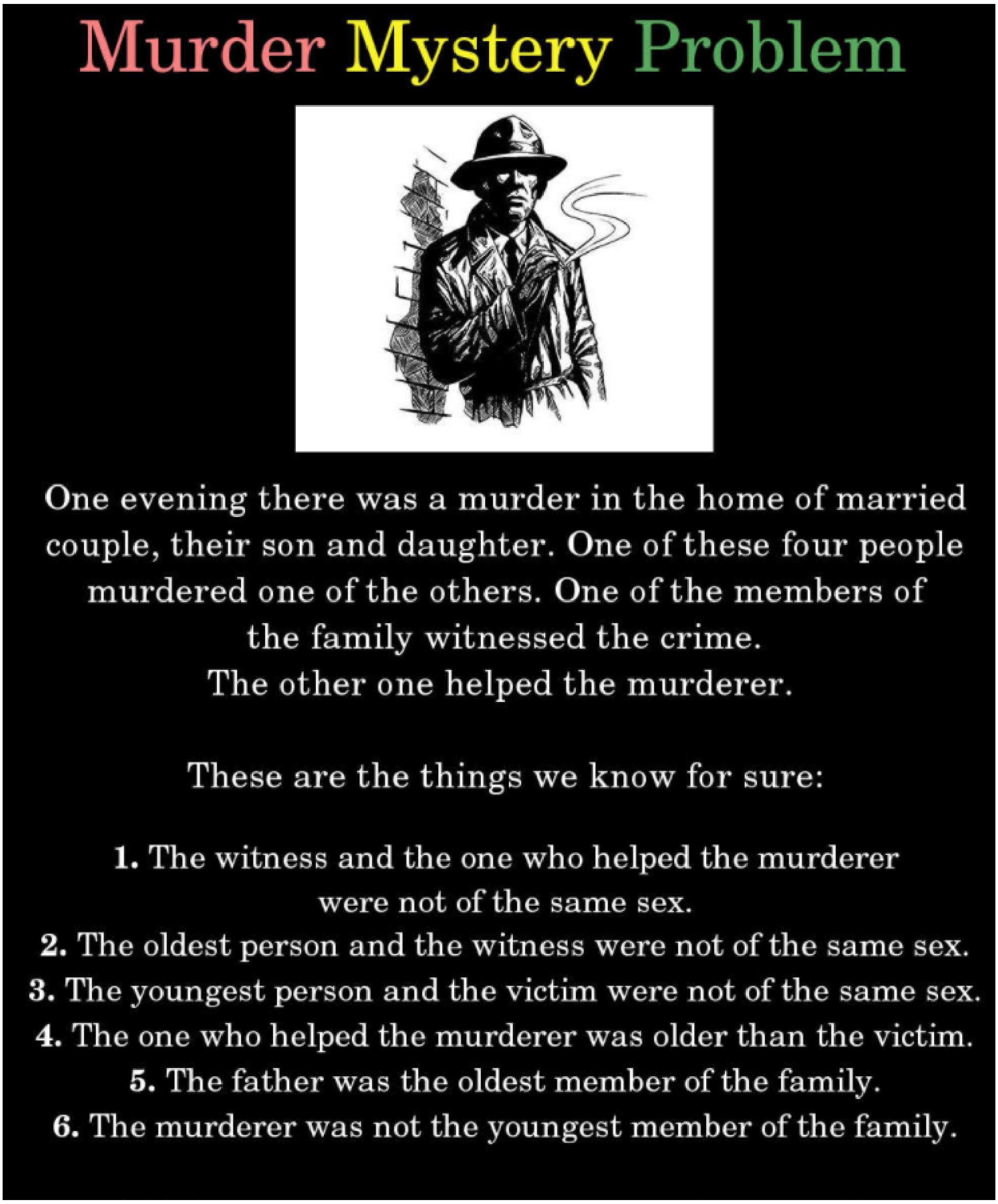

Figure 2.4: WHO IS THE MURDERER

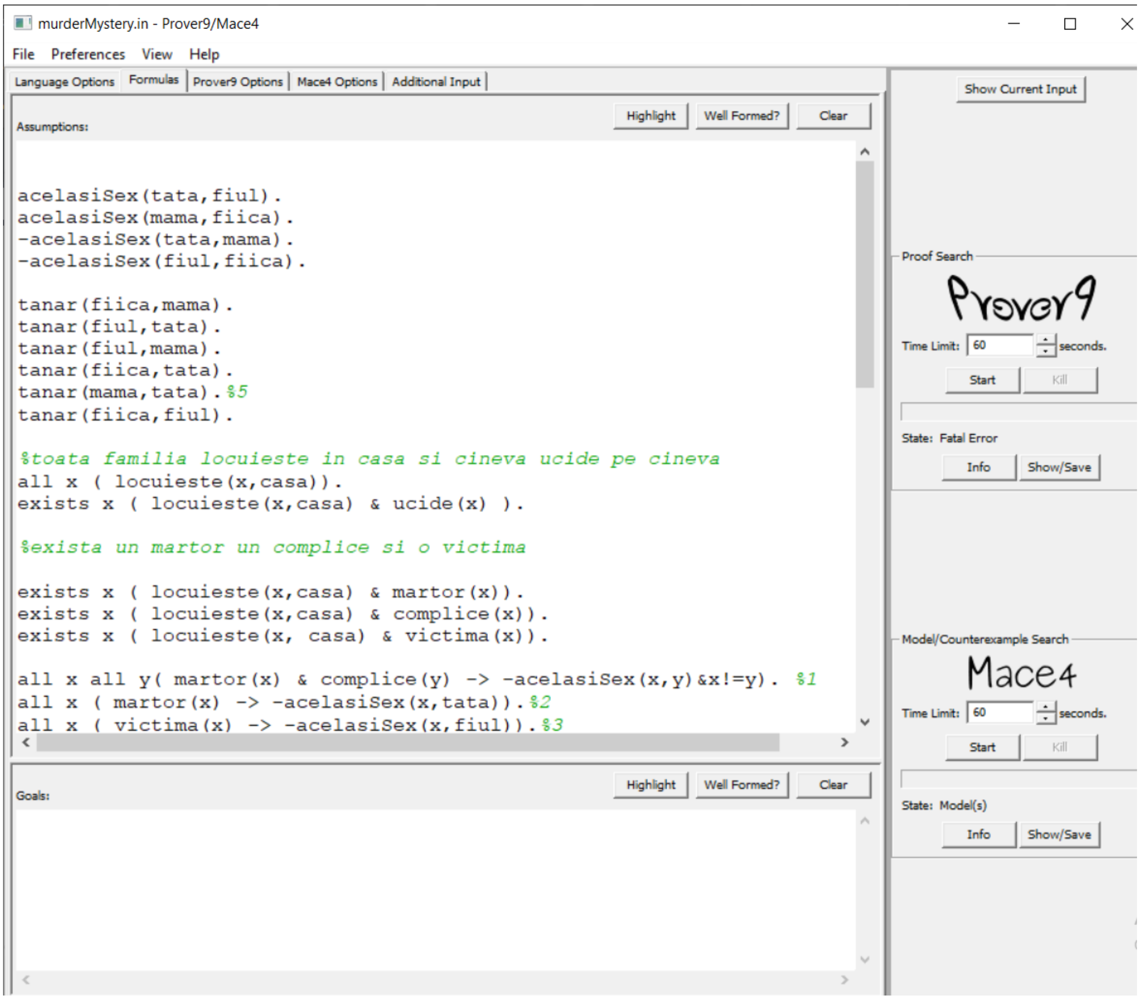

Figure 2.5: mace4

```
interpretation( 5, [number = 1,seconds = 0], [
    function(casa, [0]),
    function(fiica, [1]),
    function(fiul, [2]),
    function(mama, [3]),
    function(tata, [4]),
    function(c1, [2]),
    function(c2, [3]),
    function(c3, [4]),
    function(c4, [1]),
    relation(complice(_), [0,0,0,0,1]),
    relation(martor(_), [0,0,0,1,0]),
    relation(ucide(_), [0,0,1,0,0]),
    relation(victima(_), [0,1,0,0,0]),
    relation(acelasiSex(_,_), [
        0,0,0,0,0,
        0,0,0,0,0,
        0,0,0,0,0,
        0,1,0,0,0,
        0,0,1,0,0]),
    relation(locuieste(_,_), [
        1,0,0,0,0,
        1,0,0,0,0,
        1,0,0,0,0,
        1,0,0,0,0,
        1,0,0,0,0]),
    relation(tanar(_,_), [
        0,0,0,0,0,
        0,0,1,1,1,
        0,0,0,1,1,
        0,0,0,0,1,
        0,0,0,0,0])]).
```

Figure 2.6: INTERPRETATION

```
relation(captain(_), [ 0, 0, 1, 0, 0, 0 ]),

relation(official(_), [ 1, 0, 1, 0, 0, 1 ]),

relation(serve(_), [ 1, 0, 1, 0, 0, 1 ]),

relation(treasurer(_), [ 0, 0, 0, 0, 0, 1 ]),

relation(vice(_), [ 1, 0, 0, 0, 0, 0 ])
```

Figure 2.7: mace4 results

3. The Ladies of the Committee

Sase doamne sunt eligibile pentru posturile de: capitan, vice-capitan, si trezorier (in ordine descrescatoare a rangului), in clubul local de golf feminin. Intrebarea este, cum se pot ocupa posturile avand urmatoarele indicii:

1. Audrey nu va ocupa post daca Elaine e capitan sau Freda trezorier.
2. Betty nu va fi trezorier daca Cynthia e unul din oficiali.
3. Audrey nu va servi niciun post impreuna cu Betty si Elaine.
4. Freda nu va servi daca Elaine e si ea oficial.
5. Betty refuza sa fie vice-capitan.
6. Freda nu va servi, daca o depaseste in rang pe Audrey.
7. Cynthia nu va servi cu Audrey sau Betty, decat daca ea e capitanul.
8. Doris nu va servi decat daca Betty e capitan.
9. Betty nu va servi cu Doris, decat daca Elaine e si ea un oficial.
10. Elaine nu va servi decat daca ea sau Audrey sunt capitan.

Dupa ce am rulat folosind mace4 cu comanda mace4 -f ladiesofthecommittee.in rezulta urmatoarea solutie:
De unde rezulta ca Cynthia este capitan, Audrey este vice-capitan, iar Freda este trezorier.

4.The Labyrinth Guardians.

Mergi într-un labirint și dintr-o dată gasesti în față trei drumuri posibile: drumul din stânga este pavat cu aur, cel din fața ta este pavată cu marmură, în timp ce cel din dreapta este pavat cu pietre. Fiecare drum este protejată de un gardian.

Vorbești cu gardienii și iată ce îți spun:

Paznicul străzii de aur: ”Acest drum te va aduce direct in capat Mai mult, dacă drumul de piatra te duce în capat, atunci și drumul de marmura te duce în capat. ”

Paznicul străzii de marmură: ”Nici drumul de aurul, nici cel de piatra nu vor te duce in capat. ”

Paznicul străzii de piatră: ”Urmați drumul de aur și vei ajunge la capat, urmează drumul de marmura și te vei pierde ”.

Având în vedere că știi că toți gardienii sunt mincinoși, poți alege un drum fiind sigur că te va conduce spre centrul labirintului? Daca acesta este cazul,ce drum alegi?

Language

- g: "the gold road brings to the center"
- m: "the marble road brings to the center"
- s: "the stone road brings to the center"

Axioms

1. "The guardian of the gold street is a liar"

$$\neg(g \wedge (s \to m))$$

which can be simplified to obtain

$$\neg g \vee (s \wedge \neg m)$$

2. "The guardian of the marble street is a liar"

$$\neg(\neg g \wedge \neg s)$$

Figure 2.8: language

3. "The guardian of the stone street is a liar"

$$\neg(g \wedge \neg m)$$

which can be simplified to obtain

$$\neg g \vee m$$

Solution

g	m	s	2.7	2.8	2.9	2.7 $\wedge$ 2.8 $\wedge$ 2.9
1	1	1	0	1	1	0
1	1	0	0	1	1	0
1	0	1	1	1	0	0
1	0	0	0	1	0	0
0	1	**1**	1	1	1	**1**
0	1	0	1	0	1	0
0	0	**1**	1	1	1	**1**
0	0	0	1	0	1	0

Figure 2.9: results

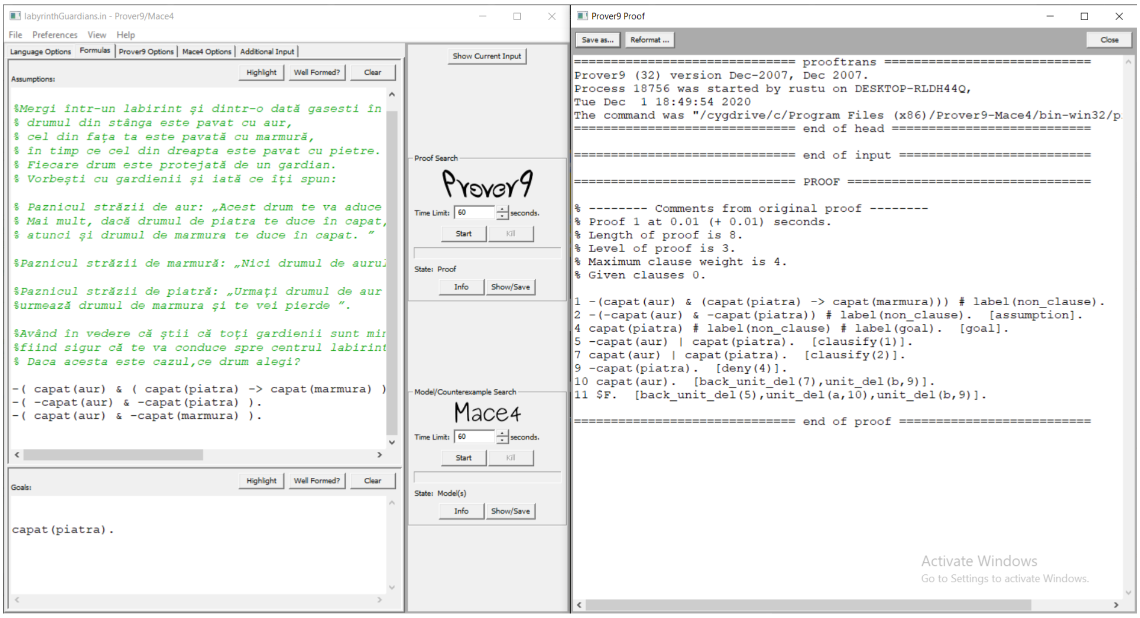

Figure 2.10: prover9 code

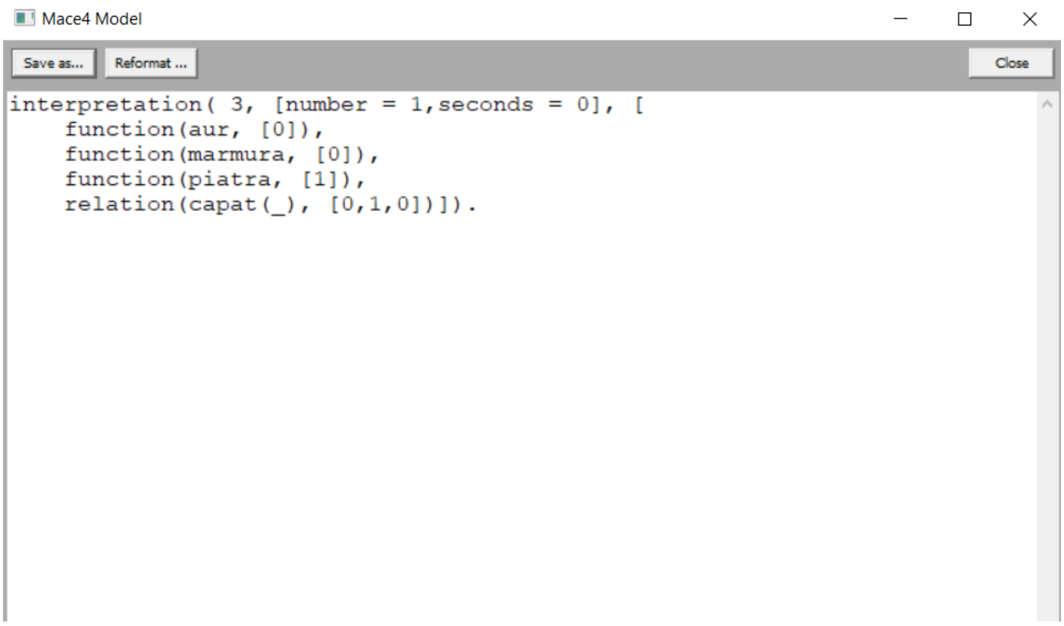

Figure 2.11: mace4 interpretation

Chapter 3

A3: Planning

Chapter 4

Bibliography

http://theory.stanford.edu/~amitp/GameProgramming/Heuristics.html
https://courses.cs.washington.edu/courses/cse473/14au/pacman/ps1/search.html
https://www.truthinsideofyou.org/riddle-murder-mystery-problem/2/
https://leanprover.github.io/logic_and_proof/first_order_logic.html
https://udel.edu/~os/riddle.html

Appendix A

Your original code

Search

Common code:

```
def aStarSearch(problem, heuristic=nullHeuristic);
    initialState = problem.getStartState()
    visitedState = []
    statesQueue = util.PriorityQueue()
    statesQueue.push((initialState, []), nullHeuristic(initialState, problem))
    cost = 0
    while not statesQueue.isEmpty():
state, actions = statesQueue.pop()
if problem.isGoalState(state):
return actions
if state not in visitedState:
successors = problem.getSuccessors(state)
for succ in successors:
coordinates = succ[0]
if coordinates not in visitedState:
directions = succ[1]
nActions = actions + [directions]
cost = problem.getCostOfActions(nActions) +
heuristic(coordinates, problem)
statesQueue.push((coordinates, actions +
[directions]), cost)
visitedState.append(state)
    return actions
    util.raiseNotDefined()
```

Tudor's code:

```
def euclideanHeuristic(position,problem,info={}):

y1=position
xy2=problem.goal
```

```
dx=abs(xy1[0]-xy2[0])
dy=abs(xy1[1]-xy2[1])
return sqrt(dx*dx+dy*dy )

def euclideanSquaredHeuristic(position,problem,info={}):

xy1=position
xy2=problem.goal
dx=abs(xy1[0]-xy2[0])
dy=abs(xy1[1]-xy2[1])
return (dx*dx+dy*dy)

### REFLEX AGENT

#doesn t work everytime but at least it makes some good scores and pass all the autograd

        distance = 0
        foodList = oldFood.asList()

        if action == 'Stop':
            #dont do that, its dangerous
            return -10000000

        for state in newGhostStates:# for each state of the ghost,
            # get position from current position posibilities to avoid that state
            if state.getPosition() == tuple(currentPos) and (state.scaredTimer == 0):
                return -10000000

        for food in foodList:#looking for food with manhattan heursitic
            distance = -1 * (manhattanDistance(food, currentPos))

            if (distance > maxDistance):
                maxDistance = distance
            #the max distance to the food
        return maxDistance
```

###MINIMAX FUNCTION

```
        def minMax(gameState, deepness, agent):
            #min function
            if agent >= gameState.getNumAgents():
                agent = 0
                deepness += 1
            if (deepness == self.depth or gameState.isWin() or gameState.isLose()):
                return self.evaluationFunction(gameState)
            elif (agent == 0):
                return maxValue(gameState, deepness, agent)
            else:
                return minValue(gameState, deepness, agent)

        def maxValue(gameState, deepness, agent):
            #max function
            output = ["meow", -float("inf")]
            pacActions = gameState.getLegalActions(agent)

            if not pacActions:
                return self.evaluationFunction(gameState)

            for action in pacActions:
                currState = gameState.generateSuccessor(agent, action)
                currValue = minMax(currState, deepness, agent + 1)
                if type(currValue) is list:
                    testVal = currValue[1]
                else:
                    testVal = currValue
                if testVal > output[1]:
                    output = [action, testVal]
            return output

        def minValue(gameState, deepness, agent):
            #min function
            output = ["meow", float("inf")]
            ghostActions = gameState.getLegalActions(agent)

            if not ghostActions:
                return self.evaluationFunction(gameState)

            for action in ghostActions:
                currState = gameState.generateSuccessor(agent, action)
                currValue = minMax(currState, deepness, agent + 1)
```

```
            if type(currValue) is list:
                testVal = currValue[1]
            else:
                testVal = currValue
            if testVal < output[1]:
                output = [action, testVal]
        return output

    outputList = minMax(gameState, 0, 0)
    return outputList[0]
```

Andreea's code:

```
def octileDistanceHeuristic(position,problem,info={}):
   dx = abs(position[0] - problem.goal[0])
   dy = abs(position[1] - problem.goal[1])
   return 1 * (dx + dy) + (math.sqrt(2) - 2 * 1) * min(dx, dy)

def tieBreakHeuristic(position, problem, info={}):
xy1 = position
xy2 = problem.goal
xy3 = problem.getStartState()
dx1 = xy1[0] - xy2[0]
dy1 = xy1[0] - xy2[1]
dx2 = xy3[0] - xy2[0]
dy2 = xy3[1] - xy2[1]
return ( abs(dx1 * dy2 - dx2 * dy1) )
```

Logics

Einstein's riddle :

```
set(integer_ring).

set(order_domain).
set(arithmetic).
assign(domain_size, 5).

%Pe o strada exista 5 case. Fiecare are cate un locatar de nationalitate diferita, culoa
%si in care casa se bea gin? Indicii:

%1. Englezul locuieste in casa rosie.
%2. Spaniolul detine un caine.
%3. Danezul traieste in prima casa de pe stanga.
%4. In casa galbena se fumeaza Marlboro.
%5. Persoana care fumeaza Chesterfield locuieste langa individul care detine o pisica.
%6. Norvegianul locuieste langa casa albastra.
%7. Persoana care fumeaza Winston detine ornitorinci.
%8. Persoana care fumeaza Lucky Strike prefera sucul de portocale.
%9. Canadianul bea ceai.
%10. Japonezul fumeaza Parliament.
%11. Casa in care se fumeaza Marlboro se afla langa cea in care exista un sarpe ca anima
%12. Cafeaua este preferata in casa verde.
%13. Casa verde se afla in dreapta casei roz.
%14. In casa din mijloc este preferat laptele.


formulas(assumptions).

%Indicii
England(x) <-> Rosu(x).

Spain(x) <-> Caine(x).

Denmark(0).

Marlboro(x) <-> Galben(x).

Chesterfield(x) & Pisica(y) -> neighbors(x,y).

Denmark(x) & Albastru(y) -> neighbors(x,y).

Winston(x) <-> Ornitorinc(x).

LuckyStrike(x) <-> Suc(x).

Canada(x) <-> Ceai(x).
```

```
Japan(x) <-> Parliament(x).

Marlboro(x) & Sarpe(y) -> neighbors(x,y).

Cafea(x) <-> Verde(x).

Verde(x) & Roz(y) -> successor(y,x).

Lapte(2).


% Definirea vecinilor si a succesorilor bazata pe indicii

successor(x,y) <-> x+1 = y & x < y.
neighbors(x,y) <-> successor(x,y) | successor(y,x).

%Fiecare casa are cel putin una dintre urmatoarele: o nationalitate, un animal, o bautur

England(x) | Spain(x) | Canada(x) | Japan(x) | Denmark(x).
Caine(x) | Ornitorinc(x) | Sarpe(x) | Zebra(x) | Pisica(x).
Gin(x) | Lapte(x) | Suc(x) | Ceai(x) | Cafea(x).
Rosu(x) | Albastru(x) | Galben(x) | Roz(x) | Verde(x).
LuckyStrike(x) | Winston(x) | Marlboro(x) | Chesterfield(x) | Parliament(x).

%Fiecare proprietate se aplica unei singure case.

England(x) & England(y) -> x = y.
Spain(x) & Spain(y) -> x = y.
Canada(x) & Canada(y) -> x = y.
Japan(x) & Japan(y) -> x = y.
Denmark(x) & Denmark(y) -> x = y.

Caine(x) & Caine(y) -> x = y.
Ornitorinc(x) & Ornitorinc(y) -> x = y.
Sarpe(x) & Sarpe(y) -> x = y.
Zebra(x) & Zebra(y) -> x = y.
Pisica(x) & Pisica(y) -> x = y.

Gin(x) & Gin(y) -> x = y.
Lapte(x) & Lapte(y) -> x = y.
Suc(x) & Suc(y) -> x = y.
Ceai(x) & Ceai(y) -> x = y.
Cafea(x) & Cafea(y) -> x = y.

Rosu(x) & Rosu(y) -> x = y.
Albastru(x) & Albastru(y) -> x = y.
Galben(x) & Galben(y) -> x = y.
Roz(x) & Roz(y) -> x = y.
Verde(x) & Verde(y) -> x = y.
```

```
LuckyStrike(x) & LuckyStrike(y) -> x = y.
Winston(x) & Winston(y) -> x = y.
Marlboro(x) & Marlboro(y) -> x = y.
Chesterfield(x) & Chesterfield(y) -> x = y.
Parliament(x) & Parliament(y) -> x = y.

end_of_list.
```

Mystery Murder:

```
set(ignore_option_dependencies).


asign(max_seconds,60).
asign(max_models,4).

formulas(assumptions).

acelasiSex(tata,fiul).
acelasiSex(mama,fiica).
-acelasiSex(tata,mama).
-acelasiSex(fiul,fiica).

tanar(fiica,mama).
tanar(fiul,tata).
tanar(fiul,mama).
tanar(fiica,tata).
tanar(mama,tata).%5
tanar(fiica,fiul).

%toata familia locuieste in casa si cineva ucide pe cineva
all x ( locuieste(x,casa)).
exists x ( locuieste(x,casa) & ucide(x) ).

%exista un martor un complice si o victima

exists x ( locuieste(x,casa) & martor(x)).
exists x ( locuieste(x,casa) & complice(x)).
exists x ( locuieste(x, casa) & victima(x)).

all x all y( martor(x) & complice(y) -> -acelasiSex(x,y)&x!=y). %1
all x ( martor(x) -> -acelasiSex(x,tata)).%2
all x ( victima(x) -> -acelasiSex(x,fiul)).%3
all x all y ( complice(x) & victima(y)-> -tanar(x,y)).%4
all x ( ucide(x)-> x!=fiica).%6

%casa nu poate sa fie nici martor nici victima nici complice nici criminal
%casa este neutra dpdv al sexului
%casa nu poate sa fie tanara
```

```
martor(x)->x!=casa.
complice(x)->x!=casa.
victima(x)->x!=casa.
acelasiSex(x,y)->(x!=casa & y!=casa).
tanar(x,y)->(x!=casa & y!=casa).
-ucide(casa).

%martorul, victima,criminalul si complicele sunt persoane diferite

martor(x) & ucide(y)->x!=y.
martor(x) & victima(y)->x!=y.
martor(x) & complice(y)->x!=y.
complice(x) & victima(y)->x!=y.
complice(x) & ucide(y)->x!=y.
victima(x) & ucide(y)->x!=y.

%mama,tata,fiul si fiica sunt persoane diferite

mama!=tata.
mama!=fiica.
mama!=fiul.
mama!=casa.
tata!=fiul.
tata!=fiica.
tata!=casa.
fiul!=fiica.
fiul!=casa.
fiica!=casa.

end_of_list.

formulas(goals).

end_of_list.
```

The ladies of the committee

```
assign(domain_size,6).

list(distinct).

[Audrey, Elaine, Betty, Freda, Cynthia, Doris].

end_of_list.

% (1) Audrey won't serve if Elaine is Captain, or if Freda is Treasurer.
% (2) Betty won't be Treasurer if Cynthia is one of the officials.
% (3) Audrey won't serve with both Betty and Elaine.
% (4) Freda won't serve if Elaine is also an official.
% (5) Betty refuses to be Vice-captain.
```

```
% (6) Freda won't serve if she outranks Audrey.
% (7) Cynthia won't serve with Audrey or Betty unless she is Captain.
% (8) Doris won't serve unless Betty is Captain.
% (9) Betty won't serve with Doris unless Elaine is also an official.
% (10) Elaine won't serve unless she or Audrey is Captain.

% Captain -> Vice-captain -> treasurer

% constant: A(Audrey), E(Elaine), B(Betty)
% constant: F(Freda), C(Cyntia), D(Doris)

% predicate: captain(x) meaning x is captain
% predicate: treasurer(x) meaning x is a treasurer
% predicate: official(x) meaning x is an official
% predicate: vice(x) meanig x is vice-president
% predicate: serve(x) (meaning x serves one of the functions)


formulas(assumptions).

 all x (serve(x) -> official(x)).
 all x (official(x) -> serve(x)).
 all x (official(x) -> captain(x) | vice(x) | treasurer(x)).
 all x (captain(x) -> -vice(x) & -treasurer(x)).
 all x (vice(x) -> -captain(x) & -treasurer(x)).
 all x (treasurer(x) -> -captain(x) & -vice(x)).



 captain(Audrey) -> -captain(Elaine) & -captain(Betty) & -captain(Freda) & -
 captain(Cynthia) & - captain(Doris).
 captain(Elaine) -> -captain(Audrey) & -captain(Betty) & -captain(Freda) & -
 captain(Cynthia) & -captain(Doris).
 captain(Betty) -> -captain(Elaine) & -captain(Audrey) & -captain(Freda) & -
 captain(Cynthia) & -captain(Doris).
 captain(Freda) -> -captain(Elaine) & -captain(Betty) & -captain(Audrey) & -
 captain(Cynthia) & -captain(Doris).
 captain(Cynthia) -> -captain(Elaine) & -captain(Betty) & -captain(Freda) & -
 captain(Audrey) & -captain(Doris).
 captain(Doris) -> -captain(Elaine) & -captain(Betty) & -captain(Freda) & -captain(Cynth
 & -captain(Audrey).



 vice(Audrey) -> -vice(Elaine) & -vice(Betty) & -vice(Freda) & -vice(Cynthia) & -vice(Do
 vice(Elaine) -> -vice(Audrey) & -vice(Betty) & -vice(Freda) & -vice(Cynthia) & -vice(Do
 vice(Betty) -> -vice(Elaine) & -vice(Audrey) & -vice(Freda) & -vice(Cynthia) & -vice(Do
 vice(Freda) -> -vice(Elaine) & -vice(Betty) & -vice(Audrey) & -vice(Cynthia) & -vice(Do
 vice(Cynthia) -> -vice(Elaine) & -vice(Betty) & -vice(Freda) & -vice(Audrey) & -vice(Do
 vice(Doris) -> -vice(Elaine) & -vice(Betty) & -vice(Freda) & -vice(Cynthia) & -vice(Aud
```

```
treasurer(Audrey) -> -treasurer(Elaine) & -treasurer(Betty) & -treasurer(Freda) & -
treasurer(Cynthia)  & -treasurer(Doris).
treasurer(Elaine) -> -treasurer(Audrey) & -treasurer(Betty) & -treasurer(Freda) & -
treasurer(Cynthia)  & -treasurer(Doris).
treasurer(Betty) -> -treasurer(Elaine) & -treasurer(Audrey) & -treasurer(Freda) & -
treasurer(Cynthia)  & -treasurer(Doris).
treasurer(Freda) -> -treasurer(Elaine) & -treasurer(Betty) & -treasurer(Audrey) & -
treasurer(Cynthia)  & -treasurer(Doris).
treasurer(Cynthia) -> -treasurer(Elaine) & -treasurer(Betty) & -treasurer(Freda) & -
treasurer(Audrey)  & -treasurer(Doris).
treasurer(Doris) -> -treasurer(Elaine) & -treasurer(Betty) & -treasurer(Freda) & -
treasurer(Cynthia)   & -treasurer(Audrey).



exists x (captain(x)).
exists x (vice(x)).
exists x (treasurer(x)).



-serve(Audrey) -> captain(Elaine) | treasurer(Freda).
-treasurer(Betty) -> official(Cynthia).
-serve(Audrey) -> serve(Betty) & serve(Elaine).
-serve(Freda) -> official(Elaine).
-vice(Betty).
treasurer(Audrey) -> -serve(Freda).
vice(Audrey) -> -serve(Freda) | treasurer(Freda).
captain(Audrey) -> -serve(Freda) | vice(Freda) | treasurer(Freda).
serve(Audrey) | serve(Betty) -> captain(Cynthia).
serve(Doris) -> captain(Betty).
serve(Betty) & serve(Doris) -> official(Elaine).
serve(Elaine) -> captain(Elaine) | captain(Audrey).



end_of_list.

formulas(goals).

end_of_list.
```

The Labyrinth Guardians

```
%Mergi într-un labirint și dintr-o dată gasesti în față trei drumuri posibile:
% drumul din stânga este pavat cu aur,
% cel din fața ta este pavată cu marmură,
```

```
% în timp ce cel din dreapta este pavat cu pietre.
% Fiecare drum este protejată de un gardian.
% Vorbești cu gardienii și iată ce îți spun:

% Paznicul străzii de aur: "Acest drum te va aduce direct in capat
% Mai mult, dacă drumul de piatra te duce în capat,
% atunci și drumul de marmura te duce în capat. "

%Paznicul străzii de marmură: "Nici drumul de aurul, nici cel de piatra nu vor te duce i

%Paznicul străzii de piatră: "Urmați drumul de aur și vei ajunge la capat,
%urmează drumul de marmura și te vei pierde ".

%Având în vedere că știi că toți gardienii sunt mincinoși, poți alege un drum
%fiind sigur că te va conduce spre centrul labirintului?
% Daca acesta este cazul,ce drum alegi?

-( capat(aur) & ( capat(piatra) -> capat(marmura) ) ).
-( -capat(aur) & -capat(piatra) ).
-( capat(aur) & -capat(marmura) ).
```