Department of Computer Science
Technical University of Cluj-Napoca

Introduction to Artificial Intelligence

Laboratory activity

Name: Adam Cristina-Ioana & Manolescu Matei
Group: 30234
Email: cristina_adam97@yahoo.com & mateimanolescu@yahoo.com

Assoc. Prof. dr. eng. Adrian Groza
Adrian.Groza@cs.utcluj.ro

Contents

Chapter 1

Rules and policies

Lab organisation.

1. Laboratory work is 20% from the final grade.

2. There are 3 deliverables in total.

3. Before each deadline, you have to send your work (latex documentation/code) at moodle.cs.utcluj.ro

Class:	Introducere in Inteligenta Artificiala
Enrollment key:	Iia2017-2018

4. *Laptop policy*: you can use your own laptop as long you have Linux. One goal of the laboratory is to increase your competency in Linux. It is **your** task to set static IPs:

IP:	192.168.1.51[1]
MASK:	255.255.255.0
GATEWAY:	192.168.1.2
DNS:	192.168.1.2
PROXY	192.168.1.2:3128

Wifi:	Network:	isg
	Password:	inteligentaartificiala

5. *Group change policy.* Maximum number of students in a class is 14.

6. *For students repeating the class*: A discussion for validating the previous grade is mandatory in the first week. I usually have no problem to validate your previous grades, as long you request this in the first week. Failing to do so, leads to the grade 1 for the laboratory work in the current semester.

Grading. Assessment aims to measure your knowledge and skills needed to function in realistic AI-related tasks. Assessment is based on your written report explaining the nature of the project, findings, and recommendations. Meeting the deadlines is also important. Your report is comparable to ones you would write if you were a consultant reporting to a client.

Grade inflation makes difficult to distinguish between students. It also discourages the best students to do their best. In my quest for "optimal ranking of the students", I do not use the following heuristics:

Table 1.1: Lab scheduling.

Activity	**Deadline**
Searching agents, linux, latex, python	W_1
Uninformed search	W_2
Informed Search	W_3
Adversial search	W_4
Propositional logic	W_5
First order logic	W_6
Inference in first order logic	W_7
Knowledge representation in first order logic	W_8
Classical planning	W_9
Contingent, conformant and probabilistic planning	W_{10}
Multi-agent planing	W_{11}
Modelling planning domains	W_{12}
Individual feedback to clarify the good/bad issues related to student activity/results during the semester.	W_{14}

- "He worked hard at the project". Our society do not like anymore individuals that are *trying*, but individual that *do* stuff. Such heuristic is not admissible in education, except the primary school.
- "I knew he could do much better". Such a heuristic is not admissible because it does not encourage you to spread yourself.
- 7 means that you: i) constantly worked during classes, ii) you proved competent to use the tool and its expressivity for a realistic scenario, iii) you understood theoretical concepts on which the tool rely on.
- 8, 9 mean that your code is large enough and the results proved by your experiments are significant.
- 10 means that you did very impressive work or more efficient that I expected or handled a lot of special cases for realistic scenarios.
- 5 means that you managed to develop something of your own, functional, with your own piece of code substantially different from the examples available.
- You obtain less than 5 in one of the following situations:
  1. few code written by yourself.
  2. too much similarity with the provided examples.
  3. non-seriosity (i.e. re-current late at classes, playing games, worked for other disciplines, poor/unprofessional documentation of your work, etc.)[2].
- You get 2 if you present the project but fail to submit the documentation or code. You get 1 if you do not present your project before the deadline. You get 0 for any line of code

[2]Consider non-seriosity as a immutable boolean value that is unconsciously activated in my brain when one of the above conditions occurs for the first time.

taken from other parts that appear in section *My own code.* For information on TUCN's regulations on plagiarism do consult the active norms.

If your grade is 0, 1, or 2, you do not satisfy the preconditions for participating to the written exam. The only possibility to increase your laboratory grade is to take another project in the next year, at the same class, and to make all the steps again.

However, don't forget that focus is on learning, not on grading.

Using Latex in your documentation. You have to show some competency on writing documentation in Latex. For instance, you have to employ various latex elements: lists, citations, footnotes, verbatim, maths, code, etc.

Plagiarism. Most of you consider plagiarism only a minor form of cheating. This is far from accurate. Plagiarism is passing off the work of others as your own to gain unfair advantage.

During your project presentation and documentation, I must not be left with doubts on which parts of your project are your work or not. Always identify both: 1) who you worked with and 2) where you got your part of the code or solution. You should sign the declaration of originality.

Describe clearly the starting point of your solution. List explicitly any code re-used in your project. List explicitly any help (including debugging help, design discussions) provided by others (including colleagues or teaching assistant). Keep in mind that it is your own project and not the teaching assistant's project. Learning by collaborating does remain an effective method. You can use it, but don't forget to mention any kind of support. Learning by exploiting various knowledge-bases developed by your elder colleagues remain also an effective method for "learning by example". When comparing samples of good and poor assignments submitted by your colleagues in earlier years try to identify which is better and why. You can use this repository of previous assignments, but don't forget to mention any kind of inspiration source.

The assignment is designed to be individual and to pose you some difficulties (both technological and scientific) for which you should identify a working solution by the end of the semester. Each semester, a distinct AI tool is assigned to two students. Your are encouraged to collaborate, especially during the the installation and example understanding phases (W_1-W_4). The quicker you get throughout these preparatory stages, the more time you have for your own project.

Class attendance. I expecte active participation at all activities. Keep in mind the exam can include any topic that was covered during class, explained on the board, or which emerged from discussions among participants. Missing lab assignments or midterm leads to minimum grade for that part. You are free to manage your laboratory classes - meaning that you can submit the project earlier - as long as you meet all the constraints and deadlines.

Chapter 2

A1: Search

Algoritmul A* este un algoritm care cauta drumul avand costul minim. Calculeaza drumul de la nodul start pana la succesor si apoi estimeaza costul de acolo pana la nodul de finish. La baza algoritmului sta formula f = g + h, unde g reprezinta costul de la inceput pana la succesor, iar h reprezinta euristica.

Algoritmul A* cu ponderi reprezinta un algoritm bazat pe algoritmul A*, cu diferenta ca se va adauga o pondere atat pentru euristica cat si pentru cost. Noua functie va deveni: f = pondere1 * g + pondere2 * h. Prin acest algoritm vrem sa vedem care componenta a functiei are cel mai mare impact in eficienta algoritmului pe labirinturi de diferite dimensiuni.

Concluzii

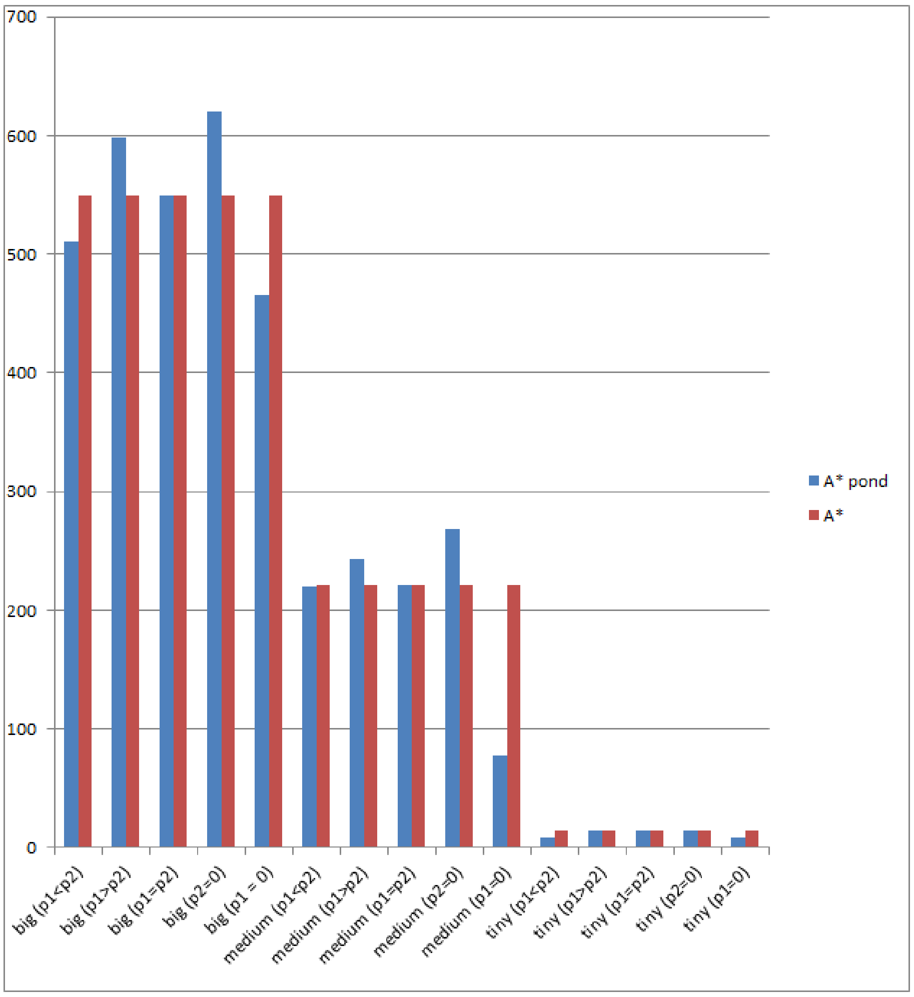
Pentru realizarea algoritmului A* cu ponderi am incercat sa transmitem cele 2 ponderi ca parametri in interiorul functiei aStarPonderiSearch(vezi anexa pentru cod). Cele 2 ponderi am vrut sa le initializam in momentul in care apelam functia din terminal. Pentru testarea pe mai multe cazuri ar fi trebuit sa definim o functie care ar fi apelat comanda din terminal de mai multe ori cu ponderi diferite, ducand la pornirea mai multor jocuri.

Deoarece nu am reusit sa implementam aceasta solutie am ales sa luam succesiv mai multe valori pentru ponderi, acoperind toate cazurile de interes. Rezultatele obtinute pot fi vizualizate in tabelul si graficul de mai jos.

Drept concluzie, putem observa faptul ca pe aceste labirinturi cea mai eficienta metoda este cea in care se tine cont exclusiv de euristica, adica ponderea lui g este 0.

Table 2.1: Rezultate A* cu ponderi vs A*.

Tip	**A* ponderi**	**A***	**Pond1**	**Pond2**	**Concluzie**
BigMaze	510	549	1	2	A* cu ponderi mai bun
BigMaze	598	549	4	2	A* mai bun
BigMaze	549	549	5	5	Acelasi rezultat
mediumMaze	220	221	1	2	A* cu ponderi mai bun
mediumMaze	243	221	4	2	A* mai bun
mediumMaze	221	221	5	5	Acelasi rezultat
tinyMaze	220	221	1	2	A* cu ponderi mai bun
tinyMaze	220	221	4	2	Acelasi rezultat
tinyMaze	220	221	5	5	Acelasi rezultat
bigMaze	620	549	5	0	Acelasi rezultat ca la uniform
mediumMaze	269	221	5	0	Acelasi rezultat ca la uniform
tinyMaze	15	14	5	0	Acelasi rezultat ca la uniform
bigMaze	466	549	0	5	Functioneaza ca BestFirst
mediumMaze	78	221	0	5	Functioneaza ca BestFirst
tinyMaze	8	14	0	5	Functioneaza ca BestFirst



Chapter 3

A2: Logics

Problema 85 - The Moskow puzzles - 359 mathematical recreations
Masha had to find the product of three numbers in order to calculate the volume of some soil.

She multiplied the first number by the second correctly and was about to multiply the result by the third number when she noticed that the second number had been written incorrectly. It was one-third larger than it should be.

To avoid recalculating, Masha decided it would be safe to merely lower the third number by one-third of itself-particularly since it equaled the second number.

"But you shouldn't do that," a girl friend said to Masha. "If you do, you will be wrong by 20 cubic yards."

"Why?" said Masha.

Why indeed? And what is the correct soil volume?

Pentru a fi un volum adevarat trebuie sa luam cele 3 valori corespunzatoare dimensiunilor acestuia diferite de 0(daca unul din ele e 0 atunci volumul devine arie).

Variabila d_partial_masha reprezinta produsul a doua dintre cele 3 valori, stiind ca una din ele este luata gresit. Mai exact b_gresit este mai mare cu o treime decat valoarea adevarata. Pornind de la acest produs volumul calculat de Masha presupune inmultirea acestuia cu cea de-a treia dimensiune a volumului real, dar luata cu o treime mai mica. Aceasta valoare este stocata in variabila c_modificat.

Stim ca doua dintre dimensiuni sunt egale.

Prietena Mashei afirma ca aceste calcule pe care le-a facut ii vor aduce un rezultat care difere cu 20 de yards fata de adevar.

Noi testam aceste lucruri utilizand comanda mace4 -f nume_fisier.in, ajungand la concluzia ca afirmatia data de prietena Mashei este adevarata, iar volumurile posibile sunt prezentate in tabelul de mai jos.

Concluzii
Valorile din tabelul de mai jos sunt pentru un domeniu fixat de 150. Dupa cum se poate observa s-au generat 21 de solutii, in care prietena Mashei a avut dreptate.

Table 3.1: Rezultate

Nr	**a**	**b**	**c**	**volum masha**	**volum real**
1	1	6	6	16	36
2	1	6	6	56	36
3	2	3	3	38	18
4	2	6	6	52	72
5	2	6	6	92	72
6	3	6	6	88	108
7	3	6	6	128	108
8	4	3	3	16	36
9	4	3	3	56	36
10	4	6	6	124	144
11	6	3	3	34	54
12	6	3	3	74	54
13	8	3	3	52	72
14	8	3	3	92	72
15	10	3	3	70	90
16	10	3	3	110	90
17	12	3	3	88	108
18	12	3	3	128	108
19	14	3	3	106	126
20	14	3	3	146	126
21	16	3	3	124	144

PROBLEMA CU ANIMALE SI PLANTE

Ursii, lupii, gainile, ramele si melcii sunt animale si exista cativa din fiecare.

De asemenea exista cereale si cerealele sunt plante. Fiecarui animal ii plac fie plantele, fie animalele mai mici decat el care mananca plante.

Ramele si melcii sunt mai mici decat gainile, care sunt mai mici decat lupii, care la randul lor sunt mai mici decat ursii.

Ursilor nu le place sa manance lupi sau cereale, in timp ce gainilor le plac ramele dar nu si melcii.

Ramle si melcii mananca plante.

Demonstreaza ca exista un animal caruia ii place sa manance un animal mancator de cereale.

In aceasta problema avem de demostrat ca exista un animal caruia ii place sa manance un animal mancator de cereale.

Din punct de vedere logic $\Rightarrow$

Prima conditie $\Leftrightarrow$

$\forall x.(urs(x) \lor lup(x) \lor gaina(x) \lor rama(x) \lor melc(x)) \to animal(x)$

$\exists x.urs(x) \land \exists x.lup(x) \land \exists x.gaina(x) \land \exists x.rama(x) \land \exists x.melc(x)$

A doua conditie $\Leftrightarrow$

$\exists x.cereala(x) \land \forall x.(cereala(x) \to planta(x))$

A treia conditie $\Leftrightarrow$

$\forall x.(animal(x) \to (\forall y.(planta(y) \to mananca(x,y)) \lor \forall z.(animal(z) \land mai\_mic(z,x) \land (\exists u(planta(u) \land mananca(z,u))) \to mananca(x,z))))$

A patra conditie $\Leftrightarrow$

$\forall x \forall y(rama(x) \land gaina(y) \to mai\_mic(x,y))$

$\forall x \forall y(melc(x) \land gaina(y) \to mai\_mic(x,y))$

$\forall x \forall y(gaina(x) \land lup(y) \to mai\_mic(x,y))$

$\forall x \forall y(lup(x) \land urs(y) \to mai\_mic(x,y))$

A cincea conditie $\Leftrightarrow$

$\forall x \forall y.(urs(x) \land (lup(y) \lor cereala(y)) \to \neg mananca(x,y)$

$\forall x \forall y.(gaina(x) \land rama(y) \to mananca(x,y))$

$\forall x \forall y.(gaina(x) \land melc(y) \to \neg mananca(x,y))$

A sasea conditie $\Leftrightarrow$

$\forall x(rama(x) \lor melc(x) \to \exists y(planta(y) \land mananca(x,y)))$

De demonstrat $\Leftrightarrow$

$\exists x \exists y.(animal(x) \land animal(y) \land mananca(x,y) \land (\forall z.(cereala(z) \to mananca(y,z)))$

Am rulat cu comanda ”prover9 -f nume_fisier.in” si am obtinut o demonstratie a cerintei cu proprietatile urmatoare:

```
================== PROOF ===================
Proof 1 at 0.00 (+ 0.01) seconds.
Length of proof is 74.
Level of proof is 10.
Maximum clause weight is 13.000.
Given clauses 44.
```

Chapter 4

A3: Planning

O versiune a problemei Cavalerului din The Moskow puzzles - 359 mathematical recreations

Elev in clasa a 4-a Kolya Sinichkin vrea sa mute un cal al unei table de sah din coltul stanga jos(a,1) in cel din dreapta sus(h,8).

Initial am luat tabla de sah cu 8 randuri si 8 coloane. In prima faza am pozitionat calul pe randul 4 si coloana 4 (d,4) si am constatat ca avem 8 miscari posibile, prin urmaare ar fi trebuit sa implementam 8 actiuni in fisierul "knight.pddl".

De exemplu urmatoare miscare din cele 8:

```
(:action move-one
   :parameters (?cal ?row ?old-row ?old-col ?new-row ?new-col)
   :precondition (and (cal-at ?cal ?old-row ?old-col)
                      (prev-row ?old-row ?new-row)
                      (next2-column ?old-col ?new-col)
                 )
    :effect (and (not (cal-at ?cal ?old-row ?old-col))
                 (cal-at ?cal ?new-row ?new-col)
            )
)
```

Avem mutarea in care calul se muta 2 coloane la dreapta si un rand in sus.

Am constatat ca nu este necesar sa cream toate cele 8 actiuni, este suficient sa existe doar 2. Actiuniile fiind simetrice atat pe orizontala, cat si pe verticala. Calul poate merge fie din 2 in 2 randuri si cate o coloana sau din 2 in 2 coloane si pe cate un rand.

Prin urmare putem rezuma totul la 2 actiuni, care pot fi analizate in anexa de mai jos.

In actiunea move am luat ca si parametri calul, linia initiala, coloana initiala, linia finala si coloana finala, cele care se vor afla la sfarsitul mutarii. Ca si preconditii este necesar sa verificam ca piesa de sah se afla pe pozitiile initiale, ca exista 2 coloane la stanga sau la dreapta si ca exista un rand mai sus sau mai jos. Ca si efect v-a rezulta mutarea calului de pe pozitia in care se afla initial intr una din pozitiile care satisface toate conditiile.

Similar va fi si pentru actiunea move1.

In Tabelul 4.1 este ilustrat un exemplu pentru tabla de sah, Calul fiind plasat in (d,4), iar cu * este reprezentat fiecare punct in care poate sa ajung calul pornind din locul in care se afla in momentul de fata.

In Tabelul 4.2 sunt reprezentate starile prin care trece Calul, care porneste din punctul (a,1) si ajunge in punctul (h,8). Putem observa ca s-a gasit o solutie care necesita 5 mutari astfel incat sa ajunga la final.

Table 4.1: EXemplu pentru cele 8 actiuni

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
a=1								
b=2				*		*		
c=3			*				*	
d=4					CAL			
e=5			*				*	
f=6				*		*		
g=7								
h=8								

Table 4.2: Rezulate finale

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
a=1	CAL				2			
b=2			1				3	
c=3								
d=4						4		
e=5								
f=6							5	
g=7								
h=8								FIN

```
./fd $LAB/knight.pddl $LAB/assignment3.pddl \
--heuristic "h=ff()" \
--search "astar(weight(h,2))"

Folosind comanda ./run.sh am obtinut urmatorul rezultat :

move cal row1 col1 row2 col3 (1)
move cal row2 col3 row1 col5 (1)
move cal row1 col5 row2 col7 (1)
move1 cal row2 col7 row4 col6 (1)
move1 cal row4 col6 row6 col7 (1)
move1 cal row6 col7 row8 col8 (1)
Plan length: 6 step(s).
Plan cost: 6
Initial state h value: 6.
Expanded 7 state(s).
Reopened 0 state(s).
Evaluated 23 state(s).
Evaluations: 23
Generated 30 state(s).
Dead ends: 0 state(s).
```

Bibliography

Appendix A

Your original code

This section should contain only code developed by you, without any line re-used from other sources. This section helps me to correctly evaluate your amount of work and results obtained. Including in this section any line of code taken from someone else leads to failure of IS class this year.

```
def aStarSearch(problem, heuristic=nullHeuristic):
  """
    Search the node that has the lowest combined cost and heuristic first.
    To test, try the following command:

    python pacman.py -l bigMaze -z .5 -p
    SearchAgent -a fn=astar,heuristic=manhattanHeuristic
  """

    time.sleep(2)
    fringe = util.PriorityQueue()
    fringe.push( (problem.getStartState(), [], 0),
    heuristic(problem.getStartState(), problem) )
    expanded = []

    while not fringe.isEmpty():
        node, actions, curCost = fringe.pop()

        if(not node in expanded):
            expanded.append(node)

            if problem.isGoalState(node):
                return actions

            for child, direction, cost in problem.getSuccessors(node):
                g = curCost + cost
                fringe.push((child, actions+[direction], curCost + cost),
                 g + heuristic(child, problem))
    return []
```

```
import sys
first_arg = sys.argv[1]
second_arg = sys.argv[2]


def aStarPonderiSearch(problem, heuristic=nullHeuristic, pond1=first_arg, pon

    fringe = util.PriorityQueue()
    fringe.push( (problem.getStartState(), [], 0),
    heuristic(problem.getStartState(), problem) )
    expanded = []

    print(pond1, pond2)

    while not fringe.isEmpty():
        node, actions, curCost = fringe.pop()

        if(not node in expanded):
            expanded.append(node)

            if problem.isGoalState(node):
                return actions

            for child, direction, cost in problem.getSuccessors(node):
                g = curCost + cost
                fringe.push((child, actions+[direction], curCost + cost),
                 pond1*g + pond2*heuristic(child, problem))
    return []
```

% PROBLEMA 85

```
set(arithmetic).
assign(domain_size, 150).
assign(max_models, -1).

formulas(assumptions).


 %dimensiunile reale ale volumului
 a != 0 .
 b != 0 .
 c != 0 .
 d = a * b * c.

 d_partial_masha = a * b_gresit.
 b1 * 3 = b.
 b_gresit = b + b1 .
 d_masha = d_partial_masa * c_modificat.
 c1 * 3 = c.
 c_modificat + c1 = c.
 c = b.

 d_masha + 20 = d | d + 20 = d_masha.

end_of_list.

formulas(goals).

end_of_list.
```

% PROBLEMA CU ANIMALE SI PLANTE

```
formulas(assumptions).

 all x (a_urs(x) -> animal(x)).
 all x (a_lup(x) -> animal(x)).
 all x (a_gaina(x) -> animal(x)).
 all x (a_rama(x) -> animal(x)).
 all x (a_melc(x) -> animal(x)).
 all x (a_cereala(x) -> planta(x)).

 exists x a_urs(x).
 exists x a_lup(x).
 exists x a_gaina(x).
 exists x a_rama(x).
 exists x a_melc(x).
 exists x a_cereala(x).

% Fiecarui animal ii plac fie plantele, fie animalele mai mici decat el care

 all x (animal(x) -> (all y (planta(y) -> mananca(x,y)))
                | (all z (animal(z) & mai_mic(z,x) & (exists u (planta(u) & m

% Ramele si melcii sunt mai mici decat gainile, care sunt mai mici decat lupi
% sunt mai mici decat ursii.

 all x all y (a_rama(x) & a_gaina(y) -> mai_mic(x,y)).
 all x all y (a_melc(x) & a_gaina(y) -> mai_mic(x,y)).
 all x all y (a_gaina(x) & a_lup(y) -> mai_mic(x,y)).
 all x all y (a_lup(x) & a_urs(y) -> mai_mic(x,y)).

% Ursilor nu le place sa manance lupi sau cereale, in timp ce gainilor le pla

 all x all y (a_gaina(x) & a_rama(y) -> mananca(x,y)).
 all x all y (a_urs(x) & a_lup(y) -> -mananca(x,y)).
```

```
  all x all y (a_urs(x) & a_cereala(y) -> -mananca(x,y)).
  all x all y (a_gaina(x) & a_melc(y) -> -mananca(x,y)).

% Ramle si melcii mananca plante.

  all x (a_rama(x) -> (exists y (planta(y) & mananca(x,y)))).
  all x (a_melc(x) -> (exists y (planta(y) & mananca(x,y)))).

 end_of_list.


 formulas(goals).

% Demonstreaza ca exista un animal caruia ii place sa manance un animal manca

  exists x exists y (animal(x) & animal(y) & mananca(x,y) &
  (all z (a_cereala(z) -> mananca(y,z)))).

 end_of_list.



                                                   %Assignment3.pddl
 (define (problem problema)
   (:domain knight)
   (:objects
      cal
      row1 row2 row3 row4 row5 row6 row7 row8
      col1 col2 col3 col4 col5 col6 col7 col8)
   (:init
     (next-row row1 row2)          (next-column col1 col2)
     (next-row row2 row3)          (next-column col2 col3)
     (next-row row3 row4)          (next-column col3 col4)
     (next-row row4 row5)          (next-column col4 col5)
     (next-row row5 row6)          (next-column col5 col6)
     (next-row row6 row7)          (next-column col6 col7)
     (next-row row7 row8)          (next-column col7 col8)

     (next-row row2 row1)          (next-column col2 col1)
     (next-row row3 row2)          (next-column col3 col2)
     (next-row row4 row3)          (next-column col4 col3)
     (next-row row5 row4)          (next-column col5 col4)
     (next-row row6 row5)          (next-column col6 col5)
     (next-row row7 row6)          (next-column col7 col6)
     (next-row row8 row7)          (next-column col8 col7)

     (next2-row row1 row3)         (next2-column col1 col3)
     (next2-row row2 row4)         (next2-column col2 col4)
     (next2-row row3 row5)         (next2-column col3 col5)
     (next2-row row4 row6)         (next2-column col4 col6)
     (next2-row row5 row7)         (next2-column col5 col7)
```

```
    (next2-row row6 row8)         (next2-column col6 col8)

    (next2-row row3 row1)         (next2-column col3 col1)
    (next2-row row4 row2)         (next2-column col4 col2)
    (next2-row row5 row3)         (next2-column col5 col3)
    (next2-row row6 row4)         (next2-column col6 col4)
    (next2-row row7 row5)         (next2-column col7 col5)
    (next2-row row8 row6)         (next2-column col8 col6)

    (cal-at cal row1 col1)

)
  (:goal
   (cal-at cal row8 col8)))
```

%knight.pddl

```
    (define (domain knight)
 (:predicates (cal-at ?cal ?r ?c)
              (next-row ?r1 ?r2)
              (next-column ?c1 ?c2)
              (next2-column ?ca ?cb)
              (next2-row ?c1 ?c2))

        (:action move
                :parameters (?cal ?or ?oc ?nr ?nc)
                :precondition (and (cal-at ?cal ?or ?oc)
                                   (next2-column ?oc ?nc)
                                   (next-row ?or ?nr)

                              )
                :effect (and (not (cal-at ?cal ?or ?oc))
                             (cal-at ?cal ?nr ?nc)
                        )
        )

        (:action move1
                :parameters (?cal ?or ?oc ?nr ?nc)
                :precondition (and (cal-at ?cal ?or ?oc)
                                   (next2-row ?or ?nr)
                                   (next-column ?oc ?nc)

                              )
                :effect (and (not (cal-at ?cal ?or ?oc))
                             (cal-at ?cal ?nr ?nc)
                        )
        )
```

)

Intelligent Systems Group