

Department of Computer Science
Technical University of Cluj-Napoca

Artificial Intelligence

Laboratory activity

Name:Cozma Victoria
Tornea Adriana
Group:30235
Email:ada.tornea@gmail.com
victorya.cozma@gmail.com

Teaching Assistant: Szomiur Roxana
szomiur.roxana@utcluj.ro

Contents

1	A1: Introducere	3
2	A2: Rezolvare matematică	4
3	A3: Implementare	12
4	A4: Testare	19
5	A5: Concluzii	25
6	Anexa	27

Chapter 1

A1: Introducere

First order logic - cunoscută și sub numele de logică de predicat, logică cuantificativă și calcul de predicat de ordinul întâi - este o colecție de sisteme formale utilizate în matematică, filozofie, lingvistică și informatică.

First order logic folosește variabile cuantificate peste obiecte non-logice și permite utilizarea unor propoziții care conțin variabile, astfel încât, decât să scriem propoziții precum "**Socrate este un om**", putem avea expresii sub forma "**există x astfel încât x este Socrate și x este un om**", unde "există" este un cuantificator, în timp ce x este o variabilă. Aceasta o distinge de logica propozițională, care nu folosește cuantificatori sau relații.

Am realizat mai multe jocuri logice de tip puzzle pentru a ilustra cum funcționează First Order Logic cu ajutorul lui **prover9** și **mace4**. Dintre aceste puzzle-uri amintim: Sudoku, Povestea Malice and Alice, Knights and Knaves, Propoziții pe care trebuie să le dezlegăm referitoare la unele animale și hrana lor.

De asemenea, am adaptat și am ilustrat cum se poate castiga jocul Wumpus, prin introducerea în mace4 a regulilor jocului și a indiciilor pe care le primește agentul în timpul jocului.

Chapter 2

A2: Rezolvare matematică

1. Malice and Alice

Alice, Alice's husband, their son, their daughter, and Alice's brother were involved in a murder. One of the five killed one of the other four. The following facts refer to the five people mentioned:

1. A man and a woman were together in a bar at the time of the murder.
2. The victim and the killer were together on a beach at the time of the murder.
3. One of Alice's two children was alone at the time of the murder.
4. Alice and her husband were not together at the time of the murder.
5. The victim's twin was not the killer.
6. The killer was younger than the victim.

Which one of the five was the victim?

Solutie

Din 1,2 si 3 aflam urmatoarele informatii:

Bar: man and woman

Beach: killer and victim

Alone: child

Din 4 avem 2 posibilități:

a)- **sotul lui Alice in bar**

- **Alice pe plaja**

b)- **Alice in bar**

-**sotul lui Alice pe plaja**

A) Daca sotul lui Alice era in bar, femeia era fiica lui, copilul- fiul lui era atunci singur, iar Alice era impreuna cu fratele ei pe plaja => fie Alice e criminalul si fratele ei victima sau criminalul a fost fratele lui Alice, iar victima Alice. Insa propozitia 5 ne spune ca victima a avut un geaman, iar acest geaman nu a fost criminalul, deci este nevinovat.

Din moment ce Alice si fratele ei sunt frati, deci posibili gemeni => situatia este imposibila. Din toate aceste informatii aflam atunci ca sotul lui Alice nu a fost in bar.

B) Deci, daca sotul lui Alice nu era in bar, trecem la urmatoarea varianta care ne spune ca Alice este in bar.

Daca Alice era in bar atunci ea poate fi impreuna cu fratele ei sau cu fiul ei.

Daca Alice era cu fratele ei, atunci sotul ei este pe plaja cu unul din cei doi copii. Din propozitia 5 aflam ca victima are un geaman care nu a fost criminalul, deci victima nu a fost sotul lui Alice, deoarece nici unul dintre aceste persoane cu care se afla pe plaja nu poate sa fie geamanul

lui, sunt copii lui. De aici poate rezulta situatia ca ucigasul a fost sotul lui Alice, iar victima poate fi unul dintre copii, insa situatia aceasta nu e buna, deoarece ultima propozitie,⁶ , ne aminteste ca, criminalul este mai tanar ca si victima, iar tatal nu poate sa fie mai tanar ca si copilul lui => Alice nu a fost in bar cu fratele ei.

Ultima supozitie este ca Alice a fost in bar cu fiul ei. Ceea ce inseamna ca fida lor este singura, iar pe plaja au ramas sotul si fratele lui Alice. Asa cum am spus si anterior, sotul lui Alice nu poate fi victima, asta inseamna ca victima poate fi fratele lui Alice, la 5 ne spune de geaman, iar geamanul poate sa fie doar fratele ei.

Aflam, deci, ca **fratele lui Alice a fost victima, iar sotul lui Alice a fost criminalul.**

2. Knights and Knaves

Avem 3 persoane A, B si C, unul dintre aceste persoane este cavaler, altul este sclav , iar ultimul este o persoana normala(nu neaparat in aceasta ordine)Cavalerul spune mereu adevarul, sclavii mint mereu, iar persoanele normale care uneori mint, iar uneori spun adevarul. Ei fac urmatoarele afirmatii:

A:"I am normal"

B:"That is true"

C:"I am not normal"

Cine este A, B si C?

Solutie

In primul rand ar trebui sa reprezentam toate posibilitatile acestui puzzle, care au loc pe insula:

-luam posibilitatea ca toata lumea este cavaler sau sclav sau un om normal, dar nu pot fi mai mult de o persoana din aceste 3 tipuri.

In al doilea rand trebuie sa tinem cont de ce spune fiecare persoana; introducem 3 constante A, B si C care le reprezentam a fi 3 persoane distincte. Am stabilit notatiile pentru fiecare si acum reprezentam ceea ce au spus cele 3 persoane, si incepem sa le luam pe rand:

1. **"I am normal"**

-asta o putem reprezenta spunand ca A, pentru ca el este cel care a spus asta, este normal

2. **"That is true"**

- interpretarea la aceasta propozitie este ca persoana A a spus adevarul si aflam ca ea este normala. Daca B, persoana care a spus aceasta fraza ne spune ca A a spus adevarul deducem din aceste doua propozitii ca cele doua persoane sunt identice, deci e considerata o singura persoana.

3. **"I am not normal"**

-persoana C spune acest lucru, deci ar trebui sa negam faptul ca persoana C este normala.

Ar trebui sa stabilim unele relatii intre cele trei persoane, si anume, daca de exemplu sunt o persoana A inseamna ca nu pot sa fie si o persoana de tip B sau C. Deci practic A diferita de B, A diferita de C si B diferita de C. Daca presupunem ca A este Knave atunci persoanele nu pot sa fie Knave si Normal, si asta se intampla pentru fiecare persoana in parte, acoperind toate posibilitatile. Putem sa restrangem toate cele trei propozitii intr-una singura:

-"Nici un locuitor al insulei nu poate fi atat un cavaler, cat si un sclav, nici un

cavaler si un om normal sau nici un sclav si un om normal."

Consideram ca avem o persoana P care locuieste pe insula si care face afirmatia AF. Si pe baza acestei consideratii incercam sa transformam toate cele trei afirmatii. Dupa ce am scris aceste afirmatii in cod incercam sa le simplificam cu ajutorul formulelor lui De Morgan si sa ajungem la demonstrarea teoremei.

Formulele De Morgan aplicate:

$$\neg(P \vee Q) \iff (\neg P) \wedge (\neg Q), \neg(P \wedge Q) \iff (\neg P) \vee (\neg Q), \text{ and}$$

$$\neg(P \wedge Q) \iff (\neg P) \vee (\neg Q) \neg(P \vee Q) \iff (\neg P) \wedge (\neg Q)$$

3.Sudoku

Solutie

*Pentru inceput am luat o regula si anume "Cel putin unul din fiecare".

Regula1: Cel putin unul pentru fiecare rand.

Regula2: Cel putin unul pentru fiecare coloana.

Putem imparti tabla de joc in anumite intervale, fiind o tabla de 9x9, putem avea 3 intervale: [0,1,2],[3,4,5],[6,7,8]

Am definit intervalele, 3 la numar, acum ne gandim sa impartim tabla de joc in regiuni.

Aceste regiuni sunt determinate de intervale: spre exemplu avem intervalul (x1,x2) si intervalul (y1,y2) care implica regiunile R(x1,y1) si R(x2,y2) care se afla in aceeasi regiune pe tabla de joc; din asta rezulta ca cele doua unificate implica si echivalent conform careia x1=x2 si y1=y2.

Daca regula ar fi "**Cel mult una din fiecare**", adica cel mult una pe fiecare coloana si cel mult una pe fiecare linie include si regula de mai sus care ne specifica cel putin.

4.Schubert's Steamroller-Animale

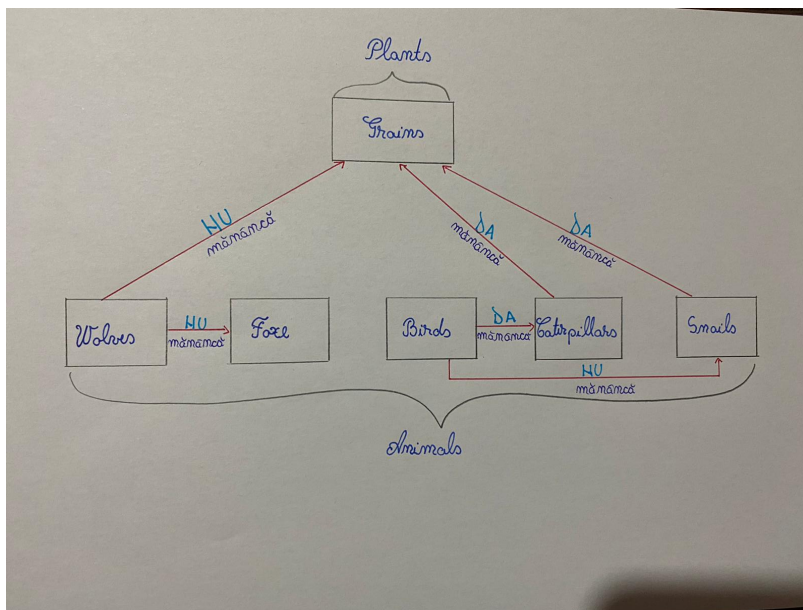
Wolves, foxes, birds, caterpillars, and snails are animals, and there are some of each of them. Also there are some grains, and grains are plants.

Every animal either likes to eat all plants or all animals much smaller than itself that like to eat some plants. Caterpillars and snails are much smaller than birds, which are much smaller than foxes, which in turn are much smaller than wolves.

Wolves do not like to eat foxes or grains, while birds like to eat caterpillars but not snails. Caterpillars and snails like to eat some plants. Which animal likes to eat a grain eating animal, and which?

Solutie

Am incercat sa fac o rezolvare matematica utilizand o schema logica si informatiile pe care le-am pimit din enuntul problemei.



5.n-Queens

Problema se refera la aranjarea a n regine pe o tabla de sah $n \times n$ astfel incat niciuna sa nu se ameninte pe alta.

Introducem astfel variabile: x, y, z .

Domeniul pe care ne desfasuram este $[0, 1, \dots, n-1]$.

X_i (unde $i=0, 1, \dots, n-1$) $X_i=a$ reprezinta ca exista o regina pe linia i si coloana a .

Trebuie sa tinem cont de urmatoarele reguli:

-i diferit de j rezulta ca X_i diferit de X_j , ceea ce inseamna ca nu exista 2 regine pe aceeaasi coloana

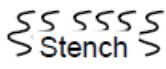




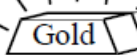

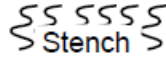





-i diferit de j rezulta ca $X_i - X_j$ diferit de $i-j$, ceea ce inseamna ca nu exista 2 regine pe diagonala stanga

-i diferit de j rezulta $X_i - X_j$ diferit de $j-i$, ceea ce inseamna ca nu exista 2 regine pe diagonala dreapta

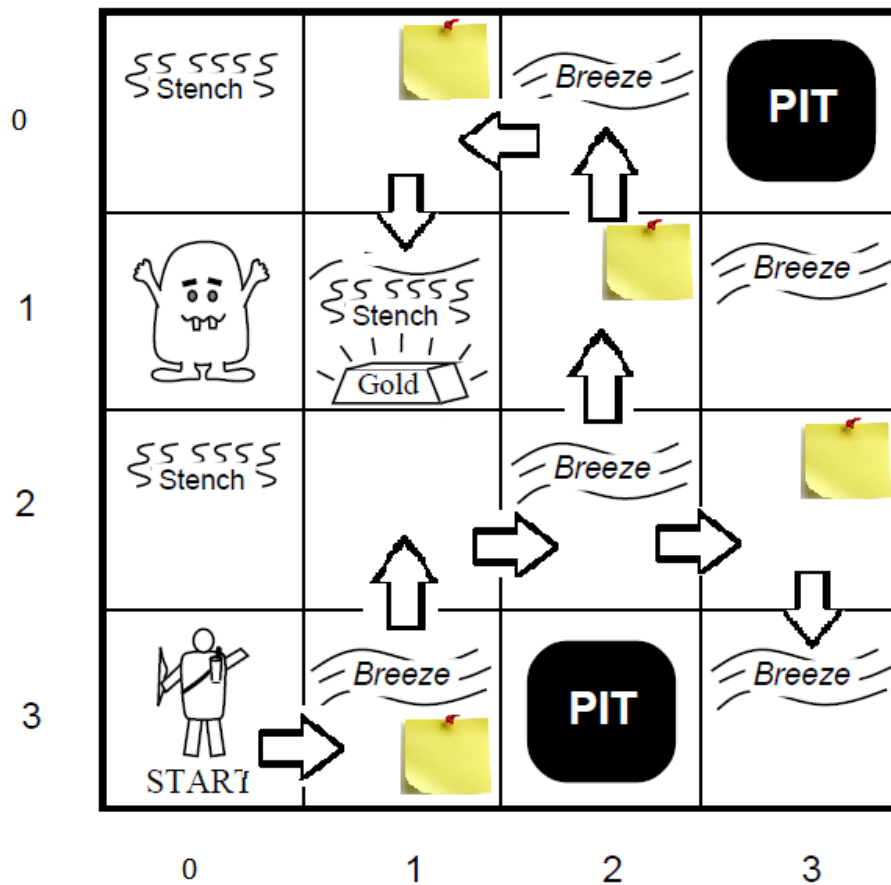
Pentru a utiliza operatori aritmetici am introdus set(arithmetic), iar unde am avut nevoie de scadere am adunat acea variabila la o valoare negata.

5. Let's find Wumpus

Actiunea din Wumpus are loc intr-o peștera. Agentul trebuie sa gaseasca comoara, fara a da peste Wumpus sau peste vreo groapa. Regulile sunt simple: daca agentul intra in casuta cu Wumpus, agentul moare si jocul se termina. Daca agentul cade in groapa, ramane blocat acolo. Agentul castiga daca gaseste comoara (care nu poate fi in casa Wumpus-ului si nici intr-o groapa). Wumpus-ul este unic si nu exista gropi in casa acestuia. O celula este considerata safe daca aceasta nu contine Wumpus si nici groapa. Pentru a gasi solutia, agentul poate folosi urmatorii senzori: briza din celulele adiacente gropilor si mirosul din celulele adiacente si cea a Wumpus-ului.

0	 Stench		 Breeze	
1		 Stench  Gold		 Breeze
2	 Stench		 Breeze	
3	 START	 Breeze		 Breeze
	0	1	2	3

Pentru a modela solutia, vom folosi o matrice 4x4. Celula (x,y) reprezinta celula de pe randul x si coloana y. Vom numerota randurile si coloanele incepand cu 0, din coltul stanga sus (ca in imaginea de mai sus). Pentru a simula un joc adevarat, vom transcrie toate regulile jocului din limbaj natural in FOL. Agentul va primi informatii aditionale la fiecare pas. In continuare, vom face o simulare a jocului, utilizand senzori si indicii.



Agentul porneste din celula (3,0) (stanga jos). Stim ca aceasta celula e safe
 $\text{safe}(3,0)$.

Simte miros in (2,0) si briza in (3,1).
 $\text{smell}(2,0)$.
 $\text{breeze}(3,1)$.

Note 1: Tot randul 2 este safe
 $\text{all } y \text{ (safe}(2,y))$.

Merge pe randul 2 si descopera briza in celula (2,2).
 $\text{breeze}(2,2)$.

Note 2: toate celulele de pe diagonala sunt safe si nu exista 2 pit-uri pe acelasi
 rand sau pe aceeasi coloana
 $\text{all } x \text{ (safe}(x,x))$.
 $\text{all } x \text{ all } y \text{ (pit}(x,y) \rightarrow \neg(\text{exists } z \text{ (pit}(z,y) \ z \neq x)))$.
 $\text{all } x \text{ all } y \text{ (pit}(x,y) \rightarrow \neg(\text{exists } z \text{ (pit}(x,z) \ z \neq y)))$.

Gaseste briza din (3,3) si descopera pit-ul din (3,2). Merge in sus pe coloana 2 si descopera briza in (0,2)
breeze(0,2).

Note3: Celula (0,1) e safe. Comoara se afla pe diagonala, dar nu poate fi in aceeaasi celula cu briza si nu e in celula (0,0).
safe(0,1).
all x all y (breeze(x,y) \rightarrow -gold(x,y)).
gold(x,y) \rightarrow x = y.
-gold(0,0).

Pentru a implementa regulile jocului, vom folosi urmatoarele predicate:

- smell(x,y) - exista miros in celula cu coordonatele (x,y)
- breeze(x,y) - exista briza in celula cu coordonatele (x,y)
- pit(x,y) - celula (x,y) contine groapa
- wumpus(x,y) - wumpus-ul e in celula (x,y)
- gold(x,y) - celula (x,y) contine comoara
- safe(x,y) - celula care este safe

Urmeaza sa transcriem regulile jocului in FOL:

Exista wumpus si gold
exists x exists y (wumpus(x,y)).
exists x exists y (gold(x,y)).

Celula e safe daca nu exista wumpus si groapa in ea.
all x all y (safe(x,y) \leftrightarrow -wumpus(x,y) -pit(x,y)).

Daca wumpus-ul e in celula (x,y) atunci persista miros in celulele adiacente.
all x all y (wumpus(x,y) smell(x, y)).
all x all y (wumpus(x,y) \rightarrow smell(inc(x), y)).
all x all y (wumpus(x,y) \rightarrow smell(dec(x), y)).
all x all y (wumpus(x,y) \rightarrow smell(x, inc(y))).
all x all y (wumpus(x,y) \rightarrow smell(x, dec(y))).

Daca suntem intr-o celula cu smell, atunci avem un wumpus in una din celulele adiacent.
all x all y (smell(x,y) \rightarrow wumpus(x, y) | wumpus(inc(x), y) | wumpus(dec(x), y) | wumpus(x, inc(y)) | wumpus(x, dec(y))).

Daca avem groapa in celula (x,y) atunci avem briza in celulele adiacente.
all x all y (pit(x,y) \rightarrow breeze(inc(x), y)).

$\text{all } x \text{ all } y (\text{pit}(x,y) \rightarrow \text{breeze}(\text{dec}(x), y)).$
 $\text{all } x \text{ all } y (\text{pit}(x,y) \rightarrow \text{breeze}(x, \text{inc}(y))).$
 $\text{all } x \text{ all } y (\text{pit}(x,y) \rightarrow \text{breeze}(x, \text{dec}(y))).$

Daca suntem intr-o celula in care simtim briza, atunci intr-una din celulele adiacente e groapa.

$\text{all } x \text{ all } y (\text{breeze}(x,y) \rightarrow \text{pit}(\text{inc}(x), y) \mid \text{pit}(\text{dec}(x), y) \mid \text{pit}(x, \text{inc}(y)) \mid \text{pit}(x, \text{dec}(y))).$

Wumpus-ul nu poate exista in aceasi celula cu groapa si cu comoara.

$\text{all } x \text{ all } y (\text{wumpus}(x,y) \rightarrow \neg \text{pit}(x,y)).$
 $\text{all } x \text{ all } y (\text{wumpus}(x,y) \rightarrow \neg \text{gold}(x,y)).$

Comoara nu poate fi celula cu groapa si nici in celula cu wumpus-ul.

$\text{all } x \text{ all } y (\text{gold}(x,y) \rightarrow \neg \text{pit}(x,y)).$
 $\text{all } x \text{ all } y (\text{pit}(x,y) \rightarrow \neg \text{gold}(x,y)).$

Wumpus ul e unic

$\text{all } x \text{ all } y (\text{wumpus}(x,y) \rightarrow \neg (\text{exists } z \text{ exists } w (\text{wumpus}(z,w) \wedge z \neq x \wedge w \neq y))).$
 $\text{all } x \text{ all } y (\text{wumpus}(x,y) \rightarrow \neg (\text{exists } z (\text{wumpus}(z,y) \wedge z \neq x))).$
 $\text{all } x \text{ all } y (\text{wumpus}(x,y) \rightarrow \neg (\text{exists } z (\text{wumpus}(x,z) \wedge z \neq y))).$

Gold-ul e unic

$\text{all } x \text{ all } y (\text{gold}(x,y) \rightarrow \neg (\text{exists } z \text{ exists } w (\text{gold}(z,w) \wedge z \neq x \wedge w \neq y))).$
 $\text{all } x \text{ all } y (\text{gold}(x,y) \rightarrow \neg (\text{exists } z (\text{gold}(z,y) \wedge z \neq x))).$
 $\text{all } x \text{ all } y (\text{gold}(x,y) \rightarrow \neg (\text{exists } z (\text{gold}(x,z) \wedge z \neq y))).$

Chapter 3

A3: Implementare

1. Knights and Knaves-Mace4

Reprezentam pe rand toate posibilitatile , toate situatiile care au loc pe insula: toata lumea este cavaler, sclav sau un om normal. Pentru asta realizam urmatoarele propozitii:

ori care ar fi x ($\text{Knight}(x) \mid \text{Knave}(x) \mid \text{Normal}(x)$)
ori care ar fi x ($\text{Knight}(x) \rightarrow (\text{Knave}(x) \mid \text{Normal}(x))$)
ori care ar fi x ($\text{Knave}(x) \rightarrow (\text{Knight}(x) \mid \text{Normal}(x))$)
ori care ar fi x ($\text{Normal}(x) \rightarrow (\text{Knight}(x) \mid \text{Knave}(x))$)

Mai trebuie sa reprezentam si faptul ca avem 3 tipuri de locuitori pe insula. Aici vom introduce constantele A,B si C.

Trebuie precizat insa ca aceste trei constante se refera la 3 tipuri de locuitori diferiti. Am facut acest lucru utilizand egalitatea si negand acolo unde nu este cazul. Mai precis persoana A este diferita de persoana B, persoana A este diferita de persoana C si in final, persoana B este diferita de persoana C.

-Egal(A,B) SI -Egal(A,C) SI -Egal(B,C).

In Mace4 reprezentam acestu lucru prin $x \neq y$.

Un alt aspect este ca trebuie sa reprezentam ca exista printre acestia 3 locuitori, cate unul din fiecare tip (cavaler, sclav si om normal).

Exista x Knight(x)
Exista x Knave(x)
Exista x Normal(x)

Ultima reprezentare este cea a propozitiilor spuse de catre cele trei persoane A,B si C.

1. "I am normal"

- o reprezentam prin :

Normal(A)

2. "That is true"

-devine

Normal(A)

,deoarece daca B ne spune ca A a spus adevarul deducem ca A si B sunt doua persoane identice.

3. "I am not normal"

-negam faptul ca persoana care a spus asta, si anume C, este normala

-Normal(C)

Putem sa facem o generalizare a acestor 3 persoane si a ceea ce spun ele prin cele trei propozitii, si anume:

"Pentru orice persoana denumita P(person) care traieste pe insula face afirmatia Af". P reprezinta astfel o constanta, iar Af reprezinta o formula(P spune Af).

(Knight(P) SI Af) SAU (Knave(P) SI Af) SAU Normal(P)

Aceasta formula generala o vom aplica pentru fiecare propozitie spusa de catre cei trei participanti, A, B si C.

(Knight(A) SI Normal(A)) SAU (Knave(A) SI Normal(A)) SAU Normal(A)
(Knight(B) SI Normal(A)) SAU (Knave(B) SI Normal(A)) SAU Normal(B)
(Knight(C) SI Normal(C)) SAU (Knave(C) SI Normal(C)) SAU Normal(C)

Aceste trei propozitiile simplificam dupa cum urmeaza:

Knave(A) SAU Normal(A)
(Knight(B) SI Normal(A)) SAU (Knave(B) SI Normal(A)) SAU Normal(B)
Knight(C) SAU Normal(C)

Simplificam aceste propozitii rescrise in FOL, deoarece:

-Propozitia **Knight(A) SI Normal(A)** in prima propozitie rescrisa va fi mereu falsa, si poate fi stearsa(fiind ca o disjunctie).

-Propozitia **Knave(A) SI Normal(A)** in prima propozitie se reduce la Knave(A), deoarece daca a fi sclav implica automat sa nu fie normal.

-Propozitia **Knight(C) SI Normal(C)** in cea de-a treia propozitie rescrisa se reduce la Knight(C), deoarece sa fii Knight implica automat sa nu fi normal.

-Propozitia **Knave(C) SI Normal(C)** in formula 3 este falsa, si putem sa o stergem(la fel ca si in prima formula).

Dupa toate aceste informatii pe care le-am colectat, codul arata asa:

Knight(x) | Knaght(x) | Normal(x).

Knight(x) → (-Knave(x) -Normal(x)).

Knave(x) → (-Knight(x) -Normal(x)).

Normal(x) → (-Knight(x) -Knave(x)).

$(\text{exists } x \text{ Knight}(x)).$
 $(\text{exists } x \text{ Knaght}(x)).$
 $(\text{exists } x \text{ Normal}(x)).$

$A \neq B \quad A \neq C \quad B \neq C.$

$\text{Knaght}(A) \mid \text{Normal}(A).$
 $(\text{Knight}(B) \mid \text{Normal}(A)) \mid (\text{Knaght}(B) \mid \neg \text{Normal}(A)) \mid \text{Normal}(B).$
 $\text{Knight}(C) \mid \text{Normal}(C).$

1. Knights and Knaves-Prover9

Pentru demonstratia in Prover9 am ales sa incepem de la urmatoarea formula:

oricare ar fi x ($x=A$ SAU $x=B$ SAU $x=C$)

si reprezinta faptul ca oricum am alege acest x el va reprezenta un om de pe aceasta insula, mai precis variabilele A, B sau C .

In mace4 am reprezentat faptul ca nu avem intre cele trei persoane, mai mult de un tip, astfel:
 $\text{Knight}(x) \rightarrow (\neg \text{Knave}(x) \mid \neg \text{Normal}(x)).$

In prover9 am simplificat propozitiile si le-am definit astfel:

$\neg(\text{Knight}(x) \text{ SI } \text{Knave}(x)).$

$\neg(\text{Knight}(x) \text{ SI } \text{Normal}(x)).$

$\neg(\text{Knave}(x) \text{ SI } \text{Normal}(x)).$

Am luat toate cele trei posibilitati, daca este Knight inseamna ca nu e normal, daca este knight inseamna ca nu este knave si daca este knave inseamna ca nu este normal. Am aplicat formulele lui De Morgan si am obtinut urmatoarele transformari:

$\neg \text{Knight}(x) \text{ SAU } \neg \text{Knave}(x).$

$\neg \text{Knight}(x) \text{ SAU } \neg \text{Normal}(x).$

$\neg \text{Knave}(x) \text{ SAU } \neg \text{Normal}(x).$

Codul arata asa:

$\text{Knight}(x) \mid \text{Knave}(x) \mid \text{Normal}(x).$

$\neg \text{Knight}(x) \mid \neg \text{Knave}(x).$

$\neg \text{Knight}(x) \mid \neg \text{Normal}(x).$

$\neg \text{Knave}(x) \mid \neg \text{Normal}(x).$

$x=A \mid x=B \mid x=C.$

$A \neq B.$

$A \neq C.$

$B \neq C.$

$(\text{exists } x \text{ Knight}(x)).$

$(\text{exists } x \text{ Knave}(x)).$

$(\text{exists } x \text{ Normal}(x)).$

$\text{Knave}(A) \mid \text{Normal}(A).$

$(\text{Knight}(B) \mid \text{Normal}(A)) \mid (\text{Knave}(B) \mid \neg \text{Normal}(A)) \mid \text{Normal}(B).$

$\text{Knight}(C) \mid \text{Normal}(C).$



2.Sudoku

Obiectivul nostru pentru acest joc este de a umple o tabla de 9x9 cu cifre, astfel incat fiecare coloana, fiecare rand si fiecare dintre cele 9 sub-grile de 3x3 contin toate cifrele de la 1 la 9. Acelasi numar nu poate sa apara de doua ori in acelasi rand sau in aceeaasi coloana, dintre cele 9 subgrile de 3x3 ale placii de 9x9.

Pentru a rezolva puzzle-ulam introdus unele reguli.

Prima regula este:"Cel mult una pe fiecare dintre randuri".

$S(x,y1)=S(x,y2)$ in cele doua celule, asta inseamna ca ne aflam in aceeaasi linie, dar in coloane diferite.

$S(x1,y)=S(x2,y)$ cele doua celule, ne aflam pe aceeaasi coloana, iar liniile difera.

Introducem constanta z care ne spune daca acel numar se afla pe acea linie sau acea coloana.

$\forall x \forall z \exists y S(x,y) = z.$ pentru linie

$\forall y \forall z \exists x S(x,y) = z.$ pentru coloana

Tabla de joc este impartita in 9 sub-grile de 3x3 care le-am reprezentat ca niste intervale si am stabilit o relatie de echivalenta intre aceste intervale. Ceea ce inseamna ca $\text{interval}(x,y)=\text{interval}(y,x).$

Urmatoarea regula este cel mult unul in fiecare regiune, regiunea fiind intervalele definite. Ceea ce inseamna ca daca suntem in regiunea $(x1,y1)$ si $(x2,y2)$, atunci ne aflam si in acelasi interval deci rezulta ca $x1=x2$ si $y1=y2.$

Sudoku

Initial State									Solution								
5	3			7					5	3	4	6	7	8	9	1	2
6			1	9	5				6	7	2	1	9	5	3	4	8
	9	8					6		1	9	8	3	4	2	5	6	7
8				6				3	8	5	9	7	6	1	4	2	3
4			8		3			1	4	2	6	8	5	3	7	9	1
7				2				6	7	1	3	9	2	4	8	5	6
	6					2	8		9	6	1	5	3	7	2	8	4
			4	1	9			5	2	8	7	4	1	9	6	3	5
				8			7	9	3	4	5	2	8	6	1	7	9

3.n-Queens

2 Regine pe tabla de joc
set(arithmetic).

2 regine pe tabla de joc Relatia $Queen(a)=n$ reprezinta faptul ca este o regina pe randul i coloana n,prin aceasta remarca curpindem si faptul ca nu exista nici o regina pe tabla de joc.

formulas(assumptions).

Pentru a reprezenta scaderea m-am ajutat de negarea lui x si adunare

Aici reprezentam ca nu exista doua regine pe aceeasi coloana

$$x \neq y \rightarrow Q(x) \neq Q(y).$$

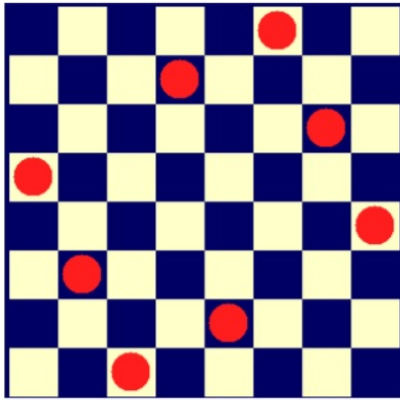
Nu exista doua regine pe diagonala stanga

$$x \neq y \rightarrow y + -x \neq Q(y) + -Q(x).$$

Nu exista doua regine pe diagonala dreapta

$$x \neq y \rightarrow y + -x \neq Q(x) + -Q(y).$$

end_of_list.



4. Animals

La problema animalelor si ceea ce mananca ele, am introdus 9 attribute, 7 dintre ele reprezinta ca sunt animale si ce tip de animale sunt, iar ultimele doua am reprezentat faptul ca x mananca pe y ($Eats(x,y)$: x eats y) si ca x este mai mic decat y ($Small(x,y)$: x is smaller y).

In primele linii de cod am transpus ideea ca fiecare dintre aceste nume sunt animale prin implicatie(v), si asta am facut si cu cerealele singurul aliment de care stim ca este planta si ca este mancata de unele animale.

Una dintre propozitiile care ne-au ajutat sa descifram puzzle-ul este aceea ca animalele fie mananca toate plantele, si anume cerealele, fie mananca toate animalele care sunt mai mici decat el si care mananca plante.

Astfel am spus ca orice x, x fiind animal implica faptul ca orice y care este planta implica ca x mananca planta y SAU orice z, z fiind animal si fiind mai mic, pentru a putea respecta regula, exista un m, care este planta-cereala si animalul z care mananca un animal mai mic decat el mananca si planta m.

Propozitiile urmatoare ne mai ofera informatii despre ceea ce mananca si ce nu mananca unele animale, astfel am reprezentat omizile si pasarile stiind ca sunt mai mici, melcii mai mici ca si pasarile, pasarile care sunt mai mici decat vulpile si vulpile care sunt mai mici decat lupii. Aflam ca pasarile mananca omizi, ca lupii nu mananca vulpi, lupii nu mananca cereale si pasarile nu mananca melci. Pentru goal am reprezentat faptul ca exista un animal care mananca un alt animal ce mananca plante si l-am descoperit ca fiind pasarea.

$Animal(x)$: x is a animal

$Plante(x)$: x is a plant

$Wolf(x)$: x is a wolf

$Fox(x)$: x is a fox

$Bird(x)$: x is a bird

$Caterpillar(x)$: x is a caterpillar

$Snail(x)$: x is a snail

$Eats(x,y)$: x eats y

$Small(x,y)$: x is smaller y

formulas(assumptions).

all x (Wolf(x) \rightarrow Animal(x)).

all x (Fox(x) \rightarrow Animal(x)).

all x (Bird(x) \rightarrow Animal(x)).

all x (Caterpillar(x) \rightarrow Animal(x)).

all x (Snail(x) \rightarrow Animal(x)).

all x (Grain(x) \rightarrow Plante(x)).

exists x Wolf(x).

exists x Fox(x).

exists x Bird(x).

exists x Caterpillar(x).

exists x Snail(x).

exists x Grain(x).

Toate animalele fie mananca toate plantele, fie mananca toate animalele mai mici ca el
care mananca unele plante

all x (Animal(x) \rightarrow (all y (Plante(y) \rightarrow Eats(x,y))) | (all z (Animal(z)
Small(z,x) (exists m (Plante(m) Eats(z,m))) \rightarrow Eats(x,z))))).

all x all y (Caterpillar(x) Bird(y) \rightarrow Small(x,y)).

all x all y (Snail(x) Bird(y) \rightarrow Small(x,y)).

all x all y (Bird(x) Fox(y) \rightarrow Small(x,y)).

all x all y (Fox(x) Wolf(y) \rightarrow Small(x,y)).

all x all y (Bird(x) Caterpillar(y) \rightarrow Eats(x,y)).

all x (Caterpillar(x) \rightarrow (exists y (Plante(y) Eats(x,y)))).

all x (Snail(x) \rightarrow (exists y (Plante(y) Eats(x,y)))).

all x all y (Wolf(x) Fox(y) \rightarrow -Eats(x,y)).

all x all y (Wolf(x) Grain(y) \rightarrow -Eats(x,y)).

all x all y (Bird(x) Snail(y) \rightarrow -Eats(x,y)).

end_{olist}.

formulas(goals).

exists x exists y (Animal(x) Animal(y) Eats(x,y) (all z (Grain(z) \rightarrow Eats(y,z)))).
end_{olist}.

Chapter 4

A4: Testare

1.Knights and Knaves-Mace4

Pentru problema Knights and Knaves am rulat cu comanda: Mace4 -f K.in.

Ceea ce am btinut ca si rezultat este acesta:

```
interpretation( 3, [number=1, seconds=0], [  
function(A, [ 0 ]),
```

```
function(B, [ 1 ]),
```

```
function(C, [ 2 ]),
```

```
function(goals, [ 0 ]),
```

```
function(c1, [ 2 ]),
```

```
function(c2, [ 0 ]),
```

```
function(c3, [ 1 ]),
```

```
relation(end_of_files, [1]),
```

```
relation(Knave(), [1, 0, 0]),
```

```
relation(Knight(), [0, 0, 1]),
```

```
relation(Normal(), [0, 1, 0]),
```

```
relation(formulas(), [1, 0, 0])).
```

Care poate fi interpretat astfel:

- A) Prima linie indica ca exista un singur model, adica o solutie pentru acest puzzle si acesta contine cei trei locuitori.
- B) Urmatoarele 3 linii reprezinta ca primul locuitor al insulei denumit A ii este asociat numarul 0, locuitorului B ii este asociat numarul 1, iar ultimului locuitor C ii este asociat valoarea 2.
- C) Cele trei linii din mijloc in care regasim c1,c2 si c3 reprezinta faptul ca mace4 si-a creat 3 nume pentru locuitorii insulei si sunt reprezentati prin 2, 0 si 1. Aceste 3 valori vor fi utilizate pentru a se face referire la persoanele din matricea de elemente.
- D)Ultimele 3 linii indica ca un singur locuitor al insulei, care este A, cu indicele 0 este

sclav(Knave), locuitorul B cu indicele 1 este persoana normala, iar locuitorul cu indicele 2 este cavalerul(Knight).

Astfel mace4 rezolva puzzle-ul, deci demonstreaza teorema ca A este sclav, B este un om normal si C este cavaler.

Am obtinut urmatorul rezultat sub forma de tabel:

1.Knights and Knaves-Prover9

Comanda cu care am rulat programul este: prover9 -f KnightProver.in

Demonstratia teoremei este:

===== PROOF =====

```
5 Knave(C) Normal(B) Knight(A) label(non_clause).[assumption].
6 - Knight(x) | - Knave(x).[assumption].
8 - Knight(x) | - Normal(x).[assumption].
13 Knight(A).[clausify(5)].
25 Knave(A) | Normal(A).[assumption].
35 - Knave(A).[resolve(13, a, 6, a)].
36 - Normal(A).[resolve(13, a, 8, a)].
```

Luam linia 36 care se rezolva utilizand liniile 13 si 8.

13 Knight(A).

8 -Knight(x) | -Normal(x).

Aplicam Unit Deletion astfel:

Q este Knight(x)

-Q este -Knight(x)

R este -Normal(x)

x este la noi persoana A.

Acelasi rationament este si pentru linia 35.

2.Sudoku-Mace4

Pentru Sudoku am rulat astfel:

-pentru a genera prima solutie am rulat:mace4 -f Sudoku.in

Rezultatul primei rulari este prezentat in partea 5 a documentatiei.

Primul tabel reprezinta primul rezultat posibil generat de catre mace4, interval reprezinta intervalele luate de catre program pentru a genera solutia.

3.Problema animale-Prover9

Am rulat aceasta probleme cu prover9 :prover9 -f Animale.in

Rezultatul pe care l-am obtinut este ca teorema s-a demonstrat:

```

adriana@adriana-VirtualBox: ~/Downloads/LADR-2009-11A-M...
Forward_subsumed=17. Back_subsumed=12.
Sos_limit_deleted=0. Sos_displaced=0. Sos_removed=0.
New_demodulators=0 (0 lex), Back_demodulated=0. Back_unit_deleted=0.
Demod_attempts=0. Demod_rewrites=0.
Res_instance_prunes=0. Para_instance_prunes=0. Basic_paramod_prunes=
Nonunit fsub feature tests=9. Nonunit bsub feature tests=66.
Megabytes=0.18.
User_CPU=0.01, System_CPU=0.01, Wall_clock=0.

===== end of statistics =====
===== end of search =====

THEOREM PROVED

THEOREM PROVED

Exiting with 1 proof.

----- process 29379 exit (max_proofs) -----

Process 29379 exit (max_proofs) Tue Nov 24 13:21:01 2020
adriana@adriana-VirtualBox:~/Downloads/LADR-2009-11A-ME/LADR-2009-11
adriana@adriana-VirtualBox:~/Downloads/LADR-2009-11A-ME/LADR-2009-11

```

Ca pentru fiecare problema am definit attribute pentru fiecare animal si am incercat sa reprezentam cu acestea propozitiile pe care l stia m despre animale si cu ce tip de hrana se alimenteaza. Predicatele folosite sunt: Animal, Plante, Wolf, Fox, Bird, Caterpillar, Snail si am mai inclus faptul ca mananca-Eats si ca sunt mai mici-Smaller, pentru ca stiam ca animalul mananca doar animale mai mici ca si el.

Am ales trei propozitii pentru a arata ca se incadreaza in una dintre forme: Back Unit Deletion si Unit Deletion

40 -Bird(x) | -Snail(y) | -Eats(x,y). [clausify(23)].

35 Bird(c3). [clausify(9)].

53 -Snail(x) | -Eats(c3,x). [resolve(40,a,35,a)].

Ne incadram in Unit Deletion, in care am realizat urmatoarele aspecte:

Q=Bird(c3).

-Q=-Bird(x).

R=-Snail(y) | Eats(x,y).

,unde x reprezinta c3, iar y este x si rezulta ca teorema este demonstrata:-Snail(x) | -Eats(c3,x)

```

Ubuntu (Running) - Oracle VM VirtualBox
File Machine View Input Devices Help
adriana@adriana-VirtualBox: ~/Downloads/LADR-2009-11A-ME/LADR-2009-11A/bin

given #43 (T,wt=4): 115 Eats(c3,f1(c4)). [resolve(109,a,69,a)].
given #44 (T,wt=4): 124 Plante(f3(c2,c3)). [resolve(117,a,77,c),unit_del(a,65),unit_del(b,67)].
----- Proof 1 -----

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

1 (all x (Wolf(x) -> Animal(x))) # label(non_clause). [assumption].
2 (all x (Fox(x) -> Animal(x))) # label(non_clause). [assumption].
3 (all x (Bird(x) -> Animal(x))) # label(non_clause). [assumption].
4 (all x (Snail(x) -> Animal(x))) # label(non_clause). [assumption].
5 (all x (Grain(x) -> Plante(x))) # label(non_clause). [assumption].
6 (all x (Wolf(x) -> Eats(x,y))) # label(non_clause). [assumption].
7 (exists x (Wolf(x))) # label(non_clause). [assumption].
8 (exists x (Fox(x))) # label(non_clause). [assumption].
9 (exists x (Bird(x))) # label(non_clause). [assumption].
10 (exists x (Snail(x))) # label(non_clause). [assumption].
11 (exists x (Grain(x))) # label(non_clause). [assumption].
12 (exists x (Plante(x))) # label(non_clause). [assumption].
13 (all x (Animal(x) -> (all y (Plante(y) -> Eats(x,y))) | (all z (Animal(z) & Small(z,x) & (exists y (Plante(y) & Eats(z,y))) -> Eats(x,y)))) # label(non_clause). [assumption].
14 (all x all y (Snail(x) & Bird(y) -> Small(x,y))) # label(non_clause). [assumption].
15 (all x all y (Bird(x) & Fox(y) -> Small(x,y))) # label(non_clause). [assumption].
16 (all x all y (Fox(x) & Wolf(y) -> Small(x,y))) # label(non_clause). [assumption].
17 (all x all y (Wolf(x) & Grain(y) -> Small(x,y))) # label(non_clause). [assumption].
18 (all x (Snail(x) -> (exists y (Plante(y) & Eats(x,y))) # label(non_clause). [assumption].
19 (all x all y (Wolf(x) & Fox(y) -> Eats(x,y))) # label(non_clause). [assumption].
20 (all x all y (Wolf(x) & Grain(y) -> Eats(x,y))) # label(non_clause). [assumption].
21 (all x all y (Bird(x) & Snail(y) -> Eats(x,y))) # label(non_clause). [assumption].
22 (exists x exists y (Animal(x) & Animal(y) & Eats(x,y) & (all z (Grain(z) -> Eats(y,z)))) # label(non_clause) # label(goal). [goal].
23 -Wolf(c1). [clausify(7)].
24 -Wolf(x) | Animal(x). [clausify(1)].
25 -Fox(c1) | -Wolf(y) | Small(x,y). [clausify(17)].
26 -Fox(x) | -Fox(y) | Eats(x,y). [clausify(21)].
27 -Wolf(x) | -Grain(y) | Eats(x,y). [clausify(22)].
28 Fox(c2). [clausify(8)].
29 -Fox(x) | Animal(x). [clausify(2)].
30 -Bird(x) | -Fox(y) | Small(x,y). [clausify(16)].
31 -Bird(x) | -Snail(y) | Small(x,y). [clausify(15)].
32 -Fox(x) | Small(x,c1). [resolve(27,b,25,a)].
33 -Fox(x) | Eats(c1,x). [resolve(28,a,25,a)].
34 Bird(c3). [clausify(9)].
35 -Bird(x) | Animal(x). [clausify(3)].
36 -Snail(x) | -Bird(y) | Small(x,y). [clausify(15)].
37 -Bird(x) | -Snail(y) | Small(x,y). [clausify(15)].

```

```

adriana@adriana-VirtualBox: ~/Downloads/LADR-2009-11A-ME/LADR-2009-11A/bin
49 -Small(x) | Animal(x). [clausify(5)].
50 -Small(x) | Plante(f2(x)). [clausify(20)].
51 -Small(x) | Eats(x,f2(x)). [clausify(20)].
52 -Small(x) | Small(x,c3). [resolve(38,b,35,a)].
53 -Small(x) | Eats(c3,x). [resolve(40,a,35,a)].
54 Grain(c0). [clausify(12)].
55 -Grain(x) | Plante(x). [clausify(6)].
56 -Animal(x) | -Animal(y) | -Eats(x,y) | Grain(f3(x,y)). [deny(24)].
57 -Grain(x) | -Eats(c1,x). [resolve(29,a,25,a)].
58 Small(c2,c1). [resolve(33,a,30,a)].
59 -Animal(x) | -Plante(y) | Eats(x,y) | -Animal(z) | -Small(z,x) | -Plante(u) | -Eats(z,u) | Eats(x,z). [clausify(13)].
60 Small(c3,c2). [resolve(41,a,35,a)].
62 Small(c5,c3). [resolve(52,a,48,a)].
63 -Animal(x) | -Animal(y) | -Eats(x,y) | -Eats(y,f3(x,y)). [deny(24)].
64 Animal(c1). [resolve(25,a,26,a)].
65 Animal(c2). [resolve(39,a,31,a)].
66 -Eats(c1,c2). [resolve(34,a,30,a)].
67 Animal(c3). [resolve(35,a,36,a)].
72 Animal(c5). [resolve(48,a,49,a)].
73 Plante(f2(c5)). [resolve(50,a,40,a)].
74 Eats(c5,f2(c5)). [resolve(51,a,48,a)].
75 -Eats(c3,c5). [resolve(53,a,48,a)].
76 Plante(c6). [resolve(54,a,55,a)].
77 -Animal(x) | -Animal(y) | -Eats(x,y) | Plante(f3(x,y)). [resolve(56,d,55,a)].
78 -Eats(c1,c6). [resolve(57,a,54,a)].
80 -Animal(c1) | Plante(x) | Eats(c1,x) | -Animal(c2) | -Plante(y) | -Eats(c2,y) | Eats(c1,c2). [resolve(58,a,59,e)].
81 -Plante(x) | Eats(c1,x) | -Plante(y) | -Eats(c2,y). [copy(80),unit_del(a,64),unit_del(d,65),unit_del(g,66)].
82 -Animal(c2) | -Plante(x) | Eats(c2,x) | -Animal(c3) | -Plante(y) | -Eats(c3,y) | Eats(c2,c3). [resolve(60,a,59,e)].
83 -Plante(x) | Eats(c2,x) | -Plante(y) | -Eats(c3,y) | Eats(c2,c3). [copy(82),unit_del(a,65),unit_del(d,67)].
84 -Animal(c3) | -Plante(x) | Eats(c3,x) | -Animal(c5) | -Plante(y) | -Eats(c5,y) | Eats(c3,c5). [resolve(62,a,59,e)].
85 -Plante(x) | Eats(c3,x) | -Plante(y) | -Eats(c5,y). [copy(84),unit_del(a,67),unit_del(d,72),unit_del(g,73)].
90 -Plante(x) | Eats(c2,x) | -Eats(c3,x) | Eats(c2,c3). [factor(83,a,c)].
103 -Eats(c2,c6). [ur(81,a,76,a,b,78,a,c,76,a)].
109 -Plante(x) | Eats(c3,x). [resolve(85,d,74,a),unit_del(c,73)].
113 Eats(c3,c6). [resolve(109,a,76,a)].
117 Eats(c2,c3). [resolve(113,a,90,c),unit_del(a,76),unit_del(b,103)].
124 Plante(f3(c2,c3)). [resolve(117,a,77,c),unit_del(a,65),unit_del(b,67)].
125 -Eats(c3,f3(c2,c3)). [resolve(117,a,63,c),unit_del(a,65),unit_del(b,67)].
133 SF. [resolve(124,a,109,a),unit_del(a,125)].

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

```

```

adriana@adriana-VirtualBox: ~/Downloads/LADR-2009-11A-ME/LADR-2009-11A/bin
82 -Animal(c2) | -Plante(x) | Eats(c2,x) | -Animal(c3) | -Plante(y) | -Eats(c3,y) | Eats(c2,c3). [resolve(60,a,59,e)].
83 -Plante(x) | Eats(c2,x) | -Plante(y) | -Eats(c3,y) | Eats(c2,c3). [copy(82),unit_del(d,67)].
84 -Animal(c3) | -Plante(x) | Eats(c3,x) | -Animal(c5) | -Plante(y) | -Eats(c5,y) | Eats(c3,c5). [resolve(62,a,59,e)].
85 -Plante(x) | Eats(c3,x) | -Plante(y) | -Eats(c5,y). [copy(84),unit_del(a,67),unit_del(g,73)].
90 -Plante(x) | Eats(c2,x) | -Eats(c3,x) | Eats(c2,c3). [factor(83,a,c)].
103 -Eats(c2,c6). [ur(81,a,76,a,b,78,a,c,76,a)].
109 -Plante(x) | Eats(c3,x). [resolve(85,d,74,a),unit_del(c,73)].
113 Eats(c3,c6). [resolve(109,a,76,a)].
117 Eats(c2,c3). [resolve(113,a,90,c),unit_del(a,76),unit_del(b,103)].
124 Plante(f3(c2,c3)). [resolve(117,a,77,c),unit_del(a,65),unit_del(b,67)].
125 -Eats(c3,f3(c2,c3)). [resolve(117,a,63,c),unit_del(a,65),unit_del(b,67)].
133 SF. [resolve(124,a,109,a),unit_del(a,125)].

===== end of proof =====
===== STATISTICS =====
Given=44. Generated=85. Kept=67. proofs=1.
Usable=39. Sols=16. Denods=0. Limbo=0. Disabled=70. Hints=0.
Kept_by_rule=0. Deleted_by_rule=0.
Forward_subsumed=17. Back_subsumed=12.
Sols_limited=0. Sols_displaced=0. Sols_removed=0.
New_denodulators=0 (0 lex). Back_demodulated=0. Back_unit_deleted=0.
Demod_attempts=0. Demod_rewrites=0.
Res_instance_prunes=0. Para_instance_prunes=0. Basic_paramod_prunes=0.
Nonunit_fsub_feature_tests=9. Nonunit_bsub_feature_tests=66.
Megabytes=0.18.
User_CPU=0.01. System_CPU=0.00. Wall_clock=0.

===== end of statistics =====
===== end of search =====

THEOREM PROVED
THEOREM PROVED
Exiting with 1 proof.
----- process 31575 exit (max_proofs) -----
Process 31575 exit (max_proofs) Tue Nov 24 22:05:45 2020
adriana@adriana-VirtualBox: ~/Downloads/LADR-2009-11A-ME/LADR-2009-11A/bin$

```

n-Queens

Rezultatele obtinute sunt urmatoarele:

=== Mace4 starting on domain size 4. ===

===== MODEL =====

interpretation(4, [number=1, seconds=0], [

function(f1(), [1, 3, 0, 2]),

relation(Q(,), [

0, 1, 0, 0,

```
0,0,0,1,
1,0,0,0,
0,0,1,0])
)).
```

```
===== end of model =====
```

Pentru nici o regina pe tabla:

```
===== MODEL =====
```

```
interpretation( 4, [number=1, seconds=0], [
```

```
function(Q(), [1, 3, 0, 2])
]).
```

```
===== endofmodel =====
```

Testare Wumpus

Rulare:

```
mace4 -c -f wumpusWorld.in | interpformat | isofilter ignore_constants
```

```
=== Mace4 starting on domain size 4. ===
```

```
—— process 6948 exit (all_models) — — — — —
```

```
interpretation(4, [number = 1, seconds = 0], [
function(dec(), [1, 0, 1, 2]),
function(inc(), [1, 2, 3, 2]),
relation(breeze(), [
0,0,1,0,
0,0,0,1,
0,0,1,0,
0,1,0,1]),
relation(gold(), [
0,0,0,0,
0,1,0,0,
0,0,0,0,
0,0,0,0]),
relation(pit(), [
0,0,0,1,
0,0,0,0,
0,0,0,0,
0,0,1,0]),
relation(safe(), [
1,1,1,0,
0,1,1,1,
1,1,1,1,
1,1,0,1]),
relation(smell(), [
```

```
1, 0, 0, 0,  
1, 1, 0, 0,  
1, 0, 0, 0,  
0, 0, 0, 0]),  
relation(wumpus(, [  
0, 0, 0, 0,  
1, 0, 0, 0,  
0, 0, 0, 0,  
0, 0, 0, 0]))).
```


Chapter 5

A5: Concluzii

In concluzie First Order Logic este un bagaj de instrumente pentru validarea argumentelor. La baza logicii de ordinul intai sta o familie de limbi definite matematic, si anume de ordinul intai. Prin acestea putem garanta din forma lor gramaticala ca anumite argumente scrise in aceasta limba sunt valide.

Logica de ordinul inati contine expresii pe care le putem interpreta ca o serie de modalitati care sunt posibile bazandu-se pe gramatica expresiilor scrise. Expresii precum:

- $\forall x (P(x) \rightarrow Q(x))$.
- $(P(x) \rightarrow Q(x))$

Am inclus aici tabelele pentru Sudoku(tabelul 5.1), generate de mace4 si pentru problema Knight and Knave(tabelul 5.2):

=== Mace4 starting on domain size 9. ===

S:	0	1	2	3	4	5	6	7	8	f1:	0	1	2	3	4	5	6	7	8
0	5	3	4	6	7	8	0	1	2	0	6	7	8	1	2	0	3	4	5
1	6	7	2	1	0	5	3	4	8	1	4	3	2	6	7	5	0	1	8
2	1	0	8	3	4	2	5	6	7	2	1	0	5	3	4	6	7	8	2
3	8	5	0	7	6	1	4	2	3	3	2	5	7	8	6	1	4	3	0
4	4	2	6	8	5	3	7	0	1	4	7	8	1	5	0	4	2	6	3
5	7	1	3	0	2	4	8	5	6	5	3	1	4	2	5	7	8	0	6
6	0	6	1	5	3	7	2	8	4	6	0	2	6	4	8	3	1	5	7
7	2	8	7	4	1	0	6	3	5	7	5	4	0	7	3	8	6	2	1
8	3	4	5	2	8	6	1	7	0	8	8	6	3	0	1	2	5	7	4
f2:	0	1	2	3	4	5	6	7	8	interval:	0	1	2	3	4	5	6	7	8
0	6	2	7	8	4	0	1	5	3	0	1	1	1	0	0	0	0	0	0
1	2	5	4	0	8	3	6	1	7	1	1	1	1	0	0	0	0	0	0
2	3	6	1	5	0	8	4	7	2	2	1	1	1	0	0	0	0	0	0
3	5	1	8	2	7	6	0	3	4	3	0	0	0	1	1	1	0	0	0
4	1	7	5	6	2	4	3	0	8	4	0	0	0	1	1	1	0	0	0
5	7	3	2	4	5	1	8	6	0	5	0	0	0	1	1	1	0	0	0
6	0	8	6	1	3	2	7	4	5	6	0	0	0	0	0	0	1	1	1
7	4	0	3	7	1	5	2	8	6	7	0	0	0	0	0	0	1	1	1
8	8	4	0	3	6	7	5	2	1	8	0	0	0	0	0	0	1	1	1

Table 5.1:

	A: 0	B: 1	C: 2	goals: 0	c1: 2	c2: 0	c3: 1	end_of_files :				
1	Knave:	0	1	2	Knight:	0	1	2	formulas:	0	1	2
		1	0	0		0	0	1		1	0	0

Table 5.2:

Chapter 6

Anexa

Sudoku *assign(domain_size, 9).*
assign(max_secs, 2).
set(print_models).

Cel puțin unul pe fiecare rand

all x all z exists y $S(x,y) = z$.

Cel puțin unul pe fiecare coloana

all y all z exists x $S(x,y) = z$.

*Ne folosim de intervale, pentru o tabla de joc pe 9x9 Intervalele sunt 0,1,2,3,4,5,6,7,8
,interval este o relatie de echivalenta*

Cel puțin unul pe fiecare rand

all x all z exists y $S(x,y) = z$.

Cel puțin unul pe fiecare coloana

all y all z exists x $S(x,y) = z$.

*Ne folosim de intervale, pentru o tabla de joc pe 9x9 Intervalele sunt 0,1,2,3,4,5,6,7,8
,interval este o relatie de echivalenta*

formulas(assumptions).

Cel mult unul pe fiecare rand

$S(x,y1) = S(x,y2) \rightarrow y1=y2$.

Cel mult unul pe fiecare coloana

$$S(x1,y) = S(x2,y) \rightarrow x1=x2.$$

Cel puțin unul pe fiecare rând

$$\text{all } x \text{ all } z \text{ exists } y \ S(x,y) = z.$$

Cel puțin unul pe fiecare coloana

$$\text{all } y \text{ all } z \text{ exists } x \ S(x,y) = z.$$

Ne folosim de intervale, pentru o tabla de joc pe 9x9 Intervalele sunt 0,1,2,3,4,5,6,7,8 ,interval este o relatie de echivalenta

$$\text{interval}(x,x).$$

$$\text{interval}(x,y) \rightarrow \text{interval}(y,x).$$

$$\text{interval}(x,y) \ \text{interval}(y,z) \rightarrow \text{interval}(x,z).$$

$$\text{interval}(0,1).$$

$$\text{interval}(1,2).$$

$$\text{interval}(3,4).$$

$$\text{interval}(4,5).$$

$$\text{interval}(6,7).$$

$$\text{interval}(7,8).$$

$$- \text{interval}(0,3).$$

$$- \text{interval}(3,6).$$

$$- \text{interval}(0,6).$$

Ne ajutam si de regiuni Practic aceste regiuni reprezinta celule in care se afla cifrele

$$(S(x1,y1) = S(x2,y2) \ \text{interval}(x1,x2) \ \text{interval}(y1,y2) \rightarrow x1 = x2 \ y1 = y2).$$

Starile initiale pe care le are tabla de joc

$$S(0,0)=5.$$

$$S(1,0)=6.$$

$$S(2,1)=0.$$

$$S(3,0)=8.$$

$$S(4,0)=4.$$

$$S(5,0)=7.$$

$$S(6,1)=6.$$

$$S(7,3)=4.$$

$$S(8,4)=8.$$

$$S(0,1)=3.$$

$$S(1,3)=1.$$

$$S(2,2)=8.$$

$$S(3,4)=6.$$

$$S(4,3)=8.$$

$$S(5,4)=2.$$

$$S(6,6)=2.$$

$$S(7,4)=1.$$

$$S(8,7)=7.$$

$$S(0,4)=7.$$

$$S(1,4)=0.$$

$$S(2,7)=6.$$

$$S(3,8)=3.$$

$$S(4,5)=3.$$

$$S(5,8)=6.$$

$$S(6,7)=8.$$

$$S(7,5)=0.$$

$$S(8,8)=0.$$

$$S(1,5)=5.$$

$$S(4,8)=1.$$

$$S(7,8)=5.$$

end_of_list.

formulas(assumptions).end_of_list.

Knight and knave-mace4

end_of_files.

end_of_files.

Knight and knave-prover9

formulas(assumptions).

Knight(x) | Knave(x) | Normal(x).

-Knight(x) | -Knave(x).

-Knight(x) | -Normal(x).

-Knave(x) | -Normal(x).

$x = A \mid x = B \mid x = C.$
 $A \neq B.$
 $A \neq C.$
 $B \neq C.$

$(\text{exists } x \text{ Knight}(x)).$
 $(\text{exists } x \text{ Knave}(x)).$
 $(\text{exists } x \text{ Normal}(x)).$

$\text{Knave}(A) \mid \text{Normal}(A).$
 $(\text{Knight}(B) \mid \text{Normal}(A)) \mid (\text{Knave}(B) \mid \neg \text{Normal}(A)) \mid \text{Normal}(B).$
 $\text{Knight}(C) \mid \text{Normal}(C).$

$\text{end_of_files}.$

$\text{formulas}(\text{goals}).$

$\text{Knave}(C) \text{Normal}(B) \text{Knight}(A).$

$\text{end_of_file}.$

Animale-prover9

$\text{formulas}(\text{assumptions}).$

Animale

$\text{all } x (\text{Wolf}(x) \rightarrow \text{Animal}(x)).$
 $\text{all } x (\text{Fox}(x) \rightarrow \text{Animal}(x)).$
 $\text{all } x (\text{Bird}(x) \rightarrow \text{Animal}(x)).$
 $\text{all } x (\text{Caterpillar}(x) \rightarrow \text{Animal}(x)).$
 $\text{all } x (\text{Snail}(x) \rightarrow \text{Animal}(x)).$

$\text{all } x (\text{Grain}(x) \rightarrow \text{Plante}(x)).$

$\text{exists } x \text{ Wolf}(x).$
 $\text{exists } x \text{ Fox}(x).$
 $\text{exists } x \text{ Bird}(x).$
 $\text{exists } x \text{ Caterpillar}(x).$
 $\text{exists } x \text{ Snail}(x).$

$\text{exists } x \text{ Grain}(x).$

Toate animalele fie mananca toate plantele, fie mananca toate animalele mai mici ca el care

$all\ x\ (Animal(x) \rightarrow (all\ y\ (Plante(y) \rightarrow Eats(x,y))) \mid (all\ z\ (Animal(z) \ Small(z,x) \\ (exists\ m\ (Plante(m) \ Eats(z,m))) \rightarrow Eats(x,z))))).$

$all\ x\ all\ y\ (Caterpillar(x) \ Bird(y) \rightarrow Small(x,y)).$
 $all\ x\ all\ y\ (Snail(x) \ Bird(y) \rightarrow Small(x,y)).$
 $all\ x\ all\ y\ (Bird(x) \ Fox(y) \rightarrow Small(x,y)).$
 $all\ x\ all\ y\ (Fox(x) \ Wolf(y) \rightarrow Small(x,y)).$

$all\ x\ all\ y\ (Bird(x) \ Caterpillar(y) \rightarrow Eats(x,y)).$

$all\ x\ (Caterpillar(x) \rightarrow (exists\ y\ (Plante(y) \ Eats(x,y))))).$
 $all\ x\ (Snail(x) \rightarrow (exists\ y\ (Plante(y) \ Eats(x,y))))).$

$all\ x\ all\ y\ (Wolf(x) \ Fox(y) \rightarrow -Eats(x,y)).$
 $all\ x\ all\ y\ (Wolf(x) \ Grain(y) \rightarrow -Eats(x,y)).$
 $all\ x\ all\ y\ (Bird(x) \ Snail(y) \rightarrow -Eats(x,y)).$

$end_{of_{list}}.$

$formulas(goals).$

Exista un animal care mananca(un animal care mananca toate cerealele)
 $exists\ x\ exists\ y\ (Animal(x) \ Animal(y) \ Eats(x,y) \ (all\ z\ (Grain(z) \rightarrow Eats(y,z))))).$

$end_{of_{list}}.$

n-Queens

Varianta2
 $set(arithmetic).$

2 regine pe tabla de joc Relatia Queen(a)=n reprezinta faptul ca este o regina pe randul i coloana n,prin aceasta remarca curpindem si faptul ca nu exista nici o regina pe tabla de joc.

$formulas(assumptions).$

Pentru a reprezenta scaderea m-am ajutat de negarea lui x si adunare

Aici reprezentam ca nu exista doua regine pe aceeasi coloana

$x \neq y \rightarrow Q(x) \neq Q(y).$

Nu exista doua regine pe diagonala stanga

$x \neq y \rightarrow y + -x \neq Q(y) + -Q(x).$

Nu exista doua regine pe diagonala dreapta

$x \neq y \rightarrow y + -x \neq Q(x) + -Q(y).$

end_of_list.

Varianta2

set(arithmetic).

2 regine pe tabla de joc Relatia Queen(a)=n reprezinta faptul ca este o regina pe randul i coloana n,prin aceasta remarca curpindem si faptul ca nu exista nici o regina pe tabla de joc.

formulas(assumptions).

Pentru a reprezenta scaderea m-am ajutat de negarea lui x si adunare

Aici reprezentam ca nu exista doua regine pe aceeasi coloana

$x \neq y \rightarrow Q(x) \neq Q(y).$

Nu exista doua regine pe diagonala stanga

$x \neq y \rightarrow y + -x \neq Q(y) + -Q(x).$

Nu exista doua regine pe diagonala dreapta

$x \neq y \rightarrow y + -x \neq Q(x) + -Q(y).$

end_of_list.

Let's find Wumpus. Mace4

Reguli

set(arithmetic).

assign(max_secs, 30).

assign(domain_size, 4).

assign(max_models, -1).

formulas(assumptions).

$ds = domain_size + -1.$
 $x < ds - > inc(x) = x + 1.$
 $inc(ds) = dec(ds).$
 $x > 0 - > dec(x) = x + -1.$
 $dec(0) = inc(0).$

$existsxexistsy(wumpus(x, y)).$
 $existsxexistsy(gold(x, y)).$
 $allxally(safe(x, y) < - > -wumpus(x, y) - pit(x, y)).$
 $allxally(wumpus(x, y) - > smell(x, y)).$
 $allxally(wumpus(x, y) - > smell(inc(x), y)).$
 $allxally(wumpus(x, y) - > smell(dec(x), y)).$
 $allxally(wumpus(x, y) - > smell(x, inc(y))).$
 $allxally(wumpus(x, y) - > smell(x, dec(y))).$
 $allxally(smell(x, y) - > wumpus(x, y) | wumpus(inc(x), y) | wumpus(dec(x), y) | wumpus(x, inc(y)) | wumpus(x, dec(y))).$
 $allxally(pit(x, y) - > breeze(inc(x), y)).$
 $allxally(pit(x, y) - > breeze(dec(x), y)).$
 $allxally(pit(x, y) - > breeze(x, inc(y))).$
 $allxally(pit(x, y) - > breeze(x, dec(y))).$
 $allxally(breeze(x, y) - > pit(inc(x), y) | pit(dec(x), y) | pit(x, inc(y)) | pit(x, dec(y))).$
 $allxally(wumpus(x, y) - > -pit(x, y)).$
 $allxally(wumpus(x, y) - > -gold(x, y)).$
 $allxally(gold(x, y) - > -pit(x, y)).$
 $allxally(pit(x, y) - > -gold(x, y)).$
 $allxally(wumpus(x, y) - > -(existszexistsw(wumpus(z, w)z! = xw! = y))).$
 $allxally(wumpus(x, y) - > -(existsz(wumpus(z, y)z! = x))).$
 $allxally(wumpus(x, y) - > -(existsz(wumpus(x, z)z! = y))).$
 $allxally(gold(x, y) - > -(existszexistsw(gold(z, w)z! = xw! = y))).$
 $allxally(gold(x, y) - > -(existsz(gold(z, y)z! = x))).$
 $allxally(gold(x, y) - > -(existsz(gold(x, z)z! = y))).$

Indicii

$safe(3, 0).$
 $smell(2, 0).$
 $breeze(3, 1).$
 $all\ y\ (safe(2, y)).$
 $breeze(2, 2).$
 $all\ x\ (safe(x, x)).$
 $all\ x\ all\ y\ (pit(x, y) -> -(exists\ z\ (pit(z, y)\ z \neq x))).$
 $all\ x\ all\ y\ (pit(x, y) -> -(exists\ z\ (pit(x, z)\ z \neq y))).$
 $breeze(0, 2).$
 $safe(0, 1).$
 $all\ x\ all\ y\ (breeze(x, y) -> -gold(x, y)).$
 $gold(x, y) -> x = y.$
 $-gold(0, 0).$
 $end_{of_list}.$

$formulas(goals).$
 $end_{of_list}.$

Bibliography

- 1. What Is the Name of This Book, The Riddle of Dracula and Other Logical Puzzles*
- 2. Knight and knaves*
- 3. Cursuri Adrian Groza*
- 4. Schubert's Steamroller*