VIETNAM NATIONAL UNIVERSITY, HO CHI MINH CITY

UNIVERSITY OF SCIENCE

FACULTY OF INFORMATION TECHNOLOGY

Project Report

Topic: First Order Logic

Subject: Introduction to Artificial Intelligence

Students:
Hoang Anh Tra (21127453)
Nguyen Thanh Binh (21127232)
Huynh Ba Huy (19127420)
Vu Dinh Chuong (21127236)

Instructors:
Nguyen Tien Huy
Nguyen Tran Duy Minh
Bui Duy Dang

August 21, 2023

Contents

1 Work Assignment

1.1 Member list

Student ID	Name	Role
21127232	Nguyen Thanh Binh	Member
21127453	Hoang Anh Tra	Leader
21127326	Vu Dinh Chuong	Member
19127420	Huynh Ba Huy	Member

.

1.2 Assignment

Member	Task
Nguyen Thanh Binh	Write a report on the main features of the Prolog language.
Hoang Anh Tra	Write a report on how to implement Prolog language on SWI-Prolog. Present at least 5 illustrative examples.
Vu Dinh Chuong	Give a set of at least 20 questions to ask the newly constructed knowledge system. (British Royal family)
Vu Dinh Chuong	Build a knowledge base describing the relationships of the British Royal family.
Huynh Ba Huy	Select a new topic, describing the relationships and draw the diagram of the relationship between the objects in the selected topic.
Huynh Ba Huy	Build a knowledge base with a minimum of predicates representing the underlying relations in the selected topic.
Huynh Ba Huy	Give a set of at least 20 questions to ask the newly constructed knowledge system. (New Topic)
Hoang Anh Tra	Build a logical inference program.
Nguyen Thanh Binh	Build a logical inference program.
Vu Dinh Chuong	Final test and finish the report.
Huynh Ba Huy	Final test and finish the report.

1.3 Work completeness

Id	Work	Completeness
1	1.1	100%
2	1.2	100%
3	1.3	100%

.

2 Working with the Prolog tool

2.1 The main features of Prolog language

- **Declarative programming**
  - Prolog allows you to express problems in a declarative way, without specifying how the problem should be solved.
  - In Prolog, the programmer defines a set of logical rules and facts, and the language system searches for a solution that satisfies those rules.
  - For example, to express that "a person is happy if they have a great health and money"
    happy(person) :- has_great_health(Person), has_money(Person)
- **Logic programming**
  - Prolog is based on first-order logic and allows for the representation and manipulation of logical statements and their relationships.
  - Prolog programs consist of a set of rules and facts that define the domain of discourse and a query that asks the system to find a solution to a particular problem.
  - This is an example to express that "X is the husband of Y if X is the male who married to Y:"
    husband(Person,Wife) :- married(Person,Wife), male(Person).
- **Pattern Matching**
  - Prolog allows for pattern matching which is the process of comparing a query to the rules and facts in the knowledge base. If a match is found, the system returns the result
  - Prolog uses pattern matching to unify terms and variables. For instance, to find a person who is happy:
    find_happy_person(Person) :- happy(Person).
  - When this query is run, Prolog will try to unify the variable Person with all the persons who are happy.
- **Backtracking**
  - Prolog uses backtracking to search for a solution. If the system fails to find a solution, it backtracks to the previous choice point and tries another option.
  - Prolog allows exploring multiple solutions to a problem by backtracking. In particular, to find all the happy persons:
    findall(Person, happy(Person), HappyPersons).
  - When this query is executed, Prolog will backtrack over all the possible values of Person that make happy(Person) true, and collect them in the list HappyPersons.
- **Recursion**
  - Prolog is well-suited for recursive programming, which is when a function calls itself repeatedly until a base case is reached.

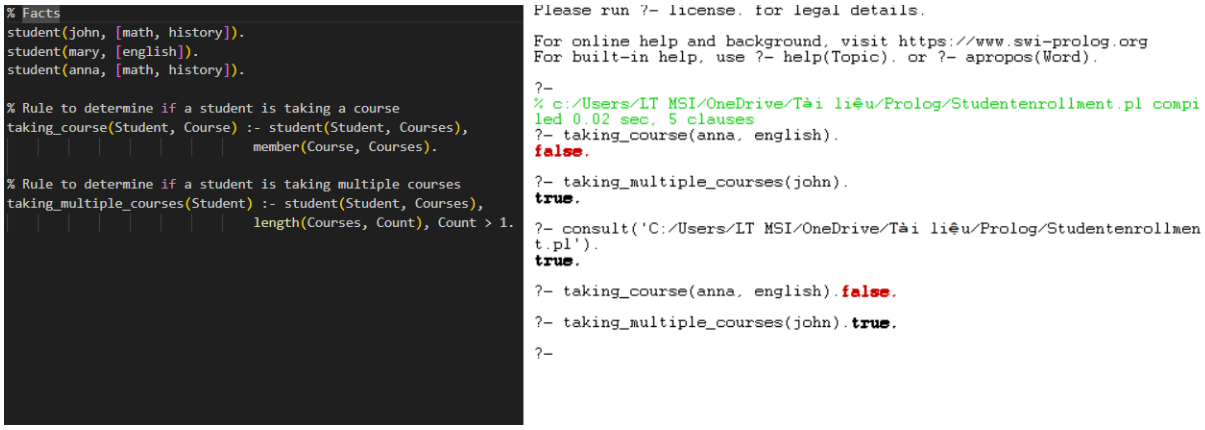
  - In Prolog, recursion is typically defined using a base case and a recursive case.
  - Assuming that having a knowledge base of parent_child relationships, represented as facts:
    parent_child(An, Hien)
    parent_child(An, Anh)
    parent_child(Hien, Truong)
    parent_child(Hien, Thao)
  - The next step is defining some rules that determines whether one person is an ancestor of another person:
    ancestor(X, Y) :- parent_child(X, Y).
    ancestor(X, Y) :- parent_child(X, Z), ancestor(Z, Y).
  - For example, if the query ancestor(An, Thao) is required, Prolog will recursively apply the second rule:
    ancestor(An, Thao) :- parent_child(An, Z), ancestor(Z, Thao).
  - This rule matches the fact parent_child(An, Hien)., so Prolog will now try to evaluate the query ancestor(Hien, Thao).. This leads to another recursive application of the second rule:
    ancestor(An, Thao) :- parent_child(An, Hien), ancestor(Hien, Thao).
    ancestor(Hien, Thao) :- parent_child(Hien, Thao).
  - This rule matches the fact parent_child(Hien, Thao)., so the query ancestor(An, Thao). succeeds.

- **Built-in predicates**
  - Prolog comes with a set of built-in predicates that can be used to solve common problems. [1]
  - One example of a built-in predicate related to the knowledge of first-order logic is 'not/1'. Assuming that ThangFamily is a valid Prolog list containing the names of members of Thang's family defined as "[Thang, Ha, Danh]", the query will evaluate to true:
    not(member(Kien, ThangFamily)).
  - Recursively, the query below will return false:
    not(member(Thang, ThangFamily)).

2.2 Implementing Prolog language on SWI-Prolog tool

2.2.1 Setting up the programming environment

- Launch SWI-Prolog and open a new file for your Prolog program.
- Write your Prolog program in the file using the correct syntax and rules.
- Save the file with a .pl extension.
- Load the Prolog program into SWI-Prolog using the consult predicate or by clicking on the "Consult" button in the SWI-Prolog GUI or can write "consult()." SWI-Prolog console.

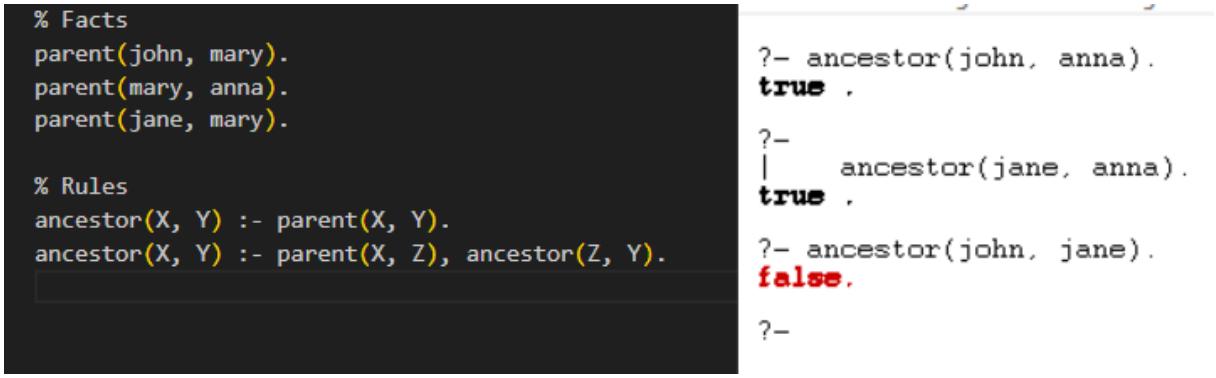


```
% Facts
student(john, [math, history]).
student(mary, [english]).
student(anna, [math, history]).

% Rule to determine if a student is taking a course
taking_course(Student, Course) :- student(Student, Courses),
                                  member(Course, Courses).

% Rule to determine if a student is taking multiple courses
taking_multiple_courses(Student) :- student(Student, Courses),
                                    length(Courses, Count), Count > 1.
```

```
Please run ?- license. for legal details.

For online help and background, visit https://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).

?-
% c:/Users/LT MSI/OneDrive/Tài liệu/Prolog/Studentenrollment.pl compi
led 0.02 sec, 5 clauses
?- taking_course(anna, english).
false.

?- taking_multiple_courses(john).
true.

?- consult('C:/Users/LT MSI/OneDrive/Tài liệu/Prolog/Studentenrollmen
t.pl').
true.

?- taking_course(anna, english).false.

?- taking_multiple_courses(john).true.

?-
```

2.2.2 Presenting 5 illustrative examples

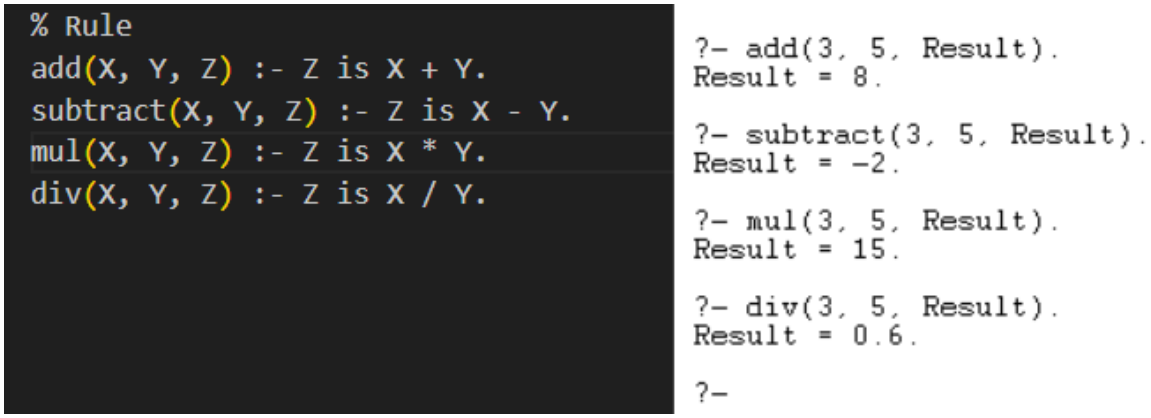
- **Family Relationships**
  - This program establishes a family tree with parent-child relations. Facts state John and Jane are parents of Mary, and Mary is a parent of Anna. Rules define ancestor relationships based on parentage.



```
% Facts
parent(john, mary).
parent(mary, anna).
parent(jane, mary).

% Rules
ancestor(X, Y) :- parent(X, Y).
ancestor(X, Y) :- parent(X, Z), ancestor(Z, Y).
```

```
?- ancestor(john, anna).
true .

?-
|    ancestor(jane, anna).
true .

?- ancestor(john, jane).
false.

?-
```

- **Arithmetic**
  - This program defines rules for basic arithmetic operations: addition, subtraction, multiplication, and division.



```
% Rule
add(X, Y, Z) :- Z is X + Y.
subtract(X, Y, Z) :- Z is X - Y.
mul(X, Y, Z) :- Z is X * Y.
div(X, Y, Z) :- Z is X / Y.
```

```
?- add(3, 5, Result).
Result = 8.

?- subtract(3, 5, Result).
Result = -2.

?- mul(3, 5, Result).
Result = 15.

?- div(3, 5, Result).
Result = 0.6.

?-
```

- **Animal Classification**

– Facts categorize animals as mammals or birds. The animal/1 rule classifies entities as animals based on these categories.

```
% Facts
mammal(dog).
mammal(cat).
bird(sparrow).
bird(pigeon).

% Rule
animal(X) :- mammal(X).
animal(X) :- bird(X).
```

```
?- animal(dog).
true .

?- animal(sparrow).
true.

?- animal(fish).
false.

?-
```

- **List Operations**

– This code demonstrates list concatenation through recursion. The append/3 rule combines lists by breaking one down and adding its elements to another.

```
% Rule
append([], L, L).
append([H|T], L, [H|R]) :- append(T, L, R).
```

```
?- append([1, 2], [3, 4], Result).
Result = [1, 2, 3, 4].

?- append([a, b], [c], Result).
Result = [a, b, c].

?- append([1, 2], [3, 4], [1, 2, 3, 4]).
true.
```

- **Student Grades**

– This program simulates student grades. Facts indicate student names and their grades. The passing/1 rule decides if a student passed (grade ≥ 70).

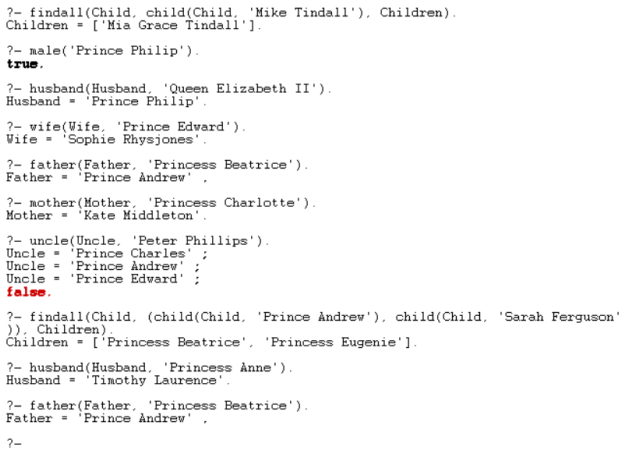
```
% Facts
grade(john, 85).
grade(mary, 92).
grade(anna, 78).

% Rule
passing(X) :- grade(X, Score), Score >= 70.
```

```
?- passing(john).
true.

?- passing(mary).
true.

?- passing(anna).
true.

?- passing(mark).
false.
```

2.3 Building a family tree of the British Royal family

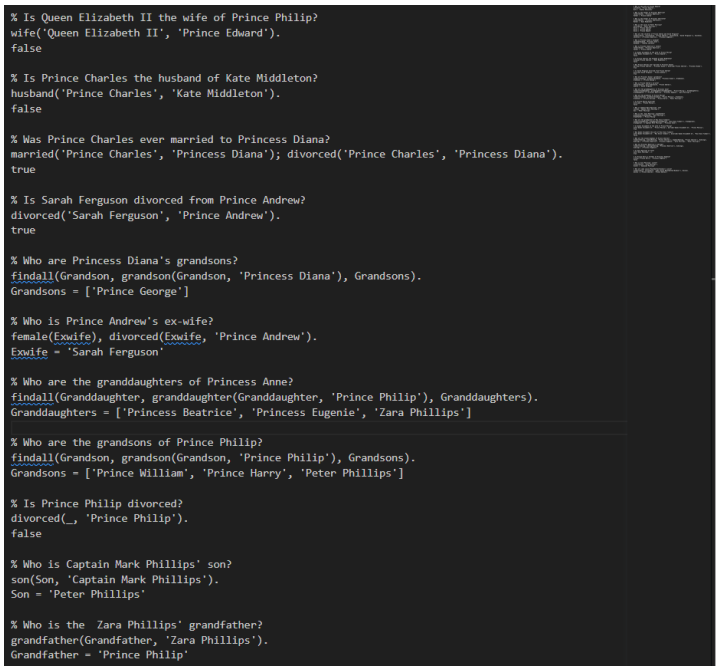
- **Tests 1-10**

```
% Who are Mike Tindall's children?
findall(Child, child(Child, 'Mike Tindall'), Children).
Children = ['Mia Grace Tindall']

% Is Prince Philip male?
male('Prince Philip').
true

% Who is the husband of Queen Elizabeth II?
husband(Husband, 'Queen Elizabeth II').
Husband = 'Prince Philip'

% Who is the wife of Prince Edward?
wife(Wife, 'Prince Edward').
Wife = 'Sophie Rhysjones'

% Who is the father of Princess Beatrice?
father(Father, 'Princess Beatrice').
Father = 'Prince Andrew'

% Who is the mother of Princess Charlotte?
mother(Mother, 'Princess Charlotte').
Mother = 'Kate Middleton'

% Who is the uncle of Peter Phillips?
uncle(Uncle, 'Peter Phillips').
Uncle = 'Prince Charles'
Uncle = 'Prince Andrew'
Uncle = 'Prince Edward'

% Who are the children of Prince Andrew and Sarah Ferguson?
findall(Child, (child(Child, 'Prince Andrew'), child(Child, 'Sarah Ferguson')), Children).
Children = ['Princess Beatrice', 'Princess Eugenie']

% Who is Princess Anne's husband?
husband(Husband, 'Princess Anne').
Husband = 'Timothy Laurence'

% Who is Princess Beatrice's father?
father(Father, 'Princess Beatrice').
Father = 'Prince Andrew'
```



```
?- findall(Child, child(Child, 'Mike Tindall'), Children).
Children = ['Mia Grace Tindall'].

?- male('Prince Philip').
true.

?- husband(Husband, 'Queen Elizabeth II').
Husband = 'Prince Philip'.

?- wife(Wife, 'Prince Edward').
Wife = 'Sophie Rhysjones'.

?- father(Father, 'Princess Beatrice').
Father = 'Prince Andrew' ,

?- mother(Mother, 'Princess Charlotte').
Mother = 'Kate Middleton'.

?- uncle(Uncle, 'Peter Phillips').
Uncle = 'Prince Charles' ;
Uncle = 'Prince Andrew' ;
Uncle = 'Prince Edward' ;
false.

?- findall(Child, (child(Child, 'Prince Andrew'), child(Child, 'Sarah Ferguson'
)), Children).
Children = ['Princess Beatrice', 'Princess Eugenie'].

?- husband(Husband, 'Princess Anne').
Husband = 'Timothy Laurence'.

?- father(Father, 'Princess Beatrice').
Father = 'Prince Andrew' ,

?-
```

- **Tests 11-21**



```
% Is Queen Elizabeth II the wife of Prince Philip?
wife('Queen Elizabeth II', 'Prince Edward').
false

% Is Prince Charles the husband of Kate Middleton?
husband('Prince Charles', 'Kate Middleton').
false

% Was Prince Charles ever married to Princess Diana?
married('Prince Charles', 'Princess Diana'); divorced('Prince Charles', 'Princess Diana').
true

% Is Sarah Ferguson divorced from Prince Andrew?
divorced('Sarah Ferguson', 'Prince Andrew').
true

% Who are Princess Diana's grandsons?
findall(Grandson, grandson(Grandson, 'Princess Diana'), Grandsons).
Grandsons = ['Prince George']

% Who is Prince Andrew's ex-wife?
female(Exwife), divorced(Exwife, 'Prince Andrew').
Exwife = 'Sarah Ferguson'

% Who are the granddaughters of Princess Anne?
findall(Granddaughter, granddaughter(Granddaughter, 'Prince Philip'), Granddaughters).
Granddaughters = ['Princess Beatrice', 'Princess Eugenie', 'Zara Phillips']

% Who are the grandsons of Prince Philip?
findall(Grandson, grandson(Grandson, 'Prince Philip'), Grandsons).
Grandsons = ['Prince William', 'Prince Harry', 'Peter Phillips']

% Is Prince Philip divorced?
divorced(_, 'Prince Philip').
false

% Who is Captain Mark Phillips' son?
son(Son, 'Captain Mark Phillips').
Son = 'Peter Phillips'

% Who is the  Zara Phillips' grandfather?
grandfather(Grandfather, 'Zara Phillips').
Grandfather = 'Prince Philip'
```

```
?- wife('Queen Elizabeth II', 'Prince Edward').
false.

?- husband('Prince Charles', 'Kate Middleton').
false.

?- married('Prince Charles', 'Princess Diana'); divorced('Princ
e Charles', 'Princess Diana').
true.

?- divorced('Sarah Ferguson', 'Prince Andrew').
true.

?- findall(Grandson, grandson(Grandson, 'Princess Diana'), Gran
dsons).
Grandsons = ['Prince George'].

?- female(Exwife), divorced(Exwife, 'Prince Andrew').
Exwife = 'Sarah Ferguson' ;
false.

?- findall(Granddaughter, granddaughter(Granddaughter, 'Prince
Philip'), Granddaughters).
Granddaughters = ['Princess Beatrice', 'Princess Eugenie', 'Zar
a Phillips'].

?- findall(Grandson, grandson(Grandson, 'Prince Philip'), Grand
sons).
Grandsons = ['Prince William', 'Prince Harry', 'Peter Phillips'
].

?- divorced(_, 'Prince Philip').
false.

?- son(Son, 'Captain Mark Phillips').
Son = 'Peter Phillips' ;
false.

?- grandfather(Grandfather, 'Zara Phillips').
Grandfather = 'Prince Philip' ;
false.

?-
```

- **Tests 22-30**

```
% Who are the grandparent of Mia Grace Tindall?
findall(Grandparent, grandparent(Grandparent, 'Mia Grace Tindall'), Grandparent).
Grandparent = ['Captain Mark Phillips', 'Princess Anne']

% Is Queen Elizabeth II the wife of Prince Philip?
wife('Queen Elizabeth II', 'Prince Philip'), married('Queen Elizabeth II', 'Prince Philip').
true

% Was Queen Elizabeth the wife of Mia Grace Tindall?
wife('Queen Elizabeth II', 'Mia Grace Tindall'), divorced('Queen Elizabeth II', 'Mia Grace
false

% Who are the nieces/nephews of Prince Charles?
findall(Nibling, niece(Nibling, 'Prince Charles'); nephew(Nibling, 'Prince Charles'), Nibli
Niblings = ['Princess Beatrice', 'Princess Eugenie', 'Zara Phillips', 'Peter Phillips']

% Who are Princess Beatrice's siblings?
findall(Sibling, sibling(Sibling, 'Princess Beatrice'), Siblings).
Siblings = ['Princess Eugenie']

% Is Zara Phillips an aunt?
aunt('Zara Phillips', _).
true

% Is Prince Harry a brother of Princess Eugenie?
brother('Prince Harry', 'Princess Eugenie').
false

% Who is Isla Phillips' sister?
sister(Sister, 'Isla Phillips').
Sister = 'Savannah Phillips'

% Who are Lady Louise Mountbatten-Windsor's uncle?
findall(Uncle, uncle(Uncle, 'Lady Louise Mountbatten-Windsor'), Uncles).
Uncles = ['Prince Charles', 'Prince Andrew']
```

```
?- findall(Grandparent, grandparent(Grandparent, 'Mia Grace Tindall'),
Grandparent).
Grandparent = ['Captain Mark Phillips', 'Princess Anne'].

?- wife('Queen Elizabeth II', 'Prince Philip'), married('Queen Elizabet
h II', 'Prince Philip').
true.

?- wife('Queen Elizabeth II', 'Mia Grace Tindall'), divorced('Queen Eli
zabeth II', 'Mia Grace Tindall').
false.

?- findall(Nibling, niece(Nibling, 'Prince Charles'); nephew(Nibling, '
Prince Charles'), Niblings).
Niblings = ['Princess Beatrice', 'Princess Eugenie', 'Zara Phillips', '
Peter Phillips'].

?- findall(Sibling, sibling(Sibling, 'Princess Beatrice'), Siblings).
Siblings = ['Princess Eugenie'].

?- aunt('Zara Phillips', _).
true .

?- brother('Prince Harry', 'Princess Eugenie').
false.

?- sister(Sister, 'Isla Phillips').
Sister = 'Savannah Phillips' ;
false.

?- findall(Uncle, uncle(Uncle, 'Lady Louise Mountbatten-Windsor'), Uncl
es).
Uncles = ['Prince Charles', 'Prince Andrew'].

?-
```

3 Build a Knowledge Base with Prolog

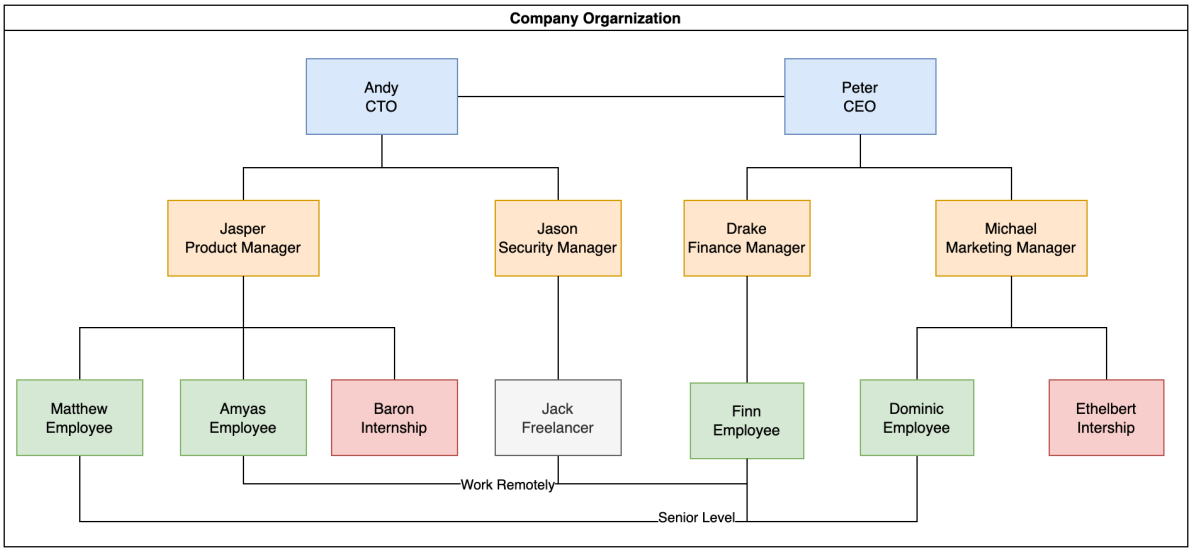
Topic: Company organization

There are 2 main kinds of objects: Manager and Employee.

- **Manager**: Andy, Peter

- **Employee**: Jasper, Jason, Drake, Micheal, Matthew, Amyas, Baron, Jack, Finn, Dominic, Ethebert

3.1 Diagram of



3.2 Knowledge Base

Knowledge Base is described clearly in the section1.2 in final zip.

- **Base predicates:**
  - cLevel(X).
  - manager(X).
  - employee(X).
  - internship(X).
  - workRemote(X).
  - senior(X).
  - female(X).
  - leadOf(X, X).
  - managementAbility(X).
  - valuable(X).
  - hadMarriage(X).
  - single(X).
  - experiment(X, number).
  - onProbation(X).
- **Predicates based on the predicates above:**
  - male(X) :- not(female(X)).
  - abilityToPromote(X) :- not(cLevel(X)).
  - workAtOffice(X) :- not(workRemote(X)).
  - commingEmployee(X) :- (internship(X); onProbation(X)), valuable(X).
  - commingSenior(X) :- abilityToPromote(X), not(manager(X); senior(X); internship(X)), valuable(X), experiment(X, ExpYear), ExpYear > 2.
  - commingManager(X) :- senior(X), managementAbility(X), valuable(X), workAtOffice(X).
  - highSalary(X):- manager(X); cLevel(X); senior(X).
  - mediumSalary(X):- not(highSalary(X)), not(internship(X)).
  - lowSalary(X) :- internship(X).

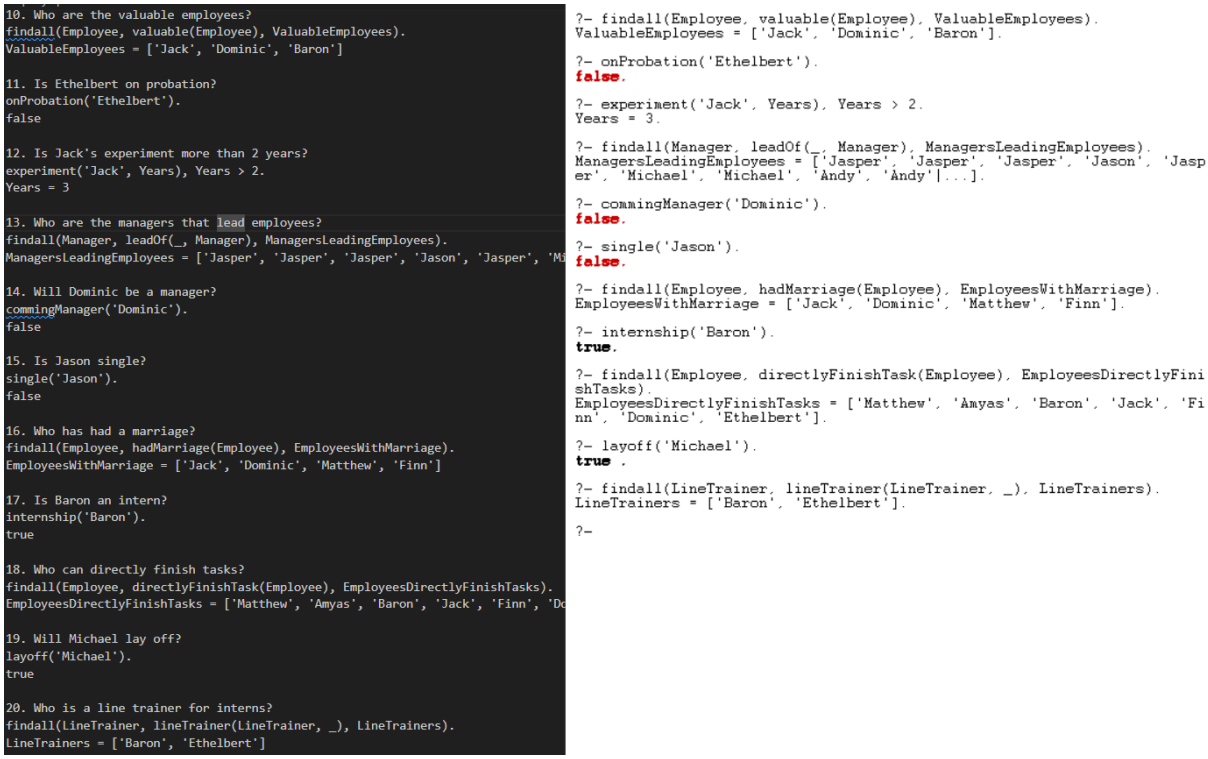
- lineTrainer(X,Y) :- internship(X), manager(Y), leadOf(X,Y).
- givingTask(X,Y) :- leadOf(X,Y).
- directlyFinishTask(X) :- leadOf(X,Y), manager(Y).
- easyFinishTask(X) :- experiment(X, Y), Y > 1.
- sameTeam(X,Y) :- (leadOf(X,Z), leadOf(Y,Z)); leadOf(X,Y); leadOf(Y,X).
- needBoyfriend(X) :- female(X), single(X).
- needGirlfriend(X) :- male(X), single(X).
- hasWife(X) :- male(X), hadMarriage(X).
- hadHusband(X) :- female(X), hadMarriage(X).
- hadTrust(X) :- workRemote(X), valuable(X).
- layoff(X) :- not(cLevel(X)), highSalary(X), not(valuable(X)).

- **Tests 1-10**

```
1. Who are the c-level employees?
findall(CLevel, cLevel(CLevel), CLevels).
CLevels = ['Andy', 'Peter']

2. Is Drake a manager?
manager('Drake').
true

3. Who are the employees under Jasper?
findall(Employee, leadOf(Employee, 'Jasper'), EmployeesUnderJasper).
EmployeesUnderJasper = ['Matthew', 'Amyas', 'Baron', 'Finn']

4. Does Amyas work remotely?
workRemote('Amyas').
true

5. Is Matthew a senior?
senior('Matthew').
true

6. Who are the male employees?
findall(Employee, (employee(Employee),male(Employee)), MaleEmployees).
MaleEmployees = ['Matthew', 'Jack', 'Finn', 'Dominic']

7. Who are the employees who need a boyfriend?
findall(Employee, needBoyfriend(Employee), EmployeesNeedBoyfriend).
EmployeesNeedBoyfriend = ['Ethelbert']

8. Does Finn have a wife?
hasWife('Finn').
true

9. Will Amyas be senior?
commingSenior('Amyas').
false

10. Who are the valuable employees?
findall(Employee, valuable(Employee), ValuableEmployees).
ValuableEmployees = ['Jack', 'Dominic', 'Baron']
```

```
?- findall(CLevel, cLevel(CLevel), CLevels).
CLevels = ['Andy', 'Peter'].

?- manager('Drake').
true.

?- findall(Employee, leadOf(Employee, 'Jasper'), EmployeesUnderJasper).
EmployeesUnderJasper = ['Matthew', 'Amyas', 'Baron', 'Finn'].

?- workRemote('Amyas').
true.

?- senior('Matthew').
true.

?- findall(Employee, (employee(Employee),male(Employee)), MaleEmployees)
.
MaleEmployees = ['Matthew', 'Jack', 'Finn', 'Dominic'].

?- findall(Employee, needBoyfriend(Employee), EmployeesNeedBoyfriend).
EmployeesNeedBoyfriend = ['Ethelbert'].

?- hasWife('Finn').
true.

?- commingSenior('Amyas').
false.

?- findall(Employee, valuable(Employee), ValuableEmployees).
ValuableEmployees = ['Jack', 'Dominic', 'Baron'].

?-
```

- **Tests 11-20**



```
10. Who are the valuable employees?
findall(Employee, valuable(Employee), ValuableEmployees).
ValuableEmployees = ['Jack', 'Dominic', 'Baron']

11. Is Ethelbert on probation?
onProbation('Ethelbert').
false

12. Is Jack's experiment more than 2 years?
experiment('Jack', Years), Years > 2.
Years = 3

13. Who are the managers that lead employees?
findall(Manager, leadOf(_, Manager), ManagersLeadingEmployees).
ManagersLeadingEmployees = ['Jasper', 'Jasper', 'Jasper', 'Jason', 'Jasper', 'Mi

14. Will Dominic be a manager?
commingManager('Dominic').
false

15. Is Jason single?
single('Jason').
false

16. Who has had a marriage?
findall(Employee, hadMarriage(Employee), EmployeesWithMarriage).
EmployeesWithMarriage = ['Jack', 'Dominic', 'Matthew', 'Finn']

17. Is Baron an intern?
internship('Baron').
true

18. Who can directly finish tasks?
findall(Employee, directlyFinishTask(Employee), EmployeesDirectlyFinishTasks).
EmployeesDirectlyFinishTasks = ['Matthew', 'Amyas', 'Baron', 'Jack', 'Finn', 'Do

19. Will Michael lay off?
layoff('Michael').
true

20. Who is a line trainer for interns?
findall(LineTrainer, lineTrainer(LineTrainer, _), LineTrainers).
LineTrainers = ['Baron', 'Ethelbert']
```

```
?- findall(Employee, valuable(Employee), ValuableEmployees).
ValuableEmployees = ['Jack', 'Dominic', 'Baron'].

?- onProbation('Ethelbert').
false.

?- experiment('Jack', Years), Years > 2.
Years = 3.

?- findall(Manager, leadOf(_, Manager), ManagersLeadingEmployees).
ManagersLeadingEmployees = ['Jasper', 'Jasper', 'Jasper', 'Jason', 'Jasp
er', 'Michael', 'Michael', 'Andy', 'Andy'|...].

?- commingManager('Dominic').
false.

?- single('Jason').
false.

?- findall(Employee, hadMarriage(Employee), EmployeesWithMarriage).
EmployeesWithMarriage = ['Jack', 'Dominic', 'Matthew', 'Finn'].

?- internship('Baron').
true.

?- findall(Employee, directlyFinishTask(Employee), EmployeesDirectlyFini
shTasks).
EmployeesDirectlyFinishTasks = ['Matthew', 'Amyas', 'Baron', 'Jack', 'Fi
nn', 'Dominic', 'Ethelbert'].

?- layoff('Michael').
true .

?- findall(LineTrainer, lineTrainer(LineTrainer, _), LineTrainers).
LineTrainers = ['Baron', 'Ethelbert'].

?-
```

4 Implement logic deductive system in the programming language

4.1 Syntax

- In this project, we have standardized the Prolog syntax in our way for easily processing and reading input files. The input files include three types of clauses which are Fact, Rule and Query.

- When it comes to Fact sentences, we still keep the same Prolog syntax. One sentence is a predicate expression that makes a declarative statement about the problem domain.
  For example: “female(’Queen Elizabeth II’).”, “parent(’Queen Elizabeth II’, ’Prince Charles’).”

- However, in Rule sentences, we have some following notes:
  - The “NOT” relation of Prolog: “not(P)” is changed to “ P”.
  - The conjunction (AND logic) is the comma “,” operator and the disjunction (OR logic) is the semi-colon “;” operator, So they are the same as

Prolog but we need to add a space after them when putting them between two predicates.

  - Our deductive system just processes the Rule in CNF form so we can not use the "->" operator like Prolog.

- In Query sentences, they must start with "?-".
- Some general restrictions must be follow by both Rule, Fact and Question:
  - The variable symbol can be written normally in both uppercase and lowercase without the apostrophe, but the constant symbol must be in apostrophe.
  - Each sentence is just expressed on one line.
- Some examples about valid input files:
  - Knowledge base:

```
/********************** Facts **********************/
female('Queen Elizabeth II').
male('James, Viscount Severn').
parent('Queen Elizabeth II', 'Princess Anne').
married('Queen Elizabeth II', 'Prince Phillip').

/************************ Rules ************************/
husband(Person, Wife) :- male(Person), female(Wife), married(Person, Wife).
wife(Person, Husband) :- female(Person), male(Husband), married(Husband, Person).
father(Parent, Child) :- parent(Parent, Child), male(Parent).
sibling(Person1, Person2) :- parent(PR, Person1), parent(PR, Person2), Person1 \== Person2.
aunt(Person, NieceNephew) :- (sister(Person, Parent); sister_in_law(Person, Parent)), parent(Parent,
NieceNephew).
nephew(Person, AuntUncle) :- male(Person), (aunt(AuntUncle, Person); uncle(AuntUncle, Person)).
```

  - Query:

```
/****************** Queries **********************/

% ============== WHO? Questions ==============
?- husband(X, 'Princess Anne')
?- wife(X, 'Prince Phillip')
?- sister('Princess Anne', X)
?- aunt(X, 'Princess Beatrice')
?- uncle(X, 'Peter Phillips')
% ============== T/F Questions ==============
?- husband('Captain Mark Phillips', 'Princess Anne')
?- husband('Timothy Laurence', 'Princess Anne')
?- wife('Kate Middleton', 'Prince William')
?- father('Prince Andrew', 'Princess Eugenie')
?- mother('Zara Phillips', 'Autumn Kelly')
```

4.2 Structure

- In this project, for the convenience of reuse and fixing bugs, we have designed some necessary classes for storing data, utility functions for reading data from files.
- In terms of data, we have some classes that are **KB**, **Fact**, **Rule** and **SingleQuestion**.
  - **KB**: This class represents the knowledge bases which may be facts or rules flexibly. It has some attributes like **name**: string, **objs**: list of string, **negative**: boolean.
  - **Fact**: This class inherits from **KB**, it is used to make an object for storing facts. The properties of this class are the same as the **KB** class.
  - **Rule**: This class inherits from **KB**, it is used to make an object for storing rules. Beside the same properties as **KB**, it also has other properties attributes like **kbs**: list of **KB**, **operator**: list of int, **isDistinct**: boolean.
  - **SingleQuestion**: This class represents the queries, it has some attributes like **name**: string, **pos**: int, **objs**: list of string, **canSolve**: boolean, **negative**: boolean.

4.3 Algorithm

- In this system, we implement **Backward chaining** [2] to solve all of the queries by using the given Knowledge base.

- **Backward chaining** is a method of reasoning in which the goal is to prove a given statement. It starts with the statement to be proved and works backward through the rules of inference to find the premises that support it.
- Based on the idea of backward chaining, when we read a query we need to find it in our rule and fact, if it's a fact statement, we just simply return the result according to that fact.
- Otherwise, if it's a rule statement, we need to loop over all of the sub-rules of that rule and solve them one by one. We find this process similar to the concept of recursion because if we have implemented a function which solves a query we can use it over and over again to solve all of the new questions that are generated by sub-rules.
- When we have solved all of the sub-rules of a rule, we will have a list of results. The next mission we have to do is combining all of these results to make a final result. The combination occurs according to the logic of AND and OR. If the operator is AND we will keep the common elements of two results. On the other hand, we will keep the union of two results if the operator is OR.
- Pseudocode:

Algorithm 1 Solve single question (**Backward chaining**)

Input: rules: list of Rule, **facts**: list of Fact, **q**: SingleQuestion
Output: result of a single question
kbsIndex ← find question **q** in **rules**
if kbsIndex = 0 **then**
    kbsIndex ← find question **q** in **facts**
    return solve question **q** by **facts**
end if
for each kbi **in** kbsIndex **do**
    result ← list of None
    questions ← list of None
    for each kb_k **in** $rules_{kbi}$ **do**
        newQuestion ← generate a new question from kb_k
        if newQuestion canSolve **then**
            key, res ← solve single question for newQuestion
        else
            $questions_k$ ← newQuestion
        end if
        $result_k$ ← (key, res)
    end for
    solve question in questions
    result ← combine result
end for
return result

4.4 Usage

- In order to use our deductive system, first of all, we have to create an input file named "kb.txt" in the same folder with the file "fol.py". In this input file, we have to declare all of the facts and rules that have syntax that satisfies all the criteria above.
- Secondly, we need to prepare a query file named "query.txt" and write some valid queries in this file, then we run the file "fol.py" to answer our desired queries. All of the answers will be written into "answer.txt".

References

[1] Subana D. Prolog-programming: Features and disadvantages — artificial intelligence.

[2] JavaTPoint. Forward chaining and backward chaining in ai.