

UNIVERSITY OF SCIENCE
VIETNAM NATIONAL UNIVERSITY, HO CHI MINH CITY



LAB REPORT

Logic

CS420 – Artificial Intelligence

Reporter

Student ID : 22125072

Student Name : Vo Thinh Phat

Theoretical Teacher : Nguyen Ngoc Thao

Laboratory Teacher : Nguyen Tran Duy Minh

Ho Chi Minh City, 22/11/2024

Contents

1	Self-assessment of the completion level	3
2	Requirement 1.1: Introduction to Prolog	3
2.1	Main Features of Prolog	3
2.1.1	Declarative Paradigm	3
2.1.2	Facts, Rules, and Queries	3
2.1.3	Unification	4
2.1.4	Backtracking	4
2.1.5	Data Structures	4
2.2	Implementing Prolog with SWI-Prolog	4
2.2.1	Installation	4
2.2.2	Setting Up the Environment	5
2.2.3	Writing and Running Prolog Programs	5
2.3	Illustrative Examples	5
2.3.1	Example 1: Family Relationships	5
2.3.2	Example 2: Simple Arithmetic	6
2.3.3	Example 3: List Processing	6
2.3.4	Example 4: Natural Language Processing	7
2.3.5	Example 5: Expert System for Diagnosing Diseases	8
3	Requirement 1.2: British Royal family	8
3.1	Problem statement	8
3.2	Explanation of Variables	9
3.3	Knowledge base, Predicates and Overall Implementation	10
3.3.1	Base Facts	11
3.3.2	Derived Predicates	12
3.3.3	Test Cases	12
3.4	Showcase the output	13
4	Requirement 1.3: Geometry World	15
4.1	Problem statement	15
4.2	Knowledge base, Predicates and Overall Implementation	16
4.2.1	Base facts	16
4.2.2	Derived predicates	18

4.2.3 Test Cases 19

4.3 Showcase the output 20

1 Self-assessment of the completion level

I have self-assessed that I have successfully completed my lab assignment. Through this lab, I have gained a good understanding of using the Prolog programming language and applying it to exercises specified in sections 1.2 and 1.3 of the requirements.

Requirements	Self-assessment
Requirement 1.1	100%
Requirement 1.2	100%
Requirement 1.3	100%

Table 1: Self-assessment of the completion level for each requirement

I would like to express my sincere gratitude to Dr. Nguyen Ngoc Thao and Mr. Nguyen Tran Duy Minh for their valuable support throughout the process of completing this lab. Their guidance and assistance have been crucial to my learning and the successful completion of this assignment.

2 Requirement 1.1: Introduction to Prolog

Prolog, short for “PROgramming in LOGic”, is a high-level programming language rooted in formal logic. Developed in the early 1970s, Prolog has been instrumental in the fields of artificial intelligence, computational linguistics, theorem proving, and expert systems. Unlike imperative languages that focus on how to perform tasks, Prolog emphasizes the what through a declarative paradigm, allowing programmers to define facts and rules about problems and let the language infer solutions through logical reasoning. [1, 2]

2.1 Main Features of Prolog

2.1.1 Declarative Paradigm

Prolog is inherently declarative, meaning that programs are expressed in terms of *what the problem is* rather than *how to solve it*. This contrasts with imperative languages where the focus is on step-by-step instructions. In Prolog, the programmer specifies relationships and constraints, and the Prolog engine deduces the solutions.

2.1.2 Facts, Rules, and Queries

Prolog programs are composed of three primary constructs:

Facts. Basic assertions about the world.

```

1 % Facts
2 parent(john, mary).
3 parent(mary, susan).
```

Rules. Logical implications that define relationships based on facts or other rules.

```
1 % Rule
2 grandparent(X, Y) :- parent(X, Z), parent(Z, Y).
```

Queries. Questions posed to the Prolog system to infer information based on the facts and rules.

```
1 % Query
2 ?- grandparent(john, susan).
```

2.1.3 Unification

Unification is the process by which Prolog matches terms. It attempts to make two terms identical by finding appropriate substitutions for variables. For example:

```
1 ?- parent(john, X).
2 X = mary.
```

2.1.4 Backtracking

Prolog employs backtracking to explore different possibilities when searching for solutions. If a particular path does not lead to a solution, Prolog automatically backtracks to try alternative paths.

```
1 parent(john, mary).
2 parent(john, mike).
3
4 ?- parent(john, X).
5 X = mary ;
6 X = mike.
```

2.1.5 Data Structures

Prolog supports various data structures, with lists being the most prominent. Lists are fundamental for representing sequences and can be manipulated using built-in predicates.

```
1 % Defining a list
2 [head | tail]
3
4 % Example
5 fruits([apple, banana, cherry]).
```

2.2 Implementing Prolog with SWI-Prolog

2.2.1 Installation

SWI-Prolog is a popular, open-source Prolog environment known for its robustness and extensive libraries. To install SWI-Prolog:

1. Download: Visit the [SWI-Prolog website](#) and navigate to the download section.
2. Choose the appropriate version for your operating system (Windows, macOS, Linux).
3. Install by following the provided instructions for your platform.

2.2.2 Setting Up the Environment

After installation:

1. Launch SWI-Prolog: You can start the interactive Prolog shell.
2. IDE options: While SWI-Prolog comes with its own editor, integrating it with editors like Visual Studio Code using extensions (e.g., [VSC-Prolog by arthurwang](#)) can enhance the development experience.
3. Directory setup: Organize your Prolog files (`.pl` extensions) in dedicated directories for better project management.

2.2.3 Writing and Running Prolog Programs

1. Create a Prolog file: Use any text editor or the SWI-Prolog editor to write your Prolog code and save it with a `.pl` extension.

```

1  % family.pl
2  parent(john, mary).
3  parent(mary, susan).
4
5  grandparent(X, Y) :- parent(X, Z), parent(Z, Y).
```

2. Load the Program: In the SWI-Prolog shell, load your program using the `[filename].` syntax.

```

1  ?- [family].
2  true.
```

3. Run Queries: Pose queries to interact with your program.

```

1  ?- grandparent(john, susan).
2  true.
```

4. Exit: To exit the SWI-Prolog environment, use `halt.` or press `Ctrl + D`.

2.3 Illustrative Examples

2.3.1 Example 1: Family Relationships

Objective: Define family relationships and query grandparent relationships.

```

1  % family.pl
2
3  % Facts
```

```

4 parent(john, mary).
5 parent(mary, susan).
6 parent(john, mike).
7 parent(mike, linda).
8
9 % Rules
10 grandparent(X, Y) :- parent(X, Z), parent(Z, Y).
11 sibling(X, Y) :- parent(Z, X), parent(Z, Y), X \= Y.
12
13 % Queries
14 ?- grandparent(john, susan).
15 % Expected Output: true.
16
17 ?- sibling(mary, mike).
18 % Expected Output: true.

```

Explanation

- Defines parent relationships.
- The grandparent rule deduces grandparent relationships based on parent facts.
- The sibling rule determines if two individuals share a common parent and are not the same person.

2.3.2 Example 2: Simple Arithmetic

Objective: Perform arithmetic operations and solve equations.

```

1 % arithmetic.pl
2
3 % Define addition
4 add(X, Y, Z) :- Z is X + Y.
5
6 % Define multiplication
7 multiply(X, Y, Z) :- Z is X * Y.
8
9 % Queries
10 ?- add(2, 3, Result).
11 % Expected Output: Result = 5.
12
13 ?- multiply(4, 5, Product).
14 % Expected Output: Product = 20.

```

Explanation

- Uses the `is` operator to evaluate arithmetic expressions.
- Allows querying for results of addition and multiplication.

2.3.3 Example 3: List Processing

Objective: Manipulate and query lists.

```

1  % list_processing.pl
2
3  % Check if an element is a member of a list
4  member(X, [X|_]).
5  member(X, [_|Tail]) :- member(X, Tail).
6
7  % Find the length of a list
8  list_length([], 0).
9  list_length([_|Tail], Length) :- list_length(Tail, N), Length is N + 1.
10
11 % Queries
12 ?- member(banana, [apple, banana, cherry]).
13 % Expected Output: true.
14
15 ?- list_length([1, 2, 3, 4], Length).
16 % Expected Output: Length = 4.

```

Explanation

- The `member` predicate checks for membership in a list.
- The `list_length` predicate recursively calculates the length of a list.

2.3.4 Example 4: Natural Language Processing

Objective: Simple parsing of sentences into grammatical structures.

```

1  % grammar.pl
2
3  % Define grammar rules
4  sentence --> noun_phrase, verb_phrase.
5
6  noun_phrase --> determiner, noun.
7
8  verb_phrase --> verb, noun_phrase.
9
10 % Lexicon
11 determiner --> [the].
12 determiner --> [a].
13
14 noun --> [cat].
15 noun --> [dog].
16 noun --> [mouse].
17
18 verb --> [chases].
19 verb --> [eats].
20
21 % Query to parse a sentence
22 ?- phrase(sentence, [the, cat, chases, the, mouse]).
23 % Expected Output: true.

```

Explanation

- Uses Definite Clause Grammars (DCG) to define simple sentence structures.
- The `phrase/2` predicate parses a list of words to see if it fits the sentence structure.

2.3.5 Example 5: Expert System for Diagnosing Diseases

Objective: Create a simple expert system that diagnoses diseases based on symptoms.

```

1  % expert_system.pl
2
3  % Facts
4  symptom(fever).
5  symptom(cough).
6  symptom(headache).
7  symptom(sore_throat).
8  symptom(rash).
9
10 % Rules
11 disease(flu) :- symptom(fever), symptom(cough), symptom(headache).
12 disease(common_cold) :- symptom(cough), symptom(sore_throat).
13 disease(measles) :- symptom(fever), symptom(rash).
14
15 % Query to diagnose
16 ?- disease(Diagnosis).
17 % Expected Output:
18 % Diagnosis = flu ;
19 % Diagnosis = common_cold ;
20 % Diagnosis = measles.
```

Explanation

- Defines possible symptoms and associates them with diseases.
- Queries the system to infer possible diseases based on available symptoms.

3 Requirement 1.2: British Royal family

To demonstrate the use of Prolog in representing complex relationships, the British Royal Family's family tree was modeled. This task involved defining relationships using Prolog facts and deriving additional relationships like using Prolog rules.

3.1 Problem statement

The task at hand involves solving a deductive problem using the Prolog programming language on the SWI-Prolog tool. The goal is to construct a knowledge base representing the family tree of the British Royal family, as shown in the Figure 1. Using this knowledge base, we will define various predicates to describe relationships within the family. After building the knowledge base, we will write queries to test the system and ensure that it correctly models these relationships.

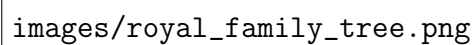
The image is a placeholder for a diagram of the British Royal Family Tree. It is represented by the text "images/royal_family_tree.png" within a large rectangular frame.

Figure 1: The British Royal Family Tree.

3.2 Explanation of Variables

In the knowledge base, several variables represent members of the British Royal family. The explanation of variables and their meanings is shown in Table 2:

Number	Variable	Corresponding Person
1	elizabeth_ii	HM Queen Elizabeth II
2	philip	Philip Duke of Edinburgh
3	charles	Prince Charles of Wales
4	diana	Diana Princess of Wales
5	anne	Princess Anne
6	mark	Mark Phillips
7	andrew	Prince Andrew Duke of York
8	sarah	Sarah Ferguson
9	edward	Prince Edward Earl of Wessex
10	sophie	Sophie Countess of Wessex
11	camilla	Camilla Duchess of Cornwall
12	timothy	Timothy Laurence
13	beatrice	Princess Beatrice
14	edoardo	Edoardo Mapelli Mozzi
15	eugenie	Princess Eugenie
16	jack	Jack Brooksbank
17	louise	Lady Louise
18	sienna	Sienna Elizabeth Mapelli Mozzi
19	august	August Brooksbank
20	james	James Viscount Severn
21	william	Prince William Duke of Cambridge
22	catherine	Catherine Duchess of Cambridge
23	harry	Prince Harry Duke of Sussex
24	meghan	Meghan Duchess of Sussex
25	peter	Peter Phillips
26	autumn	Autumn Phillips
27	zara	Zara Tindall
28	mike	Mike Tindall
29	george	Prince George
30	charlotte	Princess Charlotte
31	louis	Prince Louis
32	archie	Archie Harrison
33	lilibet	Lilibet Diana
34	savannah	Savannah Phillips
35	isla	Isla Phillips
36	mia	Mia Tindall
37	lena	Lena Tindall
38	lucas	Lucas Tindall

Table 2: The variables used in the Prolog and their corresponding names.

3.3 Knowledge base, Predicates and Overall Implementation

Based on the Explanation of Variables mentioned in Table 2. The implementation of the British Royal family tree in Prolog consists of three main parts: Base facts, Derived predicates, Test cases (file `SourceCode/knowledge_base.pl`). These components allow us to represent the family tree accurately, derive complex relationships, and verify the correctness of the model

using Prolog queries.

3.3.1 Base Facts

The base facts represent the fundamental relationships within the family tree as listed in Table 3. These facts form the foundation of the knowledge base.

Number	Fundamental Predicates	Meaning
1	parent(Parent,Child)	Parent is a parent of Child.
2	deceased(Person)	Person is deceased.
3	male(Person)	Person is male.
4	married(Person1,Person2)	Person1 and Person2 are married.
5	female(Person)	Person is female.
6	divorced(Person1,Person2)	Person1 and Person2 are divorced.

Table 3: Fundamental relationships used in the knowledge base and their meanings.

The blueprint of Prolog code for Base facts implementation is as follows:

```

1  % Base facts
2  % Parents
3  parent(elizabeth_ii, charles).
4  parent(elizabeth_ii, anne).
5  parent(elizabeth_ii, andrew).
6  parent(elizabeth_ii, edward).
7  parent(philip, charles).
8  parent(philip, anne).
9  parent(philip, andrew).
10 parent(philip, edward).
11 ...
12
13 % Deceased
14 deceased(elizabeth_ii).
15 deceased(philip).
16 deceased(diana).
17
18 % Genders
19 male(philip).
20 male(charles).
21 male(mark).
22 male(andrew).
23 ...
24 female(elizabeth_ii).
25 female(diana).
26 female(anne).
27 female(sarah).
28 ...
29
30 % Marriages
31 married(elizabeth_ii, philip).
32 married(philip, elizabeth_ii).
33 married(charles, camilla).
34 married(camilla, charles).
```

```

35 ...
36 divorced(charles, diana).
37 divorced(diana, charles).
38 divorced(anne, mark).
39 divorced(mark, anne).

```

3.3.2 Derived Predicates

Derived predicates are Prolog rules that allow us to infer relationships and characteristics beyond what is directly stated in the base facts. These predicates capture more complex relationships such as *husband*, *wife*, *father*, *mother*, *grandparent*, *grandmother*, *sibling*, *aunt*, *uncle*, *niece*, *nephew*,... The detailed Prolog implementation is as follows:

```

1  % Derived predicates
2  husband(Person, Wife) :- male(Person), married(Person, Wife).
3  wife(Person, Husband) :- female(Person), married(Person, Husband).
4  father(Parent, Child) :- male(Parent), parent(Parent, Child).
5  mother(Parent, Child) :- female(Parent), parent(Parent, Child).
6  child(Child, Parent) :- parent(Parent, Child).
7  son(Child, Parent) :- male(Child), parent(Parent, Child).
8  daughter(Child, Parent) :- female(Child), parent(Parent, Child).
9  grandparent(GP, GC) :- parent(GP, X), parent(X, GC).
10 grandmother(GM, GC) :- female(GM), grandparent(GM, GC).
11 grandfather(GF, GC) :- male(GF), grandparent(GF, GC).
12 grandchild(GC, GP) :- grandparent(GP, GC).
13 grandson(GS, GP) :- male(GS), grandchild(GS, GP).
14 granddaughter(GD, GP) :- female(GD), grandchild(GD, GP).
15 sibling(Person1, Person2) :- parent(Parent, Person1), parent(Parent, Person2),
    ↪ Person1 \= Person2.
16 brother(Person, Sibling) :- male(Person), sibling(Person, Sibling).
17 sister(Person, Sibling) :- female(Person), sibling(Person, Sibling).
18 aunt(Person, NieceNephew) :- female(Person), sibling(Person, Parent), parent(Parent,
    ↪ NieceNephew).
19 uncle(Person, NieceNephew) :- male(Person), sibling(Person, Parent), parent(Parent,
    ↪ NieceNephew).
20 niece(Person, AuntUncle) :- female(Person), parent(Parent, Person), sibling(Parent,
    ↪ AuntUncle).
21 nephew(Person, AuntUncle) :- male(Person), parent(Parent, Person), sibling(Parent,
    ↪ AuntUncle).

```

3.3.3 Test Cases

To validate the knowledge base and ensure the correctness of derived predicates, several Prolog queries were written as test cases. These queries check the expected relationships within the family tree.

I implemented four types of questions:

- **Who is ... of ...?** (e.g., Who is the mother of Charles?)
- **Is ...?** (e.g., Is Elizabeth II deceased?)

- **Who is ... among ...?** (e.g., Who is the male among Charles, Anne, and Harry?)
- **Are there any ...?** (e.g., Are there any daughters of Philip?)
- **How many ...?** (e.g., How many grandmothers does Zara have?)

Examples of test cases and its implementation:

```

1 :- write('Q1: Who is the mother of Charles? '),
2   (findall(Mother, mother(Mother, charles), Mothers), list_to_set(Mothers,
3     ↪ UniqueMothers),
4     (UniqueMothers = [] -> write('not found'), nl; write(UniqueMothers), nl)).
5 ...
6 :- write('Q4: Is Elizabeth II deceased? '),
7   (deceased(elizabeth_ii) -> write('Yes'), nl; write('No'), nl).
8 ...
9
10 :- write('Q8: Who is the male among Charles, Anne, and Harry? '),
11   (findall(MalePerson, (member(MalePerson, [charles, anne, harry]),
12     ↪ male(MalePerson)), Males),
13   (Males = [] -> write('No one'), nl; write(Males), nl)).
14 ...
15 :- write('Q38: Are there any daughters of Philip? '),
16   (findall(Daughter, daughter(Daughter, philip), Daughters),
17   (Daughters = [] -> write('No'), nl; write('Yes'), nl)).
18 ...
19
20 :- write('Q45: How many grandmothers does Zara have? '),
21   (findall(GM, grandmother(GM, zara), Grandmothers),
22   (Grandmothers = [] -> write('0'), nl; length(Grandmothers, Count), write(Count),
23     ↪ nl)).

```

3.4 Showcase the output

Run the knowledge_base.pl file in SourceCode Folder, we will obtain the following output:

```

1 ?- [knowledge_base].
2 Q1: Who is the mother of Charles? [elizabeth_ii]
3 Q2: Who is the father of Beatrice? [andrew]
4 Q3: Who is the father of Harry? [charles]
5 Q4: Is Elizabeth II deceased? Yes
6 Q5: Who is deceased among Elizabeth II, Diana, and Philip?
7   ↪ [elizabeth_ii, philip, diana]
8 Q6: Who is deceased: Elizabeth II, Philip, or Charles? [elizabeth_ii, philip]
9 Q7: Who is male: Charles, Anne, or William? [charles, william]
10 Q8: Who is the male among Charles, Anne, and Harry? [charles, harry]
11 Q9: Is Andrew male? Yes
12 Q10: Who is married to Elizabeth II? [philip]
13 Q11: Who is married to Charles? [camilla]
14 Q12: Who is married to Anne? [timothy]
15 Q13: Who is female: Diana, Philip, or William? [diana]

```

15 Q14: Who is **female**: Zara, William, or Harry? [zara]
16 Q15: Is Diana female? Yes
17 Q16: Who is divorced from Charles? [diana]
18 Q17: Who is divorced from Anne? [mark]
19 Q18: Who is divorced from Andrew? [sarah]
20 Q19: Who is the husband of Elizabeth II? [philip]
21 Q20: Who is the husband of Camilla? [charles]
22 Q21: Who is the husband of Anne? [timothy]
23 Q22: Who is the wife of Philip? [elizabeth_ii]
24 Q23: Who is the wife of Charles? [camilla]
25 Q24: Who is the wife of Andrew? [sarah]
26 Q25: Who is the father of William? [charles]
27 Q26: Who is the father of Harry? [charles]
28 Q27: Who is the father of Zara? [mark]
29 Q28: Who is the mother of William? [diana]
30 Q29: Who is the mother of Beatrice? [sarah]
31 Q30: Who is the mother of Andrew? [elizabeth_ii]
32 Q31: Who is the child of Elizabeth II? [charles,anne,andrew,edward]
33 Q32: Who is the child of Philip? [charles,anne,andrew,edward]
34 Q33: Who is the child of Charles? [william,harry]
35 Q34: Who are the sons of Elizabeth II? [andrew,charles,edward]
36 Q35: How many sons does Philip have? 3
37 Q36: Which son of Diana is also the son of Charles? [harry,william]
38 Q37: Who is the daughter of Elizabeth II? [anne]
39 Q38: Are there any daughters of Philip? Yes
40 Q39: How many daughters does Diana have? 0
41 Q40: Who are the grandparents of William? [elizabeth_ii,philip]
42 Q41: Which grandparents does Beatrice have? [elizabeth_ii,philip]
43 Q42: How many grandparents does Zara have? 2
44 Q43: Which grandmother is related to William? [elizabeth_ii]
45 Q44: Who is the grandmother of Beatrice? [elizabeth_ii]
46 Q45: How many grandmothers does Zara have? 1
47 Q46: Who is the grandfather of William? [philip]
48 Q47: How many grandfathers does Beatrice have? 1
49 Q48: Are there any grandfathers of Zara? Yes
50 Q49: How many grandchildren does Elizabeth II have? 8
51 Q50: Which grandchildren are related to Diana?
↪ [archie,charlotte,george,lilibet,louis]
52 Q51: Who is the grandchild of Philip?
↪ [beatrice,eugenie,harry,james,louise,peter,william,zara]
53 Q52: Who is the grandson of Elizabeth II? [harry,james,peter,william]
54 Q53: Are there any grandsons of Diana? Yes
55 Q54: How many grandsons does Philip have? 4
56 Q55: Which granddaughter is related to Elizabeth II? [beatrice,eugenie,louise,zara]
57 Q56: Who are the granddaughters of Diana? [charlotte,lilibet]
58 Q57: How many granddaughters does Philip have? 4
59 Q58: Are Charles and Anne siblings? Yes
60 Q59: How many siblings does Edward have? 6
61 Q60: Who are the siblings of Charles? [andrew,anne,edward]
62 Q61: Who is the brother of Anne? [andrew,charles,edward]
63 Q62: How many brothers does Beatrice have? 0
64 Q63: Are there any brothers of Sarah? No
65 Q64: Who is the sister of William? not found

```
66 Q65: How many sisters does Charles have? 2
67 Q66: Who are the sisters of Edward? [anne]
68 Q67: Who is the aunt of Beatrice? [anne]
69 Q68: Who is the aunt of Sienna? [eugenie]
70 Q69: Which aunt is related to Eugenie? [anne]
71 Q70: Are there any aunts of Beatrice? Yes
72 Q71: How many aunts does Sienna have? 2
73 Q72: Who is the uncle of Beatrice? [charles,edward]
74 Q73: Who is the uncle of Sienna? not found
75 Q74: Which uncle is related to Eugenie? [charles,edward]
76 Q75: Are there any uncles of Beatrice? Yes
77 Q76: How many uncles does Sienna have? 0
78 Q77: Who is the niece of Anne? [beatrice,eugenie,louise]
79 Q78: Who is the niece of Sarah? not found
80 Q79: Which niece is related to Beatrice? not found
81 Q80: How many nieces does Anne have? 6
82 Q81: Are there any nieces of Sarah? No
83 Q82: Who is the nephew of Anne? [harry,james,william]
84 Q83: Who is the nephew of Sarah? not found
85 Q84: Which nephew is related to Beatrice? [august]
86 Q85: How many nephews does Anne have? 6
87 Q86: Are there any nephews of Sarah? No
88 true.
```

4 Requirement 1.3: Geometry World

4.1 Problem statement

Imagine a world named **Geometry** where each citizen is characterized by a unique name and a distinct shape. The classification of these citizens is presented in Figure 2, which illustrates the hierarchy and relationships between various plane geometry shapes. For example:

- Martin is a citizen with a square shape.
- Ava is a citizen with a quadrilateral shape, her children are also some of her variants, such as a trapezoid, rhombus, or square.

One special feature of Geometry World is that **each child has at least one property that their parent does not have**. For instance, Martin, as a square, has two perpendicular diagonals—a property not found in his parent, Lee (a rectangle).

The objective is to model the relationships and properties of these geometric citizens using logical predicates in Prolog. The citizens of Geometry include circles, polygons, triangles, quadrilaterals, and their specialized types, such as squares, rectangles, rhombuses, trapezoids, and kites.

Geometry World

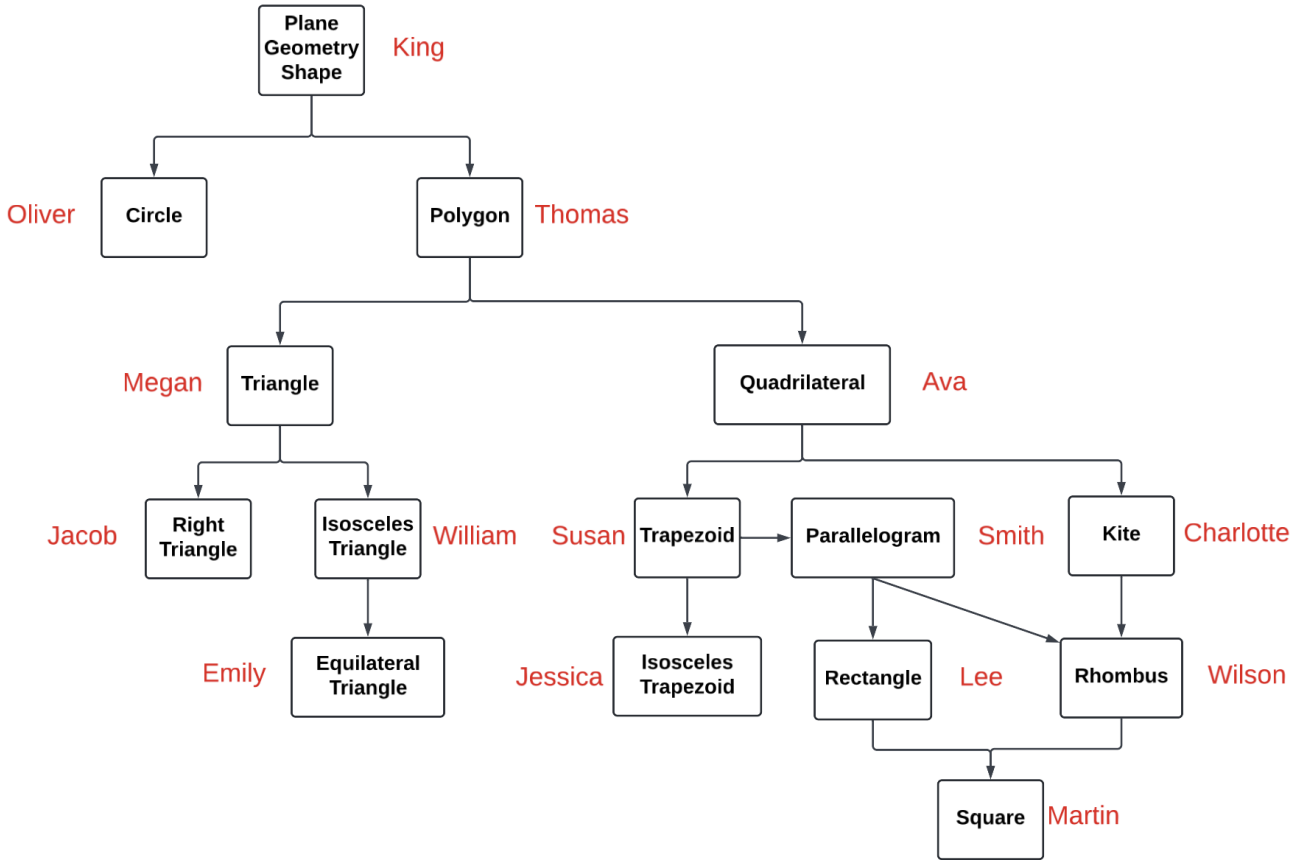


Figure 2: Hierarchy of Plane Geometry Shapes in the **Geometry World**

4.2 Knowledge base, Predicates and Overall Implementation

In this section, I will define the predicates that represent the properties and relationships between geometric citizens in the Geometry World. These predicates form the backbone of our knowledge base, enabling us to model the characteristics of each shape and their hierarchical connections. By using these predicates, we can reason about the relationships between different types of shapes, such as how one shape can be a variant of another or how certain properties are inherited or extended in specialized shapes.

The knowledge base provides the foundation for the modeling of the citizens of Geometry World, where each citizen corresponds to a unique geometric shape, and the relationships among shapes reflect their geometrical properties. For example, a square is a specialized version of a rectangle, possessing additional properties such as orthogonal diagonals.

The Prolog implementation is put in `SourceCode2/knowledge_base.pl`.

4.2.1 Base facts

Base facts in our knowledge base define the essential characteristics of various shapes. These facts are established to capture the relationships that are universal and apply to all shapes in the Geometry World. The detailed description of Base facts is shown in Table 4.

N.o	Fundamental Predicates	Meaning
1	parent(Parent,Child)	Child is a variant of Parent. Child has at least one property that Parent doesn't have.
2	isCircle(Shape)	Shape is a circle.
3	isPolygon(Shape)	Shape is a polygon.
4	hasThreeEdges(Shape)	Shape has exact three edges.
5	hasFourEdges(Shape)	Shape has exact four edges.
6	containTwoEqualEdges(Shape)	Shape contains at least two equal edges.
7	containRightAngle(Shape)	Shape contains at least one right angle.
8	containParalellEdges(Shape)	Shape contains at least two parallel edges.
9	hasOrthogonalDiagonals(Shape)	Shape has two orthogonal diagonals.

Table 4: Fundamental relationships used in the knowledge base and their meanings.

The blueprint of Prolog code for Base facts implementation is as follows:

```

1  % Base facts
2  % Parents
3  parent(king, oliver).
4  parent(king, thomas).
5
6  parent(thomas, megan).
7  parent(thomas, ava).
8
9  parent(megan, jacob).
10 parent(magan, william).
11 parent(ava, susan).
12 parent(ava, charlotte).
13 ...
14
15 % Shapes
16 isCircle(oliver).
17
18 isPolygon(thomas).
19 isPolygon(megan).
20 isPolygon(ava).
21 isPolygon(jacob).
22 ...
23
24 % Number of Edges
25 hasThreeEdges(megan).
26 hasThreeEdges(jacob).
27 hasFourEdges(ava).
28 hasFourEdges(susan).
29 ...
30
31 % Equal Edges
32 containTwoEqualEdges(william).
33 containTwoEqualEdges(emily).
34 containTwoEqualEdges(jessica).
35 ...
36
37 % Right Angle

```

```

38 containRightAngle(jacob).
39 containRightAngle(lee).
40 containRightAngle(martin).
41 ...
42
43 % Parallel edges
44 containParalellEdges(susan).
45 containParalellEdges(smith).
46 ...
47
48 % Orthogonal diagonals
49 hasOrthogonalDiagonals(charlotte).
50 hasOrthogonalDiagonals(wilson).

```

4.2.2 Derived predicates

Derived predicates are logical rules built upon the base facts. These predicates allow us to infer new properties and relationships that are not explicitly stated in the knowledge base. By combining different base facts, we can derive additional properties about shapes, such as whether a polygon has equal edges, right angles, or orthogonal diagonals.

The meaning of each derived predicate is shown in Table 5.

N.o	Derived Predicates	Meaning
1	isTriangle(X)	X is a triangle.
2	isQuadrilateral(X)	X is a quadrilateral.
3	isRightTriangle(X)	X is a right triangle.
4	isIsoscelesTriangle(X)	X is an isosceles triangle.
5	isEquilateralTriangle(X)	X is an equilateral triangle.
6	isTrapezoid(X)	X is a trapezoid.
7	isKite(X)	X is a kite.
8	isRhombus(X)	X is a rhombus.
9	isParallelogram(X)	X is a parallelogram.
10	isIsoscelesTrapezoid(X)	X is an isosceles trapezoid.
11	isRectangle(X)	X is a rectangle.
12	isSquare(X)	X is a square.
13	isScaleneTriangle(X)	X is a scalene triangle.
14	isSymmetric(X)	X is symmetric.
15	isAlwaysCyclic(X)	X is always cyclic.
16	isAlwaysConvex(X)	X is always convex.
17	isSuccessor(X, Y)	X is a descendant of Y.
18	isAbleConcave(X)	X can be concave.
19	isAbleNonCyclic(X)	X can be non-cyclic.
20	isNonSymmetric(X)	X is non-symmetric.
21	hasDiagonalsIntersectAtMidpoint(X)	X has diagonals intersecting at midpoint.

Table 5: Derived predicates built upon the Base facts.

The detailed Prolog implementation is as follows:

```

1  % Derived predicates
2  isTriangle(X) :- hasThreeEdges(X).
3  isQuadrilateral(X) :- hasFourEdges(X).
4  isRightTriangle(X) :- isTriangle(X), containRightAngle(X).
5  isIsoscelesTriangle(X) :- isTriangle(X), containTwoEqualEdges(X).
6  isEquilateralTriangle(X) :- isIsoscelesTriangle(X), parent(william, X).
7  isTrapezoid(X) :- isQuadrilateral(X), containParalellEdges(X).
8  isKite(X) :- isQuadrilateral(X), hasOrthogonalDiagonals(X), containTwoEqualEdges(X).
9  isRhombus(X) :- isKite(X), containParalellEdges(X).
10 isParallelogram(X) :- isTrapezoid(X), parent(X, lee).
11 isIsoscelesTrapezoid(X) :- isTrapezoid(X), containTwoEqualEdges(X).
12 isRectangle(X) :- isParallelogram(X), containRightAngle(X).
13 isSquare(X) :- isRectangle(X), isRhombus(X).
14 isScaleneTriangle(X) :- isTriangle(X), \+ containTwoEqualEdges(X).
15 isSymmetric(X) :-
16     (isSquare(X);
17     isRectangle(X);
18     isRhombus(X);
19     isKite(X),
20     containParalellEdges(X),
21     containTwoEqualEdges(X)).
22 isAlwaysCyclic(X) :-
23     (isTriangle(X);
24     isRectangle(X);
25     isSquare(X);
26     isIsoscelesTrapezoid(X)).
27 isAlwaysConvex(X) :-
28     (isTriangle(X);
29     isTrapezoid(X);
30     isParallelogram(X)).
31 isSuccessor(X, Y) :-
32     (parent(Y, X);
33     parent(Y, Z), isSuccessor(X, Z)).
34 isAbleConcave(X) :-
35     (isPolygon(X), \+ isAlwaysConvex(X)).
36 isAbleNonCyclic(X) :-
37     (isPolygon(X), \+ isAlwaysCyclic(X)).
38 isNonSymmetric(X) :- \+ isSymmetric(X).
39 hasDiagonalsIntersectAtMidpoint(X) :-
40     (isRectangle(X);
41     isKite(X)).

```

4.2.3 Test Cases

Similar to section 3.3.3, to validate the knowledge base and ensure the correctness of derived predicates, several Prolog queries were written as test cases. Examples of test cases and its implementation:

```

1 :- write('Q1: Is Thomas parent of Ava? '),
2     (parent(thomas, ava) -> write('Yes'), nl; write('No'), nl).
3 ...

```

```

4
5 :- write('Q10: Does Megan have three edges? '),
6     (hasThreeEdges(megan) -> write('Yes'), nl; write('No'), nl).
7 ...
8
9 :- write('Q19: List all polygons that contain a right angle: '),
10    (findall(Polygon, containRightAngle(Polygon), Polygons), list_to_set(Polygons,
11    ↪ UniquePolygons)),
12    (UniquePolygons = [] -> write('No polygons contain a right angle'), nl;
13    ↪ write(UniquePolygons), nl).
14 ...

```

4.3 Showcase the output

Run the `knowledge_base.pl` file in SourceCode Folder, we will obtain the following output:

```

1 ?- [knowledge_base].
2 Q1: Is Thomas parent of Ava? Yes
3 Q2: Who is parent of Martin? [lee,wilson]
4 Q3: Is King parent of Oliver? Yes
5 Q4: Is Oliver a Circle? Yes
6 Q5: Is Thomas a Circle? No
7 Q6: Is Megan a Circle? No
8 Q7: Is Thomas a Polygon? Yes
9 Q8: Is Megan a Polygon? Yes
10 Q9: Is Ava a Polygon? Yes
11 Q10: Does Megan have three edges? Yes
12 Q11: Does Jacob have three edges? Yes
13 Q12: Does William have three edges? Yes
14 Q13: Are there any triangles with four edges? Yes
15 Q14: Does Ava have four edges? Yes
16 Q15: Does Susan have four edges? Yes
17 Q16: Does William contain two equal edges? Yes
18 Q17: Does Emily contain two equal edges? Yes
19 Q18: Does Jessica contain two equal edges? Yes
20 Q19: List all polygons that contain a right angle: [jacob,lee,martin]
21 Q20: Does Jacob contain a right angle? Yes
22 Q21: Does Lee contain a right angle? Yes
23 Q22: Are there any polygons with parallel edges? Yes
24 Q23: Does Susan contain parallel edges? Yes
25 Q24: Does Smith contain parallel edges? Yes
26 Q25: How many polygons have orthogonal diagonals? 3
27 Q26: Does Charlotte have orthogonal diagonals? Yes
28 Q27: Are there any polygons that have orthogonal diagonals? Yes
29 Q28: Is Jacob a Triangle? Yes
30 Q29: Is Megan a Triangle? Yes
31 Q30: Is William a Triangle? Yes
32 Q31: How many polygons are Quadrilaterals? 8
33 Q32: Is Ava a Quadrilateral? Yes
34 Q33: Is Susan a Quadrilateral? Yes
35 Q34: How many polygons are Right Triangles? 1
36 Q35: Is Jacob a Right Triangle? Yes

```

```

37 Q36: Is Lee a Right Triangle? No
38 Q37: How many polygons are Isosceles Triangles? 2
39 Q38: Is Jacob an Isosceles Triangle? No
40 Q39: Is William an Isosceles Triangle? Yes
41 Q40: List out all Equilateral Triangles: [emily]
42 Q41: Is William an Equilateral Triangle? No
43 Q42: Is Emily an Equilateral Triangle? Yes
44 Q43: How many polygons are Trapezoids? 6
45 Q44: Is Ava a Trapezoid? No
46 Q45: Is Susan a Trapezoid? Yes
47 Q46: How many polygons are Kites? 3
48 Q47: Is Charlotte a Kite? Yes
49 Q48: List out all Kites: [charlotte,wilson,martin]
50 Q49: How many polygons are Rhombus? 2
51 Q50: Is Charlotte a Rhombus? No
52 Q51: Is Wilson a Rhombus? Yes
53 Q52: How many polygons are Parallelograms? 1
54 Q53: Is Lee a Parallelogram? No
55 Q54: Is Susan a Parallelogram? No
56 Q55: Are there any Isosceles Trapezoids? Yes
57 Q56: Is Jessica an Isosceles Trapezoid? Yes
58 Q57: List out all Isosceles Trapezoids: [smith,jessica,lee,wilson,martin]
59 Q58: How many polygons are Rectangles? 0
60 Q59: Is Lee a Rectangle? No
61 Q60: Is Smith a Rectangle? No
62 Q61: How many polygons are Squares? 0
63 Q62: Is Martin a Square? No
64 Q63: Is Lee a Square? No
65 Q64: How many polygons are Scalene Triangles? 2
66 Q65: Is Jacob a Scalene Triangle? Yes
67 Q66: Is Megan a Scalene Triangle? Yes
68 Q67: How many polygons are Symmetric? 17
69 Q68: Is Martin a Symmetric polygon? Yes
70 Q69: Is Lee a Symmetric polygon? Yes
71 Q70: How many polygons are always cyclic? 9
72 Q71: Is Jacob always cyclic? Yes
73 Q72: List out all polygons that are always cyclic:
    ↪ [megan,jacob,william,emily,smith,jessica,lee,wilson,martin]
74 Q73: How many polygons are always convex? 11
75 Q74: Is Jacob always convex? Yes
76 Q75: List out all polygons that are always convex:
    ↪ [megan,jacob,william,emily,susan,smith,jessica,lee,wilson,martin]
77 Q76: How many polygons are able to be concave? 3
78 Q77: Is Jacob able to be concave? No
79 Q78: List out all polygons that are able to be concave: [thomas,ava,charlotte]
80 Q79: How many polygons are able to be non-cyclic? 4
81 Q80: Is Jacob able to be non-cyclic? No
82 Q81: List out all polygons that are able to be non-cyclic:
    ↪ [thomas,ava,susan,charlotte]
83 Q82: How many polygons are non-symmetric? 5
84 Q83: Is Jacob non-symmetric? Yes
85 Q84: List out all polygons that are non-symmetric: [thomas,megan,ava,jacob,susan]
86 Q85: How many polygons have diagonals intersect at midpoint? 3

```

```
87 Q86: Does Charlotte have diagonals intersect at midpoint? Yes
88 Q87: List out all polygons that have diagonals intersect at midpoint:
    ↪ [charlotte,wilson,martin]
89 Q88: Is Thomas a successor of King? Yes
90 Q89: Is Ava a successor of King? Yes
91 Q90: Is Martin a successor of King? Yes
92 Q91: Is Martin a successor of Thomas? Yes
93 Q92: Is Martin a successor of Ava? Yes
94 Q93: Is Martin a successor of Susan? Yes
95 Q93: Is Martin a successor of Susan? Yes
96 Q94: Is Martin a successor of Emily? No
97 Q95: Is Martin a successor of Oliver? No
98 true.
```

References

- [1] Merritt, D. (2012). Adventure in Prolog. Springer New York. ISBN: 9781461234265.
<https://books.google.com.vn/books?id=hfHSBwAAQBAJ>
- [2] Bratko, I. (2001). Prolog Programming for Artificial Intelligence. Addison-Wesley.