UNIVERSITY OF ASIA PACIFIC

Department of Computer Science and Engineering(CSE)

Course Code: CSE 404
Course Title: Artificial Intelligence and Expert Systems Lab

Project 01 : Harry Potter Family Tree with Prolog

Submitted By:
Md Yasin Hossain Akash
Reg. ID: 22101153
Section - D

Submitted To:
Bidita Sarkar Diba
Lecturer
Computer Science and Engineering

Problem Statement:

The goal of this project is to create a knowledge base in Prolog that represents the extended family relationships within the Harry Potter universe. This knowledge base enables querying of both direct (e.g., parent-child, siblings) and indirect (e.g., ancestors, grandparents) relationships across several major wizarding families using recursive rules.

Problem Description:

Representing complex family relationships such as those in the Harry Potter series involves defining facts and rules that capture the parent-child relationships, gender information, and other familial connections. Prolog's logical programming paradigm is well-suited for this task, allowing us to easily encode:

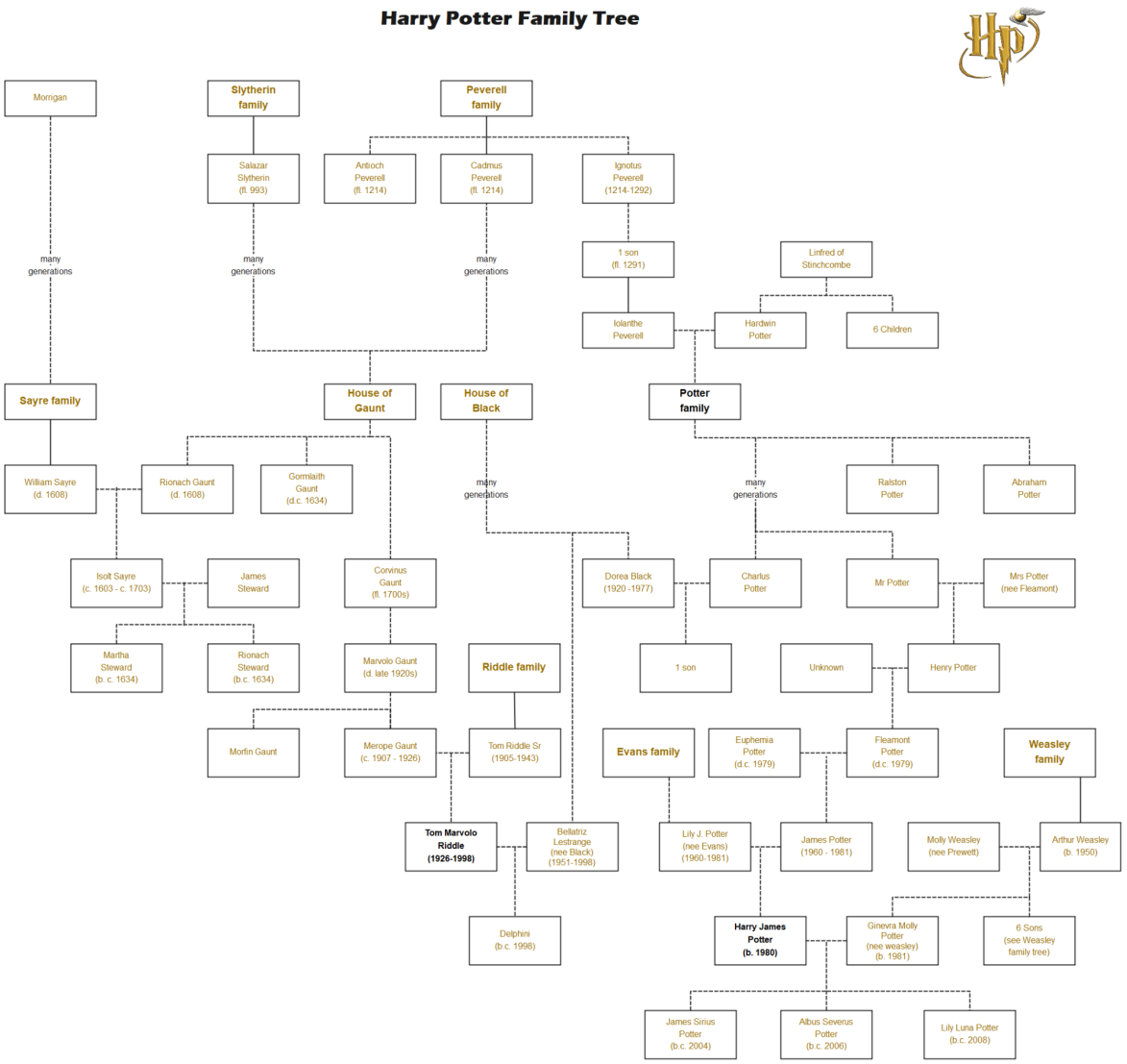
- Parent-child relationships
- Gender distinctions (male, female)
- Derived relationships like father, mother, siblings, grandparents, and ancestors

Additionally, recursion plays a critical role in querying indirect relations like ancestors, which may span multiple generations. This allows the system to answer complex family queries such as "Who are Harry Potter's ancestors?" or "Who are the siblings of Ron Weasley?".

Tools and Languages Used:

- **Prolog (SWI-Prolog):** For encoding family facts and rules, and executing queries.
- **Canva:** To create visual diagrams of the family tree and relationships for presentation.
- **Microsoft Word:** To write and format this report documenting the design, implementation, and results.

Diagram:



Sample Input / Output:

```
?- female(harry_potter).
false.

?- male(harry_potter).
true.
```

```
?- gradparent(X, harry_potter).
Correct to: "grandparent(X,harry_potter)"?
Please answer 'y' or 'n'? yes
X = fleamont_potter .

?- grandparent(X, harry_potter).
X = fleamont_potter ;
X = euphemia_potter ;
X = mr_evans ;
X = mrs_evans ;
false.
```

```
?- ancestor(X, lily_luna_potter).
X = harry_potter ;
X = ginny_weasley ;
X = fleamont_potter ;
X = euphemia_potter ;
X = james_potter ;
X = lily_potter ;
X = mr_evans ;
X = mrs_evans ;
X = arthur_weasley ;
X = molly_weasley ;
false.
```

```
?- parent(harry_potter, X).
X = james_sirius_potter .

?- parent(harry_potter, X).
X = james_sirius_potter ;
X = albus_severus_potter ;
X = lily_luna_potter.

?- gradparent(X, harry_potter).
Correct to: "grandparent(X,harry_potter)"?
Please answer 'y' or 'n'? yes
X = fleamont_potter .

?- grandparent(X, harry_potter).
X = fleamont_potter ;
X = euphemia_potter ;
X = mr_evans ;
X = mrs_evans ;
false.
```

Conclusion:

This project demonstrates that Prolog is an excellent tool for representing and querying complex family trees. Its logical syntax allows straightforward definition of basic relations and recursive rules to explore extended genealogies. The recursive ancestor predicate, in particular, showcases the power of logic programming in handling indirect and multi-generational queries seamlessly.

Find All Files Here: