Presented to the College of Computer Studies

De La Salle University - Manila

Term 3, A.Y. 2024-2025

In partial fulfillment of the course

In CSINTSY S11

MCO2 Report

Group No. 3

Submitted by:

Arianne Yari Artus

Chastine Cabatay

Gerylyn Guiller

Heather Lynn Soper

Submitted to:

Dr. Norshuhani Zamin

July 31, 2025

Introduction

Chatbots have become increasingly prevalent in various domains, from customer service to personal virtual assistants, due to their ability to simulate human-like conversation and provide automated responses. While many modern chatbots leverage machine learning and large language models (LLMs) like ChatGPT to generate responses dynamically [1], another class of chatbots relies on rule-based systems rooted in symbolic AI. These logic-driven chatbots follow explicitly defined rules, offering more predictable and explainable behavior, especially in domains with clear relationships and constraints [2].

This project explores the development of a logic-based chatbot that models familial relationships using first-order logic. Implemented with PROLOG as the inference engine and a Python frontend using the pyswip library, the chatbot processes structured user inputs to store facts or answer relationship queries. It recognizes and infers family connections—such as siblings, grandparents, and uncles—based on the facts encoded in its knowledge base. The project serves as a practical application of symbolic reasoning and knowledge representation, highlighting both the capabilities and limitations of logic-based systems compared to adaptive, data-driven models.

Knowledge Base

This knowledge base encodes familial relationships using first-order logic. The following formulas define basic facts, derived relationships, and recursive rules for complex family structures.

PROLOG CODE	First-Order Logic

`:- dynamic(parent_of/2).`
`:- dynamic(is_male/1).`
`:- dynamic(is_female/1).`
	• parent_of/2 is a dynamic relation between two entities (e.g., parent_of(alice, bob) means Alice is the parent of Bob).

	○ parent_of(x,y) ● is_male/1 and is_female/1 are dynamic unary predicates that denote gender. ○ is_male(x) ○ is_female(x)
`father_of(X, Y) :- parent_of(X, Y), is_male(X).` `mother_of(X, Y) :- parent_of(X, Y), is_female(X).`	∀x,y[father_of(x,y)↔parent_of(x,y)∧male(x)] ∀x,y[mother_of(x,y)↔parent_of(x,y)∧female(x)]
`child_of(Y, X) :- parent_of(X, Y).`	- Inverse of parent ∀x,y [child_of(y,x)↔parent_of(x,y)]

`sibling_of(X, Y) :-`
`  parent_of(Z, X),`
`  parent_of(Z, Y),`
`  X \= Y.`
	∀x,y [sibling_of(x,y)↔∃z (parent_of(z,x)∧parent_of(z,y)∧x=y)]

`brother_of(X, Y) :- sibling_of(X, Y), is_male(X).` `sister_of(X, Y) :- sibling_of(X, Y), is_female(X).`	∀x,y [brother_of(x,y)↔sibling_of(x,y)∧male(x)] ∀x,y [sister_of(x,y)↔sibling_of(x,y)∧female(x)]

`grandparent_of(X, Y) :-`
`  parent_of(X, Z),`
`  parent_of(Z, Y).`
	∀x,y [grandparent_of(x,y)↔∃z (parent_of(x,z)∧parent_of(z,y))]

`grandfather_of(X, Y) :- grandparent_of(X, Y), is_male(X).` `grandmother_of(X, Y) :- grandparent_of(X, Y), is_female(X).`	grandfather_of(x,y)↔grandparent_of(x,y)∧male(x) grandmother_of(x,y)↔grandparent_of(x,y)∧female(x)

`aunt_of(X, Y) :-`
`  sibling_of(X, Z),`
`  parent_of(Z, Y),`
`  is_female(X).`
	∀x,y [aunt_of(x,y)↔ ∃ z (sibling_of(x,z) ∧ parent_of(z,y) ∧ female(x))]

`uncle_of(X, Y) :-`
`  sibling_of(X, Z),`
`  parent_of(Z, Y),`
`  is_male(X).`
	∀x,y [uncle_of(x,y)↔ ∃ z (sibling_of(x,z) ∧ parent_of(z,y) ∧ male(x))]

Chatbot Implementation

The chatbot was implemented to understand and respond to natural language statements and questions about family relationships. It utilizes a hybrid architecture combining Python for the interface and pattern recognition and Prolog for knowledge representation and inference. The design allows the chatbot to learn new familial facts during a conversation and answer queries using logical rules.

The chatbot accepts user input in a simple natural language format, following a given set of sentence and question prompts**.**

Sentence Pattern	
[ ] and [ ] are siblings[1]	[ ] is a brother of [ ].
[ ] is a sister of [ ].	[ ] is the father of [ ].
[ ] is the mother of [ ].	[ ] and [ ] are the parents of [ ].
[ ] is a grandmother of [ ].	[ ] is a grandfather of [ ].
[ ] is a child of [ ].	[ ], [ ] and [ ] are children of [ ][2]
[ ] is a daughter of [ ].	[ ] is a son of [ ].
[ ] is an uncle of [ ].	[ ] is an aunt of [ ].

Figure 1: Sentence Pattern

Sentence Pattern	
Are [ ] and [ ] siblings?	Who are the siblings of [ ]?
Is [ ] a sister of [ ]?	Who are the sisters of [ ]?
Is [ ] a brother of [ ]?	Who are the brothers of [ ]?
Is [ ] the mother of [ ]?	Who is the mother of [ ]?
Is [ ] the father of [ ]?	Who is the father of [ ]?
Are [ ] and [ ] the parents of [ ]?	Who are the parents of [ ]?
Is [ ] a grandmother of [ ]?	Is [ ] a grandfather of [ ]?
Is [ ] a daughter of [ ]?	Who are the daughters of [ ]?
Is [ ] a son of [ ]?	Who are the sons of [ ]?
Is [ ] a child of [ ]?	Who are the children of [ ]?
Are [ ], [ ] and [ ] children of [ ]?	Is [ ] an aunt of [ ]?
Is [ ] an uncle of [ ]?	Are [ ] and [ ] relatives?

Figure 2: Question Pattern

Results

Welcome Message and Instructions

At the start of the program, there is a welcome message and instructions on how to use the program for the users. Examples of valid statements can also be found through the 'help' command.

```
Welcome to the Family Chatbot!
I can learn about family relationships and answer questions about them.
Type 'help' for examples, or 'exit' to quit.

> help

=== Family Chatbot Help ===
You can teach me about family relationships or ask questions!

Example statements (end with '.'):
- John is the father of Mary.
- Sarah is the mother of Tom.
- Mike is the son of Sarah.
```

Acquiring Facts:

For the chatbot to have knowledge of the family relations from the user, the user needs to input valid statements that ends with '.' (period). If the sentence is valid, the chatbot will say "Okay, I learned something new."

```
Type 'help' for examples, or 'exit' to quit.

> Janna is the daughter of Anna.
Okay, I learned something new.
> Anna is female.
Okay, I learned something new.
```

Asking and Answering Questions:

With the facts given, the chatbot can now answer questions regarding the family/relations. It can answer 'who' and confirmation questions. Note: The chatbot can only answer valid questions. If the question is not valid or there are no available facts about it, it will display a reply that it does not know the answer to what the user is asking.

```
> Who is the mother of Janna?
Anna.
> Is Anna the mother of Janna?
Yes.
```

```
> Is Anna the mother of Johanna?
No.
```

```
> Who are the daughters of Anna?
Janna.
```

Limitations and Challenges

In comparison to LLMs such as ChatGPT, the following are the limitations and challenges of the current implementation:

1. **Pattern Rigidness** - Due to the nature of the program's implementation, it does not take semantics into account. With the use of regular expressions, it requires the user to conform to a strict syntax. For instance, "Alice is Bob's mother" would fail to parse, whereas "Alice is the mother of Bob" would succeed despite being semantically similar.
2. **Lack of Support for Complexity** - Unlike LLMs which are able to understand complex family dynamics, the current implementation cannot deal with more intricate family trees. Facts must stick to predetermined categories which limits the types of families that the model can represent. This means that concepts such as adoption, blended families, or cultural variations in family structure are a challenge to represent.
3. **Limited Learning** - The model's learning is only limited to what the user inputs. It cannot retrieve additional concepts and information from the Internet, thus inhibiting the level of intelligent behavior it displays. Moreover, it cannot learn from contextual cues or adapt based on user behavior, unlike LLMs.
4. **Specialized Design** - The program is specifically designed to handle family relationships and family trees. It cannot be adapted to other domains without completely redesigning the logic. Moreover, it requires prior knowledge regarding the rules of the domain. This is unlike LLMs which exhibit broad knowledge across multiple domains.

Conclusion

This project served as a hands-on exploration of symbolic artificial intelligence through the creation of a family-based logic chatbot. By encoding relationship rules into a

PROLOG knowledge base and interfacing it with a Python frontend, the chatbot was able to respond to user inputs with logical accuracy and limited inference capabilities. While the system showed promise in understanding direct and transitive relationships, its performance was constrained by the rigidity of syntax patterns and the limited scope of its knowledge. Nonetheless, the project provided valuable insight into the strengths and weaknesses of logic-based systems compared to more adaptive models like LLMs. Through this endeavor, the team gained a deeper appreciation for knowledge representation, rule-based reasoning, and the intricacies of translating human relationships into machine-understandable logic.

References

[1] OpenAI. (2023). ChatGPT: Optimizing Language Models for Dialogue. https://openai.com/chatgpt

[2] Russell, S., & Norvig, P. (2021). Artificial Intelligence: A Modern Approach (4th ed.). Pearson.

Contributions

1. Arianne Yari Artus
    1.1. Code - Provided the base for the code and added additional logic to handle inferences and intelligent behavior
    1.2. Report - Wrote the limitations and challenges section

2. Chastine Cabatay
    2.1 Code - Implemented the input parser in Python and integrated the chatbot interface
    2.2 Report - wrote knowledge base

3. Gerylyn Guiller

   3.1 Code - Helped encode in knowledge base and assisted in testing the chatbot

   3.2 Report - Wrote the results section

4. Heather Lynn Soper

   4.1. Code - Focused on exception handling, chatbot response messaging

   4.2. Report - Wrote the introduction and conclusion section