**AIEA Logic+LLM**

(Sameera Sudarshan Kashyap and Pavithra Selvaraj)

# Introduction

The primary goal of this project was to build an intelligent symbolic reasoning system by integrating Large Language Models (LLMs) with logic-based frameworks like Prolog. The system aims to understand and solve natural language reasoning tasks by translating them into symbolic forms and using solvers for inference.

# Methodology

### Understanding LLM + Logic

To begin with, we read about OpenAI’s Agentic AI framework to understand how agent systems automate reasoning tasks. To test the capabilities, we conducted an experiment where we used the GPT-4.1 model to generate prolog facts from natural language input. The model gave a very detailed explanation which was then fed to the model with a second prompt asking it to convert the explanation to prolog code. We then used Pyswip to infer the prolog code in python. With this we were able to get a basic understanding of Large Language Models role in symbolic reasoning

### Logical Reasoning in Prolog

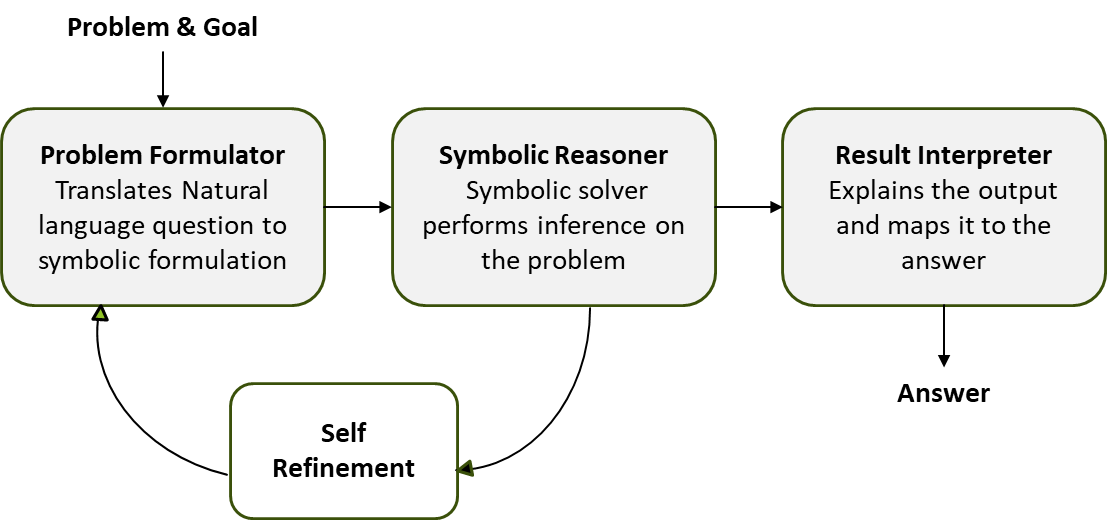
To take this work further, we explored some of the foundational concepts of Prolog using Simpsons KB and also referred to “The Art of Prolog” and SWI-Prolog’s official documentation to gain a better understanding of the concepts. Along with this we explored some other additional tools like PySwip and Pylog which would help us to seamlessly integrate it with Python.

After the environment setup, we experimented with Simpsons KB and following this we created a custom KB including facts and rules and then we ran various queries on this KB in both SWI-Prolog and Pylog. During testing we encountered minor discrepancies between the outputs in the two environments and we eventually figured out the issue that led to confusion after this we were able to confirm the consistency of results across both the environments. With this we were able to successfully demonstrate building and querying a logic based Prolog system through practical exercises which deepened our understanding of logical reasoning.

### Understanding Logic Reasoning and reimplementing Logic-LM framework

Building on our foundational knowledge of logical reasoning with Prolog, we extended our work by exploring and replicating the core framework from one of the research papers “Logic-LM: Empowering Large Language Models with Symbolic Solvers for Faithful Logical Reasoning”.

We thoroughly studied this research paper and created summary slides to capture the core concepts. The key idea of the proposed framework is to take natural language input questions, use an LLM to translate it into a symbolic format, apply symbolic solvers to infer the results and then convert the output back to human readable answers. And one of the standout features is its self refinement loop, which allows the system to iteratively improve the results from gaining feedback from the previous mistakes.



### Core framework of Logic-LM

To deepen our understanding, we cloned the Logic-LM codebase, installed all the required dependencies and executed the code locally which allowed us to understand the code flow as described in the paper.

We then re-implemented the code framework of the paper using our own modular architecture. Our system accepts a natural language question and first classifies the problem statement as either a logic problem or constraint satisfaction problem. Based on this classification, it activates the relevant solver module. Both solvers involve iterative prompting using GPT-4.o to refine and generate the symbolic version of the problem statement. These representations are then passed to the result interpreter module, which translates them into natural language response.

To ensure accuracy we implemented the self-refinement module. If the system is not able to interpret the result, the self refinement module re-evaluates the response by using insights from the previous iteration to improve the symbolic representation.

Through this framework, we successfully recreated the core mechanisms of the Logic-LM, using our own knowledge base and queries. This enhanced our understanding of combining Prolog logic with language models for intelligent problem-solving.

### 

### Probabilistic Soft Logic

In addition to our symbolic reasoning experiments, we explored Probabilistic Soft Logic - which blends logic based representations with probabilistic reasoning. Unlike Prolog, PSL allows soft truth values in the range of 0 to 1. PSL allows users to define weighted rules using logical and arithmetic operators. It also supports special operations such a Similarity function operators, weight based operators and prior negative allowing to model complex logical operations.

We followed a PSL tutorial that predicts voting behaviour based on social influences. We enforced that a person can only vote for one party at a time and added a negative prior to reflect that people do not vote unless influenced. We defined PSL models with rules and predicates and the model effectively captured real-world voting patterns.

### Logical Inference - Forward and Backward chaining

As part of our exploration of logic-based reasoning and symbolic inference, we also studied and implemented the core concepts of Forward and Backward chaining which are the foundational concepts in rule based systems and logic programming.

We began by reading some of the introductory materials from GFG and MIT which helped us understand how inference works. To gain practical experience, we applied our understanding by working on class assignments for CSE240 which progressively explored the application of chaining techniques.

Initial problems were straightforward and helped us understand the forward chaining working and the relationship chains. The last problem was a bit challenging because it not just required rule writing but also understanding the construction of goal tree which is a structure used in backward chaining to trace the paths. To solve this problem, we carefully reviewed the explanations and structure of the functions and the datasets, after which we were able to complete the problem successfully. Overall, this task helped gain more insights into how the logical reasoning can be structured from facts and to goals.

### 

### Integration of LangChain

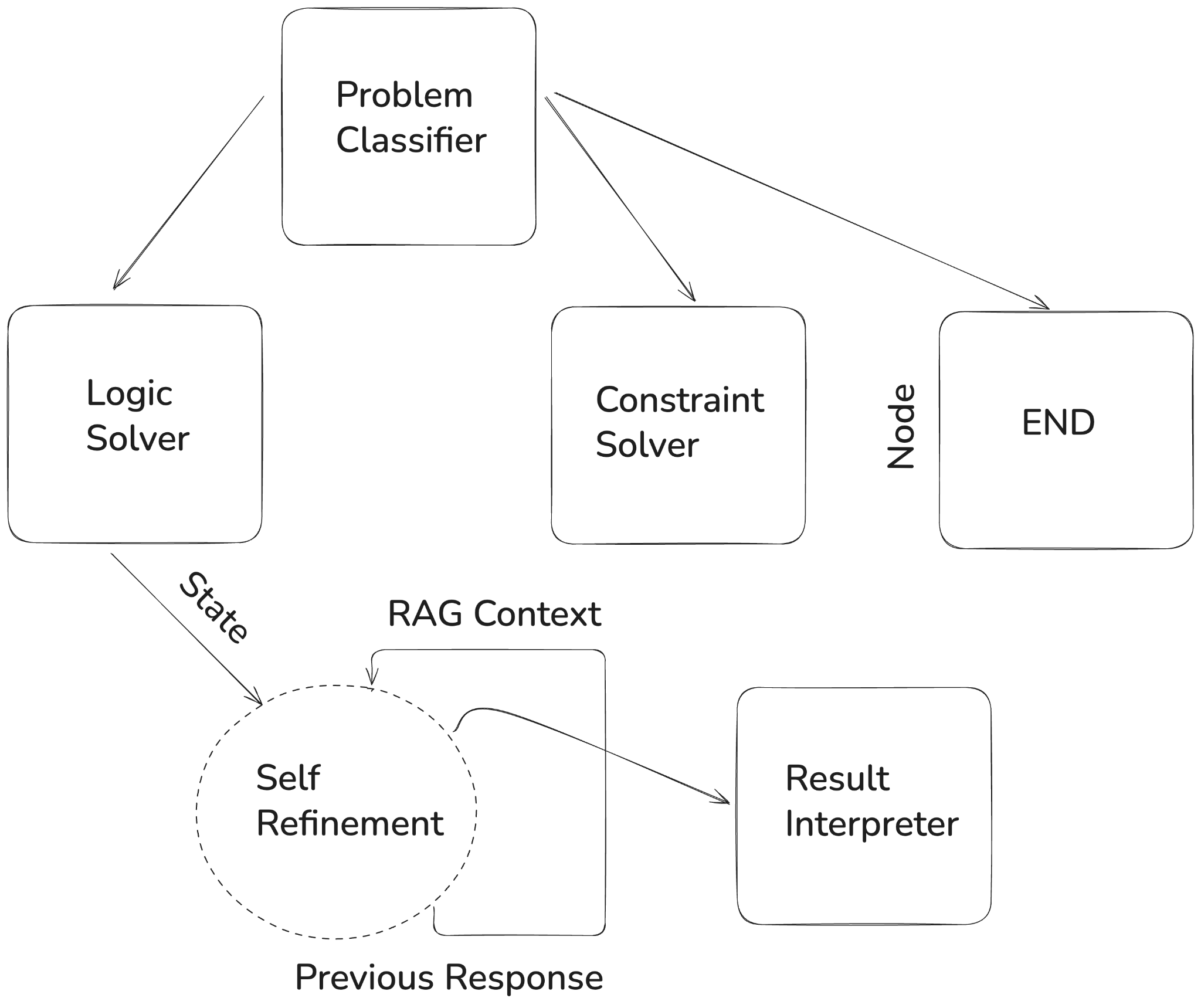
To improve our symbolic reasoning system, we integrated LangChain, which provides a modular abstraction over language models. We began by setting up the development environment with the required packages. We then migrated the existing model to Langchain. We reused the prolog KB and implemented a RAG pipeline using Langchain and Chromadb to test the inference with additional context. To evaluate the effectiveness of LangChain along with RAG, we designed a set of problem statements. The system was tasked with finding the truth values of various natural language statements based on the available KB.

We observed that when the relevant facts were already present in the context, the system was able to accurately identify the correct answers. For queries that required information which was not included in the KB, the system was unable to make inferences. However, once the missing facts were added and the embeddings were regenerated, the system successfully produced accurate results. These results demonstrated that combining LangChain with vector-based retrieval enhances the reasoning ability of the model.

### Migration to LangGraph

As the next stage, we migrated our implementation to LangGraph, a powerful framework built on LangChain that enables the orchestration of production-ready agent workflows. To begin the migration, we first analyzed our existing system and identified main components:

Classifier – Determines whether the user’s natural language input is a logic or constraint-based problem. Solvers – Includes a logic\_solver and a constraint\_solver, both of which generate symbolic representations via LLM prompting, using context from a retrieval-augmented generation (RAG) database. Refinement Module – If the initial result is unclear, this module triggers iterative LLM prompting to improve the outcome until a confident answer is reached. Result Interpreter – Converts the final symbolic output into a human-readable, natural language response. Using LangGraph, we mapped these components into a structural flow so that the system knows exactly what to do at each point based on what it has learned or decided.



### LangGraph integration into Logic LM

As part of the evaluation process of our system, we tested it across various types of natural language questions. These questions varied in how directly or indirectly the required information was presented and the system was able to correctly interpret the result. Here's an overview of some of the question types.

Questions missing background information: This question does not contain all the information needed to answer the question directly. However, the system can access its memory to complete the reasoning.

Questions with indirect information: This question contains all the necessary information, but the answer is not obvious. The system must interpret relationships and sequence of the given statements.

Questions with unclear statements: These are the most challenging questions. The facts are present, but they may be unclear or require comparison and elimination to reach conclusions. The system needs to carefully analyze the relationship between them to choose the right answer. By handling these different types of questions, our system demonstrates the ability to combine natural language with logical reasoning even in complex situations.

In conclusion, our project successfully demonstrates how Large Language Models can be integrated with symbolic reasoning frameworks to solve complex natural language problems. By combining symbolic inference and iterative self-refinement, we developed a system capable of translating natural language into symbolic representations and reasoning over them effectively. The addition of LangChain and LangGraph further enhanced the system’s modularity and scalability, allowing structured workflows and improved performance through retrieval-augmented generation. Our experiments confirmed that the system performs well when provided with sufficient context and is robust enough to handle direct, indirect, and ambiguous queries. Overall, this work highlights the value of hybrid AI systems that integrate symbolic logic and Large Language Models to achieve reliable and interpretable reasoning.

# Results

### Probabilistic Soft Logic

Alice Republican 0.0

Alice Green 0.0

Bob Democrat 0.94710255

Bob Republican 0.027526164

Bob Green 0.028656185

Charlie Democrat 0.8932856

Charlie Republican 0.05020552

Charlie Green 0.056753688

Dave Democrat 0.6370932

Dave Republican 0.19213226

Dave Green 0.17359492

Eve Democrat 0.6085047

Eve Republican 0.208066

Eve Green 0.18499015

Frank Democrat 0.0

Frank Green 0.0

Grace Democrat 0.0

Grace Republican 2.8908253E-5



Evaluation results:

Evaluator: DiscreteEvaluator,

Predicate: VOTESFOR,

Results -- Accuracy: 0.722222, F1: 0.285714,

Positive Class Precision: 0.250000,

Positive Class Recall: 0.333333,

Negative Class Precision: 0.857143,

Negative Class Recall: 0.800000

### Logic LLM with Langchain and Langgraph

**Input**

question'''Monica, Rachel and Phoebe are Female

Pheobe is friends with both Rachel and Monica

Ross is Monica\'s brother

Provide the correct answer to the below question

Whose sister is Amy ?

'''



**Output**

Chain of Thought Reasoning

REASONING:

1. We need to determine whose sister Amy is, which means we will look for a sibling relationship involving Amy.

2. From the facts, we have a sibling relationship: sibling(amy, rachel).

3. This indicates that Amy is a sister to Rachel.

4. Therefore, when we query sibling(amy, X), it matches with X being Rachel, since Amy is Rachel's sister based on our fact.

5. Hence, the answer to the query "?- sibling(amy, X)." would resolve to X = rachel.

Determined Answer:

Refined result [{'X': 'rachel'}]

Query: sibling(amy, X).

[{'X': 'rachel'}]



1. Complex query with multiple outputs where some facts are from kb and others from context.

**Input**

question:

'''Ross and Joey are best friends

Pheobe is friends with both Rachel and Monica

Ross is Monica\'s brother

Describe all bestfriends of Joey

'''



Chain of Thought Reasoning

REASONING:

1. To determine bestfriends, we need mutual friendship between persons.

2. According to the facts, joey is friends with chandler and ross.

3. Check if chandler is also friends with joey; this is confirmed in the facts.

4. Therefore, joey and chandler are best friends.

5. Check if ross is also friends with joey; the friendship is mutual according to the facts.

6. Therefore, joey and ross are best friends.

7. No other facts suggest a mutual friendship involving joey.

8. Hence, the query returns both chandler and ross as best friends of joey.

Query: bestfriends(joey, BestFriend).

[{'BestFriend': 'chandler'}, {'BestFriend': 'ross'}]

# 

**Division of work:**

| **Tasks** | **Auditor** |
| --- | --- |
| Task 1 - AIEA Auditor Onboarding | Sameer Kashyap and Pavithra Selvaraj |
| Task 2 - LLM + Logic Onboarding | Sameer Kashyap and Pavithra Selvaraj |
| Task 3 - Nautilus | Sameer Kashyap |
| Task 4 - Logical Reasoning in Prolog | Pavithra Selvaraj |
| Task 5 - Understanding Logical Reasoning | Pavithra Selvaraj |
| Task 6 - Researching other Logics + LLMs | Sameer Kashyap |
| Task 7 - Logical Inference | Pavithra Selvaraj |
| Task 8 - LangChain | Sameer Kashyap and Pavithra Selvaraj |
| Task 9 - LangGraph | Sameer Kashyap and Pavithra Selvaraj |
| Task 10 - AIEA Auditor Offboarding | Sameer Kashyap and Pavithra Selvaraj |

| Sameer Kashyap | [GitHub](https://github.com/Sameerkash/aiea-auditor/tree/main) |
| --- | --- |
| Pavithra Selvaraj | [GitHub](https://github.com/PavithraKandhasamySelvaraj/aiea_llm_logic) |