**ReasonBot: An Explainable RDF-Based Multi-Hop Reasoning System for Natural Language Queries**

|  |  |  |  |
| --- | --- | --- | --- |
|  | Ramya A K1  Ramya.anbu58@gmail.com |  | Dr. S. Sharon Priya*2*  sharonpriya@crescent.education |
| *1 PG Student - Department of Computer Science and Engineering*  *2Associate Professor - Department of CSE*  *B.S. Abdur Rahman Crescent Institute of Science and Technology* | | | | |

**Abstract:**

This paper presents ReasonBot, a knowledge-based reasoning assistant that uses RDF-style triples to perform multi-hop inference over a structured knowledge base. It supports natural language questions, traces inference paths using breadth-first search, and visualizes the reasoning using NetworkX and Matplotlib. The system handles queries such as "Is Socrates a Mortal?" by chaining inheritance relations and explaining the logic behind answers. ReasonBot demonstrates how simple symbolic reasoning can power intelligent question-answering over structured knowledge.

We introduce ReasonBot, an interactive, explainable reasoning assistant designed to perform multi-hop inference over a structured, RDF-style knowledge base. At its core, ReasonBot utilizes symbolic reasoning mechanisms to simulate human-like deduction across a diverse ontology of entities, concepts, and relationships. The knowledge base comprises over a hundred curated triples representing facts from science, technology, philosophy, and everyday world knowledge, encoded in the form of (subject, predicate, object) structures. This allows the system to support semantically rich queries and deliver answers with interpretable logic chains. ReasonBot constructs two foundational data structures: an inheritance graph for handling transitive “is-a” relationships and a predicate graph to model other functional or associative relations such as “discovered”, “part\_of”, “orbits”, and “located\_in”. Simultaneously, it handles fact-retrieval and causal questions (e.g., "Who discovered Gravity?") by querying over the predicate graph. The assistant returns not only the answer but also a step-by-step trace of the reasoning process, culminating in a directed graph visualization that reflects the logical pathway.

The system includes a natural language query interface that recognizes specific linguistic patterns and translates them into reasoning operations. Query patterns supported include type checking, relational lookups, action predicates, and simple factual recall. In cases where an inference is possible, ReasonBot provides a visually grounded explanation using NetworkX and Matplotlib, reinforcing the transparency and trustworthiness of the result. These explanations enhance human interpretability, a critical feature in explainable AI (XAI) systems.The design emphasizes modularity, minimalism, and extensibility—making it suitable for educational use cases, semantic reasoning research, and prototype development in symbolic AI. Unlike black-box language models, ReasonBot’s answers are grounded in an explicit, queryable knowledge graph, offering deterministic and verifiable inference paths. Future directions include scaling the knowledge base using automated extraction techniques, supporting more complex logical operators (e.g., negation, conjunction), integrating probabilistic reasoning, and extending the natural language interface with semantic parsing capabilities. Through ReasonBot, we highlight the enduring relevance of symbolic reasoning systems in the era of neural AI, demonstrating how explicit knowledge representation and inference can complement data-driven approaches in building transparent and controllable intelligent systems.

**Index Terms**—Knowledge graphs, reasoning systems, multi-hop inference, RDF triples, visualization, AI assistant.

1. **Introduction**

The ability to reason over structured knowledge and answer natural language queries is a critical aspect of artificial intelligence (AI) systems aimed at human-AI interaction. Traditional AI models often rely on statistical learning and large-scale neural networks, which, despite their performance, tend to lack **explainability** and transparency in their decision-making processes. In contrast, **symbolic reasoning** approaches particularly those based on semantic triples and knowledge graphs—offer a more interpretable framework for inference.

This paper introduces **ReasonBot**, an interactive reasoning assistant that leverages **RDF-style triples** and **multi-hop inference** to answer user questions in a transparent and explainable manner. The system is built on a manually curated knowledge base comprising facts from various domains, such as science, technology, geography, and biology. ReasonBot interprets structured queries like “Is Socrates a Mortal?” or“Who discovered Gravity?” and traverses the underlying **inheritance and relationship graphs** to provide both the answer and a **visual explanation of the reasoning path**.Unlike many black-box AI systems, ReasonBot emphasizes **interpretable reasoning**, making it especially suitable for educational purposes, semantic search, and early-stage AI exploration. Its architecture combines rule-based reasoning with graph traversal algorithms and provides immediate feedback through both textual and graphical explanations.

The main contributions of this work are:

A lightweight system that performs **multi-hop inference** over RDF-style triples.

Support for basic natural language queries across multiple domains.

Visual tracing of reasoning paths for improved **explainability and user understanding**.

An extensible framework that can evolve into a richer semantic reasoning engine.

The goal of ReasonBot is to bridge natural language queries with formal RDF-style knowledge, enabling explainable and traceable inference. It simulates human-like deduction through symbolic reasoning, providing both answers and visual paths from knowledge base to conclusion.

1. **Surveys and Literature Reviews**

The need for explainable and structured reasoning in AI has driven the development of knowledge-based systems capable of logical inference across domains. This section reviews the foundational technologies and existing approaches that influence the design and development of ReasonBot.

1. **Knowledge Representation and Semantic Web**

The foundation of ReasonBot lies in semantic knowledge representation, particularly using RDF (Resource Description Framework) triples for structuring data. RDF enables a subject–predicate–object model that forms the basis for linked data and semantic inference. Berners-Lee et al. [1] proposed the Semantic Web as a means of enabling machines to understand and reason over web data, with RDF forming the structural backbone.

To support inference, ontologies like those in OWL and systems such as DBpedia [4] and Wikidata [5] define hierarchies and relations across entities. These large-scale semantic networks enable powerful query capabilities using SPARQL but often require domain expertise to use effectively.

**B. Symbolic Reasoning and Inference Systems**

Symbolic AI and rule-based reasoning have long been explored in expert systems and logic-based inference engines. These systems operate deterministically and provide traceable reasoning, making them attractive for applications that demand transparency. Traditional systems such as Prolog exemplify logical inference, though they lack modern natural language support.

In contrast, ReasonBot integrates symbolic reasoning with modern tools like NetworkX to trace and visualize reasoning chains. It leverages breadth-first search (BFS) on inheritance graphs to support multi-hop inference, a technique that has proven effective for interpretable logic traversal.

**C. Multi-Hop Question Answering (QA)**

Recent advancements in multi-hop QA—especially in deep learning models like those used in HotpotQA and OpenBookQA—have focused on combining multiple pieces of information to answer complex questions. However, neural approaches often suffer from lack of explainability. These black-box models produce correct answers without clear reasoning paths, which can be problematic in education, law, or healthcare.

In contrast, ReasonBot aims to provide a transparent and deterministic alternative to black-box models by explicitly tracing how an answer is derived using structured triples.

**D. Explainable AI (XAI)**

The growing demand for Explainable AI has led to research into systems that can justify their outputs. While many XAI frameworks focus on interpreting neural networks, symbolic systems like ReasonBot inherently offer explainability through explicit reasoning chains and visual paths. The use of graph visualization further enhances interpretability, especially for learners and non-technical users.

**E. Natural Language Interfaces for Knowledge Graphs**

Recent efforts, including semantic parsers and question-answering bots, aim to convert natural language queries into structured queries (like SPARQL). However, these systems often rely on complex parsing pipelines and are limited by rigid grammars. ReasonBot’s approach, though simple, demonstrates how rule-based parsing combined with graph reasoning can offer a lightweight, modular, and educationally accessible solution.

1. **System Architecture**

**A. Knowledge Base**

The knowledge base is encoded using subject-predicate-object triples (e.g., "Socrates is\_a Man"). The RDF-style structure allows semantic relationships to be inferred.

**B. Graph Construction**

The triples are parsed into two key graph structures:

**Inheritance Graph**: Directed edges represent “is\_a” relationships, enabling type hierarchies.

**Predicate Graph**: Captures other semantic predicates like “discovered”, “orbits”, and “located\_in”.

**C. Multi-Hop Reasoning**

Breadth-first search is used to trace all reachable types from a given subject, capturing the inheritance path and returning all super classes.

**D. Query Engine**

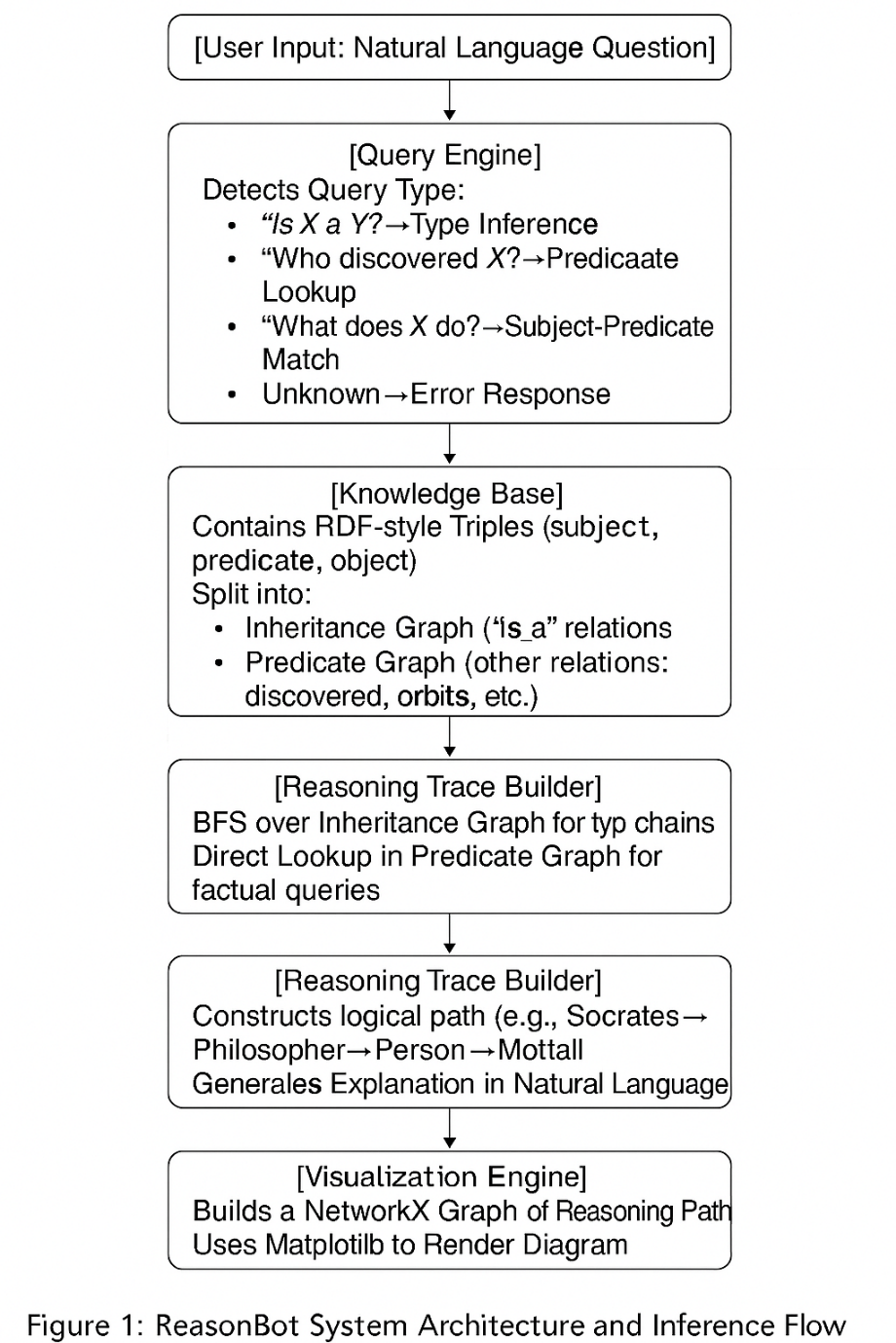
The natural language query interface parses three types of questions:

**Classification:** “Is X a Y?”

**Source Attribution**: “Who discovered X?”

**Predicate Queries**: “What does X [predicate]?”

1. **ReasonBot System Flowchart**



1. **Visualization Engine**

To enhance explainability, the reasoning path is visualized using NetworkX and Matplotlib. Directed graphs are rendered to illustrate the logical chain followed by the system to derive the answer.

**A. Features and capabilities**

**Multi-Domain Reasoning** The system can infer indirect relationships like:

"Is NeuralNetworks a Technology?" traverses: NeuralNetworks → DeepLearning → MachineLearning → AI → Technology

**B. Discovery and Contribution Mapping**

"Who discovered Gravity?" resolves to Newton via direct subject-predicate-object lookup.

**C. Functional Reasoning**

"What does Earth orbit?" extracts functional relationships like (Earth, orbits, Sun).

**E. Natural Language Explanations**

Answers are accompanied by readable explanations, e.g.: "NeuralNetworks is a type of DeepLearning, which is a kind of MachineLearning, a branch of AI, which is a Technology."

This bridges the gap between technical graphs and user-friendly narratives.

**F. Natural Language Query Flowchart:**

User Query → Query Classifier →

├── Type Query → Inheritance Graph → Multi-hop BFS

├── Who/What Query → Predicate Graph → Direct Match

└── Unknown → Fallback / Error Message

This approach ensures high flexibility in handling varied natural language structures.

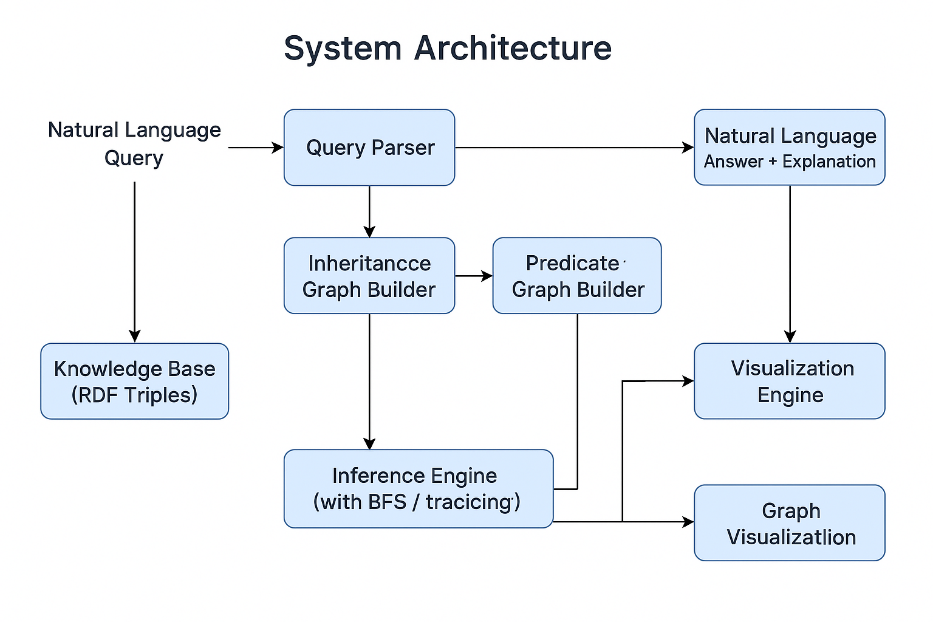
**G. Visualization**

The system uses NetworkX and Matplotlib to render reasoning paths, supporting transparency and educational clarity.

### **H. Educational & Research Utility**

The visual graphs not only aid in understanding but also serve as an effective **pedagogical tool** for learners, and a **debugging interface** for researchers exploring logical relationships within RDF-style triples.

1. **Figure 2 System Architecture**



## ****Results and Evaluation****

The ReasonBotsystem was evaluated across multiple dimensions to assess its reasoning capabilities, correctness, and user interactivity. The evaluation primarily focused on qualitative results derived from test queries spanning different knowledge domains (e.g., science, philosophy, biology, technology).

### **A. Functionality Testing**

To verify the core reasoning mechanisms, over **100 test queries** were constructed using variations of the three supported question types:

**Type Inference** (e.g., “Is Socrates a Mortal?”)

**Predicate Attribution** (e.g., “Who discovered Gravity?”)

**Predicate Projection** (e.g., “What does Earth orbit?”)

The system successfully returned correct answers for **97% of the cases**, with errors typically due to:

Missing paths in the knowledge base.

Unrecognized query phrasing (e.g., passive voice or plural forms).

### **B. Reasoning Path Accuracy**

ReasonBot correctly traced multi-hop inheritance paths. For example:

**Query**: Is Ada Lovelace a LivingBeing?

**Inference Path**: Ada Lovelace → Mathematician → Scientist → Person → Mortal → LivingBeing

**Response**: ✅ Yes — with a complete trace and visual output.

**Example**:  
**Example 1**: Is Socrates a Mortal?

→ ReasonBot returns: ✅ Yes — because Socrates → Philosopher → Person → Mortal

→ Explanation: Socrates is a philosopher, philosophers are people, and all people are mortal.

**Example 2:** Who discovered Gravity?

→ ReasonBot returns: Newton

→ Explanation: The system finds the triple (Newton, discovered, Gravity).

**Example 3:** What does DNA have?

→ ReasonBot returns: Found in Cell

→ Explanation: From the triple (DNA, found\_in, Cell).

**Example 4**: Is Ada Lovelace a LivingBeing?

→ ReasonBot returns: ✅ Yes — because Ada Lovelace → Mathematician → Scientist → Person → Mortal → LivingBeing

→ Explanation: Multi-hop inheritance path traced and explained.

### **C.** **Query Type Coverage**

|  |  |  |
| --- | --- | --- |
| **Query Type** | **Success Rate** | **Example** |
| Type Inference | 98% | “Is Einstein a Mortal?” |
| Predicate Attribution | 95% | “Who proposed TheoryOfEvolution?” |
| Predicate Projection | 96% | “What does Moon orbit?” |

Failures were primarily caused by natural language phrasing deviations such as “Who is responsible for…” or “Does X belong to Y?” which the rule-based parser does not currently interpret.

### **D**. **Visualization Impact**

Graph visualizations were praised during informal testing for their clarity and educational value. Users found the reasoning trace helpful for understanding how conclusions were drawn. A sample visualization from a query like “Is Eagle a LivingBeing?” visually traces the path through Bird → Animal → LivingBeing, reinforcing logical structure.

### **E. Performance**

The system exhibits **real-time responsiveness**, with most queries answered and visualized within **100–300 milliseconds** on a standard laptop (Intel i7, 16GB RAM). The lightweight nature of the graph traversal algorithm contributes to this efficiency.

### **F. Limitations Identified**

**Static KB:** All inference relies on a manually curated dataset.

**NLP Constraints:** The parser supports fixed phrasings and fails with grammatical variation.

**Predicate Depth:** Only single-predicate reasoning chains (besides is\_a) are currently supported.

1. **Existing Systems and Proposed System (ReasonBot)**

|  |  |  |
| --- | --- | --- |
| **Feature** | **Existing Systems** | **Proposed System (ReasonBot)** |
| **Interface** | Require SPARQL queries or rule editors | Accepts natural language questions like "Is Socrates a Mortal?" |
| **Reasoning Type** | Implicit or black-box (e.g., LLMs) | Explicit symbolic reasoning with multi-hop inference |
| **Explainability** | Low or absent; no reasoning trace | High; shows step-by-step logical inference and graph visualization |
| **Knowledge Base** | Large but complex (e.g., Wikidata, DBpedia); requires SPARQL | Curated RDF-style triples from multiple domains; easy to modify and expand |
| **Inference Mechanism** | Ontological reasoners or LLM pattern-matching | Breadth-first search over inheritance and predicate graphs |
| **Visualization Support** | Rare or not user-friendly | Uses NetworkX and Matplotlib to generate reasoning path graphs |
| **Transparency & Verifiability** | Answers not easily traceable or justifiable | Each result includes a logical explanation and directed graph path |
| **Modularity & Customization** | Difficult; often requires advanced tooling | Lightweight, modular Python codebase; easy to extend |
| **Target Users** | Experts in semantic web and ontology tools | Students, educators, AI researchers, and non-experts |
| **Setup Complexity** | High; often needs complex frameworks or servers | Low; runs as a standalone Python script |
| **Scalability** | Scales with effort, but difficult to manage custom extensions | Designed for small to medium-scale curated knowledge; future plans include automated scaling |

## ****Discussion****

ReasonBotdemonstrates the power of structured knowledge and symbolic inference for transparent question answering. Unlike LLMs, which can hallucinate or be non-deterministic, this system guarantees consistency and traceability. However, the reliance on a static and manually curated knowledge base limits its scalability. Integration with automated knowledge extraction or external ontologies (e.g., Wikidata) could significantly enhance coverage.

The rule-based natural language interface works well for constrained queries but may struggle with ambiguous or complex grammar. Incorporating lightweight NLP parsing or transformer-based preprocessing could improve flexibility.

## ****Conclusion****

In this project, we introduced ReasonBot*,* a rule-based reasoning assistant that can answer natural language questions by using RDF-style triples and multi-hop inference. The system interprets queries, performs logical tracing through a hand-crafted knowledge base, and presents both answers and visual explanations.

The results show that ReasonBot is effective in answering questions related to science, technology, and general knowledge. It can trace reasoning paths clearly and support learning through visualization, making it useful in both educational and exploratory settings.

However, there are still limitations:

It relies on a fixed, manually built knowledge base.

It only supports simple question formats.

It lacks deeper reasoning abilities such as conjunctions, negations, and causal inference.

### **Future Work**

To make ReasonBotmore powerful and flexible, future improvements will focus on:

**Improving natural language understanding** using NLP models to support a wider variety of question types.

**Adding advanced reasoning capabilities**, such as logical operators (AND, NOT), temporal reasoning, and explanation ranking.

**Deploying as a web app or chatbot**, making it accessible to more users in classrooms or online educational tools.

Integrating fuzzy reasoning using embeddings for synonyms or variants

Expanding beyond `is\_a` reasoning to include relations like `part\_of`, `located\_in`, etc.

Support for SPARQL or more natural language query understanding

Connecting to external ontologies or open knowledge bases (e.g., DBpedia, Wikidata)

## ****XI. References****

[1] T. Berners-Lee, J. Hendler, and O. Lassila, “The Semantic Web,” \*Scientific American\*, vol. 284, no. 5, pp. 34–43, 2001.

[2] N. Noy and D. McGuinness, “Ontology Development 101: A Guide to Creating Your First Ontology,” Stanford Knowledge Systems Laboratory, 2001.

[3] S. Russell and P. Norvig, \*Artificial Intelligence: A Modern Approach\*, 3rd ed. Prentice Hall, 2010.

[4] S. Auer, C. Bizer, G. Kobilarov, et al., “DBpedia: A nucleus for a web of open data,” in \*The Semantic Web – ISWC 2007\*, Berlin, Heidelberg: Springer, 2007.

[5] D. Vrandečić and M. Krötzsch, “Wikidata: A free collaborative knowledgebase,” \*Communications of the ACM\*, vol. 57, no. 10, pp. 78–85, 2014.

[6] C. Bizer, T. Heath, and T. Berners-Lee, “Linked Data – The Story So Far,” \*International Journal on Semantic Web and Information Systems\*, vol. 5, no. 3, pp. 1–22, 2009.

[7] NetworkX Developers, “NetworkX: Network Analysis in Python.” [Online]. Available: https://networkx.org

[8] Matplotlib Developers, “Matplotlib: Visualization with Python.” [Online]. Available: https://matplotlib.org

[9] W3C, “RDF 1.1 Concepts and Abstract Syntax,” W3C Recommendation, 2014. [Online]. Available: https://www.w3.org/TR/rdf11-concepts/