KRR and First Order Logic

1 Introduction

One of the key aspects of human intelligence we agree upon is that we *know* things. But what does it mean to *know* something at all? Consider the statement "If it is daytime and there are no clouds in the sky, then the sky is blue". As humans, at some point we learn that this statement is generally true. Somehow, through various neural mechanisms and pathways, the brain stores this knowledge and enables us to reason about it.

For instance, if you were asked "what can you infer if the sky is not blue?", you would likely respond that either it is not daytime or it is cloudy (or both). This reasoning capability suggests that humans possess some form of internal logical system. This answer might emerge through one of two processes: (1) we draw upon all of our prior experiences observing the sky—every time we looked up and did not see a blue sky, it was either cloudy or nighttime—or (2) we perform some form of logical reasoning over the conditional statement about sky color.

In this unit, we explore the problem of Knowledge Representation and Reasoning (KRR). KRR investigates how an intelligent agent can represent knowledge **and** reason effectively about that knowledge. Knowledge-based agents consist of two main components: a **Knowledge Base** (KB) that stores facts and rules about the world, and an **Inference Engine** that performs logical reasoning over the knowledge base to derive new conclusions.

In this lecture, we present logical agents that use first-order logic to represent **facts** about the world and then answer queries about the knowledge base through logical reasoning. For instance, returning to our sky example, our knowledge base might contain the following logical statement:

$$\neg\text{cloudy} \wedge \text{daytime} \Rightarrow \text{blue\_sky}$$

Our inference engine could then be queried with the negation

$$\neg\text{blue\_sky}$$

and asked what else we can conclude. Through logical inference (specifically, modus tollens), the engine would return

$$\text{cloudy} \vee \neg\text{daytime}$$

—meaning that if the sky is not blue, then it must be either cloudy or not daytime.

1.1 Why Do We Need Knowledge Representation?

Why do we need specialized research on knowledge representation? After all, we have represented knowledge previously in this course using various data structures. For example, a tic-tac-toe board can be represented with a 2D array containing X's, O's, or empty cells. Simple data structures are often sufficient for storing straightforward facts about the world (e.g., board state, current player, game score), but human knowledge encompasses far more complex relationships and abstract concepts than these simple facts.

Human knowledge includes:

- Complex relationships between entities ("All birds can fly, except penguins")
- Temporal reasoning ("If it rained yesterday and it's sunny today, the ground will be wet")

[1] These notes were compiled in conjunction with Professor Greenwald.

- Uncertainty and probability ("It will likely rain tomorrow")
- Abstract concepts ("Justice requires fairness")
- General rules with exceptions ("Normally, students attend class, unless they are sick")

These types of knowledge require more sophisticated representation schemes that can capture logical relationships, handle uncertainty, and support complex reasoning patterns.

1.2 KRR in the Era of Large Language Models

Do we still need KRR in the world of Large Language Models (LLMs)? LLMs learn to generate text by predicting the most likely next token based on patterns in vast amounts of text scraped from the internet and other written sources. Modern LLMs appear to have memorized many facts about our world—a capable LLM can certainly tell you that George Washington was the first U.S. president or that $2 + 2 = 4$ without performing explicit reasoning or consulting external sources.

However, LLMs face significant limitations in more complex settings. When tasked with reading and understanding a new codebase, for instance, LLMs can struggle with hallucinating incorrect facts, even when provided with search capabilities through the codebase. At its core, hallucination often stems from suboptimal representation and retrieval of existing knowledge.

This has led to renewed interest in connecting LLMs to structured knowledge bases. Techniques like Retrieval Augmented Generation (RAG), which combine the language generation capabilities of LLMs with the structured reasoning of knowledge bases, represent one of the most active areas of current LLM research. Rather than replacing KRR, LLMs have highlighted the continued importance of principled knowledge representation and reasoning systems.

1.3 Historical Context: Expert Systems

KRR played a pivotal role in AI's history as one of the field's first successful applications to real-world industry problems. Expert Systems were designed to mimic the performance of human domain experts by encoding their knowledge and expertise into computer programs. These systems translated human expertise into formal rules and logical relationships that computers could process and reason about.

Medical diagnosis systems exemplify this approach—expert systems could encode the knowledge of experienced doctors about symptoms, diseases, and treatments, then use logical reasoning to suggest diagnoses for new patients. The success was remarkable: by the 1980s, the majority of Fortune 500 companies were using expert systems in production environments.

This success spawned an entire industry. Lisp became the preferred programming language for building expert systems due to its symbolic processing capabilities and flexible syntax for representing logical statements. Specialized hardware called "Lisp machines" were developed specifically to run these expert systems efficiently. However, this specialized hardware industry was eventually displaced by the rapid development of cheaper, more powerful personal computers that could run the same Lisp programs without requiring dedicated hardware.

The expert systems boom demonstrated both the potential and limitations of symbolic AI approaches, lessons that continue to inform modern AI development.

2 First Order Logic

One powerful way of representing knowledge is through logical predicates and relations using First Order Logic (FOL). FOL extends propositional logic by introducing objects, predicates, functions, and quantifiers, allowing us to express much richer and more nuanced knowledge about the world.

2.1 Components of First Order Logic

FOL consists of several key components:

- **Objects**: Individual entities in the world (people, places, concepts, things)
- **Predicates**: Properties of objects or relationships between objects
- **Functions**: Mappings from objects to other objects
- **Quantifiers**: Universal ($\forall$) and existential ($\exists$) quantification over objects
- **Logical connectives**: AND ($\wedge$), OR ($\vee$), NOT ($\neg$), IMPLIES ($\Rightarrow$), etc.

2.2 Predicates: The Heart of FOL

Predicates are the primary way we express knowledge in FOL. A predicate represents either:

1. **Properties of objects**: Unary predicates that describe characteristics
2. **Relationships between objects**: Binary, ternary, or n-ary predicates that describe connections

2.2.1 Unary Predicates (Properties)

Unary predicates express properties that objects can possess:

- Student(x) - "x is a student"
- Tall(x) - "x is tall"
- Red(x) - "x is red"
- Cloudy() - "It is cloudy" (zero-ary predicate, like propositional logic)

2.2.2 Binary Predicates (Relationships)

Binary predicates express relationships between two objects:

- Loves(x, y) - "x loves y"
- ParentOf(x, y) - "x is a parent of y"
- GreaterThan(x, y) - "x is greater than y"
- Enrolled(x, y) - "x is enrolled in y"

2.2.3 Higher-Arity Predicates

We can have predicates with three or more arguments:

- Between(x, y, z) - "x is between y and z"
- Teaches(x, y, z) - "x teaches course y in semester z"
- Distance(x, y, z) - "The distance from x to y is z"

2.3 Syntax of First Order Logic

The syntax of FOL defines how we can combine these components into well-formed formulas:

2.3.1 Terms

A **term** represents an object and can be:

- **Constant**: A specific object (e.g., John, Brown, CSCI0410)
- **Variable**: A placeholder for objects (e.g., x, y, z)
- **Function**: Maps objects to objects (e.g., MotherOf(x), Plus$(2, 3)$)

2.3.2 Atomic Formulas

An **atomic formula** is formed by applying a predicate to the appropriate number of terms:

$$P(t_1, t_2, \ldots, t_n)$$

where P is an n-ary predicate and $t_1, \ldots, t_n$ are terms.

Examples:

- Student(Alice) - "Alice is a student"
- Enrolled$(x, \text{CSCI0410})$ - "x is enrolled in CSCI0410"
- Loves(Romeo, Juliet) - "Romeo loves Juliet"

2.3.3 Complex Formulas

Complex formulas are built from atomic formulas using logical connectives and quantifiers:

Logical Connectives:

- $\neg P$ - negation
- $P \wedge Q$ - conjunction (AND)
- $P \vee Q$ - disjunction (OR)
- $P \Rightarrow Q$ - implication
- $P \Leftrightarrow Q$ - biconditional (if and only if)

Quantifiers:

- $\forall x\, P(x)$ - "For all x, $P(x)$ is true"
- $\exists x\, P(x)$ - "There exists an x such that $P(x)$ is true"

2.4 Example: Representing Knowledge with Predicates

Let's return to our sky example and see how predicates allow us to represent knowledge more expressively:

Instead of just:

$$\neg\text{cloudy} \wedge \text{daytime} \Rightarrow \text{blue\_sky}$$

We can use predicates to be more specific:

$$\forall t\, (\text{Daytime}(t) \wedge \neg\text{Cloudy}(t) \Rightarrow \text{Blue}(\text{Sky}, t))$$

This reads: "For all times t, if it is daytime at t and not cloudy at t, then the sky is blue at time t."

We can represent even richer knowledge:

$$\forall x\,(\text{Student}(x) \Rightarrow \text{Person}(x)) \tag{1}$$
$$\forall x\,(\text{Person}(x) \Rightarrow \text{Mortal}(x)) \tag{2}$$
$$\forall x, y\,(\text{ParentOf}(x, y) \Rightarrow \text{Loves}(x, y)) \tag{3}$$
$$\exists x\,(\text{Student}(x) \wedge \text{Enrolled}(x, \text{CSCI0410})) \tag{4}$$

These statements express: "All students are people", "All people are mortal", "All parents love their children", and "There exists a student enrolled in CSCI0410".

3 Inference in First Order Logic

Once we have represented knowledge using predicates and logical formulas, we need mechanisms to derive new knowledge from what we already know. This is the role of **inference** - the process of deriving logical conclusions from premises.

3.1 Key Inference Mechanisms

The examples above demonstrate several important inference mechanisms in FOL:

- **Universal Instantiation**: Replace universally quantified variables with specific constants
- **Existential Generalization**: Conclude that something exists based on a specific instance
- **Modus Ponens**: If we know $P \Rightarrow Q$ and P, we can conclude Q
- **Modus Tollens**: If we know $P \Rightarrow Q$ and $\neg Q$, we can conclude $\neg P$
- **Reasoning Chains**: Combining multiple inference steps to reach complex conclusions

These inference mechanisms allow our knowledge-based agent to answer complex queries by systematically applying logical rules to derive new facts from existing knowledge.

3.2 Example Knowledge Base: Digital Library System

Let's build a knowledge base for a digital library system to illustrate inference in FOL. Our domain includes books, authors, users, and various relationships between them.

3.2.1 Predicates

- Book(x) - "x is a book"
- Author(x) - "x is an author"
- User(x) - "x is a user"
- Genre(x) - "x is a genre"
- WroteBook(a, b) - "author a wrote book b"
- HasGenre(b, g) - "book b belongs to genre g"
- CheckedOut(u, b) - "user u has checked out book b"
- Recommends(u_1, b, u_2) - "user u_1 recommends book b to user u_2"

- Influences(a_1, a_2) - "author a_1 influences author a_2"
- PopularGenre(g) - "genre g is popular"
- Bestseller(b) - "book b is a bestseller"

3.2.2 Knowledge Base Facts

$$\text{Author(Asimov)} \land \text{Author(Clarke)} \land \text{Author(Bradbury)} \tag{5}$$
$$\text{User(Alice)} \land \text{User(Bob)} \land \text{User(Carol)} \tag{6}$$
$$\text{Book(Foundation)} \land \text{Book(Childhood'sEnd)} \land \text{Book(Fahrenheit451)} \tag{7}$$
$$\text{Genre(SciFi)} \land \text{Genre(Mystery)} \land \text{Genre(Fantasy)} \tag{8}$$
$$\text{WroteBook(Asimov, Foundation)} \tag{9}$$
$$\text{WroteBook(Clarke, Childhood'sEnd)} \tag{10}$$
$$\text{WroteBook(Bradbury, Fahrenheit451)} \tag{11}$$
$$\text{HasGenre(Foundation, SciFi)} \tag{12}$$
$$\text{HasGenre(Childhood'sEnd, SciFi)} \tag{13}$$
$$\text{HasGenre(Fahrenheit451, SciFi)} \tag{14}$$
$$\text{CheckedOut(Alice, Foundation)} \tag{15}$$
$$\text{CheckedOut(Bob, Childhood'sEnd)} \tag{16}$$
$$\text{Influences(Asimov, Clarke)} \tag{17}$$
$$\text{Bestseller(Foundation)} \tag{18}$$
$$\text{PopularGenre(SciFi)} \tag{19}$$

3.2.3 Knowledge Base Rules

$$\forall a, b\,(\text{WroteBook}(a, b) \Rightarrow \text{Author}(a) \land \text{Book}(b)) \tag{R1}$$
$$\forall u, b\,(\text{CheckedOut}(u, b) \Rightarrow \text{User}(u) \land \text{Book}(b)) \tag{R2}$$
$$\forall b, g\,(\text{HasGenre}(b, g) \land \text{PopularGenre}(g) \Rightarrow \text{Bestseller}(b)) \tag{R3}$$
$$\forall a_1, a_2, b\,(\text{Influences}(a_1, a_2) \land \text{WroteBook}(a_1, b) \land \text{Bestseller}(b) \tag{R4}$$
$$\Rightarrow \exists b_2\,(\text{WroteBook}(a_2, b_2) \land \text{Bestseller}(b_2))) \tag{20}$$
$$\forall u, b, g\,(\text{CheckedOut}(u, b) \land \text{HasGenre}(b, g) \tag{R5}$$
$$\Rightarrow \forall b_2\,(\text{HasGenre}(b_2, g) \Rightarrow \text{Recommends}(u, b_2, u))) \tag{21}$$

These rules express:

- **R1**: If someone wrote a book, they are an author and it is a book
- **R2**: If someone checked out a book, they are a user and it is a book
- **R3**: Books in popular genres become bestsellers
- **R4**: If author A influences author B, and A wrote a bestseller, then B will also write a bestseller
- **R5**: Users recommend books in the same genre as books they've checked out

3.3 Inference Examples

Now let's see how we can derive new knowledge through logical inference:

3.3.1 Example 1: Simple Modus Ponens

Query: Is Childhood'sEnd a bestseller?
Reasoning:

1. From our KB: $\text{HasGenre}(\text{Childhood'sEnd}, \text{SciFi})$ (given fact)
2. From our KB: $\text{PopularGenre}(\text{SciFi})$ (given fact)
3. From rule R3: $\forall b, g\,(\text{HasGenre}(b, g) \land \text{PopularGenre}(g) \Rightarrow \text{Bestseller}(b))$
4. By universal instantiation with $b = \text{Childhood'sEnd}$, $g = \text{SciFi}$:

$$\text{HasGenre}(\text{Childhood'sEnd}, \text{SciFi}) \land \text{PopularGenre}(\text{SciFi}) \Rightarrow \text{Bestseller}(\text{Childhood'sEnd})$$

5. By modus ponens: $\text{Bestseller}(\text{Childhood'sEnd})$

Answer: Yes, Childhood's End is a bestseller.

3.3.2 Example 2: Existential Reasoning

Query: Will Clarke write a bestseller?
Reasoning:

1. From our KB: $\text{Influences}(\text{Asimov}, \text{Clarke})$ (given fact)
2. From our KB: $\text{WroteBook}(\text{Asimov}, \text{Foundation})$ (given fact)
3. From our KB: $\text{Bestseller}(\text{Foundation})$ (given fact)
4. From rule R4:

$$\forall a_1, a_2, b\,(\text{Influences}(a_1, a_2) \land \text{WroteBook}(a_1, b) \land \text{Bestseller}(b) \Rightarrow \exists b_2\,(\text{WroteBook}(a_2, b_2) \land \text{Bestseller}(b_2)))$$

5. By universal instantiation with $a_1 = \text{Asimov}$, $a_2 = \text{Clarke}$, $b = \text{Foundation}$:

$$\text{Influences}(\text{Asimov}, \text{Clarke}) \land \text{WroteBook}(\text{Asimov}, \text{Foundation}) \land \text{Bestseller}(\text{Foundation}) \Rightarrow \exists b_2\,(\text{WroteBook}(\text{Clarke}, b_2) \land \text{Bestseller}(b_2))$$

6. By modus ponens: $\exists b_2\,(\text{WroteBook}(\text{Clarke}, b_2) \land \text{Bestseller}(b_2))$

Answer: Yes, there exists some book that Clarke will write that will be a bestseller.

3.3.3 Example 3: Complex Reasoning Chain

Query: What books does Alice recommend to herself?
Reasoning:

1. From our KB: $\text{CheckedOut}(\text{Alice}, \text{Foundation})$ (given fact)
2. From our KB: $\text{HasGenre}(\text{Foundation}, \text{SciFi})$ (given fact)
3. From rule R5:

$$\forall u, b, g\,(\text{CheckedOut}(u, b) \land \text{HasGenre}(b, g) \Rightarrow \forall b_2\,(\text{HasGenre}(b_2, g) \Rightarrow \text{Recommends}(u, b_2, u)))$$

4. By universal instantiation with $u = \text{Alice}$, $b = \text{Foundation}$, $g = \text{SciFi}$:

$$\text{CheckedOut}(\text{Alice}, \text{Foundation}) \wedge \text{HasGenre}(\text{Foundation}, \text{SciFi})$$
$$\Rightarrow \forall b_2 \, (\text{HasGenre}(b_2, \text{SciFi}) \Rightarrow \text{Recommends}(\text{Alice}, b_2, \text{Alice}))$$

5. By modus ponens:

$$\forall b_2 \, (\text{HasGenre}(b_2, \text{SciFi}) \Rightarrow \text{Recommends}(\text{Alice}, b_2, \text{Alice}))$$

6. From our KB, we know: $\text{HasGenre}(\text{Childhood'sEnd}, \text{SciFi})$ and $\text{HasGenre}(\text{Fahrenheit451}, \text{SciFi})$

7. By universal instantiation and modus ponens:

$$\text{Recommends}(\text{Alice}, \text{Childhood'sEnd}, \text{Alice}) \tag{22}$$
$$\text{Recommends}(\text{Alice}, \text{Fahrenheit451}, \text{Alice}) \tag{23}$$
$$\text{Recommends}(\text{Alice}, \text{Foundation}, \text{Alice}) \tag{24}$$

Answer: Alice recommends all science fiction books to herself: Foundation, Childhood's End, and Fahrenheit 451.