Logical Agents

- A **logical agent** is a type of intelligent agent that **uses logic** to represent knowledge and **make decisions** through **inference**. These agents are a **subset** of knowledge-based agents.

- A logical agent gathers percepts (observations), updates a knowledge base (KB) with them, and uses logical reasoning to infer new facts and decide what action to take.

- **Entailment (⊨):**
  A sentence α is entailed by a knowledge base KB (written KB ⊨ α) if α is true in all models where KB is true. **Entailment means that one statement (or set of statements) logically implies another**.

Component	Description
Knowledge Base (KB)	Stores facts about the world as **logical sentences** (e.g., propositional or first-order logic).
TELL(KB, sentence)	Adds a new fact (sentence) to the knowledge base.
ASK(KB, query)	Checks whether a fact logically follows from the KB (i.e., whether it is **entailed**).
Inference Engine	Uses logic rules to derive new facts from the known ones.
Agent Function	Decides on actions based on the answers to logical queries.

Working of Logical Agents

1. **Perceive the Environment**
   Example: The agent perceives a **breeze** in the current square.

2. **Update the Knowledge Base (TELL)**
   The agent adds this percept as a sentence to the KB.
   TELL(KB, Breeze[1,2])

3. **Reasoning (ASK)**
   The agent queries the KB to infer whether there might be a pit nearby.
   ASK(KB, Pit[1,1] ∨ Pit[1,3] ∨ Pit[2,2])

4. **Decision Making**
   Based on logical inference, the agent decides on a safe move.

5. **Act**
   The agent executes the action (e.g., move to a safe square).

Example

In the Wumpus World:

- **Squares may contain pits or the Wumpus.**
- **The agent uses propositional logic to represent rules like:**

  $$\text{Breeze}(x,y) \Leftrightarrow (\text{Pit}(x+1,y) \lor \text{Pit}(x-1,y) \lor \text{Pit}(x,y+1) \lor \text{Pit}(x,y-1))$$

"There is a breeze in square (x, y) if and only if there is a pit in any of the four neighboring squares."

Logical Agents vs Knowledge-Based Agents

Aspect	Logical Agent	Knowledge-Based Agent (KBA)
Definition	An agent that uses logic (like propositional or first-order logic) to reason and make decisions.	A broader category of agents that represent, store, and use knowledge to act intelligently.
Main Tool	Uses **formal logic** (e.g., propositional logic, FOL) for inference.	May use **logic**, **rules**, **semantic networks**, **ontologies**, etc.
Knowledge Representation	Focuses specifically on **logical sentences** (e.g., "If breeze → pit nearby").	Uses any **structured knowledge representation**, not limited to logic.
Type	A **subset** of knowledge-based agents.	A **general class** that includes logical agents and other types.
Examples	Wumpus World agent using propositional logic.	Agents using logic, rule-based systems, or other forms of knowledge.
Inference Mechanism	Relies on **logical inference** (ASK, TELL, entailment).	May use **logical inference** or **other reasoning** mechanisms.

All Logical Agents are Knowledge-Based Agents. But **not all Knowledge-Based Agents are Logical**

Unit 4

Knowledge

Knowledge Based Agent

"Education is the most powerful weapon which you can use to change the world." – Nelson Mandela

Knowledge Based Agent

- A knowledge-based agent is an artificial intelligence (AI) system that uses a knowledge base and reasoning to make decisions or take actions.
- These agents are designed to simulate human-like problem-solving by representing and manipulating knowledge about the world in a structured way.
- The knowledge base contains facts, rules, and heuristics that describe the environment or the domain the agent is working in.

- The central component of a knowledge-based agent is its knowledge base, or KB.
- A knowledge base is a set of sentences. (Here “sentence” is used as a technical term. It is related but not identical to the sentences of English and other natural languages.)
- Each sentence is expressed in a language called a knowledge representation language and represents some assertion about the world.
- There must be a way to add new sentences to the knowledge base and a way to query what is known.

- The standard names for these operations are TELL and ASK, respectively.
- Both operations may involve inference—that is, deriving new sentences from old.
- Inference must obey the requirement that when one ASKs a question of the knowledge base, the answer should follow from what has been told (or TELLed) to the knowledge base previously.
- Like all our agents, it takes a percept as input and returns an action. The agent maintains a knowledge base, KB, which may initially contain some background knowledge.

- Each time the agent program is called, it does three things. First, it TELLs the knowledge base what it perceives.
- Second, it ASKs the knowledge base what action it should perform. In the process of answering this query, extensive reasoning may be done about the current state of the world, about the outcomes of possible action sequences, and so on.
- Third, the agent program TELLs the knowledge base which action was chosen, and the agent executes the action.

function KB-AGENT(*percept*) **returns** an *action*
  persistent: *KB*, a knowledge base
    t, a counter, initially 0, indicating time

  TELL(*KB*, MAKE-PERCEPT-SENTENCE(*percept*, *t*))
  action ← ASK(*KB*, MAKE-ACTION-QUERY(*t*))
  TELL(*KB*, MAKE-ACTION-SENTENCE(*action*, *t*))
  $t \leftarrow t + 1$
  return *action*

Figure 7.1 A generic knowledge-based agent. Given a percept, the agent adds the percept to its knowledge base, asks the knowledge base for the best action, and tells the knowledge base that it has in fact taken that action.

The details of the representation language are hidden inside three functions that implement the interface between the sensors and actuators on one side and the core representation and reasoning system on the other.

1. MAKE-PERCEPT-SENTENCE constructs a sentence asserting that the agent perceived the given percept at the given time.
2. MAKE-ACTION-QUERY constructs a sentence that asks what action should be done at the current time. Finally,
3. MAKE-ACTION-SENTENCE constructs a sentence asserting that the chosen action was executed.

Example

- Imagine a medical diagnosis system. The system would have a knowledge base filled with facts about diseases, symptoms, and treatments.
- When given patient data (perceptions), the system can infer likely diagnoses and recommend treatment based on the information in its knowledge base.
- Knowledge-based agents are particularly useful in domains where complex reasoning and decision-making are required, and where expert knowledge can be structured and encoded in a way that the agent can use effectively.

Unit 4

Knowledge

Propositional Logic in AI

Lecture 2

Propositional logic

- Propositional logic is one of the simplest and most fundamental systems of logic used in artificial intelligence (AI) and other fields like mathematics and computer science.
- It provides a way to reason about statements that can be either true or false.

- A proposition is a statement that can be either true or false but not both. For example:
- "The sky is blue" (True or False depending on the actual condition).
- "It is raining" (True or False based on whether it's raining).

Types of Syntax in Propositional Logic

1. Atomic
2. Complex

Atomic Syntax (Atomic Propositions)

- The simplest type of syntax.
- Consists of a single propositional symbol (variable).
- Represented by uppercase letters: P, Q, R, A, B, etc.
- Example:

  P (It is raining)

  Q (The ground is wet)

Compound Syntax (Complex Propositions)

Formed by combining atomic propositions using logical connectives.

Logical connectives include:

- Negation (¬) → "NOT"
- Conjunction (∧) → "AND"
- Disjunction (∨) → "OR"
- Implication (→) → "IF... THEN..."
- Biconditional (↔) → "IF AND ONLY IF"

Logical Connectives:

- These are symbols used to combine propositions and create more complex logical statements. The main logical connectives are:
  - AND ($\wedge$): True if both propositions are true.
  - OR ($\vee$): True if at least one of the propositions is true.
  - NOT ($\neg$): Negates the truth value of a proposition.
  - IMPLIES ($\rightarrow$): If the first proposition is true, then the second must be true as well.
  - IF AND ONLY IF ($\leftrightarrow$): True if both propositions have the same truth value.

Logical Connectives:

- Examples:
- $\neg P$: "It is NOT raining."
- $P \land Q$: "It is raining AND the ground is wet."
- $P \lor Q$: "It is raining OR the ground is wet."
- $P \rightarrow Q$: "If it is raining, then the ground is wet."
- $P \leftrightarrow Q$: "It is raining if and only if the ground is wet."

- Propositional logic is widely used in AI, particularly in areas like:
- **Knowledge representation:** Where we represent facts as propositions.
- **Reasoning:** Where AI systems use propositional logic to infer new information from known facts.
- **Decision making:** Logic is used in decision support systems where AI needs to evaluate conditions and make decisions based on true/false values.

Example

- For example, in a simple AI system, you might have the following propositions:
- P: "It is raining."
- Q: "The ground is wet."
- You could represent the knowledge that "if it rains, the ground gets wet" as:
- $P \rightarrow Q$ (If it rains, the ground is wet).
- If you know P is true (it is raining), you can infer Q (the ground is wet).

Disadvantage

- It doesn’t handle more complex statements like those involving quantifiers (e.g., "some," "all") or relationships between different objects.
- For such cases, more advanced logic systems like first-order logic are used.

First Order Logic in AI

First Order Logic

- First-Order Logic (FOL), also called **Predicate Logic**, is a way to **represent knowledge and reason about it using logic**. It helps us talk about objects, their properties, and relationships between them.
- It's like a smarter version of basic logic (propositional logic).
- First-Order Logic is like a powerful language to describe and reason about things in the world — using variables, objects, and logical rules.

Need of First Order Logic

- **Expressing complex statements**: Like "Every student in class passed the exam" — which can't be expressed easily in propositional logic.
- **Understanding real-world scenarios**: It can represent facts, relationships, and rules — useful in AI, databases, mathematics, etc.
- **Reasoning and inference**: FOL allows us to deduce new facts logically from existing knowledge (e.g., If all birds fly and Tweety is a bird → Tweety can fly).
- **Building intelligent systems**: Used in Artificial Intelligence to make decisions, solve problems, or prove theorems logically.

Components of First Order Logic

1. **Variables**: These represent objects in the domain (the set of things we're talking about). For example, a variable might represent a person, a car, or a book.
   - Examples: x, y, z
2. **Constants:** These represent specific, fixed objects in the domain.

For example, John, Alice, or 123 could be constants representing specific individuals or entities.

   - Examples: John, Alice

3. Predicates (or Relations): Predicates represent properties of objects or relationships between objects. They are like functions that return a true/false value.

- Examples:
  - Loves(John, Alice) means "John loves Alice."
  - IsEven(x) means "x is even."
  - GreaterThan(x, y) means "x is greater than y."

4. Functions: These represent mappings from objects to other objects in the domain.

- Example: Father(x) might map, x (a person) to that person’s father.

5. **Quantifiers**: Quantifiers are symbols used to express the quantity of objects we are referring to. There are two main types:

- **Universal Quantifier (∀)**: This means "for all." It indicates that a statement holds for every object in the domain.

  "All students pass the exam."

  $\forall x(\text{Student}(x) \rightarrow \text{Passes}(x, \text{Exam}))$

- **Existential Quantifier (∃)**: This means "there exists." It indicates that there is at least one object in the domain that satisfies a given property.

  "There exists a person who likes ice cream."

  $\exists x \quad (\text{Person}(x) \land \text{Likes}(x, \text{IceCream}))$

6. Logical Connectives: These are similar to those in propositional logic, but now they operate on predicates and terms, rather than simple true/false values.

- **AND (∧)**: Conjunction.
- **OR (∨)**: Disjunction.
- **NOT (¬)**: Negation.
- **IMPLIES (→)**: Implication.
- **IF AND ONLY IF (↔)**: Biconditional. (if and only iff)

Translation Rules to follow to translate a sentence in FOL.

✅ **Step 1: Identify the domain of discourse**

- What kind of things are we talking about? (People, animals, numbers, etc.)

✅ **Step 2: Identify constants (specific named things)**

- If names like “John”, “Math”, “India” appear, treat them as constants.

✅ **Step 3: Identify variables (for generalization)**

- Replace general terms like "someone", "everyone", "something", etc., with variables.

✅ **Step 4: Define predicates**

- Based on properties or relations like "is a teacher", "likes AI", "teaches math".

✅ Step 5: Choose correct quantifiers

- "All", "Every" → ∀
- "Some", "There exists" → ∃

✅ Step 6: Use appropriate connectives

- Use ∧ when combining conditions
- Use → when one condition leads to another
- Use ¬ when something is not true

Examples

1. Every student studies

2. Some students like programming

3. No one likes failure

4. John loves everyone

5. Some animals can fly

How to convert any statement into FOL

6 easy steps to convert to FOL

Step 1: Understand the Meaning of the Statement

- Break the sentence down into its **logical components** (subject, predicate, conditions).
- Identify **who** or **what** the sentence is about, and **what is being said** about it.

Step 2: Identify Variables

- Choose variable names (e.g., x, y, `t`) to represent general individuals or entities in the domain.
- Each variable typically stands for one object/person/time/event.

Step 3: Define Predicates

- Assign **predicate symbols** to describe properties or relationships.
- A predicate is usually a function like `P(x)` for a property or `R(x, y)` for a relation.

x is a Parent

$P(x)$

$R(x, y)$

Examples:

- `Student(x)` → "x is a student"
- `Loves(x, y)` → "x loves y"

x loves y

$L(x, y)$

Parent(x, y) → x is parent

Parent(x, y).

Step 4: Identify Quantifiers : Use quantifiers to express **how many** individuals are involved.

Phrase	Quantifier	FOL Symbol
All / Every / Each	Universal	∀ (for all)
Some / At least one	Existential	∃ (there exists)
None / No	Negated ∃	¬∃ or ∀x ¬P(x)
Not all	Negated ∀	¬∀ or ∃x ¬P(x)

Step 5: Combine Using Logical Connectives

Use connectives to build complex statements:

- ∧ = and
- ∨ = or
- → = implies
- ↔ = if and only if
- ¬ = not

Step 6: Assemble the Full FOL Statement

Put it all together by inserting quantifiers, variables, and predicates in the correct logical order.

Example 1 : Every student who studies passes the exam.

$$\forall x\,(\mathrm{Student}(x) \wedge \mathrm{Studies}(x) \rightarrow \mathrm{Passes}(x))$$

1. Variables: x for a person
2. Predicates:
   - Student(x) — x is a student
   - Studies(x) — x studies
   - Passes(x) — x passes the exam
3. Quantifier: ∀x (for all x)
4. Combine:
   - "If x is a student and x studies, then x passes"

     FOL:
     $\forall x\,((\mathrm{Student}(x) \wedge \mathrm{Studies}(x)) \rightarrow \mathrm{Passes}(x))$

Q4) a) Write the following sentences in FOL(using types of quantifiers) [9]

i) All birds fly

ii) Some boys play cricket

iii) A first cousin is a child of a parent's sibling

iv) You can fool all the people some of the time, and some of the people all the time, but you cannot fool all the people all the time.

1. All birds fly

$x \rightarrow$ birds

i) Bird(x)

ii) Fly(x)

$$\boxed{\forall_x (\text{Bird}(x) \longrightarrow \text{Fly}(x))}$$

2. Some boys play cricket

x

Boy(x)

PlayCricket(x)

$$\boxed{\exists x \, (\, Boy(x) \wedge Playcricket(x) \,)}$$

3. A first cousin is a child of parents sibling

x, y → P, S

Parent (P, y) → P is Parent of y. — ①

Sibling (s, P) → S is Sibling of P — ②

Parent (S, x) → S is a parent of x — ③

S P
$|$ $|$
$x \longleftrightarrow y$

$$\boxed{\text{cousin}(x, y) \leftrightarrow \exists p \, \exists s \, (\text{Parent}(P, y) \wedge \text{Sibling}(s, P) \wedge \text{Parent}(S, x))}$$

x is a cousin of y if and only if x is a child of y's Parent sibling.

3. A first cousin is a child of parents sibling

Parent(p, y) → p is a parent of y

Sibling(s, p) → s is a sibling of y's parent
Parent(s, x) → s is a parent of x (i.e., x is the child of p's sibling)

So x is a cousin of y.

4. You can fool all the people some of the time, and some of the people all the time, but you cannot fool all the people all the time.

Step 1: Break the sentence into parts

The sentence has three clauses:

1. You can fool all the people some of the time
2. You can fool some of the people all the time
3. But you cannot fool all the people all the time

Step 2: Define predicates and variables

We need:

Variables:

- **x: a person**
- **t: a time**

Predicates:

- **Person(x): "x is a person"**
- **Time(t): "t is a moment in time"**
- **Fool(x, t): "you fool person x at time t"**

You (the speaker) is implicit, so we don't need a variable for "you" in FOL.

Step 3: Translate each clause

Clause 1: You can fool all the people some of the time

We want to say:

For every person, there exists some time at which you can fool them.

FOL:

Clause 2: You can fool some of the people all the time

We want to say:

There exists a person who can be fooled at all times.

Step 3: Translate each clause

Clause 1: You can fool all the people some of the time

We want to say:

For every person, there exists some time at which you can fool them.

FOL:

$$\forall x\, (Person(x) \rightarrow \exists t\, (Time(t) \wedge Fool(x, t)))$$

Clause 2: You can fool some of the people all the time

We want to say:

There exists a person who can be fooled at all times.

FOL:

$$\exists x\, (Person(x) \wedge \forall t\, (Time(t) \rightarrow Fool(x, t)))$$

Clause 3: You cannot fool all the people all the time

We want to say:

It is **not** true that all people can be fooled at all times.

FOL (negation of a universal):

$$\neg\forall x\ \forall t\ (Person(x) \land Time(t) \rightarrow Fool(x, t))$$

Final FOL Expression (all 3 parts together):

$$(\forall x\ (Person(x) \rightarrow \exists t\ (Time(t) \land Fool(x, t)))) \land (\exists x\ (Person(x) \land \forall t\ (Time(t) \rightarrow Fool(x, t)))) \land \neg\forall x\ \forall t\ (Person(x) \land Time(t) \rightarrow Fool(x, t))))$$

NEXT VIDEO

1. **Write the following sentences in FOL (any 2) (using types of quantifiers). [9]**

   i) Every number is either negative or has a square root .

   ii) Every connected and circuit-free graph is a tree .

   iii) Some people are either religious or pious

   iv) There is a barber who shaves all men in the town who do not shave themselves (May 2023)

Unit 4

Knowledge

Inference Rules Propositional Logic in AI

Agenda

1. What is Propositional Logic
2. Rules of Inference
3. Examples
4. How to write answer to get maximum marks

Propositional logic

- Propositional logic is one of the simplest and most fundamental systems of logic used in artificial intelligence (AI) and other fields like mathematics and computer science.
- It provides a way to reason about statements that can be either true or false.

- A proposition is a statement that can be either true or false but not both. For example:
- "The sky is blue" (True or False depending on the actual condition).
- "It is raining" (True or False based on whether it's raining).

Types of Syntax in Propositional Logic

1. Atomic
2. Complex

Atomic Syntax (Atomic Propositions)

- The simplest type of syntax.
- Consists of a single propositional symbol (variable).
- Represented by uppercase letters: P, Q, R, A, B, etc.
- Example:

  P (It is raining)

  Q (The ground is wet)

Compound Syntax (Complex Propositions)

Formed by combining atomic propositions using logical connectives.

Logical connectives include:

- Negation (¬) → "NOT"
- Conjunction (∧) → "AND"
- Disjunction (∨) → "OR"
- Implication (→) → "IF... THEN..."
- Biconditional (↔) → "IF AND ONLY IF"

Logical Connectives:

- These are symbols used to combine propositions and create more complex logical statements. The main logical connectives are:
  - AND ($\wedge$): True if both propositions are true.
  - OR ($\vee$): True if at least one of the propositions is true.
  - NOT ($\neg$): Negates the truth value of a proposition.
  - IMPLIES ($\rightarrow$): If the first proposition is true, then the second must be true as well.
  - IF AND ONLY IF ($\leftrightarrow$): True if both propositions have the same truth value.

Logical Connectives:

- Examples:
- $\neg P$: "It is NOT raining."
- $P \wedge Q$: "It is raining AND the ground is wet."
- $P \vee Q$: "It is raining OR the ground is wet."
- $P \rightarrow Q$: "If it is raining, then the ground is wet."
- $P \leftrightarrow Q$: "It is raining if and only if the ground is wet."

- Propositional logic is widely used in AI, particularly in areas like:
- **Knowledge representation:** Where we represent facts as propositions.
- **Reasoning:** Where AI systems use propositional logic to infer new information from known facts.
- **Decision making:** Logic is used in decision support systems where AI needs to evaluate conditions and make decisions based on true/false values.

Example

- For example, in a simple AI system, you might have the following propositions:
- P: "It is raining."
- Q: "The ground is wet."
- You could represent the knowledge that "if it rains, the ground gets wet" as:
- $P \rightarrow Q$ (If it rains, the ground is wet).
- If you know P is true (it is raining), you can infer Q (the ground is wet).

Inference Rules

University Questions

1. List the inference rules used in propositional logic? Explain them in detail with suitable example. [9 Marks]   May 2022
2. Explain Inference in Propositional Logic. [8 Marks] May 2024

Inference in First Order Logic

Inference

Inference is the process of **deriving new logical conclusions (statements)** from a set of **known facts** using valid **rules of reasoning.**

- **Examples of Inference:**

Example 1 :

- Known: All metals conduct electricity.
- Known: Copper is a metal.
- 👉 Inference:

Example 2 :

- Known: If someone is running, they are probably in a hurry.
- Known: Ravi is running.
- 👉 Inference:

Examples of Inference

- **Examples of Inference:**

Example 1 :

- Known: All metals conduct electricity.
- Known: Copper is a metal.
- 👉 Inference: Copper conducts electricity.

Example 2 :

- Known: If someone is running, they are probably in a hurry.
- Known: Ravi is running.
- 👉 Inference: Ravi is probably in a hurry.

Inference means using what you already know to figure out something new that logically follows from it.

First Order Inference

Inference in First-Order Logic helps us derive new knowledge from what we already know.

First-order inference can be done by converting the knowledge base to propositional logic and propositional inference.

Steps to generate inferences from first-order logic statements.

1. Inference Rules for Quantifiers
2. Reducing FOL to Propositional Logic

1. Inference Rules for Quantifiers

1. **Universal Instantiation (UI)**

2. Existential Instantiation (EI)

1. Universal Instantiation (UI)

If we know a rule applies to **all objects**, we can apply it to a **specific one**.

Example:

```
∀x (King(x) ∧ Greedy(x) ⇒ Evil(x))
→ King(John) ∧ Greedy(John) ⇒ Evil(John)
```


Here, we just replace x with a specific name like John.

Universal Instantiation

Let’s say we know this universal rule:

> 'All greedy kings are evil.'

In logic, that’s written as:

$\forall x\ (King(x) \land Greedy(x) \Rightarrow Evil(x))$

Universal Instantiation means:

If it’s true for all, then it must be true for specific people too.

So we can plug in names:

- $King(John) \land Greedy(John) \Rightarrow Evil(John)$
- $King(Richard) \land Greedy(Richard) \Rightarrow Evil(Richard)$

2. Existential Instantiation (EI)

If we know **something exists**, we can assume a name for it (but it should be a new name not used elsewhere).

Example:

```
∃x (Crown(x) ∧ OnHead(x, John))
→ Crown(C1) ∧ OnHead(C1, John)
```

Here, C1 is a made-up name for some crown — it’s called a **Skolem constant**.

Existential Instantiation

Now let’s talk about Existential Instantiation. This rule says:

> 'If something exists, let’s give it a name.'

For example:

$\exists x\ (Crown(x) \land OnHead(x, John))$

Means – there exists some crown on John’s head.

We give it a name like `C1` and write:

$Crown(C1) \land OnHead(C1, John)$

2. Reducing FOL to Propositional Logic