Knowledge Representation and Reasoning
تمثيل المعرفة والتفكير

Knowledge representation is a fundamental aspect of artificial intelligence (AI) and plays a crucial role in various AI applications. It involves encoding information and knowledge in a form that can be understood and processed by computers. Logical representations are a common approach to knowledge representation, and they enable computers to reason and make inferences based on the provided knowledge.

عد تمثيل المعرفة جانبًا أساسيًا من الذكاء الاصطناعي (AI(ويلعب دورًا حاسمًا في تطبيقات الذكاء الاصطناعي المختلفة. ويتضمن ترميز المعلومات والمعرفة في شكل يمكن فهمه ومعالجته بواسطة أجهزة الكمبيوتر. التمثيلات المنطقية هي أسلوب شائع لتمثيل المعرفة، وهي تمكن أجهزة الكمبيوتر من التفكير والاستنتاجات بناءً على المعرفة المقدمة.

1. Knowledge Representation Using Logic:

- Knowledge in AI is often represented using formal logic, a systematic way to represent information and make inferences. Logic provides a structured and rigorous framework for representing and reasoning about the world.

- Logical knowledge representation typically involves using symbols, operators, and rules to represent facts, relationships, and rules in a formal way.

- It's essential for knowledge to be represented accurately to enable intelligent systems to draw meaningful conclusions.

1. تمثيل المعرفة باستخدام المنطق:

- غالبًا ما يتم تمثيل المعرفة في الذكاء الاصطناعي باستخدام المنطق الرسمي، وهي طريقة منهجية لتمثيل المعلومات وإجراء الاستدلالات. يوفر المنطق إطارًا منظمًا وصارمًا لتمثيل العالم والتفكير فيه.
- يتضمن تمثيل المعرفة المنطقية عادةً استخدام الرموز والعوامل والقواعد لتمثيل الحقائق والعلاقات والقواعد بطريقة رسمية.
- من الضروري أن يتم تمثيل المعرفة بدقة لتمكين الأنظمة الذكية من استخلاص استنتاجات ذات معنى.

Here are 10 examples of knowledge representation using logic in AI:

Example 1:

Fact: All birds have wings.

Logical representation: ∀x (bird(x) → has_wings(x))

Example 2:

Rule: If it is raining, then the ground is wet.

Logical representation: (raining → wet_ground)

Example 3:

Fact: John is a bird.

Logical representation: bird(John)

Example 4:

Fact: Tweety is a bird.

Logical representation: bird(Tweety)

Example 5:

Rule: If a bird has wings, then it can fly.

Logical representation: (has_wings(x) → can_fly(x))

Example 6:

Fact: Tweety has wings.

Logical representation: has_wings(Tweety)

Example 7:

Inference: Therefore, Tweety can fly.

Logical representation: can_fly(Tweety)

Example 8:

Fact: All animals are mortal.

Logical representation: $\forall x\ (animal(x) \rightarrow mortal(x))$

Example 9:

Fact: John is an animal.

Logical representation: animal(John)

Example 10:

Inference: Therefore, John is mortal.

Logical representation: mortal(John)

Here are a few more examples:

Fact: All humans are mortal.
Logical representation: ∀x (human(x) → mortal(x))
Fact: Socrates is a human.
Logical representation: human(Socrates)
Inference: Therefore, Socrates is mortal.
Logical representation: mortal(Socrates)

Fact: All squares have four sides.
Logical representation: ∀x (square(x) → has_four_sides(x))
Fact: A rectangle is a square.
Logical representation: rectangle(x) → square(x)
Inference: Therefore, all rectangles have four sides.
Logical representation: ∀x (rectangle(x) → has_four_sides(x))

Rule: If a student is studying hard, then they will get good grades.
Logical representation: (studying_hard(x) → good_grades(x))
Fact: John is a student.
Logical representation: student(John)
Inference: If John is studying hard, then he will get good grades.
Logical representation: studying_hard(John) → good_grades(John)

These examples illustrate the power of logic for knowledge representation in AI. Logic allows us to represent knowledge in a clear, unambiguous, and efficient way, and it enables us to develop AI systems that can reason about and solve problems intelligently.

توضح هذه الأمثلة قوة المنطق في تمثيل المعرفة في الذكاء الاصطناعي. يسمح لنا المنطق بتمثيل المعرفة بطريقة واضحة لا لبس فيها وفعالة، ويمكّننا من تطوير أنظمة الذكاء الاصطناعي القادرة على التفكير وحل المشكلات بذكاء.

2. Propositional Logic:

- Propositional logic deals with propositions or statements that can be either true or false.
- It uses logical operators like AND, OR, NOT, IMPLIES, and IF AND ONLY IF to connect and manipulate propositions.
- Propositional logic is useful for representing simple, binary relationships and facts, but it lacks the ability to represent more complex relationships or variables.

2. المنطق المقترح:

-يتعامل المنطق الافتراضي مع القضايا أو العبارات التي يمكن أن تكون صحيحة أو خاطئة.

-يستخدم عوامل تشغيل منطقية مثل ANDو ORو NOTو IMPLIESو IFو ONLY IFلتوصيل المقترحات ومعالجتها.

-المنطق الافتراضي مفيد في تمثيل العلاقات والحقائق الثنائية البسيطة، لكنه يفتقر إلى القدرة على تمثيل العلاقات أو المتغيرات الأكثر تعقيدًا.

Here are 10 examples of propositional logic:

Example 1:

Fact: It is raining outside.

Propositional representation: P ≡ "It is raining outside."

Example 2:

Rule: If it is raining outside, then the ground is wet.

Propositional representation: $(P \rightarrow Q)$ ≡ "If it is raining outside, then the ground is wet."

Example 3:

Fact: The ground is wet.

Propositional representation: Q ≡ "The ground is wet."

Example 4:

Inference: Therefore, it is raining outside.

Propositional representation: $(P \rightarrow Q) \land Q \rightarrow P$ ≡ "If it is raining outside, then the ground is wet. And if the ground is wet, then it is raining outside."

Example 5:

Fact: It is not raining outside.

Propositional representation: $\neg P$ ≡ "It is not raining outside."

Example 6:

Rule: **If it is not raining outside, then the sun is shining.**

Propositional representation: **$(\neg P \rightarrow R) \equiv$ "If it is not raining outside, then the sun is shining."**

Example 7:

Fact: **The sun is shining.**

Propositional representation: **$R \equiv$ "The sun is shining."**

Example 8:

Inference: **Therefore, it is not raining outside.**

Propositional representation: **$(\neg P \rightarrow R) \land R \rightarrow \neg P \equiv$ "If it is not raining outside, then the sun is shining. And if the sun is shining, then it is not raining outside."**

Example 9:

Fact: **It is raining outside or the sun is shining.**

Propositional representation: **$P \lor R \equiv$ "It is raining outside or the sun is shining."**

Example 10:

Rule: **If it is raining outside and the sun is shining, then there is a rainbow.**

Propositional representation: **$(P \land R \rightarrow S) \equiv$ "If it is raining outside and the sun is shining, then there is a rainbow."**

Here are 10 examples of using quantifiers and predicate logic:

Example 1:

Statement: All birds have wings.

Quantifier: ∀x

Predicate: bird(x) ∧ has_wings(x)

Logical representation: ∀x (bird(x) → has_wings(x))

Example 2:

Statement: No birds are mammals.

Quantifier: ∀x

Predicate: bird(x) ∧ ¬mammal(x)

Logical representation: ∀x (bird(x) → ¬mammal(x))

Example 3:

Statement: Some birds can fly.

Quantifier: ∃x

Predicate: bird(x) ∧ can_fly(x)

Logical representation: ∃x (bird(x) ∧ can_fly(x))

Example 4:

Statement: Most birds live in trees.

Quantifier: ∀x

Predicate: bird(x) → lives_in_tree(x)

Logical representation: ∀x (bird(x) → lives_in_tree(x))

Example 5:

Statement: Every student who studies hard will get good grades.

Quantifier: **∀x**

Predicate: **student(x) ∧ studies_hard(x) → good_grades(x)**

Logical representation: **∀x (student(x) ∧ studies_hard(x) → good_grades(x))**

Example 6:

Statement: There is at least one student who is both intelligent and hardworking.

Quantifier: **∃x**

Predicate: **student(x) ∧ intelligent(x) ∧ hardworking(x)**

Logical representation: **∃x (student(x) ∧ intelligent(x) ∧ hardworking(x))**

Example 7:

Statement: All even numbers are divisible by 2.

Quantifier: **∀x**

Predicate: **even(x) → divisible_by(x, 2)**

Logical representation: **∀x (even(x) → divisible_by(x, 2))**

Example 8:

Statement: No prime number is even.

Quantifier: **∀x**

Predicate: **prime(x) ∧ ¬even(x)**

Logical representation: **∀x (prime(x) → ¬even(x))**

Example 9:

Statement: Some numbers are both odd and prime.

Quantifier: ∃x

Predicate: odd(x) ∧ prime(x)

Logical representation: ∃x (odd(x) ∧ prime(x))

Example 10:

Statement: Most circles are not perfect.

Quantifier: ∀x

Predicate: circle(x) → ¬perfect(x)

Logical representation: ∀x (circle(x) → ¬perfect(x))

3. First-Order Logic (FOL):

- First-order logic is an extension of propositional logic that allows for the representation of complex relationships and variables.
- FOL introduces quantifiers (e.g., ∀ for "for all" and ∃ for "there exists") to express statements about all or some objects in a domain.
- It uses predicates, functions, and variables to represent and reason about knowledge in a more expressive manner.

3. منطق الدرجة الأولى:

- منطق الدرجة الأولى هو امتداد للمنطق الافتراضي الذي يسمح بتمثيل العلاقات والمتغيرات المعقدة.
- يقدم FOLمحددات الكمية (على سبيل المثال، ∀ لـ "للكل" و ∃ لـ "يوجد") للتعبير عن بيانات حول كل أو بعض الكائنات في المجال.
- يستخدم المسندات والوظائف والمتغيرات لتمثيل المعرفة وتفسيرها بطريقة أكثر تعبيراً.

Here are 10 examples of using first-order logic (FOL):

Example 1:

Statement: All birds have wings.
FOL representation: $\forall x\,(bird(x) \rightarrow has\_wings(x))$
The quantifier ∀x means "for all x," the predicate bird(x) means "x is a bird," and the predicate has_wings(x) means "x has wings."

Example 2:

Statement: No birds are mammals.

FOL representation: ∀x (bird(x) → ¬mammal(x))

The ¬ symbol means "not."

Example 3:

Statement: Some birds can fly.

FOL representation: ∃x (bird(x) ∧ can_fly(x))

The quantifier ∃x means "there exists an x."

Example 4:

Statement: Most birds live in trees.

FOL representation: ∀x (bird(x) → lives_in_tree(x))

Example 5:

Statement: Every student who studies hard will get good grades.

FOL representation: ∀x (student(x) ∧ studies_hard(x) → good_grades(x))

Example 6:

Statement: There is at least one student who is both intelligent and hardworking.

FOL representation: ∃x (student(x) ∧ intelligent(x) ∧ hardworking(x))

Example 7:

Statement: All even numbers are divisible by 2.

FOL representation: ∀x (even(x) → divisible_by(x, 2))

Example 8:

Statement: No prime number is even.

FOL representation: ∀x (prime(x) → ¬even(x))

Example 9:

Statement: Some numbers are both odd and prime.

FOL representation: ∃x (odd(x) ∧ prime(x))

Example 10:

Statement: Most circles are not perfect.

FOL representation: ∀x (circle(x) → ¬perfect(x))

Example 11:

Statement: John is a student.

FOL representation: student(John)

Example 12:

Statement: Mary is a teacher.

FOL representation: teacher(Mary)

Example 13:
Statement: John loves Mary.
FOL representation: loves(John, Mary)

FOL can also be used to represent more complex relationships and variables, such as:
Parent-child relationships: parent(John, Mary)
Friendship relationships: friend(John, Mary)
Ownership relationships: owns(John, car)
Location relationships: is_located_at(John, home)

4. Inference Rules and Reasoning with Logical Statements:

- Inference rules are a set of rules and procedures that enable logical reasoning. They allow us to draw conclusions based on the knowledge represented in logic.
- Common inference rules include modus ponens, modus tollens, generalization, specialization, and resolution.
- Reasoning with logical statements involves applying these inference rules to deduce new knowledge from existing knowledge.

4. قواعد الاستدلال والاستدلال باستخدام العبارات المنطقية:

- قواعد الاستدلال هي مجموعة من القواعد والإجراءات التي تمكن من التفكير المنطقي. إنها تسمح لنا باستخلاص استنتاجات بناءً على المعرفة الممثلة في المنطق.
- تتضمن قواعد الاستدلال الشائعة طريقة القياس، وطريقة الضرب، والتعميم، والتخصص، والحل.
- يتضمن الاستدلال باستخدام العبارات المنطقية تطبيق قواعد الاستدلال هذه لاستنتاج معرفة جديدة من المعرفة الموجودة.

Here are 10 examples of modus ponens reasoning:

Example 1:

Logical statements:

If it is raining, then the ground is wet.

It is raining.

Conclusion: The ground is wet.

Example 2:

Logical statements:

All birds have wings.

Tweety is a bird.

Conclusion: Tweety has wings.

Example 3:

Logical statements:

If you study hard, you will get good grades.

I studied hard.

Conclusion: I will get good grades.

Example 4:

Logical statements:

All mammals are animals.

Humans are mammals.

Conclusion: Humans are animals.

Example 5:

Logical statements:
If you break the law, you will go to jail.
I broke the law.
Conclusion: I will go to jail.

Example 6:

Logical statements:
All squares have four sides.
A rectangle is a square.
Conclusion: A rectangle has four sides.

Example 7:

Logical statements:
If you eat too much sugar, you will get cavities.
I ate too much sugar.
Conclusion: I will get cavities.

Example 8:

Logical statements:
All even numbers are divisible by 2.
10 is an even number.
Conclusion: 10 is divisible by 2.

Example 9:

Logical statements:
If you get an A on this test, you will pass the class.
I got an A on this test.
Conclusion: I will pass the class.

Example 10:

Logical statements:
All prime numbers are greater than 1.
11 is a prime number.
Conclusion: 11 is greater than 1.

Practical part:

If you have this text paragraph: "Sunlight is essential for photosynthesis. Photosynthesis is the process by which plants convert sunlight into energy. This energy is used for plant growth and development."

Use propositional logic to represent the following paragraph. Then write full code in python. First ask in away in which answer is true and then ask to make the answer false.

```python
import re

# Sample text paragraph with three sentences
text_paragraph = (
    "Sunlight is essential for photosynthesis. "
    "Photosynthesis is the process by which plants convert sunlight into energy. "
    "This energy is used for plant growth and development."
)

# Function to extract propositions from sentences
def extract_propositions(sentence):
    # Split the sentence into words
    words = sentence.split()
    propositions = []
    for word in words:
        # Use regular expression to match propositions (e.g., "Sunlight is essential
        matches = re.findall(r"\b[a-zA-Z']+[- ]?[a-zA-Z']+[- ]?[a-zA-Z']*\b", word)
        propositions.extend(matches)
    return propositions

# Function to convert propositions to propositional variables
def propositions_to_variables(propositions):
    variable_map = {}
    variables = []
    variable_count = 1
    for prop in propositions:
        if prop not in variable_map:
            variable_map[prop] = f"P{variable_count}"
            variable_count += 1
        variables.append(variable_map[prop])
    return variables

# Function to convert text paragraph to propositional logic
def text_to_propositional_logic(text):
    sentences = text.split('. ')
    propositions = []
    for sentence in sentences:
        propositions.extend(extract_propositions(sentence))

    variables = propositions_to_variables(propositions)
```

```python
    propositional_logic = ""
    for i, sentence in enumerate(sentences):
        if i > 0:
            propositional_logic += " AND "
        propositional_logic += "("
        for var in extract_propositions(sentence):
            propositional_logic += variables[propositions.index(var)] + " AND "
        propositional_logic = propositional_logic[:-5]  # Remove the trailing " AND
        propositional_logic += ")"

    return variables, propositional_logic

# Convert the text paragraph to propositional logic
variables, propositional_logic = text_to_propositional_logic(text_paragraph)

# Display the results
print("Mapping of Propositional Variables to Text:")
for i, var in enumerate(variables, start=1):
    print(f"{var}: {extract_propositions(text_paragraph)[i-1]}")

print("\nPropositional Logic:")
print(propositional_logic)
# Function to answer a question about the propositional logic
def answer_question(question, variable_truth_values):
    # Check if the question is true based on the propositional logic
    if eval(question, variable_truth_values):
        return "Answer: Yes, the statement is true."
    else:
        return "Answer: No, the statement is not true."

# Specify a question (as a Python expression)
question = "(P1 and P2) and P3"

# Answer the question
result = answer_question(question, {var: True for var in variables})

# Display the question and answer
print("Question:", question)
print(result)
```

When running the code :

```
Question: (P1 and P2) and (not P3)
Answer: No, the statement is not true.
```

As shown the answer will be true.

If you change Question into:

```
Question: (P1 and P2) and (not P3)
Answer: No, the statement is not true.
```

The answer will be no.

New Text Paragraph:

"Photosynthesis is the process by which plants convert sunlight into energy."
"This energy is vital for plant growth and development."
"Sunlight plays a crucial role in photosynthesis."

Use the same code to answer three questions where answer is true. Then another 3 questions in which answer is false.

For true answer

question1 = "P1 and P2"
question2 = "P2 and P3"
question3 = "P1 and P2 and P3"

For false answer

question1 = "P1 and not P2"
question2 = "P2 and not P3"
question3 = "not P1 and not P2 and not P3"

New Text Paragraph:

"Water is essential for life on Earth. Plants and animals depend on water for survival. The water cycle ensures the continuous circulation of water on our planet. Human activities can impact the quality and availability of freshwater resources.“

Use propositional logic to represent the above paragraph then write 3 questions with their answers “true”. Then write 3 question so their answers will be “false”.

Questions with True Answers:

question1 = "P1 and P2"
question2 = "P1 and P3"
question3 = "P2 and P4"

Questions with False Answers:

question4 = "P1 and not P2"
question5 = "P2 and not P4"
question6 = "P3 and not P1"

Expert Systems النظم الخبيرة

Expert systems are computer programs that use artificial intelligence (AI) to simulate the decision-making and problem-solving abilities of a human expert in a particular domain. Expert systems are typically used to solve complex problems that require a deep understanding of the domain and the ability to reason about a variety of factors.

الأنظمة الخبيرة هي برامج كمبيوتر تستخدم الذكاء الاصطناعي (AI) لمحاكاة قدرات اتخاذ القرار وحل المشكلات لدى خبير بشري في مجال معين. تُستخدم الأنظمة الخبيرة عادةً لحل المشكلات المعقدة التي تتطلب فهمًا عميقًا للمجال والقدرة على التفكير بشأن مجموعة متنوعة من العوامل.

Expert systems are typically composed of two main components: a knowledge base and an inference engine. The knowledge base contains the expert knowledge about the domain, which is typically represented in the form of rules, facts, and cases. The inference engine uses the knowledge base to generate conclusions and solutions to problems.

تتكون الأنظمة الخبيرة عادة من مكونين رئيسيين: قاعدة المعرفة ومحرك الاستدلال. تحتوي قاعدة المعرفة على معرفة الخبراء حول المجال، والتي يتم تمثيلها عادةً في شكل قواعد وحقائق وحالات. يستخدم محرك الاستدلال قاعدة المعرفة لتوليد استنتاجات وحلول للمشكلات.

Rule-based Expert Systems

Rule-based expert systems are the most common type of expert system. Rule-based expert systems represent the expert knowledge in the form of a set of rules. The rules are typically in the form of IF-THEN statements. For example, a rule for diagnosing a disease might be:

الأنظمة الخبيرة المبنية على القواعد هي النوع الأكثر شيوعًا من الأنظمة الخبيرة. تمثل الأنظمة الخبيرة المبنية على القواعد المعرفة المتخصصة في شكل مجموعة من القواعد. القواعد عادة ما تكون في شكل عبارات .IF-THENعلى سبيل المثال، قد تكون قاعدة تشخيص المرض:

IF the patient has a fever AND the patient has a cough THEN the patient has the flu.

The inference engine in a rule-based expert system uses the rules to reason about the problem and generate a conclusion. The inference engine starts with the known facts about the problem and then applies the rules to generate new facts. This process continues until the inference engine reaches a conclusion or determines that there is no solution to the problem.

يستخدم محرك الاستدلال في النظام الخبير القائم على القواعد القواعد للتفكير في المشكلة والتوصل إلى نتيجة. يبدأ محرك الاستدلال بالحقائق المعروفة حول المشكلة ثم يطبق القواعد لتوليد حقائق جديدة. تستمر هذه العملية حتى يصل محرك الاستدلال إلى نتيجة أو يقرر أنه لا يوجد حل للمشكلة.

Here is an example of a medical diagnosis system:

Knowledge base:

The knowledge base of a medical diagnosis system contains information about various diseases, their symptoms, and the diagnostic tests that can be used to diagnose them. The knowledge base is typically represented in the form of rules, facts, and cases.

For example, the knowledge base might contain the following rule:

IF the patient has a fever AND the patient has a cough AND the patient has chest pain THEN the patient has pneumonia.

The knowledge base might also contain the following fact:
A fever is a temperature above 98.6 degrees Fahrenheit.

And the knowledge base might contain the following case:
Patient A is a 50-year-old male with a fever, cough, and chest pain. Patient A was diagnosed with pneumonia.

Inference engine:
The inference engine of a medical diagnosis system uses the knowledge base to generate conclusions and solutions to problems. The inference engine starts with the known facts about the patient (such as the patient's symptoms and medical history) and then applies the rules in the knowledge base to generate new facts. This process continues until the inference engine reaches a conclusion (such as a diagnosis) or determines that there is no solution to the problem.

For example, the inference engine might start by reasoning about the patient's fever:

IF the patient has a temperature above 98.6 degrees Fahrenheit THEN the patient has a fever.

The inference engine knows that the patient's temperature is 100 degrees Fahrenheit, so it can conclude that the patient has a fever.

The inference engine might then reason about the patient's cough:

IF the patient has a cough AND the patient has a fever THEN the patient has a respiratory infection.

The inference engine knows that the patient has a fever and that the patient has a cough, so it can conclude that the patient has a respiratory infection.

The inference engine might then reason about the patient's chest pain:

IF the patient has a respiratory infection AND the patient has chest pain THEN the patient has pneumonia.

The inference engine knows that the patient has a respiratory infection and that the patient has chest pain, so it can conclude that the patient has pneumonia.
The inference engine has now reached a conclusion: the patient has pneumonia.

This is just a simple example of a medical diagnosis system. Medical diagnosis systems can be much more complex, and they can be used to diagnose a wide variety of diseases. Medical diagnosis systems are a valuable tool for doctors, but they are not a substitute for human judgment. Doctors should always use their own clinical judgment when making a diagnosis.

Now we will convert our example to predicate logic:

Knowledge base:

Rule section

fever(Patient) :- temperature(Patient) > 98.6.
respiratory_infection(Patient) :- fever(Patient), cough(Patient).
pneumonia(Patient) :- respiratory_infection(Patient), chest_pain(Patient).

Fact section

fever(Patient). **The patient has a fever.**
respiratory_infection(Patient). **The patient has a respiratory infection.**
pneumonia(Patient). **The patient has pneumonia.**

Inference engine section

The inference engine would work by first querying the knowledge base to determine if the patient has a fever:
?- fever(Patient).

If the patient has a fever, then the inference engine would query the knowledge base to determine if the patient has a respiratory infection:
?- respiratory_infection(Patient).

If the patient has a respiratory infection, then the inference engine would query the knowledge base to determine if the patient has pneumonia:
?- pneumonia(Patient).

If the patient has pneumonia, then the inference engine would conclude that the patient has pneumonia.

This is a very simple example, but it shows how predicates can be used to represent the knowledge base and the inference engine of a medical diagnosis system.

Here is an example of how the inference engine would work to diagnose a patient with pneumonia:
?- fever(Patient).
Yes.
?- respiratory_infection(Patient).
Yes.
?- pneumonia(Patient).
Yes.
Therefore, the inference engine concludes that the patient has pneumonia.

Here is an example of how the inference engine could use a rule to ask if the patient has pneumonia:

```
rule_ask_pneumonia(Patient) :-
    fever(Patient), respiratory_infection(Patient),
    write('Does the patient have chest pain? (y/n) '),
    read(ChestPain),
    ChestPain = 'y'.
```

```
?- fever(Patient), respiratory_infection(Patient).
Yes.

?- rule_ask_pneumonia(Patient).

Does the patient have chest pain? (y/n) y.
Yes.
```