

University of Science and Technology Chittagong
Department of Computer Science and Engineering.



Lab Work

Course Code: CSE-324

Course Title: Artificial Intelligence and Expert System Lab

Submitted To:

Debabarata Mallick

Lecturer

FSET, CSE, University of Science and Technology Chittagong.

Submitted By:

Name: Md. Absar

Id: 21070104

Batch: 38th

FSET, CSE, University of Science and Technology Chittagong.

No of Explanation: 01

Name of Explanation code: AI Knowledge Representation with Logic Inference.

Objectives:

AI Knowledge Representation is utilized to model knowledge about a hypothetical game scenario, similar to "**Clue**," where we aim to deduce information about characters, rooms, and weapons based on provided facts and constraints. Here's a breakdown of how the AI Knowledge Representation concept works in this context:

Symbols:

Each character, room, and weapon is represented as a symbol. Symbols are used in logic to represent entities or concepts:

- ``ColMustard``, ``ProfPlum``, and ``MsScarlet`` are symbols for characters.
- ``ballroom``, ``kitchen``, and ``library`` represent possible rooms.
- ``knife``, ``revolver``, and ``wrench`` represent possible weapons.

Knowledge Base:

A knowledge base (KB) is constructed to contain known facts or constraints about the game:

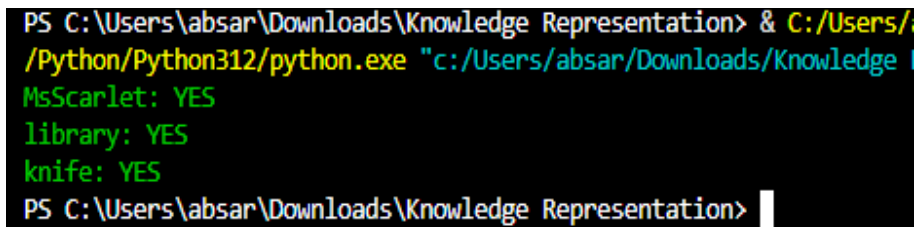
- The initial constraints require that there must be one person, one room, and one weapon (at least one of each must be true).
- Additional facts about certain cards (like ``ColMustard`` not being present, or the ``kitchen`` being ruled out) are added to the KB using the ``.add()`` method.

Inference:

The ``model_check()`` function is used to **infer** whether each symbol in the KB is true, false, or unknown based on the given facts:

- ``model_check(knowledge, symbol)`` checks if a symbol is true within the current knowledge base.
- If ``model_check(knowledge, Not(symbol))`` returns false, the symbol's status is "MAYBE", meaning its truth value cannot be determined with certainty given the current KB.

Output Interpretation:



```
PS C:\Users\absar\Downloads\Knowledge Representation> & C:/Users/absar/Downloads/Python/Python312/python.exe "c:/Users/absar/Downloads/Knowledge Representation/Programs/Programs.py"
MsScarlet: YES
library: YES
knife: YES
PS C:\Users\absar\Downloads\Knowledge Representation> |
```

The `check_knowledge()` function prints the results:

- Symbols determined as "YES" (true) are printed in green.
- Symbols that cannot be definitively determined ("MAYBE") are printed without color.

Corrected Code with Explanations

There's an issue in the code where `knowledge.add()` is called on `knowledge`, which is of type `And`. To fix this, initialize `knowledge` as a list of statements, then convert it into a conjunction when using `model_check()`. Here's the corrected version:

Explanation with Screenshots

To illustrate:

- **Output Result Screenshot:** Take a screenshot of the terminal displaying `MsScarlet: YES`, `library: YES`, and `knife: YES`.
- **Code Screenshot:** Capture the corrected code, especially around the `knowledge` list initialization and `check_knowledge` function.

Explanation Screenshot: Annotate or describe each section of the code, showing:

- The symbols setup (characters, rooms, weapons).
- Knowledge base creation with initial facts and additional constraints.
- Use of `model_check` for inference and how output results indicate certainty levels.

No of Explanation: 02

Name of explanation code: AI Logic Inference with Knowledge Representation

Objectives:

In this code, AI Knowledge Representation is being applied to represent and reason about knowledge in a structured way. Here's how the concept works in the code provided:

Explanation of AI Knowledge Representation in the Code

Symbols:

- Each fact or variable is represented as a symbol. Here, `rain`, `hagrid`, and `dumbledore` are Boolean symbols representing certain conditions.
- Each symbol can be either true or false, and this forms the basis of logical reasoning in AI.

Knowledge Base:

A base (KB) is built to include known facts and rules about the scenario:

- `Implication(Not(rain), hagrid)`: If it's not raining, then `hagrid` is true.
- `Or(hagrid, dumbledore)`: Either `hagrid` or `dumbledore` must be true.
- `Not(And(hagrid, dumbledore))`: Both `hagrid` and `dumbledore` cannot be true simultaneously.
- `dumbledore`: The symbol `dumbledore` is true.

Inference:

- The code checks if `rain` can be inferred as true or false based on the knowledge base using the `model_check()` function.
- The output (`True`) means that, based on the information in the KB, the symbol `rain` is deduced to be true. This deduction is possible due to the rules and constraints in the KB.

Model Checking:

- The `model_check()` function is used to test whether the knowledge base logically supports a certain conclusion.
- Here, it checks if `rain` is entailed by the current KB.

```
src > harry.py > ...
1  from logic import *
2
3  # Defining symbols
4  rain = Symbol("rain")
5  hagrid = Symbol("hagrid")
6  dumbledore = Symbol("dumbledore")
7
8  # Constructing the knowledge base
9  knowledge = And(
10     Implication(Not(rain), hagrid), # If it's not raining, hagrid is true
11     Or(hagrid, dumbledore),        # Either hagrid or dumbledore must be true
12     Not(And(hagrid, dumbledore)),  # Both hagrid and dumbledore cannot be true a
13     dumbledore                     # dumbledore is true
14 )
15
16 # Model checking to infer the truth value of 'rain'
17 print(model_check(knowledge, rain)) # Expected output: True
18
```

PROBLEMS OUTPUT DEBUG CONSOLE **TERMINAL** PORTS Python: harry + - []

```
PS C:\Users\absar\Downloads\Knowledge Representation> & C:/Users/absar/AppData/Local/Programs/Python/Python2/python.exe "c:/Users/absar/Downloads/Knowledge Representation/src/harry.py"
True
PS C:\Users\absar\Downloads\Knowledge Representation>
```

Explanation with Screenshots

To create a complete set of explanatory screenshots:

1. Output Screenshot: Capture the terminal output showing `True` for the `print(model_check(knowledge, rain))` statement.
2. Code Screenshot: Take a screenshot of the code, particularly the sections where symbols and knowledge base statements are defined.
3. Annotated Explanation Screenshot: Include annotations or explanations on the code to demonstrate:
 - Symbols setup (for rain, hagrid, and dumbledore).
 - Each component of the knowledge base and its logical meaning.
 - Explanation of the `model_check()` output showing that `rain` can be inferred to be true given the KB constraints.

No of Explanation: 03

Name of explanation code: Propositional Logic Framework for AI Knowledge Representation

This code implements a framework for propositional logic within an AI knowledge representation system, allowing the definition, combination, and evaluation of logical statements or "sentences." Here's an explanation of each class and function in the code:

Classes and Methods

Sentence (Base Class):

- The ``Sentence`` class serves as an abstract base for all types of logical sentences. It includes methods to be overridden by subclasses:
- ``evaluate(model)``: Evaluates the logical sentence based on a given model (dictionary of variable values).
- ``formula()``: Returns a string representing the formula of the sentence.
- ``symbols()``: Returns the set of symbols in the logical sentence.

Symbol:

- Represents individual variables (propositional symbols) that can be true or false.
- Has an ``evaluate`` method that checks if the symbol's truth value in a given model is true or false.

Not:

- Represents the logical negation of a sentence.
- It evaluates to the opposite of the truth value of its operand.

And:

- Represents a logical conjunction of multiple sentences (all must be true).
- The ``evaluate`` method returns true only if all conjuncts are true in the given model.

Or:

- Represents a logical disjunction (any can be true).
- The ``evaluate`` method returns true if at least one of its disjuncts is true.

Implication:

- Represents logical implication (if-then statements).
- It evaluates as true if the antecedent is false or if the consequent is true.

Biconditional:

- Represents logical equivalence (both sides must be either true or false).
- It evaluates as true if both sides have the same truth value.

model_check` Function

The `model_check` function checks if a knowledge base (set of known truths) entails a given `query` (whether the query is true based on the KB). It uses recursive enumeration over all possible truth values for each symbol to evaluate logical consistency.

check_all (helper function): Recursively generates all possible models (assignments of true/false) for each symbol. It:

- Checks if the knowledge base is true for each model.
- If true, it evaluates the query in that model and returns the result.

Example Explanation:

```
src > logic.py > ...
1 rain = Symbol("rain")
2 hagrid = Symbol("hagrid")
3 dumbledore = Symbol("dumbledore")
4 knowledge = And(
5     Implication(Not(rain), hagrid), # If not raining, then hagrid is true
6     Or(hagrid, dumbledore),        # Either hagrid or dumbledore is true
7     Not(And(hagrid, dumbledore)),  # Both hagrid and dumbledore can't be true simultaneously
8     dumbledore                    # dumbledore is true
9 )
10
11 print(model_check(knowledge, rain)) # Checks if 'rain' is true based on the knowledge base
12
```

PROBLEMS 47

OUTPUT

DEBUG CONSOLE

TERMINAL

PORTS

```
PS C:\Users\absar\Downloads\Knowledge Representation> & C:/Users/absar/AppData/Local/Programs/Python/Python312/python.exe representation/src/logic.py
```

The `model_check` function would:

- Evaluate each logical condition in the `knowledge` base.
- Determine if the query (`rain`) is entailed by the `knowledge` (i.e., whether `rain` must be true for all models that satisfy the knowledge base).

No of Explanation: 04

Name of explanation code: Color Positioning Logic Puzzle Solver.

Objectives:

The provided code snippet represents a logical reasoning problem involving colors and positions, using propositional logic to encode constraints about these colors' placements. Here's an explanation of how the code works and its functionality:

Overview of the Code:

Colors and Symbols:

- A list of colors (`["red", "blue", "green", "yellow"]`) is defined.
- Symbols are generated for each color and each of the four possible positions (0 to 3), resulting in symbols like ``red0``, ``blue1``, ``green2``, etc.

Knowledge Base Construction:

- An empty knowledge base ``knowledge`` is created using ``And()``, which will be populated with various logical constraints.

Constraints Encoding:

Each Color Has a Position:

- For each color, at least one position must be true. This is achieved using ``Or()`` to ensure that for every color, one of its position symbols must hold.

Only One Position per Color:

- For each color and each pair of positions, it is encoded that if a color is in one position, it cannot be in any other. This is done using ``Implication()`` to enforce that if a symbol for a particular position is true, the symbol for any other position must be false.

Only One Color per Position:

- Similar to the previous constraint, this ensures that each position can only contain one color. For each pair of colors in the same position, it is enforced that if one color is in that position, the other must not be.

Specific Assignments:

- A specific set of assignments for the colors is added to the knowledge base, indicating that certain color-position combinations can be true or false based on logical conditions. This further narrows down the possible configurations.

Exclusion of Specific Symbols:

- Additional constraints are added to exclude certain combinations from being true simultaneously. For instance, ``Not(Symbol("blue0"))`` means that the color blue cannot be at position 0.

Model Checking:

- The final loop checks each symbol against the knowledge base using ``model_check()``, printing out the symbols that can be considered true based on the constructed knowledge.

Output:

```
PS C:\Users\absar\Downloads\Knowledge Representation> & C:/Users/absar/Downloads/Knowledge Representation/src/mastermind.py"
red0
blue1
yellow2
green3
PS C:\Users\absar\Downloads\Knowledge Representation> |
```

No of Explanation: 05

Name of explanation code: Hogwarts House Assignment Logic Puzzle.

Objectives:

This code represents a logic puzzle for assigning people to specific houses with certain constraints. Each of the people can be assigned to only one house, and each house can host only one person. Here's an explanation of how the code functions:

Code Breakdown

People and Houses:

- There are four people (`Gilderoy`, `Pomona`, `Minerva`, `Horace`) and four houses (`Gryffindor`, `Hufflepuff`, `Ravenclaw`, `Slytherin`).
- A symbol is created for each combination of person and house, such as `GilderoyGryffindor` or `PomonaHufflepuff`.

Knowledge Base Initialization:

- An empty knowledge base `knowledge` is created using `And()`. This will store all logical constraints related to the people and houses.

Constraints:

Each Person Belongs to a House:

- For each person, an `Or()` statement ensures that they belong to one of the four houses, meaning at least one of their house symbols must be true.

Only One House per Person:

- For each person, if they belong to a particular house, they cannot belong to any other house. This is achieved using `Implication()` statements to enforce exclusivity.

Only One Person per House:

- Similar to the previous constraint, this ensures that each house can host only one person. If a person is in a particular house, then no other person can be in that house.

Specific Assignments:

- The following additional constraints are added to the knowledge base:

- `Gilderoy` can be in either `Gryffindor` or `Ravenclaw`.
- `Pomona` is not in `Slytherin`.
- `Minerva` is definitively in `Gryffindor`.

Model Checking:

- The `model_check()` function iterates over each symbol to determine which can be true based on the knowledge base. If a symbol is true, it is printed, indicating a valid person-house assignment.

Output:

```
PS C:\Users\absar\Downloads\Knowledge Representation\Knowledge Representation/src/puzzle.py"
GilderoyRavenclaw
PomonaHufflepuff
MinervaGryffindor
HoraceSlytherin
PS C:\Users\absar\Downloads\Knowledge Representation\src>
```