LAB-8: FOL using Unification.

Observation book:

classmate
Date 19/11/2024
Page

First Order logic: [Unification]

"If every dog has a tail, and all dogs that are friendly are also playful, and Fido is a friendly dog, then Fido is playful and has a tail"

* Dog(x) : "x is a dog"
* Tail(x) : "x has a tail"
* Friendly(x) : "x is friendly"
* Playful(x) : "x is playful"

FOL representation:

1 * $\forall x\ (Dog(x) \rightarrow Tail(x))$
- For every x, if x is a dog, then x has a tail

2 * $\forall x\ ((Dog(x) \wedge Friendly(x)) \rightarrow Playful(x))$
- For every x, if x is a dog and x is friendly, then x is playful.

3 * Friendly (Fido)
- Fido is friendly

4 * Dog (Fido)
- Fido is a dog

Unification:

* From 1 & 4, substituting x with Fido,
  Dog(Fido) → Tail(Fido)
- By Modus Ponens, Tail (Fido) —— 5
  (Fido has a tail)

* From 2, 3 & 4, substituting x with Fido,

$(Dog(Fido) \land Friendly(Fido)) \rightarrow Playful(Fido)$

- By Modus Ponens, $Playful(Fido)$ —— 6
  (Fido is playful)

* From 5 and 6,

$Playful(Fido) \land Tail(Fido)$

- Fido is playful and Fido has a tail

code::

```
import re
# Define a simple function for extracting predicates from sentences
def extract_predicate(sentence):
# Regular expression to find patterns like Predicate(Argument)
pattern = r"([A-Za-z]+)\((\w+)\)"
match = re.search(pattern, sentence)
if match:
predicate = match.group(1)
subject = match.group(2)
return predicate, subject
return None, None
# Function for unification
def unify(fact, query):
# Check if the fact and query are the same
if fact == query:
return True
# Extract predicate and subject from fact and query
```

```
fact_predicate, fact_subject = extract_predicate(fact)
query_predicate, query_subject = extract_predicate(query)
# If predicates match, unify the subjects
if fact_predicate == query_predicate:
if fact_subject == query_subject:
return True
else:
# Here, we could handle variable substitution (unification)
return False
return False
# Function to deduce the goal using given rules
def deduct(rules, goal):
# Try to find unification for the goal from the rules
for rule in rules:
if unify(rule, goal):
print(f"Unification successful: {rule} matches with {goal}.")
return True
return False
# Main function to handle user input
def main():
# Step 1: Get the rules (facts/implications) from the user
print("Enter the rules (facts/implications). Type 'done' to finish entering rules.")
rules = []
while True:
rule_input = input("Enter rule: ")
if rule_input.lower() == 'done':
break
else:
rules.append(rule_input.strip())
# Step 2: Get the goal (query) from the user
goal_input = input("Enter the goal (query) to prove: ").strip()
```

Step 3: Try to deduce the goal using the given rules

print("\nAttempting to deduce the goal...")

if deduct(rules, goal_input):

print(f"Conclusion: The goal '{goal_input}' is true based on the rules.")

else:

print(f"Conclusion: The goal '{goal_input}' cannot be proven with the provided rules.")

Run the program

main()

print("Nikhilesh 1bm22cs181")

Output: Output:

```
Enter the rules (facts/implications). Type 'done' to finish entering rules.
Enter rule: all birds can fly
Enter rule: bluey is a bird
Enter rule: done
Enter the goal (query) to prove: bluey can fly

Attempting to deduce the goal...
Unification successful: all birds can fly matches with bluey can fly.
Conclusion: The goal 'bluey can fly' is true based on the rules.
Nikhilesh 1bm22cs181
```