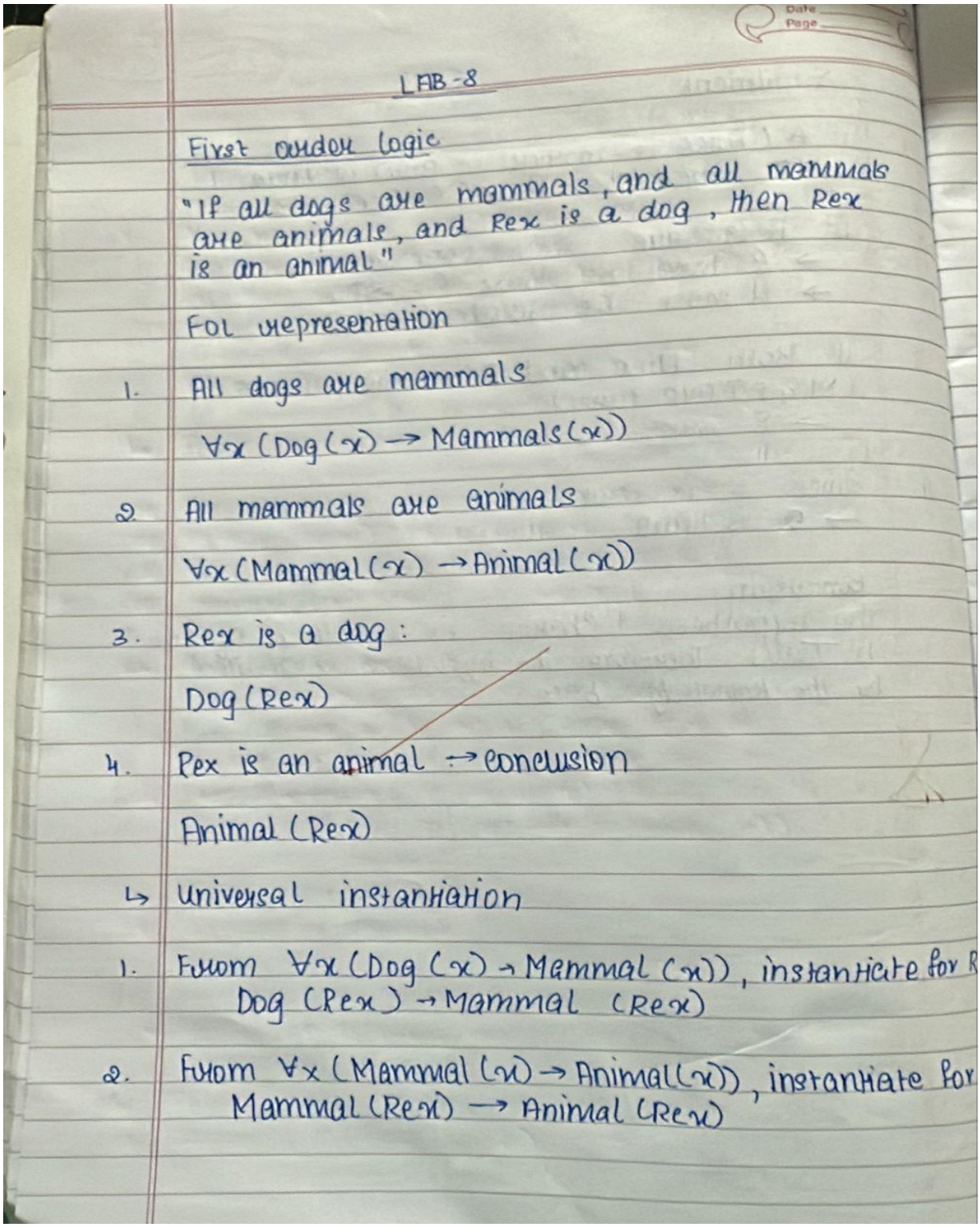
LAB-8 - FOL using Unification.

Observation book:



LAB-8

First order logic

"If all dogs are mammals, and all mammals are animals, and Rex is a dog, then Rex is an animal"

FOL representation

1. All dogs are mammals

   $\forall x (Dog(x) \rightarrow Mammals(x))$

2. All mammals are animals

   $\forall x (Mammal(x) \rightarrow Animal(x))$

3. Rex is a dog :

   $Dog(Rex)$

4. Rex is an animal → conclusion

   $Animal(Rex)$

↳ Universal instantiation

1. From $\forall x (Dog(x) \rightarrow Mammal(x))$, instantiate for R
   $Dog(Rex) \rightarrow Mammal(Rex)$

2. From $\forall x (Mammal(x) \rightarrow Animal(x))$, instantiate for
   $Mammal(Rex) \rightarrow Animal(Rex)$

→ Modus Ponens (first step)

Dog(Rex) and Dog(Rex) → Mammal(Rex) imply Mammal(Rex)

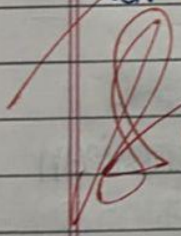
So, we conclude : Mammal(Rex)

→ Modus Ponens (second step)

conclude that : Animal(Rex)

Conclusion:

$\forall x$(Dog(x) → Mammal(x))
$\forall x$(Mammal(x) → Animal(x))
Dog(Rex)
Rex is an animal



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:
```

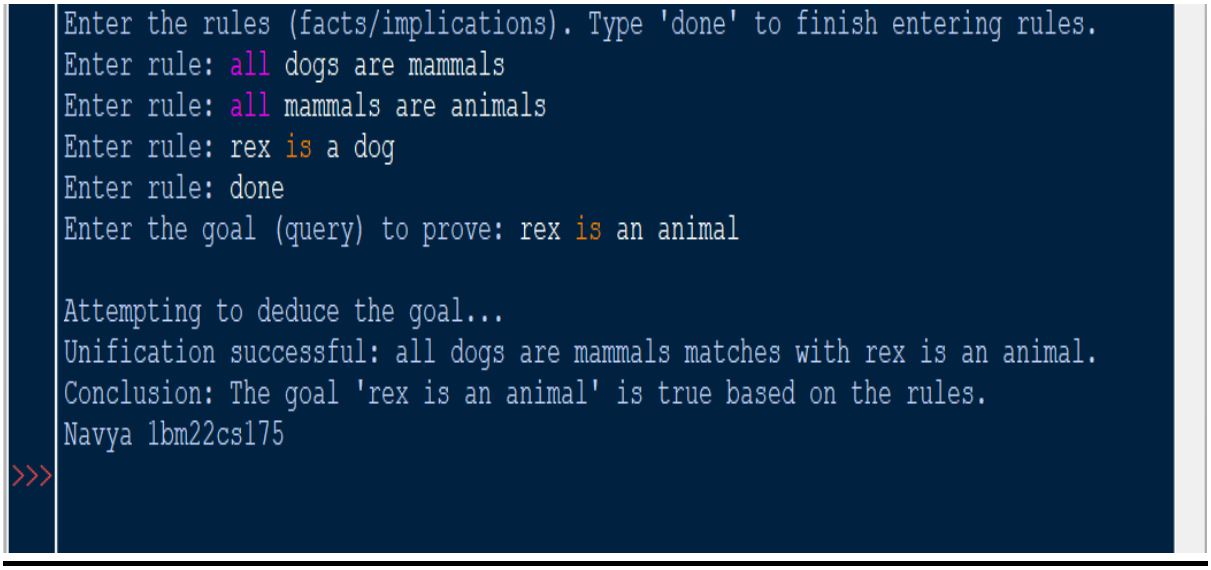
```
        rules.append(rule_input.strip())

    # 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("Navya 1bm22cs175")
```

Output:


Enter the rules (facts/implications). Type 'done' to finish entering rules.
Enter rule: all dogs are mammals
Enter rule: all mammals are animals
Enter rule: rex is a dog
Enter rule: done
Enter the goal (query) to prove: rex is an animal
Attempting to deduce the goal...
Unification successful: all dogs are mammals matches with rex is an animal.
Conclusion: The goal 'rex is an animal' is true based on the rules.
Navya 1bm22cs175
>>>