Week 9

FOL-prove the query using resolution

Code:

```
#Define the knowledge base (KB)
KB = {
    "food(Apple)": True,
    "food(vegetables)": True,
    "eats(Anil, Peanuts)": True,
    "alive(Anil)": True,
    "likes(John, X)": "food(X)",  # Rule: John likes all food
    "food(X)": "eats(Y, X) and not killed(Y)",  # Rule: Anything eaten and
not killed is food
    "eats(Harry, X)": "eats(Anil, X)",  # Rule: Harry eats what Anil eats
    "alive(X)": "not killed(X)",  # Rule: Alive implies not killed
    "not killed(X)": "alive(X)",  # Rule: Not killed implies alive
}

# Function to evaluate if a predicate is true based on the KB
def resolve(predicate):
    # If it's a direct fact in KB
    if predicate in KB and isinstance(KB[predicate], bool):
        return KB[predicate]

    # If it's a derived rule
    if predicate in KB:
        rule = KB[predicate]
        if " and " in rule:  # Handle conjunction
            sub_preds = rule.split(" and ")
            return all(resolve(sub.strip()) for sub in sub_preds)
        elif " or " in rule:  # Handle disjunction
            sub_preds = rule.split(" or ")
            return any(resolve(sub.strip()) for sub in sub_preds)
        elif "not " in rule:  # Handle negation
            sub_pred = rule[4:]  # Remove "not "
            return not resolve(sub_pred.strip())
        else:  # Handle single predicate
            return resolve(rule.strip())
```

```
    # If the predicate is a specific query (e.g., likes(John, Peanuts))
    if "(" in predicate:
        func, args = predicate.split("(")
        args = args.strip(")").split(", ")
        if func == "food" and args[0] == "Peanuts":
            return resolve("eats(Anil, Peanuts)") and not
resolve("killed(Anil)")
        if func == "likes" and args[0] == "John" and args[1] == "Peanuts":
            return resolve("food(Peanuts)")

    # Default to False if no rule or fact applies
    return False

# Query to prove: John likes Peanuts
query = "likes(John, Peanuts)"
result = resolve(query)

# Print the result
print(f"Does John like peanuts? {'Yes' if result else 'No'}")
```

Output:

```
Does John like peanuts? Yes
```

Observation:

Week-9

Create a knowledge Base consisting of first order logic statements and prove the query using resolution.

Given knowledge base

(a) John likes all kinds of food
(b) Apple and vegetables are food
(c) Anything anyone eats and not killed is food
(d) Anil eats peanuts and still alive
(e) Harry eats everything that anil eats
(f) Anyone who is alive implies not killed.
(g) Anyone who is not killed is alive.
(h) John likes peanuts.

↓ To be proved by resolution.

Representation:

(a) $\forall x$: food(x) → likes (John, x)
(b) food (Apple) ∧ food (vegetables)
(c) $\forall x \forall y$: eats (x, y) ∧ ¬ killed (x) → food (y)
(d) eats (Anil, peanuts) ∧ alive (Anil)
(e) $\forall x$: eats (Anil, x) → eats (Harry, x)
(f) $\forall x$: ¬killed (x) → alive (x)
(g) $\forall x$: alive (x) → ¬ killed (x)
h. likes (John, peanuts)

Eliminate implications:

$\alpha \Rightarrow \beta$ with $\neg \alpha \vee \beta$

(a) $\forall x$ ¬ food (x) ∨ likes (John, x)
(b) food (apple) ∧ food (vegetables)
(c) $\forall x \forall y$ ¬ [eats (x, y) ∧ ¬ killed (x)] ∨ food (y)
(d) eats (Anil, peanuts) ∧ alive (Anil)
(e) $\forall x$ ¬ eats (Anil, x) ∨ eats (Harry, x)

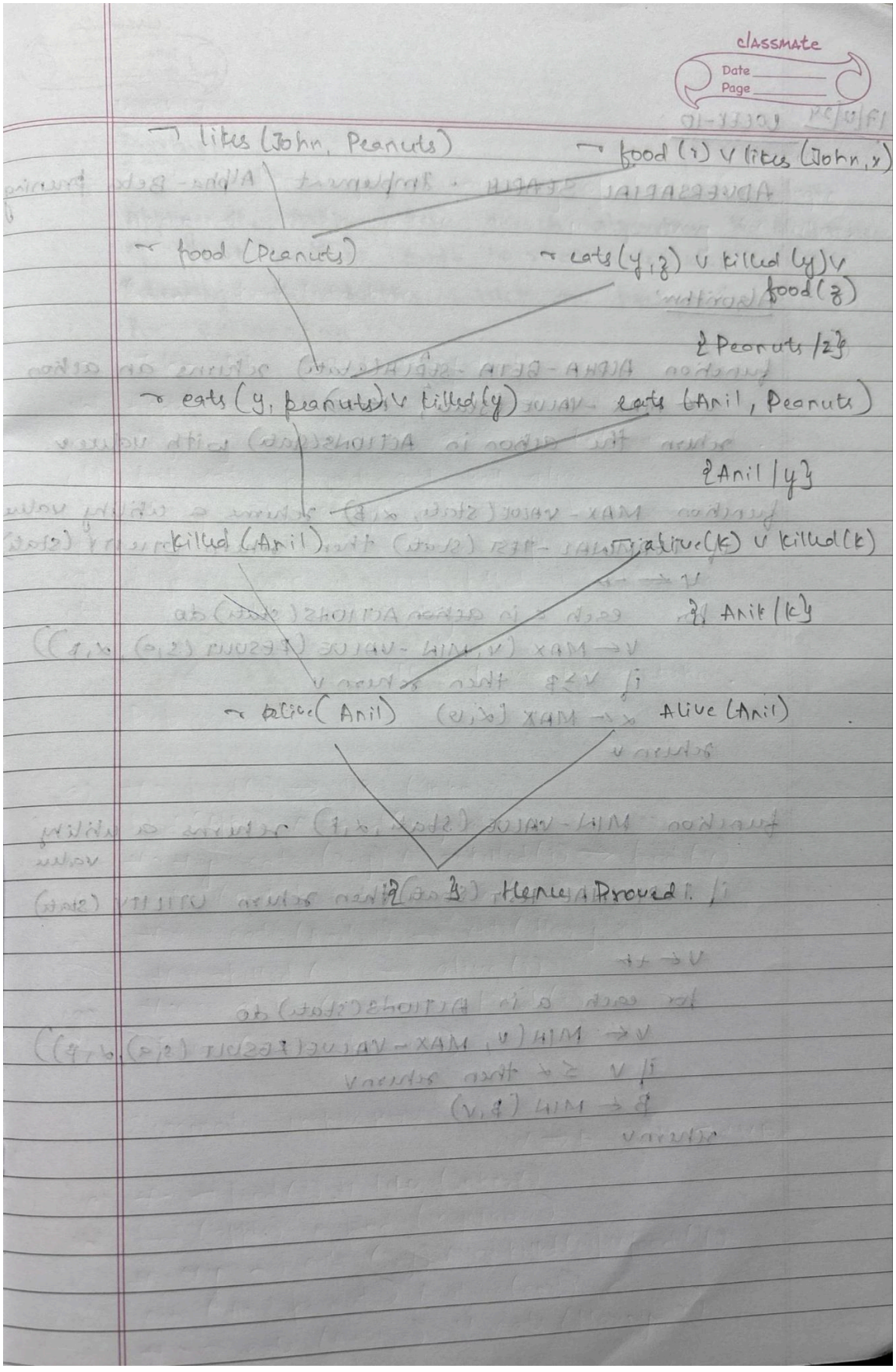
(f) $\forall x \rightarrow [(\neg \text{killed}(x)] \vee \text{alive}(x)$
(g) $\forall x \rightarrow \text{alive}(x) \vee \neg \text{killed}(x)$
h. likes (~~peanuts~~ John, peanuts)

Move negation ($\neg$) inwards and rewrite

(a) $\forall x \neg \text{food}(x) \vee \text{likes}(\text{John}, x)$
(b) food (Apple) $\wedge$ food (vegetables)
(c) $\forall y \forall z \neg \text{eats}(y, z) \vee$ ~~if~~ killed (y) $\vee$ food (z)
(d) eats (Anil, peanuts) $\wedge$ alive (Anil)
(e) $\forall w \neg \text{eats}(\text{Anil}, w) \vee \text{eats}(\text{Harry}, w)$
(f) $\forall g [\text{killed}(g)] \vee \text{alive}(g)$
(g) $\forall k \neg \text{alive}(k) \vee \neg$ ~~after~~ killed (k)
h. likes (John, peanuts)

Drop universals.

(a) $\neg \text{food}(x) \vee \text{likes}(\text{John}, x)$
(b) food (Apple)
(c) food (vegetables)
(d) $\neg \text{eats}(y, z) \vee \text{killed}(y) \vee \text{food}(z)$
(e) eats (Anil, peanuts)
(f) alive (Anil)
(g) $\neg \text{eats}(\text{Anil}, w) \vee \text{eats}(\text{Harry}, w)$
(h) killed (g) $\vee$ ~~killed (k)~~ alive (g)
(i) $\neg$ alive (k) $\vee \neg$ killed (g)
(j) likes (John, peanuts)


¬ likes (John, Peanuts)
¬ food (x) V likes (John, x)
¬ food (Peanuts)
¬ eats (y, z) V killed (y) V food (z)
{Peanuts / z}
¬ eats (y, Peanuts) V killed (y)
eats (Anil, Peanuts)
{Anil / y}
killed (Anil)
¬ alive (k) V killed (k)
{Anil / k}
¬ alive (Anil)
Alive (Anil)
Hence Proved !