Week 6

PropositionalLogic_TruthTableEnumeration

Code:

```
#Create a knowledge base using propositional logic and show that the given
query entails the knowledge base or not.
import itertools

# Function to evaluate an expression
def evaluate_expression(a, b, c, expression):
    # Use eval() to evaluate the logical expression
    return eval(expression)

# Function to generate the truth table and evaluate a logical expression
def truth_table_and_evaluation(kb, query):
    # All possible combinations of truth values for a, b, and c
    truth_values = [True, False]
    combinations = list(itertools.product(truth_values, repeat=3))

    # Reverse the combinations to start from the bottom (False -> True)
    combinations.reverse()

    # Header for the full truth table
    print(f"{'a':<5}   {'b':<5} {'c':<5} {'KB':<20}{'Query':<20}")

    # Evaluate the expressions for each combination
    for combination in combinations:
        a, b, c = combination

        # Evaluate the knowledge base (KB) and query expressions
        kb_result = evaluate_expression(a, b, c, kb)
        query_result = evaluate_expression(a, b, c, query)

        # Replace True/False with string "True"/"False"
        kb_result_str = "True" if kb_result else "False"
        query_result_str = "True" if query_result else "False"

        # Convert boolean values of a, b, c to "True"/"False"
        a_str = "True" if a else "False"
```

```python
        b_str = "True" if b else "False"
        c_str = "True" if c else "False"

        # Print the results for the knowledge base and the query
        print(f"{a_str:<5} {b_str:<5} {c_str:<5} {kb_result_str:<20} {query_result_str:<20}")

    # Additional output for combinations where both KB and query are true
    print("\nCombinations where both KB and Query are True:")
    print(f"{'a':<5}    {'b':<5} {'c':<5} {'KB':<20}{'Query':<20}")

    # Print only the rows where both KB and Query are True
    for combination in combinations:
        a, b, c = combination

        # Evaluate the knowledge base (KB) and query expressions
        kb_result = evaluate_expression(a, b, c, kb)
        query_result = evaluate_expression(a, b, c, query)

        # If both KB and query are True, print the combination
        if kb_result and query_result:
            a_str = "True" if a else "False"
            b_str = "True" if b else "False"
            c_str = "True" if c else "False"
            kb_result_str = "True" if kb_result else "False"
            query_result_str = "True" if query_result else "False"
            print(f"{a_str:<5} {b_str:<5} {c_str:<5} {kb_result_str:<20} {query_result_str:<20}")

# Define the logical expressions as strings
kb = "(a or c) and (b or not c)"  # Knowledge Base
query = "a or b"  # Query to evaluate

# Generate the truth table and evaluate the knowledge base and query
truth_table_and_evaluation(kb, query)
```

Output:

```
a       b     c     KB                   Query
False False False False                 False
False False True  False                 False
False True  False False                 True
False True  True  True                  True
True  False False True                  True
True  False True  False                 True
True  True  False True                  True
True  True  True  True                  True

Combinations where both KB and Query are True:
a       b     c     KB                   Query
False True  True  True                  True
True  False False True                  True
True  True  False True                  True
True  True  True  True                  True
```

Observation:

12/11/24

Week-6

Implementation of truth-table enumeration algorithm for deciding propositional enlightment.

Algorithm:

Step-1 : define symbols and expressions

(i) Define propositional symbols (A, B, C)

(ii) define components of the knowledge base and the query (α)

Step 2 : Generate the truth table

(i) use itertools, products to generate all truth assignments for A, B, C

(ii) evaluate and print the values of A, B, C, A∨C, B∨¬C, KB and query(α) for each.

Step 3 : Check alignment

(i) for each truth assignment, check if KB implies the queries. (if KB is true, then query (α) must also be true)

Step 4 : print enligtnment result.

(i) print wheather the KB entails the query or not based on the enlightment check.

Output:

A	B	C	A∨C	B∨¬C	KB	α(A∨B)
False	False	False	False	True	False	False
False	False	True	true	False	False	False
False	true	False	False	True	False	true
False	true	True	true	True	True	true
True	False	False	true	true	true	True
True	False	true	true	False	False	true
True	True	False	true	true	True	True
True	True	true	true	true	True	True

Result:

KB entails the query (α)

19/11/24

PREPOSITION LOGIC:

1. Either John is not stupid and he is lazy or he is stupid. John is stupid therefore john is not lazy

S = stupid

L = lazy

either john is not stupid and he is lazy or he is stupid.

$(\neg S \wedge L) \vee S$

premise 1 = either john is not stupid and he is lazy or he is stupid

premise 2 : John is stupid.

conclusion: John is not lazy.

S	L	¬S	¬L	$(\neg S \wedge L) \vee S$	Premise 1	P2
T	T	F	F	T	T	T
T	F	F	T	T	T	T
F	T	T	F	T	T	F
F	F	T	T	F	F	F

argument is invalid because both premise are true ∴ conclusion is false.

2. ~~if P then Q else R~~

Algorithm:

~~P → Q ∨ ¬P → R~~

(i) Preprocess the sentence (normalize and tokenize)

(ii) identify sentence components (subject, predicate, quantifier, connectives)

(iii) convert to FOL

(iv) universal, existential, quantification, conjunction, implication, negation

(v) construct FOL expression

(vi) Return FOL expression.

2. If P then Q else R

$P \to Q \lor \neg P \to R$

P	Q	R	$P \to Q$	$\neg P$	$\neg P \to R$	if P then Q else R
T	T	T	T	F	T	T
T	T	F	T	F	T	T
T	F	T	F	F	T	T
T	F	F	F	F	T	F
F	T	T	T	T	T	T
F	T	F	T	T	F	F
F	F	T	T	T	T	T
F	F	F	T	T	F	F

First order logic:

1. John is human
   human(John)
2. Every human is mortal
   $\forall x(\text{human}(x) \to \text{mortal}(x))$
3. John lovesy mary
   loves (John, mary)
4. there is someone who loves mary
   $\exists x(\text{loves}(x, \text{mary}))$
5. all dogs are animals
   $\forall x(\text{dog}(x) \to \text{animals}(x))$
6. some dogs are brown
   $\exists x(\text{dog}(x) \land \text{brown}(x))$
7. there is no person who is both a bachelor and married
   $\neg \exists x(\text{bachelor}(x) \land \text{married}(x))$
8. mary is the mother of john
   mother(mary, john)
9. john and mary both are students
   student(John) $\land$ student(mary)
10. if it is raining, then the ground is wet
    raining $\to$ wet(ground)

11. there is a person who knows every other person
$\exists x \forall y (Person(x) \wedge person(y)) \wedge x \neq y \rightarrow knows(x, y))$

12. Nobody is taller than themselves
$\forall x \rightarrow taller(x, x)$

13. all students in the class passed the exam
$\forall x (student(x) \wedge inclass(x) \rightarrow passed(x))$

14. mary has a pet dog
$\exists x (dog(x) \wedge petof(mary, x))$

15. if alice is a teacher, then alice teaches math
$teacher(alice) \rightarrow teaches(alice, math)$

16. everyone loves someone
$\forall x \exists y (loves(x, y))$

17. no one is both teacher and a student
$\forall x \neg (teacher(x) \wedge student(x))$

18. every man respects his parents
$\forall x (man(x) \rightarrow \exists y (parent(y, x) \wedge respects(x, y)))$

19. not all students like both math and science
$\neg \forall x (student(x) \rightarrow (likes(x, math) \wedge likes(x, science)))$

2. Transition of formal statements to english

1. $x . (H(x) y . \neg M(x, y)) \; U(x)$ where $H(x)$ = man = x
$M(x, y)$ x is married to y
$U(x)$ = x is unhappy
x, y = range over people

There exists a person who is not married to anyone and is unhappy

2. 2. $P(z, x) \; S(z, y) \; W(y)$ where $P(z, x)$ = z is parent of x
$S(z, y)$ = z is sibling of y
$W(y)$ = y is women

There exists a person who is parent of x, is a sibling of y and y is women.