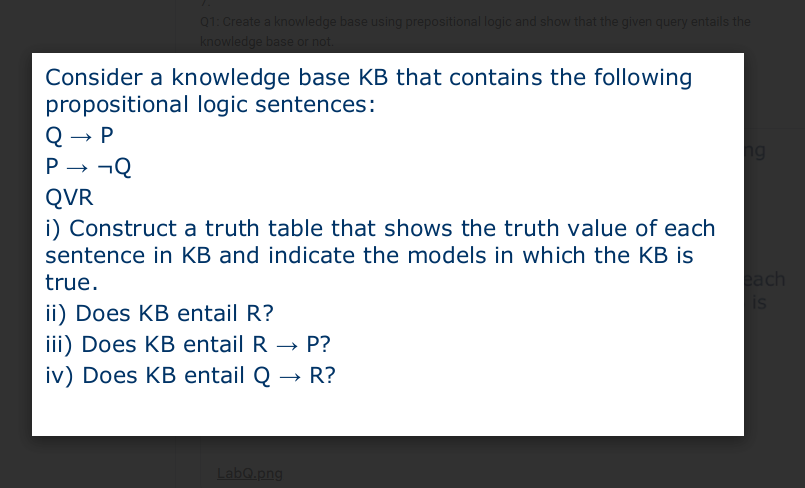
LAB 6

Propositional Logic

1.



from itertools import product

# Define propositional variables

variables = ['P', 'Q', 'R']

# Define formulas in KB

def implies(a, b):

return (not a) or b

def KB(P, Q, R):

# Q → P

s1 = implies(Q, P)

# P → ¬Q

s2 = implies(P, not Q)

# Q ∨ R

s3 = Q or R

return s1, s2, s3

# Queries

def query\_R(P, Q, R):

return R

def query\_R\_implies\_P(P, Q, R):

return implies(R, P)

def query\_Q\_implies\_R(P, Q, R):

return implies(Q, R)

# Print table header

print(f"{'P':^3} {'Q':^3} {'R':^3} {'Q→P':^5} {'P→¬Q':^6} {'Q∨R':^5} {'KB True?':^8} {'R':^3} {'R→P':^5} {'Q→R':^5}")

models\_true = []

for P, Q, R in product([False, True], repeat=3):

s1, s2, s3 = KB(P, Q, R)

kb\_true = s1 and s2 and s3

if kb\_true:

models\_true.append((P, Q, R))

print(f"{int(P):^3} {int(Q):^3} {int(R):^3} {int(s1):^5} {int(s2):^6} {int(s3):^5} {int(kb\_true):^8} "

f"{int(query\_R(P,Q,R)):^3} {int(query\_R\_implies\_P(P,Q,R)):^5} {int(query\_Q\_implies\_R(P,Q,R)):^5}")

# Determine entailments

def entails(query\_fn):

for P, Q, R in models\_true:

if not query\_fn(P, Q, R):

return False

return True

print("\nKB True in Models:")

for m in models\_true:

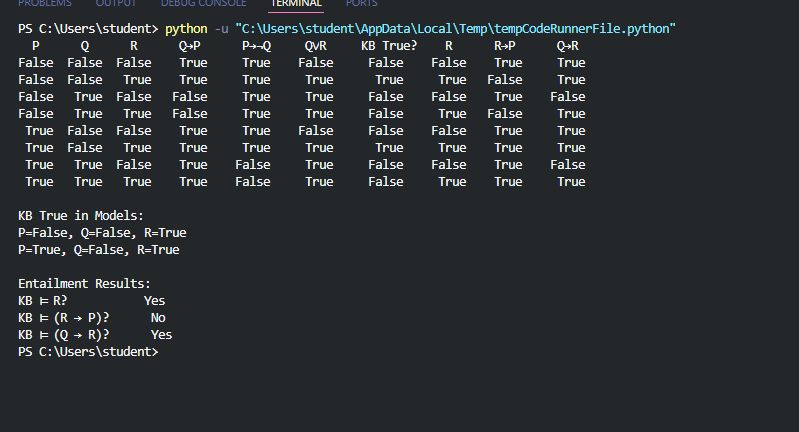
print(f"P={m[0]}, Q={m[1]}, R={m[2]}")

print("\nEntailment Results:")

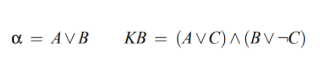
print(f"KB ⊨ R? {'Yes' if entails(query\_R) else 'No'}")

print(f"KB ⊨ (R → P)? {'Yes' if entails(query\_R\_implies\_P) else 'No'}")

print(f"KB ⊨ (Q → R)? {'Yes' if entails(query\_Q\_implies\_R) else 'No'}")



2.



from itertools import product

def KB(A, B, C):

s1 = (A or C) # (A ∨ C)

s2 = (B or (not C)) # (B ∨ ¬C)

return s1, s2, s1 and s2

# Query α = A ∨ B

def alpha(A, B, C):

return A or B

print(f"{'A':^6} {'B':^6} {'C':^6} {'A∨C':^8} {'B∨¬C':^8} {'KB':^8} {'α=A∨B':^8} {'KB True?':^10}")

models\_true = []

# Generate truth table

for A, B, C in product([False, True], repeat=3):

s1, s2, kb\_true = KB(A, B, C)

a\_val = alpha(A, B, C)

if kb\_true:

models\_true.append((A, B, C))

print(f"{str(A):^6} {str(B):^6} {str(C):^6} {str(s1):^8} {str(s2):^8} {str(kb\_true):^8} {str(a\_val):^8} {str(kb\_true):^10}")

def entails():

for A, B, C in models\_true:

if not alpha(A, B, C):

return False

return True

print("\nKB True in Models:")

for m in models\_true:

print(f"A={m[0]}, B={m[1]}, C={m[2]}")

print("\nEntailment Result:")

print(f"KB ⊨ α (A ∨ B)? {'Yes' if entails() else 'No'}")

