```python
import itertools

def pl_true(sentence, model):
    A = model.get('A', False)
    B = model.get('B', False)
    C = model.get('C', False)

    if sentence == "A or B":
        return A or B
    elif sentence == "(A or C) and (B or not C)":
        return (A or C) and (B or not C)
    return False

def tt_entails(kb, alpha):
    symbols = ['A', 'B', 'C']
    return tt_check_all(kb, alpha, symbols, {})

def tt_check_all(kb, alpha, symbols, model):
    if not symbols:
        if pl_true(kb, model):
            return pl_true(alpha, model)
        else:
            return True
    else:
        p = symbols[0]
        rest = symbols[1:]

        model_true = model.copy()
        model_false = model.copy()
        model_true[p] = True
        model_false[p] = False

        return (tt_check_all(kb, alpha, rest, model_true) and
                tt_check_all(kb, alpha, rest, model_false))

kb = "(A or C) and (B or not C)"
alpha = "A or B"

result = tt_entails(kb, alpha)
print(f"KB entails α: {result}")

def generate_truth_table():
    print(f"{'A':<10}{'B':<10}{'C':<10}{'A∨C':<10}{'B∨¬C':<10}{'KB':<10}{'α (A∨B)':<10}")
    print("-" * 70)

    for A, B, C in itertools.product([False, True], repeat=3):
        A_or_C = A or C
        B_or_not_C = B or not C
        KB = (A or C) and (B or not C)
        alpha = A or B

        row = f"{str(A):<10}{str(B):<10}{str(C):<10}{str(A_or_C):<10}{str(B_or_not_C):<10}{str(KB):<10}{str(alpha):<10}"

        if KB and alpha:
            print(row)
            print("-" * len(row))
        else:
            print(row)

generate_truth_table()
```

```
KB entails α: True
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
-----------------------------------------------------------------
```

```
import itertools

def pl_true(sentence, model):
    A = model.get('A', False)
    B = model.get('B', False)
    C = model.get('C', False)

    if sentence == "A or B":
        return A or B
    elif sentence == "(A or C) and (B or not C)":
        return (A or C) and (B or not C)
    return False

def tt_entails(kb, alpha):
    symbols = ['A', 'B', 'C']
    return tt_check_all(kb, alpha, symbols, {})

def tt_check_all(kb, alpha, symbols, model):
    if not symbols:
        if pl_true(kb, model):
            return pl_true(alpha, model)
        else:
            return True
    else:
        p = symbols[0]
        rest = symbols[1:]

        model_true = model.copy()
        model_false = model.copy()
        model_true[p] = True
        model_false[p] = False

        return (tt_check_all(kb, alpha, rest, model_true) and
                tt_check_all(kb, alpha, rest, model_false))

kb = "(A or C) and (B or not C)"
alpha = "A or B"

result = tt_entails(kb, alpha)
print(f"KB entails α: {result}")

def generate_truth_table():
    print(f"{'A':<10}{'B':<10}{'C':<10}{'A∨C':<10}{'B∨¬C':<10}{'KB':<10}{'α (A∨B)':<10}")
    print("-" * 70)

    for A, B, C in itertools.product([False, True], repeat=3):
        A_or_C = A or C
        B_or_not_C = B or not C
        KB = (A or C) and (B or not C)
        alpha = A or B

        row = f"{str(A):<10}{str(B):<10}{str(C):<10}{str(A_or_C):<10}{str(B_or_not_C):<10}{str(KB):<10}{str(alpha):<10}"

        if KB and alpha:
            print(row)
            print("-" * len(row))
        else:
            print(row)

generate_truth_table()
```

```
KB entails α: True
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
----------------------------------------------------------------------
```