**Code:**

from copy import deepcopy

from re import sub

O\_REL = ['A', 'B', 'C', 'D']

FD = [

['AB', 'B'],

['B', 'C'],

['C', 'D'],]

NEW\_REL = [

['A', 'B'],

['B', 'C'],

['C', 'D'],]

def in\_count(\_itr: list[str], elem: str):

return sum([1 for i in \_itr if elem in i])

def initialize(old\_rel, new\_rel):

table: list[list[str]] = [["" for \_ in old\_rel] for \_ in new\_rel]

for row\_index, row in enumerate(table):

for col\_index, col in enumerate(row):

if old\_rel[col\_index] in new\_rel[row\_index]:

table[row\_index][col\_index] = f"a({col\_index})"

else:

table[row\_index][col\_index] = f"b({row\_index},{col\_index})"

return table

def attrs\_intersection(attrs: list[str]) -> str:

for s in attrs:

# remove content inside parenthesis

s = sub(r'\([^)]\*\)', '', s)

s = s.replace(" ", "")

return 'a' if all([True if 'a' in attr else False for attr in attrs]) else 'b'

def get\_lhs\_values(dep, table, old\_rel):

check\_lhs: list[str] = []

compound\_attr: list[list[str]] = []

for attr in dep[0]:

compound\_attr.append([row[old\_rel.index(attr)] for row in table])

for i, c\_attr in enumerate(map(list, zip(\*compound\_attr))):

check\_lhs.append(attrs\_intersection(c\_attr))

# at this point we have a comparison list, that we can use to check for the step 2 conditions

return check\_lhs

def print\_table(table: list[list[str]]):

pretty\_table = ""

for row in table:

pretty\_table += "|"

for columns in row:

pretty\_table += f"{columns}\t|"

pretty\_table += "\n"

length = len(pretty\_table.split("\n")[0].expandtabs())

print("\_" \* length)

print(pretty\_table)

def woolman\_algorithm\_test(old\_rel: list[str], new\_rel:list[list[str]], fd: list[list[str]]) -> bool:

# Step 1: Initialize

table = initialize(old\_rel, new\_rel)

print("Initial Table: ")

print\_table(table)

while True:

compare\_table = deepcopy(table)

for dep in fd:

print(f"Checking FD: {dep[0]} -> {dep[1]}")

check\_lhs = get\_lhs\_values(dep, table, old\_rel)

rows\_to\_change = [i for i, x in enumerate(check\_lhs) if x == "a"]

columns\_to\_change = [old\_rel.index(attr) for attr in dep[1]]

column\_values = ["X" for \_ in columns\_to\_change]

if check\_lhs.count("a") <= 1:

print("No common attributes found")

continue

for column\_index in columns\_to\_change:

test\_column = [row[column\_index] for row in table if table.index(row) in rows\_to\_change]

# Find if there is "A" in the columns we wish to update, else get the first "B" value

if in\_count(test\_column, "a") >= 1:

column\_values[columns\_to\_change.index(column\_index)] = f"a({column\_index})"

else:

column\_values[columns\_to\_change.index(column\_index)] = test\_column[0]

# Update

for row\_index, row in enumerate(table):

for col\_index, col in enumerate(row):

if col\_index in columns\_to\_change and row\_index in rows\_to\_change:

table[row\_index][col\_index] = column\_values[columns\_to\_change.index(col\_index)]

# Print Table

print\_table(table)

if compare\_table == table:

for row in table:

if in\_count(row, "a") == len(row):

print(f"Found a row with all 'a' values. Row: {table.index(row) + 1}")

return True

print("No more changes possible.")

return False

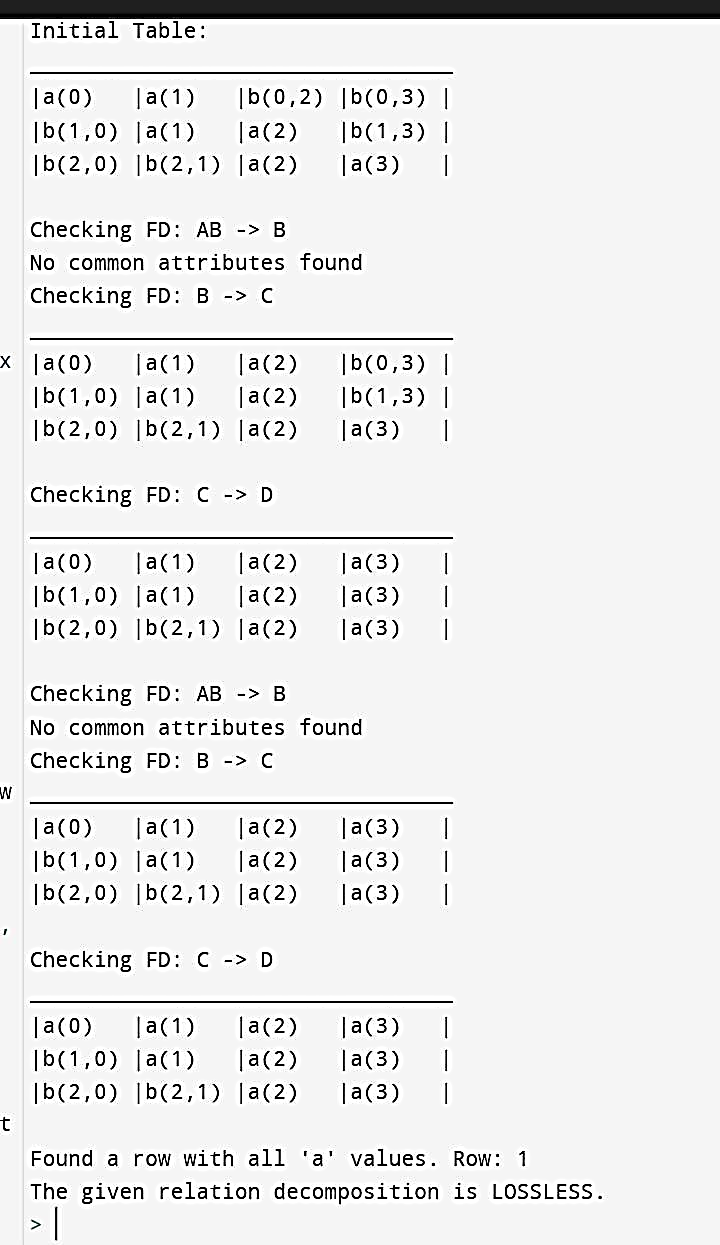
if \_\_name\_\_ == '\_\_main\_\_':

if woolman\_algorithm\_test(O\_REL, NEW\_REL, FD):

print("The given relation decomposition is LOSSLESS.")

else:

print("The given relation decomposition is LOSSY.**")**

**Output:**