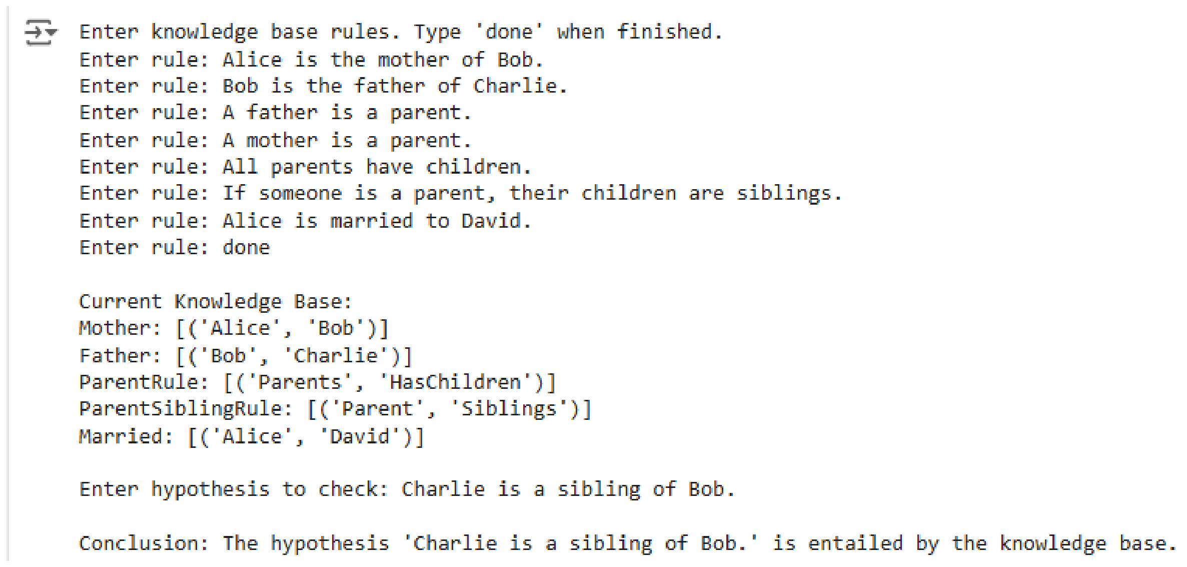
LAB-7 - Entailment Using Literals

Code:

```
import re

# Helper function to parse user input into logical predicates
def parse_input(input_sentence, knowledge_base):
    # Convert the sentence to lowercase for consistency
    input_sentence = input_sentence.lower()

    # Match patterns for predicates and facts (e.g., 'X is the mother of Y' or 'X is married to Y')
    # Fact or Rule: "X is the mother of Y"
    mother_match = re.match(r"(\w+) is the mother of (\w+)", input_sentence)
    # Fact or Rule: "X is the father of Y"
    father_match = re.match(r"(\w+) is the father of (\w+)", input_sentence)
    # General rule: "All X have children"
    parent_match = re.match(r"all (\w+) have children", input_sentence)
    # Rule for parent-child relation and siblings
    parent_rule_match = re.match(r"if someone is a parent, their children are siblings",
input_sentence)
    # General fact: "X is married to Y"
    married_match = re.match(r"(\w+) is married to (\w+)", input_sentence)

    # Parsing rules and facts
    if mother_match:
        mother, child = mother_match.groups()
        # Add the mother-child relationship to knowledge base
        knowledge_base["Mother"].append((mother.capitalize(), child.capitalize()))
    elif father_match:
        father, child = father_match.groups()
        # Add the father-child relationship to knowledge base
        knowledge_base["Father"].append((father.capitalize(), child.capitalize()))
    elif parent_match:
        parent = parent_match.group(1)
        # Rule: All X are parents with children
        knowledge_base["ParentRule"].append((parent.capitalize(), "HasChildren"))
    elif parent_rule_match:
        # General rule: If someone is a parent, their children are siblings
        knowledge_base["ParentSiblingRule"].append(("Parent", "Siblings"))
    elif married_match:
        spouse1, spouse2 = married_match.groups()
        # Add the married relationship to knowledge base
        knowledge_base["Married"].append((spouse1.capitalize(), spouse2.capitalize()))

# Function to check if two children are siblings
def are_siblings(child1, child2, knowledge_base):
    # Check if both children share the same parent
    parents = set()
    for mother, child in knowledge_base["Mother"]:
        if child == child1:
            parents.add(mother)
        if child == child2:
            parents.add(mother)
    for father, child in knowledge_base["Father"]:
        if child == child1:
            parents.add(father)
        if child == child2:
            parents.add(father)
    return len(parents) > 1  # If both children share a parent, they are siblings

# Function to check the hypothesis "Charlie is a sibling of Bob"
def check_hypothesis(hypothesis, knowledge_base):
    # Parse the hypothesis
    hyp_match = re.match(r"(\w+) is a sibling of (\w+)", hypothesis.lower())

    if hyp_match:
        child1, child2 = hyp_match.groups()
        # Check if the children are siblings
        if are_siblings(child1.capitalize(), child2.capitalize(), knowledge_base):
            return True
    return False

# Main function for user input and entailment reasoning
def main():
    # Create an empty knowledge base
    knowledge_base = {
        "Mother": [],
        "Father": [],
        "ParentRule": [],
        "ParentSiblingRule": [],
        "Married": []
    }

    print("Enter knowledge base rules. Type 'done' when finished.")

    # Allow the user to input knowledge base facts, rules, or actions
    while True:
        user_input = input("Enter fact/rule/action: ").strip()
        if user_input.lower() == "done":
            break
        parse_input(user_input, knowledge_base)

    # Print the current knowledge base
    print("\nCurrent Knowledge Base:")
    for category, items in knowledge_base.items():
        print(f"{category}: {items}")

    # Ask for the hypothesis (the statement to check)
    hypothesis = input("\nEnter hypothesis to check : ").strip()

    # Check if the hypothesis is entailed
    if check_hypothesis(hypothesis, knowledge_base):
        print(f"\nConclusion: The hypothesis '{hypothesis}' is entailed by the knowledge base.")
    else:
        print(f"\nConclusion: The hypothesis '{hypothesis}' is NOT entailed by the knowledge
base.")

# Run the program
main()
```

Output:



```
Enter knowledge base rules. Type 'done' when finished.
Enter rule: Alice is the mother of Bob.
Enter rule: Bob is the father of Charlie.
Enter rule: A father is a parent.
Enter rule: A mother is a parent.
Enter rule: All parents have children.
Enter rule: If someone is a parent, their children are siblings.
Enter rule: Alice is married to David.
Enter rule: done

Current Knowledge Base:
Mother: [('Alice', 'Bob')]
Father: [('Bob', 'Charlie')]
ParentRule: [('Parents', 'HasChildren')]
ParentSiblingRule: [('Parent', 'Siblings')]
Married: [('Alice', 'David')]

Enter hypothesis to check: Charlie is a sibling of Bob.

Conclusion: The hypothesis 'Charlie is a sibling of Bob.' is entailed by the knowledge base.
```