Experiment no: 10

```
class Rule:
    def _init_(self, antecedent, consequent):
        self.antecedent = antecedent
        self.consequent = consequent

class KnowledgeBase:
    def _init_(self):
        self.rules = []

    def add_rule(self, antecedent, consequent):
        rule = Rule(antecedent, consequent)
        self.rules.append(rule)

    def forward_chaining(self, initial_facts):
        inferred_facts = set(initial_facts)
        while True:
            new_facts = set()
            for rule in self.rules:
                if all(antecedent in inferred_facts for antecedent in rule.antecedent):
                    new_facts.add(rule.consequent)
            if not new_facts:
                break
            inferred_facts |= new_facts
        return inferred_facts

    def backward_chaining(self, goal, initial_facts):
        inferred_facts = set(initial_facts)
        agenda = [goal]
        while agenda:
            current_goal = agenda.pop()
            if current_goal in inferred_facts:
                continue
            is_proved = False
            for rule in self.rules:
                if rule.consequent == current_goal:
                    if all(antecedent in inferred_facts for antecedent in rule.antecedent):
                        is_proved = True
                        break
                    else:
                        for antecedent in rule.antecedent:
                            agenda.append(antecedent)
            if is_proved:
                inferred_facts.add(current_goal)
        return inferred_facts

# Example usage:
kb = KnowledgeBase()

# Add rules to the knowledge base
kb.add_rule(['flu', 'cough'], 'fever')
```

```
kb.add_rule(['fever'], 'sickness')

# Forward chaining example
initial_facts = ['flu', 'cough']
print("Forward chaining result:")
print(kb.forward_chaining(initial_facts))

# Backward chaining example
goal = 'sickness'
print("\nBackward chaining result:")
print(kb.backward_chaining(goal, initial_facts))
```

Output:

```
Forward chaining result:
{'fever', 'sickness'}

Backward chaining result:
{'fever', 'sickness'}
```