Forward Chaining Algorithm

```python
# Define the knowledge base (rules and facts)
knowledge_base = {
    'A': ['B', 'C'],
    'D': ['E'],
    'E': ['F'],
    'G': ['H', 'I'],
    'H': ['J'],
    'I': ['K'],
    'F': ['L']
}

# Define the initial facts
facts = ['A', 'D', 'G']

# Define the goal
goal = 'L'

# Implement the forward chaining algorithm
update_facts = list(facts)
agenda = list(facts)
while len(agenda) > 0:
    current_fact = agenda.pop(0)
    if current_fact == goal:
        print("Goal reached!")
        break
    elif current_fact in knowledge_base:
        new_facts = knowledge_base[current_fact]
        for x in new_facts :
            if x not in facts:
                update_facts.append(x)
                agenda.append(x)
if goal not in update_facts:
    print("Goalnot  reached!")
```

```python
# Define the knowledge base (rules and facts)
knowledge_base = {
    'A': ['B', 'C'],
    'D': ['E'],
    'E': ['F'],
    'G': ['H', 'I'],
    'H': ['J'],
    'I': ['K'],
    'F': ['L']
}

# Define the initial facts
facts = ['A', 'D', 'G']

# Define the goal
goal = 'L'

# Implement the forward chaining algorithm
update_facts = list(facts)
agenda = list(facts)
while len(agenda) > 0:
    current_fact = agenda.pop(0)
    if current_fact == goal:
        print("Goal reached!")
        break
    elif current_fact in knowledge_base:
        new_facts = knowledge_base[current_fact]
        for x in new_facts :
            if x not in facts:
                update_facts.append(x)
                agenda.append(x)
if goal not in update_facts:
    print("Goalnot  reached!")
```

Backward Chaining Algorithm

```python
# Define the knowledge base (rules and facts)
knowledge_base = {
    'A': ['B', 'C'],
    'D': ['E'],
    'E': ['F'],
    'G': ['H', 'I'],
    'H': ['J'],
    'I': ['K'],
    'F': ['L'],
    'J': ['L'],
    'K': ['L']
}

# Define the goal
goal = 'L'

# Implement the backward chaining algorithm
def backward_chaining(kb, goal, facts):
    if goal in facts:
        return True
    else:
        for fact in facts:
            if fact in kb:
                for rule in kb[fact]:
                    if backward_chaining(kb, rule, facts):
                        return True
        return False

# Test the algorithm
if backward_chaining(knowledge_base, goal, set(['A', 'D', 'G'])):
    print("Goal reached!")
else:
    print("Goal not reachable.")
```

```python
# Define the knowledge base (rules and facts)
knowledge_base = [
    (['A'], 'B'),
    (['A', 'C'], 'D'),
    (['B'], 'E'),
    (['D', 'E'], 'F'),
    (['F'], 'G')
]

# Define the goal
goal = 'G'

# Implement the backward chaining algorithm
def backward_chaining(kb, goal, facts):
    if goal in facts:
        return True
    for rule in kb:
        if rule[1] == goal:
            if all(backward_chaining(kb, premise, facts) for premise in rule[0]):
                return True
    return False

# Test the algorithm
result = backward_chaining(knowledge_base, goal, set(['A', 'C']))
print(result)
```

```python
# Define the knowledge base (rules and facts)
knowledge_base = [
    (['A'], 'B'),
    (['A', 'C'], 'D'),
    (['B'], 'E'),
    (['D', 'E'], 'F'),
    (['F'], 'G')
]

# Define the goal
goal = 'G'

# Implement the backward chaining algorithm
def backward_chaining(kb, goal, facts):
    if goal in facts:
        return True
    for rule in kb:
        if rule[1] == goal:
            if all(backward_chaining(kb, premise, facts) for premise in rule[0]):
                return True
    return False

# Test the algorithm
result = backward_chaining(knowledge_base, goal, set(['A', 'C']))
print(result)
```