```
def backward_chaining(rules, facts, goal, visited=None):
  if visited is None:
     visited = set()


  # If goal already known as fact
  if goal in facts:
     return True


  # Avoid infinite loops on cyclic rules
  if goal in visited:
     return False


  visited.add(goal)


  # Check each rule: head is conclusion, body are premises
  for rule in rules:
     head = rule[0]
     body = rule[1:]


     # If rule concludes the goal
     if head == goal:
        # Check if all premises can be proven
        if all(backward_chaining(rules, facts, subgoal, visited) for subgoal in body):
           facts.add(goal)  # Add inferred fact
           return True
  return False
```

```
# ---------------------------------
# Define the knowledge base (rules)
# Format: [conclusion, premise1, premise2, ...]
rules = [
  ['a'],          # fact a
  ['b'],          # fact b
  ['c', 'a', 'b'],  # c  a  b
  ['d', 'b'],       # d  b
]


# Initial facts
facts = set(['a', 'b'])


# Test goals
goals = ['f', 'd']


for goal in goals:
  result = backward_chaining(rules, facts.copy(), goal)
  if result:
      print(f"The goal '{goal}' can be achieved.")
  else:
      print(f"The goal '{goal}' cannot be achieved.")
```