```
class ForwardChaining:
    def __init__(self, facts, rules):
        self.facts = facts
        self.rules = rules
    def apply_rule(self, rule):
        condition, conclusion = rule
        if all(cond in self.facts for cond in condition):
            if conclusion not in self.facts:
                self.facts.add(conclusion)
                print(f"Fact inferred: {conclusion}")
                return True
        return False

    def forward_chain(self):
        """Perform forward chaining until no new facts can be inferred."""
        while True:
            new_fact_inferred = False
            for rule in self.rules:
                if self.apply_rule(rule):
                    new_fact_inferred = True
            if not new_fact_inferred:
                break



initial_facts = {"has fur", "has legs", "gives milk"}



rules = [
    ({"has fur", "gives milk"}, "is a mammal"),
    ({"has fur", "has legs"}, "is an animal"),
    ({"gives milk", "has legs"}, "is a mammal"),
```

]

```
fc = ForwardChaining(initial_facts, rules)
fc.forward_chain()
print("\nFinal facts:", fc.facts)
```

Output

```
Fact inferred: is a mammal
Fact inferred: is an animal

Final facts: {'gives milk', 'is an animal', 'has legs', 'has fur', 'is a mammal'}

=== Code Execution Successful ===
```