```
# ================================================================
#  Forward Chaining
# ================================================================

# Knowledge Base: list of tuples (premises..., conclusion)
rules = [
   ("has_feathers", "has_wings", "is_bird"),
   ("is_bird", "can_fly", "is_flying_bird"),
   ("lays_eggs", "has_feathers", "is_bird"),
]

# Working Memory: known facts
working_memory = ["has_feathers", "has_wings", "can_fly"]

# Final goal
goal = "is_flying_bird"

# Track inferred facts and fired rules
proved = []
fired_rules = []

print(f"Initial working memory : {working_memory}")
print(f"Final goal             : '{goal}'")

# ================================================================
#  Main Loop
# ================================================================
changed = True
while changed:
   changed = False

   for rule in rules:
      premises = list(rule[:-1])   # IF parts
      conclusion = rule[-1]        # THEN part

      if all(p in working_memory for p in premises) and conclusion not in working_memory:
         print(f" Rule fired: {premises} → '{conclusion}'")
         working_memory.append(conclusion)
         proved.append(conclusion)
         fired_rules.append(rule)
```

```
            print(f" Added '{conclusion}' to working memory.")
            changed = True

            if conclusion == goal:
                break

        if goal in working_memory:
            break

# ============================================================
#  Final Report
# ============================================================
print(f"\n Result: '{goal}' → {'PROVED' if goal in working_memory else 'FAILED'}")
print(f" Final working memory : {working_memory}")
print(f" Proved facts         : {proved}")
```

Execution

```
Initial working memory : ['has_feathers', 'has_wings', 'can_fly']
Final goal             : 'is_flying_bird'
 Rule fired: ['has_feathers', 'has_wings'] → 'is_bird'
 Added 'is_bird' to working memory.
 Rule fired: ['is_bird', 'can_fly'] → 'is_flying_bird'
 Added 'is_flying_bird' to working memory.


 Result: 'is_flying_bird' → PROVED
 Final working memory : ['has_feathers', 'has_wings', 'can_fly', 'is_bird', 'is_flying_bird']
 Proved facts         : ['is_bird', 'is_flying_bird']
```