Date:31.10.25

TASK:12

MULTI-AGENT DECISION SYSTEM

CO4, CO5 S3

PROBLEM STATEMENT :

Design a multi-agent decision support system for emergency alerts based on weather forecasts, using logical knowledge representation and reasoning under uncertainty. Agents must collaborate, represent rules & facts, and reason about uncertain forecasts to make decisions (e.g., issue storm warnings, allocate resources).

AIM :

To build a multi-agent decision system where agents represent and reason with knowledge (facts & rules) and handle uncertainty (via probabilistic/fuzzy logic or resolution) to make decisions derived from weather-forecast inputs.

OBJECTIVE :

1. Define agent architecture: e.g., DataAgent (collects forecast), ForecastAgent (processes forecast), DecisionAgent (applies rules), AlertAgent (issues alerts).
2. Represent knowledge as facts and rules using logical representation (first-order logic or rule-based engine).
3. Incorporate uncertainty: represent confidence in forecast, assess conflicting inputs, reason under uncertainty using resolution / probabilistic logic.
4. Implement communication & coordination between agents (exchange facts, update beliefs).
5. Evaluate system performance: correct alerts (true positives/negatives), timeliness, robustness under uncertain/contradictory data.

DESCRIPTION :

Agents and roles:

- DataAgent : receives processed forecast sequence from module 1, sends facts (e.g., “Forecast(Rainfall>50mm, RegionA, TimeT)=0.8”).
- ForecastAgent : analyses forecast data, derives additional facts (e.g., “HighRiskStorm(RegionA, TimeT)”).
- DecisionAgent : uses logical rules (if-then) and reasoning under uncertainty to decide actions (e.g., issue “StormWarning(RegionA, TimeT)”).
- AlertAgent : executes the decision by sending alerts or triggering resource allocation.

Knowledge representation:

- Facts: e.g., HighRiskStorm(RegionA, 15:00), Confidence (HighRiskStorm(RegionA, 15:00)) = 0.85. Logical representation of facts + rules.
- **Uncertainty reasoning**: Agents attach confidence to facts; decision logic may apply thresholds (e.g., if confidence > 0.7). Conflicting facts (one agent says high risk, another says low risk) resolved via resolution or belief revision. Framework such as Independent Choice Logic (ICL) can model choices under uncertainty.
- **Communication**: Agents exchange messages: e.g., DataAgent → ForecastAgent: fact message; ForecastAgent → DecisionAgent: derived fact with confidence; DecisionAgent → AlertAgent: decision message.
- **Decision process**: DecisionAgent collates facts, applies logical rules, performs reasoning under uncertainty (e.g., uses rough set logic or probabilistic logic) to conclude whether to send alert. Rough logic for MAS uses formal logic for reasoning under incomplete information.
- **Evaluation**: Measure correct alerts, false positives/negatives, average decision time, how often uncertainty reasoning prevented wrong alerts.

ALGORITHM :

 Define agent architecture and roles: e.g., DataAgent, ForecastAgent, DecisionAgent, AlertAgent.

 Represent knowledge base: define facts (e.g., forecast values with confidence) and rules (IF-THEN logic) for decision making (logical knowledge representation).

 DataAgent receives input (forecast sequences from prediction module) and sends facts to ForecastAgent (e.g., Rainfall > X, Confidence = C).

 ForecastAgent processes those facts, derives higher-level facts or risk indicators (e.g., HighRiskStorm(region, time) with associated probability/confidence).

 DecisionAgent receives derived facts, then:

- Filters facts whose confidence meets a threshold.
- Applies rule base: for each applicable rule, checks conditions against knowledge base.
- Uses reasoning under uncertainty/resolution: if conflicting facts exist, apply belief revision or fuzzy/probabilistic logic to resolve.
- Determines decision(s) (e.g., IssueAlert(region, time), AllocateResources(region, time)).

 AlertAgent executes decisions: logs action, sends alert, updates system state (e.g., mark region as warned).

 Agents communicate/co-ordinate: e.g., DecisionAgent may query DataAgent or ForecastAgent for additional facts; agents update their belief/fact base as new information arrives.

 Handle uncertainty and resolution: when facts have confidence levels, or contradictory facts exist, apply resolution methods (e.g., belief revision, fuzzy inference, probabilistic reasoning) to decide which facts to accept and which rules to trigger.

 Monitor performance: log actual outcomes (did an event occur?), compare decisions with outcomes, compute decision accuracy (true positives, false alarms), average confidence at decision time, timeliness.

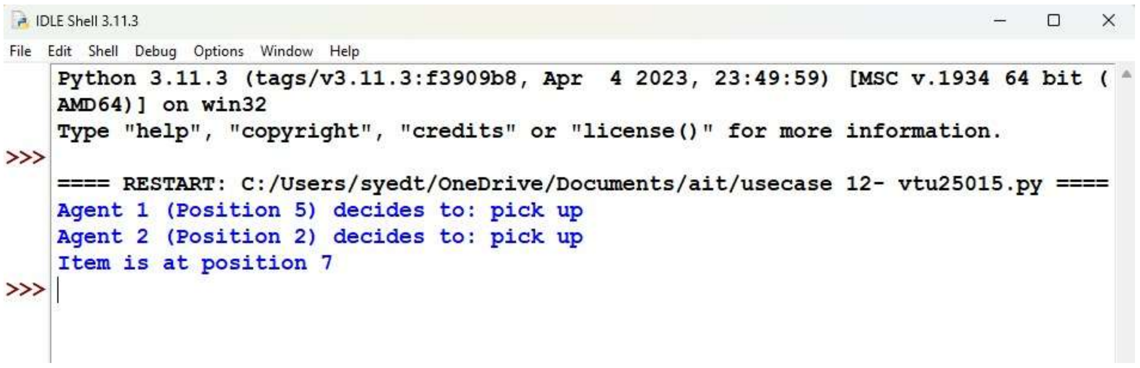
 Feedback loop: use decision outcome data to update agent knowledge/rules or adjust thresholds for future decisions.

 Document and analyse: how logical knowledge representation enabled decision making, how uncertainty reasoning improved (or not) decision quality, limitations and next-steps.

PROGRAM :

```
import random

class Agent:

    def _init_(self, id, position):

        self.id = id

        self.position = position

    def decide(self, item_position):

        distance = abs(self.position - item_position)

        if distance <= 5:

            action = "pick up"

        else:

            action = "ignore"

        return action

agent1 = Agent(id=1, position=random.randint(0, 10))

agent2 = Agent(id=2, position=random.randint(0, 10))

item_position = random.randint(0, 10)

action1 = agent1.decide(item_position)

action2 = agent2.decide(item_position)

print(f"Agent 1 (Position {agent1.position}) decides to: {action1}")

print(f"Agent 2 (Position {agent2.position}) decides to: {action2}")

print(f"Item is at position {item_position}")
```

OUTPUT



```
Python 3.11.3 (tags/v3.11.3:f3909b8, Apr  4 2023, 23:49:59) [MSC v.1934 64 bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
==== RESTART: C:/Users/syedt/OneDrive/Documents/ait/usecase 12- vtu25015.py ====
Agent 1 (Position 5) decides to: pick up
Agent 2 (Position 2) decides to: pick up
Item is at position 7
>>>
```

CONCLUSION :

This use-case demonstrates how logical knowledge representation (facts & rules) and reasoning under uncertainty can be applied in a multi-agent decision system context. The architecture shows agent coordination, rule-based logic, and uncertainty handling, matching the “apply” (K3) level. You will likely observe that the inclusion of uncertainty reasoning improves decision robustness (fewer false alerts) but adds complexity in belief/reasoning modules. The system provides a structured way to support real-time decision processes based on forecasts.