



The City College
of New York

CSC 36000: Modern Distributed Computing *with AI Agents*

By Saptarashmi Bandyopadhyay

Email: sbandyopadhyay@ccny.cuny.edu

Assistant Professor of Computer Science

City College of New York and Graduate Center at City University of New York

November 5, 2025 CSC 36000

Today's Lecture

Agent Ontologies

Distributed Agent Communication

Communication vs. Cooperation

Agent Ontologies

Saptarashmi Bandyopadhyay

Agent Ontologies: How do Agents Communicate?

An ontology can be simply defined as a formal, explicit "shared dictionary" or "shared vocabulary" that AI agents use to understand one another.

- **Classes/Concepts:** The types of things that exist in the domain. For example, Vehicle, TrafficLight, Pedestrian, and RoadSegment.
- **Properties/Attributes:** The data associated with those classes. A Vehicle class would have properties like currentSpeed, position, and heading.
- **Relationships:** The rules governing how classes interact. An ontology would define relationships like Vehicle is_approaching TrafficLight
- **Logical Axioms:** Rules and constraints that allow for reasoning. For instance, an axiom might state that "All Vehicles are a type of DynamicObject,"

Semantic Ambiguity

The Problem: A thermostat agent (Agent A) broadcasts a message (temp: 70). A user's phone (Agent B) receives this. Agent B cannot know what this means. Is it 70° Fahrenheit or Celsius? Is it a current sensor reading or a desired target temperature?



The Solution: An ontology provides interoperability. By agreeing to use a "Smart Home Ontology," the message becomes unambiguous: (class: "SensorReading", property: "temperature", value: "70", unit: "fahrenheit")

This allows heterogeneous distributed agents to accomplish coordination!

Distributed Agent Communication

Saptarashmi Bandyopadhyay

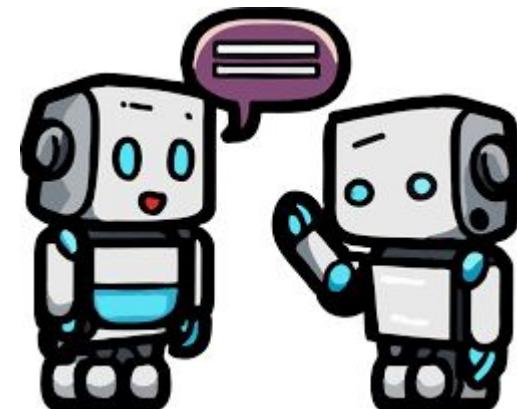
Agent Communication

How do AI Agents signal *intent* between each other?

Message Passing and Shared Memory describe how to move data between processes.

Agent Communication Languages (ACLs) are a higher-level concept: they are about moving meaning and intent.

Examples: FIPA-ACL (Foundation for Intelligent Physical Agents) and KQML (Knowledge Query and Manipulation Language)



How do AI Agents convey intent?

The key component of an ACL message is the *performative*. This is the message's intent, its purpose, or its verb. It is the only mandatory part of a FIPA-ACL message. Examples :

- inform: To tell the receiver that a fact is true.
- request: To ask the receiver to perform an action.
- cfp (Call for Proposals): To ask for proposals to perform an action.
- propose: To offer to perform an action, usually in response to a cfp.
- accept-proposal: To agree to a proposal.
- failure: To inform that a requested action has failed.

Message Structure

A FIPA-ACL message is a structured set of key-value pairs, which enables parsing and understanding. A complete message includes :

- (performative : The intent, e.g., request)
- :sender : The name of the agent sending the message
- :receiver : The intended recipient(s)
- :content : The data or "what" of the message
- :language : The format of the content (e.g., JSON, Prolog)
- :ontology : The name of the "dictionary" used for the content

Example: Interactive Scheduling

Let Agent_A (Professor's assistant) and Agent_B (Student's assistant) try to schedule a 30-minute meeting.

A naive or "brittle" interaction would be non-robust:

- A -> B: (performative: request, content: "schedule_meeting(Oct 31, 10:00)")
- B -> A: (performative: failure, content: "slot_unavailable") The conversation fails, and the agents cannot complete their task.

A "good" or robust interaction uses a formal Interaction Protocol, such as the FIPA-Contract-Net Protocol. This protocol defines a structured, multi-step negotiation.

Example: Interactive Scheduling (Call for Proposals)

```
(performative: cfp
  :sender:      Agent_A
  :receiver:    Agent_B
  :protocol:    fipa-contract-net // Declares the "rules" of
this conversation
  :ontology:    "google-calendar-ontology" // Declares the
"dictionary"
  :content:     "Find 30-min slot for 'CSC36000 Sync' next week"
)
```

Example: Interactive Scheduling (Proposals)

```
(performativ: propose
:sender:      Agent_B
:receiver:    Agent_A
:content:     "slot1(date: 2025-10-28, time: 10:00)"
)

(performativ: propose
:sender:      Agent_B
:receiver:    Agent_A
:content:     "slot2(date: 2025-10-30, time: 14:30)"
)
```

Example: Interactive Scheduling (Acceptance)

```
(performativ: accept-proposal
 :sender: Agent_A
 :receiver: Agent_B
 :content: "slot1(date: 2025-10-28, time: 10:00)"
)
```

Example: Interactive Scheduling (Confirmation)

```
(performative: inform
:sender:      Agent_B
:receiver:    Agent_A
:content:     "Confirmed: 'CSC36000 Sync' (date: 2025-10-28, time: 10:00)"
)
```

This coordinated approach is much more scalable than trying to schedule ad-hoc. In a distributed agent setting with potentially hundreds of agents, this helps to avoid deadlocks that would otherwise be inevitable.

Communication vs Cooperation

Cooperation

Distributed MARL algorithms like Value Decomposition Networks (VDN) and QMIX. These are mechanisms for Cooperation:

- Cooperative agents are a team. They share a common goal and a joint reward function. They succeed or fail together.

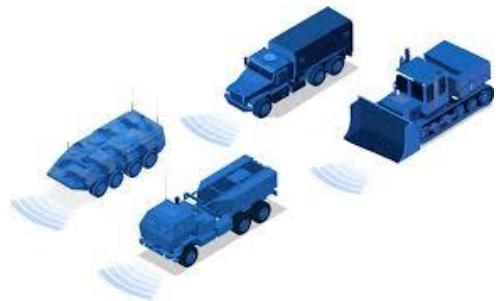


Coordination

Coordinated agents are not necessarily a team. They are autonomous, self-interested agents who may have different or even conflicting goals.

However, their actions have interdependencies: they operate in a shared environment and can interfere with one another.

Coordination is the process of managing these dependencies to avoid conflict or to enable agents to achieve their individual goals



The Key Difference

Cooperation is about aligning goals to maximize a joint reward. Coordination is about managing shared resources or constraints among self-interested agents.

For example, a group of robots simply moving around and bouncing off each other are not cooperating, but their behavior results in an emergent coordination that achieves a system-level task.

In Distributed Computing, this comes up often. Distributed GPUs need to cooperate together to train a large model, but peer-to-peer networks in a distributed file system have nodes that need to look out for their users, which have their own interests.

Example: Distributed Sensor Networks (Cooperation)

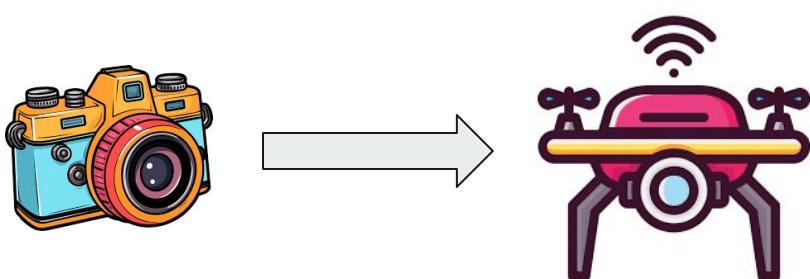
Scenario: Dozens of sensors are scattered across a forest to find a lost hiker.

The Goal: There is a single, shared goal for the entire system: "Maximize area coverage to find the hiker" or "Maximize system lifetime."

Cooperative Action: No single sensor can achieve this goal. They must cooperate:

Sensor A (a static camera) detects a possible signal. It cannot investigate.

It communicates with Sensor B (a drone), which cooperates by moving to that location to get a better look.



Example: Autonomous Traffic Control (Coordination)

Scenario: Car_A (wants to go North) and Car_B (wants to go East) arrive at an empty intersection at the same time.

The Goal: The agents' goals are not shared. Car_A's goal is "get to destination_A." Car_B's goal is "get to destination_B." Car_A is indifferent to whether Car_B ever reaches its destination.

The Problem: Their actions are interdependent. The shared resource is the physical space in the middle of the intersection. If both agents pursue their goals, they will crash.



Example: Autonomous Traffic Control (Coordination)

Coordination Mechanism: The agents are not a team, but they must coordinate.

- The intersection resource is divided into space-time blocks.
- Car_A (using FIPA-ACL) sends a request: (performativ e: request, content: "reserve(block: 5, time: 10:01:03)").
- An Intersection_Manager agent (or a decentralized protocol among the cars) receives this. It checks its state and replies: (performativ e: confirm, content: "reserved(block: 5, time: 10:01:03)").
- Car_B then requests the same block and receives: (performativ e: disconfirm, content: "conflict(block: 5, time: 10:01:03)"). Car_B must now wait or re-plan.

Further Reading: Distributed Reinforcement Learning

Dimitri P. Bertsekas

[https://web.mit.edu/dimitri
b/www/Rollout Complete
%20Book.pdf](https://web.mit.edu/dimitri/b/www/Rollout%20CompleteBook.pdf)

Questions?

Saptarashmi Bandyopadhyay