Agent and Multi-Agent Systems: architectures and reasoning

Interaction mechanisms : models and implementation

Wassila Ouerdane

14.06.2022

CentraleSupélec –SAFRAN Training

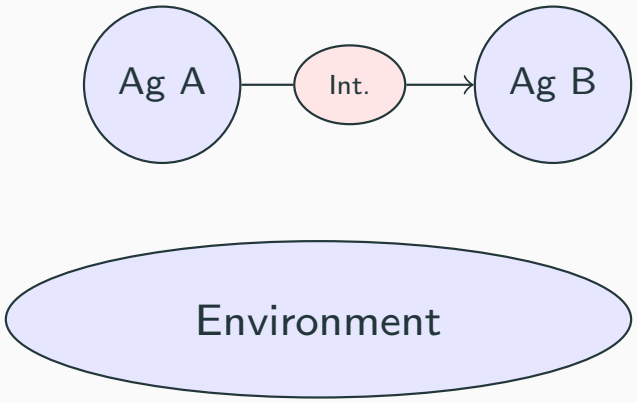
Table of contents

Interaction Mechanisms

Interaction Mechanisms

Interaction

- An interaction occurs when two or more agents are brought into a dynamic relationship through a set of reciprocal actions.
- Interactions develop out of a series of actions whose consequences in turns have an influence on the future behavior of agents.



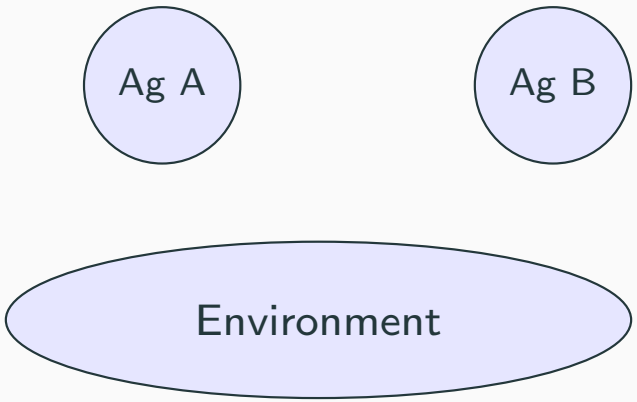
Asynchronous interactions

Problems in MAS

- Agents run asynchronously
- Method invocation is synchronous

PRS Architecture

- Actions modify the environment
- (Asynchronous) perception of the modification



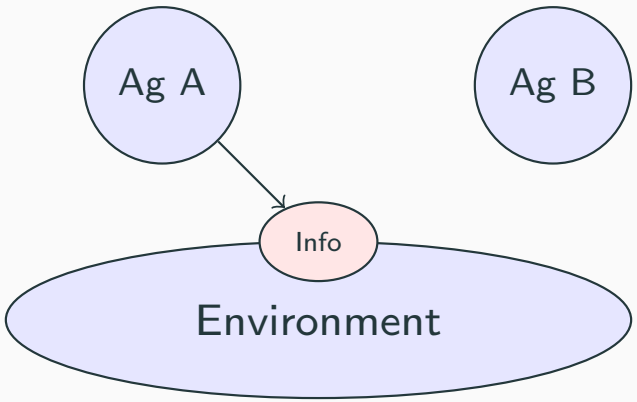
Asynchronous interactions

Problems in MAS

- Agents run asynchronously
- Method invocation is synchronous

PRS Architecture

- Actions modify the environment
- (Asynchronous) perception of the modification



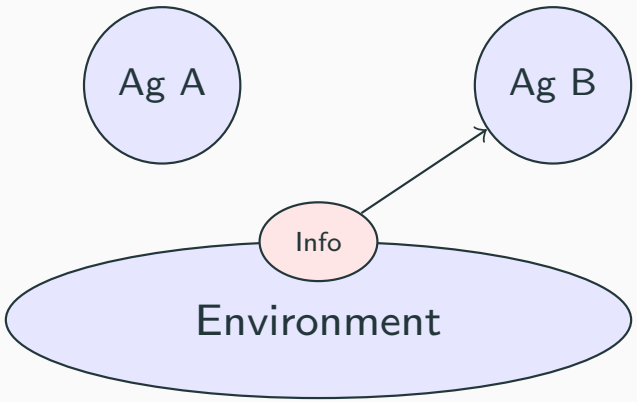
Asynchronous interactions

Problems in MAS

- Agents run asynchronously
- Method invocation is synchronous

PRS Architecture

- Actions modify the environment
- (Asynchronous) perception of the modification



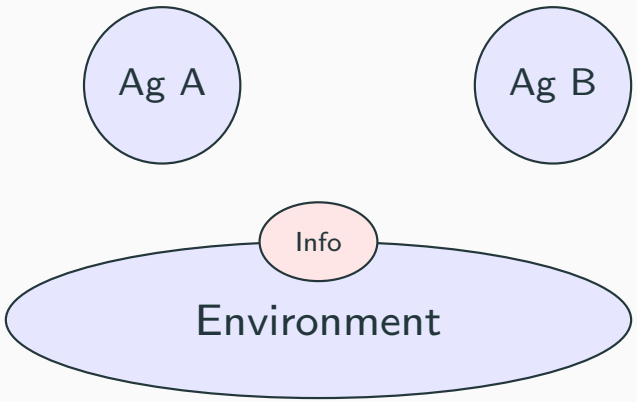
Asynchronous interactions

Problems in MAS

- Agents run asynchronously
- Method invocation is synchronous

PRS Architecture

- Actions modify the environment
- (Asynchronous) perception of the modification



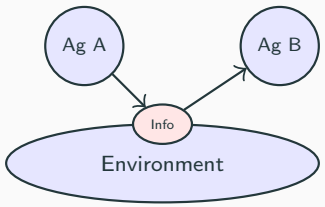
Asynchronous interactions

Problems in MAS

- Agents run asynchronously
- Method invocation is synchronous

PRS Architecture

- Actions modify the environment
- (Asynchronous) perception of the modification



Example from session 1

➜ Alice and Bob manipulate a shared variable.

The action of each agent is triggered by the fact that the other agent modified the variable

Interactions

Two sorts of interactions

- Indirect interactions

  ➜ The agents act on the environment, the interaction is a side-effect of the actions

  See Alice and Bob's example

Interactions

Two sorts of interactions

- Indirect interactions

  ➜ The agents act on the environment, the interaction is a side-effect of the actions

  See Alice and Bob's example

- Direct interactions

  ➜ The agents have explicit communication actions (message sending)

  + for cognitive agents : communication goals/intentions

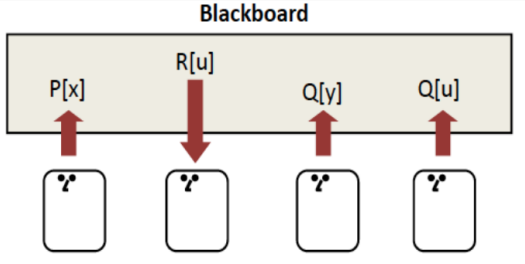
Indirect interactions

Indirect interactions

- No intention to communicate to a specific agent
- Agents interact through an intermediate entity
- This medium supplies specific interaction mechanisms and access rules
- These rules and mechanisms define agent local context and perception

Artifact-mediated interaction: Blackboard systems

- Agents access a shared artifact (stores data) that: they can observe and can modify
- Such artifact is a communication channel characterized by an intrinsically broadcast transmission.
- Specific laws regulating access to this medium
- It represents a part of agents' environment.



For more details

`https://en.wikipedia.org/wiki/Blackboard_system`

Blackboard: implementation

Environment

```
class Environment:
    blackboard = {}
    def act(self,name,value):
        self.blackboard[name] = value
    def perceive(self,name):
        if name in self.blackboard:
            return self.blackboard[name]
        return None
```

Blackboard: implementation

Agents (Alice and Bob with their preferred value)

```
1  class Agent(Thread):
2      def __init__(self, name, preferred_value, env):
3          Thread.__init__(self)
4          self.name = name
5          self.env = env
6          self.pv = preferred_value
7
8      def run(self):
9          while True:
10             self.procedural_loop()
11
12     def procedural_loop(self):
13         if self.env.perceive("v") != self.pv:
14             self.env.act("v",self.pv)
```

From Blackboard to Stigmergy

Blackboard's limitations

- Centralized mechanism $\rightarrow$ possible failures
- All agents perceive the whole environmeent!

From Blackboard to Stigmergy

Blackboard's limitations

- Centralized mechanism $\rightarrow$ possible failures
- All agents perceive the whole environmeent!

Local view on the environment

e.g.: Money model or prey-predators

- Agents are spatially situated

  $\rightarrow$ localization variables in relation to the environment, *e.g.* a grid)
- Each position in the grid is associated to a mini-blackboard

Direct interactions

Direct interactions

Principle

- Intention to communicate to a specific agent
  - Messages with sender & recipient
  - Control over the communication
- Information exchange
  - Structured content → content models and languages
  - Ontologies

Norms

- Agent Communication Language (ACL)
- Communication/Conversation rules ("protocols")

Speech Acts Theory I

Searle, 1969

Communication is an action

Communicate $\rightarrow$ change interlocutor's mental state

Speech Acts Theory I

Searle, 1969

Communication is an action

Communicate $\rightarrow$ change interlocutor's mental state

Three aspect of a speech act

- Locutionary: the act of saying

  What was said: "*Is there any salt?*"

- Illocutionary: the intention behind the speech

  What is expected to be the result: *get the salt*

- Perlocutionary: the effect of the speech

  What happened as a result, what was understood

Speech Acts Theory I

Searle, 1969

Communication is an action

Communicate $\rightarrow$ change interlocutor's mental state

Three aspect of a speech act

- Locutionary: the act of saying

  What was said: "*Is there any salt?*"
- Illocutionary: the intention behind the speech

  What is expected to be the result: *get the salt*
- Perlocutionary: the effect of the speech

  What happened as a result, what was understood

Sucessful communication

$$\text{Illocutionary} = \text{Perlocutionary}$$

Speech Acts Theory II

Searle, 1969

Illocutionary act

Performative Verb (Propositional Content)

Speech Acts Theory II

Searle, 1969

Illocutionary act

Performative Verb (Propositional Content)

Examples

Content = 'the door is closed'

- Performative = request — *please close the door*
- Performative = inform — *the door is closed*
- Performative = question — *is the door closed?*

Speech Acts Theory II

Searle, 1969

Illocutionary act

Performative Verb (Propositional Content)

Examples

Content = 'the door is closed'

- Performative = request — *please close the door*
- Performative = inform — *the door is closed*
- Performative = question — *is the door closed?*

Categories of performatives/speech acts

- Assertive (send information)
- Directive (send orders. . . or ask questions!)
- Commissive, Declarative, Expressive. . .

Agent Communication Language (ACL)

➜ Standard for messages exchanged among agents *e.g.* FIPA-ACL, KQML...

Message structure

- Sender of the message (agent ID)
- Receivers (agents IDs)
- Performative (predefined list of possible values)
- Propositional content in some content language (*e.g.* KIF)
- Conversation IDs
- Ontology

FIPA-ACL is the *de-facto* standard `http://www.fipa.org`

- FIPA: Foundation for Intelligent Physical Agents
- KQML: Knowledge Query and Manipulation Language
- KIF: Knowledge Interchange Format

FIPA-ACL standard

Example – KQML-based syntax

```
(inform
   : sender agent 1
   : receiver agent 5
   : content (price good200 150)
   : language sl
   : ontology hpl-auction
 )
```

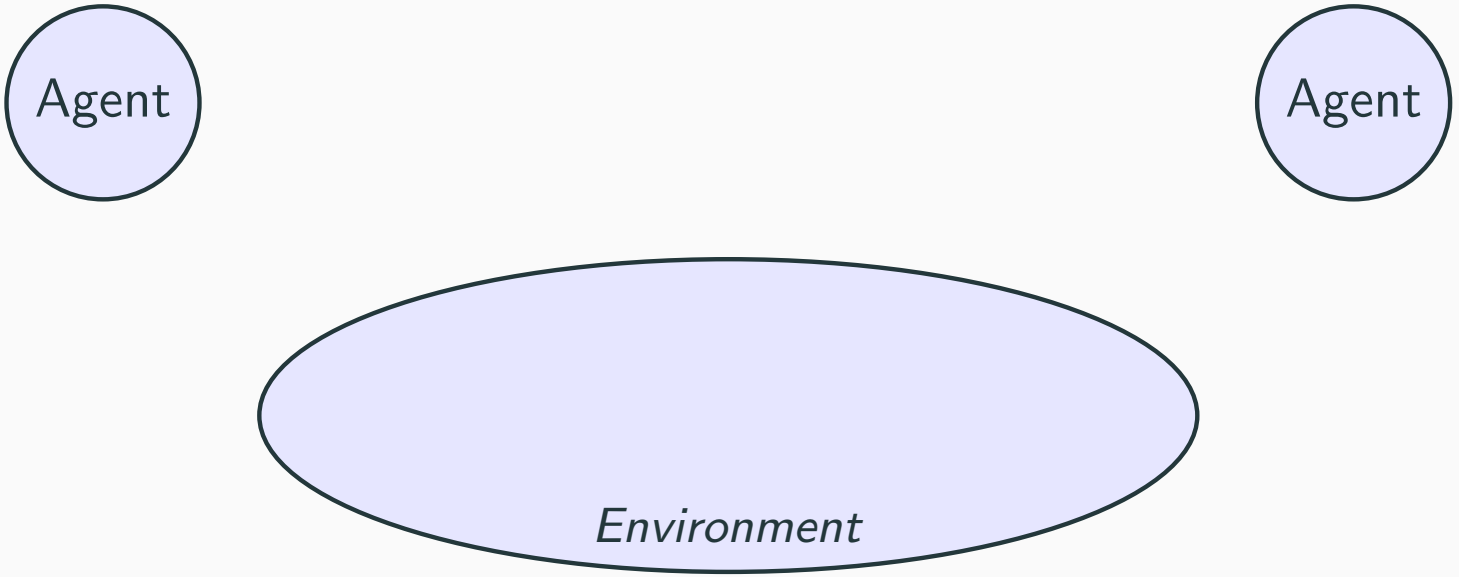
Note that the performative is the leader element!

FIPA-ACL standard

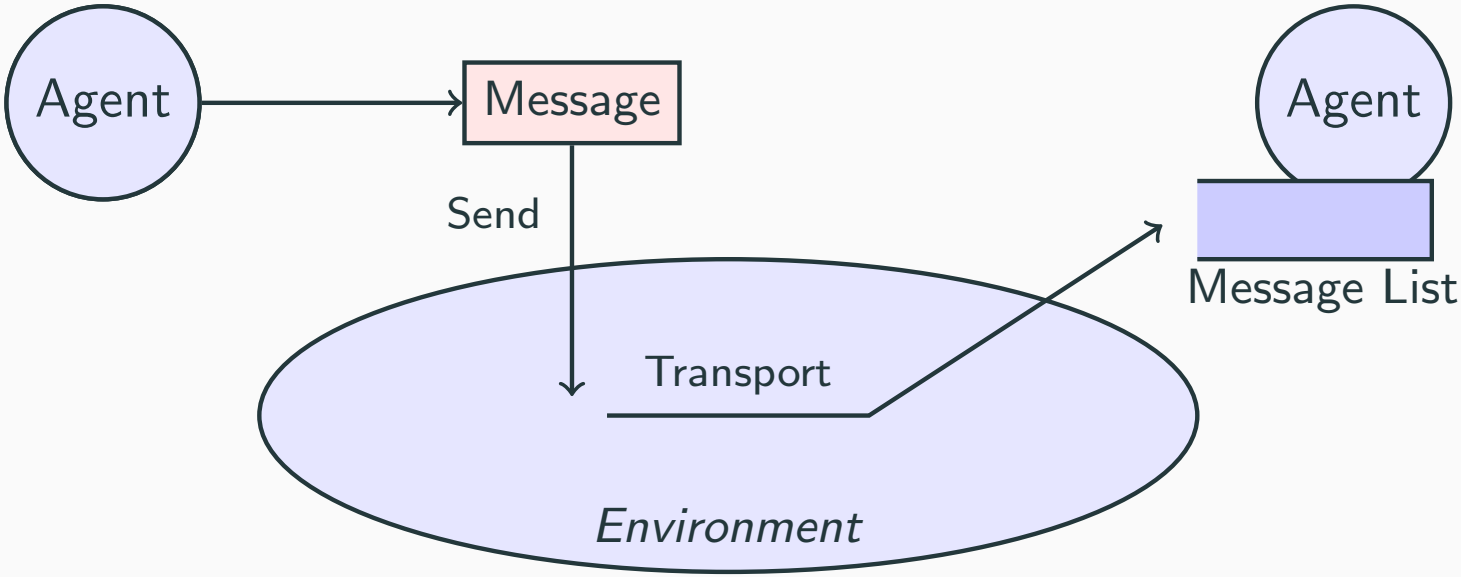
FIPA ACL Performatives

performative	passing info	requesting info	negotiation	performing actions	error handling
accept-proposal			x		
agree				x	
cancel		x		x	
cfp			x		
confirm	x				
disconfirm	x				
failure					x
inform	x				
inform-if	x				
inform-ref	x				
not-understood					x
propose			x		
query-if		x			
query-ref		x			
refuse				x	
reject-proposal			x		
request				x	
request-when				x	
request-whenever				x	
subscribe		x			

How does this work? Asynchronous message sending

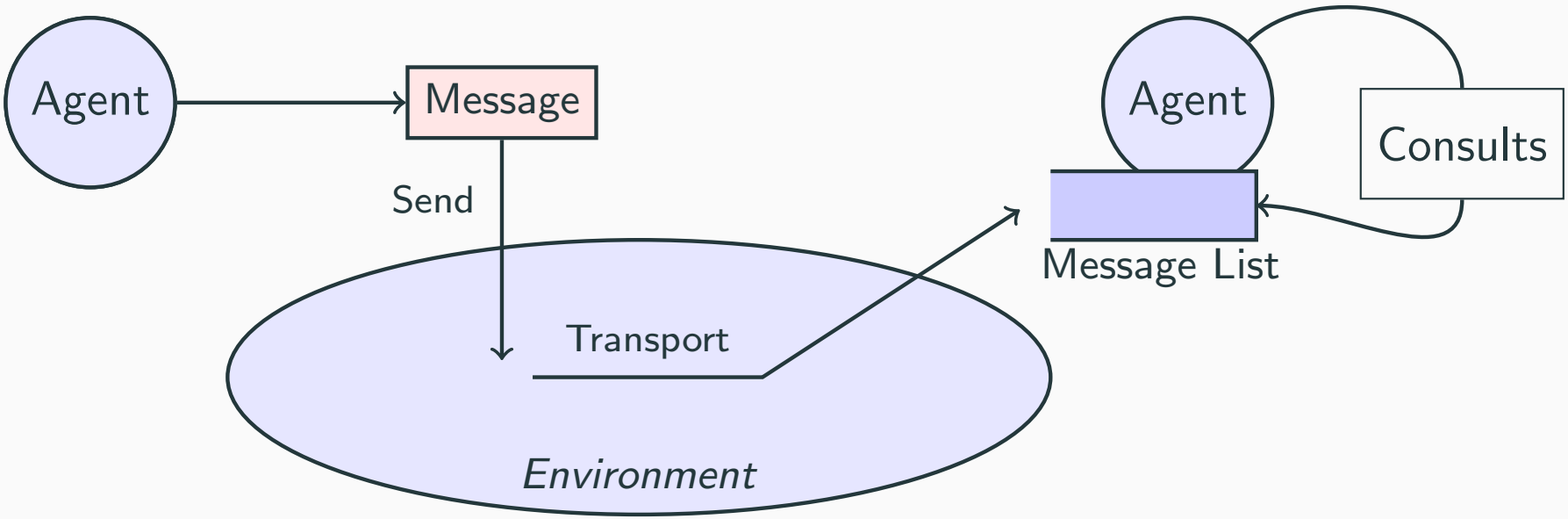


How does this work? Asynchronous message sending



Synchronous method calls

How does this work? Asynchronous message sending



Synchronous method calls	Asynchronous

Practical Work

Alice-Bob with direct interaction

- Agent Charles has a value
- Agent Alice and Bob have preferred values for Charles
- On their turn, they ask Charles its value (message)

```
m = Message("ask",self.id,"value")
m.addReceiver(charles_id)
self.environment.send(m)
```

- Depending on the answer, they ask Charles to change the value (message)

```
m = self.receive() # look in the msg box
if m.get_sender()==charles_id
  and m.get_performative()=="assert"
  and m.get_content()!=self.preferred_value: ...
```

➜ Build a model that is generic enough

Messages

```
1  class Message:
2    def __init__(self,performative,sender_id,content=
       None):
3      self.receivers = []
4      ...
5    def add_receiver(self,agent_id): ...
6    def get_content(self): ...
7    def get_performative(self): ...
8    def get_sender(self): ...
9    def get_receivers(self): ...
10
```

Practical Work

Environment

```
class Environment:
    ... # has a list of agents
    def send(self,message):
        for id in message.get_receivers():
            self.agents[id].append_message(message)
```

Agents (generic)

```
class Agent(Thread):
    ... # has a mailbox
    def append_message(self,message):
        self.mailbox.append(message)
    def receive(self):
        l = self.mailbox.copy()
        self.mailbox.clear()
        return l
```

Practical Work

Alice & Bob

```
class AliceBob(Agent):
    def procedural_loop(self):
        m = self.receive()
        if m==None: # ask Charles' value
          m = Message(...)
          self.env.send(m)
        else: # react to Charle's answer
```

Charles

```
class Charles(Agent):
    def procedural_loop(self):
        m = self.receive()
        if m!=None:
          if m.get_performative()=="ask": # answer
          elif m.get_performative()=="order": # change
```

What else?

We have

- A more or less generic Agent, Environment, Message architecture
- Code for Alice/Bob agents and Charles agents

What is missing?

What else?

We have

- A more or less generic Agent, Environment, Message architecture
- Code for Alice/Bob agents and Charles agents

What is missing?

- At the implementation level: IDs
  Agents need to know the ID of their interlocutor(s)
  ➜ Yellow Pages in the environment

```
ids = self.env.get_ids(role="charles")
for id in ids:
  m.add_receiver(id)
```

- At the conceptual level: protocols
  ➜ Define how agents should exchange information!

Interaction protocols

Interaction protocols I

Protocols

Describes how agents can interact in the MAS

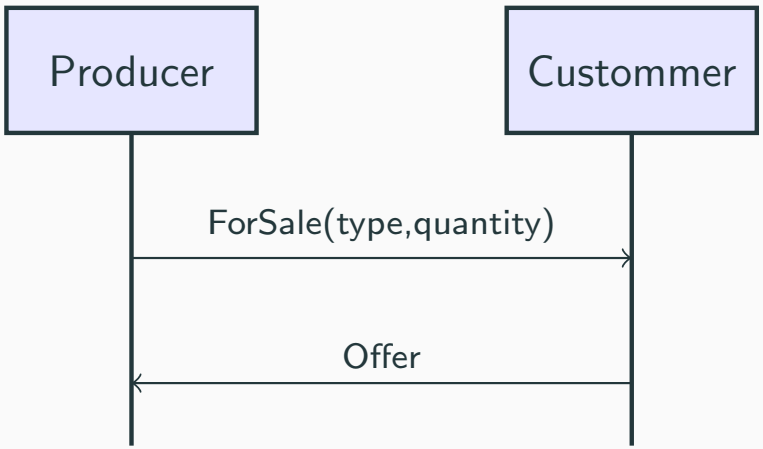
Interaction protocols I

Protocols

Describes how agents can interact in the MAS

AUML (Agent Unified Modeling language) standard

- Inspired from UML sequence diagrams
- Describes message exchange between roles
  - An agent can adopt several roles
  - A role can be fulfilled by several different agents



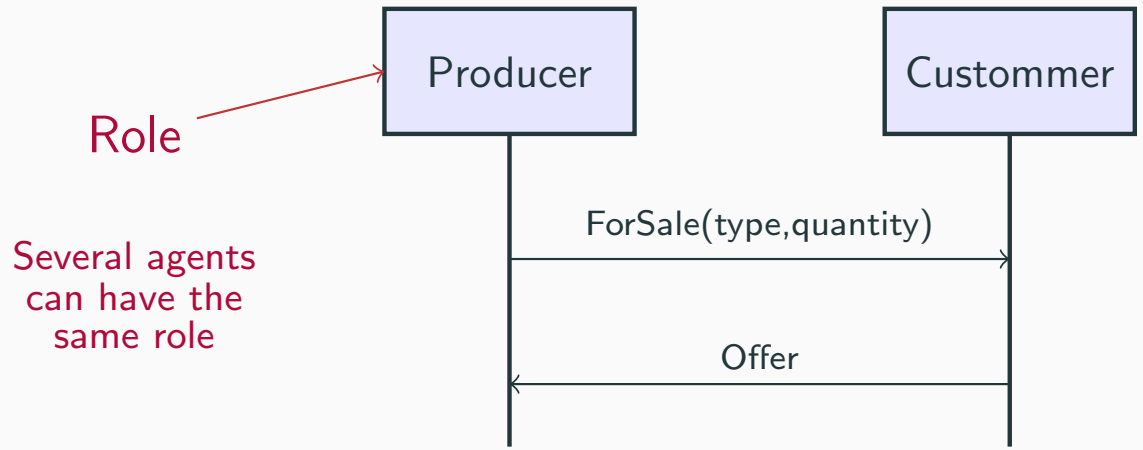
Interaction protocols I

Protocols

Describes how agents can interact in the MAS

AUML (Agent Unified Modeling language) standard

- Inspired from UML sequence diagrams
- Describes message exchange between roles
  - An agent can adopt several roles
  - A role can be fulfilled by several different agents



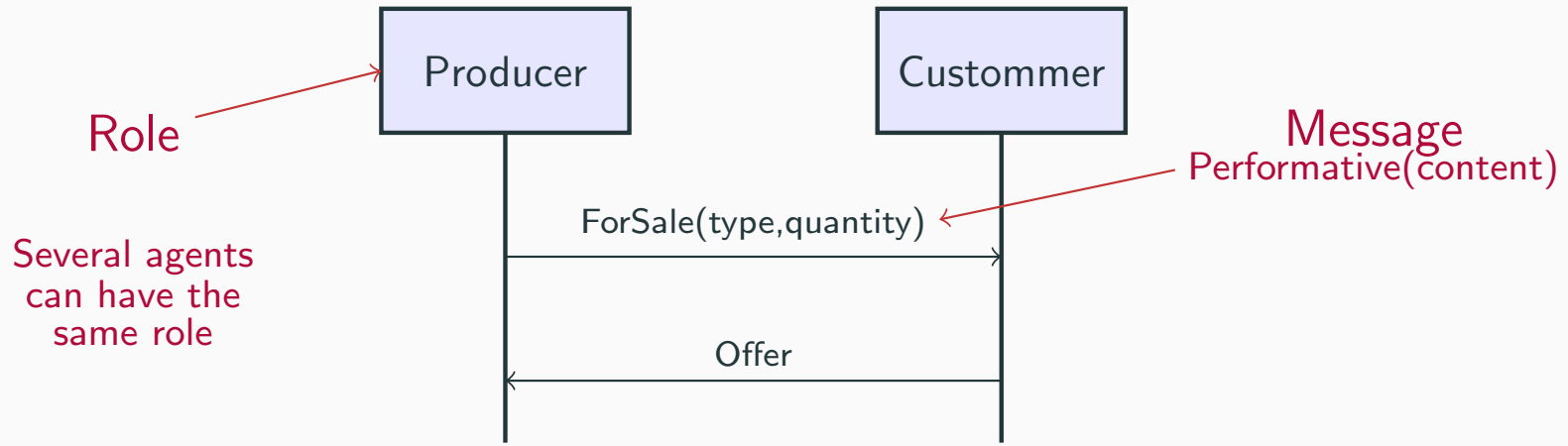
Interaction protocols I

Protocols

Describes how agents can interact in the MAS

AUML (Agent Unified Modeling language) standard

- Inspired from UML sequence diagrams
- Describes message exchange between roles
  - An agent can adopt several roles
  - A role can be fulfilled by several different agents



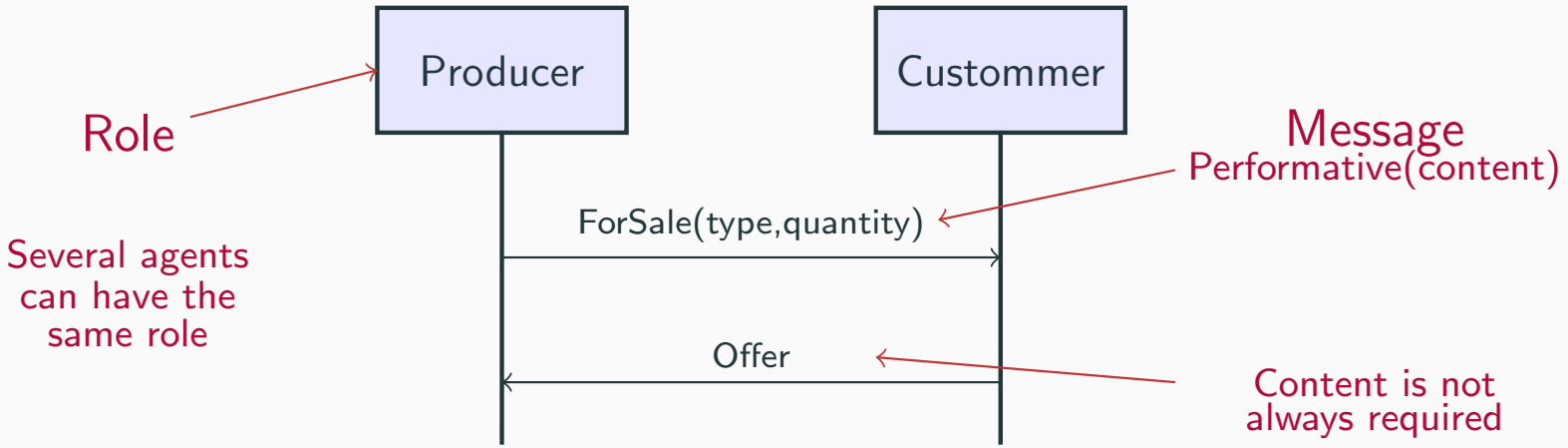
Interaction protocols I

Protocols

Describes how agents can interact in the MAS

AUML (Agent Unified Modeling language) standard

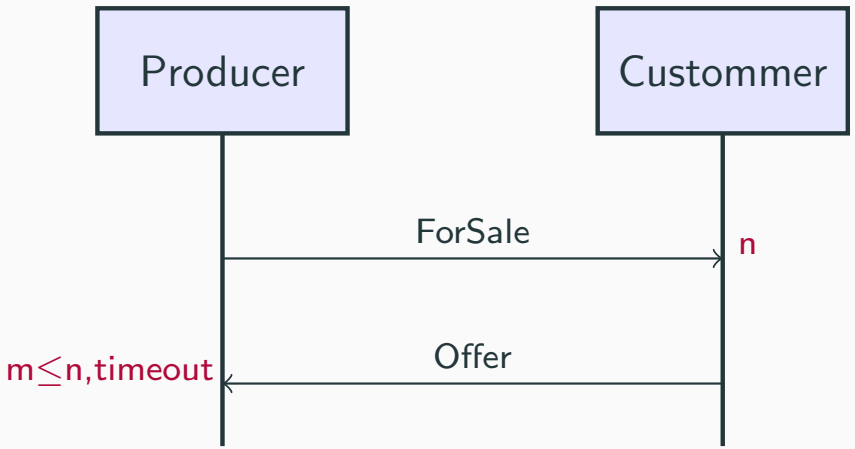
- Inspired from UML sequence diagrams
- Describes message exchange between roles
  - An agent can adopt several roles
  - A role can be fulfilled by several different agents



Interaction protocols II

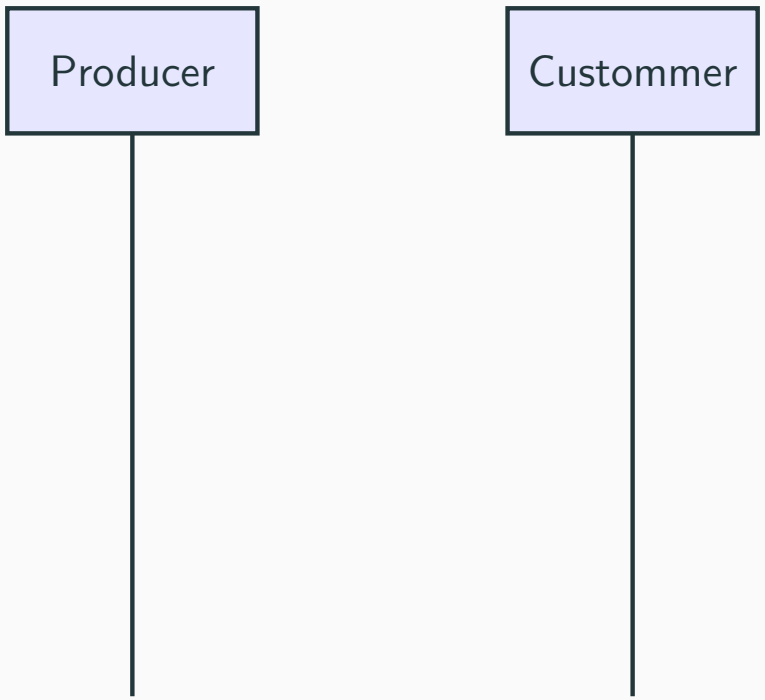
Conditions

- Number of messages sent (arrow end)
- Timeouts
  - Messages received after timeout are considered out of the protocol



Interaction protocols III

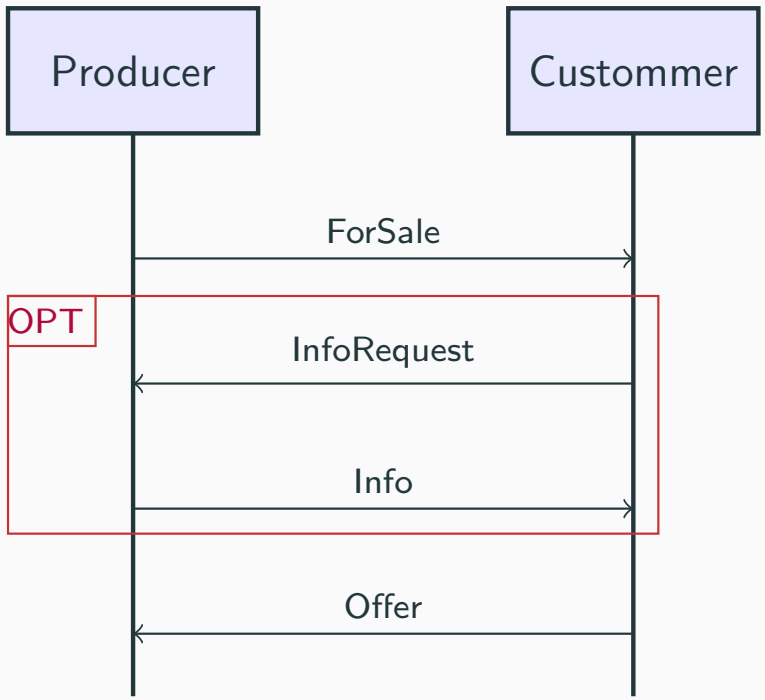
Operators



Interaction protocols III

Operators

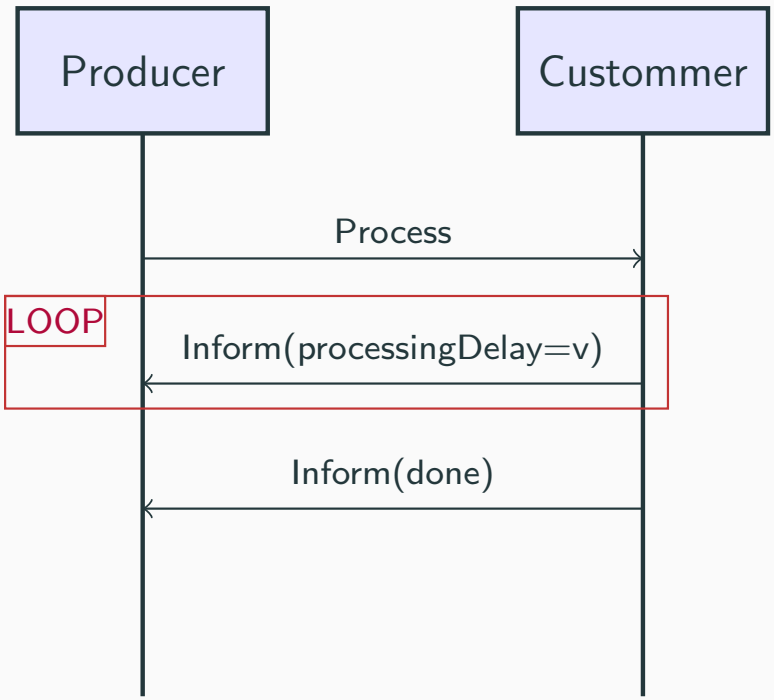
- OPT $\rightarrow$ some parts can be optional



Interaction protocols III

Operators

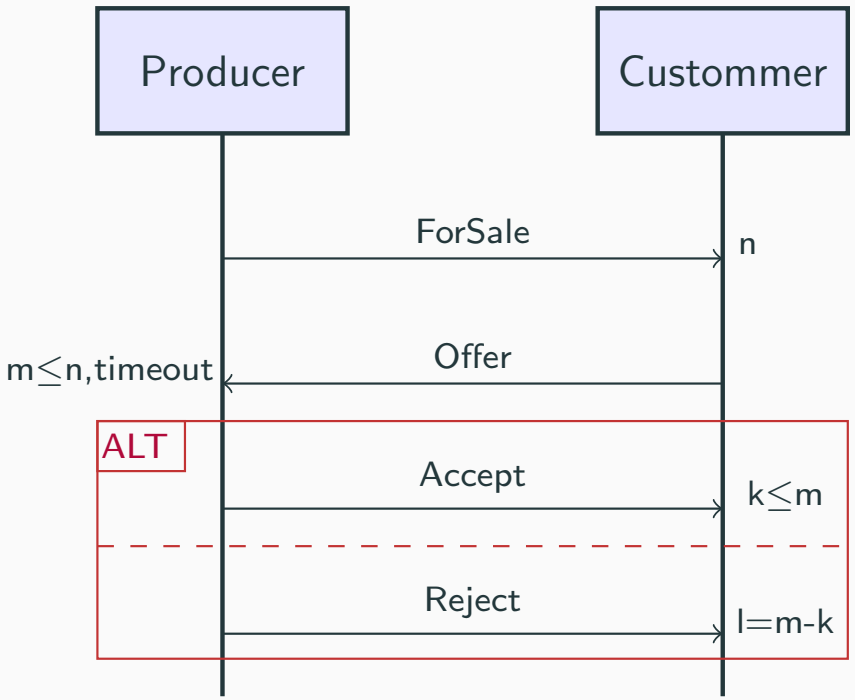
- OPT $\rightarrow$ some parts can be optional
- LOOP $\rightarrow$ some parts can be repeated randoml



Interaction protocols III

Operators

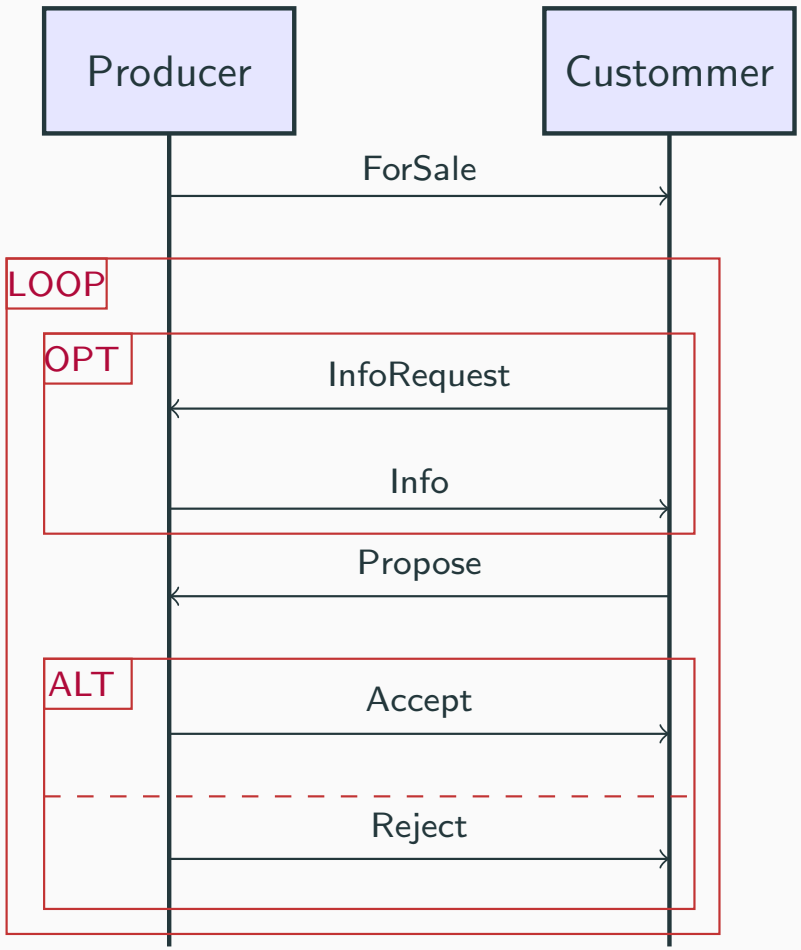
- OPT $\rightarrow$ some parts can be optional
- LOOP $\rightarrow$ some parts can be repeated randoml
- ALT $\rightarrow$ one or the other



Interaction protocols IV

Operators

- Operators can be combined



Interaction protocols

Contract-Net Protocol (CNP)

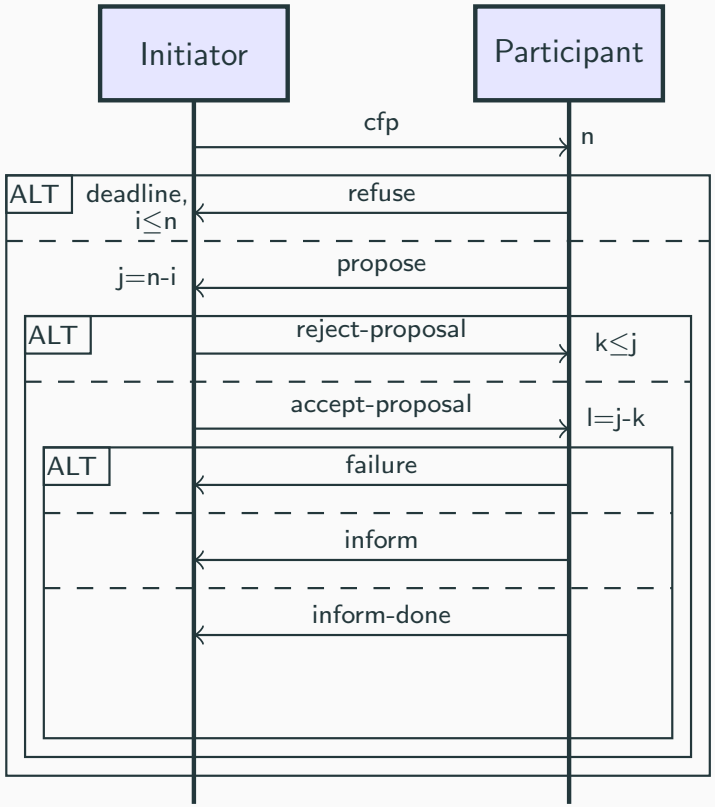
- One of the oldest, most widely used agent interaction protocols
- A manager agent announces one or several tasks, agents place bids for performing them
- Task is assigned by manager according to evaluation function applied to agents' bids (e.g., choose cheapest agent)
- Idea of exploiting local cost function (agents' private knowledge) for distributed optimal task allocation
- Even in purely cooperative settings, decentralization can improve global performance
- Successfully applied to different domains (e.g. transport logistics)

Interaction protocols V

Contract-Net Protocol (CNP)

Standard for agents to agree on a transaction.

- FIPA standard
- The "must-know" protocol



Communication in Mesa

Using of messaging communication in Mesa

open and explore mesa.zip

with the help of the file PW3_ExploringMesa.pdf

(correction provided, but you need to understand the mechanism)