Interaction and Communication

Autonoumous Software Agents

A.A. 2022-2023

Prof. Paolo Giorgini

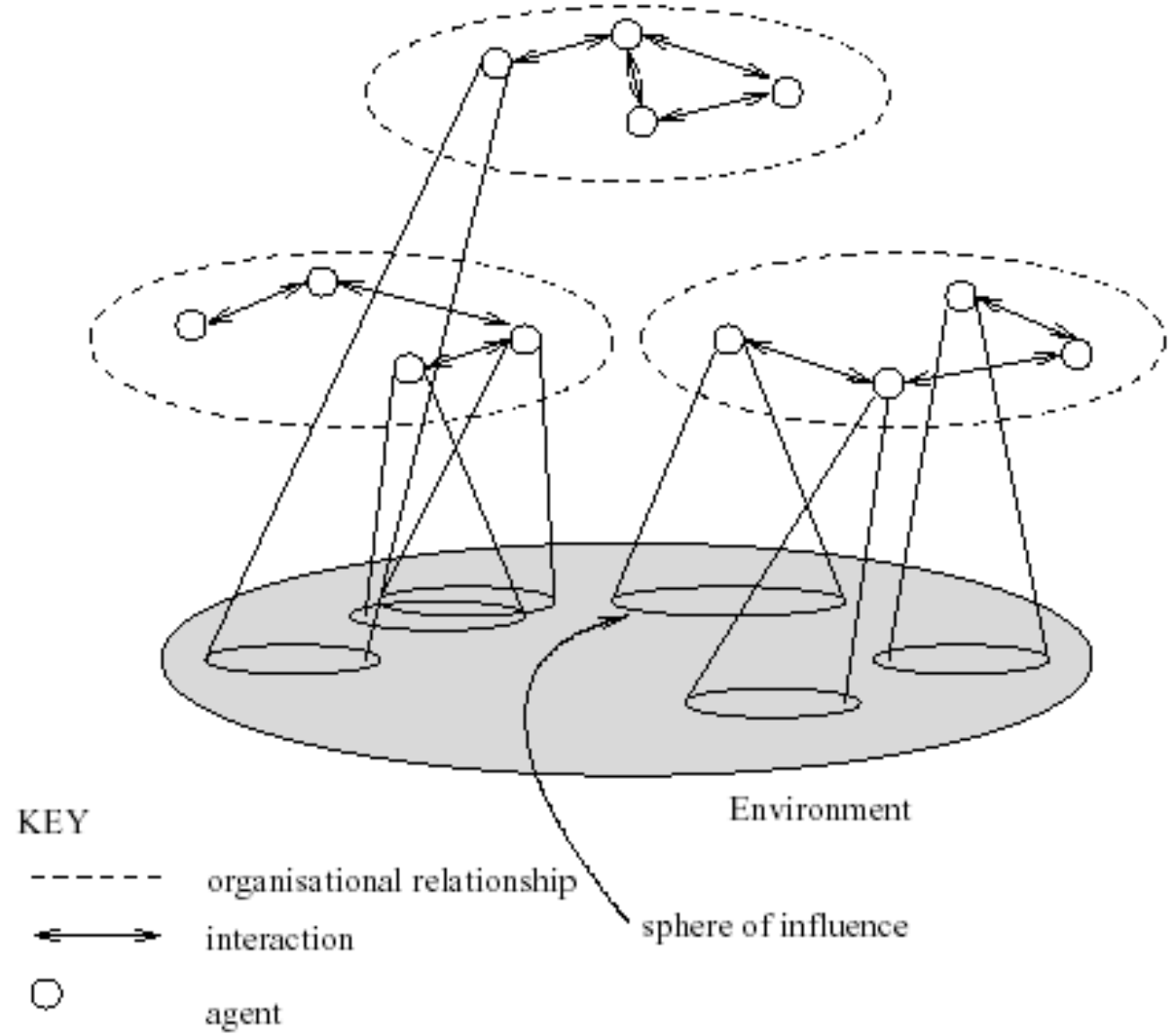
UNIVERSITY OF TRENTO - Italy

Department of Information
and Communication Technology

Agents interact one another

A multiagent system contains a number of agents

- which may interact through communication
- are able to act in an environment
- have different “spheres of influence” (which may coincide)
- will be linked by other (organizational) relationships`

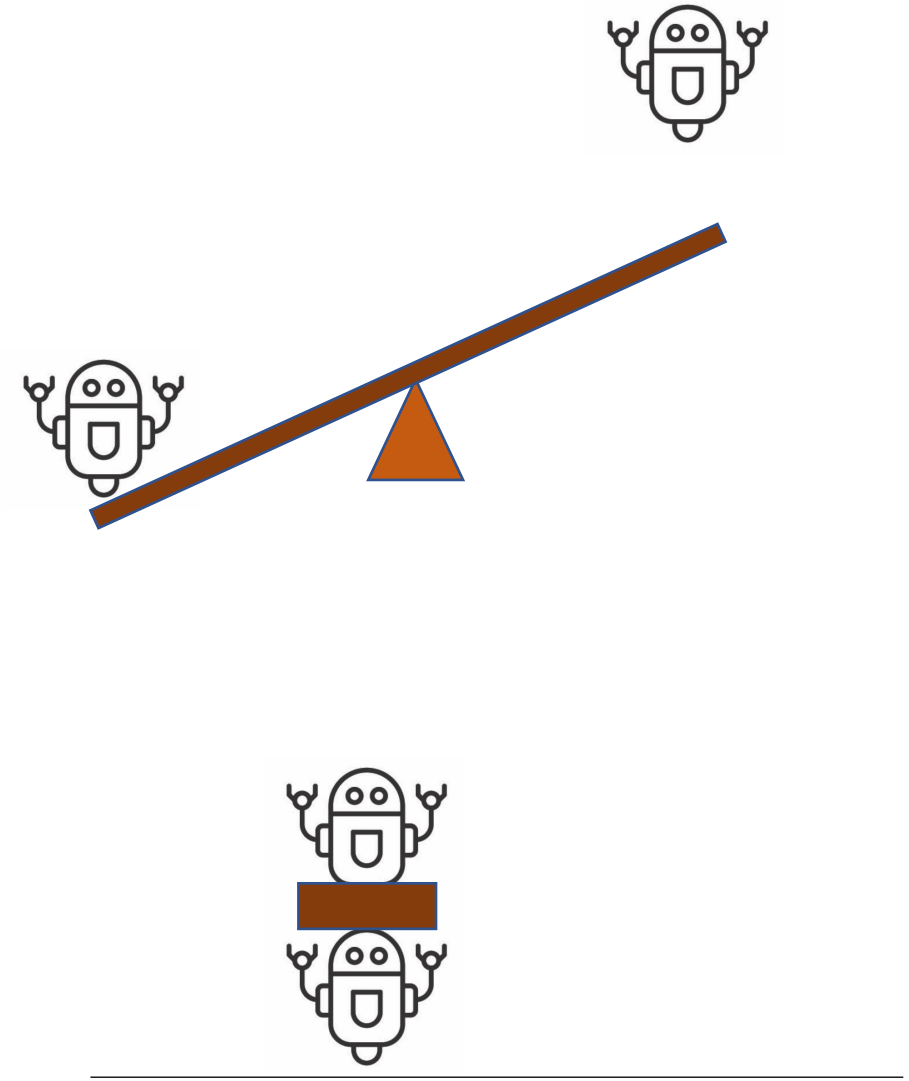


Agents' interaction

Physical interaction

- Doing an action in the environment the agent may affect the status of another agent

- An agent can act physically on another agent

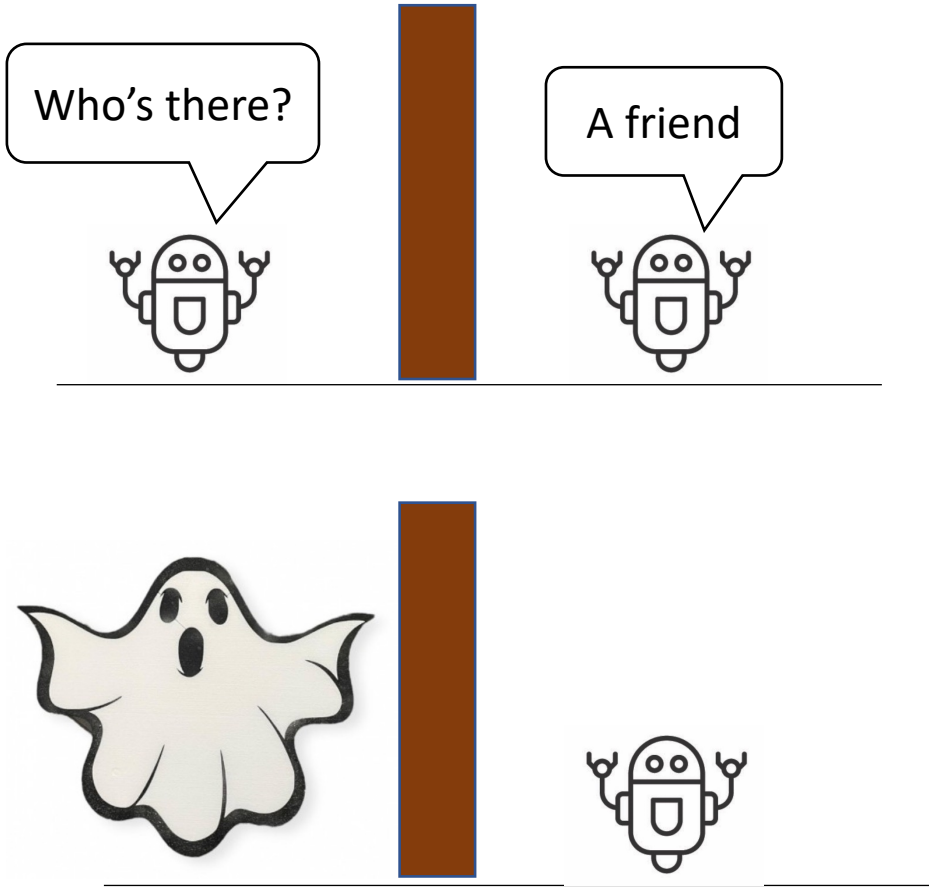


Agents' interaction

Communicative Interaction

- Agents interact through communication
- A common communication language

Other form of interaction



Why should agents interact ?

- They are part of the same environment
- They compete
  - E.g., for the use of resources
  - Self-interested agents
- They cooperate
  - E.g., being part of the same system (common overall goal)
  - Benevolent agents

Benevolent Agents

- If we "own" the whole system, we can design agents to help each other whenever asked
- In this case, we can assume agents are *benevolent*: our best interest is their best interest
- Problem-solving in benevolent systems is Cooperative Distributed Problem Solving (CDPS)
- Benevolence simplifies the system design task enormously!

Task Sharing and Result Sharing

- Two main modes of cooperative problem solving:
  - Task sharing
    components of a task are distributed to component agents
  - Result sharing
    information (partial results, etc.) is distributed

Multi-agent planning

The goal state is reached combining actions of a team of agents

- Elaborating an overall plan
  - Putting together the capabilities (actions) of the agents
- Coordinating agents' activities
- Centralized or distributed point of control
  - E.g., A master agent responsible to elaborate the plan, dispatch activities to the team, control the flow of activities and monitoring their correct execution
  - E.g., The team negotiate the plan and coordinate activities one anothr
- Use of interaction protocols

The Contract Net

- A well-known task-sharing protocol for task allocation is the contract net
  - Recognition
  - Announcement
  - Bidding
  - Awarding
  - Expediting

Recognition

- In this stage, an agent recognizes it has a problem it wants help with
  Agent has a goal, and either…
  - realizes it cannot achieve the goal in isolation — does not have capability
  - realizes it would prefer not to achieve the goal in isolation (typically because of solution quality, deadline, etc.)

Announcement

- In this stage, the agent with the task sends out an announcement of the task which includes a specification of the task to be achieved
- Specification must encode:
  - description of task itself (maybe executable)
  - any constraints (e.g., deadlines, quality constraints)
  - meta-task information (e.g., “bids must be submitted by…”)
- The announcement is then broadcast

Bidding

- Agents that receive the announcement decide for themselves whether they wish to bid for the task
- Factors:
  - agent must decide whether it is capable of performing the task
  - agent must determine quality constraints & price information (if relevant)
- If they do choose to bid, then they submit a tender

Awarding & Execution

- Agent that sent task announcement must choose between bids & decide who to "award the contract" to
- The result of this process is communicated to agents that submitted a bid
- The successful contractor then execute the task
- May involve generating further manager-contractor relationships (e.g., sub-contracting)

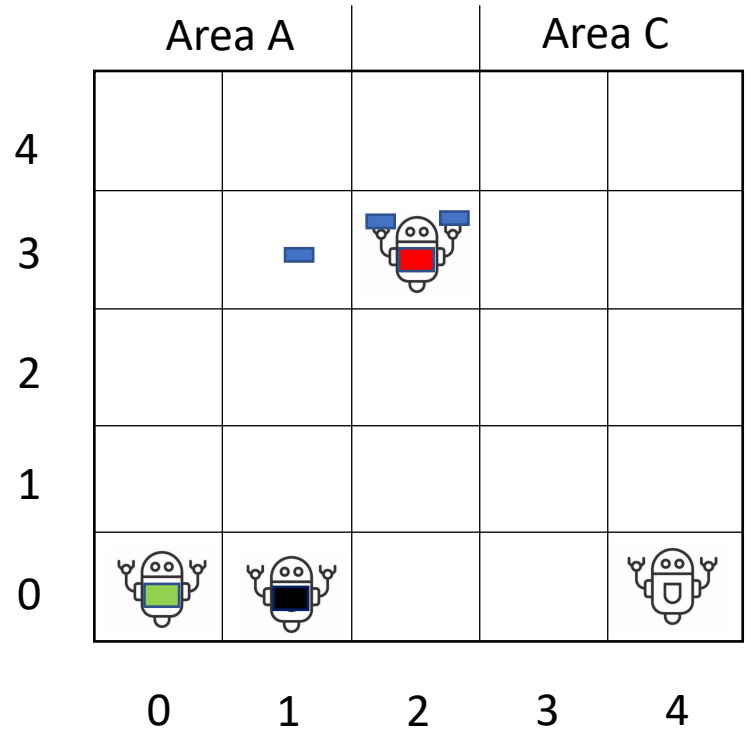
Efficiency Modifications

- Focused addressing — when general broadcast isn't required
- Directed contracts — when manager already knows which node is appropriate
- Request-response mechanism — for simple transfer of information without overhead of contracting
- Node-available message — reverses initiative of negotiation process

Example: packages collection

Announcement

```
To:                 *
From:               RED
Type:               Task Announcement
Contract:           23-05-02
Eligibility Specification:
                    Must-Have GRIPPER
                    Must-be-in Area A
Task Abstraction:
                    Task-Type COLLECT
                    Position (1,3)
                    Area A Specification ((0,0),(1,4))
Bid Specification:
                    Position (X,Y)
                    Agent Name
Expiration Time:
                    02 1730 MAY 2023
```



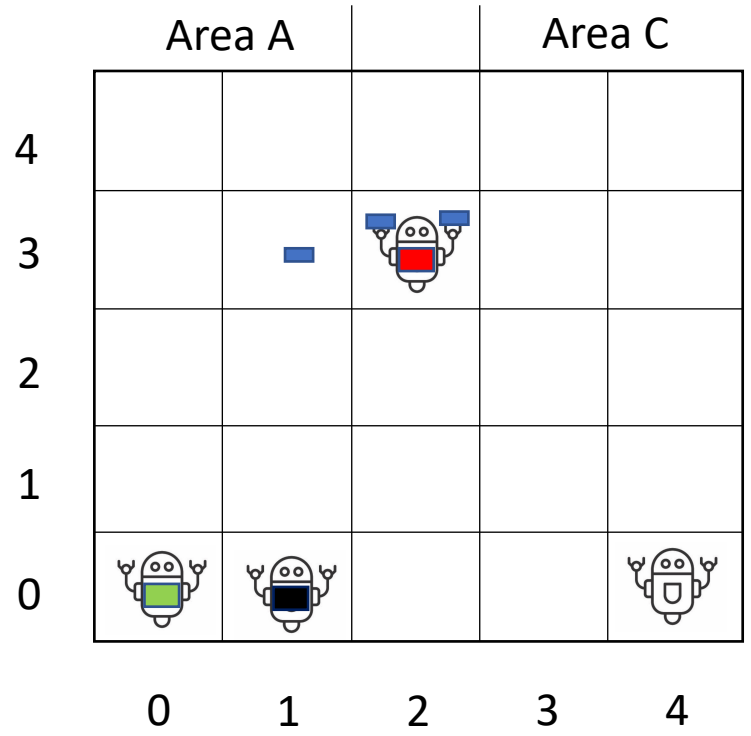
Example: packages collection

Bid

To: RED
From: GREEN
Type: BID
Contract: 23-05-02
Bid Specification:
Position (0,0)
Agent GREEN

Award

To: GREEN
From: RED
Type: AWARD
Contract: 23-05-02
Task Specification:
Task-Type COLLECT
Position (1,3)



BDI loop for the contract-net 1

Recognition – Announcement - Awarding

Intention Rules: if (prec) then Int

```
if (in(Pck,X,Y) ^ free_gripper(G)) {O += collect(Pck,X,Y)}

if (in(Pck,X,Y) ^ not free_gripper(G)) {O += ask_collect(Pck,X,Y)}

        → I={ask_collect(Pck,X,Y)}  # options -> intention
```

Plans selection or Plannig: P = planning (B,I,P)

```
        → P = {p1}                              # plan for I

        → p1={announcment(collect(pack_1)), bids_eval(collect(pack_1)),
            awarding(collect(pack_1))}
```

Announcement

To: *
From: RED
Type: Task Announcement
Contract: 23-05-02
Eligibility Specification:
Must-Have GRIPPER
Must-be-in Area A
Task Abstraction:
Task-Type COLLECT
Position (1,3)
Area A Specification ((0,0),(1,4))
Bid Specification:
Position (X,Y)
Agent Name
Expiration Time:
02 1730 MAY 2023

Award

To: GREEN
From: RED
Type: AWARD
Contract: 23-05-02
Task Specification:
Task-Type COLLECT
Position (1,3)

BDI loop for the contract-net 2

Bidding - Execution

Bid

To: RED
From: GREEN
Type: BID
Contract: 23-05-02
Bid Specification:
Position (0,0)
Agent GREEN

Intention Rules:

```
if (task_ann(Pck,X,Y) ^ free_gripper(G)) {O += bid(Pck,X,Y)}

if (award(Pck,XY)) {O += collect(Pck,X,Y)}

        → I={bid(Pck,X,Y)}  # options -> intention
```

Plans selection or Plannig: `P = planning (B,I,P)`

```
        → P = {p1}                          # plan for bid(Pack_1,X,Y)

        → p1={bidding(collect(pack_1))}

        → p2={move(UP), move(UP), move(UP), PICK_UP(pack_1))}   # plan for collect(Pack_1,X,Y)
```

Self-Interested Agents

- If agents represent individuals or organizations, (the more general case), then we cannot make the benevolence assumption
- Agents will be assumed to act to further their own interests, possibly at expense of others
- Potential for conflict
- May complicate the design task enormously

Utilities and Preferences

- Assume we have just two agents: $Ag = \{i, j\}$
- Agents are assumed to be self-interested:
  - they have preferences over how the environment is
- Assume $\Omega = \{\omega_1, \omega_2, \ldots\}$ is the set of "outcomes" that agents have preferences over
- We capture preferences by utility functions

$$u_i = \Omega \rightarrow \mathrm{R}$$
$$u_j = \Omega \rightarrow \mathrm{R}$$

- Utility functions lead to preference orderings over outcomes:

$$\omega >_i \omega' \text{ means } u_i(\omega) > u_i(\omega')$$
$$\omega \geq_i \omega' \text{ means } u_i(\omega) \geq u_i(\omega')$$

Utility-based agent

- A measure of how desirable a particular state (goal) is.
- *Utility function* maps a state to a measure of the utility of the state
  - Can describe how "happy" the agent is
- A rational utility-based agent chooses the action that maximizes the expected utility of the action outcomes
  - that is, the agent expects to derive, on average, given the probabilities and utilities of each outcome.

Multiagent Encounters

- We need a model of the environment in which these agents will act:
  - agents simultaneously choose an action to perform, and as a result of the actions they select, an outcome in Ω will result
  - the actual outcome depends on the combination of actions
  - assume each agent has just two possible actions that it can perform, *C* ("Cooperate") and *D* ("Defect")
- Environment behavior given by *state transformer function*:

$$\tau : \underbrace{Ac}_{\text{agent } i\text{'s action}} \times \underbrace{Ac}_{\text{agent } j\text{'s action}} \rightarrow \Omega$$

Multiagent Encounters

- Here is a state transformer function:

  $\tau(D,D) = \omega_1 \quad \tau(D,C) = \omega_2 \quad \tau(C,D) = \omega_3 \quad \tau(C,C) = \omega_4$

  (This environment is sensitive to actions of both agents.)

- Here is another:

  $$\tau(D,D) = \omega_1 \quad \tau(D,C) = \omega_1 \quad \tau(C,D) = \omega_1 \quad \tau(C,C) = \omega_1$$

  (Neither agent has any influence in this environment.)

- And here is another:

  $$\tau(D,D) = \omega_1 \quad \tau(D,C) = \omega_2 \quad \tau(C,D) = \omega_1 \quad \tau(C,C) = \omega_2$$

  (This environment is controlled by *j*.)

Rational Action

- Suppose we have the case where both agents can influence the outcome, and they have utility functions as follows:

$$u_i(\omega_1) = 1 \quad u_i(\omega_2) = 1 \quad u_i(\omega_3) = 4 \quad u_i(\omega_4) = 4$$
$$u_j(\omega_1) = 1 \quad u_j(\omega_2) = 4 \quad u_j(\omega_3) = 1 \quad u_j(\omega_4) = 4$$

- With a bit of abuse of notation:

$$u_i(D,D) = 1 \quad u_i(D,C) = 1 \quad u_i(C,D) = 4 \quad u_i(C,C) = 4$$
$$u_j(D,D) = 1 \quad u_j(D,C) = 4 \quad u_j(C,D) = 1 \quad u_j(C,C) = 4$$

- Then agent i's preferences are:

$$C,C \succeq_i C,D \quad \succ_i \quad D,C \succeq_i D,D$$

- "*C*" is the rational choice for i.
(Because i prefers all outcomes that arise through *C* over all outcomes that arise through *D*.)

what about *j*?

Payoff Matrices

- We can characterize the previous scenario in *a payoff matrix*

j \ i	defect	coop
defect	1 1	4 1
coop	1 4	4 4

$$u_i(D,D) = 1 \quad u_i(D,C) = 1 \quad u_i(C,D) = 4 \quad u_i(C,C) = 4$$
$$u_j(D,D) = 1 \quad u_j(D,C) = 4 \quad u_j(C,D) = 1 \quad u_j(C,C) = 4$$

- Agent i is the column player
- Agent j is the row player

Dominant Strategies

- Given any particular strategy S (either *C* or *D*) of agent i, there will be a number of possible outcomes
- We say s_1 *dominates* s_2 if every outcome possible by i playing s_1 is preferred over every outcome possible by i playing s_2
- A rational agent will never play a dominated strategy
- So, in deciding what to do, we can delete dominated strategies
- Unfortunately, there isn't always a unique undominated strategy

Nash Equilibrium

- In general, we will say that two strategies s_1 *and* s_2 are in Nash equilibrium if:
  - under the assumption that agent i plays s_1, agent j can do no better than play s_2; and
  - under the assumption that agent j plays s_2, agent i can do no better than play s_1.
- Neither agent has any incentive to deviate from a Nash equilibrium
- Unfortunately:
  - *Not every interaction scenario has a Nash equilibrium*
  - *Some interaction scenarios have more than one Nash equilibrium*

Competitive and Zero-Sum Interactions

- Where preferences of agents are diametrically opposed, we have strictly competitive scenarios
- Zero-sum encounters are those where utilities sum to zero:

$$u_i(\omega) + u_j(\omega) = 0 \quad \text{for all } \omega \in \Omega$$

- Zero sum implies strictly competitive
  - A zero-sum interaction is when one party's gain equals the other party's loss — the sum of their gains and losses is zero.
- Zero sum encounters in real life are very rare … but people tend to act in many scenarios as if they were zero sum

The Prisoner's Dilemma

- Two men are collectively charged with a crime and held in separate cells, with no way of meeting or communicating. They are told that:
  - if one confesses and the other does not, the confessor will be freed, and the other will be jailed for three years
  - if both confess, then each will be jailed for two years
  - Both prisoners know that if neither confesses, then they will each be jailed for 1 year

The Prisoner's Dilemma

Years of prison	Utility
0	4
1	3
2	2
3	1

		i: Not Confess	i: Confess
j	Not Confess	3 (1 year) 3 (1 year)	4 (freed) 1 (3 years)
j	Confess	1 (3 years) 4 (freed)	2 (2 years) 2 (2 years)

The Prisoner's Dilemma

- Best strategy is (C,C)
  - Confess ---> from 0 to 2 years
  - Not confess ---> from 1 to 3 years
- The strategy "Not confess" is dominated by "confess"
- Eliminating the dominated strategies, we have that both prisoners confess (and both get 2 years)
- However, the best for each of them is to not confess (1 year)
  - Suppose they both agree before being arrested to not confess
    - If now one of them does not respect the agreement he will be freed
    - So, what to do: confess or not confess?

The Prisoner's Dilemma

- This apparent paradox is the fundamental problem of multi-agent interactions
  - cooperation will not occur in societies of self-interested agents
- Real world examples:
  - nuclear arms reduction ("why don't I keep mine. . . ")
- The prisoner's dilemma is ubiquitous

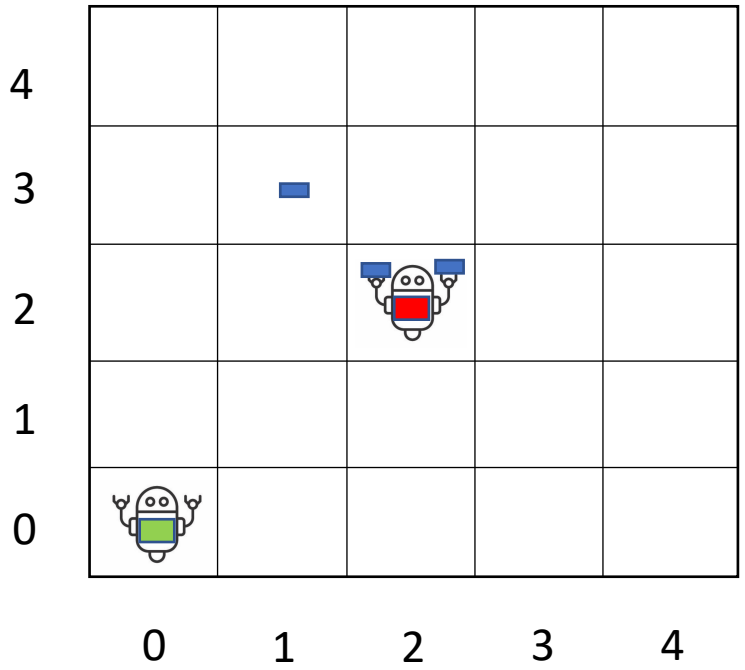
Reasoning about agents' mental state

- For a better interaction agents' may have a local representation of other agents' mental state
  - Can be used to reason about other agents' beliefs and intentions
  - Useful during a negotiation
  - Very challenging problem (e.g., keeping the state updated)
  - Using your own BDI model can be a limitation
    - E.g., Are you sure the other agents' has your intentions rules?
  - The mental state representation can be updated by observation and/or communication

Example – by observation 1

RED about GREEN

```
-- t=0
B={}
I={}
-- t=2
B={in(pack_1,1,3)}
I={pick_up(pack_1)}
```



how to ?

GREEN moves towards the packages, so it has the intention to pick_up the package and, of course, it believes the package is there

Example – by observation 1

Observing the execution of P (or part of it) and from

```
if pick_up(pack) -> Plan P
```

the agent can abduce the intention `pick_up(pack)` and from

```
if in(pack,X,Y) -> pick_up(pack)
```

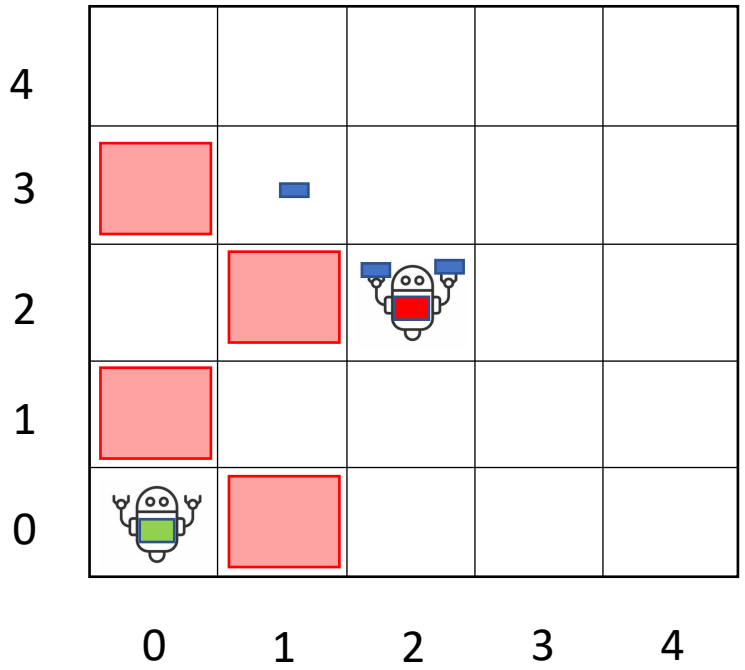
the agent can abduce the belief `in(pack,X,Y)`

(a possible explanation of the agent behaviour)

Example – by observation 2

RED about GREEN

```
-- t=0
B={}
I={}
-- t=2
B={}
I={look_for_pack()}
```



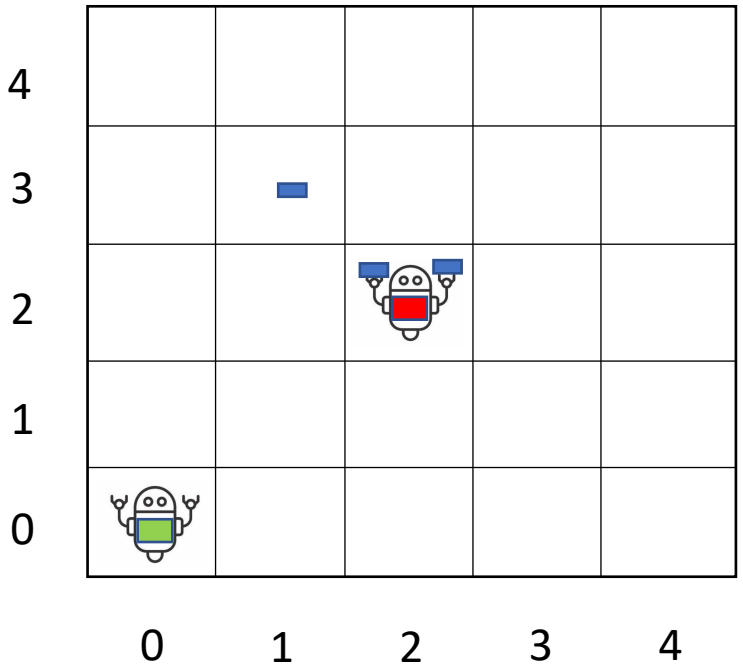
how to ?

GREEN moves towards the packages, but since it cannot see pack_1, it cannot have the intention to pick it up, it is wandering around to find a package

Example – by observation 3

GREEN about RED

```
-- t=0
B={}
I={}
-- t=1
B={in(pack_1,1,3)}
I={}
```



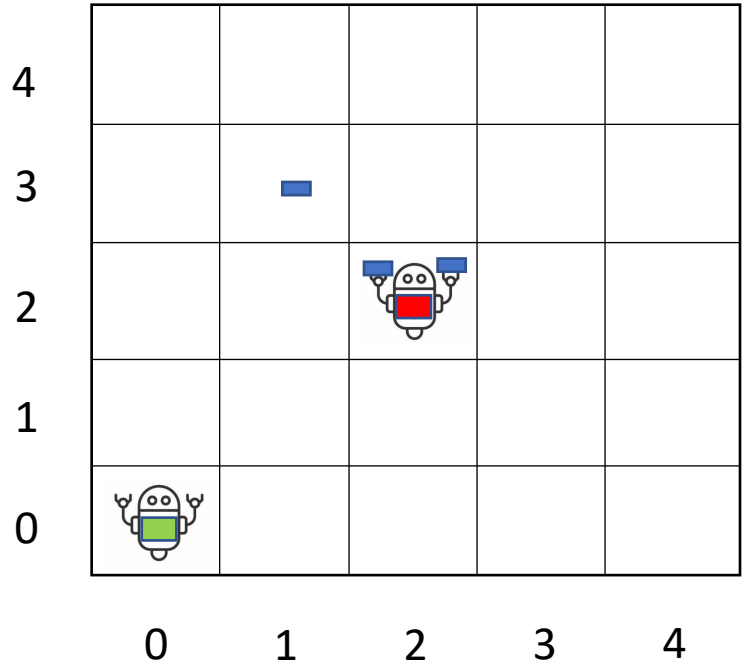
how to ?

RED can see pack_1, but since it has no free gripper, RED has no intention to pick up the package

Example – by communication 1

RED about GREEN

```
-- t=0
B={}
I={}
-- t=1 #RED inform GREEN about in(pack_1,1,3)
B={in(pack_1,1,3)}
I={pick_up(pack_1)}
```



how to ?

Now GREEN believes pack_1 is there, so it indents to pick up the package

if `in(pack,X,Y) -> pick_up(pack)`

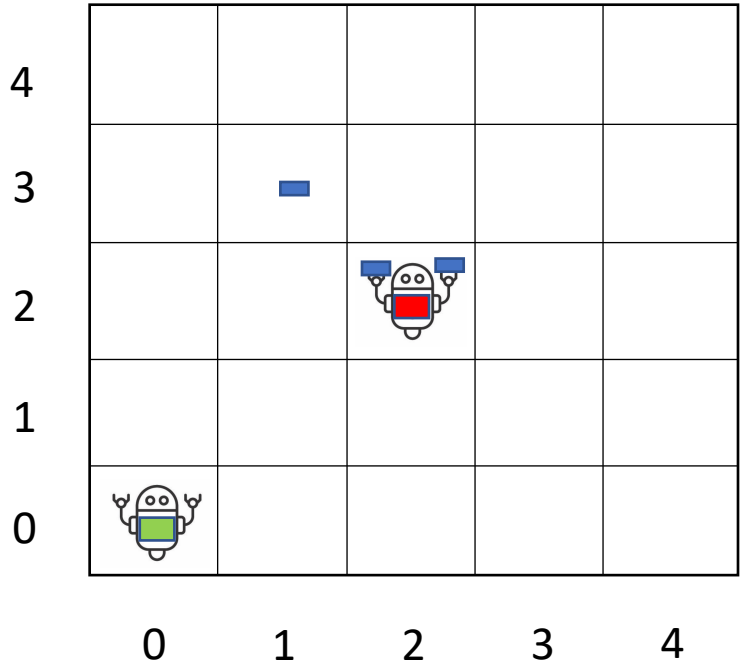
Note

- It could the intention of RED to make GREEN believe that in(pack_1,1,3), so GREEN will intend to pick_up(pack_1) and then make GREEN to move towards the package
- Communication can be used to change the mental state of the hearer to induce a specific behaviour (injection of an intention)

Example – by communication 2

RED about GREEN

```
-- t=0
B={}
I={}
-- t=1 #RED inform GREEN about pick_up(pack_1)
B={in(pack_1,1,3)}
I={}
```



how to ?

Now GREEN believes RED intends to pick up pack_1, so there should pack_1 and it does not make any sense that GREEN intends to pick up pack_1

```
if in(pack,X,Y) -> pick_up(pack)  # in(pack_1,1,3) for abduction for RED
```

```
if in(pack,X,Y) ^ not pick_up(A,pack) -> pick_up(pack)
```

Note

Also in this case, it could be a strategy of RED to make GREEN believe that RED intend to pick up pack_1 , so GREEN will not go to pick up the package

Communication is used as an action of a Plan

- Dissuade GREEN to pick up pack_1
- Delivery Packs
- Pick up pack_1

Agent Communication

- Macro-aspects of intelligent agent technology: those issues relating to the agent society, rather than the individual:
  - *communication*
    speech acts; KQML & KIF; FIPA ACL

Speech Acts

- Most treatments of communication in (multi-) agent systems borrow their inspiration from speech act theory
- Speech act theories are pragmatic theories of language,
  - i.e., theories of language use: they attempt to account for how language is used by people every day **to achieve their goals and intentions**
- The origin of speech act theories are usually traced to Austin's 1962 book, How to Do Things with Words

Speech Acts

- Austin noticed that some utterances are rather like “physical” actions that appear to change the state of the world
- Paradigm examples would be:
  - Declaring war
  - Christening
  - ‘I now pronounce you man and wife’ :-)
- But more generally, everything we utter is uttered with the intention of satisfying some goal or intention
- A theory of how utterances are used to achieve intentions is a speech act theory

Different Aspects of Speech Acts

- From "A Dictionary of Philosophical Terms and Names"
- Locutionary act the simple speech act of generating sounds that are linked together by grammatical conventions so as to say something meaningful
  - Among English speakers, for example, '`It is raining`' performs the locutionary act of saying that it is raining, as '`Grablistrod zetagflx dapu`' would not"

Different Aspects of Speech Acts

- Illocutionary act the speech act of doing something else – offering advice, for example – in the process of uttering meaningful language.
  - Thus, for example, in saying ‘`I will repay you this money next week`’ one typically performs the illocutionary act of making a promise

Different Aspects of Speech Acts

- Perlocutionary act the speech act of having an effect on those who hear a meaningful utterance
- By telling a ghost story late at night, for example, one may accomplish the cruel perlocutionary act of frightening a child

Speech Acts

- Searle (‘69) identified different types of speech act:
  - representatives
    such as *informing*, e.g., ‘`It is raining`’
  - directives
    attempts to get the hearer to do something e.g., ‘`please make the tea`’
  - commisives
    which commit the speaker to doing something, e.g., ‘`I promise to…`’
  - *expressives*
    whereby a speaker expresses a mental state, e.g., ‘`thank you!`’
  - *declarations*
    such as declaring war or christening

Speech Acts

- There is some debate about whether this (or any!) typology of speech acts is appropriate
- In general, a speech act can be seen to have two components:
  - a performative verb
    (e.g., request, inform, promise, …)
  - propositional content
    (e.g., “the door is closed”)

Speech Acts

Consider

```
performative = request
content = “the door is closed”
speech act = “please close the door”

performative = inform
content = “the door is closed”
speech act = “the door is closed!”

performative = inquire
content = “the door is closed”
speech act = “is the door closed?”
```

Plan Based Semantics

- How does one define the semantics of speech acts? When can one say someone has uttered, e.g., a request or an inform?
- Cohen & Perrault (1979) defined semantics of speech acts using the precondition-delete-add list formalism of planning research
- Note that a speaker cannot (generally) force a hearer to accept some desired mental state
- In other words, there is a separation between the illocutionary act and the perlocutionary act

Plan-Based Semantics

- Here is their semantics for **`request(s,h,ϕ)`**

  pre:

  - s believes h can do ϕ
    (you don't ask someone to do something unless you think they can do it)
  - s believes h believe h can do ϕ
    (you don't ask someone unless *they* believe they can do it)
  - s believes s wants ϕ
    (you don't ask someone unless you want it!)

  post:

  - h believes s believes s wants ϕ
    (the effect is to make them aware of your desire)

What about inform?

We could adopt that: **`inform(s,h,ϕ)`**

Pre:
s believes ϕ is true
s intends that h believes f

Post:
s believes that h believes ϕ
h believes ϕ

Other semantics of course can be adopted

What about inquire?

`Inquire(s,h,ϕ)`

pre:

- ….

post:

- ….

KQML and KIF

- We now consider *agent communication languages* (ACLs) — standard formats for the exchange of messages
- The best known ACL is KQML, developed by the ARPA knowledge sharing initiative
  KQML is comprised of two parts:
  - the Knowledge Query and Manipulation Language (KQML)
  - the Knowledge Interchange Format (KIF)

KQML and KIF

- KQML is a language, that defines various acceptable ‘communicative verbs’, or performatives
  Example performatives:
  - `ask-if` (‘is it true that. . .’)
  - `perform` (‘please perform the following action. . .’)
  - `tell` (‘it is true that. . .’)
  - `reply` (‘the answer is . . .’)
- KIF is a language for expressing message content

KQML performatives

Category	Reserved performative names
Basic informational performatives	tell, deny, untell, cancel,
Basic query performatives	evaluate, reply, ask-if, ask-about, ask-one, ask-all, sorry
Multi-response query performatives	stream-about, stream-all
Basic effector performatives	achieve, □nachieved
Generator performatives	standby, ready, next, rest, discard, generator
Capability definition performatives	advertise
Notification performatives	subscribe, monitor
Networking performatives	register, unregister, forward, broadcast, pipe, break
Facilitation performatives	broker-one, broker-all, recommend-one, recommend-all, recruit-one, recruit-all

KIF – Knowledge Interchange Format

Used to state:

- Properties of things in a domain (e.g., "`Marco is chairman`")
- Relationships between things in a domain (e.g., "`Marco and Paolo are friends`")
- General properties of a domain (e.g., "`All students are registered for at least one course`")

KIF – Knowledge Interchange Format

- “The temperature of room A110 is 21 Celsius”:

```
(= (temperature A110) (scalar 21 Celsius))
```

- “An object is a bachelor if the object is a man and is not married”:

```
(defrelation bachelor (?x) :=
    (and (man ?x) (not (married ?x))))
```

- “Any individual with the property of being a person also has the property of being a mammal”:

```
(defrelation person (?x) :=> (mammal ?x))
```

KQML and KIF

- In order to be able to communicate, agents must have agreed on a common set of terms
- A formal specification of a set of terms is known as an ontology
- The knowledge sharing effort has associated with it a large effort at defining common ontologies — software tools like ontolingua for this purpose
- Example KQML/KIF dialogue…

```
A to B: (ask-if (> (timer pack_1) (timer pack_2)))
B to A: (reply true)
B to A: (inform (= (timer pack_1) 20))
B to A: (inform (= (timer pack_2) 10))
```

KQML examples

A query about the price of a share of IBM stock

```
(ask-one
      :content (TIMER PACK_1 ?timer)
      :receiver Agent_RED
      :language Deliveroo_L
      :ontology Deliveroo_O)

(ask-all
       :content  »timer(PACK_1, [?timer, ?time])"
       :receiver Agent_RED
       :language Deliveroo_LS
       :ontology Deliveroo_OS)
```

FIPA

- The Foundation for Intelligent Physical Agents (IEEE- FIPA standard) —> the FIPA ACL
- Basic structure is quite similar to KQML:
  - performative
    20 performative in FIPA
  - housekeeping
    e.g., sender, etc.
  - content
    the actual content of the message

FIPA - example

```
(inform
      :sender         agent1
      :receiver agent5
      :content  (price good200 150)
      :language sl
      :ontology hpl-auction
)
```

FIPA-ACL performative set

Accept proposal	Inform If	Refuse
Agree	Inform Ref	Reject Proposal
Cancel	Not Understood	Request
Call for Proposal	Propagate	Request When
Confirm	Propose	Request Whenever
Disconfirm	Proxy	Subscribe
Failure	Query If	
Inform	Query Ref	

FIPA

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			

FIPA-ACL message elements

Element	Description
Performative	Denotes the type of the communicative act of the ACL message
Sender	Denotes the identity of the sender (the name of the agent) of the message
Receiver	Denotes the identity of the intended recipient of the message
Reply-To	This element indicates that subsequent messages in this conversation thread are to be directed to the agent named in the reply-to-element, instead of the agent named in the sender element
Content	Denotes the content of the message
Language	Denotes the language in which the content element is expressed
Encoding	Denotes the specific encoding of the content language expression
Ontology	Denotes the ontology(s) used to give a meaning to the symbols in the content expression
Protocol	Denotes the interaction protocol that the sending agent is employing with this ACL message
Conversation-id	Introduces an expression (a conversation identifier) which is used to identify the ongoing sequence of communicative acts that together form a conversation
Reply-with	Introduces an expression that will be used by the responding agent to identify this message
In-reply-to	Denotes an expression that references an earlier action to which this message is a reply
Reply-By	Denotes a time and/or date expression which indicates the latest time by which the sending agent would like to have received a reply

“Inform” and “Request”

- “Inform” and “Request” are the two basic performatives in FIPA.
- All others are macro definitions, defined in terms of these.
- The meaning of inform and request is defined in two parts:
  - pre-condition
    what must be true in order for the speech act to succeed
  - “rational effect”
    what the sender of the message hopes to bring about

“Inform” and “Request”

- For the “inform” performative…
- The content is a *statement*
- Pre-condition is that sender:
  - holds that the content is true
  - intends that the recipient believe the content
  - does not already believe that the recipient is aware of whether content is true or not

“Inform” and “Request”

- For the “request” performative…
- The content is an *action*
- Pre-condition is that sender:
  - intends action content to be performed
  - believes recipient is capable of performing this action
  - does not believe that receiver already intends to perform action

An example of FIPA-ACL

```
(cfp
    :sender (agent-identifier :name BuyerAgent)
    :receiver (set (agent-identifier :name
     SellerAgent))
    :content
       ((action (agent-identifier :name SellerAgent)
       (sell :movie Gladiator))
       (any ?x (and (= ?x (price Gladiator)) (< ?x 20))))
    :ontology movie-ontology
    :language FIPA-SL)
```

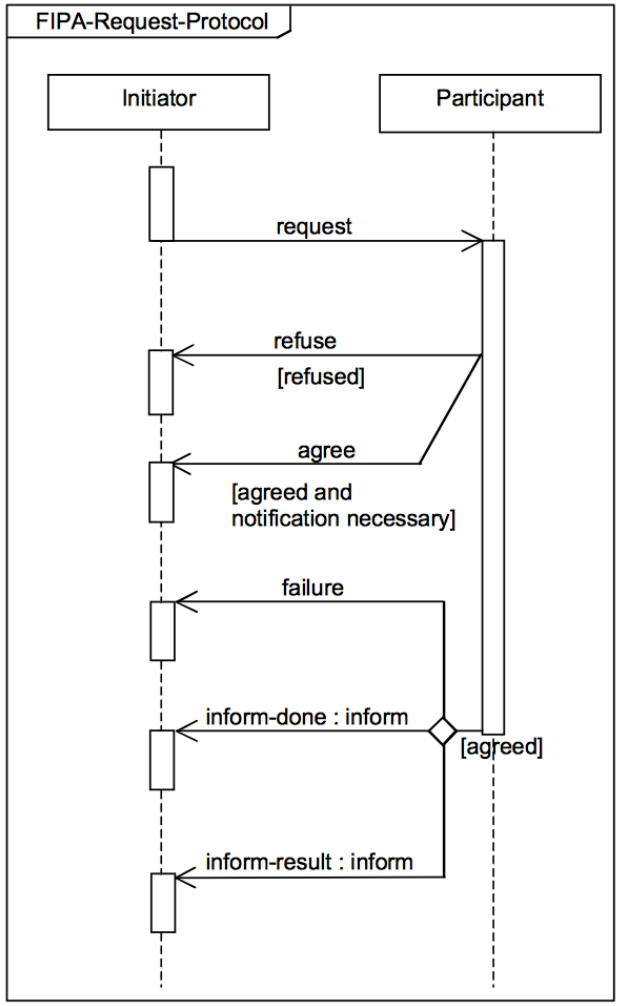
FIPA Semantic Language

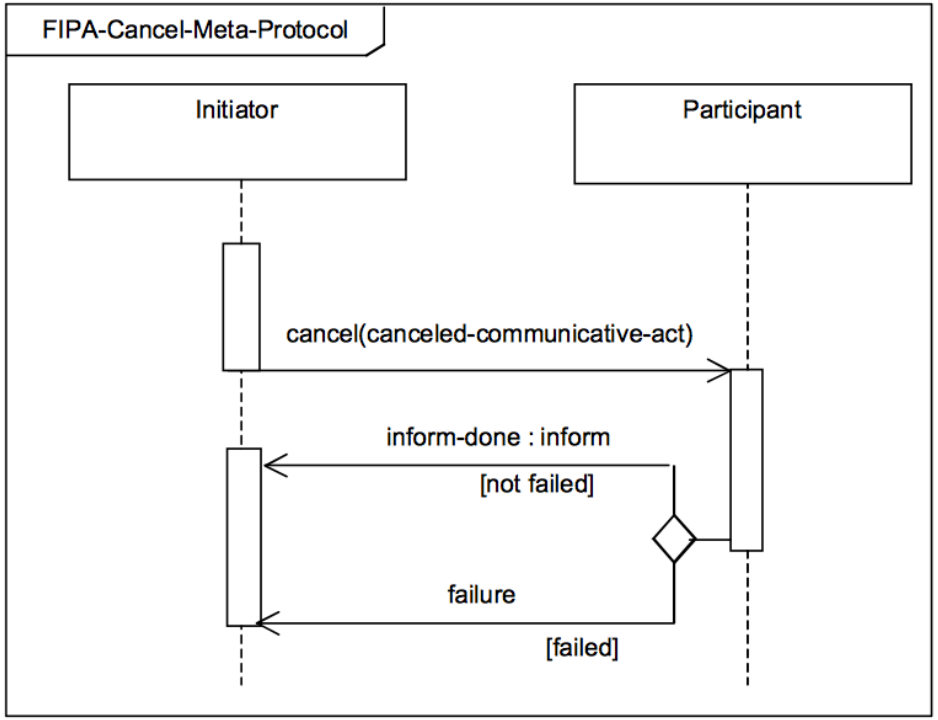
- FIPA Semantic Language (FIPA SL) is a formal language to define the content of the FIPA-ACL.
- FIPA SL can be used to express objects, propositions and actions
  - Object expression is used to declare variables and make assertions.
  - Action expressions describe some action that is either already performed, intended to be performed in the future or is currently being performed.
  - Propositions are used to represent the behavioural aspects of agents like goals, intents, beliefs and uncertainty.
- For example, agents may have persistent goals stated in the form

```
(PG < agent > < expression >)
```

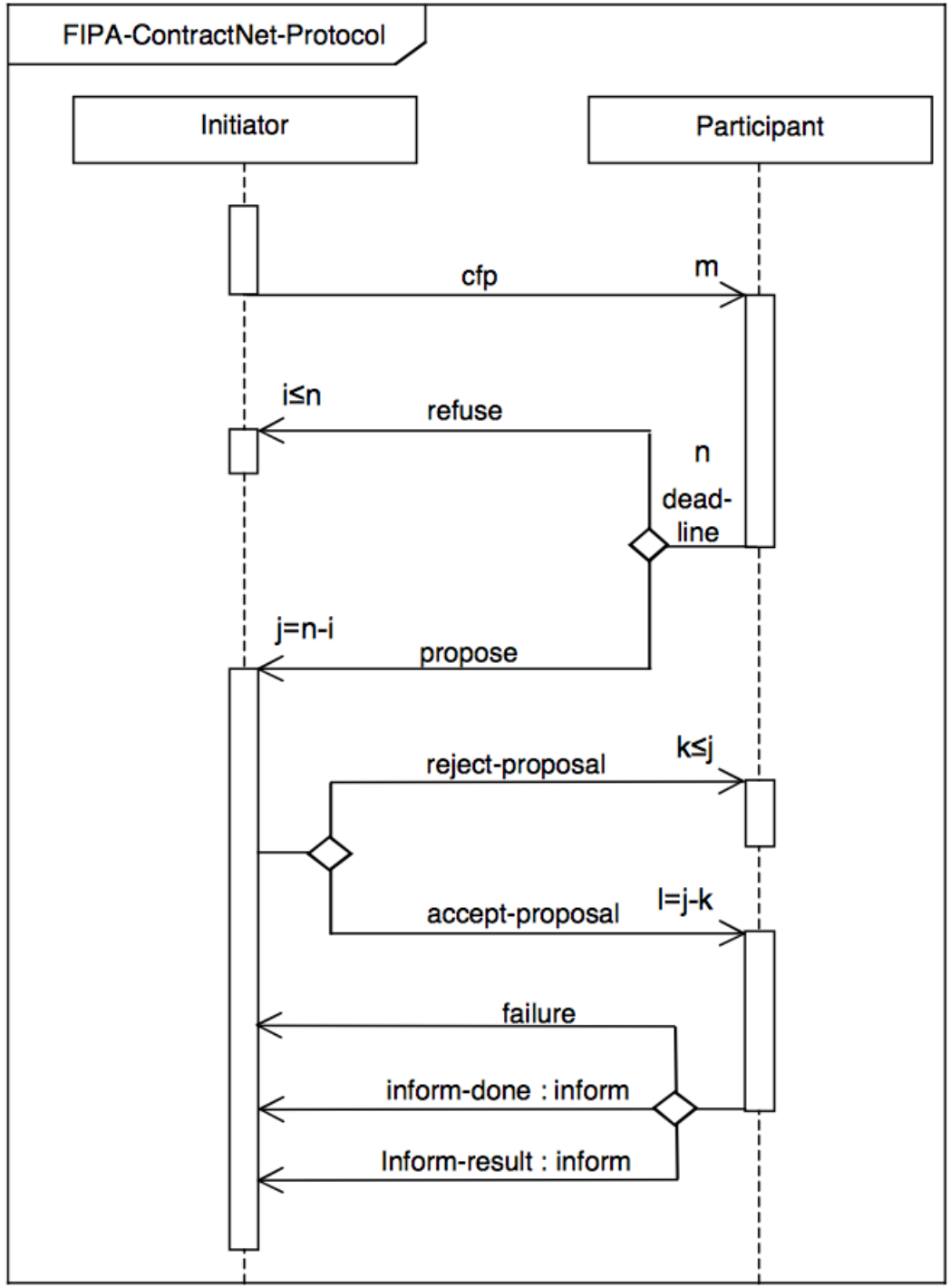
- This states that an agent holds a persistent goal that expression becomes true but will not necessarily possess a plan to achieve this.

FIPA REQUEST protocol



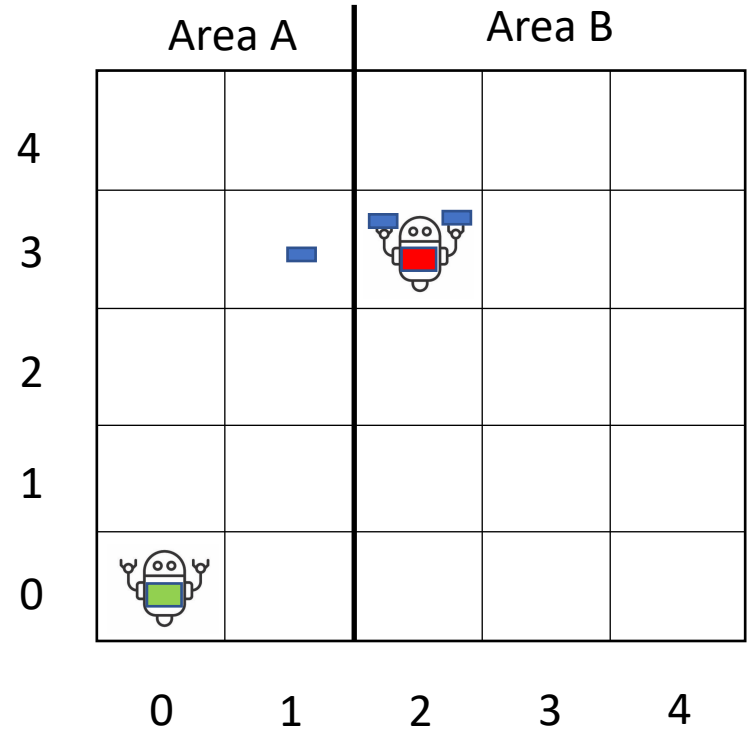


FIPA ContractNet Protocol



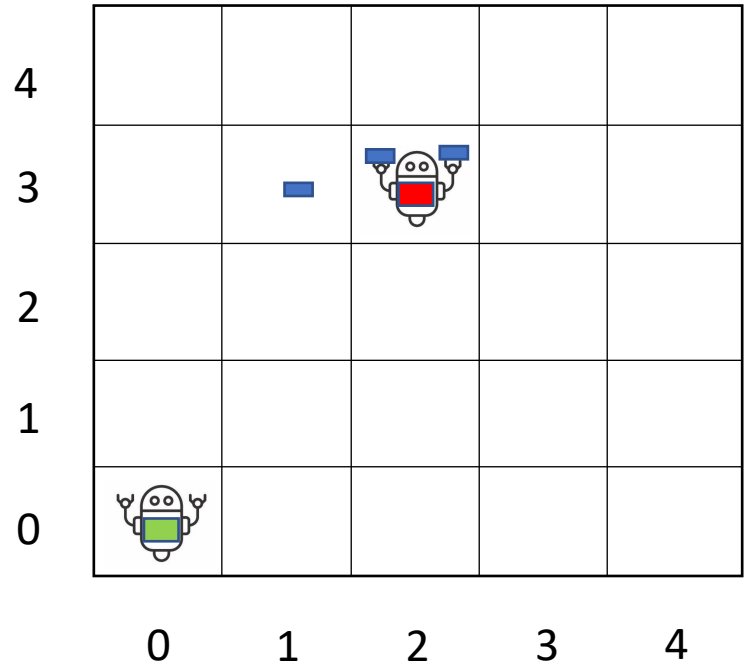
Strategies of cooperation 1

- Agent RED is responsible to collect and deliver packages in AREA B
- Agent GREEN is responsible to collect and deliver packages in AREA A
- Agents have the same goals, but they work in separate areas
- No need for communication
- No interaction
- They cooperate in the sense they have a specific role (task) in the environment (overall objective)



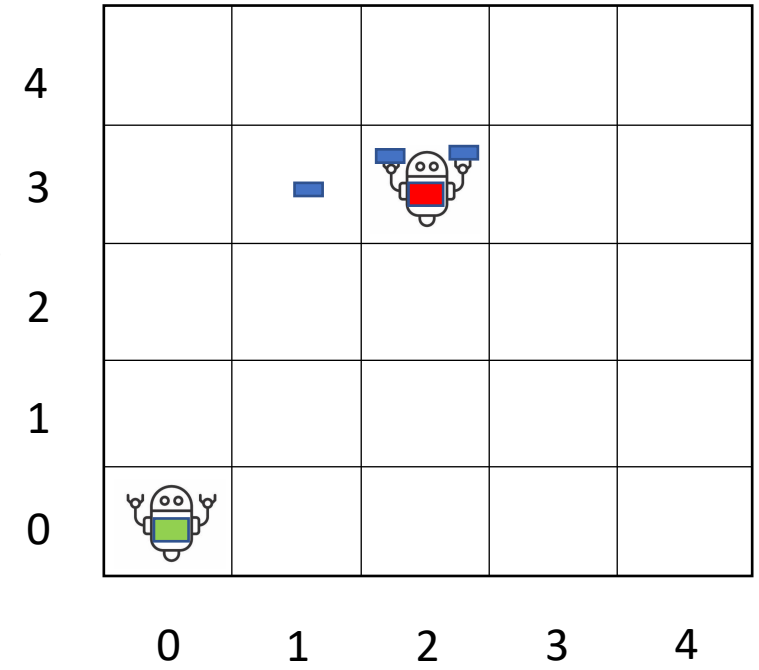
Strategies of cooperation 2

- No limitations in covering area
- Agents share information
  - RED -> GREEN: `In(1,3,pack_1)`
  - RED -> GREEN: `not_intend(pick_up(pack_1))`
- A black board approach can be used
  - Broadcasting



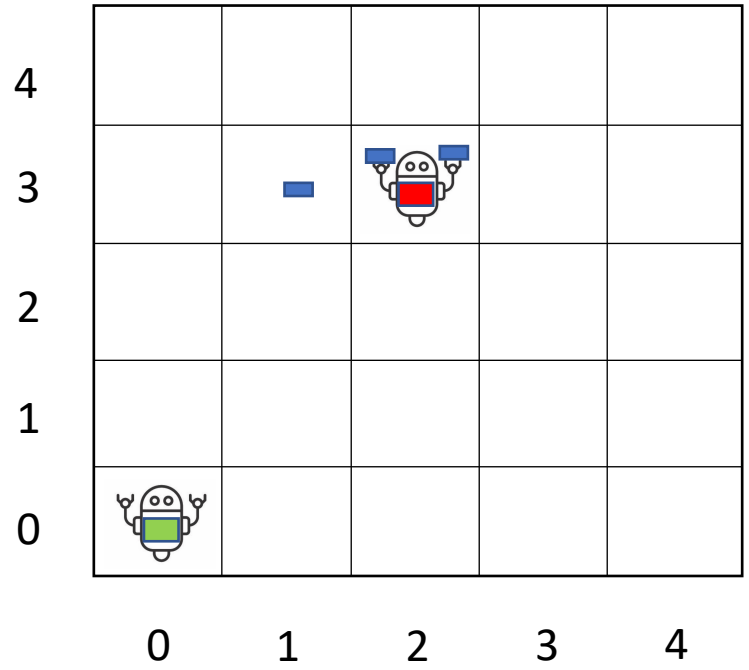
Strategies of cooperation 3

- No limitations in covering area
- Roles: Master and Slave
  - RED -> GREEN: `request(pick_up(pack_1))`
  - GREEN -> RED: `inform(intend(pick_up(pack_1)))`
- More than two agents could be used
  - E.g., supervisor (master) that assign tasks to slave agents



Strategies of cooperation 4 – negotiation

- Negotiation for the fulfilment of the request
  - RED -> GREEN: `request(pick_up(pack_1))`
  - GREEN -> RED: `inform(pick_up(pack_1),10s)`
  - RED -> GREEN: `accept(pick_up(pack_1),10s))`
- Negotiation protocol is need to be designed



Strategies for Deliveroo.js

Define your own strategy for Deliveroo.js

- (**basic**) two agents working in separate areas of the grid and that share information about packages
  - E.g., a package that is in the area of the other agent
- (**medium**) two agents that exchange information about their mental states to coordinate activities and maximize the overall score
  - E.g., if I go to pick up the pack_1 the score will be X, what about you? … ok it is better you go.
- (**advanced**) two agents that share information about their mental states and use a multi-agent plan to maximize the overall score
  - E.g., one of the two agents take the responsibility of managing the intensions of both agents and elaborating a multi-agent plan