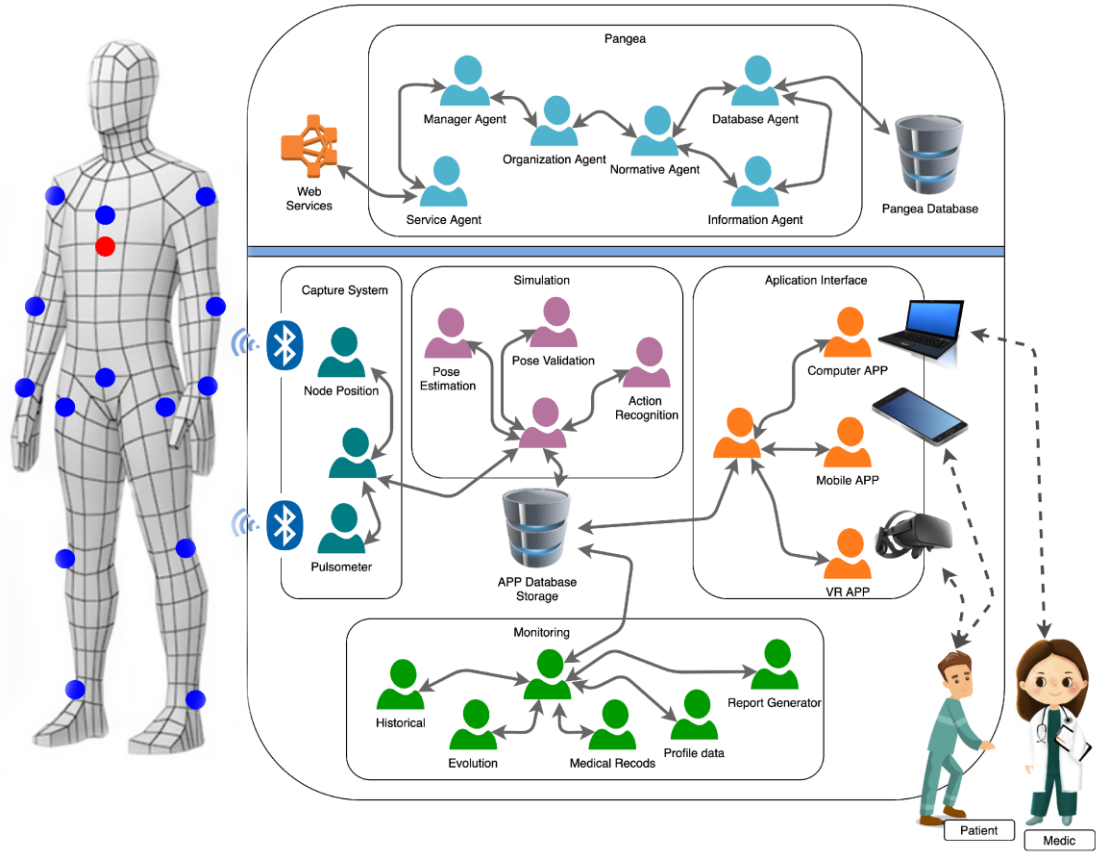


INTRODUCTION TO MULTIAGENT SYSTEMS

SUMMARY

Ramon Mateo Navarro | Master in Artificial Intelligence| 2021-2022

Introduction to Agents and MAS

Delegation and Intelligence imply the need to build computer systems and can act effectively on our behalf.

This implies:

- The hability of computer systems to act independently.
- The ability of computer systems to act in a way that represents our best interests while interacting with other humans or systems

Agent Definition

An agent is a computer system that is located in a dynamic environment and is capable of independent action on behalf of its user or owner (figuring out what needs to be done to satisfy design objectives, rather than constantly being told)

A multiagent system is one that consists of a number of agents, which interact with one another.

In the most general case, agents will be acting on behalf of users with different goals and motivations.

To successfully interact, they will require the ability to cooperate, coordinate, and negotiate with each other, much as people do.

Agent Technologies Categories

- Agent-level
  - Technologies and techniques concerned only with individual agents
- Interaction-level
  - Technologies and techniques concern the communication between agents
  - Communication languages, interaction protocols.
- Organization-level
  - Related to agent societies

Objections to MAS

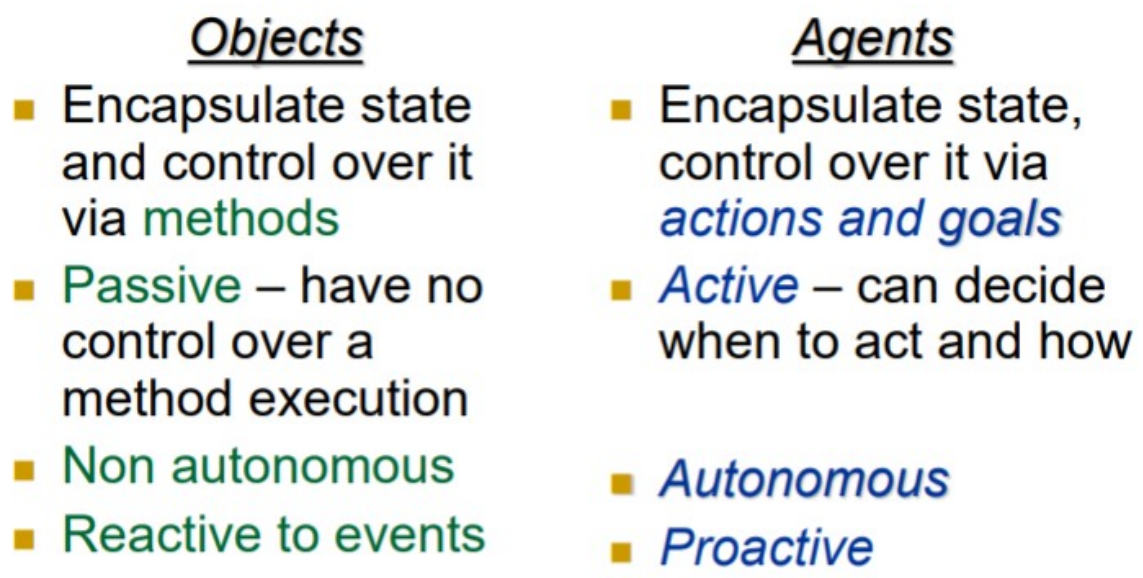
Agents are assumed to be autonomous, capable of making independent decision so they need mechanisms to synchronize and coordinate their activities at run time

We don't need to solve all the problems of Artificial Intelligence (i.e., all the components of intelligence) in order to build really useful agents

Agents vs Objects

Agents are autonomous. They decide for themselves whether or not to perform an action on request from another agent. When a method is invoked on an object, it is always executed.

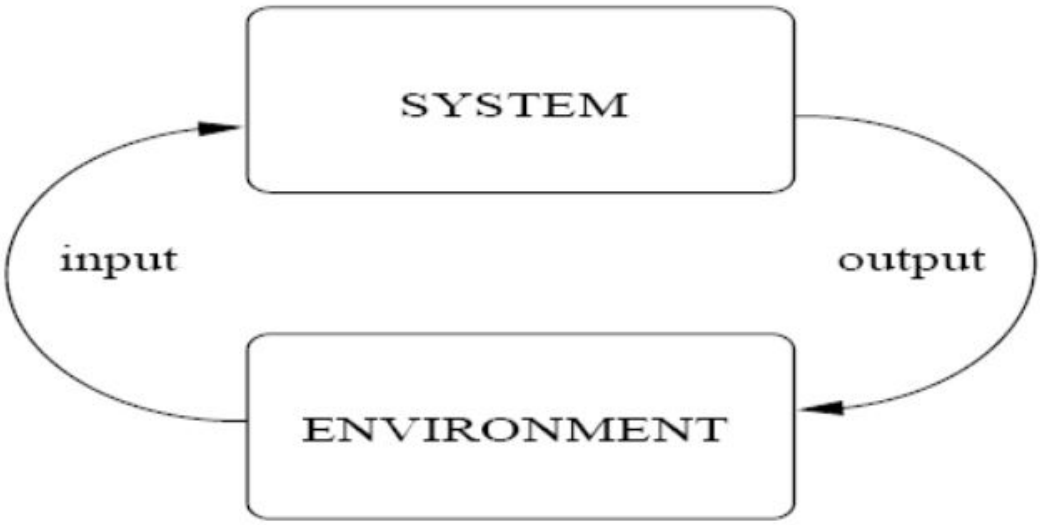
A multi-agent system is inherently multithreaded, in that each agent is assumed to have at least one thread of active control



LECTURE 2: AGENT ARCHITECTURES

Definitions of agent

1. An agent is anything that can be viewed as perceiving its environment through sensors and acting upon that environment through effectors.
2. Autonomous agents are computational systems that inhabit some complex dynamic environment, sense and act autonomously in this environment, and by doing so realize a set of goals or tasks for which they are designed
3. An autonomous agent is a system situated within and part of an environment that senses that environment and acts on it, over time, in pursuit of its own agenda and so as to affect what it senses in the future



An agent has to be able to react in an appropiate way to the dynamic changes in its "environment"

Kinds of environments

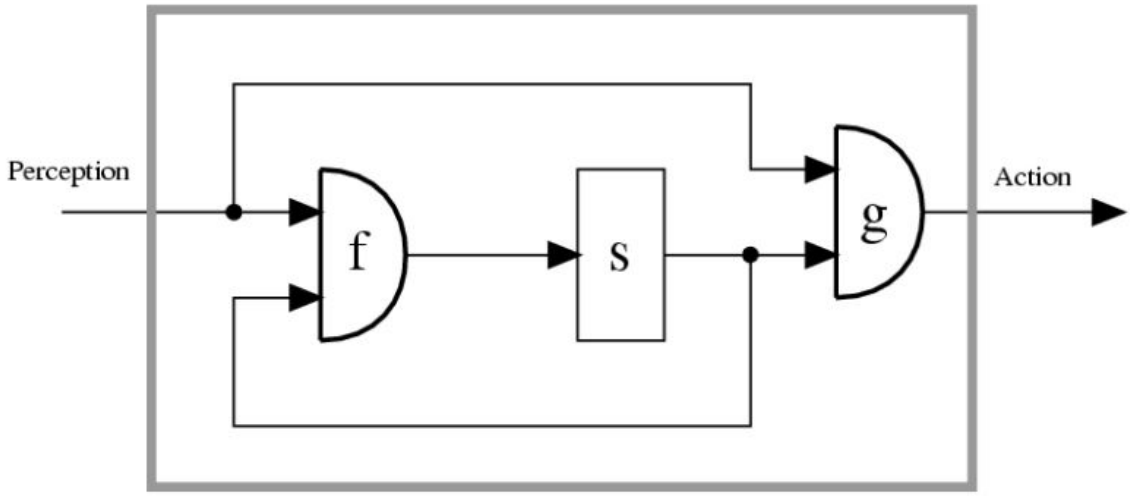
- **Accessible**: agent can obtain complete, accurate, up-to-date information about the environment state.
- **Deterministic environment** : any action has a single guaranteed effect. The physical world can to all intents purposes be regarded as a non-deterministic. **Non-deterministic** environments present greater problems for the agent designer.
- **Episodic**: The performance of an agent is dependent on a number of discrete, independent episodes, with no link between the performance of an agent in different scenarios.
- **Static environment** : Static environment is one that can be assumed to remain unchanged except by the performance of actions by the agent
- **Dynamic environment**: A dynamic environment is one that has other processes operating on it, and which hence changes in ways beyond the agent's control. The physical world and

the Web are highly dynamic environments. It is hard to design and implement agents in dynamic environments

- **Discrete environment** Is discrete if there is a fixed, finite number of actions and percepts in it. Real world is a continuous environment. Discrete environments are much simple than continuous ones.

Agent architectures

Two aspects define how the sensor data and the current internal state of the agent determine the actions (effector outputs) and the future internal state of the agent



f = state update function
s = internal state
g = output function

Types of agent architectures

1. **Reactive:** Focused on fast reactions/responses to changes detected in the environment.
2. **Deliberative (symbolic)**: Focused on a long-term planning of actions, centered on a set of basic goals.
3. **Hybrid:** Combining a reactive side and a deliberative side.

Reactive agents

Intelligence is a product of the interaction between an agent and its environment.

Behaviour-based paradigm.

Example:

- A single ant has very little intelligence, computing power or reasoning abilities.
- The union of a set of ants and the interaction between them allows the formation of a highly complex, structured and efficient system.

Main characteristics

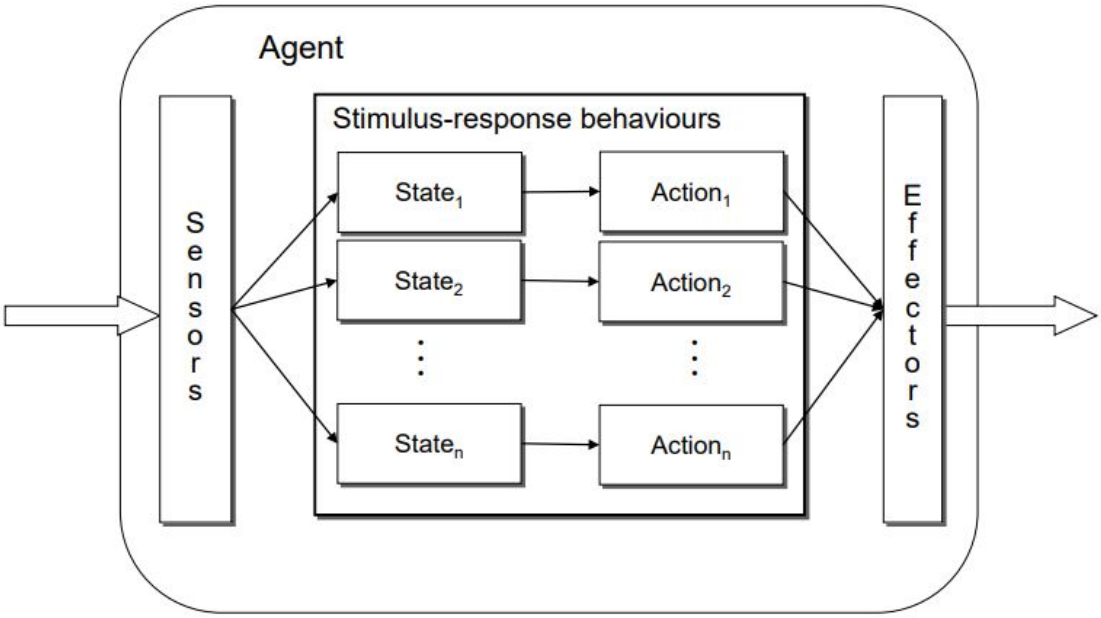
- Simple agents
- Simple interaction
- Complex behaviour patterns appear as a result of the dynamic interactions
- The global behaviour of the system is not specified a prior
- Agents composed of autonomous modules
- Each module manages a given task
    - Sensor, control, computations
- Basic data from sensors

Basic schema of reactive architecture

Reactive behaviour: action rules: **S** → **A** where S denotes the states of the environment, and A the primitive actions the agent is capable of performing

Example:

$$action(s) = \begin{cases} Heater\ off, \text{if teperature is OK in state s} \\ Heter\ on, \text{ otherwhise} \end{cases}$$



Brooks refutal of symbolic AI

- Intelligent behaviour can be generated withou explicit representations.
- Intelligent behaviour can be generated without explicit abstract reasoning.
- Intelligence is an emergent property of certain complex systems.
- Situatedness: ‘Real’ intelligence is situated in the world
    - The world is its best model
    - The world is always up-to-date
    - A model is an abstraction
- Embodiment: 'Real' intelligence requires a physical body.

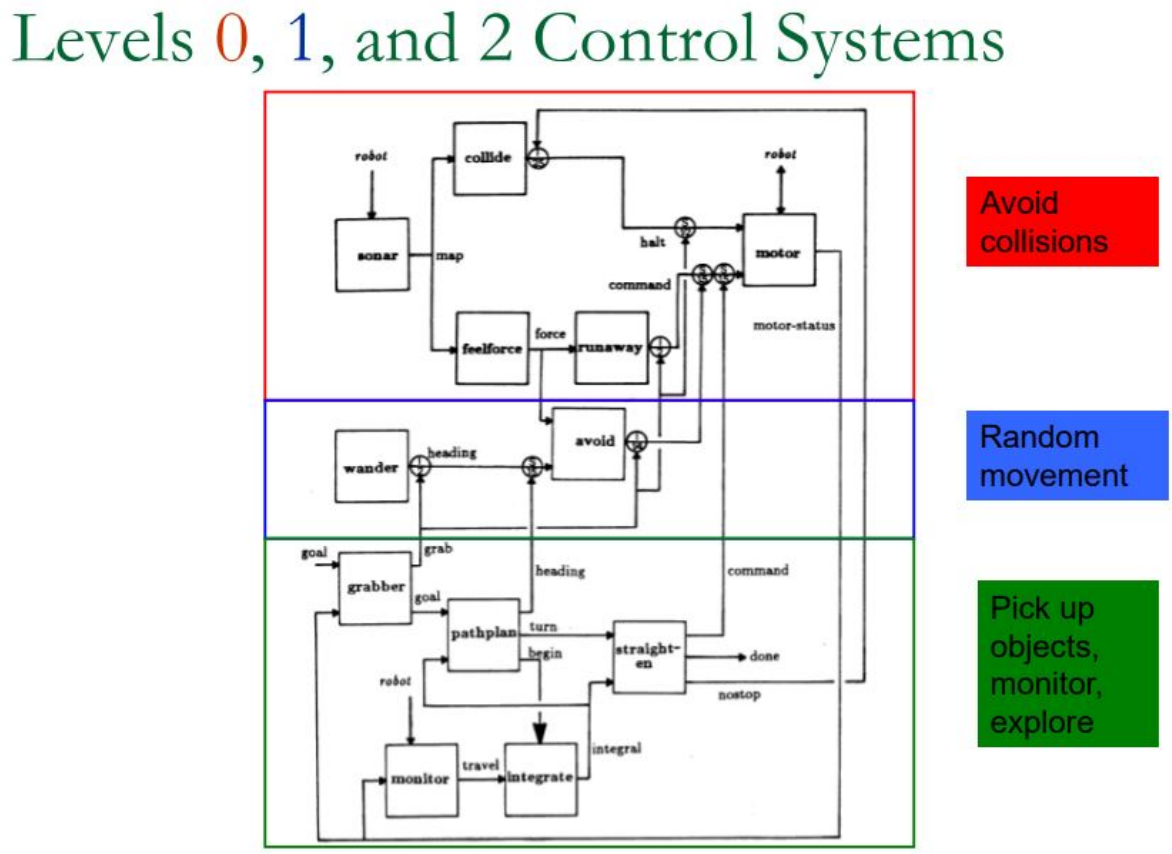
Subsumption hierarchy

- A subsumption architecture is a hierarchy of task-accomplishing behaviours.
- Each behaviour is a rather simple rule-like structure
- Each behaviour 'competes' with others to exercise control over the agent, as different behaviours may be applicable at the same time

Behaviour layers

- Lower layers represent primitive kinds of behaviour (such as avoiding obstacles)
- Higher layers represent more complex behaviours (e.g. identifying an object)
- Lower layers have precedence over layers further up the hierarchy
- The resulting systems are, in terms of the amount of computation they do, extremely simple

Exemple



Advantages of Reactive Agents

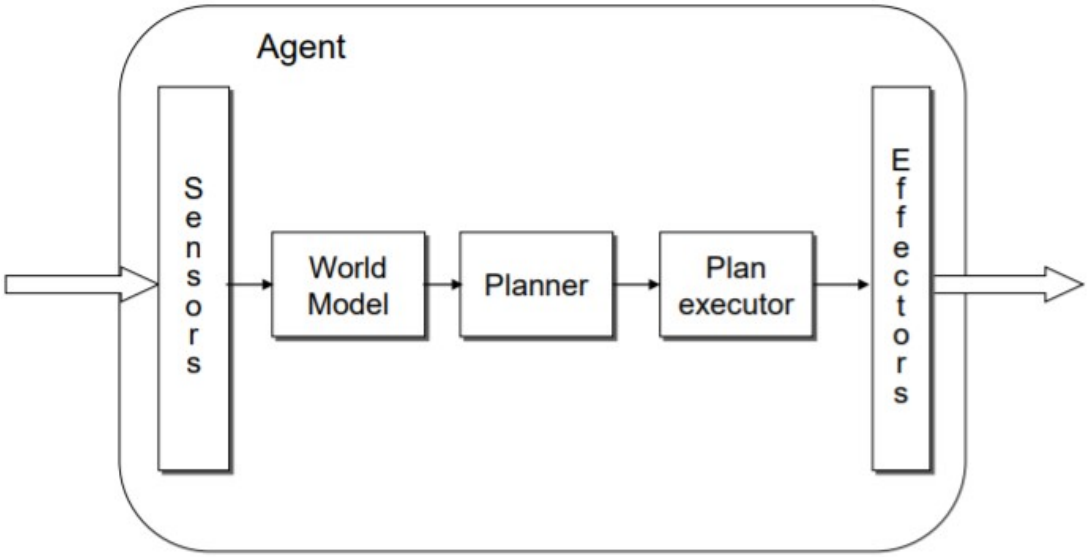
- Simplicity of individual agents
- Flexibility, adaptability
- Ideal in very dynamic and unpredictable environments
- Low computational cost
- Robustness against failure
- Elegance

Limitations of Reactive Agents

- Difficult to make reactive agents that learn dynamic evolution of rules.
- Agents without environment models must have sufficient information available from local environment
- No long-term planning capabilities
- Limited applicability

Deliberative architecture

- Explicit symbolic model of the world
- Decisions are made via logical reasoning, based on patter matching and symbolic manipulation

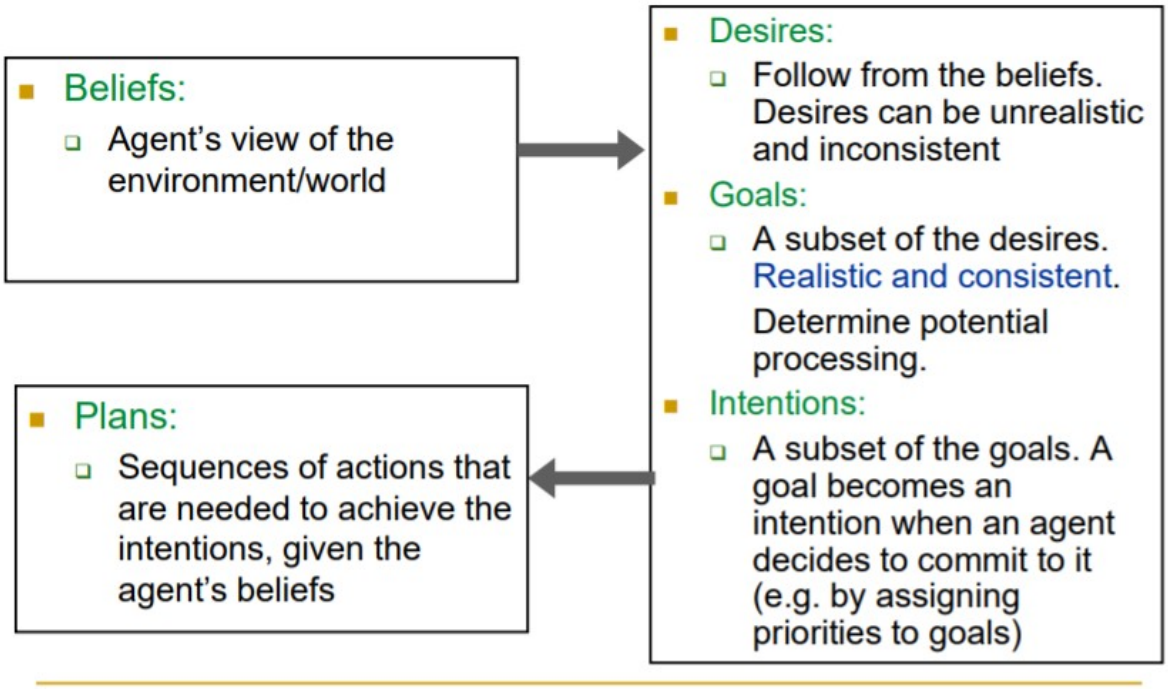


Belief - Desire - Intention (BDI) model

Human practical reasoning

Two activies:

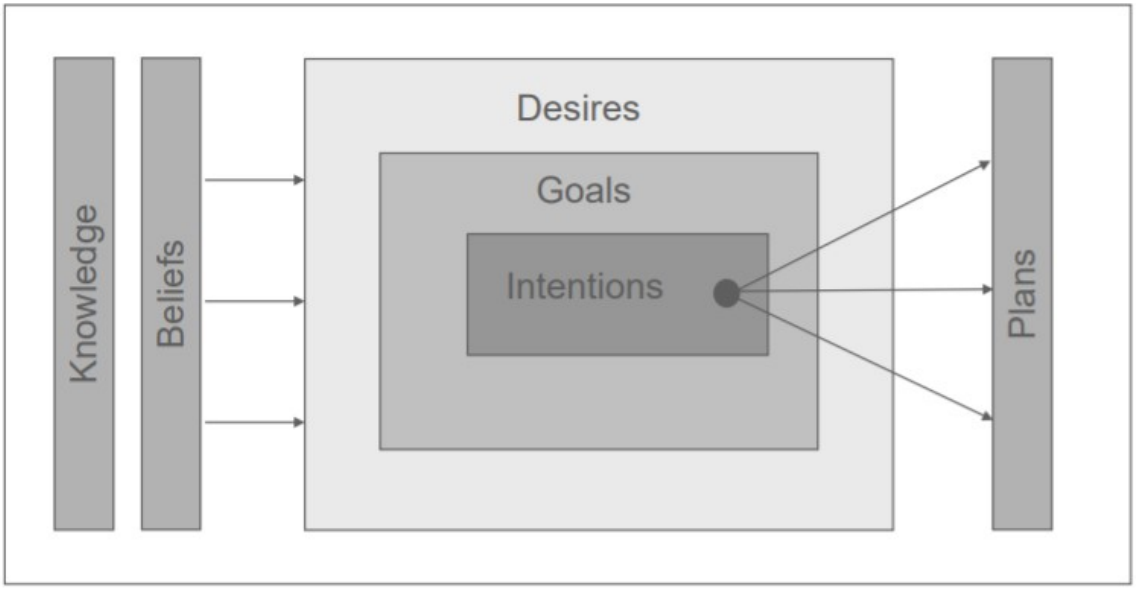
- **Deliberation**: deciding which state of affairs we want to achieve. The outputs are intentions
- **Means-ends reasoning**: deciding how to achieve these states of affairs. The outputs are plans.



BDI plans

Usually have:

- name , goal,
- pre-condition list. List of facts which must be true for plan to be executed.
- delete list. List of facts that are no longer true after plan is performed
- add list. List of facts made true by executing the actions of the plan
- body that is a list of actions.



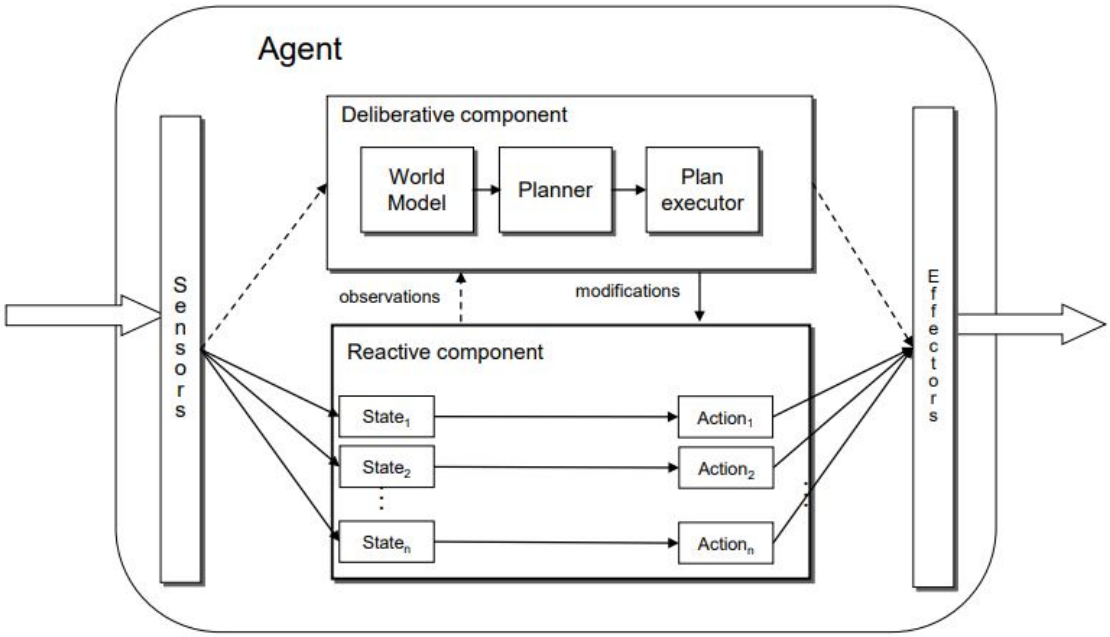
Intentions (&plans) enable the agent to be goal-driven rather than event-driven

By committing to intentions the agent can pursue longterm goals

Problems in the deliberative approach

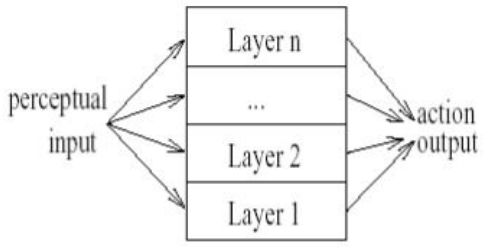
- Dynamic world
  - World changes while planning is being done
- Representation language
  - Expressive enough to be useful in any domain
- Classical planning
  - High computational cost

Hybrid Architectures



Layering techniques

1. Horizontal layering
   - *m* psibble actions suggested by each layer, *n* layers

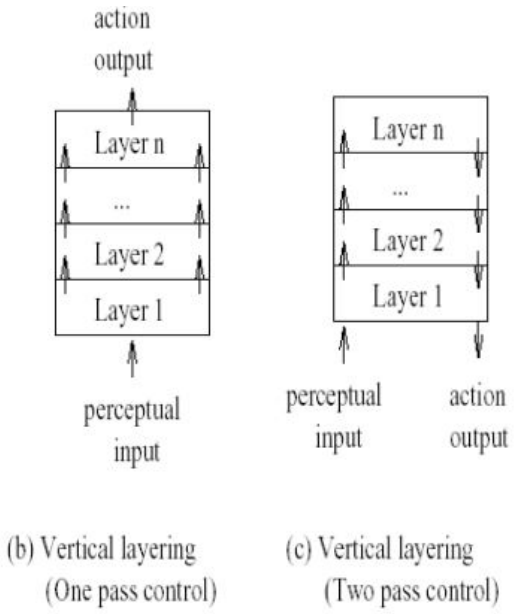


(a) Horizontal layering

Each layer is directly connected to the sensory input and action output, and suggests actions to perform.

Central control system can be complex, because there are $O(m^n)$ possible options to be considered

2. Vertical layering
   - *m* possible actions suggested by each layer, *n* layers



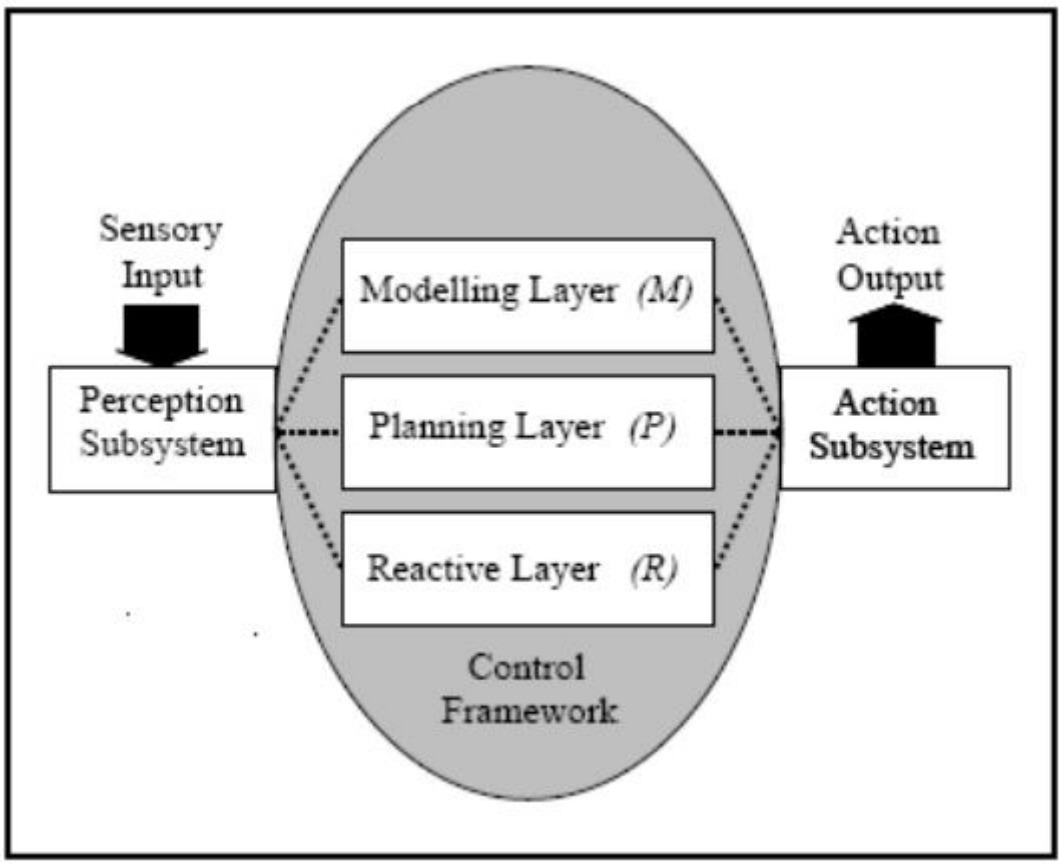
O(mn) interactions between layers

Sensory input and action output dealt with by one layer.

Not fault tolerant to layer failure

Touring machines architecture

Consists of perception and action subsystems, which interface directly with the agent's environment, and three control layers, embedded in a control framework, which mediates between the layers.

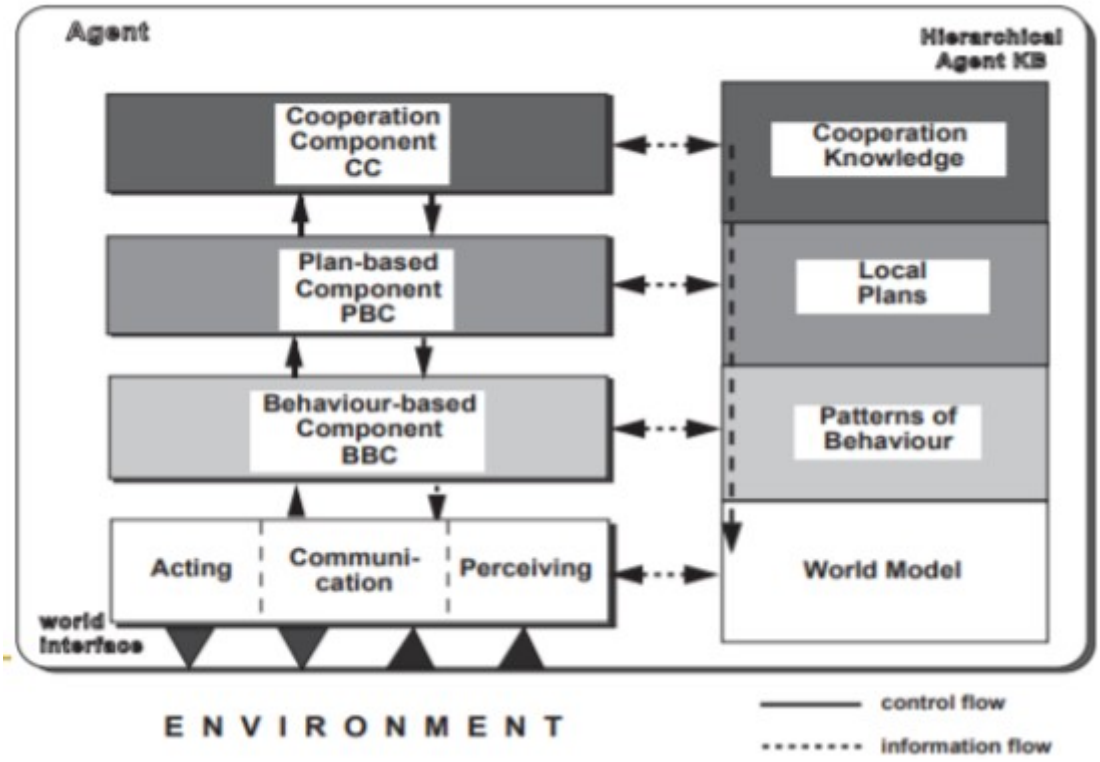


- **Reactive layer** : implemented as a set of situation-action rules (subsumption architecture)
- **Planning layer** : constructs plans and selects actions to execute in order to achieve the agent's goals
- **Modeling layer** : contains symbolic representations of the 'cognitive state' of other entities in the agent's environment

InterRRaP .

Integration of Reactive Behaviour and Rational Planning

Vertically layered



Behaviour layer

- Reactive part of the architecture
- Works with the world model
- Only one level interacts with the real world
- Has a set of "situation → action" rules

Planning layer (PBC)

- Works with the mental model
- Standard deliberative level
- Implements local behaviour

Cooperative planning layer

- Works with the social model
- Allows planning and cooperation with other agents

Critiques to hybrid architectures

- Very specific, application dependent
- Unsupported by formal theories

LECTURE 3: AGENT PROPERTIES

Flexibility

Computer system capable of flexible action in some dynamic complex environment.

- Adaptative
- proactive
- social

Reactivity

System that responds to changes that occur in environment. If a program's environment is guaranteed to be fixed, the program can just execute blindly.

Ways to achieve reactivity

- Reactive architectures
- Deliberative architectures
- Hybrid architectures

Aspects of proactiveness

Reacting to and environment is relatively easy so agents should be proactive.

- Generating and attempting to achieve goals.
- Behavior not driven solely by event
- Taking the initiative when appropriate
- Executing actions/giving advice/making recommendations/making suggestions
- Recognizing opportunities on the fly

Example

- Set of agents embedded in the home of an old or disabled person
- Detects the movement of the person around the house and the actions he/she performs
- Learns the usual daily patterns of behaviour
- Can detect abnormal situations, and proactively send warnings/alarms to family/health services

Social Ability

The real world is a multi agent environment. Some goals can only be achieved with the cooperation of others.

Social ability in agents is the ability to interact with other agents visa some kind of agent-communication language.

Requirements for communication

- Agent communication language
  - FIPA-ACL
  - Message types
  - Message attributes
- Agent communication protocols
- Languages to represent the content of the messages between agents

High-level activities

Coordination

- How to divide a task between a group of agents
- Distributed planning

Cooperation

- Share intermediate results
- Share resources

Negotiation (e-commerce)

- Agree conditions in an economic transaction
- Find the agent that can provide a service with the best conditions

Other aspects related to communication

- Security issues
  - Authentication
  - Encryption
- Trust
  - Reputation models

Rationality

- An agent will act in order to achieve its goal.
- It will not act in such a way as to prevent its goals from being achieved
- For instance, it will not apply deductive procedures without purpose

Reasoning capabilities

- Knowledge base with beliefs on the world
- Capacity to make plans

This is the characteristic that distinguishes an intelligent agent from a more "robotic" reactive-lie agent

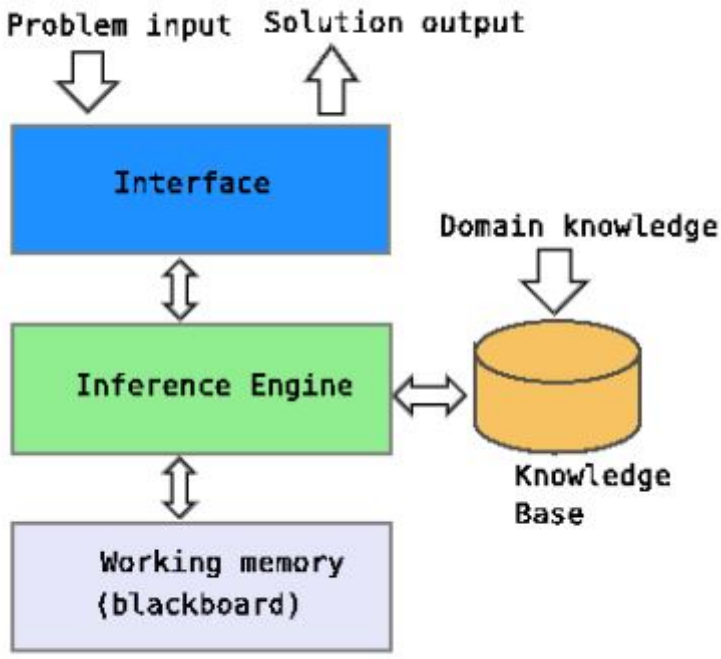
Kinds of reasoning in AI

Knowledge-based systems / expert system

- Reasoning techniques specialized in the system's domain

Rule-based systems

- Knowledge is represented as a set of rules

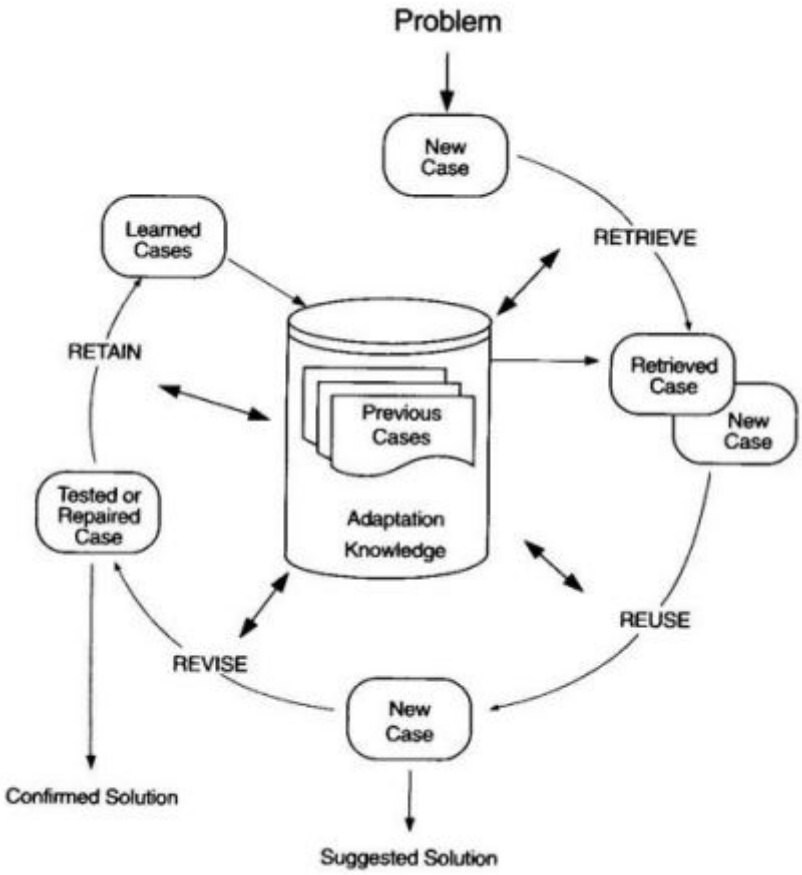


Case-based reasoning

Using similarity to solved problems

Approximate reasoning

Fuzzy logic, Bayesian networks, probabilities...



Learning for improving

- Make less mistakes
- Do not repeat computations performed in the past
- Find solutions more quickly
- Find better solutions
- Solve a wider range of problems
- Learn user preferences and adapt the behavior accordingly

Advantages of learning systems

- Can adapt better to a dynamic environment, or to unknown situations
- Can leverage previous positive/negative experiences to act more intelligently in the future

Autonomy

Ability to pursue goals in an autonomous way, without direct continuous interaction/commands from the user.

Given a vague/imprecise goal, the agent mus determine the best way to attain it.

Autonomy requirements

To have autonomy, it is necessary for and agent to have a control on its own actions (an agent cannot be obliged to do anything). Have a control on its internal state (agent state cannot be externally modified by another agent) and have the appropriate access to the resources and capabilities needed to perform its tasks.

Autonomy limitations

Sometimes the user may restrict the autonomy of the agent. For instance, the agent could have autonomy to search in Internet for the best place to buy a given book but the agent could not have the autonomy to actually buy the book, using the credit card details of the user

Issues

- **Legal issues**: Who is the responsible of the agent's actions
- **Ethical issues**: What extent should decisions be delegated to computational agents

Temporal continuity

Agents are continuously running processes. Running active in the foreground or sleeping/passive in the background until certain message arrives

Mobility

Agent can move physically through a network to another computer and continue their execution there.

In most applications the idea is to go somewhere to perform a given task and then come back to initial host with the obtained results.

***Note**: See example in Lecture 3 Slide 28*

Problems of mobile agents

- **Security** : How can I accept mobile agents in my computer?
- **Privacy** : Is it secure to send an agent with the details of my credit card or with my personal preferences?
- **Technical management** : Each computer has to be able to "pack" and agent, send it and receive agents form other machines.

Other properties

- Benevolence: An agent will always try to do what is asked of it
- Veracity: An agent will not knowingly communicate false information
- Character: Agents must seem honest, trustable
- Emotion: Agents mus exhibit emotional states.

Relationships between properties

- More learning → more flexibility
- More learning → more autonomy
- More reasoning → more rationality
- More reasoning → more proactivity
- Less autonomy → less proactivity

Conclusions

It is almost impossible for an agent to have all those properties.

Most basic properties:

- Autonomy
- Reactiveness
- Reasoning and learning
- Communication

LECTURE 4: AGENT TYPES

How to classify Agents

An initial classification was made based on the properties that are emphasized in a particular agent , reasoning, mobility etc... or by purpose

Classifying agents by purpose

Collaborative Agents

- Emphasize communication and cooperation with other agents
- Typically operate in open multi-agent environments (MAS)
- Negotiate with to reach mutually acceptable agreements during cooperative problem solving
- Collaborative agents are usually deliberative agents (BDI model), with some reasoning capabilities

Applications

- Provide solutions to physically distributed problems (air-traffic control etc...).
- Provide solutions to problems with distributed data sources. (different offices of a multi-national business)
- Provide solutions that need distributed expertise
    - health care provision (HeCaSe)

Interface Agents (or User Agents)

- Emphasize autonomy and learning in order to perform tasks for their owners
- Support and provide proactive assistance to a human that is using a particular application or solving a certain problem
    - Anticipate user needs
    - Make suggestions
    - Provide advice

Interface Agents have a limited cooperation with other agents, reasoning and planing capabilities.

Interface Agents need to learn about user. At the beginning of the service it could not be very helpful but with the experience it will improve.

Applications

- Mail management
- Scheduling meetings
- Internet browsing
- News filtering agent

Problems of interface agents

- Slow learning curve (require many examples)
- Learning from scratch

Information Agents

- Software agents that manage the access to multiple, heterogeneous and geographically distributed information sources.
- Information agents = Internet agents
- Main task: proactive acquisition, mediation and maintenance of relevant information for a user/other agents

Applications

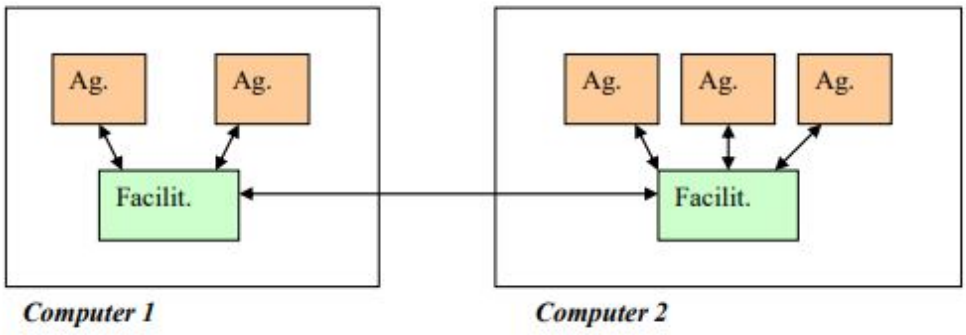
- Information acquisition and management
    - Provide transparent access to different information sources
    - Monitor information sources
    - Update relevant information on behalf of the user
    - Personalized presentation of information
        - Learn automatically user preferences
        - Adapt dynamically to changes in user preferences

Information agents must help in the task of information retrieval

---

Facilitator agents

- Flat system
    - Each agent can talk directly to any other agent
- Federated system
    - There are special agents that manage the connection of the system with the users and the communication between the agents

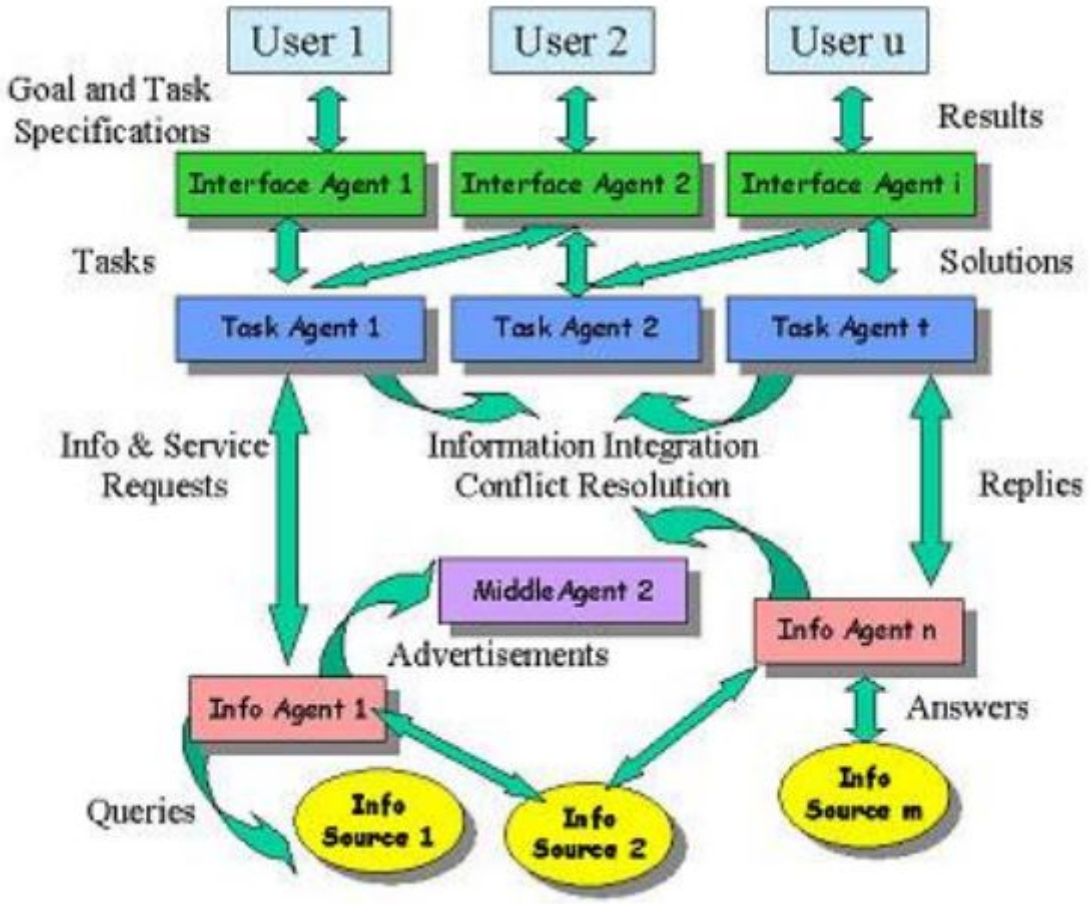


Agents do not communicate directly, but through facilitators and can help agents to find other agents in the system that provide some services

RETSINA agents

- Interface agents: input/output, learn from user actions
- Task agents: encapsulate task-especific knowledge, used to perform/request services to/from other agents/humans *
    - They can coordinate their activites

- Middle agents: infrastructure service (e.g. they know the services provided by other agents)
- Information agents: monitor and access one or more information sources



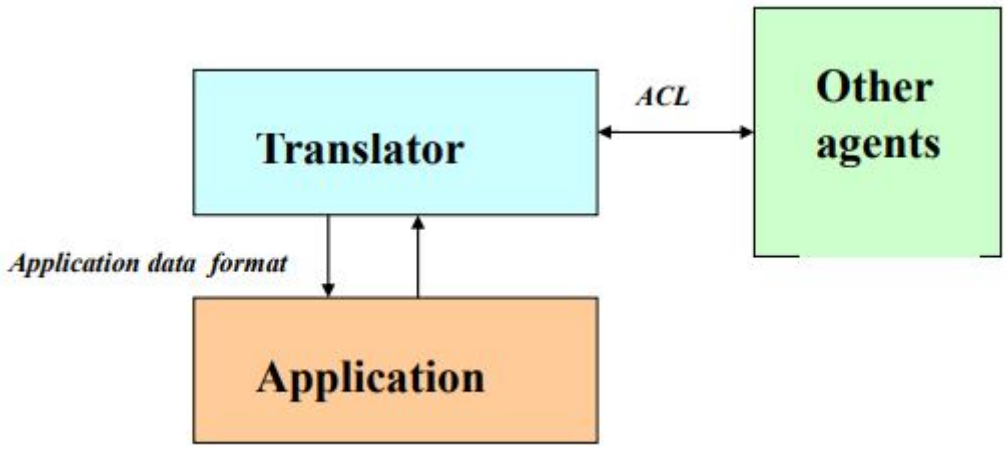
Broker agents: in addition to facilitate the connection with users or providers, they can:

- Store data about the resources available (cache)
- Select which agents should be contacted
- Collect, merge and filter answers
- Coordinate with other brokers

Translators and Wrappers

1.Translators

Agentification is the ways to transform a standard application into an “agent” that can be integrated and participate in a multi-agent system of collaborative agents to help to solve complex distributed tasks

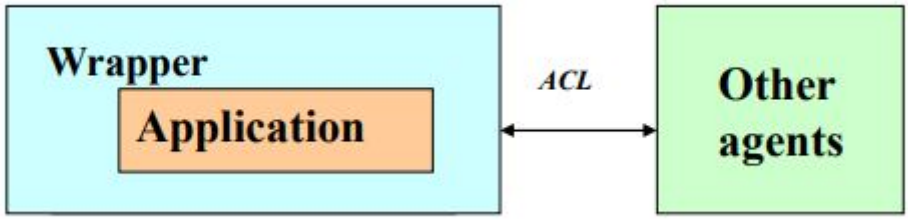


- "Bridge" between the application and the other agents
- Takes the messages of other agents, and translates them to the program communication protocol (and the other way round)

Advantages of translators

- The code of the application does not have to be modified
- We only need to know the inputs/outputs of the application
- The same translator can be used to agentify different applications/resources

2.Wrapper



Positive aspects

- More computationally efficient than translators

Negative aspects

- We need access to the application source code
- It may not be easy to understand/modify the code of a complex application
- A wrapper is not reusable as a translator
    - One wrapper for each program to be agentified

Note: Check L4 35 . Rewrite code

Agent Communication

Remember that Multi-agent Systems are a collection of deliberative, autonomous, collaborative agents, that may communicate and cooperate with each other to solve complex distributed problems.

Each agent is specialised in the solution of a particular kind of problems

The process of solving a complex problem is reduced to solving easier subproblems

Efficiency

Different agents are working at the same time in different parts of a problem.

Reliability

Avoid single point of failure in centralised systems. We can have redundancy. If an individual agent fails, the other agents can take its work and re-distribute it dynamically

Task descomposition

- Divide a complex problem into a set of tasks
- Decompose a large tasks into a subtasks that can be tackled by different agents
- Tasks can be totally independent or can have some relationships

Task allocation

Assign the subproblems to different agents

Easy case: there is only one agent capable of solving each subproblem

Interesting case: the same subproblem can be solved by different kinds of agents

Task accomplishment

Each agent solves its assigned subproblems.

This phase can be done without communication if the problems are independent

In some cases,there may be dependencies between subproblems

Conflict management

Example

- Two sub-solutions are incompatible
- Conflicts in the use of shared resources

Agents have to communicate with each other to solve these situations

Agent support in task execution

Task sharing

An agent can request the help of other agents to solve a particular task

Result sharing

Use intermediate results obtained by other agents.

Agents can provide intermediate subsolutions to help other agents in their work

Result Synthesis

Put together the results of all agents to find the complete solution

Summary Questions about the desing of a collaborative MAS

- *Who* decomposes the problem?
  - One agent / all agents / user
- *How* is the problem decomposed?
  - 1 level / n levels
  - Static – beginning / Dynamic – execution time
- Who assigns tasks to agents? How is the assignment made?
  - Static – beginning / Dynamic – execution time
- What kind of *interactions* can exist between agents when they are solving tasks?
  - Conflict detection
  - Conflict resolution
- How are tasks shared?
- How are results shared?
- Who/how merges the subresults to get the final result?

Communication

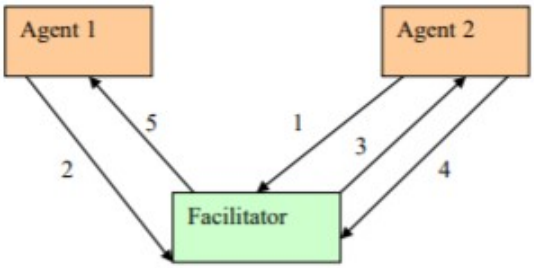
Categories

- Mechanistic manner
  - via the type of sendee-addressee link
  - via the nature of the medium
- meaning-based manner
  - via the type of intention

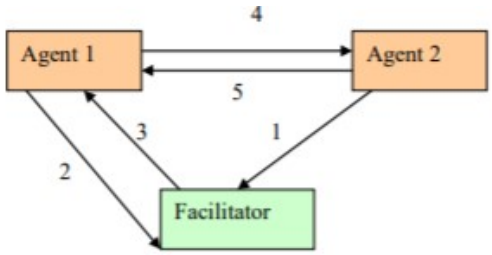
Addressee Link

- Point to point
  - Agent talks directly to another agent
- Broadcast
  - Agent sends some information to a group of agents
- Mediated
  - The communication between two agents is mediated by a third party

Communication via facilitator



1. Agent_2 tells the facilitator the services it provides
2. Agent_1 asks to the facilitator who can provide a certain service with some conditions
3. The facilitator requests the service to agent_2
4. Agent_2 provides the answer
5. The facilitator sends the answer to agent_1

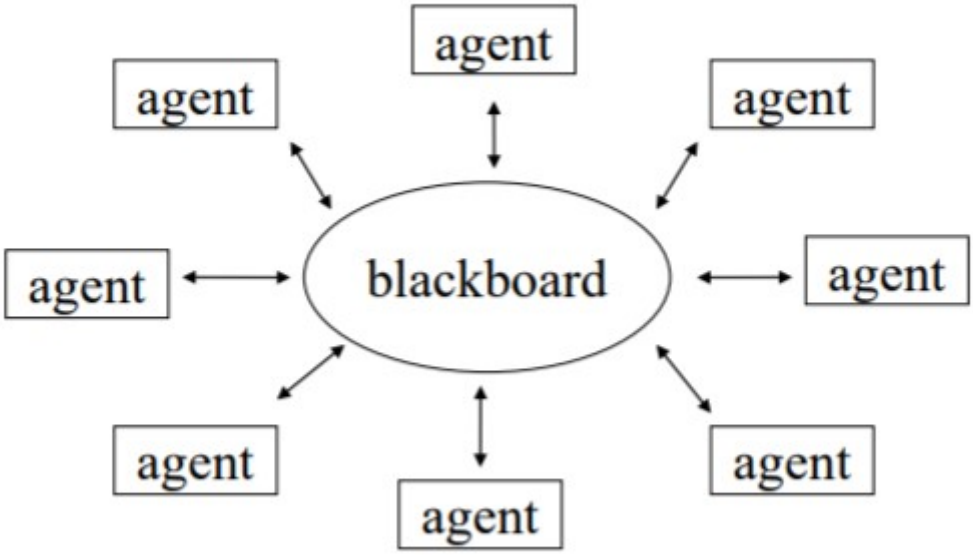


1. Agent_2 tells the facilitator the services it provides
2. Agent_1 asks to the facilitator who can provide a certain service with some conditions
3. The facilitator tells agent_1 that agent_2 can do that service
4. Agent_1 requests the service from agent_2
5. Agent_2 sends the answer to agent_1

Nature of the medium

- Direct routing
  - Message sent directly to other agent(s) with no interception
- Signal propagation routing
  - Agent sends signal whose intensity decreases according to distance
- Public notice routing
  - Blackboard systems

Blackboard systems



Information in blackboard

- Data of the common problem
- Current state of the solution
- Next subproblems to be solved

- Requests of help
- Present task of each agent
- Intermediate results

Uses of blackboard

- Detect conflicts
- Notice incompatible solutions
- Share results
- Share tasks

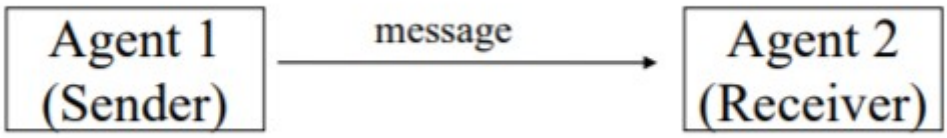
Positive aspects

- Flexible mechanism for communication/cooperation
- Independent of cooperation strategy

Negative aspects

- Centralised structure
- Single point of failure
- Everyone has to read/write info from the blackboard

Message passing



Types of speech act

- *inform* other agents about some data
- *query* others about their current situation
- *answer* questions
- *request* others to act
- *promise* to do something
- *offer* deals
- *acknowledge* offers and requests

Examples

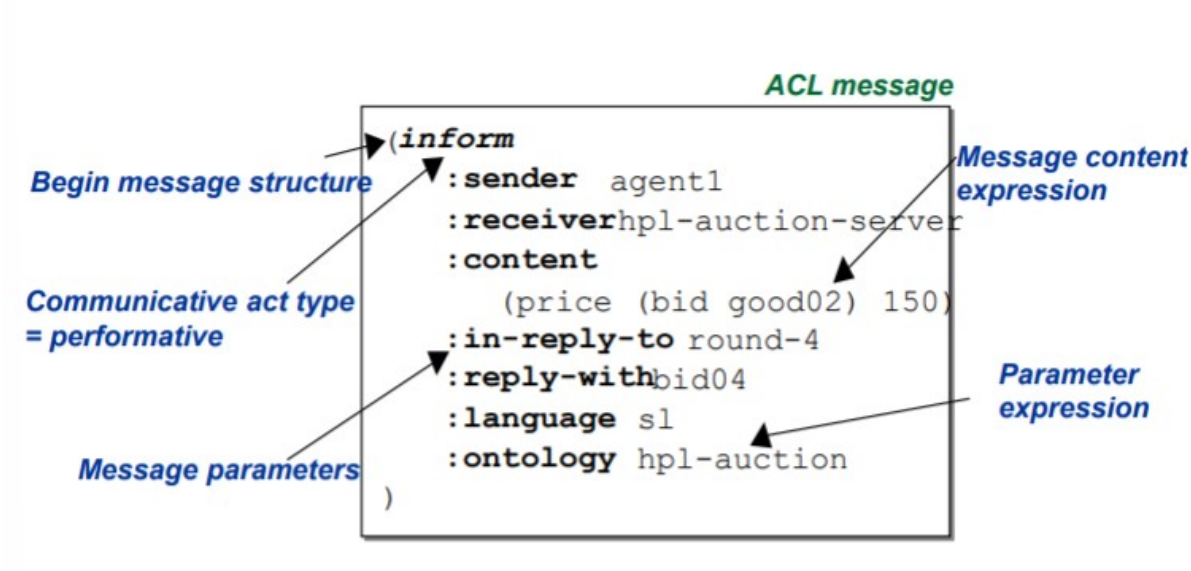
Is the door open?	query
Open the door (for me)	request
OK! I'll open the door	agree
The door is open	inform
I am unable to open the door	failure
I don't want to open the door	refuse
Tell me when the door becomes open	subscribe
Does anyone want to open the door?	CFP (call for proposals)
I can open the door for you... at a price	propose
Door? What's that? I don't understand...	not-understood

Speech act have two components, a performative verb (request, inform etc...) and a propositional content ("the door is open")

Standards in commnication allow different groups to write cooperating agents.

FIPA (Foundation for Intelligent Physical Agents)

Contents



Parameters

- :sender - who sends the message
- :receiver - who is the recipient of the message
- :content - content of the message
- :reply-with - identifier of the message
- :reply-by - deadline for replying the message
- :in-reply-to - identifier of the message being replied
- :language – language in which the content is written
- :ontology - ontology used to represent the domain concepts
- :protocol - communication protocol to be followed
- :conversation-id - identifier of conversation

1.INFORM

- Content: Statement
- The sender informs the receiver that a given proposition is true

2.Query-if

- Content: proposition
- The sender asks the receiver if a given proposition is true

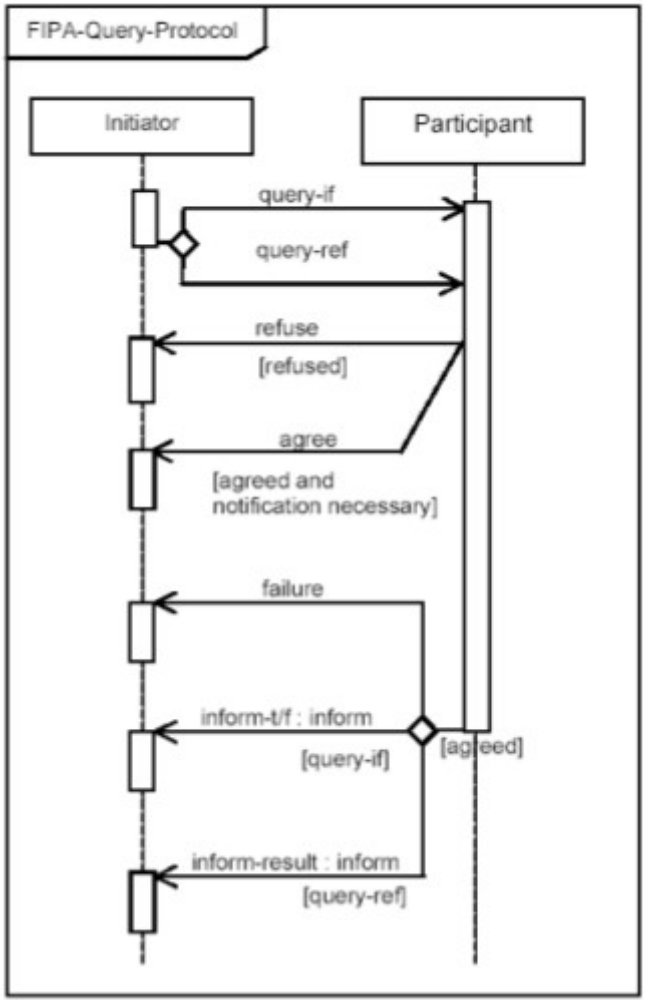
3.Query-ref

- Content: descriptor
- The sender asks the receiver for the object referred by the descriptor

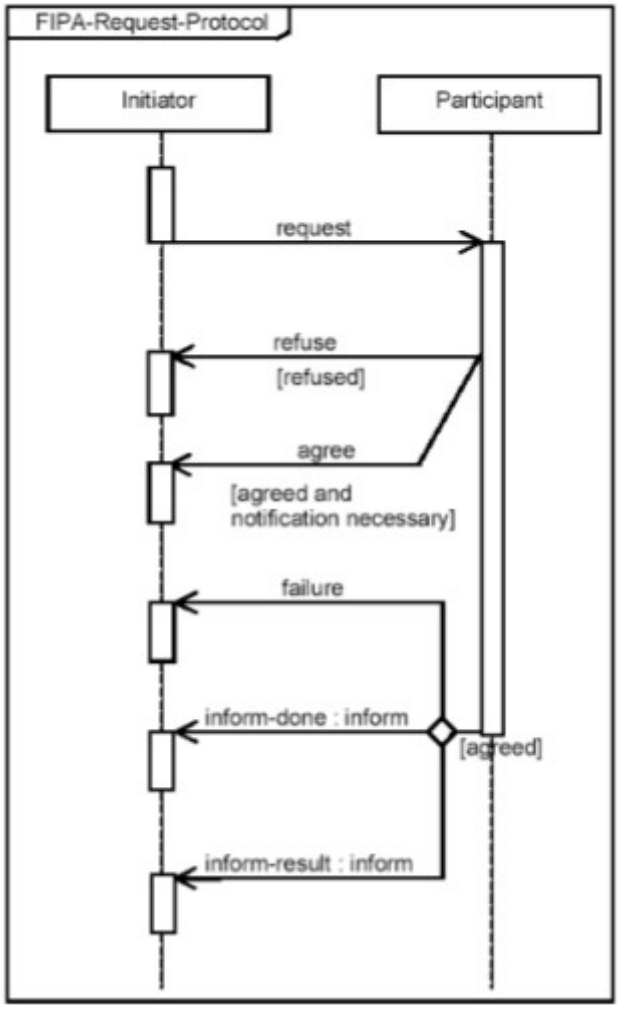
4.Request

- Content: Action
- The sender requests the receiver to perform some action

FIPA-Query protocol



FIPA-Request protocol

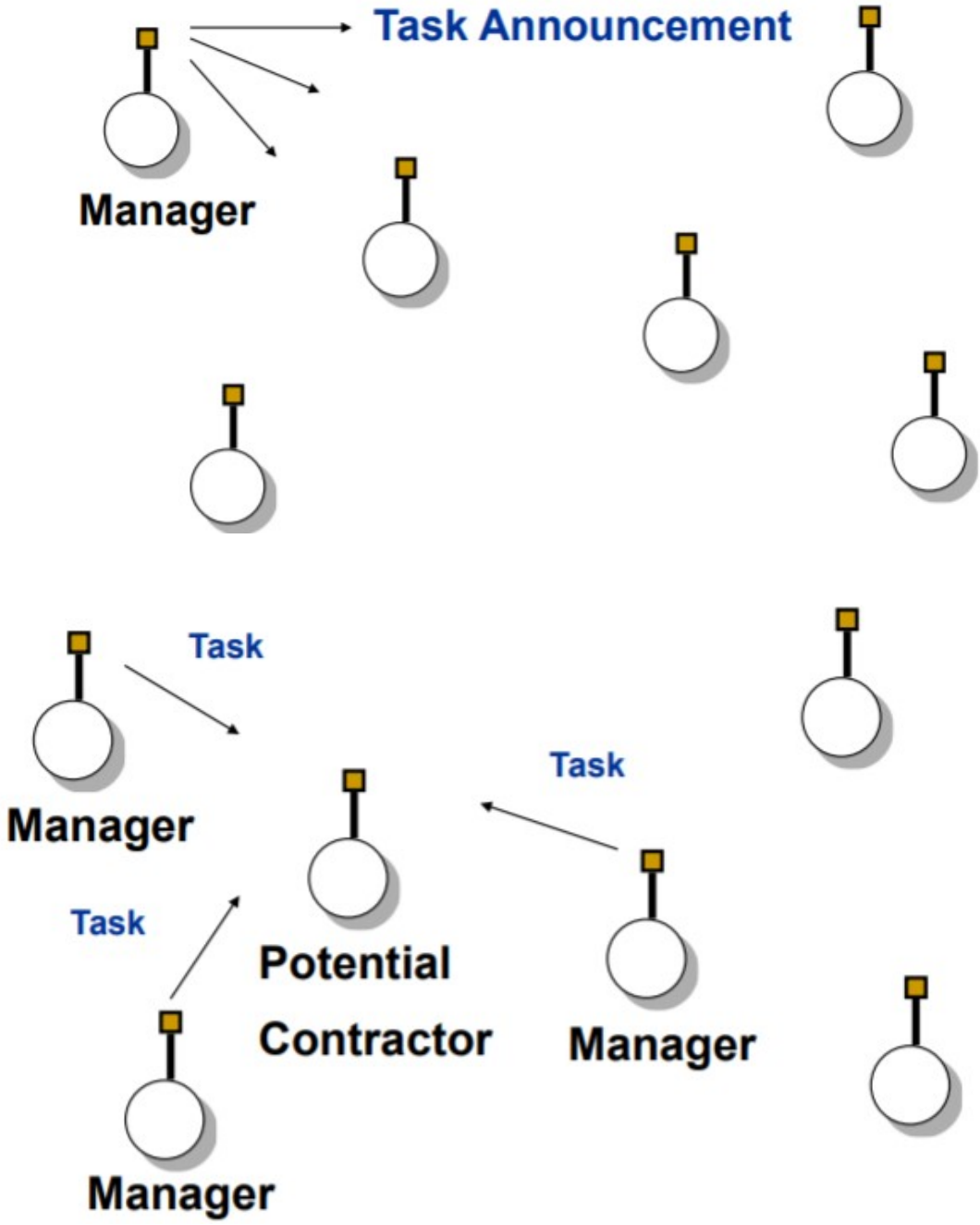


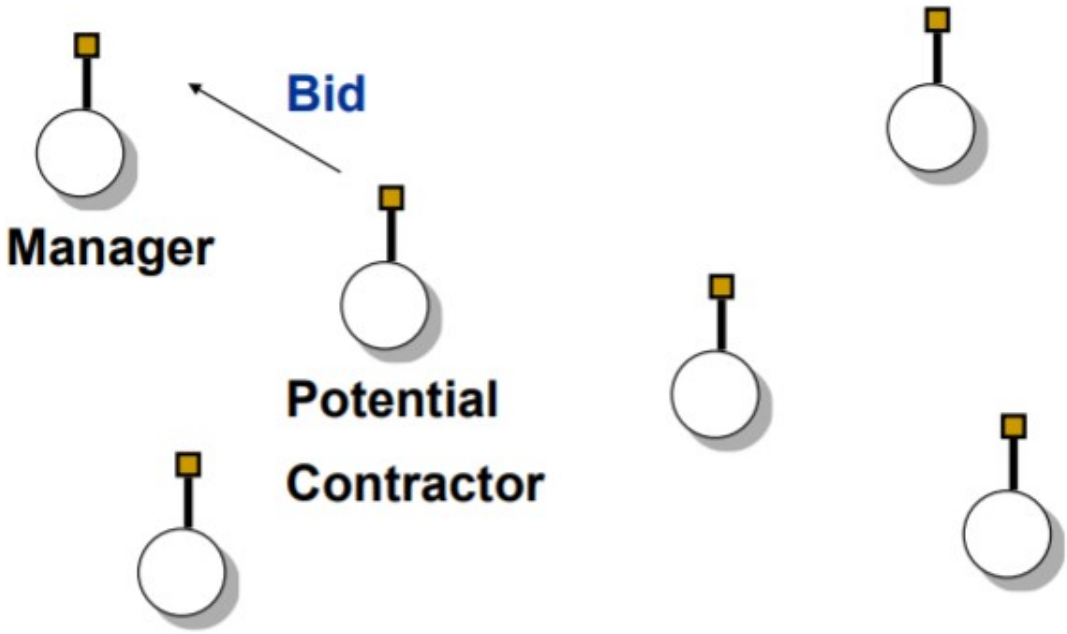
The Contract Net protocol

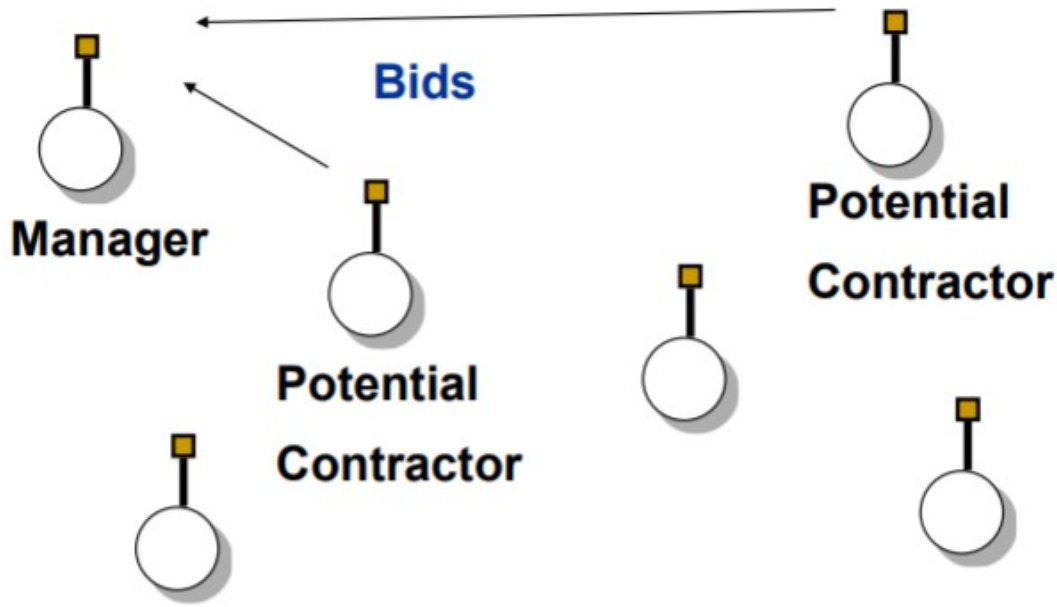
An agent asks for other agents to solve a task that it cannot do.

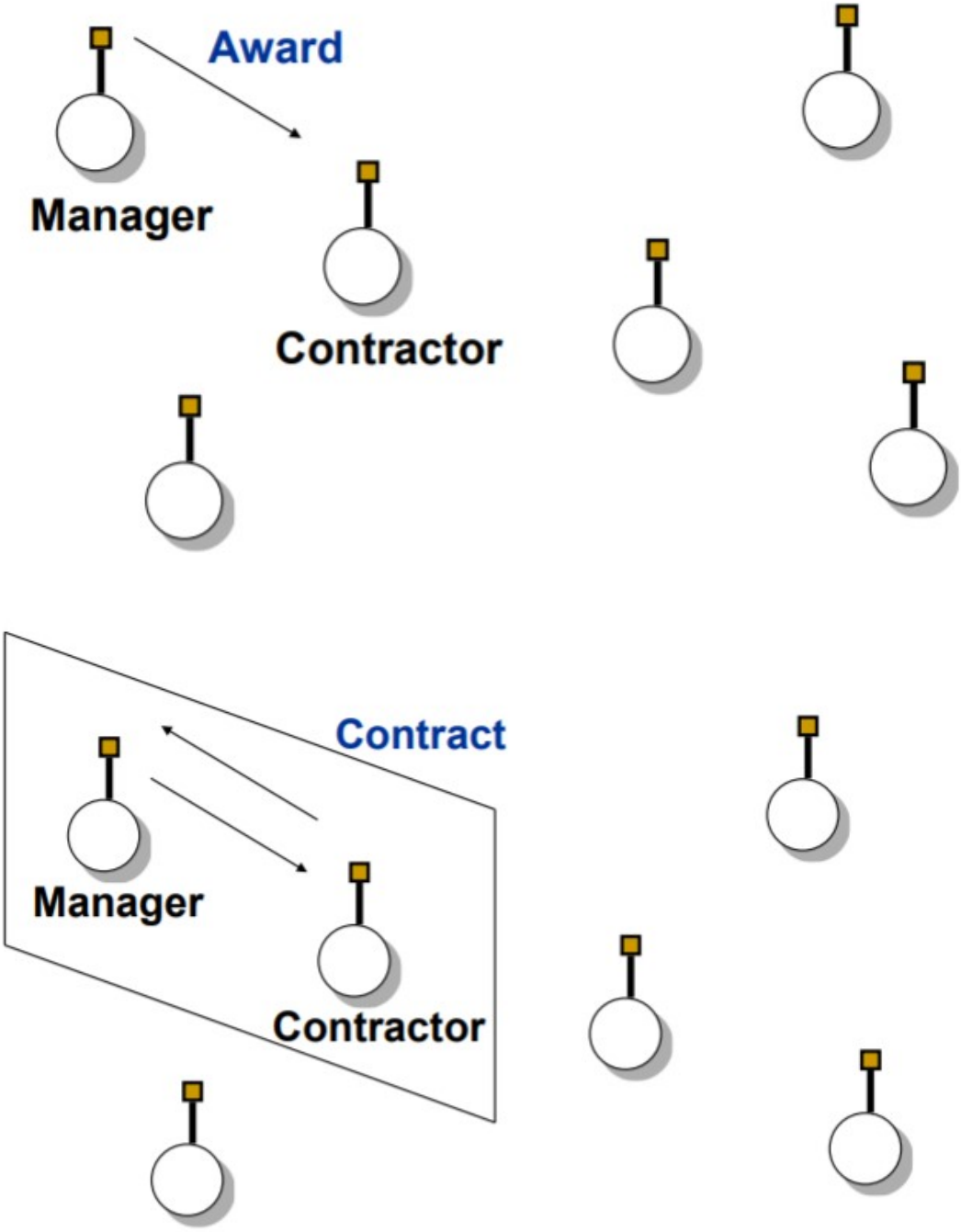
It is a task-sharing protocol consisting in:

1. Recognition
    - In this stage, an agent recognize that have a problem and need help
2. Announcement
    - Sends out an announcement which includes a specification of the task to be achieved
    - This must encode the description of the task, any constraints (deadlines, quality etc...)
    - Is in broadcast
3. Bidding
    - Agents that receive the announcement decide for themselves wheter they wish to bid for the task.
    - If an agent chooses to bid, then it submits a tender, detailing the conditions on which it can execute the task
4. Awarding
    - The agent that sent the task announcement must choose between bids and decide who to "award the contract"
5. Expediting
    - The successful contracto then expedites the task


Task Announcement
Manager
Task
Manager
Task
Potential
Contractor
Task
Manager
Manager


Bid
Manager
Potential
Contractor


Bids
Manager
Potential
Contractor
Potential
Contractor



Manager considers the submitted bids using a domain specific bid evaluation procedure

Ontologies for communication

Ontology defines a common vocabulary and a shared understanding of domain among a set of agents.

Ontology Web Language (OWL)

Components

- RDF Schemas Features
- Equality and Inequality
- Property Characteristics
- Property Restrictions

- Logical Operators

RDF Schemas Features

Define the basic ontological components

- Classes
- Subclasses
- Individuals
- Properties
- Subproperties
- Domain
- Range

Classes

Are sets of individuals with common characteristics.A Class should be described such that it is possible for it to contain Individuals.

Subclasses

Define class specializations by constraining their coverage. Ex: Breast Cancer is a subclass of Cancer

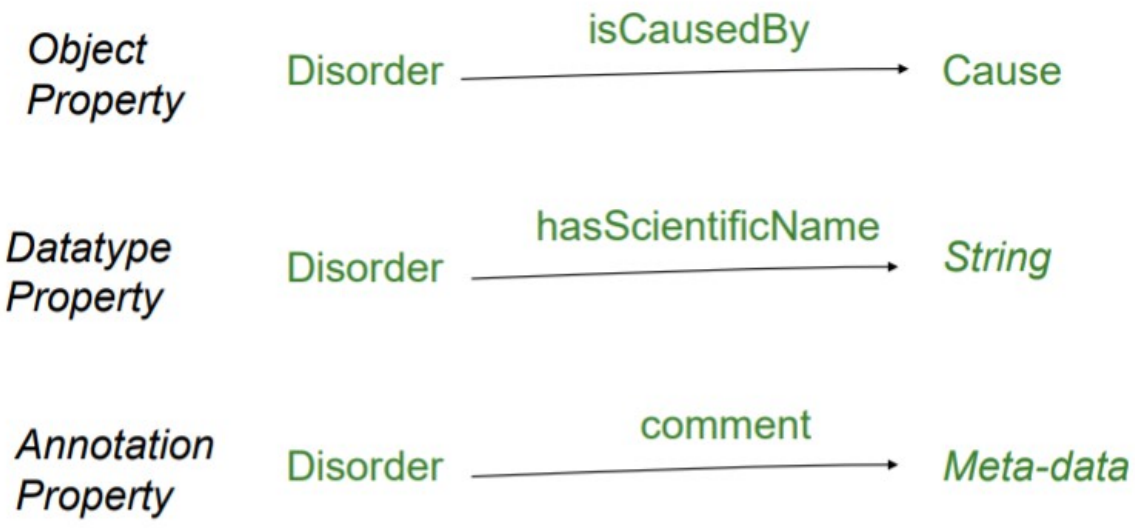
Individuals (Instances)

Specific objects in the domain. Individuals may be (and are likely to be) members of multiple Classes.

Properties

Can be used to state relationships between individuals or from individuals to data values

Eg: hasSymptom, isCausedBy, Author



Sub Properties

Defines properties specializations by constraining their coverage.

Ex. hasScientificName is a subPropertyOf hasName

Domain

It indicates the individuals to which the property is applied.

If a property relates an individual A to another individual B, and the property has a class C as its domain, then the individual A must belong to class C.

Ex. hasSymptom has the domain Disorder

- X hasSymptom Y → X is a Disorder

Range

It indicates the individuals that receive the application of the property

Ex. hasSymptom has a range of Symptom

- X hasSymptom Y → Y is a Symptom

Equality and Inequality

OWL terms that allow expressing equalities and inequalities between ontological components

Ex:

- EquivalentClass: two classes are equivalent
- EquivalentProperty: two properties are equivalent

Property Characteristic

Define the semantics of properties

- InverseProperty: one property is the inverse of another (hasSymptom,isSymptomOf)
- etc...

Property Restrictions

Define constraints on the use of properties

- AllValuesFrom: all the values in the range of a property belong to a given class
- SomeValuesFrom: at least one value in the range of a property belongs to a given class

Logical Operators

Define classes out of other classes

Ex:

- IntersectionOf
  - Tuberculosis_Symptoms = Fever IntersectionOf Coughing_Blood
- UnionOf
  - Flu_Symptoms = Fever UnionOf Vomi
- ComplementOf
  - StandardDisorder = ComplementOf ContagiousDisorder
- OneOf: defines a class by enumerating all the individuals that belong to it
  - Hospitals is OneOf {University-Hospital}, {St_John}, {City-Clinic}

Conclusions

OWL is a language for representing ontologies, which extends frame languages

OWL has a rich set of features

Lecture 6: Cooperation in MAS

What is coordination? Coordination is the process by which an agent reasons about its local actions and the (anticipated) actions of others to try and ensure that the community acts in a coherent manner.

We neet to use coordination when the order or time in which actions are carried out affects performance and there are subproblem dependencies.

Components of coordination

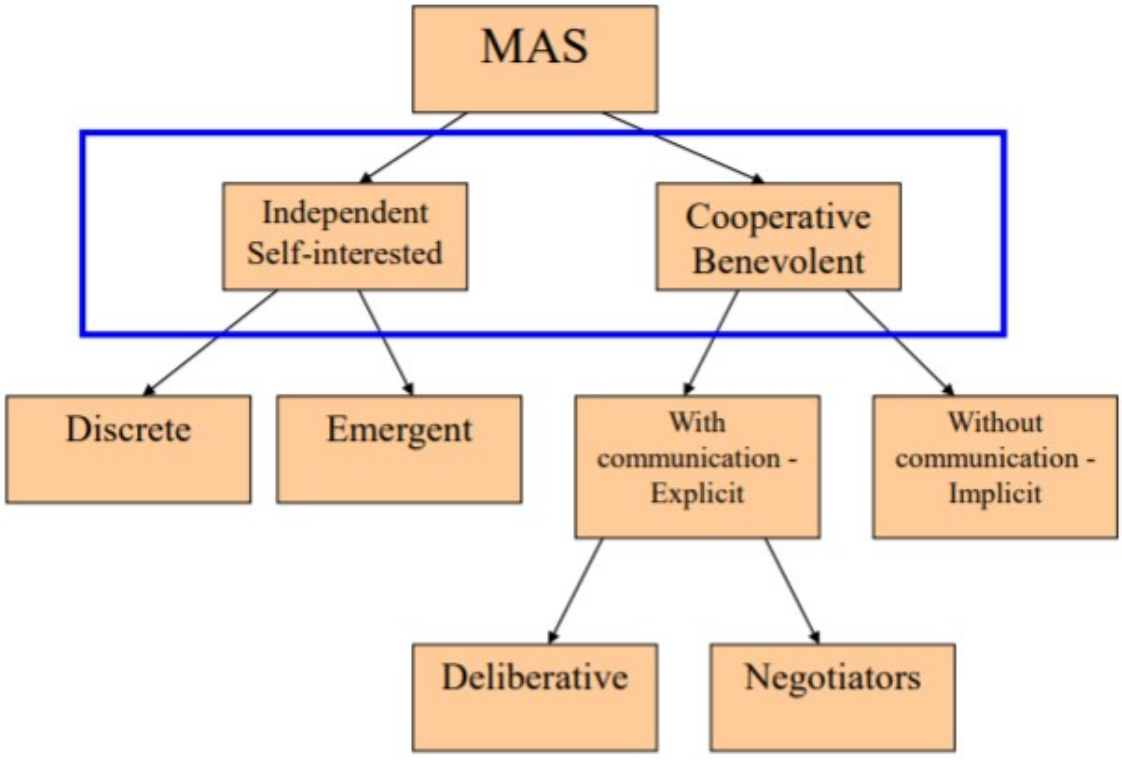
Deciding for each agent in the context of a multiagent system

- What activities it should do and when it should do them
- What it should communicate, when it should communicate and to whom

So Agents need Domain information and Control information.

Coordination depends on ability to cooperate.

Cooperation hierarchy



Benevolent Agents

Design agents to help each other whenever asked.

Problem-solving in benevolent systems is cooperative distributed problem solving.

Cooperative agents

The agendas of the agents include cooperating with other agents in the system in some way.

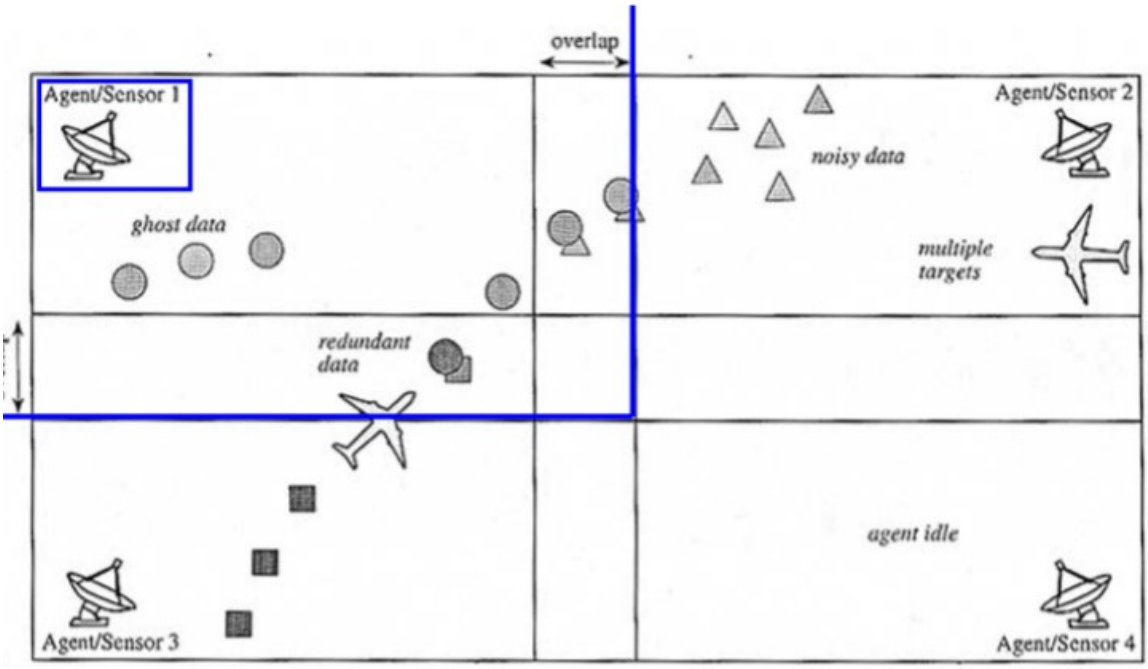
- Explicitly: intentional sending and receiving of communicative signals (e.g. via a common blackboard or via messages)
- Implicitly: without explicit messages (e.g. observing and reacting to the behaviour of the other agents of the system)

Deliberative Agents with communication explicitly

Agents with inference and planning capabilities.

Has a Partial Global Planning (PGP) that is a distrubted planning thechnique.

The tasks are inherently distributed and each agent performs its own tasks



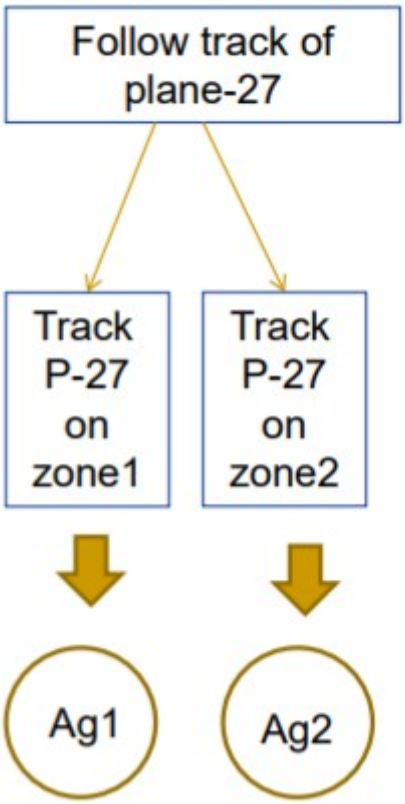
Partial Global Planning phases

1. Create local plans.
    - Each agent has a local plan
    - The plan can store other information like derived objectives.
    - Plan has to be easily modifiable at run time
2. Exchange local plans.
    - The agents exchange information about their local plans with other agents
    - Each agent must have knowledge about the MAS organisational structure, so that it can decide which information to send to which agents
3. Generate Partial Global Plans.
    - Each agent models the collective activity of the system by combining the received local partial plans into a Partial Global Plan
    - Check dependencies between the received information and its own local plan
    - Identify when the goals of one or more agents can be considered subgoals of a single global goal: partial global goal
    - Identify opportunities to improve coordination

    Example

    Partial global goal: final aim of the global plan.

Plan Activity Map: plan actions to be executed concurrently by itself and the other agents, including costs and expected results of actions.



4. Optimize Partial Global Plans
    - Each agent has a Planner Module, especialised in analyzing the received information to detect if there are several agents working on the same goal. This information is put in the Plan Activity Map, along with the expected future behaviour and expected results of the other agents.
5. Building the solution
    - From the modified Plan Acitivity Map, each agent builds a Solution construction Graph that includes specifications about a partial results, when to exchange them and who to exchange.

Possible optimizations of Local Plan

- Task reordering
- Task reallocation

The advantages of PGP is that system has a highly dynamic behaviour so all plans can be adapted to dynamic changes in the environment (Flexibility)

Generalised Partial Global Planning (GPGP)

Set of modular coordination mechanisms

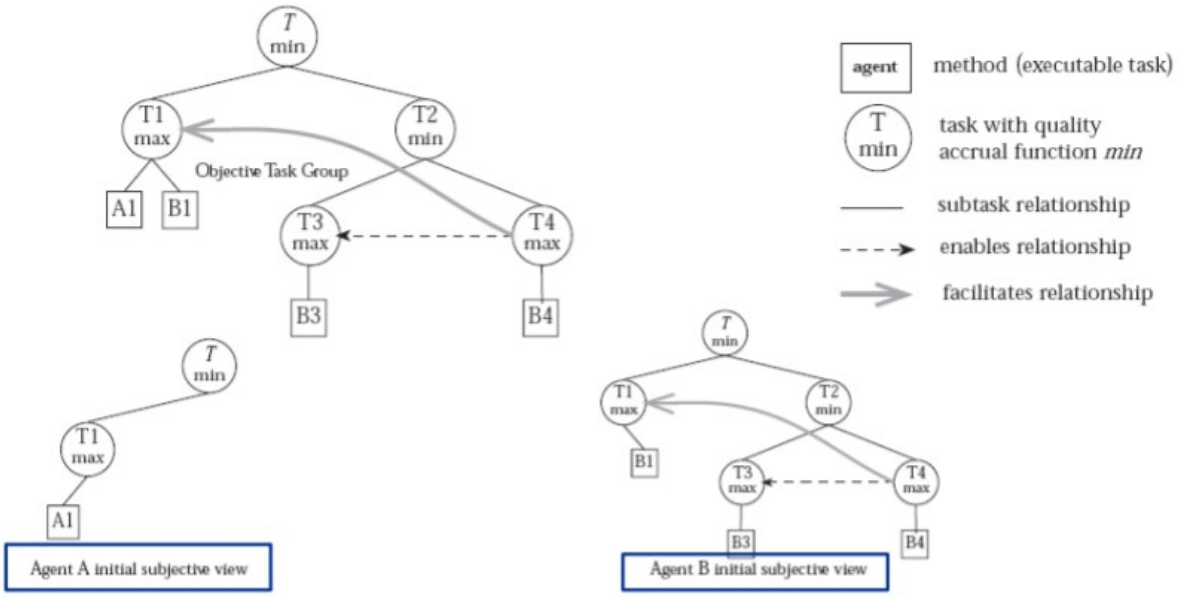
Each mechanism responds to certain features of the problem tasks, posing scheduling constraints (to the own agent or other agents).

Mostly domain-independent (except initial determination of tasks interrelationships)

Tasks relationships

Example:

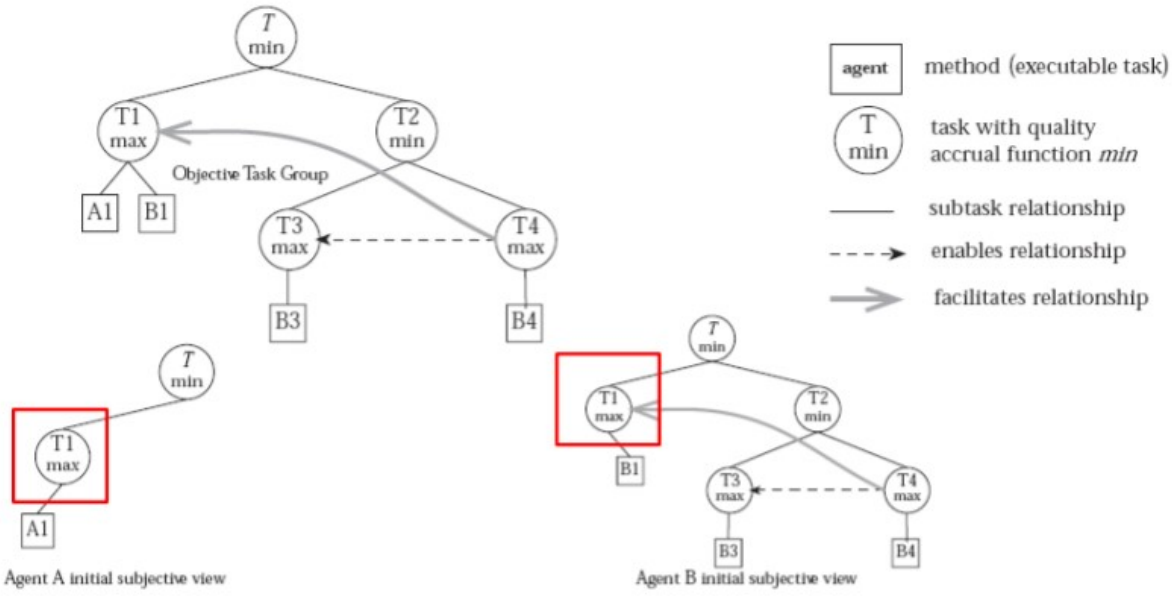
- A enables B: A must be executed before B
- A facilitates B: executing A makes it possible to solve task B more quickly and with more quality
- The general task structure is an AND-OR tree



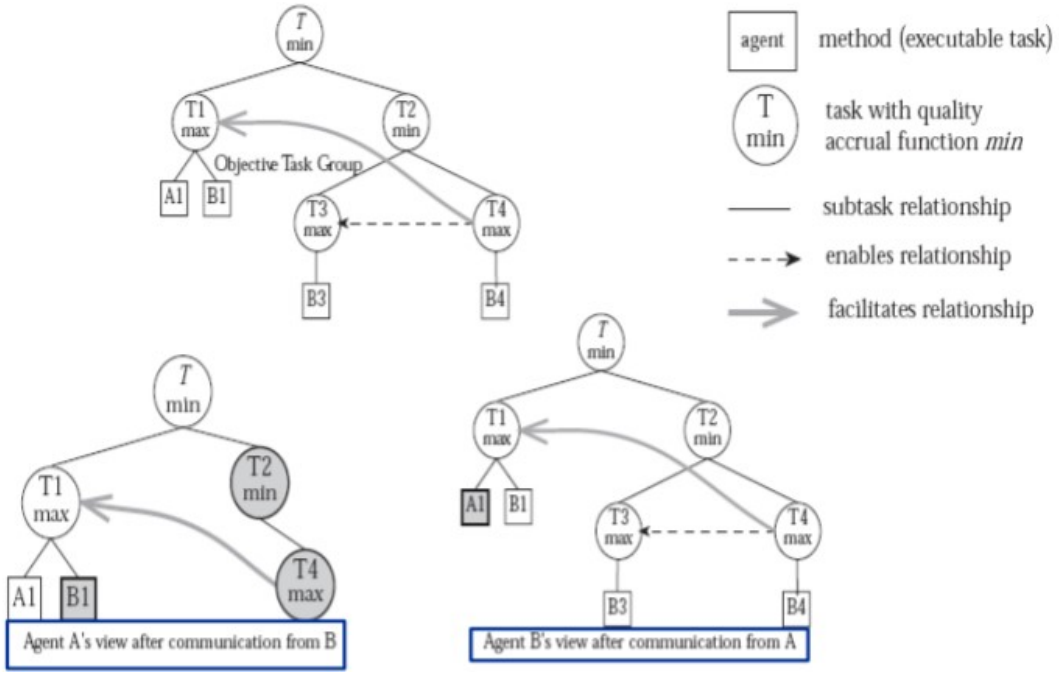
***Note**: Min= AND node , Max = OR node*

Coordination mechanisms (CMs) analyse the task structure and the relationships between tasks to determine scheduling restrictions, and make commitments based on them (a specific task should be finished before a certain deadline with a given quality)

Local scheduler constructs a possible schedule based on the local and non-local commitments.



***Note**: Task T1 may be solved by both agents*



CM: Management of hard coordination relationships

- If A enables B, the agent that can make an action to complete A makes a commitment to finish A as early as possible (so that it gives the maximum space for B to be scheduled – by the same agent or other agents)
- If A facilitates B, an agent can make a commitment to finish A as early as possible, if it is possible (so that it gives to B the possibility to be solved faster and with better results)
- If A hinders B, an agent can make a commitment to finish B as early as possible, if it is possible (so that its execution time and solution quality are not disturbed by solving A)

Redundancy detection

If two agents commit to actions that solve the same task, they apply a common algorithm to decide who takes the task and who abandons it.

Options:

- Random choice
- Choose agent that obtains solution faster
- Choose agent that obtains a better solution
- Choose agent that has less assigned tasks

Conclusions of GPGP

- Using the commitments information and the schedules, the agents agree in the distribution of the tasks and the final plan.
- The different coordination mechanisms can be adapted to each problem by using different policies (diff modules).
- Managing dynamic problems can be easy with this method.

Self-Interested Agents

If agents represent individuals or organizations, then we cannot make the benevolence assumption. Agents will be assumed to act to further their own interests, possibly at the expense of others.

Discrete

- Independent Agents
- Each agent pursues its own agenda and the agendas of the agents bear no relation to one another. No cooperation

Emergent

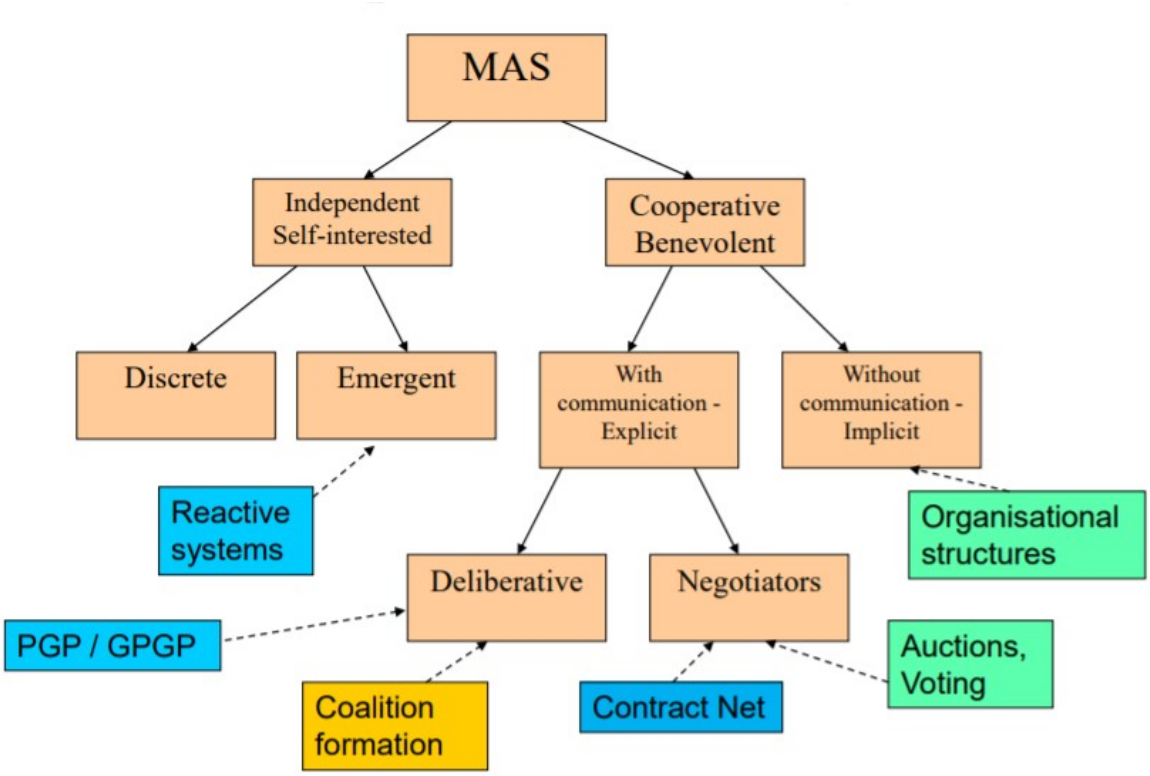
Agents can cooperate with no intention of doing so.

Important Factors

Agents don't need to know about each other, don’t communicate with each other. Don't have special roles. The environment is affected by the actions of all individuals.

LECTURE 7: Cooperation in MAS (II): coalition formation

Cooperation hierarchy



Definition: Coalitions are (temporary) collections of individuals working together for the purpose of achieving a task. So Coalition formation is the process whereby an agent decides to cooperate with other agents, because:

- The task cannot be performed by a single agent
- The task can be performed more efficiently by several agents working together

The reason is that agents usually bring different so they are not clones.

Issues

- What mechanism can an agent use for coalition formation?
- Once a coalition has been defined, how should its members distribute the work/payoff?

External coalition formation

- By imposition: an external agency makes decisions
- Agents advertise skills (capabilities) and prices (cost)
- Requestor defines properties of coalition to the external agency
- An entity external to the MAS computes the optimal coalition

Internal coalition formation

- By self-organisation: coalitions are established by group interactions
- Multi-lateral negotiation of tasks and outcomes
- Identification of tasks to be solved

Coalition formation activities

Need to calculation the coalition value (benefit of coalition for each task) and the structure for try to maximize the coalition value. Finally need to decide how distribute the payoff between coalition members.

Coalition structure generation

- Given 3 agents {a1, a2, a3}, there are seven possible coalitions

{a1}, {a2}, {a3}, {a1, a2}, {a1, a3}, {a2, a3}, {a1, a2, a3}

and five coalition structures

{{a1, a2, a3}}, {{a1}, {a2, a3}}, {{a2}, {a1, a3}}, {{a3}, {a1, a2}}, {{a1}, {a2}, {a3}}

Optimal distribution of task

Optimality could be defined in different ways:

- Minimum number of coalitions
- Minimum size of coalitionts
- etc...

Capabilities in agents

A is a set of *n* agents: $A=\{A_1,A_2,\ldots,A_n\}$

Each agent A_i has a positive vector of *r* capabilities

$$B_i=<b^i_1,\ldots,b^i_r>$$

Each capability is a *property* that *quantifies* the agent's ability in some aspect

Capabilities may be expendable (e.g. amount of material of a certain type) or non-expendable (e.g. ability to perform an action)

Capabilities for tasks

Set of *m* independent tasks $T=\{t_1, \ldots, t_m\}$

There is a vector of *r* capabilities needed to perform each task t_l, $B_l=\{b^l_1, \ldots, b^l_r\}$

The benefit gained from performing each task depends on the capabilities required for its performance

Conditions on coalitions

- A coalition can work on a signle task at a time
- Each agent can only belong to one coalition at time.
- A coalition C has a vector of capabilities Bc
- A coalition C can perform a task *t* iff

$$\text{for all } 1 <= i <= r \;\; b_i^t <= b_i^C$$

Coalitional cost

For each coalition C and specific task t, it is possible to calculate the coalitional value V, that measures the joint utility that the members of C can reach if they cooperate to satisfy t. This value depends on the capabilities contributed by the team members and the number of coalition members.

Task allocation process

- Initial state is that we have a n single agents.
- 1 new coalition at each iteration
- When an agent joins a coalition, it quits the coalition formation process

Algoirithm

- Assign a task to the best coalition
- Coalitional values for each pair <coalition,task> are (re)calculated
- One coalition C is formed
- Agents in coalition C quit the coalition formation process

Stage 1: Initial calculations of Agent.

L_i = empty_set

This list will contain, at the end of stage 1, the coalitions whose value agent A_i should calculate

P_i = all combinations up to *k* agents containing agent A_i

Those are all the possible coalitions in which agent i could participate

For each combination in P_i do

- Contact agent *j*, which is in the combination (i.e. *j* is in some possible coalition(s) with *i*)
- In the first contact, request capabilities B_j
- Commit to calculate the value of some of the coalitions including *i* and *j* = S_{ij}
- $P_i = P_i - S_{ij}$
- $L_i = L_i + S_{ij}$

End_for

- [at the same time]
  For each agent *k* that contacts *i*, $P_i = P_i - S_{ki}$

Each agent *i* has a list L_i of coalitions whose value it has to (repeatedly) calculate

Note: *Check L7 slide 26*

The distribution may not be homogeneous and not be perfect and requires a exchange of message.

Stage 2: Calculate coalitional values

$L_i^{cr} = L_i$ (in the first iteration)

For each coalition C in L_i^{cr} do

- E_c = empty set
- B_c = sum(B_k) for all agents k in C
  [potential capability of coalition C]
- For each pending task t_j, do
  - Check if coalition C can do task t_j ($B_c >= B_j$)
  - If it can, calculate the net benefit of t_j for C, e_j
    e_j = sum of market value of capabilities needed in t_j
    – sum of the capabilities costs
    – internal coordination costs
    $E_c = E_c + \{(t_j, e_j)\}$
- End for

// E_c contains the benefits that coalition C can gain from each of the tasks it is capable of performing

- Calculate Coalitional value of C and its related task as
  (t_c^{best}, V_c) = max-value(E_c)
- Calculate Coalitional cost of C
  $c_c = 1 / V_c$

End for

- At the end of this step, each agent has calculated the best coalitional value/cost for each coalition, and it knows the most profitable task for each coalition

For each coalition C, $[t_c^{best}, V_c, c_c]$

Example

$L_1 = \{ \{A_1, A_2\}, \{A_1, A_4\}, \{A_1, A_2, A_4\}, \{A_1, A_2, A_3, A_4\}\}$

- A_1 calculates the best value (and cost) of each coalition in L_1 , and its associated task
  - $V_{12}, V_{14}, V_{124}, V_{1234}$
  - $c_{12}, c_{14}, c_{124}, c_{1234}$
  - $t_{12}^{best}, t_{14}^{best}, t_{124}^{best}, t_{1234}^{best}$

	t1	t2	t3	v	c	best
A1,A2	-	10	5	10	1/10	t2
A1,A2,A4	15	18	27	27	1/27	t3
A1,A4	-	-	20	20	1/20	t3
A1,A2,A3,A4	18	15	17	18	1/18	t1

Forming one coalition

- The *weight* w_p of a coalition C_p is defined as
  $w_p = c_p / |C_p|$
- C_i = coalition of L_i with minimum weight
  Best coalition for agent *i*
- Each agent announces publicly the weight of its best coalition
- w_{low} = minimum of the announced weights
  Best coalition for all agents $[C_{low}, t_{low}^{best}]$
- If I am an agent in C_{low}, join the other agents in C_{low} to perform task t_{low}^{best}
- Delete the members of C_{low} from the list of candidates to future coalitions
- $L_i = L_i$ – coalitions with agents in C_{low}
- $T = T - t_{low}^{best}$
- L_i^{cr} = coalitions of L_i whose value has to be recalculated
  - Those coalitions whose previous best value was making t_{low}^{best}

	t1	t2	t3	v	c	best	weight
A1,A2	-	10	5	10	1/10	t2	1/20
A1,A2,A4	15	18	27	27	1/27	t3	1/81
A1,A4	-	-	20	20	1/20	t3	1/40
A1,A2,A3,A4	18	15	17	18	1/18	t1	1/72

For agent 1, its best coalition is {A1,A2,A4},

with a cost 1/81 to solve t3

Each agent would make public its best coalition

If agent 1 wins (the cost 1/81 is smaller than the one found by the other agents for their coalitions), agents A1,A2 and A4 will work together in task 3.

The rest of the agents would now have to decide how to make coalitions to solve the remaining tasks.

Overlapping coalitions

The previous algorithm formed disjoint coalitions because agents left the allocation process once they had been assigned to a coalition.

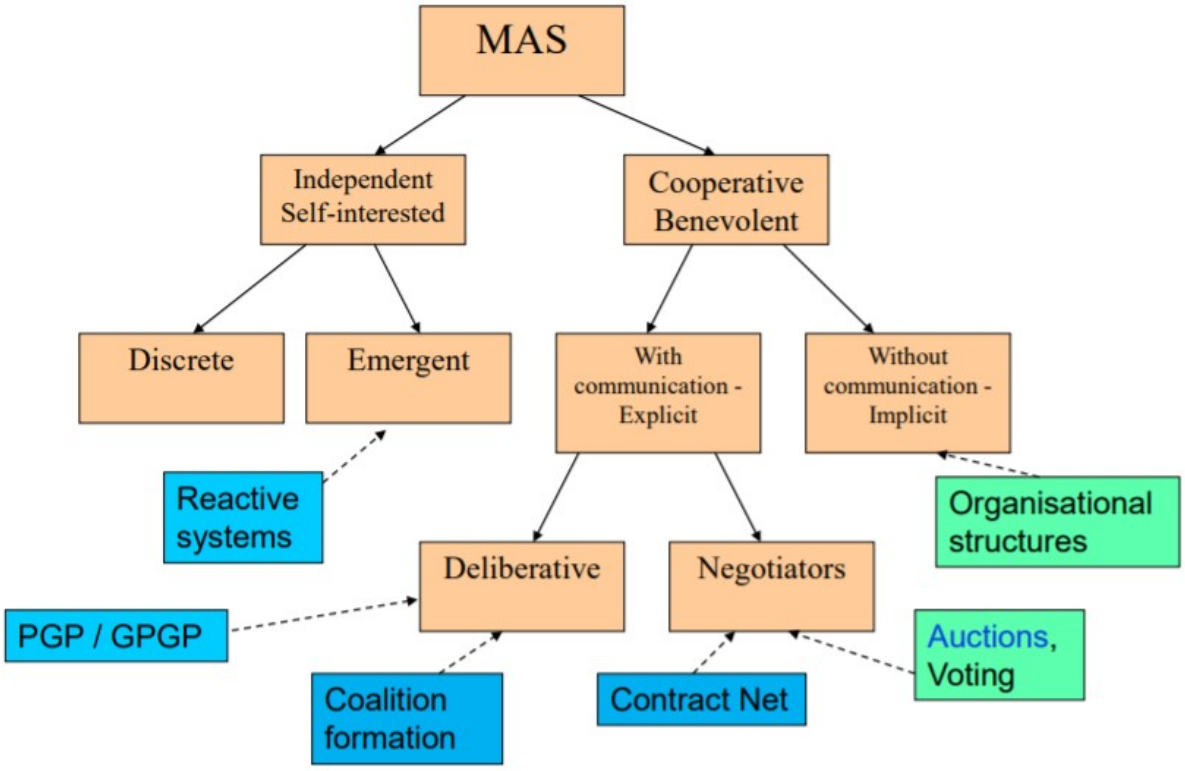
It could be possible to adapt the same task allocation algorithm to allow overlapping coalitions (an agent could participate in several coalitions, as long as it has enough resources).

For do this only need to change when a coalition is form.

- If I am an agent in C_{low}, join the other agents in C_{low} to perform task t_{low}^{best}
- ~~Delete the members of C_{low} from the list of candidates to future coalitions~~
- ~~$L_i = L_i$ – coalitions with agents in C_{low}~~
- $T = T - t_{low}^{best}$
- Update the capability-vectors of all the members of C_{low} (expendable capabilities) according to their contribution to the execution of the task t_{low}^{best}
- L_i^{cr} = coalitions of L_i whose value has to be recalculated
  - Those coalitions that have an agent with changed capabilities or whose previous best value was making t_{low}^{best}

LECTURE 8: Cooperation in MAS (III): Negotiation via auctions

Hierarchy



Why we need a negotiation?

- Agents may have incompatible goals, and the resources to achieve these goals may be limited; in such cases competition and conflicts may arise
- Agents must be able to reach compromises, resolve conflicts, allocate goods and resources by way of an agreement

Negotiation protocol elements

Private elements

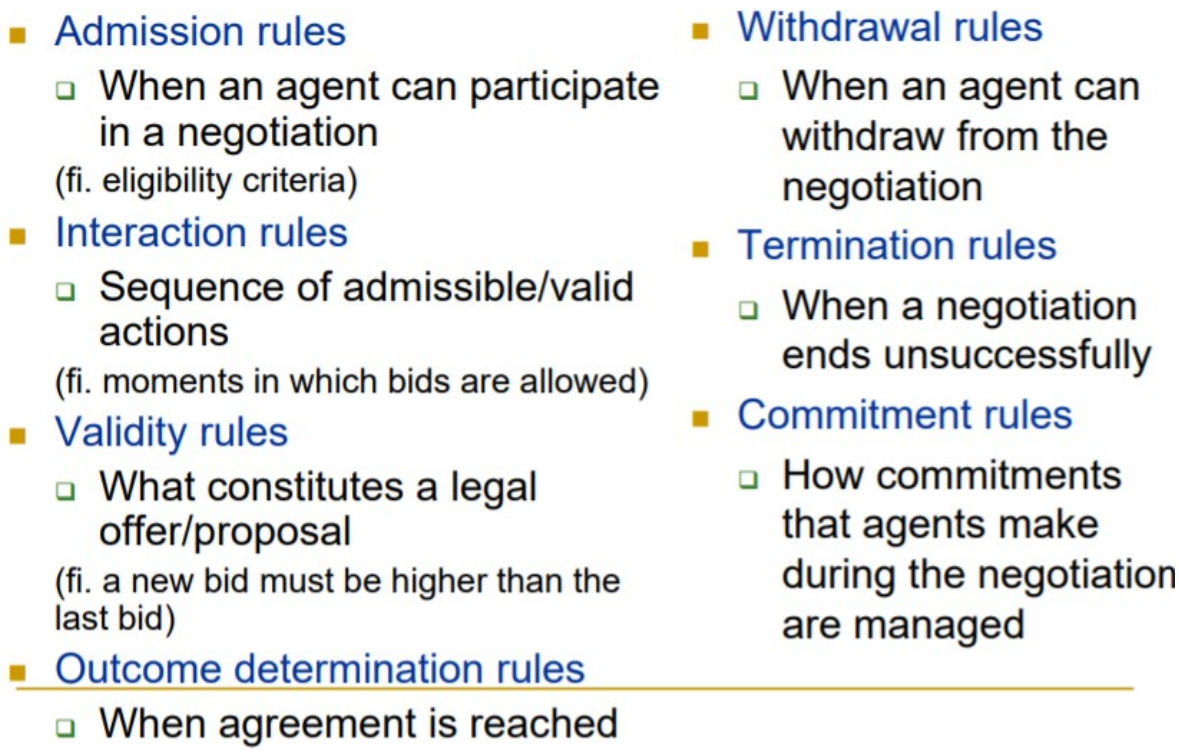
Set of strategies that the agents can use to participate in the negotiation process

Public elements

A negotiation set which represents the space of possible offers/proposals that agents can make.

The protocol rules which govern the agents interactions

Protocol rules

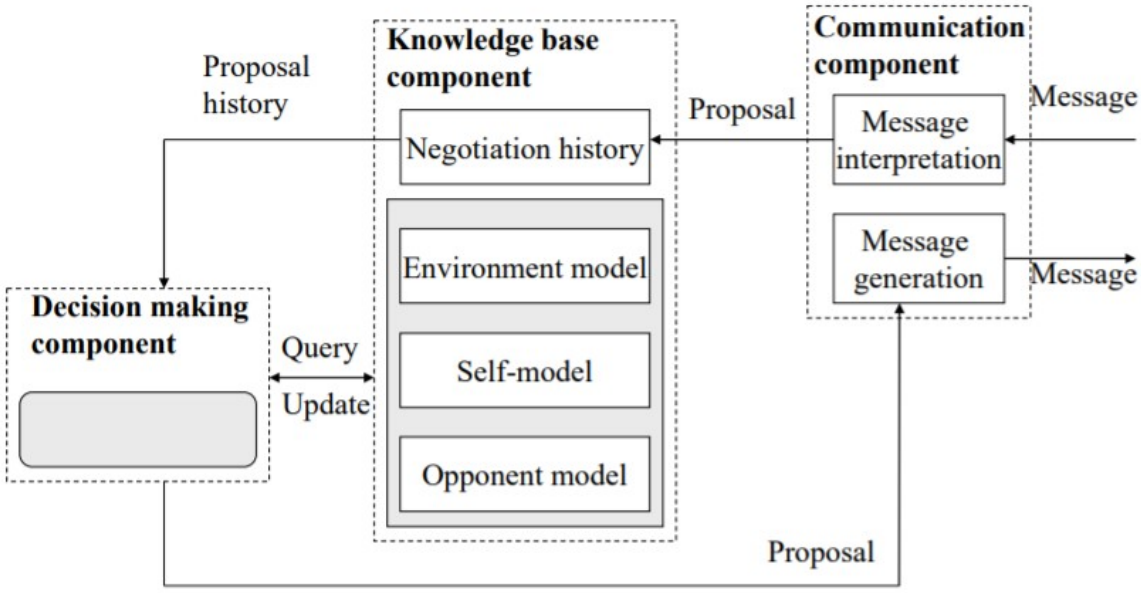


Negotiation factors

- Number of attributes: one , many
- Number of agents: one-to-one, one-to-many, many-to-many
- Number of units: one , many
- Interrelated goods.

Protocol evaluation criteria

- Social welfare – the sum of all agent's payoffs or utilities in a given solution
- Pareto efficiency – a solution x is Pareto optimal if there's no other solution x' such that at least one agent is better off in x' than in x, and no agent is worst off in x' than in x.
- Individual rationality – an agent should not lose out by participating in a negotiation.
- Stability – mechanism should be designed to be non-manipulable: motivate each agent to behave in the desired manner
- Computational efficiency – mechanisms should be designed so that when agents use them, as little computation is needed as possible
- Distribution and communication efficiency – distributed protocol vs. minimum communication (time, money,...)



Auctions

A class of negotiation protocols which provide us with methods for allocating goods/resources based upon competition among self-interested parties

Used:

- Telecommunication and TV licenses, mining rights, airport gates and takeoff/landing slots
- Collectibles (paintings, books, antiques)
- etc...

Advantages

- Markets may not exist for what the seller wants to sell.
- The seller does not know how much an item is worth.
- Create competition; enhances the seller's bargaining power.
- Flexibility

The auctioneer	The bidders
A seller who wants to sell goods at the highest possible price	Buyers who want to acquire goods at the lowest possible price
Someone who wants to subcontract out contracts at the lowest possible price	Contractors who want to get a contract at the highest price
A buyer who wants to buy a good	Sellers who want to sell their goods

The agents that participate in auctions we suppose to be interested and always prefer a larger payoff than a smaller one.

Classification of auctions

- By bidding rules (single good or combinatorial , ascending or descending , open or sealed-bid , etc…)
- Information revelation policy
    - When to reveal information
    - What information
        - Bid: the price a seller
        - Ask: the price a buyer would have to offer in order to trade
        - Auction closure: known, unknown…
- Clearing policy
    - When to clear: on each bid, on closure…
    - Who gets what
    - At what price: first, second etc..

English Auction

- Open-outcry and ascending-price auction
- The auctioneer announces an opening price or the reserve price
- Bidders raise their bids and the auction proceeds to successively higher bids
- The winner of the auction is the bidder of the highest bid

The best strategy is to bid a small amount above the previous bid until one reaches its private value and then stop

The bidders gain information by observing the others' bids

Disadvantages

The reserve price may not be met and the item may remain unsold

Phantom bid: The auctioneer calls a bid that no one has made

Bidders can become carried away and overbid

Vulnerable to shills

Dutch auction

- Open and descending-price auction
- The auctioneer announces a very high opening bid
- Then the auctioneer keeps lowering the price until a bidder accepts it – the winner pays the price of its bid

Bidders need to decide in advance the maximum amount that they are willing to bid.

No relevant information on the valuation of the other bidders is disclosed during the process of the auction until it is too late.

Used for selling fish etc…

First-price sealed-bid auction

Each bidder submits its own bid (usually in writing) without knowledge of the bids of others.

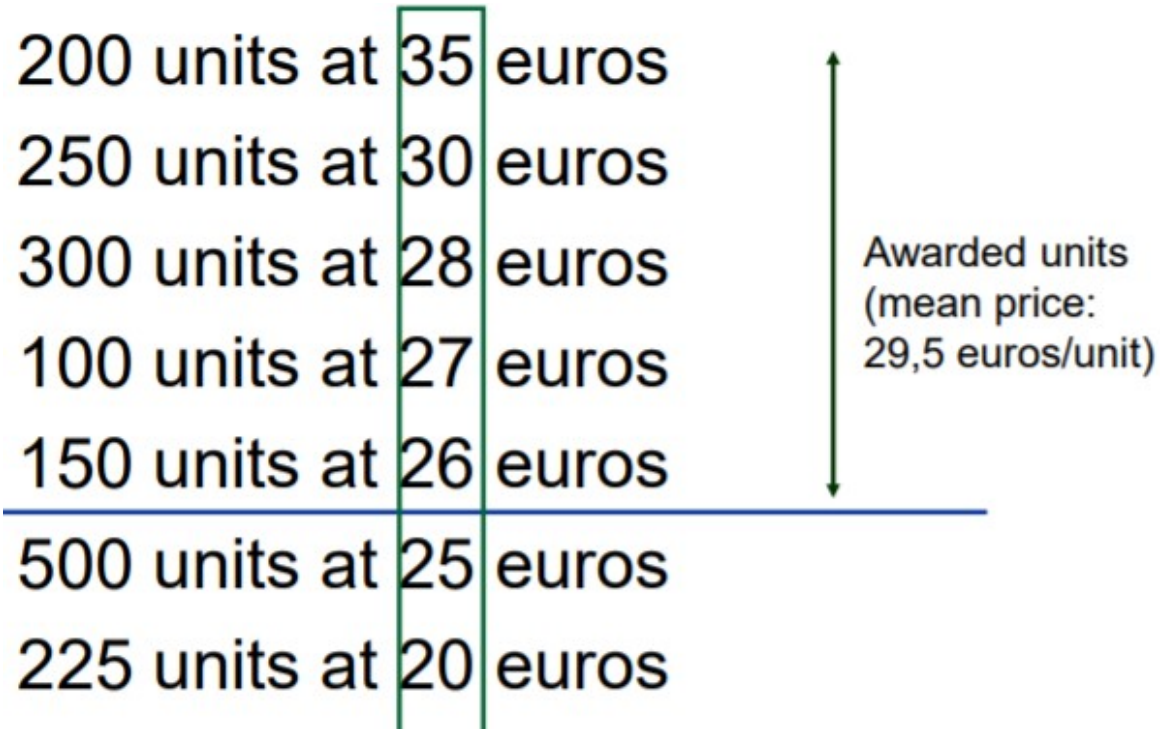
Two phases: (i) the bidding phase in which participants submit their bids (ii) the resolution phase in which the bids are opened and the winner is determined.

The highest bidder wins and pays the amount of its bid.

A higher bid raises the probability of winning, but lowers the bidder's profit if it wins

Multi-unit Auctions

In a multi-unit auction sealed bids are sorted from high to low, and the items are awarded at the highest bid price until the supply is exhausted.



Vickrey auction

A second-price sealed-bid auction, also known as uniform second price sealed-bid or the philatelist auction.

Two distinct phases: the bidding phase and the resolution phase.

The highest bid wins but the bidder pays the amount of the second highest bid.

Bidders adjust their bids upwards since they are not deterred by fear that they will have to pay too much.

The price that the winning bidder pays depends on the others' bids alone and not on any action that the bidder undertakes.

The best strategy is for the agent to bid its true valuation.

Multi-attribute auctions

Multi-attribute or multi-dimensional auctions allow bidders to submit bids on more than one attribute or dimension of a good.

The attributes under negotiation are usually defined in advance and bidders can compete in open-cry or sealed-bid auctions.

Example

- Consider a manufacturer which uses raw materials from a number of suppliers to produce finished goods.
- Assume the manufacturer requires 1000 units of A by March 31st.
- The manufacturer sends a Request For Quotes (RFQ) to all potential suppliers of A.

- A *multi-attribute offer* has the form
  $v_i=(v_{i1},\ldots,v_{ij})$ where v_{ij} is the level of attribute *j* offered by *i*
- Each bid is evaluated through a scoring function $S(v_i)$ which is a weighted average function – often scaled from 0 to 1

$$s_i = S(\mathbf{v}_i) = \sum_{j \in J} w_j S_j(v_{ij}) \text{ and } \sum_{j \in J} w_j = 1$$

Example: 1000 units needed by 31/03

S1: 500 units, 28/03, 30 euros/unit
S2: 1000 units, 4/04, 38 euros/unit
S3: 700 units, 2/04, 37 euros/unit
S4: 300 units, 30/03, 42 euros/unit

- If completeness is preferred => S3+S4 or S2
  - S2 (1000 units, 38000 euros, by April 4th)
  - S3+S4 (1000 units, 38500 euros, by April 2nd)
- If sole-sourcing is preferred => S2
  - S2 (1000 units, 38000 euros, by April 4th)
- If deadline has to be kept => S1+S4
  - S1+S4 (800 units, 27600 euros, by March 30th)

More degrees of freedom for bidders. More efficient information exchange.

Sequential vs Parallel auctions

Sequential auctions: run individual auctions one after the other.

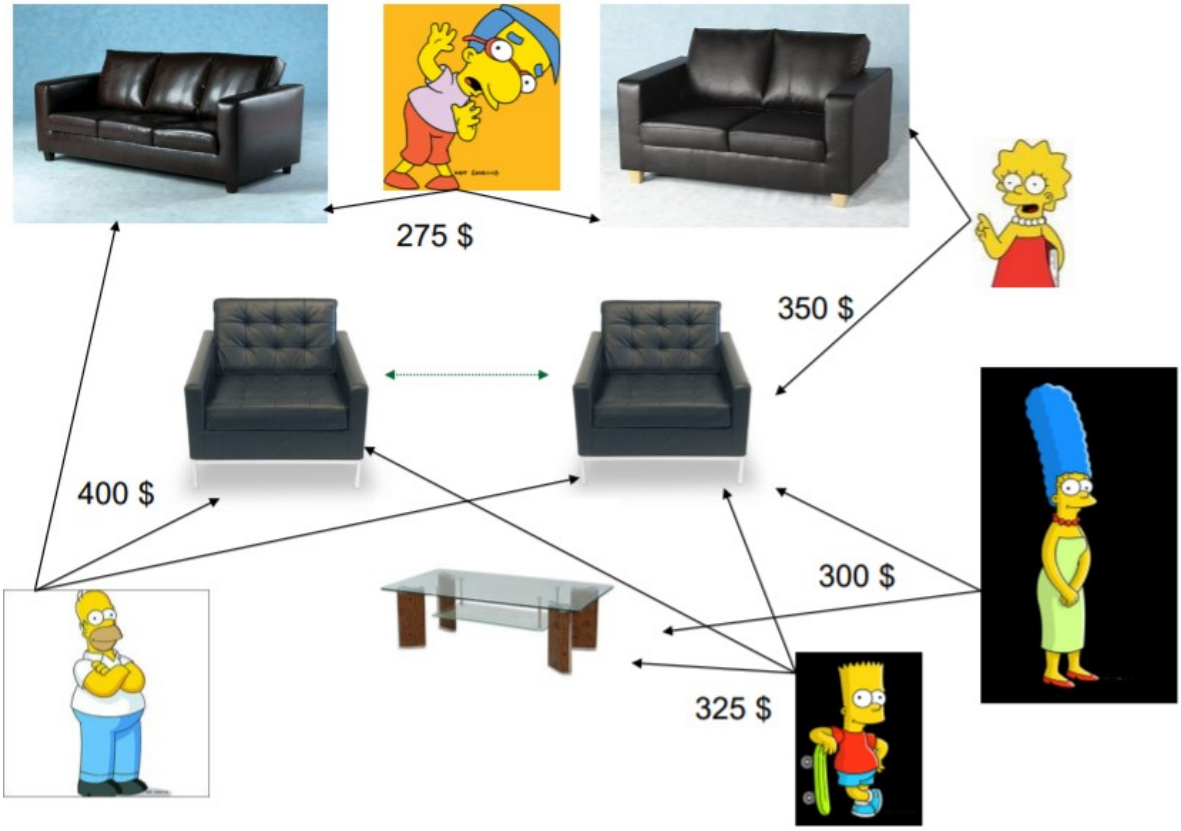
- Impossible to determine best strategy because game tree is huge

Parallel auctions: run individual auctions in parallel.

- Difficult to keep track of several simultaneous auctions with substitutable and interdependent goods

Combinatorial auctions

Bids can be submitted on a combinations (bundles) of items

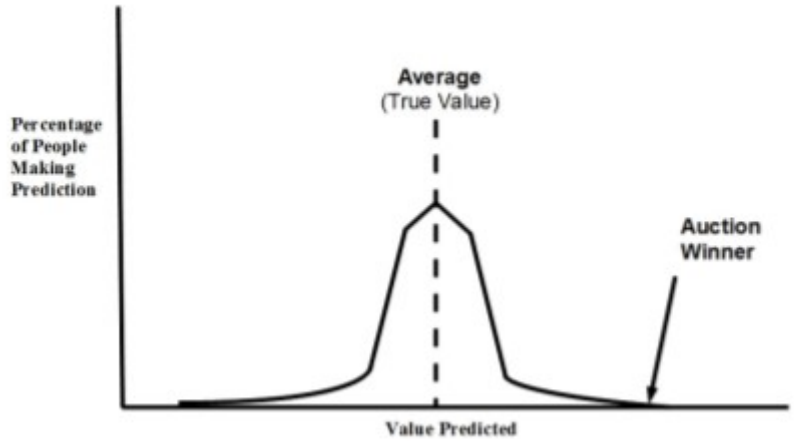


Advantages of auctions

- Flexibility, as protocols can be tailor made
- Less time-consuming and expensive than negotiating a price
- Simplicity in determining the market prices

Disadvantages of auctions

- Winner's curse
    - What bidders suffer when they win an auction by overestimating how much something is worth and therefore bidding too much



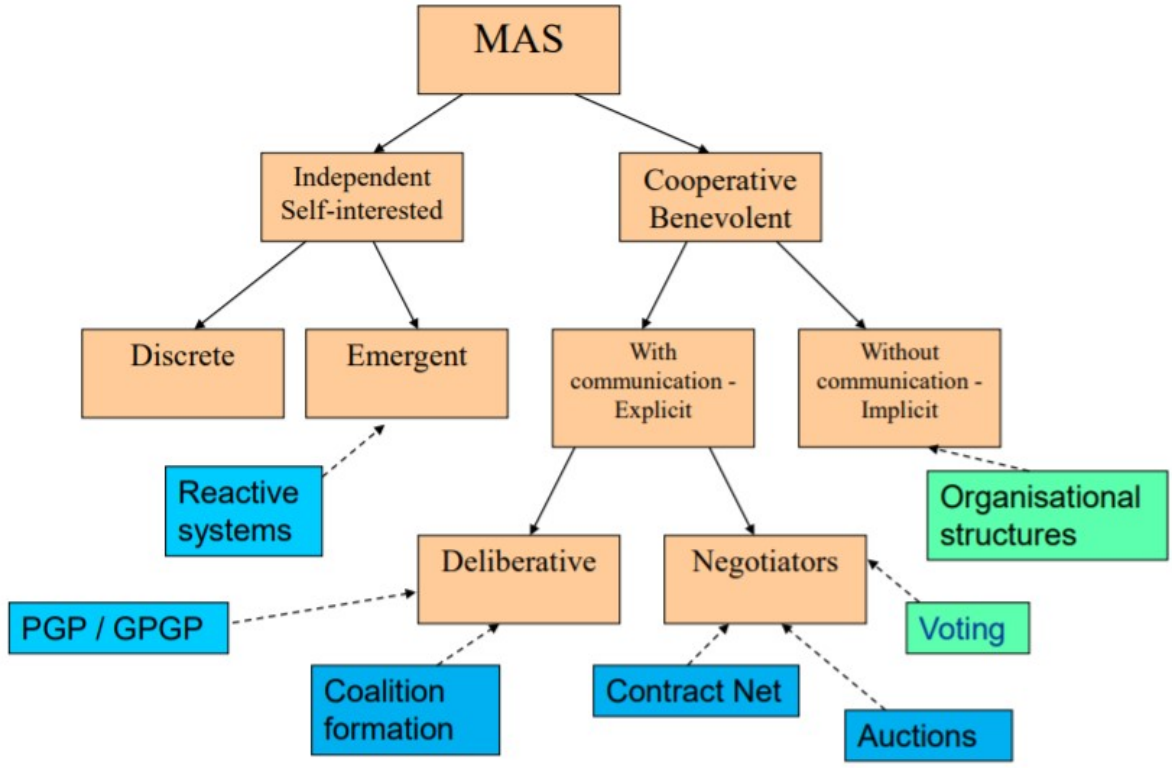
To avoid:

- Assume that you have the highest estimate
- Rational bidding: correct downwards

- Lying auctioneer
  - In the Vickrey auction, the auctioneer may overstate the second-highest bid
    - Solution: use of cryptographic electronic signatures
    - It cannot happen in the other 3 protocols
  - In sealed-bid auctions: Auctioneer may place a bid himself (reservation price)
- Sniping
  - bidding very late in the auction in the hope that other bidders do not have time to respond and you can snatch a bargain. This is an issue, in particular in online auctions.

LECTURE 9: Cooperation in MAS (IV): voting protocols

Hierarchy



Basic elements in a voting protocol

Aim of the negotiation: rank a set of alternatives based on the individual ranking of those options by each agent

- A - set of *n* agents
- O - set of *m* alternatives
- Each agent *i* has a preference relation

  <i : O x O

- Weak order: complete and transitive

R_1	R_2
x_1	x_2 x_4
x_2 x_3	x_3
x_4	x_1

- Linear order: weak order + anti-symmetric

R_3
x_3
x_1
x_2
x_4

Social Choice rule

- Input: the agents preference relations ($<_1, \ldots, <_n$)
- Output: elements of O sorted according to the input (social preference relation <* of the agent group)
- In other words, it creates an ordering of the group of alternatives, so that the most (socially) preferred alternative is chosen
  - This order may be partial

Desirable properties of the social choice rule

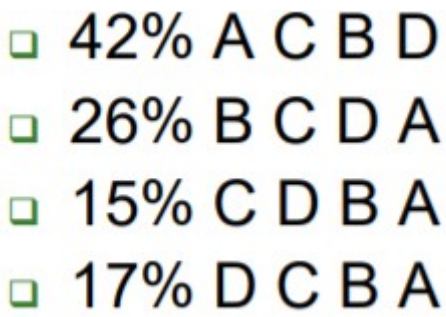
- Calculability A social preference ordering <* should exist for all possible inputs
- Completeness <* should be defined for every pair of alternatives (o, o')∈O
- Linearity <* should be antisymmetric and transitive over O
- Anonymity / No dictatorship
- Unanimity / Pareto efficiency If ∀i ∈A (o <i o') then (o <* o')
- Neutrality: The outcome of the social choice rule should not depend on how alternatives are named or ordered
- Independence of irrelevant alternatives: Removing / Adding an irrelevant alternative should not affect the winner of the vote

Plurality protocol

- Each agent can give 1 vote to 1 of the alternatives
- The alternative with the highest number of votes wins

Problems

- Each agent can only give 1 vote, even if it considers 2-3-4 “good” alternatives
- Strange effects
    - Example:



        - A wins with 42% of the votes
        - 58% of the voters preferred other options
        - 58% consider A the worst option
        - 100% consider C the 1st-2nd best option

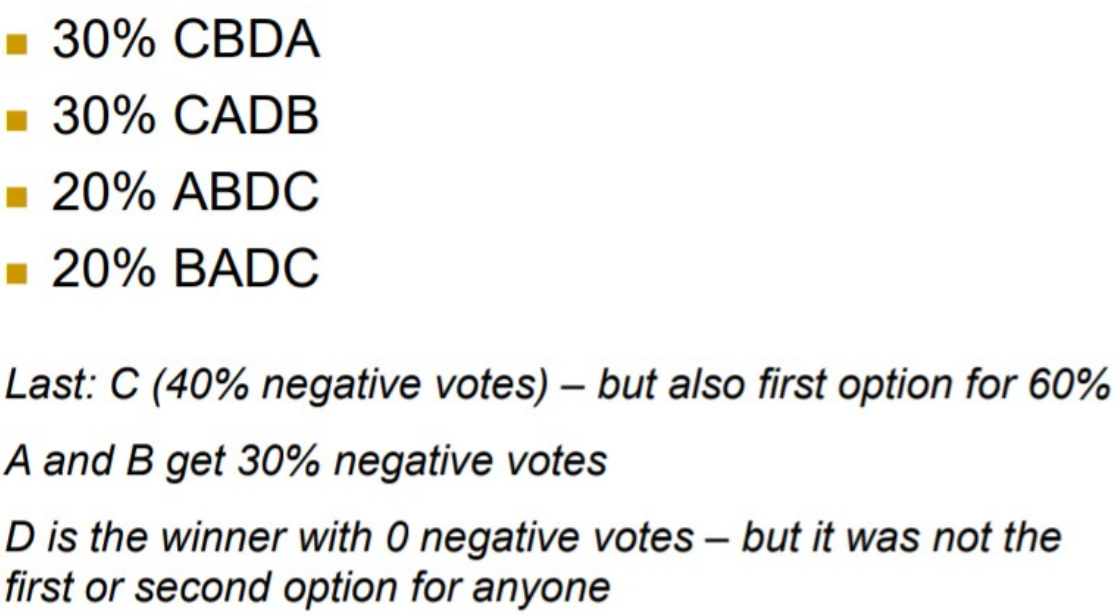
Advantages

- Simple mechanism
- Efficient from computational point of view
- Equality, 1 agent = 1 vote

Anti-plurality

Each voter gives a negative vote to the alternative he considers the worst. The option with less votes wins. In the previous example, A would have 58 negative votes!

Example



Best-worst voting system

Each agent gives a positive vote to his best alternative and a negative vote to his worst alternative

Each alternative receives $\alpha>0$ points for each positive vote and $-\delta<0$ points for each negative vote.

Example

1	2	3	4	5	6	7
C	C	B	B	A	A	A
B	B	C	C	B	C	C
A	A	A	A	C	B	B

$A \to 3\alpha - 4\delta$ points $\quad B \to 2\alpha - 2\delta$ points $\quad C \to 2\alpha - \delta$ points

Plurality $(\delta = 0)$	$A \succ B \sim C$
Antiplurality $(\alpha = 0)$	$C \succ B \succ A$
$\alpha = \delta = 1$	$C \succ B \succ A$
$\alpha = 2$ and $\delta = 1$	$C \succ A \sim B$
$\alpha = 4$ and $\delta = 1$	$A \succ C \succ B$

Protocols based on linear orders

Each voter gives a full list of the options, ordered according to his preferences (from best to worst)

A voter prefers option ***i*** to option ***j*** if option ***i*** appears before option ***j*** in his list

Binary Protocol

All the options are ordered and then evaluated in pairs (options 1 and 2, the winner with option 3, the winner with option 4, etc.)

Simple majority: option A is better than option B if and only if the number of voters that prefer A to B is greater than the number of voters that prefer B to A

The option that wins the last evaluation is the overall winner

$$\text{win}(o_5, \text{win}(o_4, \text{win}(o_3, \text{win}(o_2, o_1))))$$

Borda protocol

For each voter, we assign |O| points to the preferred option, |O|-1 points to the second, and so on

- Example: 4 possibilities, one agent considers the order ABCD A:4 points, B:3 points, C:2 points, D:1 point

Problems

- Most computationally expensive
- Eliminating (or adding) one irrelevant alternative may totally change the outcome of the protocol
- Total order changes if options are removed one by one

Borda paradox

- a > b > c > d
- b > c > d > a
- c > d > a > b
- a > b > c > d
- b > c > d > a
- c > d > a > b
- a > b > c > d

a=18, b=19, c=20, d=13

If the worst alternative –d- is removed

- a > b > c
- b > c > a
- c > a > b
- a > b > c
- b > c > a
- c > a > b
- a > b > c

a=15, b=14, c=13

Even if we keep the relative preferences between a, b and c, the final result changes completely

Condocert protocol

Each voter ranks the candidates in order of preference

Each candidate is compared to each other

If a candidate wins all the comparisons, it is the winner of the election

Example

- Election of the capital city of Tennessee
- Everybody prefers to have the capital as close as possible

42% of voters (close to Memphis)	26% of voters (close to Nashville)	15% of voters (close to Chattanooga)	17% of voters (close to Knoxville)
1. **Memphis**	1. **Nashville**	1. **Chattanooga**	1. **Knoxville**
2. Nashville	2. Chattanooga	2. Knoxville	2. Chattanooga
3. Chattanooga	3. Knoxville	3. Nashville	3. Nashville
4. Knoxville	4. Memphis	4. Memphis	4. Memphis

Problems

- Possibility of circular ambiguities: No alternative wins to all the other alternatives

More complex voting mechanisms

Use of linguistic information to represent the opinion of each voter with respect to each alternative.

Management of uncertainty in the voter's opinions.

Each voter can use a linguistic label from a predetermined set to evaluate each alternative

R	To Reject
P	Poor
A	Acceptable
G	Good
VG	Very Good
E	Excellent

	1	2	3	4	5	6	7
Alan	A	R	VG	E	A	P	A
Benny	VG	E	G	VG	G	G	G
Charles	G	P	E	R	G	G	VG

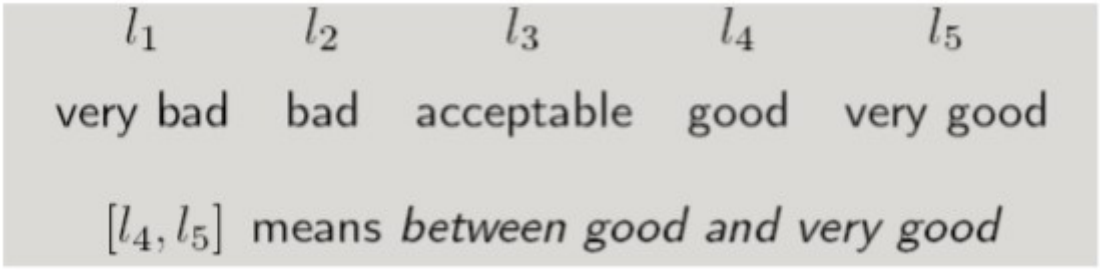
Distance measure

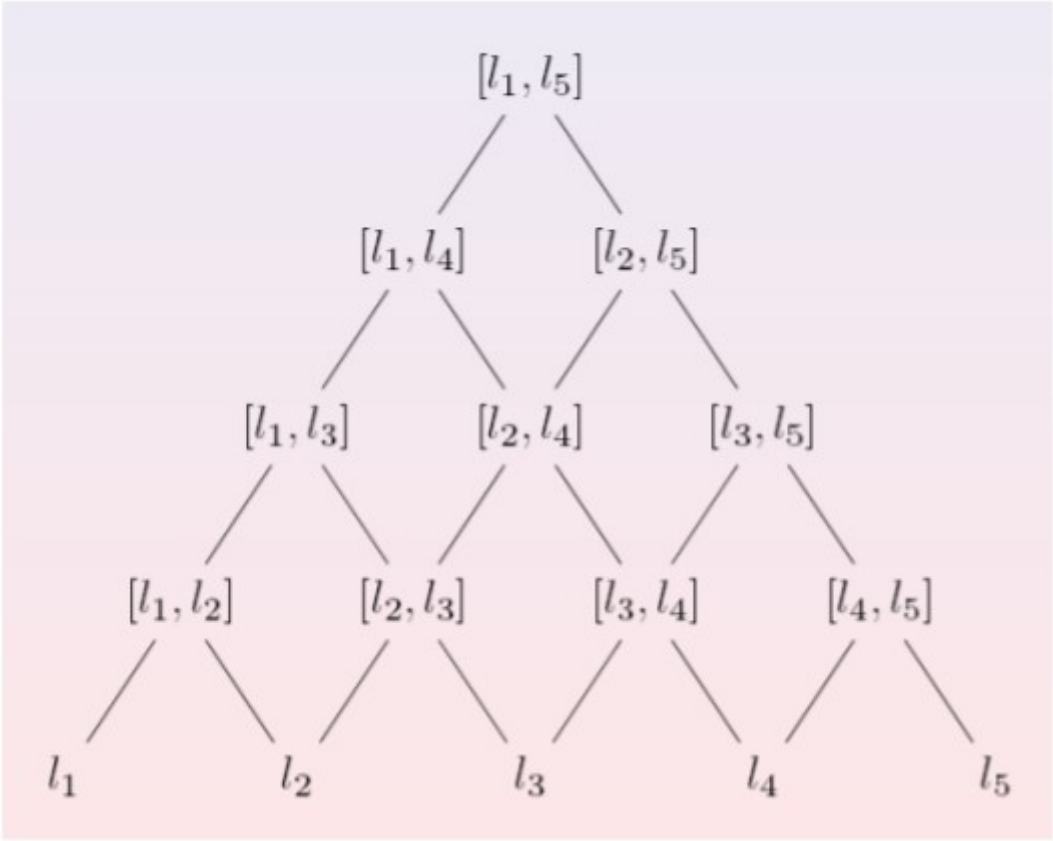
The distance between two vectors of linguistic labels may be defined as the addition of the distances between the labels in each position

- Distance(Poor, Poor) = 0
- Distance(Poor, Good) = 2
- Distance(Reject, Excellent) = 5

Uncertain preferences

Each voter represents the opinion on every alternative with an interval defined over an ordered set of linguistic labels

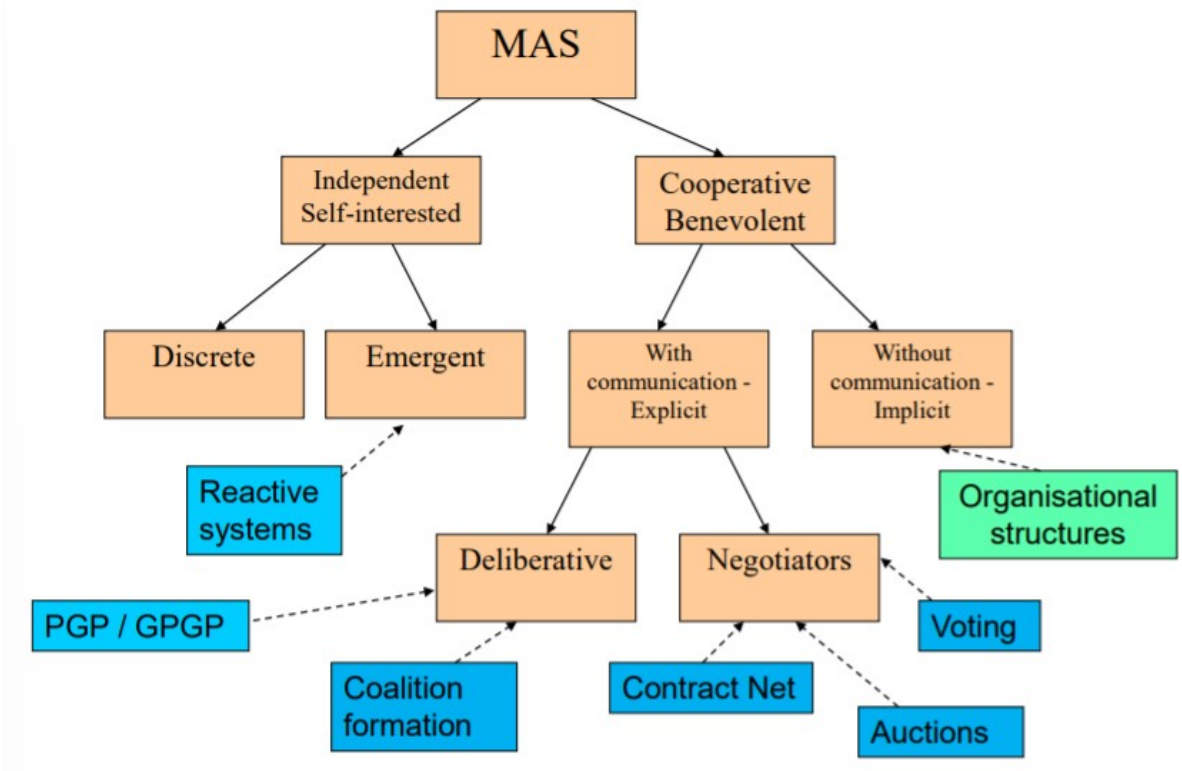




Top is more imprecise than bot.

LECTURE 10: Cooperation in MAS (V): implicit methods

Hierarchy



Implicit cooperation

A group of distributed cooperative agents behaves in a socially coordinated way in the resolution of a global problem without an explicit exchange of communication messages

In many cases the environment acts as the (indirect) interaction mechanism

Motivation

- Speed: takes to long to communicate with others
- Security: not wanting others to know what your plans are
- Complexity: some agents may be too simple to deal with the task of generating and understanding long plans

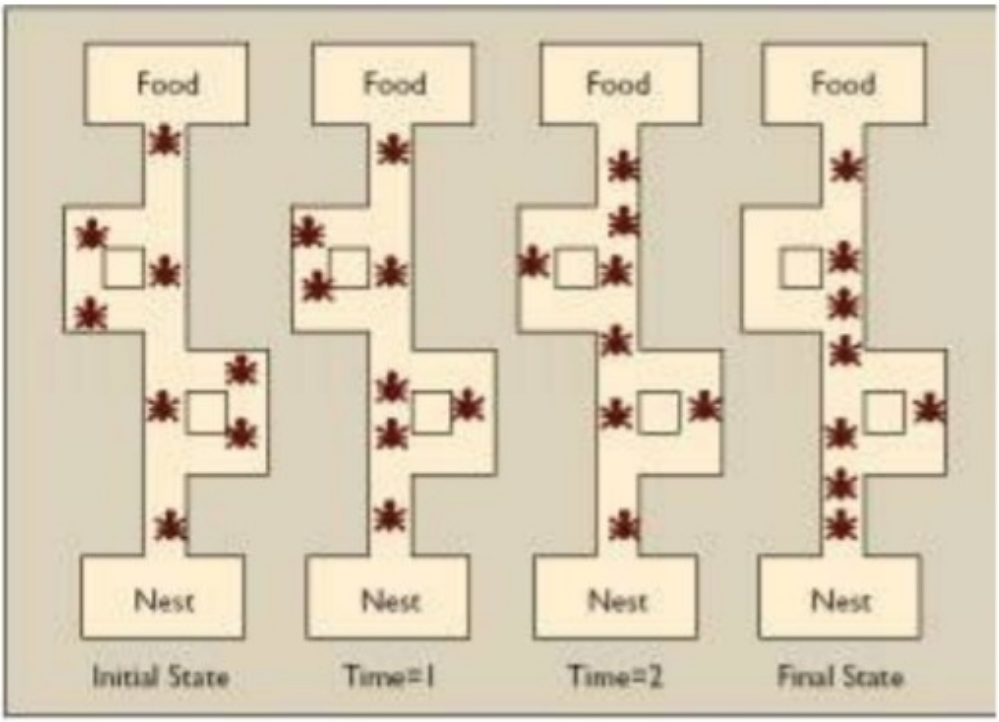
Options:

- Indirect cooperation through the effects on the environment of the actions of each agent.
- Reasoning mechanisms

Indirect cooperation

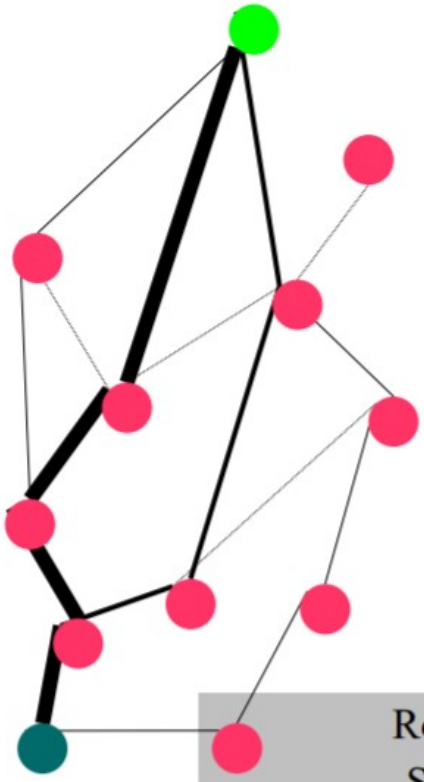
- Coordination in cases where:
    - There is no communication between agents
    - There is no mechanism for enforcing a-priori social rules

Example



Network Ants

- Ants randomly explore the network until they find a specific node
  - They mark the traversed paths with “pheromone”
- Ants seeking destinations tend to follow pheromone trails
- Pheromones degrade over time

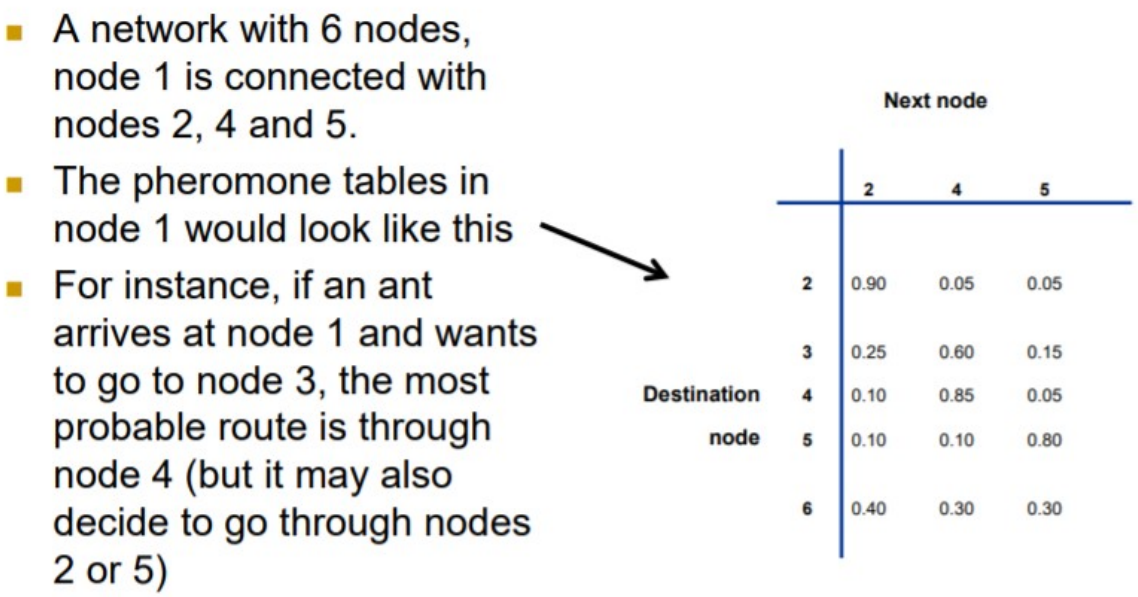


Robust
Stable
Gradual Change

Pheromone tables

- Each node contains a table of probabilities (pheromone table) for each possible destination in the network
  - In a 30-nodes network, each node keeps 29 tables

- The entries on the tables are the probabilities which influence the ants' selection of the next node on the way to their destination node
- Pheromone laying = updating probabilities



- Pheromone tables are initialized with random values.
- At each step, ants can be launched from any node in the network, with a random destination node.
- Ants move from node to node, selecting the next node to move to according to the probabilities in the pheromone tables for their destination node.
- When ants arrive at a node, they update the probabilities of that node's pheromone table entries corresponding to their source node

Basic ideas of implicit cooperation

- Agents do not talk to each other directly
- Agents can modify the environment, and these modifications influence the behaviour of the other agents in the system
- All the agents contribute towards a useful global behaviour of the community

Agent modelling

- Recursive modelling method
  - Try to deduce their beliefs/desires/intentions from their actions on the environment
- Plan recognition
  - Analyse the sequences of activities of other agents and try to discover their plans.
- Game Playing / Game Tree Search:
  - Modelling opponents (e.g. minimax search)

Institutions

Are a kind of social structure where a corpora of constraints shape the behaviour of the members of a group

The definition includes:

- Norms about the interactions
- Procedures and protocols to be followed

- Conventions: acceptable (and unacceptable) actions within the institution

An e-Institution is the computational model of an institution through

In the context of MAS, e-institutions:

- reduce uncertainty about other agents' behaviour
- allow agents to foresee the outcome of an interaction
- simplify the decision making process

AMELI

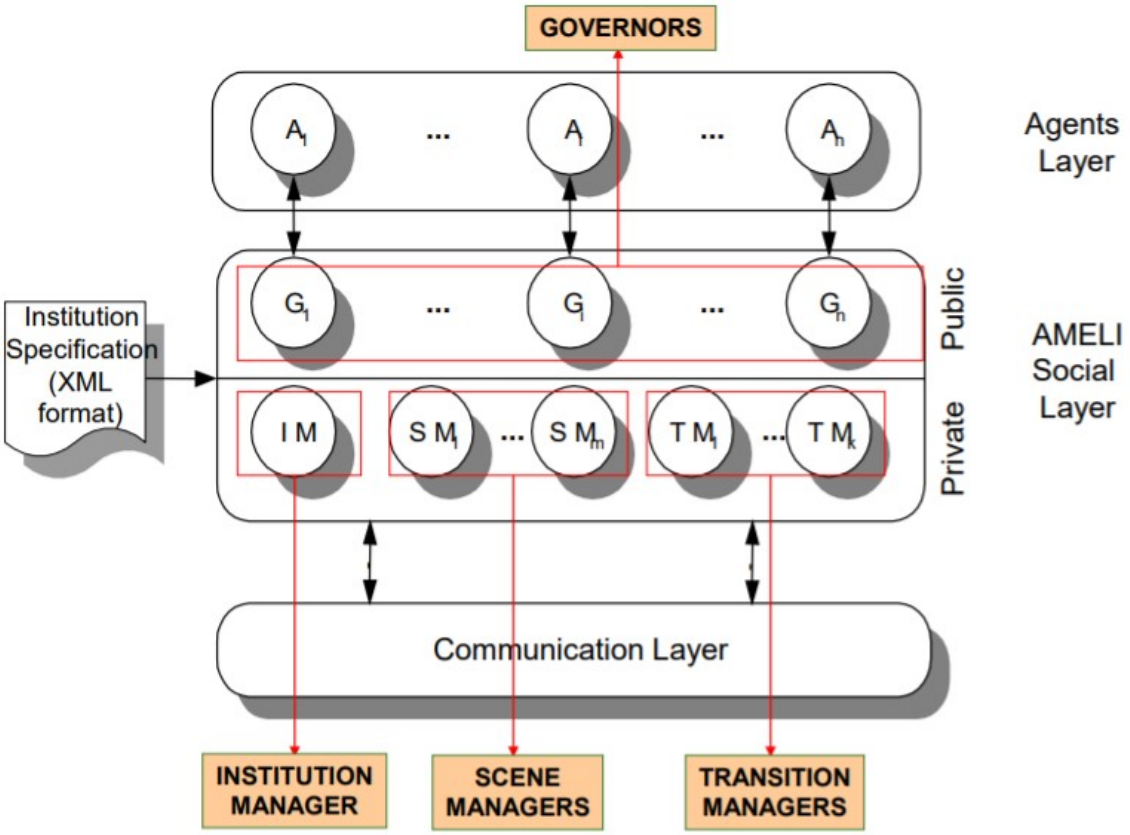
Is an institution middleware that is based in a formal electronic institution specification tool.

Two hypotheses:

- All agent actions are messages, observable by the e-institution.
- An agent should never break the norms.

Objective

- Mediate and facilitate agent communication within conversations (scenes).
- Guarantee the correct evolution of each conversation.
- To guarantee that agents' movements between scenes comply with the specification



- An *institution manager* that starts the institution, authorises agents to enter, and controls the creation of scenes
- *Scene managers* responsible for governing scenes (one for scene)
- *Transition managers* control agents' movements between scenes (one for transition)
- *Governors* mediate the interaction of an agent with the rest of the agents within the institution and control the agents' obligations (one for participating agent)

Organisational Structures

Define a pattern of information and control relationships between individuals, which shapes the types of interactions among them.

Aids coordination by specifying which actions an agent will undertake.

Social structure-based methods impose restrictions or norms on the behaviour of agents in a certain environment.

- avoid many potential conflicts, or ease their resolution

Sociology and Societies

The aim of any society is to allow its members to coexist in a shared environment and pursue their respective goals in co-operation with others.

Sociology is a discipline that results from an evolution of Philosophy in order to describe the interactions that arise among the members of a group, and the social structures that are established

Organisation studies

Organisation theory is a descriptive discipline, mainly focusing on explaining and understanding how organizations work.

Organisation design is a normative, design-oriented discipline that aims to produce the frameworks and tools required to create effective organisation.

- involves the creation of roles, communication processes and formal reporting relationships in an organization.
- Two phases:
    - Strategic grouping, which establishes the overall structure of the organization.
    - Operational design, which defines the more detailed roles and processes

Three main kinds of organizations:

- Markets
- Networks
- Hierarchies

Market Structures

There are agents that provide services, agents that require services (and pay for them), and intermediate agents.

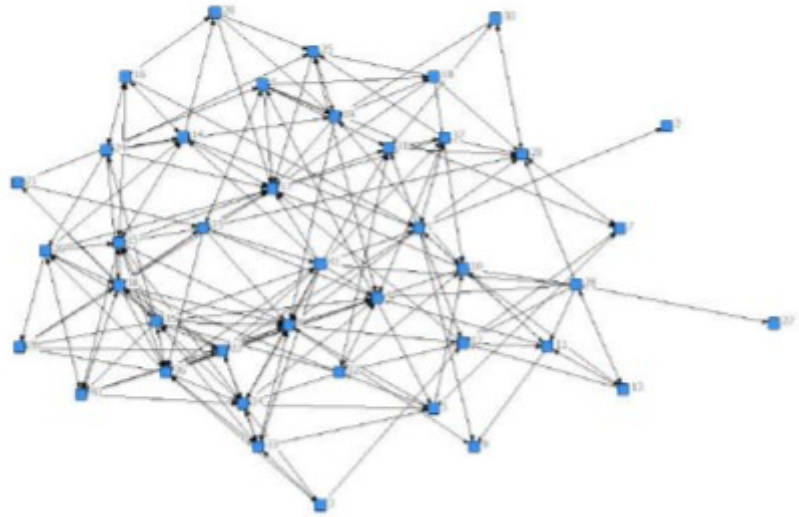
All these self-interested agents are involved in negotiations

Network structures

They are well-suited for environments where (dynamic) collaboration among parties is needed

There are contracts established in a dynamic way between the agents of the system

Coordination is achieved by mutual interest



Hierarchies

Agents are specialized in concrete tasks and fully collaborative.

Coordination is achieved through command and control lines in a closed society.

Social abstractions

Roles identify activities and services necessary to achieve social objectives and enable to abstract from the specific individuals that will eventually perform them.

- From the society design perspective, roles provide the building blocks for the agent systems that can perform the role .
- From the agent design perspective, roles specify the expectations of the society with respect to the agent's activity in the society.

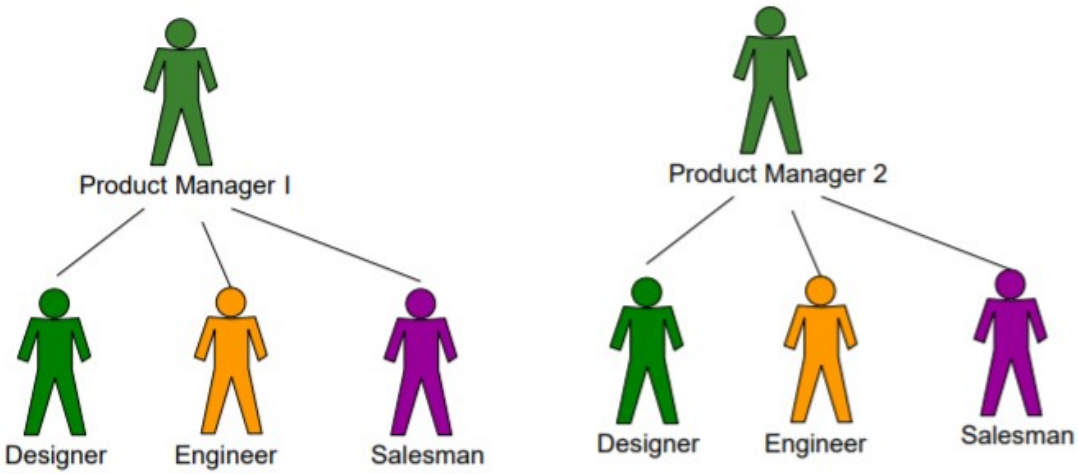
Role dependency between two roles means that one role is dependent on another role for the realization of its objectives.

Agent Societies - Models

Formal role models reflect social competence of agents (rights and obligations) and allow to ensure a global behaviour of the system while also preserving individual flexibility.

Interaction models reflect workflows and business processes.

Example Product hierarchy

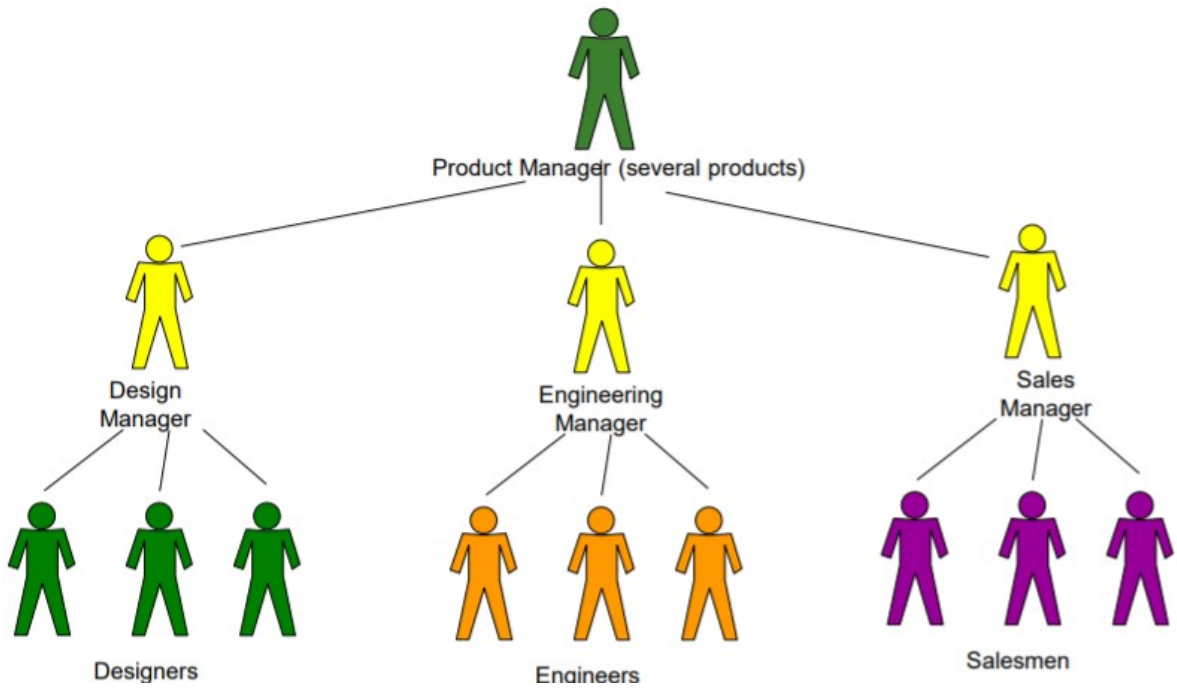


There is a dedicated team for each product (type of car) to be produced.

Easy coordination within each product team.

It might be a good option if products are quite different from each other.

Example 2 Functional Hierarchy



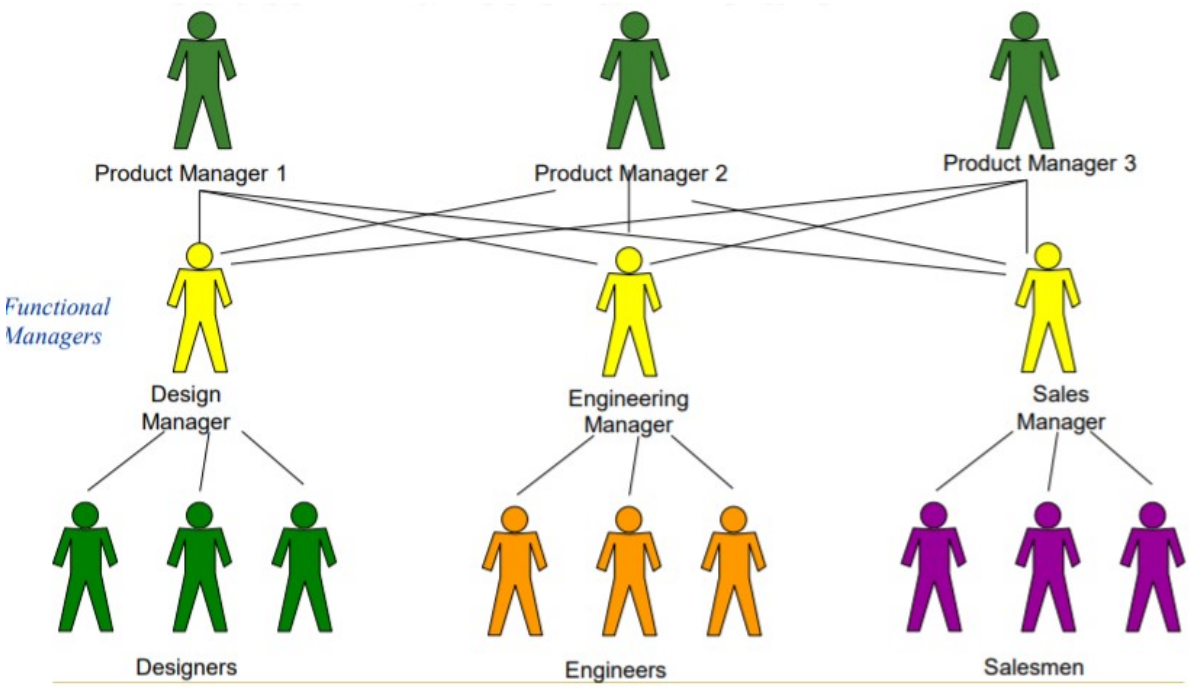
Actors with the same role work together under the supervision of a manager.

A general product manager coordinates all the activities of all the departments.

The specialised actors can work in tasks reusable in different products (e.g. designing and engineering the air-conditioning system).

The resources in each department can be easily shared by its members.

Example 3 Product + Functional Hierarchy



There are specialised departments, with a manager for each of them (department head, or functional manager).

There is a product manager for each product, who talks to the functional managers.

Functional managers act like brokers

Brokers are in contact with possible ”workers” and will choose the best for each task.

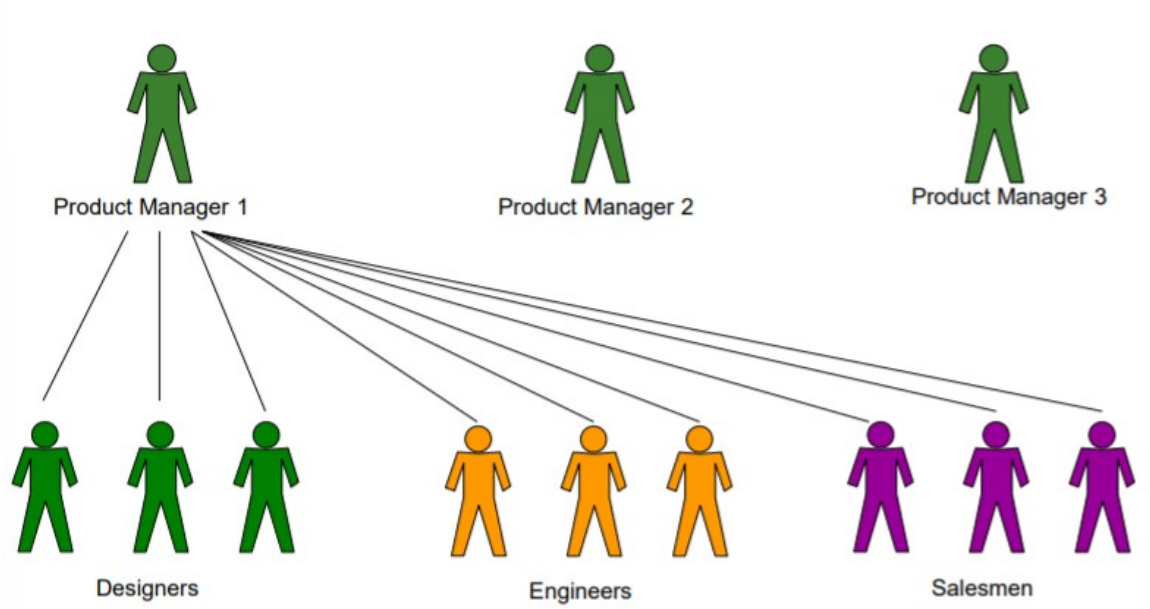
Few connections and communication messages are required.

A lot of work for functional managers.

- Receive requests from several product managers.
- Coordinate the work of a team of agents.
- Identify common subtasks, manage shared resources

The failure of one product manager does not affect the others.

Example Flat Structure



There is a product manager for each product, who talks directly to the low-level workers.

A product manager may have to communicate with many different agents, and these agents have different abilities/expertise/vocabulary.

Furthermore, there may be inefficiencies in the global behaviour:

- A designer could have work in 2 products, while another designer does not have any work.
- Two engineers could be working in similar problems in two different product.
- Difficult to solve even with a high-level global coordinator.