Multiagent Systems

Lin Zuoquan

Information Science Department

Peking University

lz@is.pku.edu.cn

http://www.is.pku.edu.cn/~lz/teaching/stm/saswws.html

Courtesy some graphic slides from online

Outline

- Multiagency
- Communication
- Interaction
- Reaching Agreement
- Cooperation and Coordination

Multiagency

- MAS
- Characteristics
- Issues

MAS

- A multiagent system (MAS) is a collection of autonomous agents in which collective intelligence emerges from interaction, cooperation, competition, or coordination among the agents.

Characteristics of MAS

- Social behavior
  - interact through communication
- Social consciousness
  - cooperate and/or compete
- Self-interested
  - own goals, own preferences

Issues of MAS

- Communication: languages and protocols
- Interaction: social behaviours
- Cooperation: task decomposition and results synthesis
- Coordination and coalition: conflict resolution and maximize social welfare
- Competition: auction and negotiation

Communication

- Agent Communication Language
- KQML
- Other ACL

Agent Communication Languages

- Language
  - syntax
  - semantics
- Examples: KQML, ACL
- Communication among *heterogeneous* agents
  - no common data structures
  - no common messages
  - no common ontology

Needed for ACL

- Coordination, cooperation and negotiation among agents
- Communicate about intentions, self-interest
- ACL provides a higher abstraction layer that allows heterogeneous agents to communicate
- Add/remove agents in a running multiagent system

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.

Speech Acts(cont.)

- The origin of speech act theories are usually traced to Austin's 1962 book (*How to Do Things with Words)*
- Austin noticed that some utterances are rather like 'physical actions' that appear to *change the state of the world*
- Examples would be:
- – declaring war;
- – '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

Speech Acts(cont.)

- Searle (1969) identified various 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.

Speech Acts(cont.)

- In general, a speech act can be seen to have two components:
  - a *performative verb*:

    (e.g., request, inform, . . .)
  - *propositional content*:

    (e.g., “the door is closed”)

Speech Acts(cont.)

- 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 (plan based semantics)
- Note that a speaker cannot (generally) *force* a hearer to accept some desired mental state

Speech Acts(cont.)

Here is their semantics for *request*:

- *Request(s,h,A)*

  pre:

  – *s* believes *h* can do *A*

  (you don't ask someone to do something unless you think they can do it)

  – *s* believe *h* believe *h* can do *A*

  (you don't ask someone unless *they* believe they can do it)

  – *s* believe *s* want *A*

  (you don't ask someone unless you want it!)

  post:

  – *h* believe *s* believe *s* want *A*

  (the effect is to make them aware of your desire)

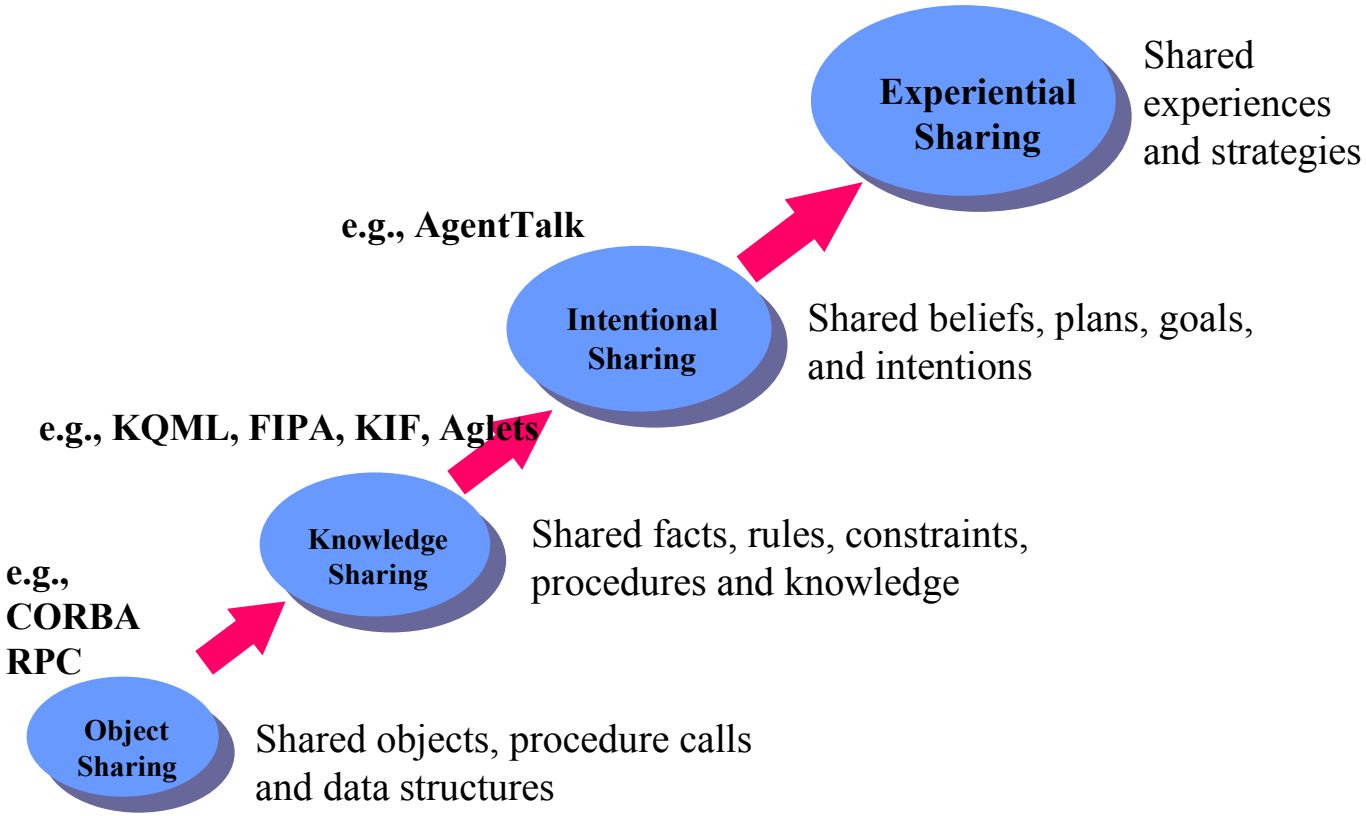
Structure of an ACL

- ACL = 2 components:
  - performative
  - content
- Vocabulary (words): e.g. reference to objects
- Messages (sentences): e.g. request for an action
- Distributed Algorithms (conversations): e.g. negotiating task sharing

Levels of ACL

- Object sharing (Corba, RPC, RMI, Splice): shared objects, procedures, data structures
- Knowledge sharing (KQML, FIPA ACL): shared facts, rules, constraints, procedures and knowledge
- Intentional sharing: shared beliefs, plans, goals and intentions
- Cultural sharing: shared experiences and strategies

Levels(cont.)


Experiential Sharing
Shared experiences and strategies
e.g., AgentTalk
Intentional Sharing
Shared beliefs, plans, goals, and intentions
e.g., KQML, FIPA, KIF, Aglets
Knowledge Sharing
Shared facts, rules, constraints, procedures and knowledge
e.g., CORBA RPC
Object Sharing
Shared objects, procedure calls and data structures

KQML

- KQML
- KIF
- ACL

KQML

- KQML (Knowledge Query and Manipulation Language)
  - a typical ACL, developed by the ARPA knowledge sharing initiative
  - a language and a set of protocols that support computer programs in identifying, connecting with and exchanging information with other programs
  - comprised of two parts:

    - KQML
    - KIF (Knowledge Interchange Format),
      - Message Format
      - Communication protocol
      - Communication Facilitators

&KIF

- KQML is an ‘outer’ 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*
  - a prefix version of the language of first-order logic with various extensions to enhance its expressiveness

Content in KQML

- KIF

  = predicate calculus in Lisp form

- agents are logical reasoners
- Several agent systems use KIF as a basis

  E.g. IBM's ABE (Agent Building Environment)

Features of KQML

- A high-level communication language and protocol for exchanging information independent of content syntax and ontology
- A language in which to wrap information offering a uniform view of an information agent as a knowledge base
- An extensible set of performatives expressing a belief or an attitude toward some information

Features

- Characteristics of KQML
  - Simple
  - Concise
  - Light weight
  - Network
  - Platform Independent

Formalisms for Content

- Content languages must make formulating and responding to performatives decidable!

→ restrict content language to be less expressive than predicate calculus

Examples:

- description logics
- constraint content language (CCL)

Ontology

- In order to be able to communicate, agents must have agreed a common set of terms.
- A formal specification of a set of terms is known as a *ontology.*
- The knowledge sharing effort has associated with it a large effort at defining common ontologie.

  — software tools like *ontolingua* for this purpose.

Example KQML/KIF dialogue. . .

```
A to B:     (ask-if
                    (> (size chip1) (size chip2)))
B to A:     (reply true)
B to A:     (inform (= (size chip1) 20))
B to A:     (inform (= (size chip2) 18))
```

Ontology(cont.)

- Ontology: a vocabulary for the domain knowledge
- Ontological engineering: representing various ontology
- The five-step methodology
  - Decide what to talk about
  - Decide on a vocabulary of predicates, functions and constants
  - Encode general knowledge about the domain
  - Encode a description of the specific problem instance
  - Pose queries to the inference procedure and get answers

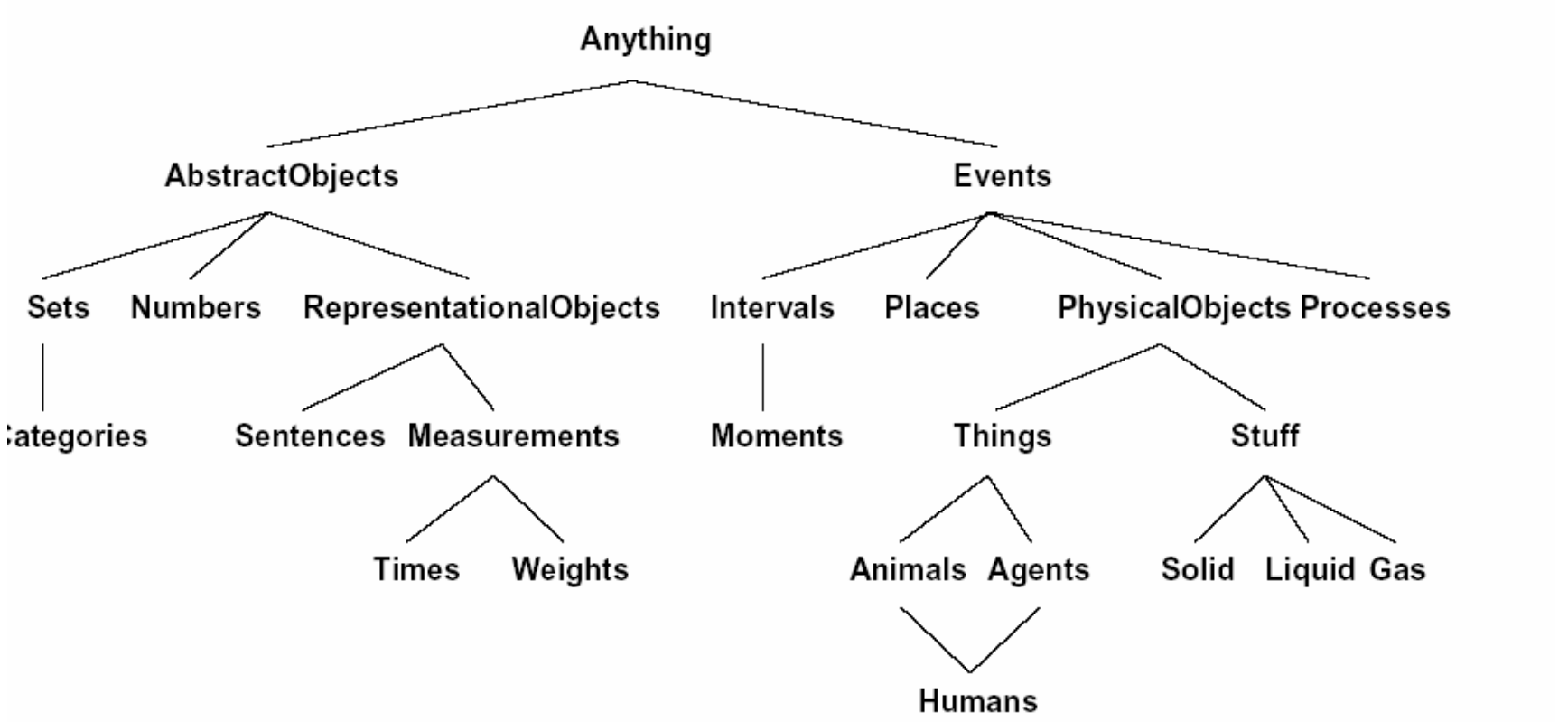
Ontology(cont.)

A general-purpose ontology has advantages over special-purpose one

- Categories
- Measures
- Composite objects
- Time, Space, and Change
- Events and Processes
- Physical objects
- Substances
- Mental objects and belief

Ontology(cont.)

The world ontology



KQML Message Format

- The following is a message in KQML from agent *joe*, representing a query about the price of a share of MS stock

```
(ask-one
        :sender joe
        :receiver stock-server
        :content price(MS, ?price)
        :reply-with ms-stock
        :language PROLOG
        :ontology NYSE-TICKS)
```

A KQML Message

- The following is a message in KQML from agent *stock-server,* answering joe's query

```
(tell
      :sender stock-server
      :receiver joe
      :content price(MS, 15.8)
      :in-reply-to ms-stock
      :language PROLOG
      :ontology NYSE-TICKS)
```

KQML Syntax

(ask-one ← performative

:sender *joe*

:receiver *stock-server*

:content *price(MS, ?price)* ← values

:reply-with *ms-stock*

:language *PROLOG*

:ontology *NYSE-TICKS*)

arameters

A performative is an expression that serves to effect a transaction or that constitutes the performance of the specified act by virtue of its utterance

Keywords and their Meaning

:sender	symbol identifying the sender
:receiver	symbol identifying the recipient
:content	the information about which the performative expresses an attitude
:language	language in which :content is expressed
:ontology	the name of the ontology (e.g., set of term definitions) used in the :content parameter
:reply-with	identifier that must appear in the reply
:in-reply-to	symbol from reply-with field of the message being answered.

Reserved Performative Names

- Basic informational performatives

  tell, deny, cancel, untell,
- Database performatives

  insert, delete, delete-one, delete-all
- Response performatives

  error, sorry
- Basic query performatives

  evaluate, reply, ask-if, ask-one, ask-all

Names(cont.)

- Multi-response query performatives

  stream-all, eos
- Basic effector performatives

  achieve, unachieve
- Generator performatives

  standby, ready, next, rest, discard, generator
- Capability-definition performatives

  advertise

Names(cont.)

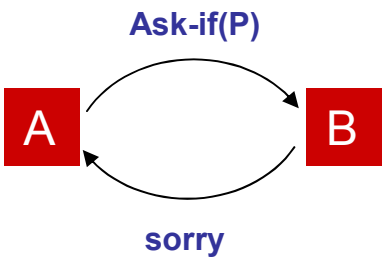
- Capability-definition performatives

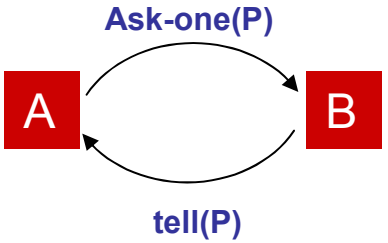
  subscribe, monitor
- Networking performatives

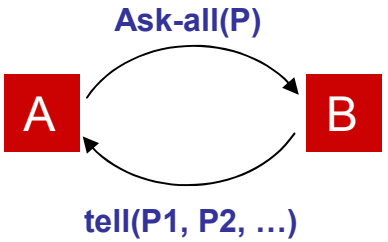
  register, unregister, forward, broadcast, transport-address
- Facilitation performatives

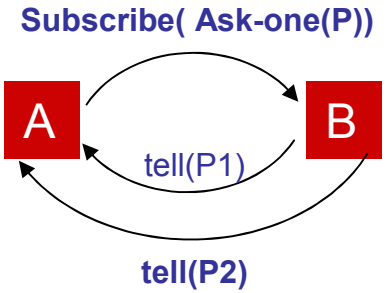
  broker-one, broker-all, recommend-one, recommend-all, recruit-one, recruit-all

KMQL queries



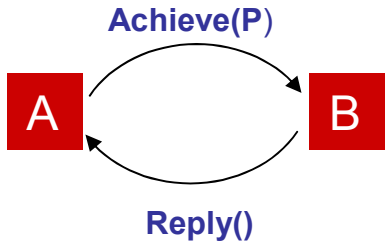






Requests: achieve & unachieve

Achieve: make a proposition true

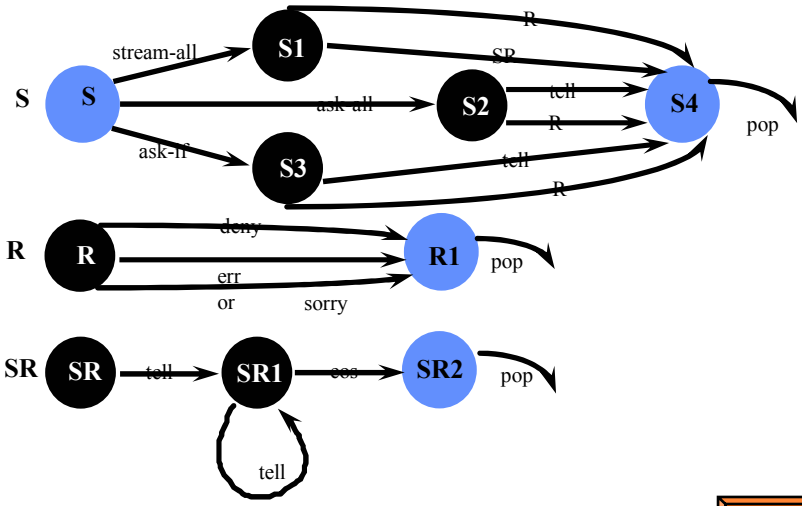


Unachieve: undo the previous achieve

Semantics for *tell(A,B,X)*

- bel(A,X), "A states to B that A believes X to be true (for A)."
- Pre(A): bel(A,X) AND know(A,want(B,know(B,S)))
  where S may be bel(A,X) or NOT(bel(A,X))
- Pre(B): intend(B,know(B,S))
- Post(A): know(A,know(B,bel(A,X)))
- Post(B): know(B,bel(A,X))
- Completion: know(B,bel(A,X))

Comment: The completion condition and postconditions hold unless a *sorry* or *error* suggests B's inability to properly acknowledge the *tell*.



KQML Dialogue 1

ask-all performative asks for all the answers to a query.

```
(ask-all
        :sender joe
        :receiver stock-server
        :content price(IBM, [?price,?time])
        :reply-with ibm-stock
        :language PROLOG
        :ontology NYSE-TICKS)
```

Dialogue 1

The response for ask-all:

(tell

:sender *stock-server*

:receiver *joe*

:content *price(IBM, [14.0, 8:30]),*

price(IBM, [14.5, 9:00])

price(IBM, [13.1, 9:30])

:in-reply-to *ibm-stock*

:language *PROLOG*

:ontology *NYSE-TICKS*)

Dialogue 2

Stream-all is another way to ask-all. The responder is asked to send a series of performatives to answer the query.

```
(stream-all
        :sender joe
        :receiver stock-server
        :content price(IBM, [?price,?time])
        :reply-with ibm-stock
        :language PROLOG
        :ontology NYSE-TICKS)
```

Dialogue 2

Response of stream-all.

(tell :sender *stock-server* :receiver *joe*
:content *price(IBM, [14.0, 8:30])* :in-reply-to *ibm-stock*
:language *PROLOG* :ontology *NYSE-TICKS*)
(tell :sender *stock-server* :receiver *joe*
:content *price(IBM, [14.5, 9:00])* :in-reply-to *ibm-stock*
:language *PROLOG* :ontology *NYSE-TICKS*)
(tell :sender *stock-server* :receiver *joe*
:content *price(IBM, [13.1, 9:30])* :in-reply-to *ibm-stock*
:language *PROLOG* :ontology *NYSE-TICKS*)
(eos :in-reply-to *ibm-stock*)

Database performatives

```
(insert
    :content enrolled(Joe, IS001)
    :sender School-Admin
    :receiver John
    :ontology University
    :reply-with ia_enrollment
    :language PROLOG)
```

The sender requests the receiver to add the :content sentence to its KB. The performative can either fail or succeed

Database(cont.)

- Delete database performatives
  - delete: the sender requests the receiver to delete the :content sentence from its KB
  - delete-one: the sender requests the receiver to delete one sentence from its KB which matches :content
  - delete-all: the sender requests the receiver to delete all sentence from its VKB which matches :content

Generator Performatives

```
(standby
        :content (stream-all :content price(?stock, ?price))
        :sender agent1
        :receiver agent2
        :language PROLOG
        :reply-with q1
        :ontology NYSE-TICKS)
```

Agent 1 wants agent 2 to send the price of all the stocks with a streams

Generator(cont.)

```
(ready
      :sender agent2
       :receiver agent1
       :reply-with 2F0B
       :in-reply-to q1)
```

Agent 2 says that I am ready to reply you.

```
(next
      :sender agent1
      :receiver agent1
      :in-reply-to 2F0B)
```

Agent 1 wishes to receive the next respose.

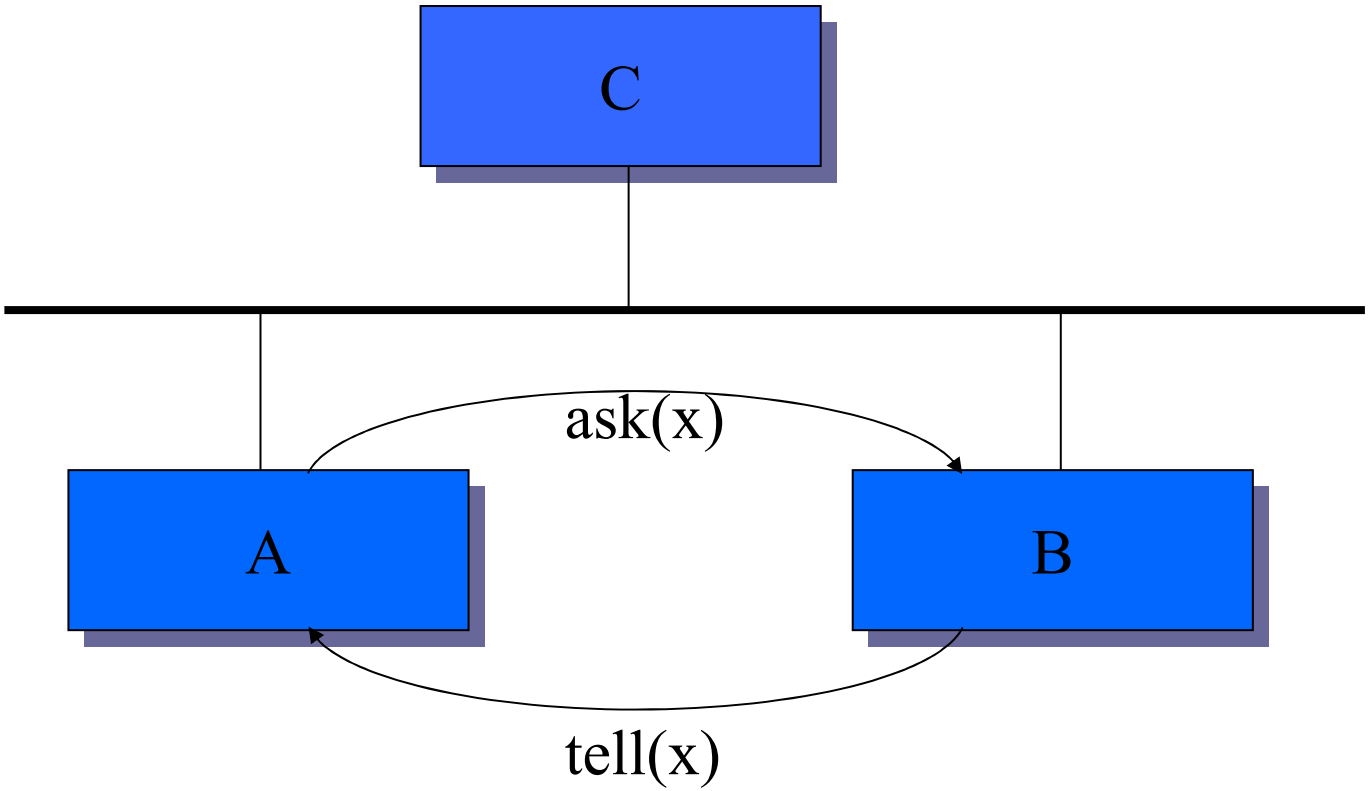
Generator(cont.)

```
(tell
        :content price(IBM, 15.0)
        :sender agent2
        :receiver agent1
        :language PROLOG
       :in-reply-to 2F0B
        :ontology NYSE-TICKS)
```

Facilitators

- Provide communication services
  - Maintaining registry of service names
  - Forwarding messages to named services
  - Routing messages based on content
  - Matching between information provider and clients

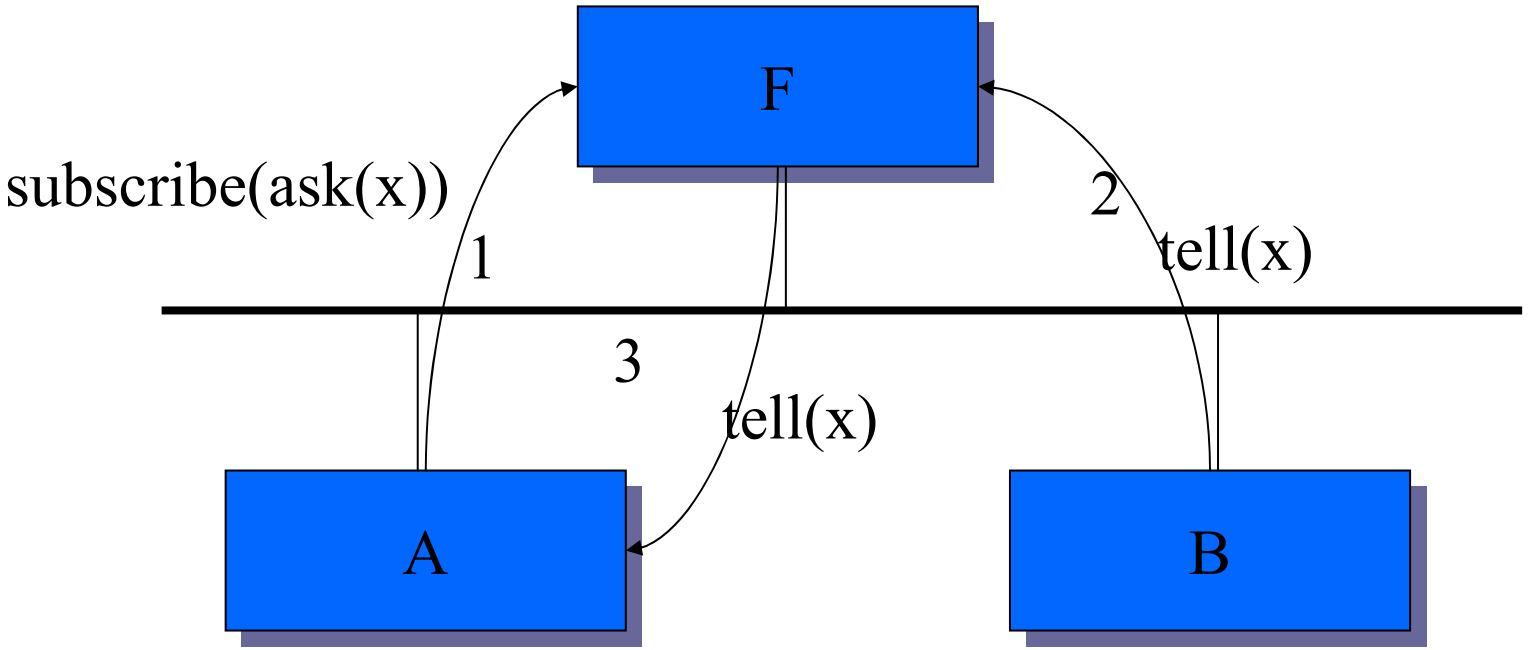
P2P Protocol



Facilitator(Monitor)

```
(subscribe
      :content  (ask-all
                        :content price(DEL-laptop ?price))
      :sender agent1)
or
(monitor
     :content  price(DEL-laptop ?price))
     :sender agent1)
```

Facilitator(Monitor)



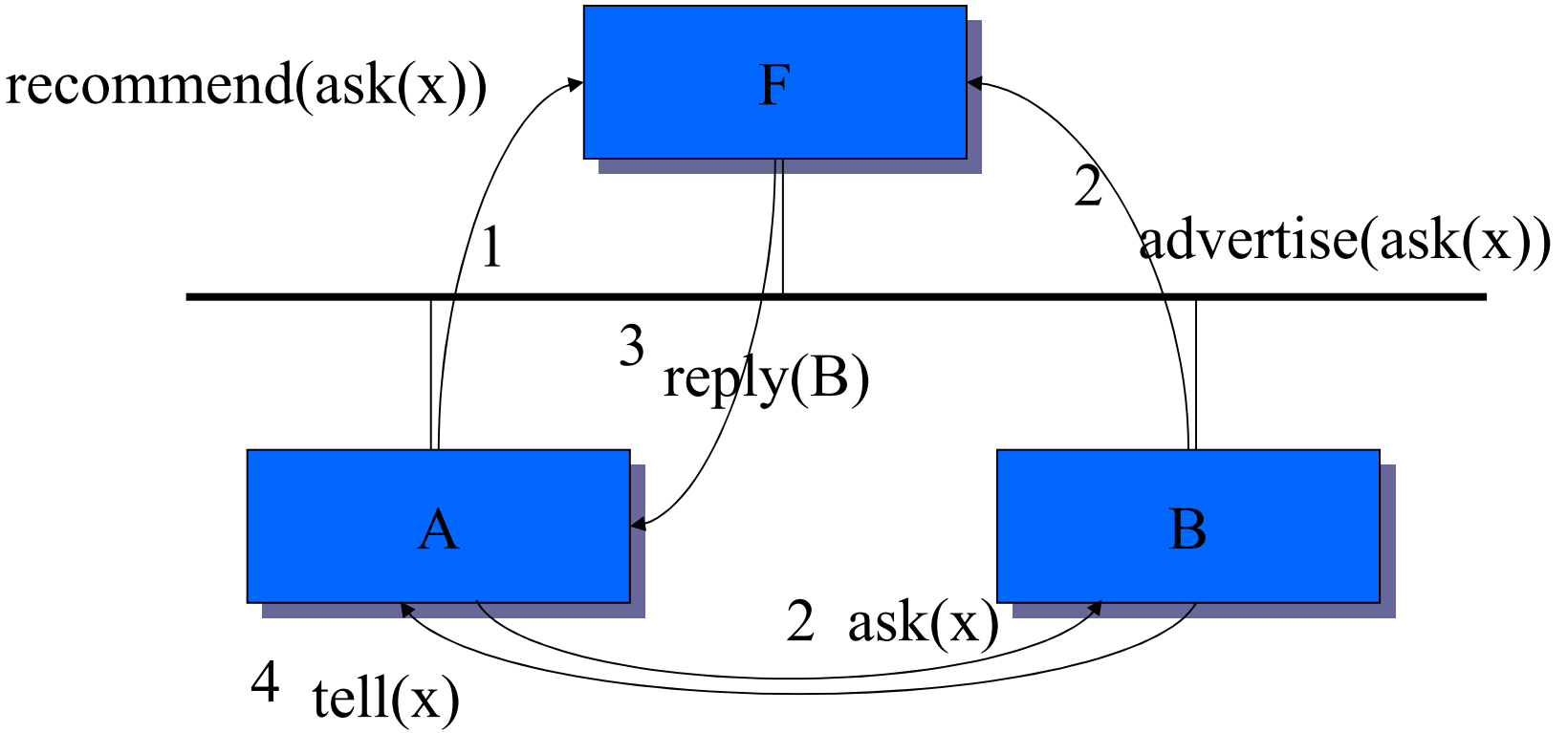
Facilitator(Recommend)

```
(recommend-one
      :content  (ask-all
                        :content price(DEL-laptop ?price))
     :sender agentA)
```

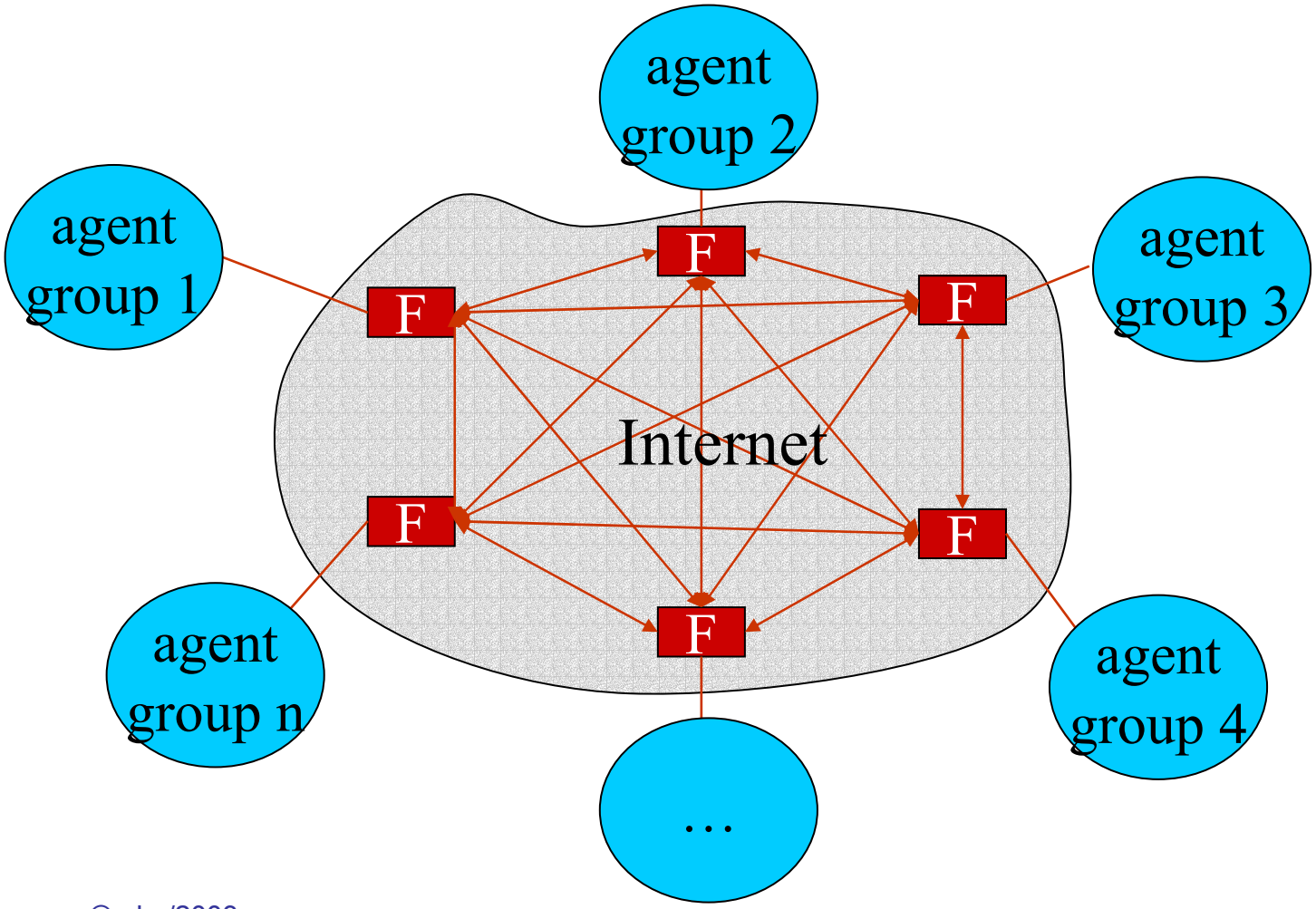
The sender wants the recipient to reply with the name of a single agent that is particularly suited to processing the embedded performative.

```
(reply
      :content  agentB)
```

Facilitator(Recommend)

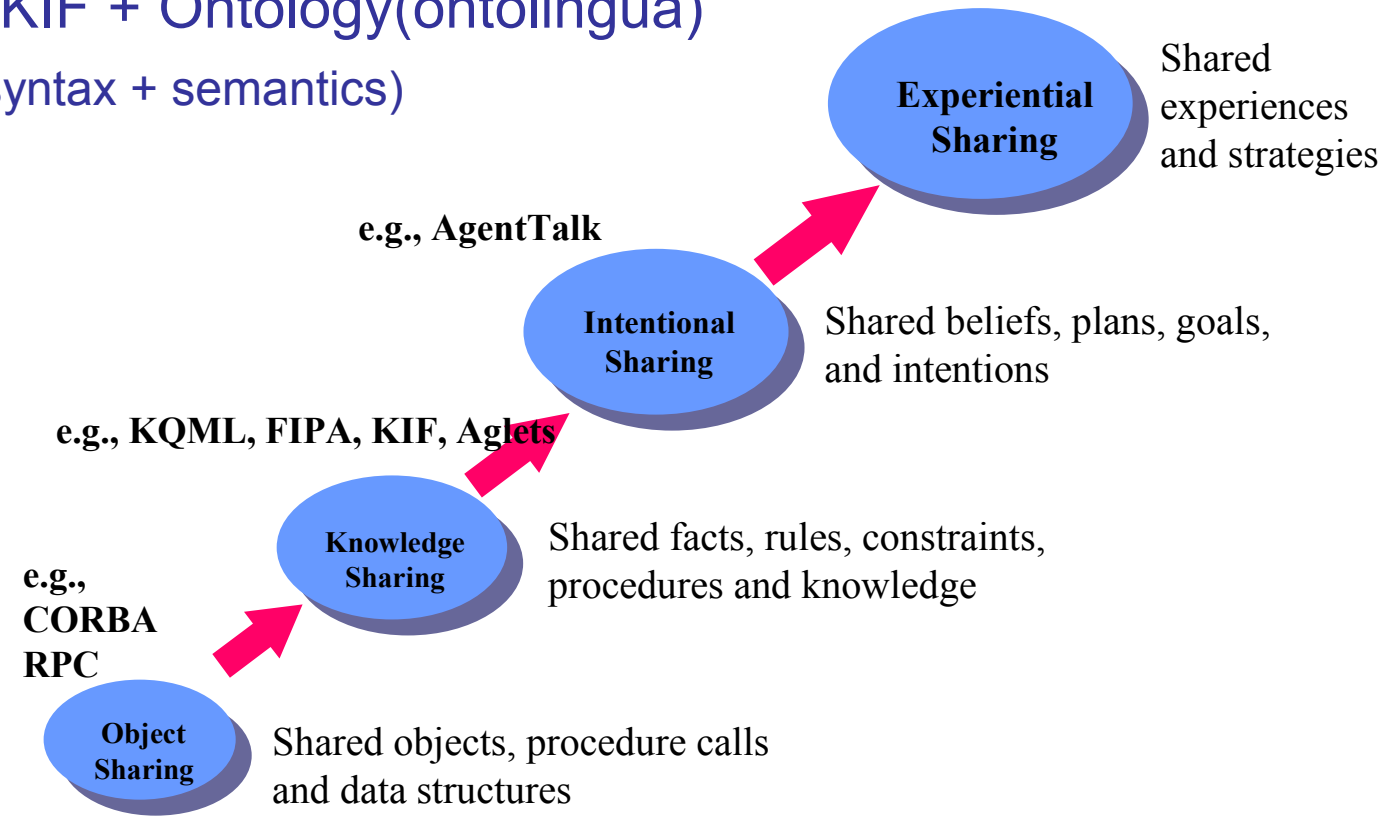


KQML Facilitators



Knowledge Sharing

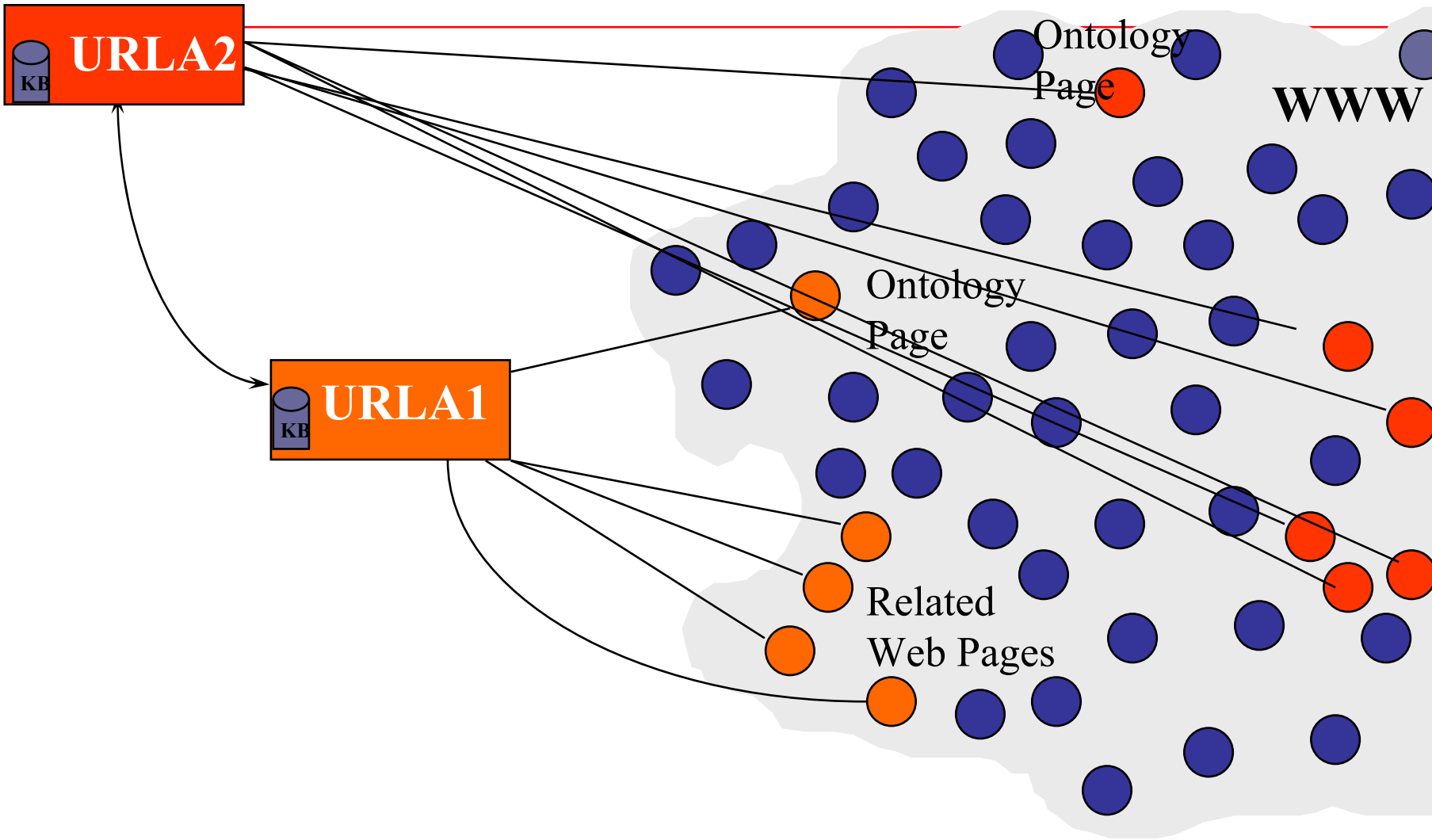
- Knowledge sharing approach
  = KQML + KIF + Ontology(ontolingua)
  (pramatics + syntax + semantics)



Semantic web markup with SHOE

- *Simple HTML Ontology Extensions* developed by UMCP PLUS Group
- Ontological mark-up tags allow the web page author to add semantic knowledge of the page's content
- Pages and sub-regions are the ground objects in the world
- Definitions for classes, relations, attributes, axioms, etc. are also defined on web pages
- *"URL agents"* are responsible for, and understand small groups of related web pages
  - each URL has an agent which can think, speak, and act for it
  - and communicate with other agents with similar pages

URL Agents speak for one or more related web pages



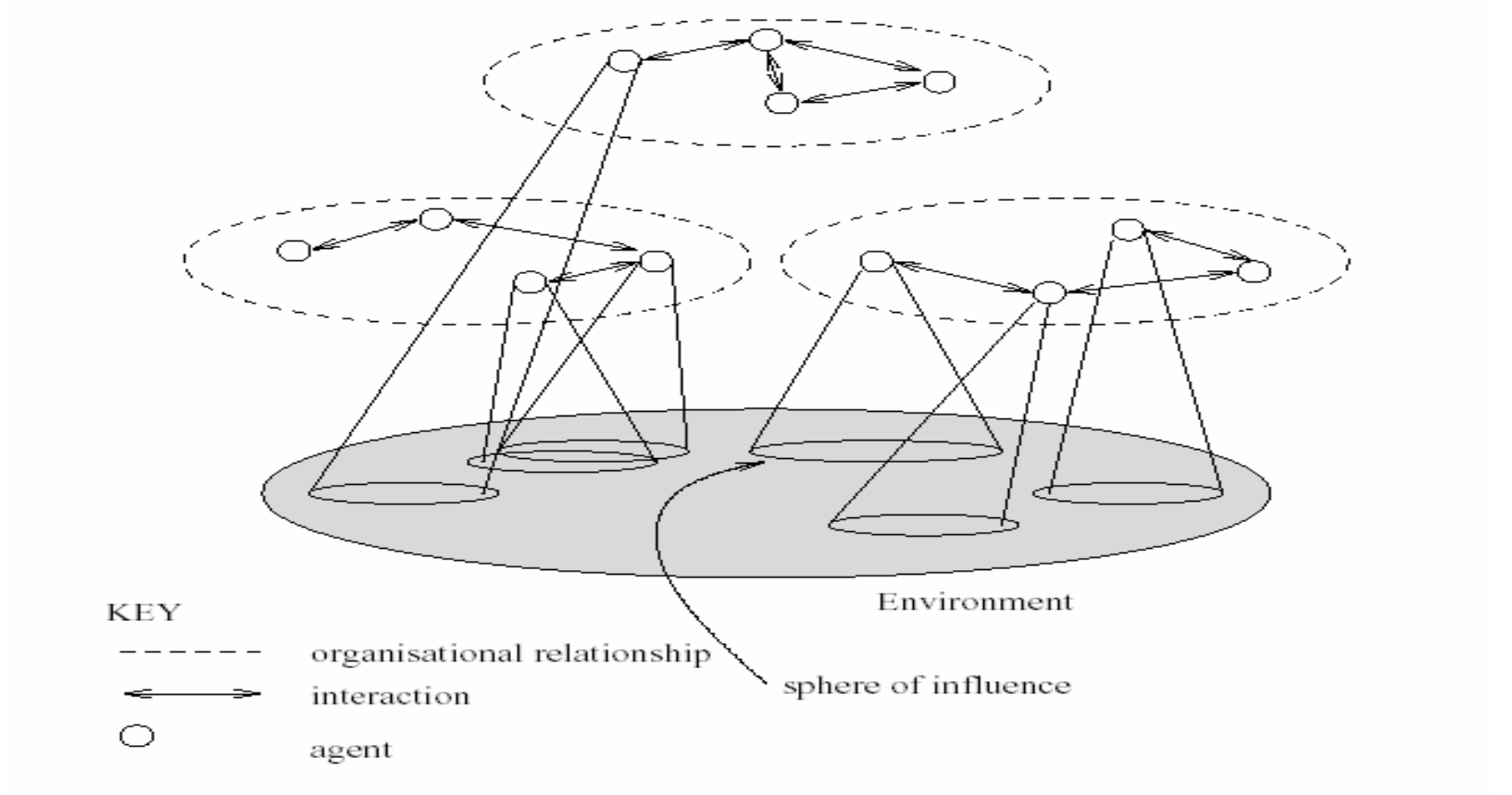
Other ACL

- Cohen & Lesveque
  - a few primitives only
  - heavily grounded on an agent theory
- FIPA (Foundation for Intelligent Physical Agents) 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

Interaction

- Interactions
- Utilities and Preferences
- Strategies

MAS


KEY
organisational relationship
interaction
agent
Environment
sphere of influence

MAS

- Thus a multiagent system contains a number of agents . . .
  - which 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

Interactions: Social Behaviors

Attributes of interaction

- frequency: low …high
- persistence: short-term … long-term
- level: signal-passing…knowledge-intensive
- variability: fixed …changeable protocol
- purpose: competitive …cooperative

A Simple Interaction Environment

Consider a MAS with only two agents

- The agents are denoted $Ag=\{i,j\}$
- All action the agents can perform are $Ac=\{a_1,a_2,\ldots\}$
- All possible outcomes (states) of the system are $\Omega=\{\omega_1,\omega_2,\ldots\}$
- The interaction between agents is modelled as a sequence of *encounters*
- In each encounter, both agents simultaneously choose an action to perform, and as a result of the actions, an outcome in Ω will result

MAS Encounters

- Environment behaviour given by *state transformer function*:

$$\tau: Ac \times Ac \rightarrow \Omega$$

system outcomes

agent i's action

agent j's action

Utilities and Preferences

- Assume agent to be *self-interested,* they *have preferences over how the environment is,* which are captured by *utility functions*:

$$u_i: \Omega \rightarrow \mathbf{R}$$

$$u_j: \Omega \rightarrow \mathbf{R}$$

- Utility functions lead to preference orderings over outcomes:

$$\omega <_i \omega' \text{ iff } u_i(\omega) < u_i(\omega')$$

$$\omega <_j \omega' \text{ iff } u_j(\omega) < u_j(\omega')$$

Decision Making

Decision-making in MAS is normally not as easy as in single utility-based agent systems because none of the agents can determine the outcome of the system. The actual outcome depends on the combination of each agent's action.

Decision-Making(cont.)

- Suppose $Ac = \{c, d\}$, *e.g.,* assume each agent has just two possible actions that it can perform c ("cooperate") and d ("defect").
- If the environment is deterministic, only four possible different outcomes can be produced by the system:

$$\tau(c,c) = \omega_1,\ \tau(c,d) = \omega_2,\ \tau(d,c) = \omega_3,\ \tau(d,d) = \omega_4$$

- Suppose that the utility function for agent i is:

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

and the utility function for agent j is:

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

If you were agent i, what would you choose to do: c or d?

Decision-Making(cont.)

- Here is a state transformer function:

  $\tau(d,d)= \omega_1$, $\tau(d,c)= \omega_2$, $\tau(c,d)= \omega_3$, $\tau(c,c)= \omega_4$

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

- Here is another:

  $\tau(d,d)= \omega_1$, $\tau(d,c)= \omega_1$, $\tau(c,d)= \omega_1$, $\tau(c,c)= \omega_1$

(Neither agent has any influence in this environment.)

- And one more:

  $\tau(d,d)= \omega_1$, $\tau(d,c)= \omega_2$, $\tau(c,d)= \omega_1$, $\tau(c,c)= \omega_2$

(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:

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

  $u_j(\omega_1)=1,\ u_j(\omega_2)=4,\ u_j(\omega_3)=1,\ u_i(\omega_4)=4$
- With a bit of abuse of notation:

  $u_i(d,d)=1,\ u_i(d,c)=1\ ,\ u_i\ (c,d)=4\ ,\ u_i(c,c)=4$

  $u_j(d,d)=1,\ u_j(d,c)=4\ ,\ u_j\ (c,d)=1\ ,\ u_j(c,c)=4$
- Then agent *i*'s preferences are:

  $c,c\ >=_j\ c,d\ >=_j d,c\ >=_j d,d$
- Therefore, *c* is the *rational choice* for *i*.

(Because *i* prefers all outcomes that arise through *c* over all outcomes that arise through *d*.)

Payoff Matrix

payoff of agent i

payoff of agent j

agent j

agent i

	action *a*	action *b*
action *a*	(1, 1)	(1, 4)
action *b*	(4, 1)	(4, 4)

Agent i prefers to do *b* because no matter what agent j does it can receive a payoff at 4.

Agent j also prefers to do *b* because no matter what agent i does it can receive a payoff at 4.

Matrix(cont.)

agent j

agent i	action *a*	action *b*
action *a*	(1, 1)	(4, 1)
action *b*	(1, 4)	(4, 4)

In this case, the payoff of each agent depends on what the other agent does.
However, if both agents are *rational*, they will simultaneously choose to perform *b*.

Matrix(cont.)

agent j

agent i		action *a*	action *b*
	action *a*	(2, 2)	(5, 0)
	action *b*	(0, 5)	(3, 3)

In this case, the payoff of each agent also depends on what the other agent does.

However, we have no obvious solution for both agents. The final decision is determined by the strategy each agent uses.

Strategies of Game Players

- In game theory, a *strategy* for a player is a complete plan for playing a game, *i.e.*, the actions it is going to take for completing the game.
- We use S^i and S^j to represent all the possible strategies of agent i and j, respectively.
- A *strategy profile* is a pair (s',s'') of strategies such that $(s',s'') \in S^i \times S^j$, *i.e.*, the strategies of both agents.
- The payoff of a strategy profile for a play is the player's utility of the outcomes that the strategy profile leads to.

Dominant Strategies

- The strategy s_1 *dominates* the strategy s_2 for an agent if s_1 can always lead to higher payoff for the agent no matter whatever strategies the other agent uses.

  Formally, s_1 *dominates* s_2 for agent i if for any strategy s' of agent j, the payoff of the profile (s_1, s') is always higher than the payoff of the profile (s_2, s').
- A *dominant strategy* for a player is one that dominates every other strategy of that player. A rational agent will adopt a dominant strategy whenever it exists.
- The strategy profile *(s',s'')* is a *dominant strategy equilibrium* if s' and s'' is a dominant strategy for agent *i* and *j*, respectively.

Dominant Strategy Equilibrium

- Example

agent i \ agent j	action *a*	action *b*
action *a*	(1, 1)	(1, 4)
action *b*	(4, 1)	(4, 4)

Strategy $\{b\}$ is a dominant strategy for agent i.
Strategy $\{b\}$ is a dominant strategy for agent j.
Therefore (b,b) is a dominant strategy equilibrium.

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 one year.

Dilemma(cont.)

p1 \ p2	Confess	Don't confess
Confess	(2, 2)	(5, 0)
Don't confess	(0, 5)	(3, 3)

Note that the numbers of the payoff matrix do not refer to years in prison. They capture how good an outcome is for the agents – the shorter jail term, the better.

?? Dominant strategy

?? Why

Dilemma(cont.)

- The *individual rational* action is *defect*.

  This guarantees a payoff of no worse than 2, whereas cooperating guarantees a payoff of at most 1.
- So defection is the best response to all possible strategies: both agents defect, and get payoff = 2.
- But *intuition* says this is *not* the best outcome:

  Surely they should both cooperate and each get payoff of 3!

Weakly Dominant Strategy

- Investment Decision

agent i \ agent j	small	large
small	(8, 8)	(0, 16)
large	(16, 0)	(0, 0)

There is no dominant strategy for each agent.

Strategy {large} is a *weakly* dominant strategy for agent i.

Strategy {large} is a *weakly* dominant strategy for agent j.

Therefore (large, large) is a *weakly* dominant strategy equilibrium.

Why?

Equilibrium

Consider the following interaction

agent i \ agent j	action *a*	action *b*
action *a*	(14, 14)	(2, 16)
action *b*	(16, 2)	(1, 1)

There is no dominant strategy or weakly dominant strategies for both agents.

Nash Equilibrium

A strategy profile (*s′,s″*) is in *Nash equilibrium* if

1. under the assumption that agent *i* plays *s′*, agent *j* can do no better than play *s″*; and
2. under the assumption that agent *j* plays *s″*, agent *i* can do no better than play *s″*.

Therefore neither agent has any incentive to deviate from a Nash equilibrium.

Nash Equilibrium Strategy

Example

p1 \ p2	Confess	Don't confess
Confess	(2, 2)	(5, 0)
Don't confess	(0, 5)	(3, 3)

The strategy profile (*Confess*, *Confess*) is a Nash Equilibrium.

If a strategy profile is a dominant equilibrium, then it is the only Nash equilibrium as well.

Nash Equilibrium Strategy(cont.)

Example

C1 \ C2	Small	Large
Small	(8, 8)	(0, 16)
Large	(16, 0)	(0, 0)

Both (Large, Small), (Small, Large), (Large, Large) are Nash equilibriums.

Nash Equilibrium Strategy(cont.)

Example

agent i \ agent j	action *a*	action *b*	action *c*
action *a*	(0, 0)	(0, 44)	(0, 31)
action *b*	(44, 0)	(14,14)	(-1, 16)
action *c*	(31, 0)	(16, -1)	(1, 1)

(c,c) is the unique Nash equilibrium.

Your Turn!

agent i \ agent j	action *a*	action *b*	action *c*
action *a*	(2, -2)	(2, -2)	(2.5, -2.5)
action *b*	(1, -1)	(3,-3)	(2.5, -2.5)
action *c*	(2, -2)	(2, -2)	(2.5, -2.5)

- Dominant equilibrium:?
- Nash equilibrium:?

Nash Equilibrium(cont.)

- Not every interaction scenario has a Nash equilibrium.
- Some interaction scenarios have more than one Nash equilibrium.

Competitive and Cooperative Interactions

- The solution we suggested here are based on the assumption that the agents are *strictly competitive*.
- If the agents tends to cooperate, the concepts of equilibrium will not necessarily make sense.

p1 \ p2	Confess	Don't confess
Confess	(2, 2)	(5, 0)
Don't confess	(0, 5)	(3, 3)

Best solution?

Compitive and Zero-Sum Interaction

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

  $u_i(\omega) + u_j(\omega) = 0$ *for all* $\omega \in \Omega$

- Zero sum implies strictly competitive.
- Zero sum encounters in real life are very rare . . . but people tend to act in many scenarios as if they were zero sum.

Strategies of Iterated Games

If you are allowed to play a game more than once, you would take a higher risk to get a better-off. In this case, the outcome can reach the best payoff if the game can be played in unlimited times and each player takes appropriate strategies.

(If you know you will be meeting your opponent again, then the incentive to defect appears to evaporate.)

Iterated Games(cont.)

Suppose you play iterated prisoner's dilemma against a *range* of opponents . . .What strategy should you choose, so as to maximise your overall payoff?

Axelrod (1984) investigated this problem, with a computer tournament for programs playing the prisoner's dilemma.

- ALL-D: "Always defect" — the *hawk* strategy;
- RNDOM: randomly generate play strategies.
- TIT-FOR-TAT: On first round cooperate, then do what the opponent did on the previous round.
- TESTER: defecting first to test its opponent to retaliate by defecting first. If so, then play TIT-FOR-TAT, otherwise play more cooperating than defecting.

Reaching Agreements

- Auctions
- Negotiations
- Argumentations

Agreement

- How do agents *reaching agreements* when they are self interested?
- In an extreme case (zero sum encounter) no agreement is possible — but in most scenarios, there is potential for *mutually beneficial agreement* on matters of common interest.
- The capabilities of *negotiation* and *argumentation* are central to the ability of an agent to reach such agreements.

Mechanisms, Protocols, and Strategies

- Negotiation is governed by a particular *mechanism*, or *protocol*.
- The mechanism defines the “rules of encounter” between agents.
- *Mechanism design* is designing mechanisms so that they have certain desirable properties.
- Given a particular protocol, how can a particular *strategy* be designed that individual agents can use?

Mechanism Design

Desirable properties of mechanisms:

- *Convergence/guaranteed success*.
- *Maximising social welfare*.
- *Pareto efficiency*.
- *Individual rationality*.
- *Stability*.
- *Simplicity*.
- *Distribution*.

Competition Mechanism

Competition is normally unavoidable in multiagent environment since agents are self-interested. However conflicts can be avoidable with appropriate competition mechanism. A well-formed competition mechanism can help agents to reach mutually beneficial agreements. The following properties can be considered in the competition mechanism design:

- **Guaranteed success**: the mechanism guarantees an agreement can be reached eventually.
- **Maximizing social welfare**: the mechanism makes the sum of the utilities of each agent to be maximal.

Competition Mechanism(cont.)

- **Pareto efficiency:** an outcome is *Pareto efficient* if there is no other outcome that will make at least one agent better off without making at least one other agent worse off.
- **Stability:** a competition mechanism is *stable* if it provides all agents with an incentive to behave in a particular way.
- **Fairness:** the mechanism does not discriminate in favour of any particular agents.
- **Distribution:** there is no a particular agent acting as arbitrator which can make decision for every agent.

Typical Competition Mechanisms

- Auction: *allocate goods or tasks to agents through market.*
- Negotiation: *reach agreements through interaction.*
- Argumentation: *resolve confliction through debates.*
- …

Nash's Solution doesn't always Work

- Investment Decision

agent i \ agent j	Small	Large
Small	(8, 8)	(0, 16)
Large	(16, 0)	(0, 0)

Nash equilibrium:

(Large, Small), (Small, Large) and (Large, Large)

What is the best strategy for each agent?

Possible Solutions

- Auction

  Only one company is allowed to enter the market. An auction is then to be set up. The winner will be admitted in. The outcome will be 16 – bid amount

- Negotiation

  All companies which are going to enter the market sit down to find a way to carve up the market. The possible outcome would be that all the company uniformly gets a portion of the market.

- Argumentation

  Argumentation is the process of attempting to convince others of something.

What Is Auction

- An auction takes place between an agent known as the *auctioneer* and a collection of agents known as the *bidders*.
- The goal of the auction is for the auctioneer to allocate the good to one of the bidders.
- In most settings the auctioneer desires to maximize the price; bidders desire to minimize price.

Typical Auction Protocols

- English Auction
- Dutch Auction
- First-price sealed-bid auction
- Vickrey auction
- Continuous single-sided auction
- Continuous double auction

English Auctions

Procedure of English Auction:

- the auctioneer starts off by suggesting a *reservation price* for the good. If no agent is willing to bid more than the reservation price, then the good is allocated to the auctioneer for the amount;
- bids are then invited from agents, who must bid more than the current highest bid.
- when no agent is willing to raise the bid, then the good is allocated to the agent that has made the current highest bid, and the price they pay for the good is the amount of this bid.

Auction Parameters

- Goods can have:
  - private value: *how much the goods is worth to you.*
  - public value: *how much the goods is worth to everybody.*
  - correlated value: *how much you'd like to pay for the goods.*
- Winner determination may be:
  - first price: *the agent that bids most gets the good.*
  - second price: *the agent that bids most gets the good at the second highest price.*
- Bids may be:
  - open cry: *every agent can see other agent's bids.*
  - sealed bid: *only auctioneer can see every agent's bid.*

Auction Parameters(cont.)

- Bidding may be:
  - *one shot: one round determine the winner.*
  - *ascending: start from low price, end with high price*
  - *descending: start from high price, end with low price*

English Auctions(cont.)

- Properties of English auction
  - first-price
  - open cry
  - ascending
- Dominant strategy is for an agent to successively b a small amount more than the current highest bid until it reaches their valuation, then withdraw.
- Susceptible to
  - winners curse;
  - shills.

Negotiation

- *Negotiation* is the process of reaching agreements on matters of common interest. It usually proceeds in a series of rounds, with every agent making a proposal at every round.

Issues in Negotiation

- Negotiation Space: All possible deals that agents can make, i.e., the set of candidate deals.
- Negotiation Protocol: – A rule that determines the process of a negotiation: how and when a proposal can be made, when a deal has been struck, when the negotiation should be terminated, and so.
- Negotiation Strategy: When and what proposals should be made.

Typical Negotiation Problems

- **Task-Oriented Domains(TOD):** Domains in which an agent's activity can be defined in terms of a set of tasks that it has to achieve. The target of a negotiation is to minimize the cost of completing the tasks.
- **State-Oriented Domains(SOD):** Domains where each agent is concerned with moving the world from an initial state into one of a set of goal states. The target of a negotiation is to achieve a common goal.
- **Worth-Oriented Domains(WOD):** Domains where agents assign a worth to each potential state, which captures its desirability for the agent. The target of a negotiation is to maximize mutual worth.

Formalization of TOD

A Task-Oriented Domain TOD is a triple $<T, Ag, c>$

where:

- T is a finite set of all possible tasks;
- $Ag=\{A_1, A_2, \ldots, A_n\}$ is a list of participant agents;
- $c: \wp(T) \rightarrow \mathbf{R}^+$ defines *cost* of executing each subset of tasks.

Assumptions on cost function:

1. $c(\phi) = 0$.
2. The cost of a subset of tasks does not depend on who carry out them (Idealized situation).
3. Cost function is monotonic, which means that more tasks, more cost.

Examples of TOD

- Parcel Delivery

Several couriers have to deliver sets of parcels to different cities. The target of negotiation is to reallocate deliveries so that the cost of travel to each courier is minimal.

- Database Queries

Several agents have access to a common database, and each has to carry out a set of queries. The target of negotiation is to arrange queries so as to maximize efficiency of database operations (Join, Projection, Union Intersection, …) .

Argumentation

- Argumentation is the process of attempting to convince others of something.
- Gilbert (1994) identified 4 modes of argument:

1. *Logical mode*.

"If you accept that *A* and that *A* implies *B*, then you must accept that *B*".

2. *Emotional mode*.

"How would you feel if it happened to you?"

3. *Visceral mode*.

"Cretin!"

4. *Kisceral mode*.

"This is against Christian teaching!"

Logic-based Argumentation

Basic form of logical arguments is as follows:

Database |- (Sentence, Grounds)

Where

- *Database* is a (possibly inconsistent) set of logical formulae;
- *Sentence* is a logical formula known as the *conclusion*; and
- *Grounds* is a set of logical formulae such that:
  - *Grounds* is contained in *Database; and*
  - *Sentence* can be proved from *Grounds*.

Cooperation and Coordination

- Cooerations
- Coordinations

Traditional Approaches for Cooperation

- Application Domains
  - Distributed Problem Solving(DPS)
  - Parallel Problem Solving(PPS)
- Main issues
  - Problem decomposition
  - Sub-problem allocation
  - Sub-problem solving
  - Solution synthesis
- Properties
  - tasks known at design time
  - cooperative mechanism can be hardwired in at design time

Benevolent vs Self-interested

- Benevolent assumption:
  - agents share a common goal explicitly or implicitly
  - there is no potential for conflicts in benefit and resources, thus
  - they help each other whenever asked.
- Agents in MAS
  - Self-interested agents: agents act to further there own interests, possibly at expense of others.
  - Goals may be different.
  - Potential for conflicts between agents.
  - Tasks may change dynamically.
  - cooperative mechanism cannot be hardwired in at design time

General Cooperation Protocols

- No centralized control
- Domain independent
- Task changeable
- Not hardwired in

Typical Modes of Cooperation

- Task sharing: components of a task are distributed to component agents.
  - Homogeneous systems: all agents are homogeneous in terms of their capabilities.
  - Heterogeneous systems: agents have different capabilities.
- Result sharing: information (partial results etc) is distributed.
  - Proactively: one agent sends another agent some information because it believes the other will need it.
  - Reactively: one agent sends another information in response to a request that was previously set.

Coordination

- When several agents are working together, it is necessary to manage a certain number of supplementary tasks that are not directly productive but serve to improve the way in which those activities are carried out.
- The coordination is a mechanism or a set of activities which manages inter-dependencies between the activities of agents in order to improve performance of the whole system.

Situations which Need Coordination

- Resources are limited
- The agents need information and results which only other agents can supply.
- Mutual dependency
- Maximize overall performance
- Optimise overall costs

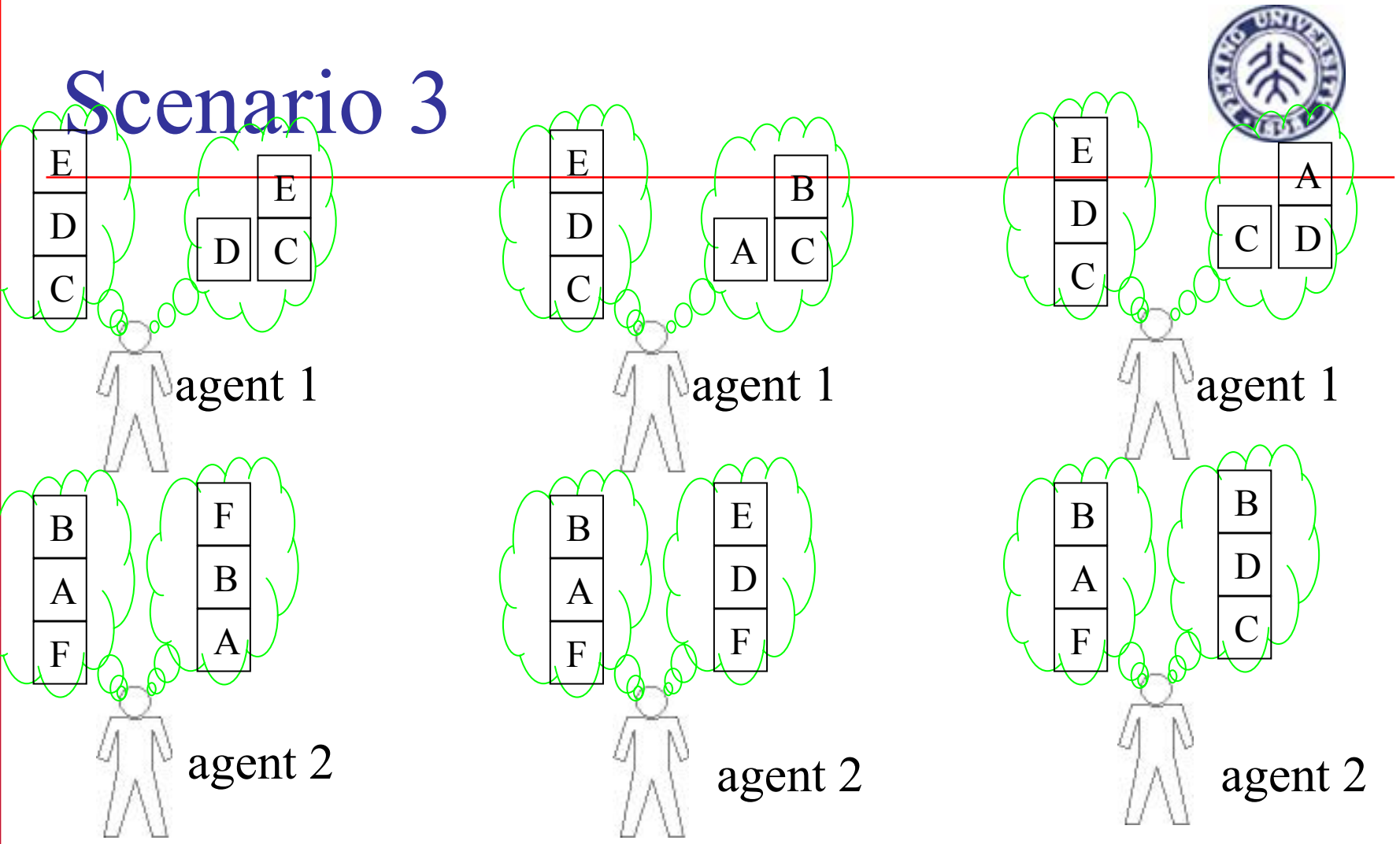
Scenario 1

Some robots are busy moving the contents of a room which served as a stone for small equipment. But the door and the corridor which give access to this store are not very wide, and two robots cannot pass each other except sideways. So the team has to get organized. *Why not form a chain and pass objects from one robot to another?*

Scenario 2

At the airport, where aircraft are taking off or landing every minute. Each plane needs to know the position and direction of the other planes which are or will be in the area of airport.

Scenario 3



(a) Neutral situation

(b) Cooperation situation with coordination

(c) Competition situation

Typical Approaches to Coordination

- Coordination by synchronisation
- Coordination by partial global planning
- Coordination by joint intention
- Coordination by mutual modelling
- Coordination by regulation and social laws

Readings & Assignment