DISTIBUTED INTELLIGENT SYSTEMS

5. Knowledge-Oriented Communication in Distributed Systems

Agent Communication

Ulises Cortés

Lluis Oliva

SID 2019

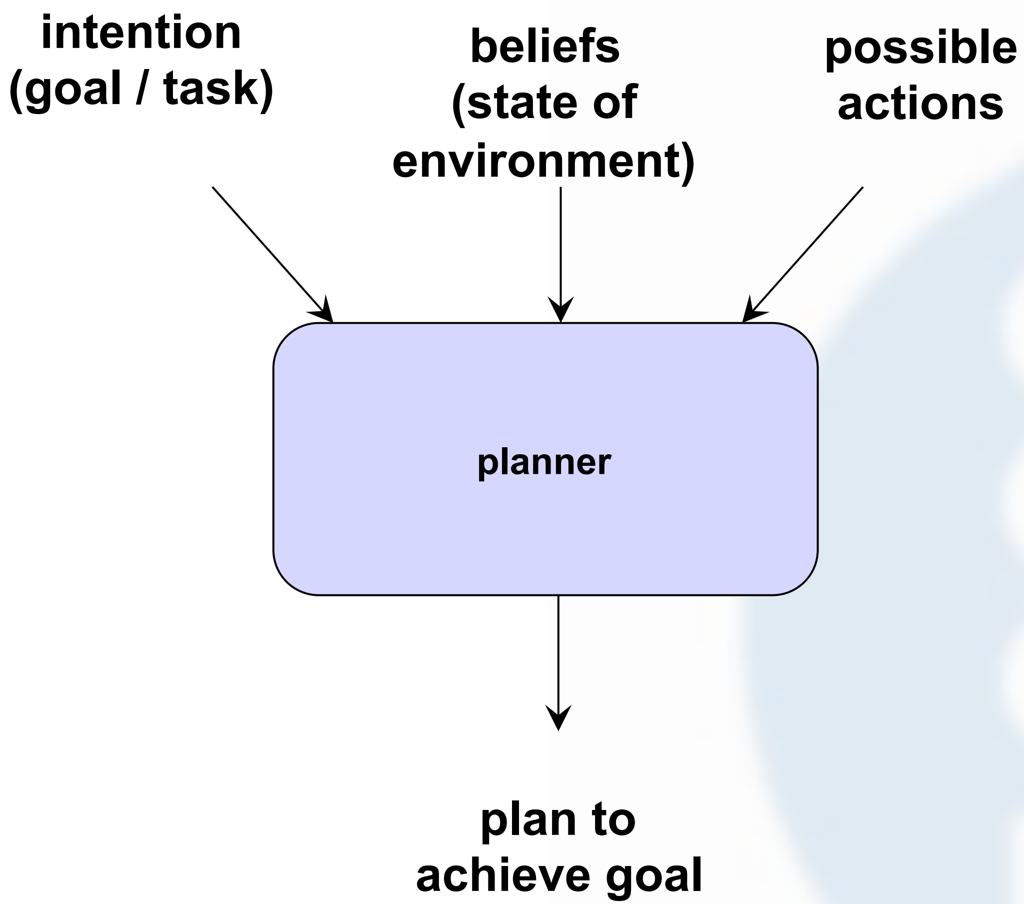
Knowledge Engineering and Machine Learning Group

UNIVERSITAT POLITÈCNICA DE CATALUNYA

https://www.kemlg.upc.edu

Contents

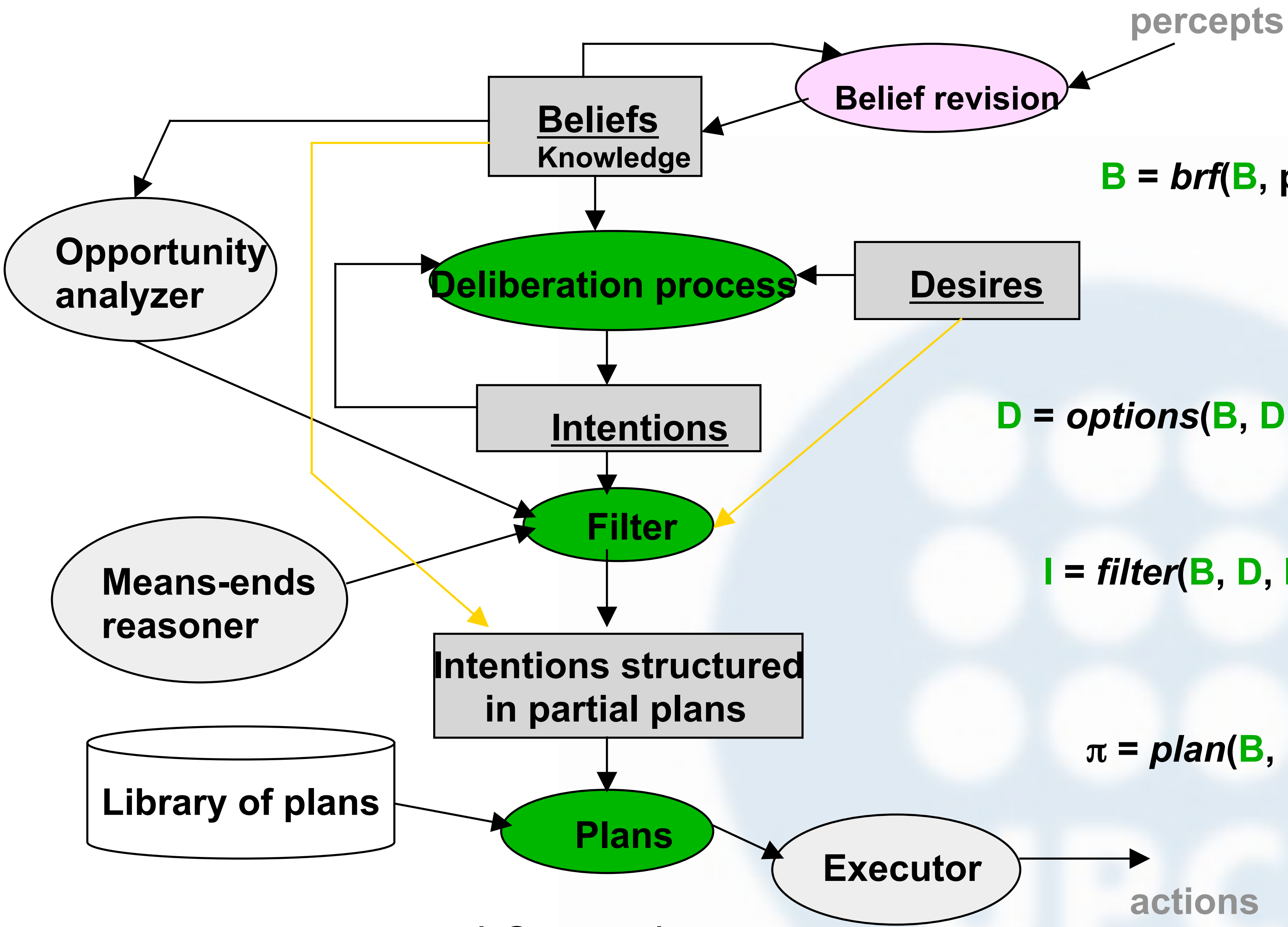
Practical Reasoning Agents: Means-ends Reasoning



means-ends reasoning

DISTRIBUTED INTELLIGENT SYSTEMS

BDI Architecture



$B = brf(B, p)$

$D = options(B, D, I)$

$I = filter(B, D, I)$

$\pi = plan(B, I)$

Practical Reasoning Agents: Intentions

1. agents are expected to **determine ways of achieving** intentions
   - *If I have an intention to Φ, you would expect me to devote resources to deciding how to bring about Φ*

2. agents **cannot** adopt intentions which **conflict**
   - *If I have an intention to Φ , you would not expect me to adopt an intention Ψ that was incompatible with Φ*

3. agents are inclined to **try again** if their attempts to achieve their intention fail
   - *If an agent's first attempt to achieve Φ fails, then all other things being equal, it will try an alternative plan to achieve Φ*

4. agents **believe** their intentions are **possible**
   - *That is, they believe there is at least some way that the intentions could be brought about.*

5. agents do **not believe** they will **not bring about** their intentions
   - *It would not be rational of me to adopt an intention to Φ if I believed that I would fail with Φ*

6. under certain circumstances, agents **believe** they **will bring about** their intentions
   - *If I intend Φ, then I believe that under "normal circumstances" I will succeed withΦ*

7. agents need **not intend** all the expected **side effects** of their intentions
   - *I may believe that going to the dentist involves pain, and I may also intend to go to the dentist — but this does not imply that I intend to suffer pain!*

Practical reasoning agents

- agent control loop

```
while true
        observe the world;
        update internal world model;
        deliberate about what intention to achieve next;
        use means-ends reasoning to get a plan for the
  intention;
        execute the plan
end while
```

- what are the options (desires) ?
- how to choose an option ?
- incl. filter
- chosen option ➔ intention …

- when to reconsider intentions !?

Implementing Practical Reasoning Agents

- Let us make the reasoning algorithm more formal:

Agent Control Loop Version 2
1. $B := B_0$; /* initial beliefs */
2. while true do
3.   get next percept ρ;
4. $B := brf(B, \rho)$;
5. $I := deliberate(B)$;
6. $\pi := plan(B, I)$;
7. $execute(\pi)$
8. end while

Implementing Practical Reasoning Agents

- this version: ***optimal behaviour if***
  - deliberation and means-ends reasoning take a vanishingly small amount of time;
    or
  - the world is guaranteed to remain static while the agent is deliberating and performing means-ends reasoning;
    or
  - an intention that is optimal when achieved at time t_0 (the time at which the world is observed) is guaranteed to remain optimal until time t_2 (the time at which the agent has found a course of action to achieve the intention).

Deliberation

- The *deliberate* function can be decomposed into two distinct functional components:

  - *option generation*
    in which the agent generates a set of possible alternatives;
    Represent option generation via a function, *options*, which takes the agent's current beliefs and current intentions, and from them determines a set of options (= *desires*)

  - *filtering*
    in which the agent chooses between competing alternatives, and commits to achieving them.
    In order to select between competing options, an agent uses a *filter* function.

Deliberation

Agent Control Loop Version 3

$B := B_0$;
$I := I_0$;
while true do
    get next percept ρ;
    $B := brf(B, \rho)$;
    $D := options(B, I)$;
    $I := filter(B, D, I)$;
    $\pi := plan(B, I)$;
    $execute(\pi)$
end while

Practical Reasoning Agents (cont.)

- If an option has successfully passed trough the filter function and is chosen by the agent as an intention, we say that

  the agent has made a commitment to that option
- Commitment implies temporal persistence of intentions; once an intention is adopted, it should not be immediately dropped out.

Question: How committed an agent should be to its intentions?

- degrees of commitments
  - **blind commitment**
    - ≈ fanatical commitment: continue until achieved
  - **single-minded commitment**
    - continue until achieved or no longer possible
  - **open-minded commitment**
    - continue until no longer believed possible

Commitment Strategies

- An agent has commitment both
  - to ***ends*** (i.e., the wishes to bring about)
  - and ***means*** (i.e., the mechanism via which the agent wishes to achieve the state of affairs)
- current version of agent control loop is overcommitted, both to means and ends

➔ modification: ***replan*** if ever a plan goes wrong

```
Agent Control Loop Version 4
B := B_0;
I := I_0;
while true do
    get next percept ρ;
    B := brf(B, ρ);
    D := options(B, I);
    I := filter(B, D, I);
    π := plan(B, I);
    while not empty(π) do
        α := hd(π);
        execute(α);
        π := tail(π);
        get next percept ρ;
        B := brf(B, ρ);
        if not sound(π, I, B) then
            π := plan(B, I)
        end-if
    end-while
end-while
```

Reactivity, replan

"Blind commitment"

Commitment Strategies

- this version still overcommitted to intentions:
  - never stops to consider whether or not its intentions are appropriate

➔ modification: stop for determining whether intentions have succeeded or whether they are impossible:

"*Single-minded commitment*"

Single-minded Commitment

```
Agent Control Loop Version 5
    B := B0;
    I := I0;
    while true do
        get next percept ρ;
        B := brf(B, ρ);
        D := options(B, I);
        I := filter(B, D, I);
        π := plan(B, I);
        while not empty(π)
                  or succeeded(I, B)
                  or impossible(I, B)) do
            α := hd(π);
            execute(α);
            π := tail(π);
            get next percept ρ;
            B := brf(B, ρ);
            if not sound(π, I, B) then
                π := plan(B, I)
            end-if
        end-while
    end-while
```

Dropping intentions that are impossible or have succeeded

Reactivity, replan

Intention Reconsideration

- Our agent gets to reconsider its intentions when:
  - it has completely executed a plan to achieve its current intentions; or
  - it believes it has achieved its current intentions; or
  - it believes its current intentions are no longer possible.

→ This is limited in the way that it permits an agent to *reconsider* its intentions

➔ modification:

Reconsider intentions after executing every action

"*Open-minded commitment*"

Open-minded Commitment

Agent Control Loop Version 6

```
B := B0;
I := I0;
while true do
    get next percept ρ;
    B := brf(B, ρ);
    D := options(B, I);
    I := filter(B, D, I);
    π := plan(B, I);
    while not (empty(π)
            or succeeded(I, B)
            or impossible(I, B)) do
        α := hd(π);
        execute(α);
        π := tail(π);
        get next percept ρ;
        B := brf(B, ρ);
        D := options(B, I);
        I := filter(B, D, I);
        if not sound(π, I, B) then
            π := plan(B, I)
        end-if
    end-while
end-while
```

Intention Reconsideration

- But intention reconsideration is *costly*!
  A dilemma:
  - an agent that does not stop to reconsider its intentions sufficiently often will continue attempting to achieve its intentions even after it is clear that they cannot be achieved, or that there is no longer any reason for achieving them
  - an agent that *constantly* reconsiders its attentions may spend insufficient time actually working to achieve them, and hence runs the risk of never actually achieving them
- Solution: incorporate an explicit ***meta-level control*** component, that decides whether or not to reconsider

```
Agent Control Loop Version 7

    B := B_0;
    I := I_0;
    while true do
        get next percept ρ;
        B := brf(B, ρ);
        D := options(B, I);
        I := filter(B, D, I);
        π := plan(B, I);
        while not (empty(π)
                  or succeeded(I, B)
                  or impossible(I, B)) do
            α := hd(π);
            execute(α);
            π := tail(π);
            get next percept ρ;
            B := brf(B, ρ);
            if reconsider(I, B) then
                D := options(B, I);
                I := filter(B, D, I);
            end-if
            if not sound(π, I, B) then
                π := plan(B, I)
            end-if
        end-while
    end-while
```

meta-level control

Possible Interactions

- The possible interactions between meta-level control and deliberation are:

Situation number	Chose to deliberate?	Changed intentions?	Would have changed intentions?	*reconsider*(...) optimal?
1	No	—	No	Yes
2	No	—	Yes	No
3	Yes	No	—	No
4	Yes	Yes	—	Yes

Intention Reconsideration

- Situations
  - In situation (1), the agent did not choose to deliberate, and as consequence, did not choose to change intentions.
    Moreover, if it *had* chosen to deliberate, it would not have changed intentions.
    the ***reconsider****(…)* function is behaving optimally.
  - In situation (2), the agent did not choose to deliberate, but if it had done so, it *would* have changed intentions.
    the ***reconsider****(…)* function is not behaving optimally.
  - In situation (3), the agent chose to deliberate, but did not change intentions.
    the ***reconsider****(…)* function is not behaving optimally.
  - In situation (4), the agent chose to deliberate, and did change intentions.
    the ***reconsider****(…)* function is behaving optimally.
- An important assumption: ***cost of reconsider****(…)* is *much* less than the cost of the deliberation process itself.

Optimal Intention Reconsideration

- Kinny and Georgeff's experimentally investigated effectiveness of intention reconsideration strategies
- Two different types of reconsideration strategy were used:
  - *bold* agents
    never pause to reconsider intentions, and
  - *cautious* agents
    stop to reconsider after every action
- *Dynamism* in the environment is represented by the *rate of world change*, γ

Optimal Intention Reconsideration

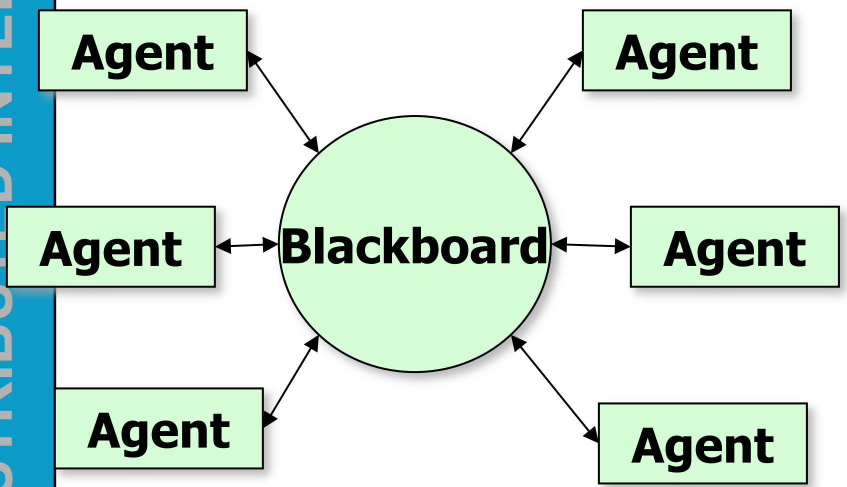
- Results (not surprising):
  - If γ is low (i.e., the environment does not change quickly),
    bold agents do well compared to cautious ones.
    - cautious ones waste time reconsidering their commitments while bold agents are busy working towards — and achieving — their intentions.
  - If γ is high (i.e., the environment changes frequently), cautious agents tend to outperform bold agents.
    - they are able to recognize when intentions are doomed, and also to take advantage of serendipitous situations and new opportunities when they arise.

Communication and ACLs

- **Communication**
  - **the basis for any interaction**
  - ***effected through signals***

Indirect communication

- **information available for all**
- **no direct communication**
- **simple architecture**



Message passing

- **direct exchange**
- **common language**
- **conversation - sequences of messages**

Motivation

Agents' interactions

- **Interaction between agents is unavoidable**
  - To achieve own goals,
  - To manage interdepencies
- **It should occur at *Knowledge-level***
  - *Which* goals?, *When*?, *Who* executes what?
- **Flexibility to start and to give answers**.
  - Synchronic, programs, *etc*

This implies a radical change in the way programs usually interact

Motivation

Knowledge sharing among agents requires communication

- The success of agent-based paradigm is based on the existence of heterogeneous and distributed software entities that communicate among themselves.

- The agents' diversity/heterogeneity implies the need for a common language.

Motivation: Communicating Agents…

- Mutual understanding:
  - Translation between representation languages
  - Share the language's semantic content
- Components in communication to be agreed:
  - Interaction protocol
    - How are conversations/dialogues structured?
  - Communication Language
    - What does each message means?
  - Transport protocol
    - How messages are actually sent and received by agents?
    - This is hidden from developers in Agent Platforms
  - Communication architecture/middleware
    - This has been fixed by FIPA Standards.

Communication and Knowledge Level

- Agents can be considered as (virtual) Knowledge Bases
- 3 representation layers
  - A language/formalism to represent domain knowledge
    - **Ontology**
  - A language to express propositions (to exchange knowledge)
    - **Content language** (for messages)
  - A language to express attitudes for those propositions
    - **Agent Communication Language** (for languages)

Agent Communication

- **Ability to exchange information requires:**

1. ability to *physically* exchange information
2. **common understanding**
   - exchanging knowledge requires mutual understanding
     → 2 keys
       - translation between languages
       - sharing semantic content
         - each agent has implicit assumptions on its own semantics
         - translation must preserve semantics!
   - to share knowledge, we must have a common semantics
   - can be shared via **common ontologies**
3. common language
4. interaction strategies / protocols

Agent Communication

- Ability to exchange information requires

1. ability to ***physically*** exchange information
2. common understanding
3. ***common language***

   incorporates two types of languages

   – content language
   – communication language

   ⟹ **Agent Communication Language**

4. interaction strategies / protocols

Levels in Agent Communication

- Four levels in communication:
  - **Message Semantics**
    - What does each message means?
    - 3 components
      - **Message type**: gives intensionality
      - **Message content**: contains the information
      - **Ontology** (the message refers to)
  - **Message Sintaxis**
    - How each message is expressed?
    - 2 components
      - Message structure: **Agent Communication Language**
      - Content codification: **Content Language**
  - **Interaction protocol**
    - How are conversations/dialogues structured?
      - **Agent Protocols**
  - **Transport protocol**
    - How messages are actually sent and received by agents?

Message Semantics: Speech Acts

- The majority of attempts to model agent communication are inspired in ***speech act theory.***

- Speech act theories are *pragmatic* theories of language, *i.e.*, theories of language use
  - they attempt to account for *how language is used* by people every day to achieve their goals and intentions

- The origin of Speech Act Theories is in the book ***How to Do Things with Words*** *(1962)* by Austin.

Message Semantics: Speech Acts (2)

- **Idea**: There are some utterances are rather like ***physical actions*** that appear to *change the state of the world*
  - **declaring war**
  - **'I now pronounce you man and wife'**
  - ***Goal!***
- In general, *everything* we utter is uttered with the intention of satisfying some goal or intention
- A theory explaining *how declarations are used to reach a goal* is a Speech Act Theory

Message Semantics: Speech Acts (3)

- “This is the Google site”

- This is an statement (TRUE or FALSE)

  - *I suggest that you use the Google site.*
  - *I command that you use the Google site.*
  - *I request that you use the Google site.*
  - *I ask that you tell me if you are using the Google site.*
  - *I inform you that I am using the Google site.*

- These are not TRUE/FALSE statements, these suggest actions

Message Semantics: Speech Acts (4)

- 3 aspects in a ***Speech Act***
  - *Locutionary act* or *locution:* what it is said or written (the sentence, the sounds
    - **Use the Expedia Site**
  - *Illocutionary act* or *illocution:* what it is not said or written explicitly, but it is meant.
    - suggest? request? commit?
    - Note: **ilocutionary force is applied to a content**
  - *Perlocutionary act* or *perlocution:* the effect provoked on those who hear a meaningful utterance
    - *e..g.* People ordering flights and hotels through the Expedia site
    - The perlocutonary force is always related to the intentions
      - *e.g.* To earn money from people's orders.

Message Semantics: Speech Acts (5)

Illocutionary speech acts (Searle, 1975)

- ***assertive*** = speech acts that *commit* a speaker to the truth of the expressed proposition
- ***directives*** = speech acts that are to cause the hearer to take a particular action, *e.g.* requests, commands and advice.
- ***commissives*** = speech acts that commit a speaker to some future action, *e.g.* promises and oaths.
- ***expressive*** = speech acts that express on the speaker's attitudes and emotions towards the proposition, *e.g.* congratulations, excuses and thanks
- ***declarations*** = speech acts that change the reality in accord with the proposition of the declaration, *e.g.* pronouncing someone guilty or pronouncing someone husband and wife.

Message Semantics: Speech Acts (6)

As a summary:

- An ***agent performs an ilocutionary act***
  - An act which carries an intention
- To ***achieve a perlocutionary effect***
  - To get some action made or a change in the world state
- But perlocutionary effects are out of control from this agent
  - The actual effect may be different than intended***.***

Message Semantics: Speech Acts (7)

- A speech act is composed by the *performative verb* and the *propositional content*
- E.g.:
  - performative = request
    content = *"the door is closed"*
    speech act = *"please close the door"*
  - performative = inform
    content = *"the door is closed"*
    speech act = *"the door is closed!"*
  - performative = inquire
    content = *"the door is closed"*
    speech act = *"is the door closed?"*

Message Semantics: Speech Acts (8)

- Formal semantics for all performatives has been defined.

- The only task left is to define when an interaction is successful (as this is domain-dependent).

- *e.g.* given a set of illocutions
  - ***(request agent1 agent2)***
  - ***(inform agent1 agent2)***
  - ***(ask agent1 agent2)***

- Specify the success conditions for each illocution
  - What are the necessary and sufficient conditions that should hold so $agent_1$ can consider its request to $agent_2$ to be successful?

Plan Based Semantics: Speech Acts (9)

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

Plan Based Semantics: (10)

- Here is their semantics for *request*:

request(s, h, ϕ)

pre:

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

post:

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

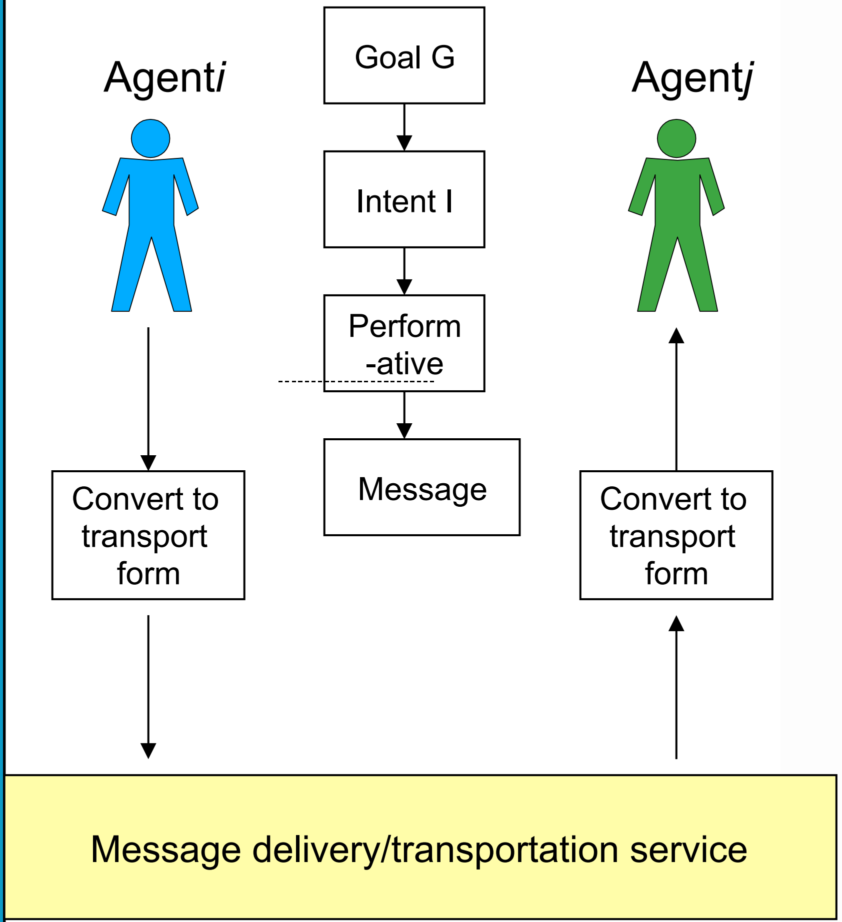
Message Sintaxis: Communication Languages

- *Procedural Approach*
  - Exchange of procedural information
  - These are simple and efficient languages
- *Declarative Approach*
  - Exchange of declarative information
  - Problem of expresiveness

Message Sintaxis: Agent Communication Languages

- Agent communication is based in Speech Act Theory
- Agents use a set of pre-defined performatives in order to communicate their intentions
- The performative semantics allow the agent receiving a message to interpret its content in a proper way
- There are two pre-defined performative sets used in Multiagent Systems:
  - **KQML** Knowledge Query and Manipulation Language
  - **FIPA-ACL** Agent Communication Language

Agent Communication Language (ACL)



ACLs allow agents to effectively communicate and exchange knowledge with other agents.

Three Important Aspects

Syntax	1.How the symbols of communication are structured.
Semantics	2. What the symbols denote.
Pragmatics	3. How the symbols are interpreted.

*(**Meaning** is a combination of semantics and pragmatics.)*

Communication Levels

Semantics

Meaning of the information

Syntax

Format of information being transferred

Communication

Method of interconnection

Requirements for an ACL

Syntactic

Semantic

Communication

- **Syntactic translation** between languages
- **Semantic content preservation** among applications
  - The concept must have a uniform meaning across applications.
- **Ability to communicate complex attitudes** about their information and knowledge.
  - Agents need to question, request, *etc*.
  - Not about transporting bits and bytes.

Origins of ACLs

- Knowledge Sharing Effort (**KSE**), funded by ARPA
  - Central concept: knowledge sharing requires communication, which in turn requires a *common language*. KSE focused on defining that common language.
- **KQML**: Knowledge Query and Manipulation Language
  - Language for both message formatting and message handling protocols.
- **KIF**: Knowledge Interchange Format
  - Langauge for expressing message content.

Message Sintaxis: KQML

- The first widely-spread ACL was KQML, developed by the ARPA knowledge sharing initiative.
- KQML is comprised of two parts:
  - the knowledge query and manipulation language (KQML)
  - the content language (usually KIF).
- KQML is an ‘outer’ language, that defines a quite large set of acceptable ‘communicative verbs’, or *performatives* for :
  - Basic requests (`evaluate`, `ask-one`, `perform` …)
  - Multiagent requests (`stream-in`, …)
  - Responses (`reply`, `sorry`, …)
  - Information (`tell`, `achieve`, `cancel`, …)
  - Coordination (`stand-by`, `ready`, `next`, …)
  - Definition of capabilities (`advertise`, `subscribe`, …)
  - Networking (`register`, `forward`, `broadcast`, …)

Origins of ACLs

- Knowledge Sharing Effort (**KSE**), funded by ARPA
  - Central concept: knowledge sharing requires communication, which in turn requires a *common language*. KSE focused on defining that common language.
- **KQML**: Knowledge Query and Manipulation Language
  - Language for both message formatting and message handling protocols.
- **KIF**: Knowledge Interchange Format
  - Langauge for expressing message content.

KIF 1

- Motivation: creation of a common language

  for expressing properties of a domain.
  - Intended to express contents of a message; **not the message itself**.
  - Based on First-Order Logic (FOL).

KIF 2

- Using KIF, it is possible to express:
  - Properties of things in a domain
    - *e.g. Michael is a vegetarian* – Michael has the property of being a vegetarian
  - Relationships between things in a domain
    - *e.g. Michael and Janine are married* – the relationship of marriage exists between Michael and Janine.
  - General properties of a domain
    - *e.g. Everybody has a mother*.

KIF 3 - Example

- Relation between 2 objects:
  - *The temperature of m1 is 83 Celsius:*

```
(=  (temperature m1) (scalar 83 Celsius))
```

- Definition of new concept:
  - *An object is a bachelor if this object is a man and not married:*

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

- Relationship between individuals in the domain:
  - *A person with the property of being a person also has the property of being a mammal:*

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

Message Sintaxis: KQML

Example

```
( ask-one
  :sender joan
  :receiver stock-server
  :reply-with IPOD-stock
  :content (PRICE IPOD ?price)
  :language LISP
  :ontology NYSE-TICKS )
```

Performative

Communication parameters

Message Content

Content Language specification

Ontology specification

Message Sintaxis: KQML

Example

```
( ask-one
  :sender joan
  :receiver stock-server
  :reply-with IPOD-stock
  :content (PRICE IPOD ?price)
  :language LISP
  :ontology NYSE-TICKS )
```

Message Layer

Communication Layer

Content Layer

Message Sintaxis: Message layers

Content Layer: formatting information

It communicates the ***content*** expressed in a ***language*** according to an ***ontology***

Typical languages include ***KIF, LISP, Prolog, FIPA-SL***

Message Layer: communication scenario

It ***tells*** the message recipient what to do with the message, which actions are implied

Transport: extra information

It contains information about the data transport. It includes the message ***sender*** and ***receiver***, and references to other messages in the dialogue (***reply-with*** and ***in-reply-to***).

Message Sintaxis: FIPA-ACL

- More recently, the Foundation for Intelligent Physical Agents (FIPA) started work on a program of agent standards — the centrepiece is an ACL

- Basic structure is quite similar to KQML:
  - *Type of communicative act: performative*
    22 performatives in FIPA (reduction from KQML)
  - *communication actors*
    e.g., sender, receiver.
  - *content*
    the actual content of the message
  - *Content description*
    e.g., language, encoding, ontology
  - *Conversation control*
    e.g., `protocol, conversation-id, reply-with, in-reply-to, reply-by`

Message Sintaxis: FIPA-ACL

- Example:

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

Message Sintaxis: FIPA-ACL

performatives

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

Message Sintaxis: FIPA-ACL

Content Language

- Almost any content language can be used with FIPA-ACL. Most used are KIF (ANSI-KIF, ISO-KIF), RDF, DAML, OWL and FIPA-SL
- Others can be used such as PROLOG, SQL, …
- **FIPA-SL** (Semantic Language)
  - Allows representation of asserts in modal
  - It is designed for agents with BDI architecture (Beliefs, Desires, Intentions)
  - Defines 3 types of content:
    - **Statements**: expressions which can be associated with a truth value
    - **Actions**: expressions defining an action that can be performed
    - **Reference expressions**: quantified formulae referring to domain objects which comply with that formulae

Message Sintaxis: FIPA-SL

Elements

- Expressions in FIPA-SL are in prefix notation (such as in KIF)
- It includes connectives from First Order Logic
  - `not, and, or, implies, <=>, forall exist`
- BDI Operators
  - `(B <agent> <exp>)` Agent believes the expression
  - `(U <agent> <exp>)` Agent has some uncertainty about the expression
  - `(I <agent> <exp>)` Agent has as an intention the one in the expression
  - `(PG <agent> <exp>)` Agent has as an objective the one in the expression

Message Sintaxis: FIPA-SL

Elements

- Temporal Logic operators
  - (feasible <action> <exp>): Action can be performed when expression holds
  - (done <action> <exp>): Action was performed before the expression held.
- Relational and list operators
  - (=, >, <, member, contains)
- Reference expressions (evaluated through a Knowledge Base)
  - (iota <terms> <exp>): refers to the unique object which, instantiating the terms, makes the expressions true
  - (any <terms> <exp>): refers to a/some objects which, instantiating the terms, make the expressions true
  - (all <terms> <exp>): refers to all objects which, instantiating the terms, make the expressions true

Message Sintaxis: FIPA-SL

Elements

- Functional Terms (predicates): expressions which refer to an object through its functional relation with other objects (e.g., 3 = (+ 2 1)). There are two alternative expressions:
  - (<predicate> <value$_1$> … <value$_n$>),
    e.g. (person "Juan" 23)
  - (<predicate <prop$_1$> <value$_1$> … <prop$_n$> <value$_n$>)
    e.g., (person :name "Juan" :age 23)
- FIPASL has some pre-defined functional terms (arithmetic operators, set operators, list operators…)
- Predicates over actions and results
  - (action <agent> <exp>): we request the agent to perform the action expressed in the expression
  - (result <action> <exp>): informs about the result of a given action

Message Sintaxis: FIPA-SL

3 subsets

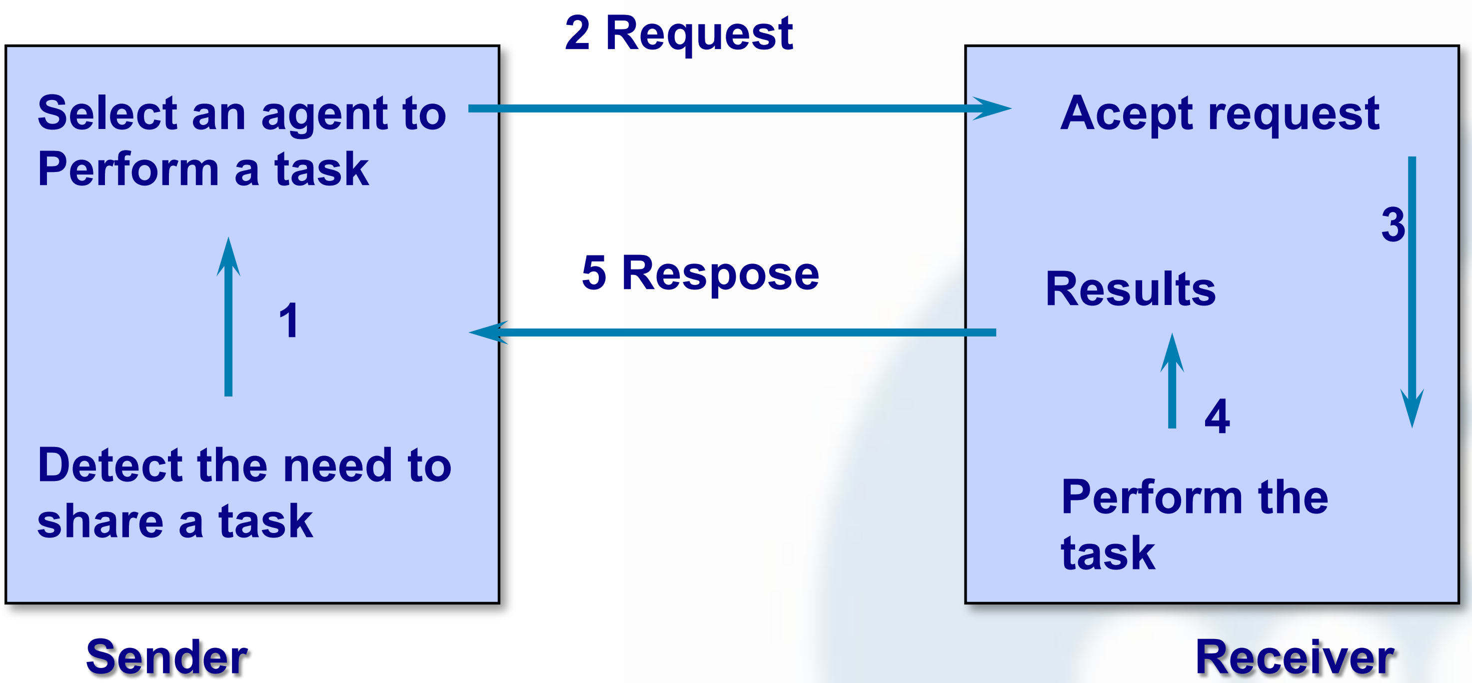
- FIPA-SL defines 3 subsets of the language with different expressiveness, for computational reasons
  - **FIPA-SL0**: Allows predicates action, result, done, simple propositions, sets and sequences
  - **FIPA-SL1**: Adds boolean connectives in expressions
  - **FIPA-SL2**: Adds referential expressions and the modal/ temporal operators, but with some restrictions to ensure that the demonstrations are decidable

Interaction protocol

Types of dialogues between agents

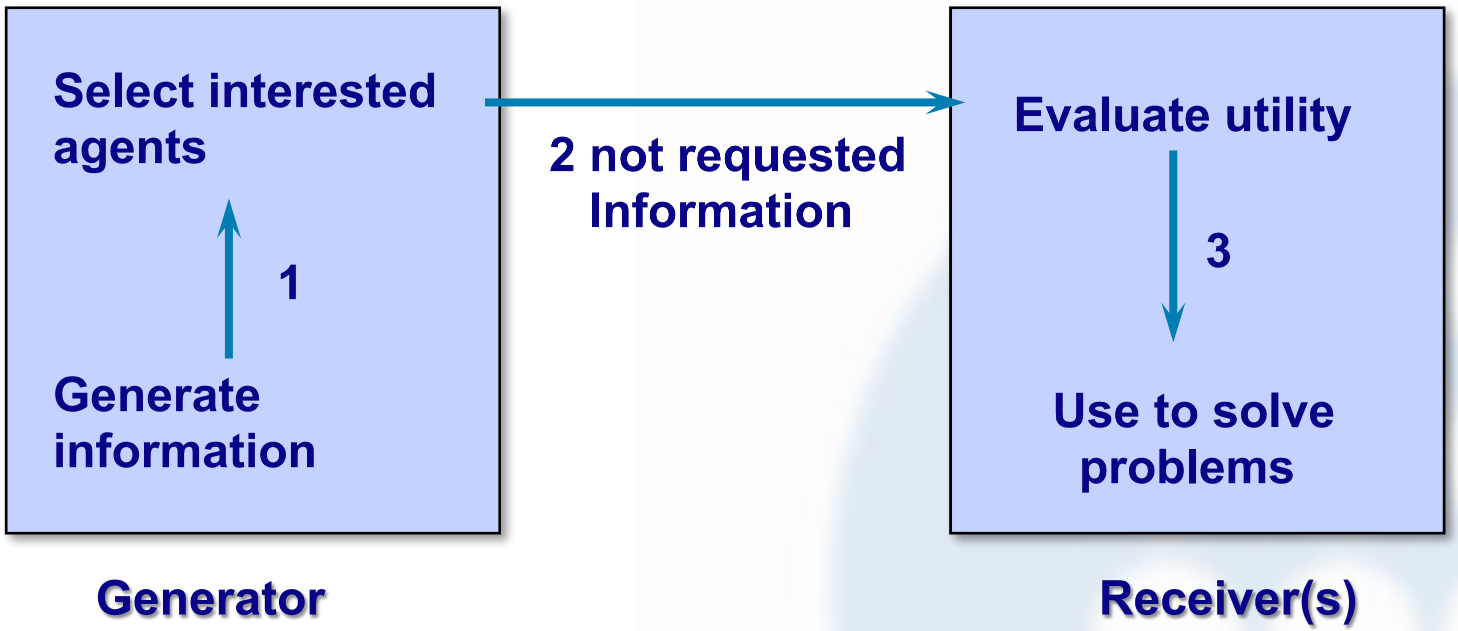
- Mínimal
  - An agent sends and receives information
    - Pasive
  - Apart from sending and receiving information, is capable of requesting information
    - In an active way
    - In an delliberative way
- General
  - Resource management/allocation, information exchange, plan generation, cooperation and negotiation

Interaction protocol
Communication and cooperation



Task sharing

Interaction protocol
Communication and cooperation



Knowledge sharing

Interaction protocol

When to communicate?

If exists a new task t to be done and
an instance of t is still running and
the conditions for execution are the same
then do not start a new task.

If an $Agent_i$ has a task t to be done and
it cannot do it locally
then search for help from another $Agent_j$

If an $Agent_k$ has generated a piece of information and
it believes k might be useful for $Agent_n$
then send k to $Agent_n$

Interaction protocol

What are (agent) communication protocols?

- Performatives cannot work alone, but they appear as part of a *protocol specification*
- A **protocol** is a conversation between agents which follows some rules defining *which performatives to use and when* in order to achieve a given goal

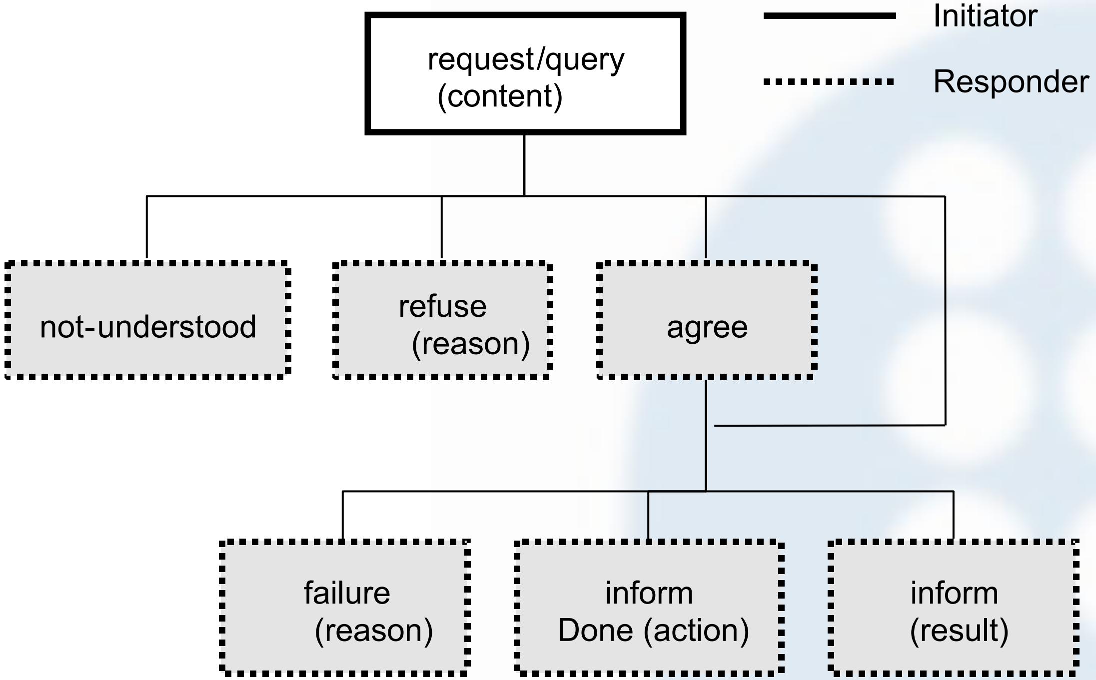
- Each protocol defines the sequencing of messages in a given dialogue as a finite-state diagram
- Advantage: agents can easily keep the current state of a dialogue and know which utterances follow in order to comply with the protocol
- Each protocol is designed for a specific type of dialogue → One should carefully choose which protocol to use for each situation.

Interaction protocol
Protocols defined by FIPA

- They have two sides: *initiator* and *responder*.
- FIPA protocols: *Request, Query, Contract Net, Iterated Contract Net, Brokering, Recruiting, Subscribe, Propose*
- The most used are::
  - *Request*: dialogue to ask an agent for an action to be performed. The responder agent gives back the result, if possible
  - *Request-When*: dialogue to ask an agent for an action to be performed whenever some conditions hold
  - *Query*: dialogue to ask an agent if a given expression is true. The responder agent answers, if possible
  - *Propose*: dialogue to propose another agent to perform a given action under given conditions. The responder agent accepts or rejects the proposal
  - *Contract Net*: dialogue to request a group of agents to send back proposals for actions to solve a given task. The initiator agent selects the best proposals

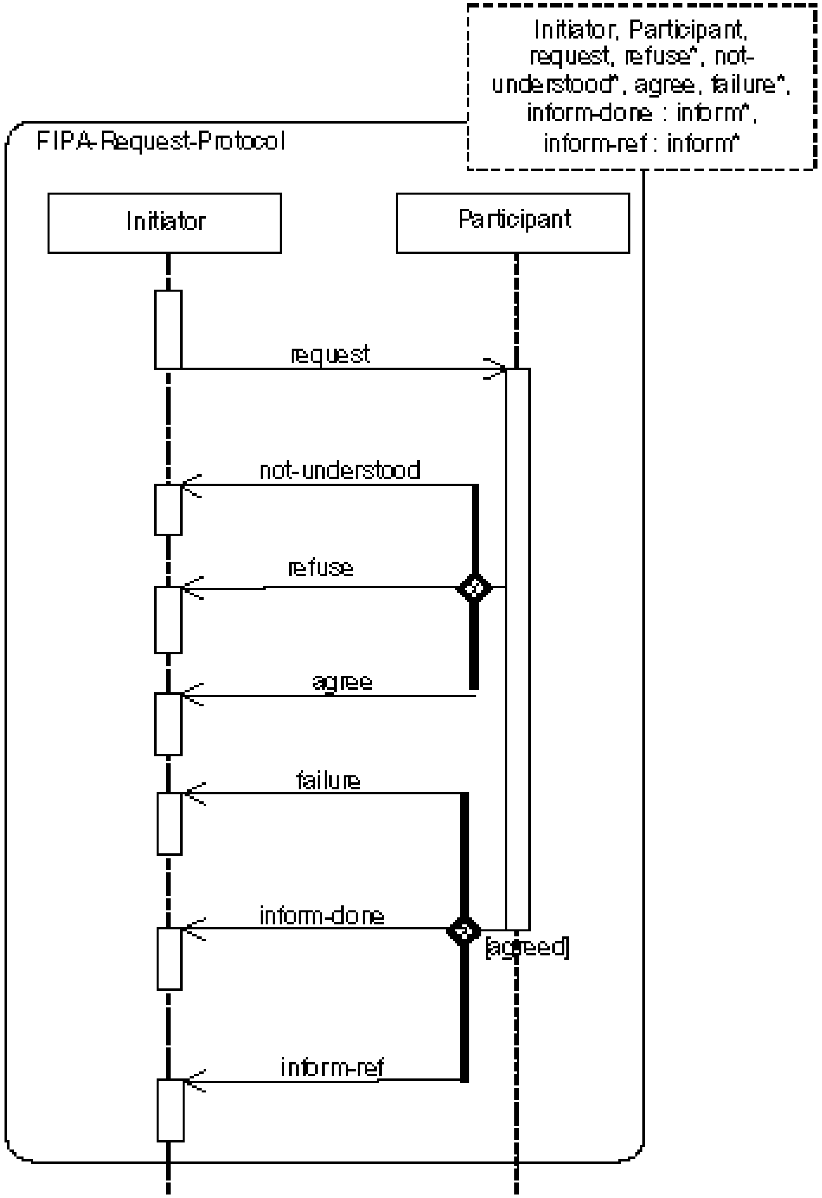
FIPA protocols
Request-Response Protocols

- E.g. FIPA specification for *FIPA-Query* and *FIPA-Request*

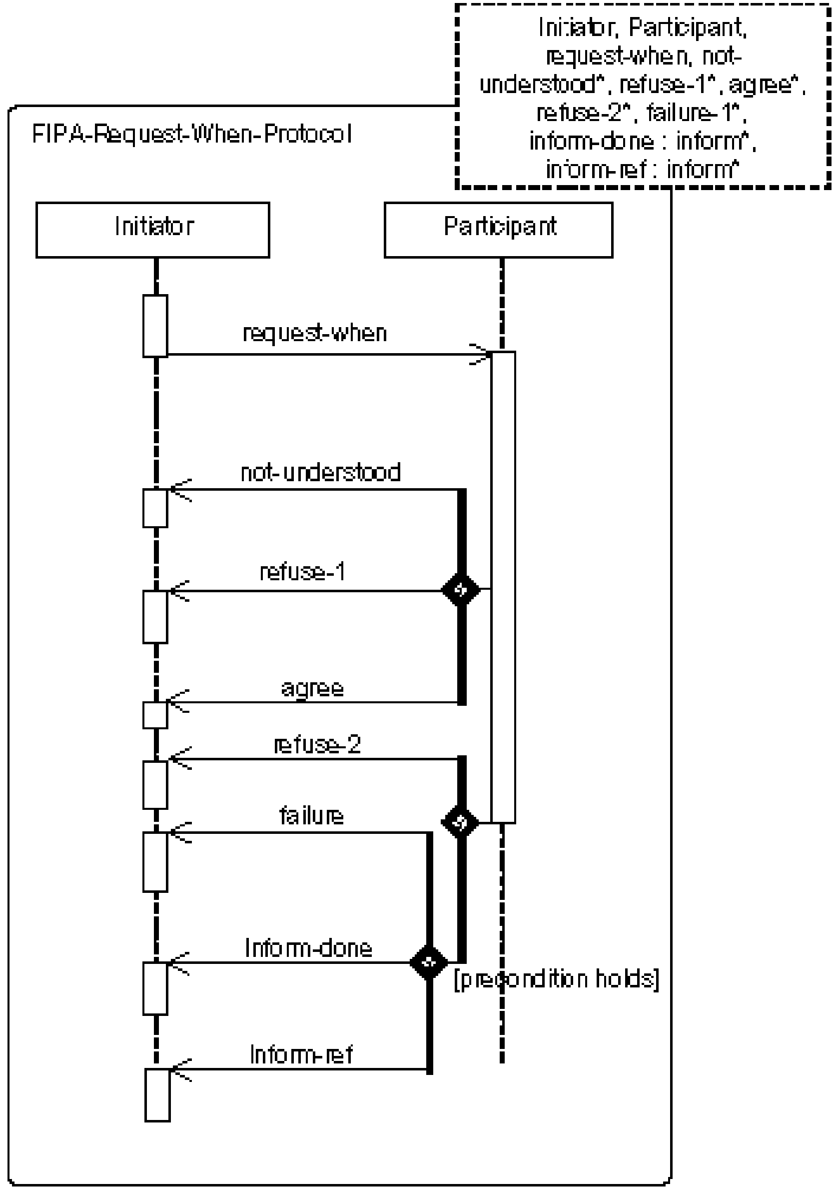


FIPA protocols

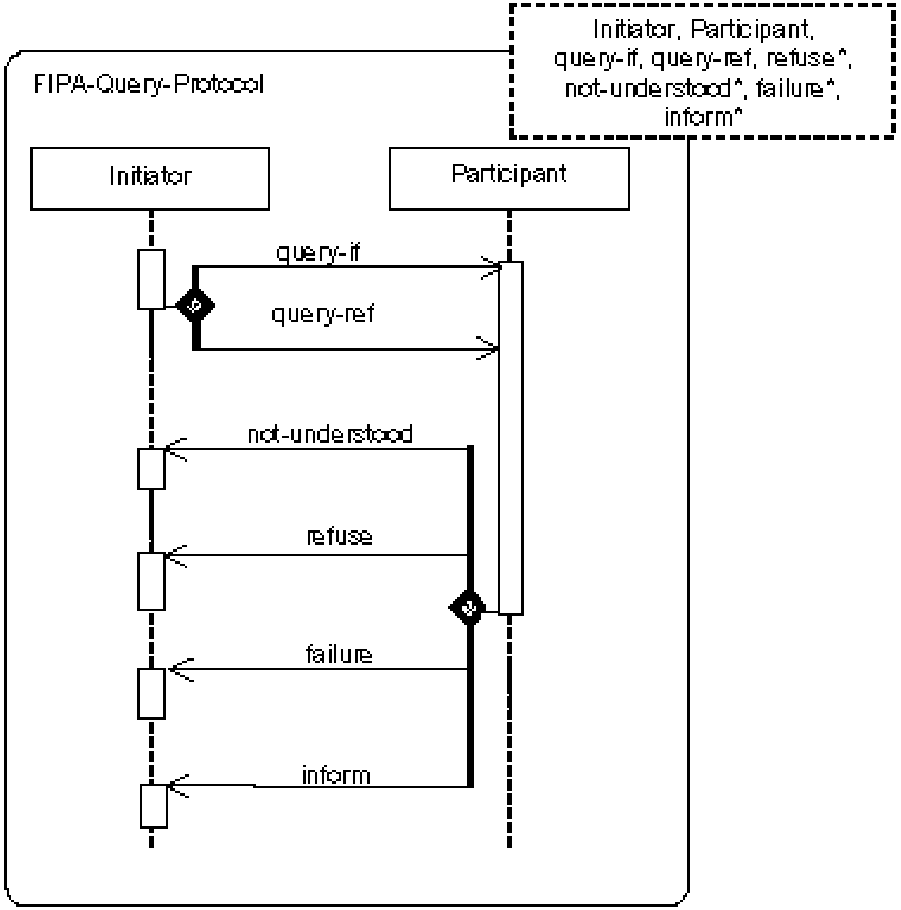
FIPA-Request



FIPA protocols
FIPA-Request-When



FIPA protocols
FIPA-Query

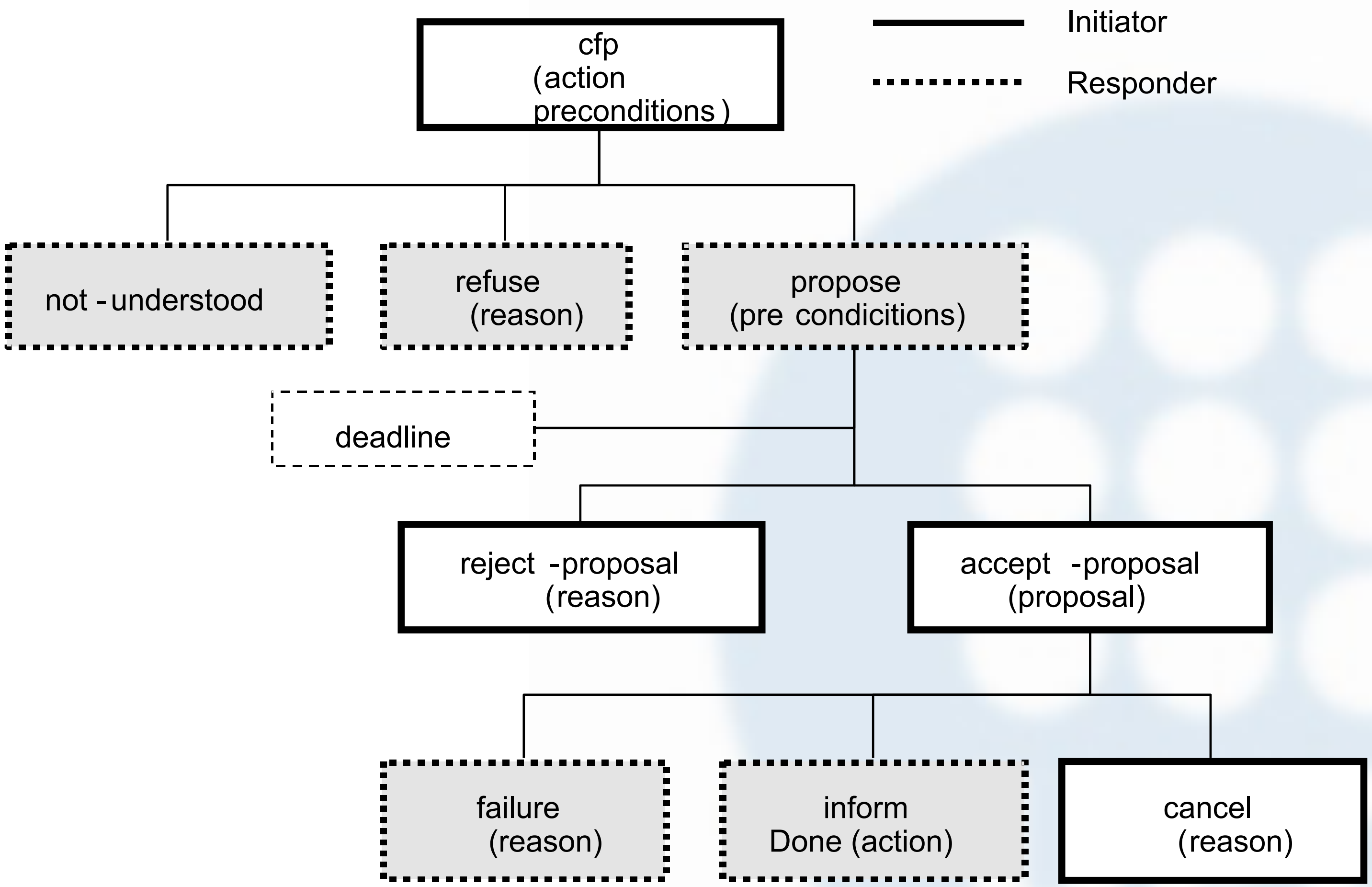


ia@cs.upc.edu

FIPA protocols

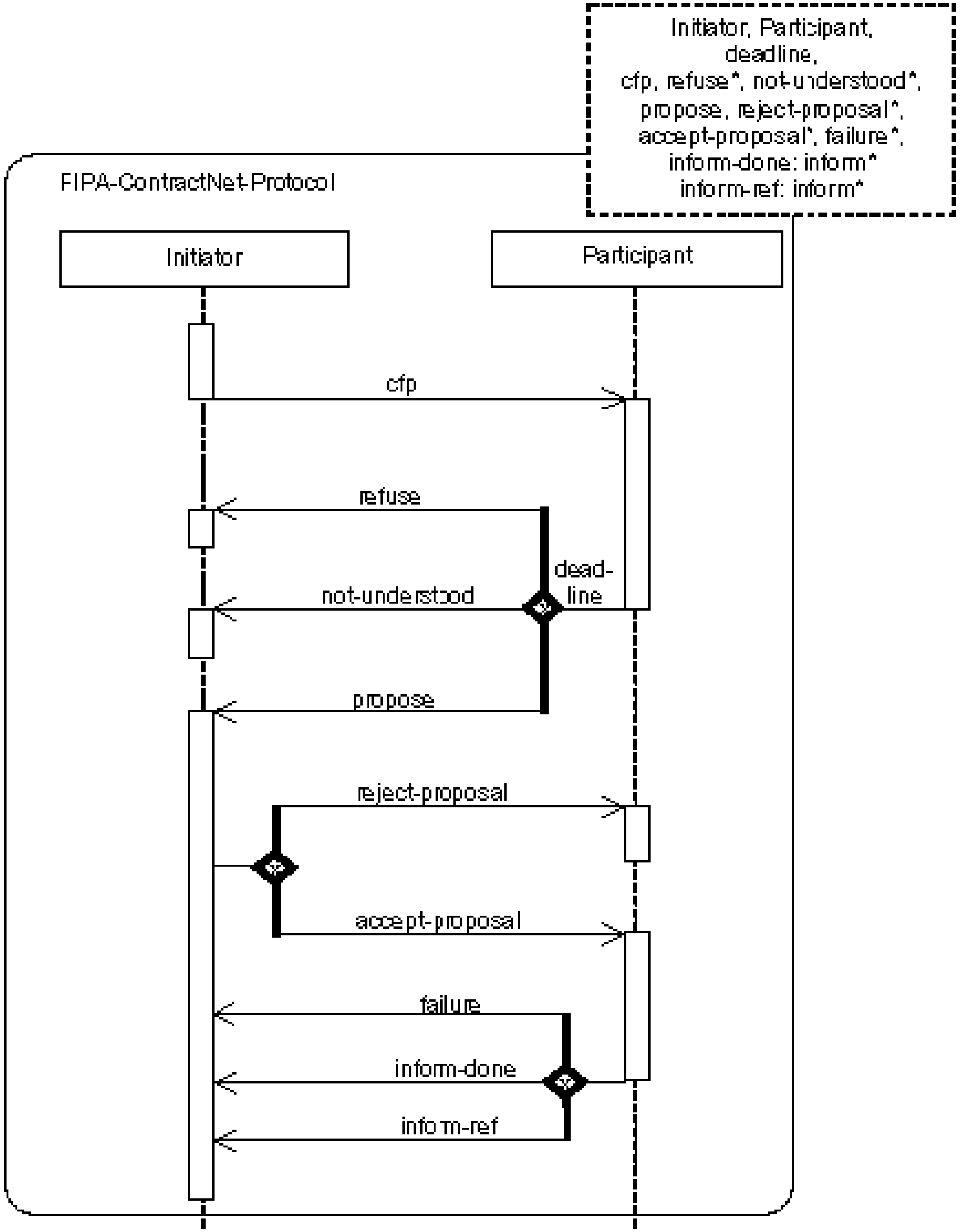
FIPA-Contract-Net (I)

- E.g. FIPA specification for *Contract Net*

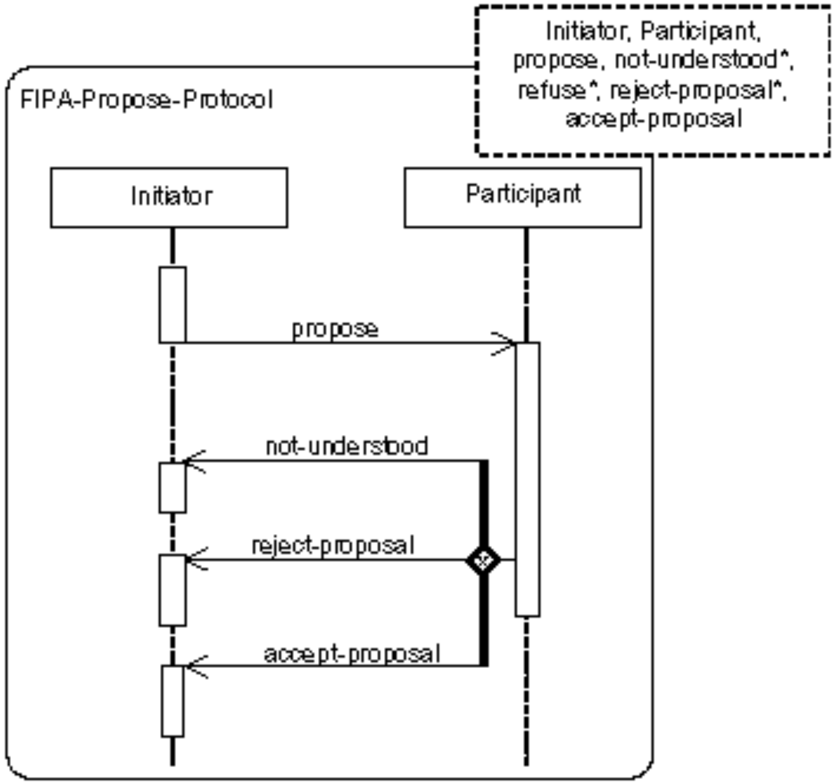


FIPA protocols

FIPA-Contract-Net (II)



FIPA protocols
FIPA-Propose



Levels in Agent Communication (summary)

- Four levels in communication:
  - **Message Semantics**
    - What does each message means?
    - 3 components
      - **Message type**: gives intensionality
      - **Message content**: contains the information
      - **Ontology** (the message refers to)
  - **Message Sintaxis**
    - How each message is expressed?
    - 2 components
      - Message structure: **Agent Communication Language**
      - Content codification: **Content Language**
  - **Interaction protocol**
    - How are conversations/dialogues structured?
      - **Agent Protocols**
  - **Transport protocol**
    - How messages are actually sent and received by agents?

DISTRIBUTED INTELLIGENT SYSTEMS

References

[1] Luck, M., McBurney, P., Shehory, Onn, Willmott, S. "Agent Technology: Computing as interaction. A Roadmap to Agent Based Computing". Agentlink, 2005. ISBN 085432 845 9

[2] Wooldridge, M. "Introduction to Multiagent Systems". John Wiley and Sons, 2002.

[3] FIPA Agent Communication specifications. http://www.fipa.org/repository/aclspecs.html

[4] Haddadi, A. "Communication and Cooperation in Agent Systems: A Pragmatic Theory" Lecture Notes in Artificial Intelligence #1056. Springer-Verlag. 1996. ISBN 3-540-61044-8

[5] Weiss, G. "Multiagent Systems: A modern Approach to Distributed Artificial Intelligence". MIT Press. 1999. ISBN 0262-23203

[6] Rosenschein, J. & Zlotkin, G. "Rules of Encounter. Designing Conventions for Automated Negotiation among Computers". MIT Press. 1994 ISBN 0-262-18159-2

These slides are based mainly in material from [2], [3] and from J. Bejar, with some additions from material by A. Moreno