



# Inteligența Artificială

---

**Universitatea Politehnica București**  
**Anul universitar 2013-2014**

Adina Magda Florea



Curs nr. 10

# Agenti intelligenti

# 1. De ce agenti?

- Sisteme complexe, pe scara larga, distribuite
- Sisteme deschise si heterogene – construirea independenta a componentelor
- Distributia resurselor
- Distributia expertizei
- Personalizare
- Interoperabilitatea sistemelor/ integrare sisteme software exsistente (legacy systems)

# Agent?

Termenul **agent** este frecvent utilizat in:

- Sociologie, biologie, psihologie cognitiva, psihologie sociala si
- Stiinta calculatoarelor  $\supset$  IA
- Ce sunt agentii?
- Ce sunt agentii in stiinta calculatoarelor?
- Aduc ceva nou?
- Cum difera agentii software de alte programe?

## 2. Definitii ale agentilor in stiinta calculatoarelor

- Nu exista o definitie unanim acceptata
- De ce este greu de definit?
- IA, agenti inteligenti, sisteme multi-agent
- Aparent agentii sunt dotati cu inteligenta
- Sunt toti agentii inteligenti?
- **Agent** = definit mai mult prin caracteristici, unele pot fi considerate ca manifestari ale unui comportament inteligent

# Definitii agenti

- “De cele mai multe ori, oamenii folosesc termenul agent pentru a referi o entitate care functioneaza **permanent** si **autonom** intr-un mediu in care exista alte procese si/sau alti agenti” (Shoham, 1993)
- “Un agent este o entitate care **percepe** mediul in care se afla si **actioneaza** asupra acestuia” (Russell, 1997)

- “**Agent** = un sistem (software sau hardware) cu următoarele proprietati:
- ↓ **autonomie** – agentii opereaza fara interventia directa a utilizatorului si au un anumit control asupra actiunilor si starilor lor;

*Actiune autonoma flexibila*

- ↓ **reactivitate**: agentii percep mediul si reactioneaza corespunzator al schimbarile din acesta;
- ↓ **pro-activitate**: agentii, pe langa reactia la schimbarile din mediu, sunt capabili sa urmareasca executia scopurilor si sa actioneze independent;
- ↓ **abilitati sociale** – agentii interactioneaza cu alti agenti sau cu utilizatorul pe baza unui limbaj de comunicare.

(Wooldridge and Jennings, 1995)

# 3. Caracteristici agenti

## 2 directii de definitie

- Definirea unui agent izolat
- Definirea agentilor in colectivitate → **dimensiune sociala → SMA**

## 2 tipuri de definitii

- Nu neaparat agenti inteligenti
- Include o comportare tipica IA → **agenti inteligenti**



# Caracteristici agenti

- Actioneaza pentru un utilizator sau un program
- Autonomie
- Percepe mediul si actioneaza asupra lui reactiv
- Actiuni pro-active
- Caracter social
- Functionare continua (persistent software)
- Mobilitate

*inteligenta?*

- Scopuri, rationalitate
- Rationament, luarea deciziilor *cognitiv*
- Invatare/adaptare
- Interactiune cu alti agenti – dimensiune sociala

*Alte moduri de a realiza inteligenta?*

# SMA – mai multi agenti in acelasi mediu

## ■ Interactiuni intre agenti

- nivel inalt

- Interactiuni pentru- coordonare
  - comunicare
  - organizare

## □ Coordonare

➔ motivati colectiv

➔ motivati individual

- scopuri proprii / indiferenta
- scopuri proprii / competitie pentru resurse
- scopuri proprii si contradictorii / competitie pentru resurse
- scopuri proprii / coalitii

## □ Comunicare

➔ protocol

➔ limbaj

- negociere

- ontologii

## □ Structuri organizationale

➔ centralizate vs decentralizate

➔ ierarhie/ piata

abordare *"agent cognitiv"*

# 3.1 Agenti cognitivi

Modelul uman al perspectivei asupra lumii →  
caracterizare agent utilizand reprezentari simbolice si  
*notiuni mentale*

- knowledge - cunostinte
- beliefs - convingeri
- desires, goals – dorinte, scopuri
- intentions - intentii
- commitments - angajamente
- obligations - obligatii

## 3.2 Agenti reactivi

- Unitati simple de prelucrare care percep mediul si reactioneaza la schimbarile din mediu
- Nu folosesc reprezentari simbolice sau rationament.
- Inteligenta nu este situata la nivel individual ci distribuita in sistem, rezulta din interactiunea entitatilor cu mediu – “emergence”

## 3.3 Exemple de probleme tipice

### Problema inteleptilor

Regele picteaza cate o pata alba si spune ca cel putin o pata este alba



### Dilema prizonierului

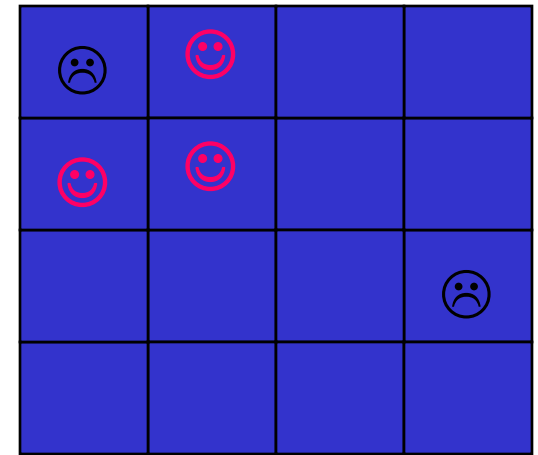
Rezultatele pentru A si B (in puncte ipotetice) in functie de actiunile fiecaruia

<b>Player A / Player B</b>	<i>Tradeaza</i>	<i>Coopereaza</i>
<i>Tradeaza</i>	<b>2</b> , <b>2</b>	<b>5</b> , <b>0</b>
<i>Coopereaza</i>	<b>0</b> , <b>5</b>	<b>3</b> , <b>3</b>

# Problema prazilor si vanatorilor

## - Abordare cognitiva

- vanatorii au scopuri, prazile nu
- Detectia prazilor
- Echipa vanatori, roluri
- Comunicare/cooperare



## Abordare reactiva

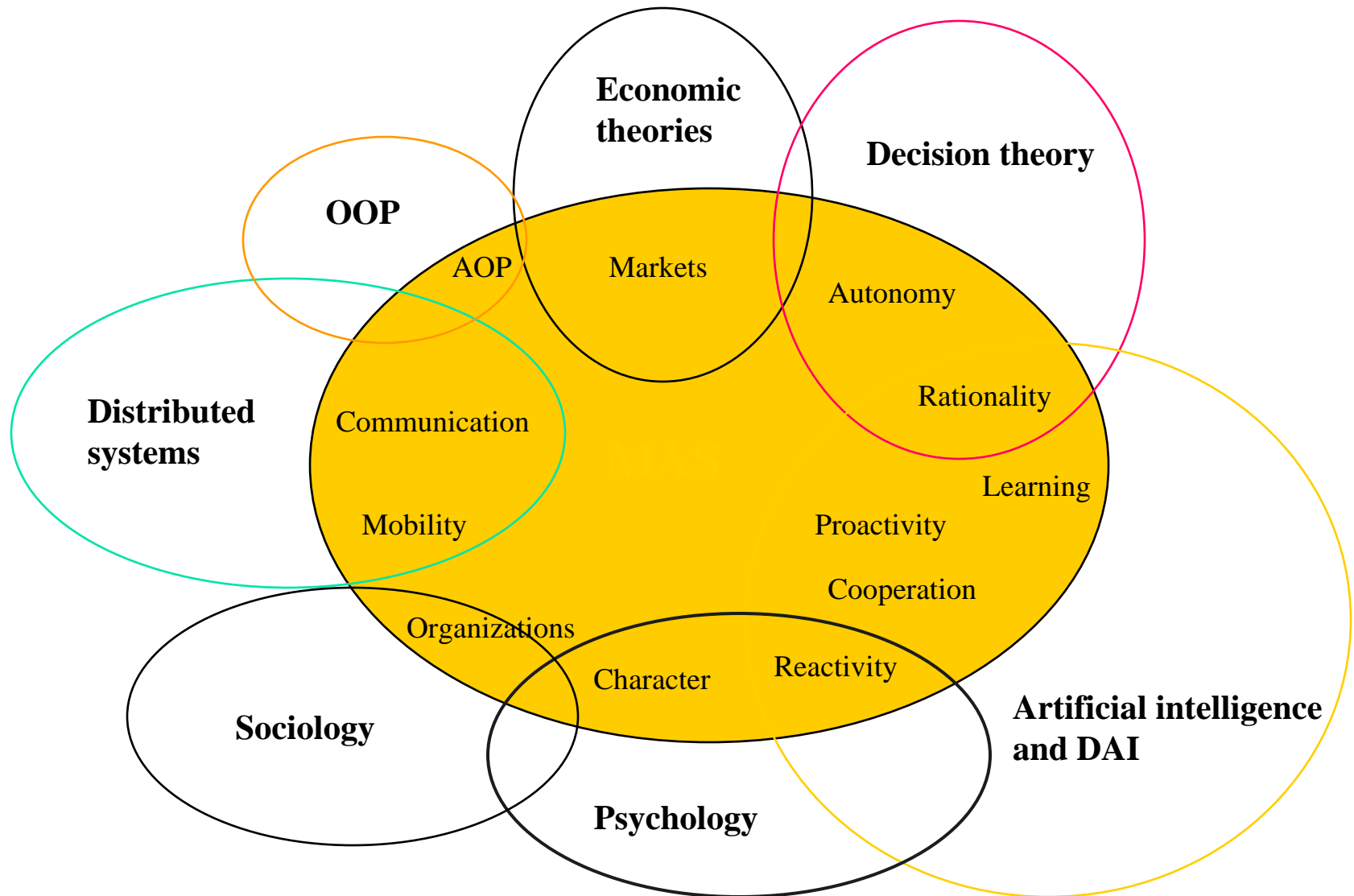
- Prazile emit semnale a caror intensitate scade pe masura cresterii distantei de vanatori
- Vanatorii emit semnale care pot fi percepute de alti vanatori
- Fiecare vanator este atras de o prada si respins de alt semnal de la un vanator

## 3.4 Agenti emotionali

- Inteligenta afectiva
- Actori virtuali
  - recunoasterea vorbirii
  - gesturi, sinteza de vorbire
- Emotii:
  - Aprecierea unei situatii sau a unui eveniment: **bucurie, suparare;**
  - valoarea unei situatii care afecteaza pe alt agent: **bucuros-pentru,, gelos, invidios, suprat-pentru;**
  - Aprecierea unui eveniment viitor: **speranta, frica;**
  - Aprecierea unei situatii care confirma o asteptare: **satisfactie, dezamagire**
- Controlarea emotiilor prin temperament



# Legături cu alte discipline



# 4. Directii de studiu si cercetare

- Arhitecturi agent
- Reprezentare cunostinte: sine, alti agenti, lume
- Comunicare: limbaje, protocol
- Planificare distribuita
- Cautare distribuita, coordonare
- Luarea deciziilor: negociere, pietele de marfuri
- Invatare
- Structuri organizationale
- **Implementare:**
  - Programarea agentilor: paradigme, limbaje
  - Platforme multi-agent
  - Middleware, mobilitate, securitate

# 5. Modele arhitecturale de agenti

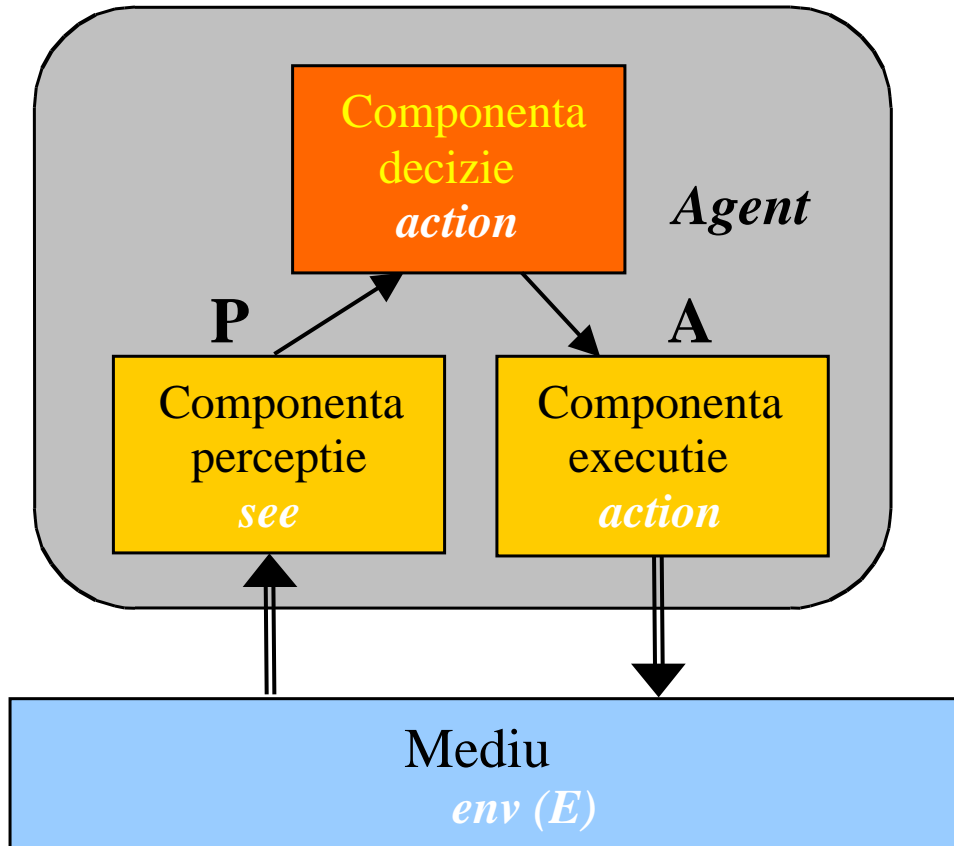
- Structura conceptuala a agentilor
- Arhitecturi de agenti cognitivi
- Arhitecturi de agenti reactivi

# 5.1 Structura conceptuala a agentilor

## 1.1 Rationalitatea unui agent

- Ce inseamna rationalitatea unui agent
- *Cum putem masura rationalitatea unui agent?*
- O masura a performantei
- Arhitectura simpla -> se detaliaza treptat

# Modelare agent reactiv



$$E = \{e_1, \dots, e, \dots\}$$

$$P = \{p_1, \dots, p, \dots\}$$

$$A = \{a_1, \dots, a, \dots\}$$

## Agent reactiv

$$see : E \rightarrow P$$

$$action : P \rightarrow A$$

$$env : E \times A \rightarrow E$$

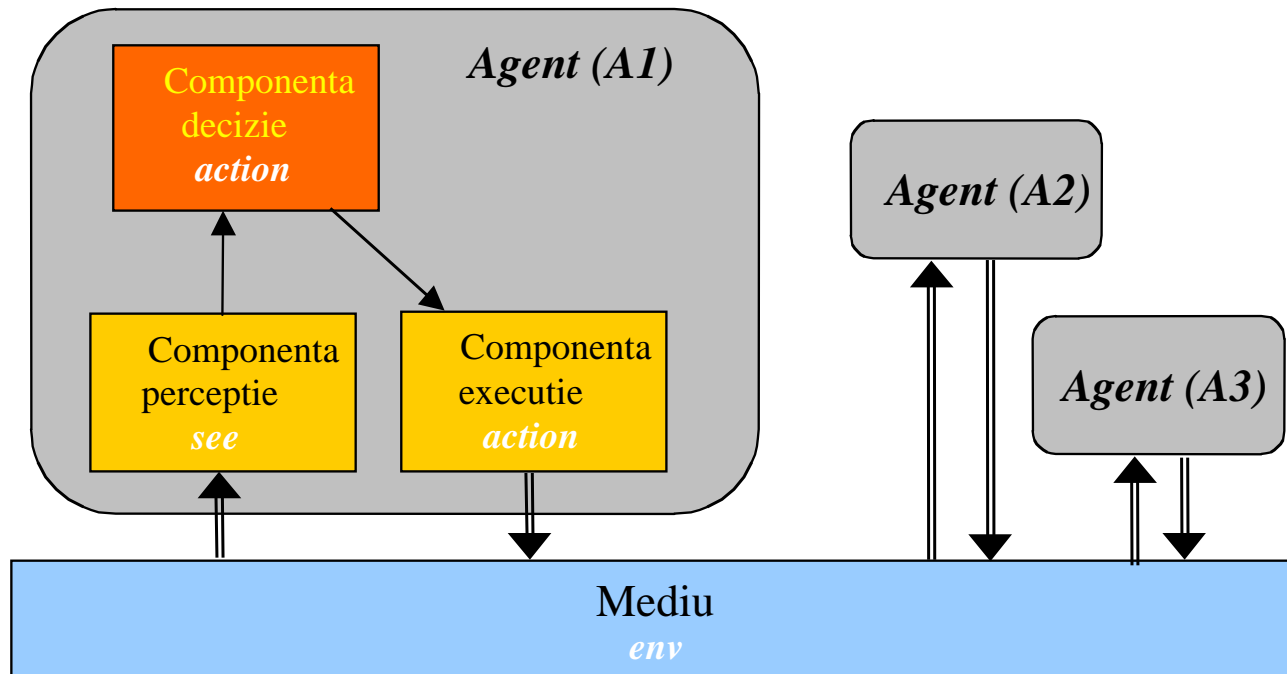
$$(env : E \times A \rightarrow P(E))$$

# Modelare agenti reactivi

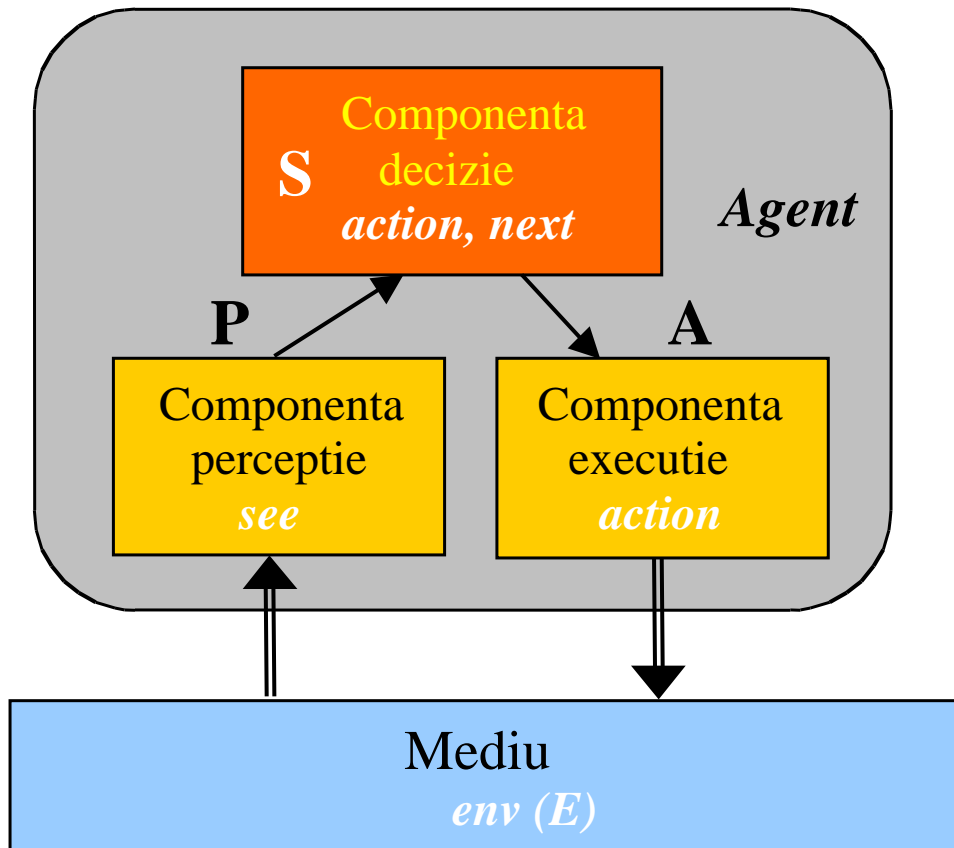
## Mai multi agenti reactivi

$A_1, \dots, A_i, \dots$   
 $P_1, \dots, P_i, \dots$   
(de obicei identice)

$see_i : E \rightarrow P_i$   
 $action_i : P_i \rightarrow A_i$   
 $env : E \times A_1 \times \dots \times A_n \rightarrow P(E)$



# Modelare agent cognitiv



$$E = \{e_1, \dots, e, \dots\}$$

$$P = \{p_1, \dots, p, \dots\}$$

$$A = \{a_1, \dots, a, \dots\}$$

$$S = \{s_1, \dots, s, \dots\}$$

## Agent cu stare

$$see : E \rightarrow P$$

$$next : S \times P \rightarrow S$$

$$action : S \rightarrow A$$

$$env : E \times A \rightarrow P(E)$$

# Modelare agenti cognitivi

$S_1, \dots, S_i, \dots$

$A_1, \dots, A_i, \dots$

$P_1, \dots, P_i, \dots$

(nu intotdeauna identice)

$I = \{i_1, \dots, i_k, \dots\}$

## Mai multi agenti cognitivi

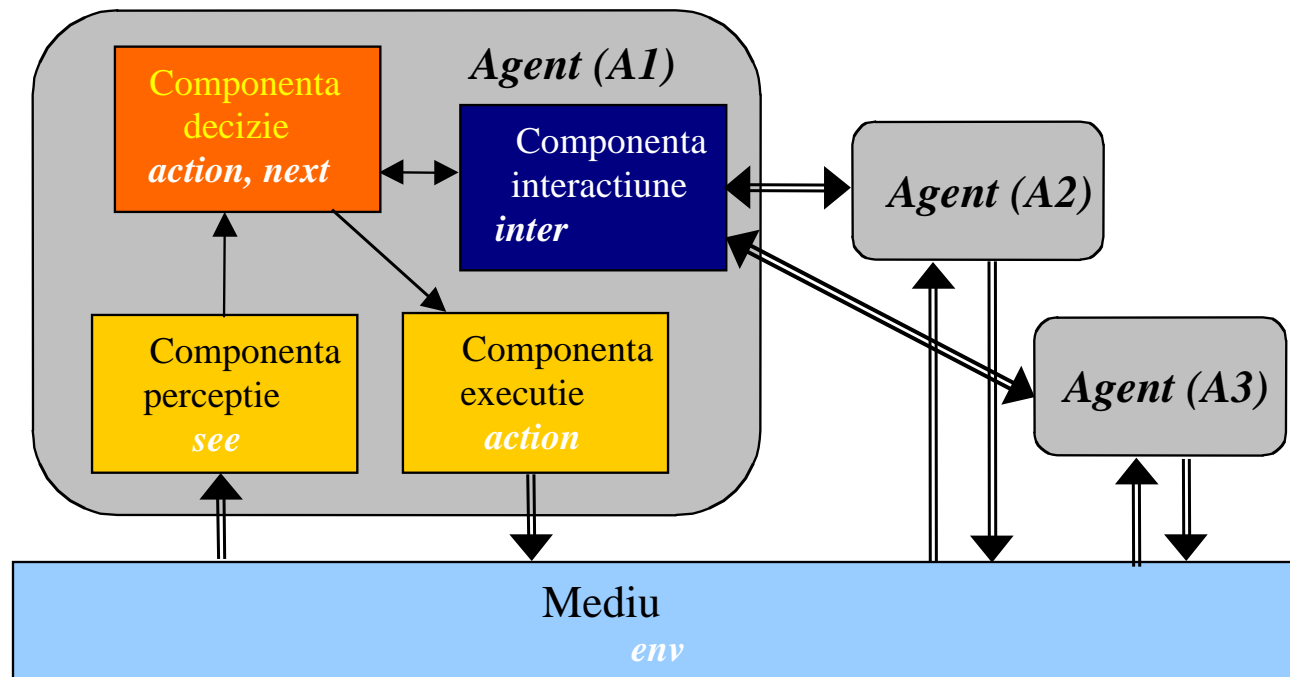
$see_i : E \rightarrow P_i$

$next_i : S_i \times P \rightarrow S_i$

$action_i : S_i \times I \rightarrow A_i$

$inter_i : S_i \rightarrow I$

$env : E \times A_1 \times \dots \times A_n \rightarrow P(E)$





# Modelare agent cognitiv

## Agenti cu stare si scopuri

$$goal : E \rightarrow \{0, 1\}$$

## Agenti cu utilitate

$$utility : E \rightarrow R$$

## Mediu nedeterminist

$$env : E \times A \rightarrow P(E)$$

**Probabilitatea estimata** de un agent ca rezultatul unei actiuni ( $a$ ) executata in  $e$  sa fie noua stare  $e'$

$$\sum_{e' \in env(e, a)} prob(ex(a, e) = e') = 1$$

# Modelare agent cognitiv

## Agenti cu utilitate

**Utilitatea estimata** (*expected utility*) a unei actiuni *a* intr-o stare *e*, dpv al agentului

$$U(a, e) = \sum_{e' \in env(e, a)} prob(ex(a, e) = e') * utility(e')$$

**Principiul utilitatii estimate maxime**

**Maximum Expected Utility (MEU)**

Masura a performantei

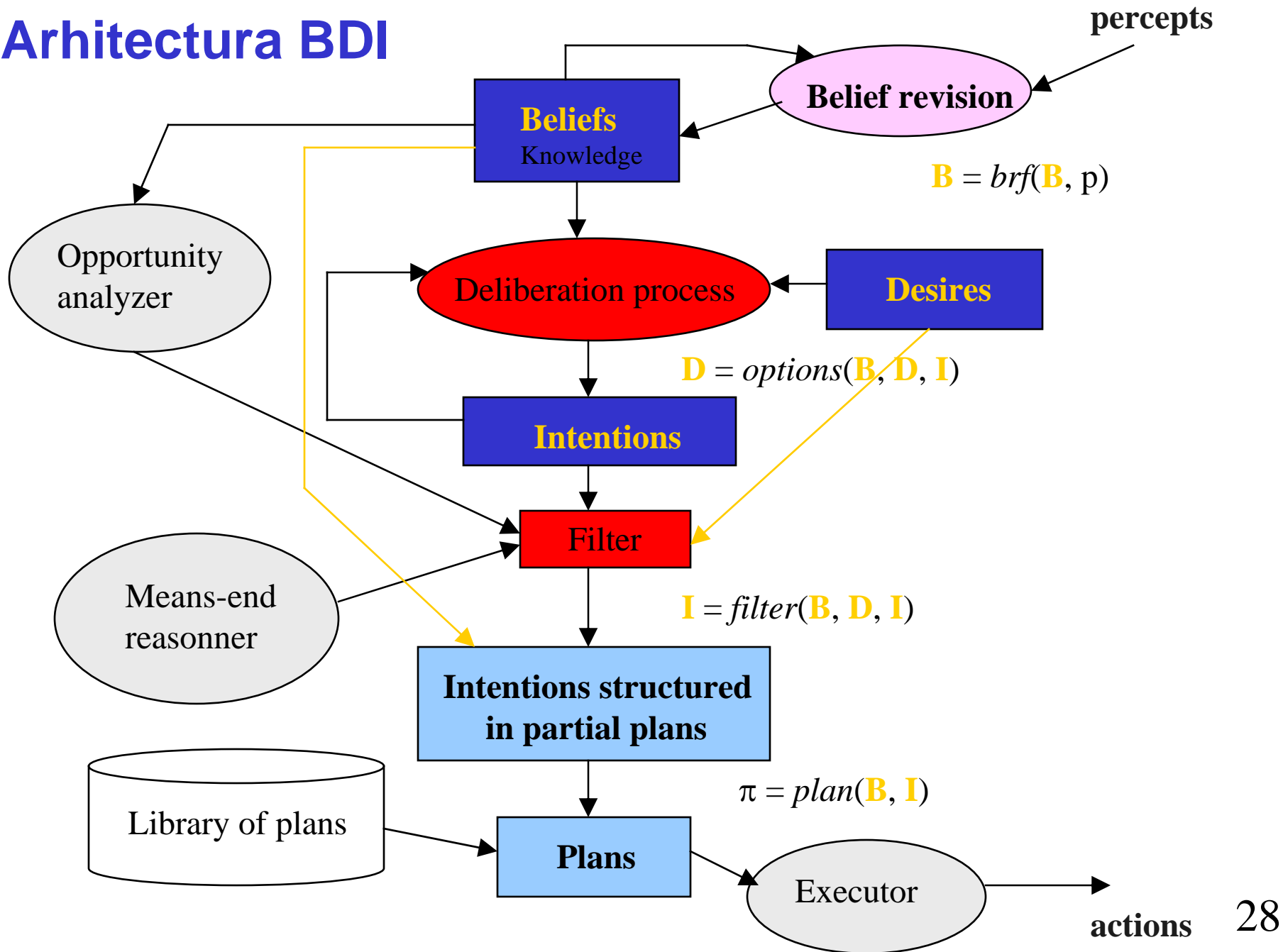


## 5.2 Arhitecturi de agenti cognitivi

### Arhitecturi BDI

- Specificatii de nivel inalt
- Means-end analysis
- **Beliefs (convingeri)** = informatii pe care agentul le are despre lume
- **Desires (dorinte)** = stari pe care agentul ar vrea sa le vada realizate
- **Intentions (intentii)** = dorinte (sau actiuni) pe care agentul s-a angajat sa le indeplineasca

# Architettura BDI



# Bucula de control a agentului

**B** =  $B_0$       **I** =  $I_0$       **D** =  $D_0$

**while** true **do**

    get next percept p

**B** = **brf**(**B**,p)

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

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

$\pi$  = **plan**(**B**, **I**)

    execute( $\pi$ )

**end while**

# Strategii de angajare

## (Commitment strategies)

- Optiune aleasa de agent ca intentie – **agentul s-a angajat pentru acea optiune**
- Persistenta intentiilor

*Interbare:* Cat timp se angajeaza un agent fata de o intentie?

- **Angajare oarba (Blind commitment)**
- **Angajare limitata (Single minded commitment)**
- **Angajare deschisa (Open minded commitment)**

$\mathbf{B} = B_0$

$\mathbf{I} = I_0$   $\mathbf{D} = D_0$

## Bucla de control BDI angajare oarba

**while** true **do**

    get next percept p

$\mathbf{B} = \text{brf}(\mathbf{B}, p)$

$\mathbf{D} = \text{options}(\mathbf{B}, \mathbf{D}, \mathbf{I})$

$\mathbf{I} = \text{filter}(\mathbf{B}, \mathbf{D}, \mathbf{I})$

$\pi = \text{plan}(\mathbf{B}, \mathbf{I})$

**while not** (empty( $\pi$ ) or succeeded ( $\mathbf{I}, \mathbf{B}$ )) **do**

$\alpha = \text{head}(\pi)$

        execute( $\alpha$ )

$\pi = \text{tail}(\pi)$

        get next percept p

$\mathbf{B} = \text{brf}(\mathbf{B}, p)$

**if not** sound( $\pi, \mathbf{I}, \mathbf{B}$ ) **then**

$\pi = \text{plan}(\mathbf{B}, \mathbf{I})$   *Reactivity, replan*

**end while**

**end while**

**B** =  $B_0$

**I** =  $I_0$  **D** =  $D_0$

**while** true **do**

get next percept p

**B** = **brf**(**B**,p)

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

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

$\pi$  = **plan**(**B**, **I**)

**while not** (empty( $\pi$ ) or succeeded (**I**, **B**) or impossible(**I**, **B**)) **do**

$\alpha$  = head( $\pi$ )

execute( $\alpha$ )

$\pi$  = tail( $\pi$ )

get next percept p

**B** = **brf**(**B**,p)

**if not** sound( $\pi$ , **I**, **B**) **then**


$\pi$  = **plan**(**B**, **I**) ← *Reactivity, replan*

**end while**

**end while**

## Bucla de control BDI angajare limitata

*Dropping intentions that are impossible  
or have succeeded*





$\mathbf{B} = \mathbf{B}_0$

$\mathbf{I} = \mathbf{I}_0$   $\mathbf{D} = \mathbf{D}_0$

## Buclo de control BDI

angajare deschisa

**while** true **do**

get next percept p

$\mathbf{B} = \text{brf}(\mathbf{B}, p)$

$\mathbf{D} = \text{options}(\mathbf{B}, \mathbf{D}, \mathbf{I})$

$\mathbf{I} = \text{filter}(\mathbf{B}, \mathbf{D}, \mathbf{I})$

$\pi = \text{plan}(\mathbf{B}, \mathbf{I})$

**while not** (empty( $\pi$ ) or succeeded ( $\mathbf{I}, \mathbf{B}$ ) or impossible( $\mathbf{I}, \mathbf{B}$ )) **do**

$\alpha = \text{head}(\pi)$

execute( $\alpha$ )

$\pi = \text{tail}(\pi)$

get next percept p

$\mathbf{B} = \text{brf}(\mathbf{B}, p)$

**if** reconsider( $\mathbf{I}, \mathbf{B}$ ) **then**

$\mathbf{D} = \text{options}(\mathbf{B}, \mathbf{D}, \mathbf{I})$

$\mathbf{I} = \text{filter}(\mathbf{B}, \mathbf{D}, \mathbf{I})$

$\pi = \text{plan}(\mathbf{B}, \mathbf{I})$

← *Replan*

**end while**

**end while**

- Nu exista o unica arhitectura BDI
- PRS - Procedural Reasoning System (Georgeff)
- dMARS
- UMPRS si JAM – C++
- JACK – Java
- JADE - Java
- JADEX – XML si Java,
- JASON – Java

## 5.3 Arhitecturi de agenti reactivi

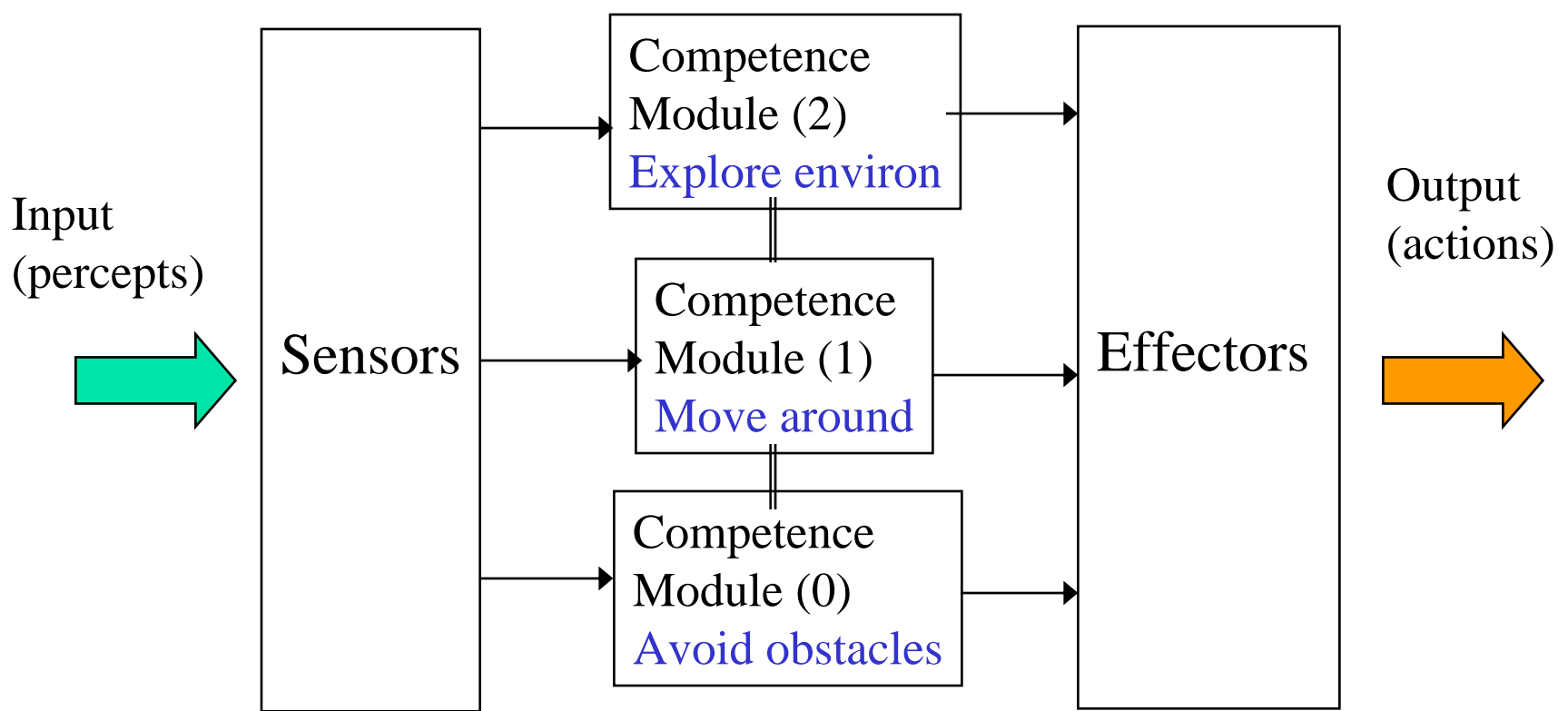
### Arhitectura de subsumare - Brooks, 1986

- (1) Luarea deciziilor = { **Task Accomplishing Behaviours** }
  - Fiecare comportare (behaviour) = o functie ce realizeaza o actiune
  - TAB – automate finite
  - Implementare: *situation* → *action*
- (2) Mai multe comportari pot fi activate in paralel

# Arhitectura de subsumare

- Un TAB este reprezentat de un modul de competenta (c.m.)
- Fiecarte c.m. executa un task simplu
- c.m. opereaza in paralel
- Nivele inferioare au prioritate fata de cele superioare
- c.m. la nivel inferior monitorizeaza si influenteaza intrarile si iesirile c.m. la nivel superior

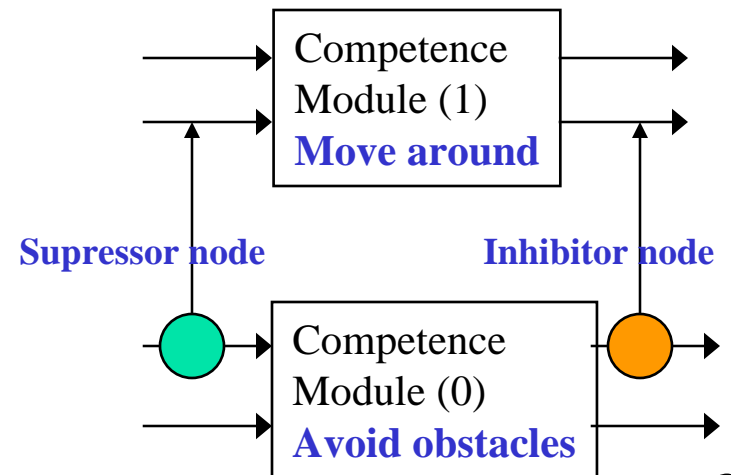
➔ *subsumtion architecture*



M1 = **move around** while *avoiding obstacles*  $\supset$  M0

M2 = **explores the environment looking for distant objects of interests** while *moving around*  $\supset$  M1

- Incorporarea functionalitatii unui c.m. subordonat de catre un c.m. superior se face prin noduri **supresoare** (modifica semnalul de intrare) si noduri **inhibitoare** (inhiba iesirea)



## Comportare

$(c, a)$  – conditie-actiune; descrie comportarea

$\mathbf{R} = \{ (c, a) \mid c \subseteq P, a \in A \}$  - multimea reguli de comportare

$\angle \subseteq R \times R$  – relatie binara totala de inhibare

**function** action(  $p: P$ )

**var** fired:  $P(R)$ , selected:  $A$

**begin**

fired =  $\{ (c, a) \mid (c, a) \in R \text{ and } p \in c \}$

**for each**  $(c, a) \in \text{fired}$  **do**

**if**  $\neg \exists (c', a') \in \text{fired}$  such that  $(c', a') \angle (c, a)$  **then return** a

**return** null

**end**

Ne aflam pe o planeta necunoscuta care contine aur. Mostre de teren trebuie aduse la nava. Nu se stie daca sunt aur sau nu. Exista mai multi agenti autonomi care nu pot comunica intre ei. Nava transmite semnale radio: gradient al campului

## Comportare

- (1) **Daca** detectez obstacol **atunci** schimb directia
- (2) **Daca** am mostre **si** sunt la baza **atunci** depune mostre
- (3) **Daca** am mostre **si** nu sunt la baza **atunci** urmez campul de gradient
- (4) **Daca** gasesc mostre **atunci** le iau
- (5) **Daca** adevarat **atunci** ma misc in mediu

(1)  $\angle$  (2)  $\angle$  (3)  $\angle$  (4)  $\angle$  (5)

Agentii pot comunica indirect:

- Depun si culeg boabe radiocative
- Pot seziza aceste boabe radioactive

- (1) **Daca** detectez obstacol **atunci** schimb directia
- (2) **Daca** am mostre **si** sunt la baza **atunci** depune mostre
- (3) **Daca** am mostre **si** nu sunt la baza **atunci** depun boaba radioactiva **si** urmez campul de gradient
- (4) **Daca** gasesc mostre **atunci** le iau
- (5) **Daca** gasesc boabe radioactive **atunci** iau una **si** urmez campul de gradient
- (6) **Daca** adevarat **atunci** ma misc in mediu

(1)  $\angle$  (2)  $\angle$  (3)  $\angle$  (4)  $\angle$  (5)  $\angle$  (6)





# 6. Negociere

## 6.1 Despre negociere

Agenti motivati colectiv = cooperare

Agenti motivati individual = competitie

Negociere = interactiune -> contract

### ■ **Negocierea include:**

- un limbaj de comunicare
- un protocol de negociere
- un proces de decizie: concesi, criterii de acceptare/refuzare

- Single party **or** multi-party negotiation: one to many or many to many (eBay <http://www.ebay.com> )

### ■ **Tehnici de negociere**

- **Negociere bazata pe teoria jocurilor**
- **Negociere euristica**
- **Negociere bazata pe argumentare**



## 6.2 Negociere bazata pe teoria jocurilor

- Criterii de evaluare protocol negociere
- Agentii se comporta rational
- **Comportare rationala** = un agent prefera o utilitate / plata (*utility* / *payoff*) mai mare fata de una mai mica
- Functia de utilitate
  - $u_i: \Omega \rightarrow \mathbf{R}$
  - $\Omega = \{s_1, s_2, \dots\}$
  - $u_i(s) \geq u_i(s')$  ( $s \geq s'$ ) – ordonarea preferintelor asupra rezultatelor

- Doi agenti au 2 actiuni posibile: **D** si **C** (  $A_c = \{C, D\}$  )
- Mediul se comporta astfel:

$$t: A_c \times A_c \rightarrow \Omega$$

$$t(D,D)=s1 \quad t(D,C)=s2 \quad t(C,D)=s3 \quad t(C,C)=s4$$

sau

$$t(D,D)=s1 \quad t(D,C)=s1 \quad t(C,D)=s1 \quad t(C,C)=s1$$

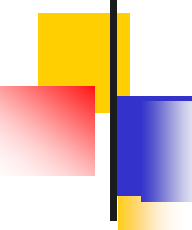
$$u1(s1)=4, \quad u1(s2)=4, \quad u1(s3)=1, \quad u1(s4)=1$$

$$u2(s1)=4, \quad u2(s2)=1, \quad u2(s3)=4, \quad u2(s4)=1$$

$$u1(D,D)=4, \quad u1(D,C)=4, \quad u1(C,D)=1, \quad u1(C,C)=1$$

$$u2(D,D)=4, \quad u2(D,C)=1, \quad u2(C,D)=4, \quad u2(C,C)=1$$

$$\text{Agent1} \quad D,D \geq D,C \geq C,D \geq C,C$$



$u_1(D,D)=4, u_1(D,C)=4, u_1(C,D)=1, u_1(C,C)=1$

$u_2(D,D)=4, u_2(D,C)=1, u_2(C,D)=4, u_2(C,C)=1$

**Agent1**  $D,D \geq D,C \geq C,D \geq C,C$

**Matricea de plata (utilitate)**

		<b>J1</b> <b>Player</b>	
		<b>D</b>	<b>C</b>
<b>J2</b> <b>Player</b>	<b>D</b>	4, 4	4, 1
	<b>C</b>	1, 4	1, 1



# Criterii in negociere

- **Comportare rationala** = utilitate (payoff) mai mare preferata fata de una mai mica
- **Maximizarea platii**: plata individuala, plata de grup, sau bunastare sociala
- **Bunastare sociala**
  - Suma utilitatii agentilor pentru o anumita situatie/solutie
  - Masoara binele general
  - Problema: cum compar utilitatile

# Eficienta Pareto

## ■ Eficienta Pareto

- O solutie  $\mathbf{x}$ , i.e., un **vector de plata**  $\mathbf{p}(\mathbf{x}_1, \dots, \mathbf{x}_n)$ , este **eficient Pareto**, i.e., Pareto optimal, daca nu exista alta solutie  $\mathbf{x}'$  a.i. cel putin un agent are o utilitate mai mare in  $\mathbf{x}'$  decat in  $\mathbf{x}$  si nici un agent nu are o utilitate mai mica in  $\mathbf{x}'$  decat in  $\mathbf{x}$ .
- Masoara bunastarea globala darnu necesita compararea utilitatilor
- Bunastarea sociala  $\subset$  eficienta Pareto

## ■ Rationalitate individuala (IR)

- **IR a participarii unui agent** = Plata agentului in urma participarii la negociere nu este mai mica decat plata lui daca nu ar participa la negociere
- **O negociere este IR** daca este IR pentru toti agentii

# Strategie dominanta

## ■ Stabilitate

- un protocol este **stabil** daca o data ce agentii au ajuns la o solutie ei nu deviaza de la aceasta
- **Strategie dominanta** = agentul are o utilitate mai mare folosind aceasta strategie indiferent de ce strategii folosesc ceilalti agenti

**t:  $Ac \times Ac \rightarrow \Omega$**

$s = t(Act_A, Act_B)$  rezultatul (starea) actiunilor  $Act_A$  a agentului A si  $Act_B$  a agentului B.

O strategie  $S_1 = \{s_{11}, s_{12}, \dots, s_{1n}\}$  **domina** o alta strategie  $S_2 = \{s_{21}, s_{22}, \dots, s_{2n}\}$  daca orice rezultat  $s \in S_1$  este preferat (este mai bun) oricarui rezultat  $s' \in S_2$ .

# Echilibru Nash

## Echilibru Nash

- Doua strategii,  $S_1$  a agentului A si  $S_2$  a agentului B sunt in **echilibru Nash** :
  - daca agentul A urmeaza  $S_1$  atunci agentul B nu poate obtine un castig mai mare decat acela obtinut daca urmeaza  $S_2$  **si**
  - daca agent B urmeaza  $S_2$  atunci agentul A nu poate obtine un castig mai mare decat acela obtinut daca urmeaza  $S_1$ .
- Multime de strategii  $\{S_1, \dots, S_k\}$  folosite de agentii  $A_1, \dots, A_k$  sunt in **echilibru Nash** daca, penru orice agent  $A_i$ , strategia  $S_i$  este cea mai buna strategie a lui  $A_i$  daca ceilalti agenti folosesc  $\{S_1, S_2, \dots, S_{i-1}, S_{i+1}, \dots, S_k\}$ .

*Probleme:*

- nici un echilibru Nash
- multiple echilibre Nash





# Dilema prizonierului

		<b>J2 player</b>	
		Defect	Cooperate
<b>J1 player</b>	Defect	2, 2	5, 0
	Cooperate	0, 5	3, 3

## Alt exemplu

		<b>J2 player</b>	
		<b>I</b>	<b>A</b>
<b>J1 player</b>	<b>I</b>	2, 1	0, 0
	<b>A</b>	0, 0	1, 2



# Strategii in jocuri repetate

## Turneul Axelrod

### Strategii

- **ALL-D** – D tot timpul
- **RANDOM** – C sau D cu probabilitate egala
- **TIT-FOR-TAT**
  - C in primul tur
  - In turul  $t > 1$  ce a ales oponentul in  $t-1$
- **TESTER**
  - D in primul tur
  - Daca oponentul a ales D atunci TIT-FOR-TAT
  - Altfel joaca 2 tururi C si 1 tur D
- **JOSS**
  - TIT-FOR-TAT – dar cu 10% D



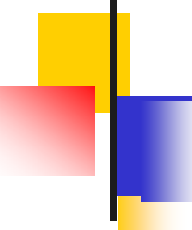
## 6.3 Licitatii

- Protocoale simple
- Centralizate
- Licitatii cu valoare privata
- Licitatii cu valoare comuna
- Licitatii cu valoare corelata



# Protocoloale de licitatii

- **English (first-price open cry) auction** – fiecare participant anunta deschis pretul pe care il liciteaza. Cel mai mare pret castiga
  - Strategie dominanta: putin mai mult decat ultimul pret, ma opresc cand ajung la valoarea privata – in licitatii cu valoare privata
  - In licitatii cu valoare corelata; creste constant pretul pana decizie stop
- **First-price sealed-bid auction** – fiecare participant anunta pretul in plic inchis. Castiga cel cu pret maxim
  - Nu exista strategie dominanta; ofera cel mult pana la valoare lui privata

- 
- **Dutch (descending) auction** - the auctioneer continuously lowers the price until one of the bidders takes the item at the current price.
    - Echivalenta cu licitatia *first-price sealed-bid*
  - **Vickrey (second-price sealed-bid) auction** – trimite oferta in plic inchis. castiga cel care a facut cea mai mare oferta dar plateste al doilea pret
    - Strategia dominanta: valoarea lui privata

Probleme in licitatii

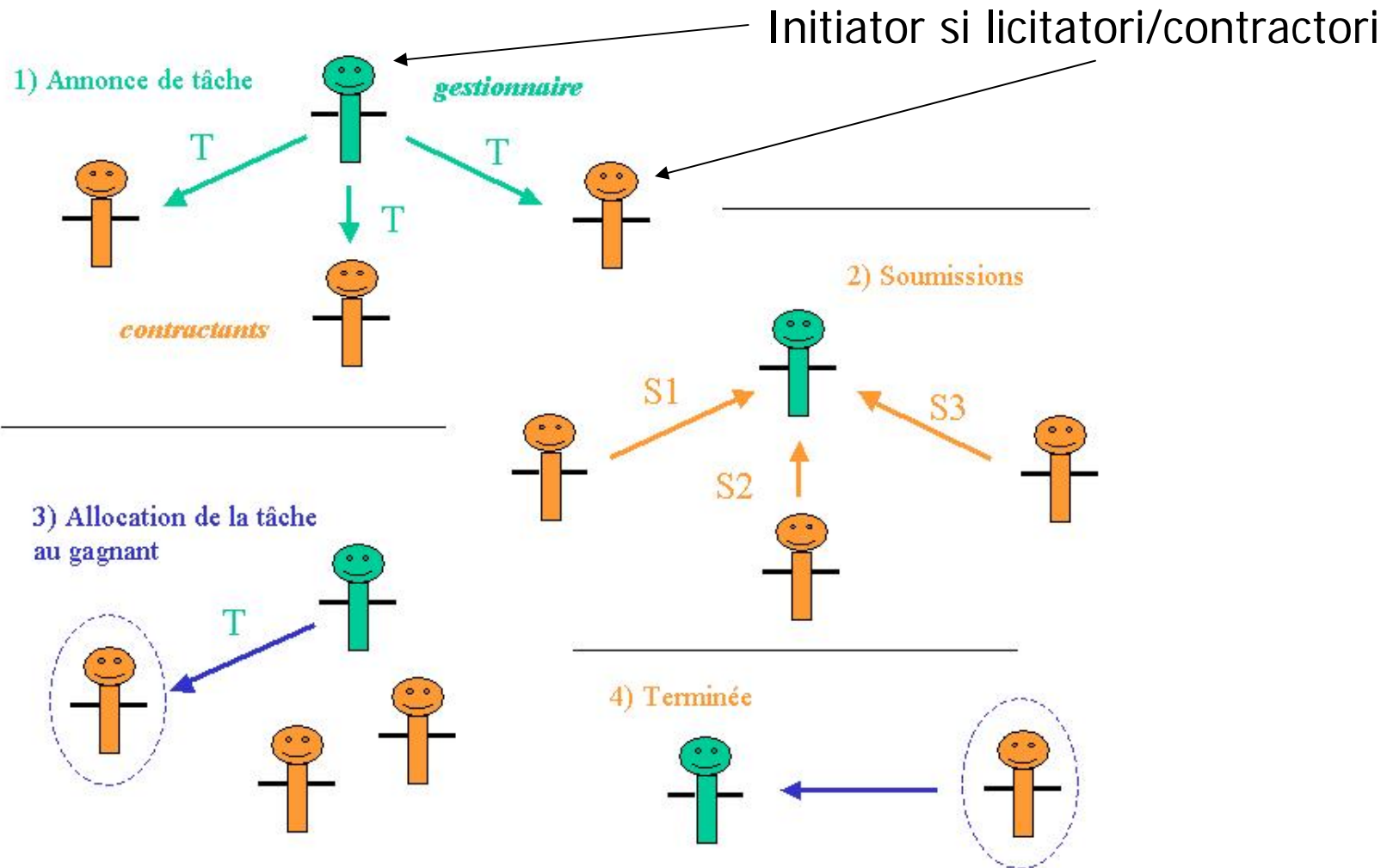


## 6.4. Alocarea taskurilor prin negociere

- Alocare prin retea de contracte
  - Contract Net
  - Iterated Contract Net

Se afla intre negociere teoretica si euristica

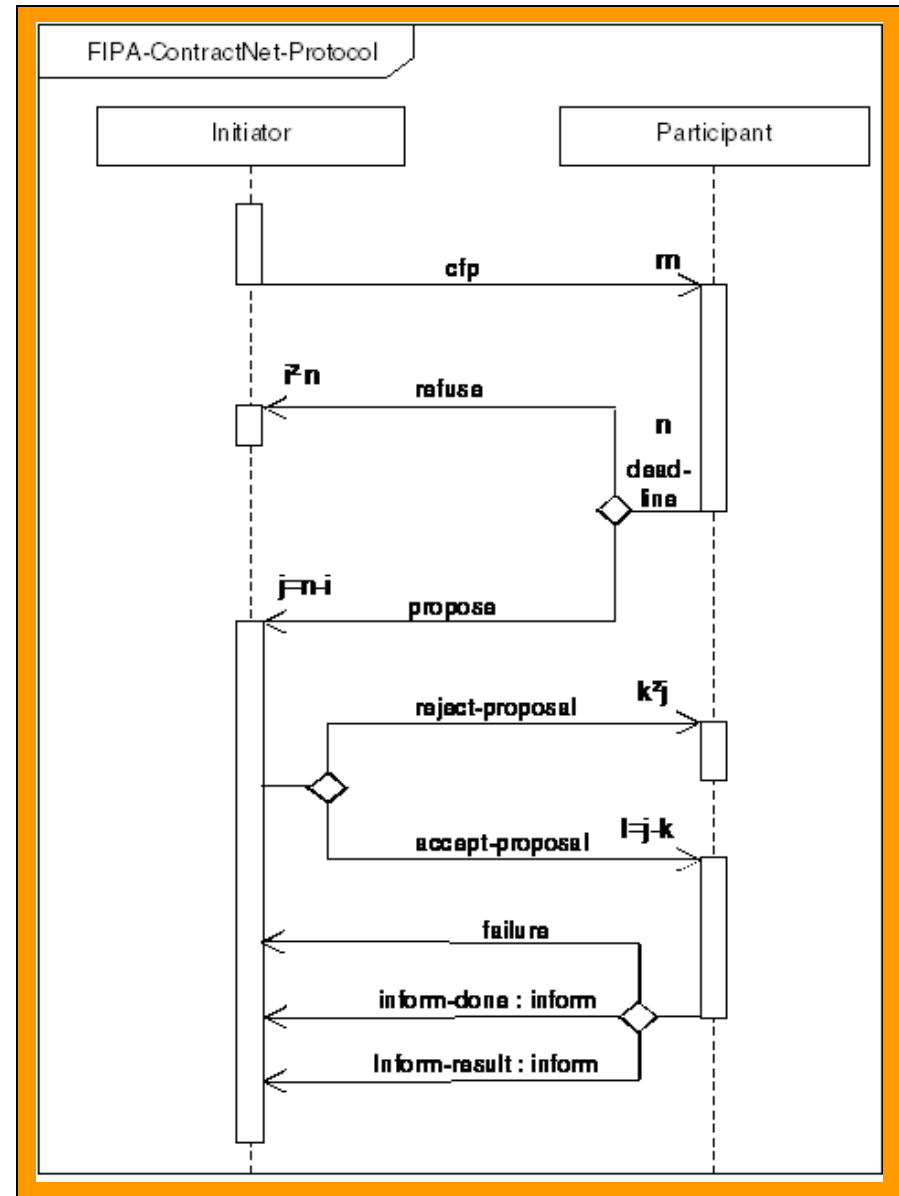
# Contract Net



# FIPA - Contract net

- **Initiatorul solicita propuneri** –  
**cfp**: specifica taskul si conditii asupra lui, de la  $n$  participanti (contractori)
- Cei  $n$  participanti **genreaza raspunsuri**:
  - $i$  refuza (**refuse**)
  - $j=n-i$  propun (**propose**), inclusiv conditii
- cfp include un deadline pt raspunsuri (apoi sunt automat excluse)
- Initiatorul **evaluateaza propunerile** si selecteaza  $l$  (intre 1 si  $j$ ) agenti –  
li se trimite mesaj de **accept-proposal** si mesaj de **reject-proposal** la  $k=j-l$  agenti
- Propunerile **angajeaza** participantii
- Un participant trebuie sa **raspunda** cu:
  - **inform-done** sau
  - **inform-result** (daca a executat taskul) sau
  - **failure**.

Interactiunea este identificata printr-un unic parametru *conversation-id*







# Exemplu

Agentul /i cere lui /j propuneri de vanzare  
a 50 cutii de prune si conditii de pret

(cfp

:sender (agent-identifier :name j)

:receiver (set (agent-identifier :name i))

:content

"((action (agent-identifier :name i)

(sell plum 50))

(any ?x (and (= (price plum) ?x) (< ?x 10))))"

:ontology fruit-market

:language fipa-sl)



# Exemplu

Agentul *j* propune lui *i* sa-i vanda  
50 cutii prune la pret de 5

(propose

:sender (agent-identifier :name j)

:receiver (set (agent-identifier :name i))

:content

"((action j (sell plum 50))

(= (any ?x (and (= (price plum) ?x) (< ?x 10)))) 5)"

:ontology fruit-market

:in-reply-to proposal2

:language fipa-sl)



# Exemplu

Agentul / accepta pe j

(accept-proposal

:sender (agent-identifier :name i)

:receiver (set (agent-identifier :name j))

:in-reply-to bid089

:content

```
" ((action (agent-identifier :name j)
          (sell plum 50))
   (= (price plum) 5))) "
```

:language fipa-sl)



# Exemplu

Agentul /il refuza pe k

(reject-proposal

:sender (agent-identifier :name i)

:receiver (set (agent-identifier :name k))

:content

"((action (agent-identifier :name k)

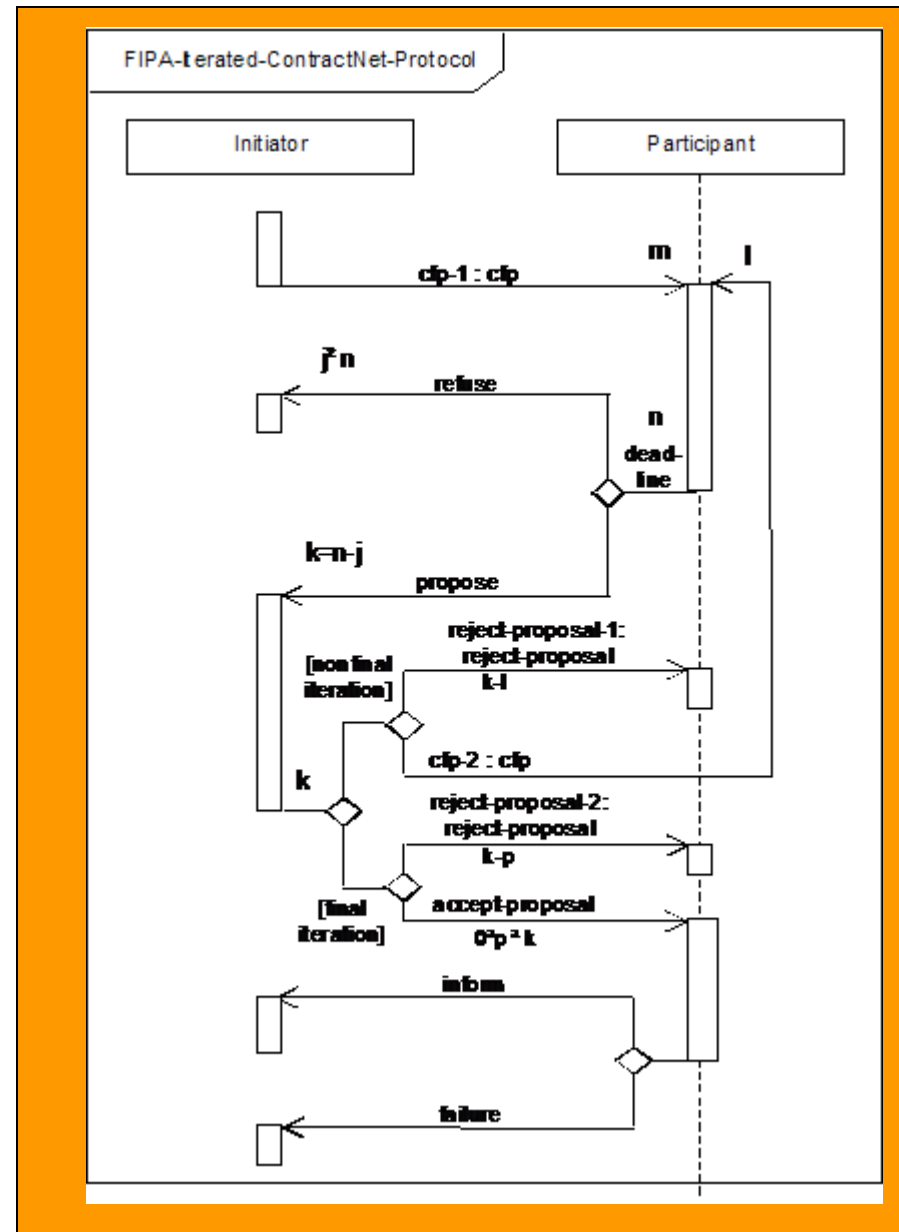
(sell plum 50))

(= (price plum) 20)

(price-too-high 20))"

:in-reply-to bid080)

# FIPA – Iterated Contract net





## 6.5 Negociere euristica

- Produce o solutie buna dar nu optima (de obicei)
- Modele de negociere informale
- Nu exista un mediator central
- Protocolul trebuie sa fie cunoscut de agenti (nu exista protocoale predefinite)
- Nu exista un curs optim de urmat prescris
- Accent pe procesul de decizie al agentului

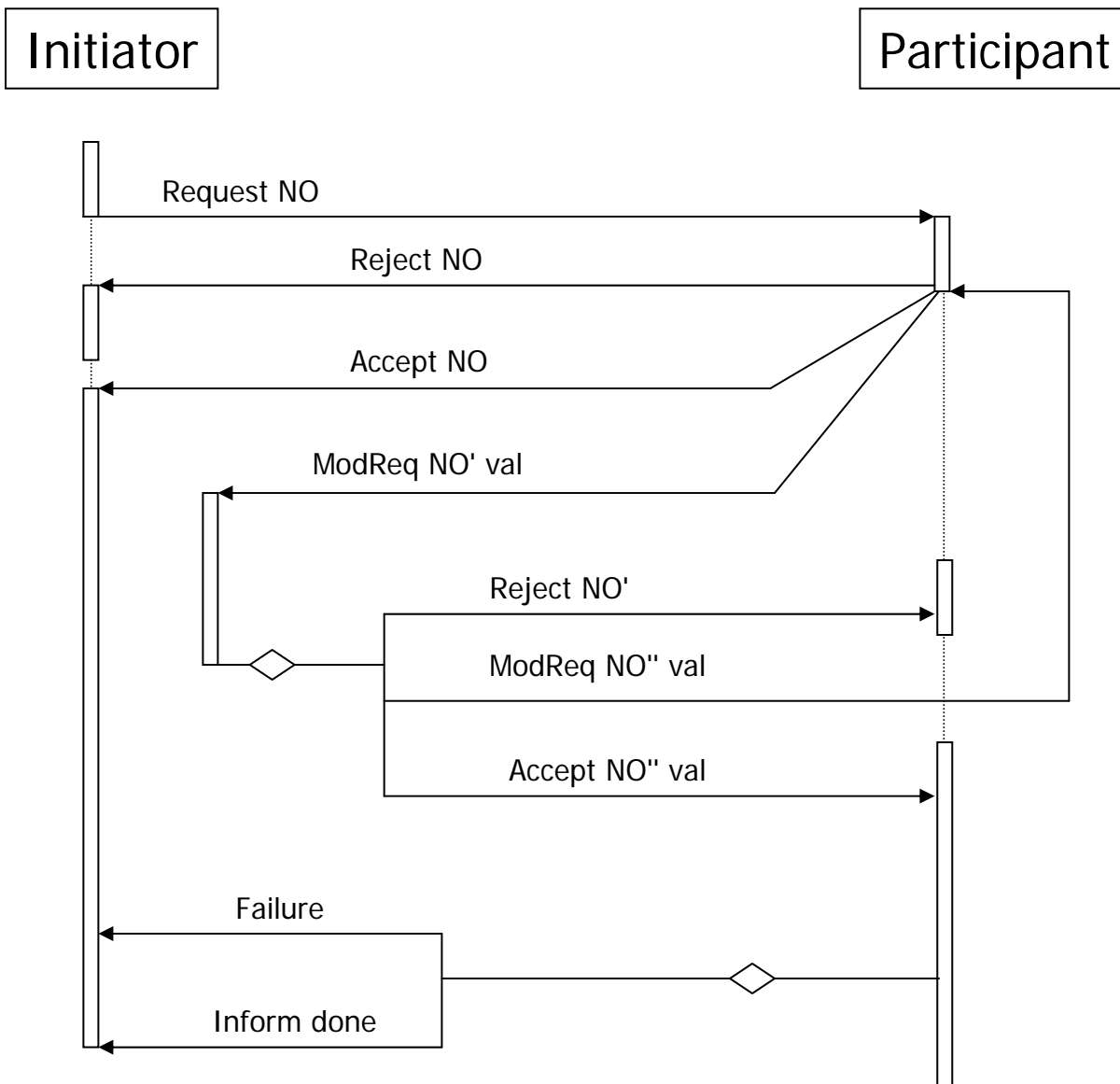
**Obiectul negocierii (NO):** aspectele asupra carora trebuie ajuns la un contract

- NO: obiect, actiune, serviciu

### **NO03: NO**

- **Name:** Paint\_House
  - **Cost:** Value:100, Type: integer, Modif=Yes;
  - **Deadline:** Value: May\_12, Type: date, Modif=No;
  - **Quality:** Value: high, Type: one of (low, average, high), Modif=Yes
- 
- **(Request NO)** – cere un obiect de negociere
  - **(Accept name(NO))** – accepta cererea pentru NO
  - **(Reject name(NO))** – refuza cererea pentru NO
  - **(ModReq name(NO) value(NO,X,V1))** – modifica cererea prin modificarea atributului X al NO la o valoare V1
- 
- Necesita definirea unui limbaj si a unui protocol

## Protocol pentru primitivile definite







## 7. Comunicare in SMA

- Comunicare indirecta
- Comunicare directa
  - ACL
  - Limbaje pentru continut
  - Teoria actelor de vorbire
  - KQML
  - FIPA and FIPA-ACL
- Protocoale de interactiune



# Comunicare in SMA

## Comunicare agenti

- nivel scazut
- nivel inalt
- Implica interactiuni

## Protoale de comunicare

**Protoale de interactiune** – **conversatii** = schimb structurat de mesaje

**ACL: Agent Communication Languages** – limbaje de nivel inalt pentru comunicarea intre agenti

# ACL: 3 straturi ale comunicarii

- **Primitive si protocol**

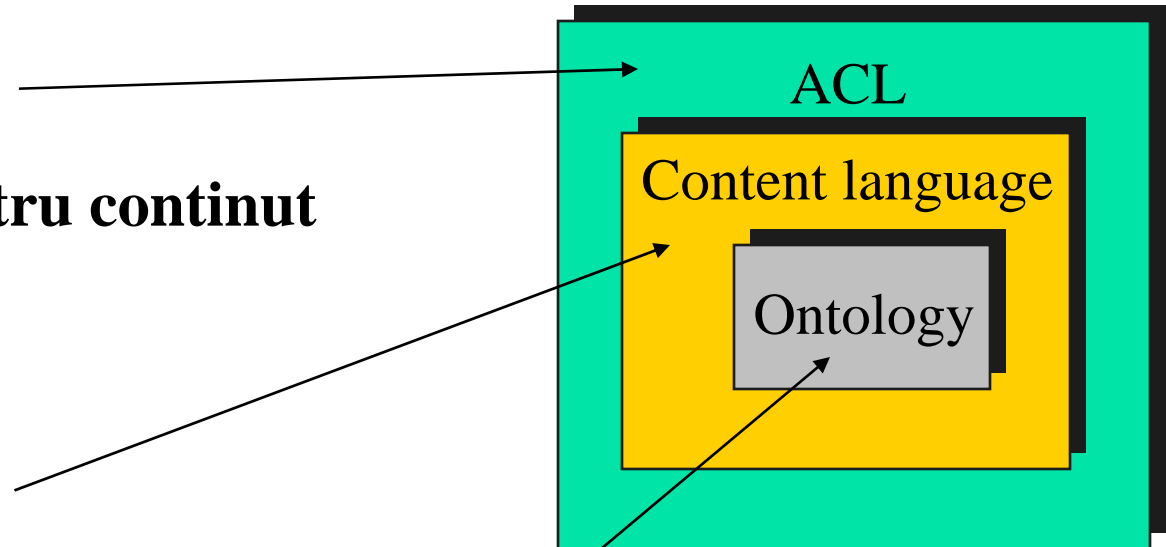
- KQML
- FIPA

- **Limbaje pentru continut**

- KIF
- Prolog
- Clips
- SQL
- DL
- FIPA-SL, FIPA-KIF

- **Ontologii**

- DAML
- OWL





# Primitive ACL

- Bazate pe acte de comunicare / acte de vorbire  
*J. Austin - How to do things with words, 1962,*  
*J. Searle - Speech acts, 1969*
- Cele 3 straturi separa:
  - continutul si semantica mesajului
  - semantica comunicarii (acte vorbire) – independenta de domeniu
- Un ACL are o semantica formala bazata pe un formalism logic
- 2 ACL-uri care s-au impus:
  - KQML
  - FIPA-ACL
- Pot include definitii de protocoale



# FIPA ACL

```
(inform  
  :sender (agent-identifier :name i)  
  :receiver (set (agent-identifier :name j))  
  :content  "weather (today, raining)"  
  :language Prolog)
```



# FIPA - exemple

```
(request :sender (agent-identifier :name i)
        :receiver (set (agent-identifier :name j)
                        :content ((action (agent-identifier :name j)
                                         (deliver box7 (loc 10 15))))
        :protocol fipa-request
        :language fipa-sl
        :reply-with order56 )
```

```
(agree sender (agent-identifier :name j)
       :receiver (set (agent-identifier :name i)
                       :content ((action (agent-identifier :name j)
                                         (deliver box7 (loc 10 15))) (priority order56 low))
       :protocol fipa-request
       :language fipa-sl
       :in-reply-to order56 )
```



# FIPA - primitive

- **FIPA – acte de comunicare**

- **Informative**

- query\_if, subscribe, inform, inform\_if, confirm, disconfirm, not\_understood

- **Distributie taskuri**

- request, request\_whenver, cancel, agree, refuse, failure

- **Negociere**

- cfp, propose, accept\_proposal, reject\_proposal

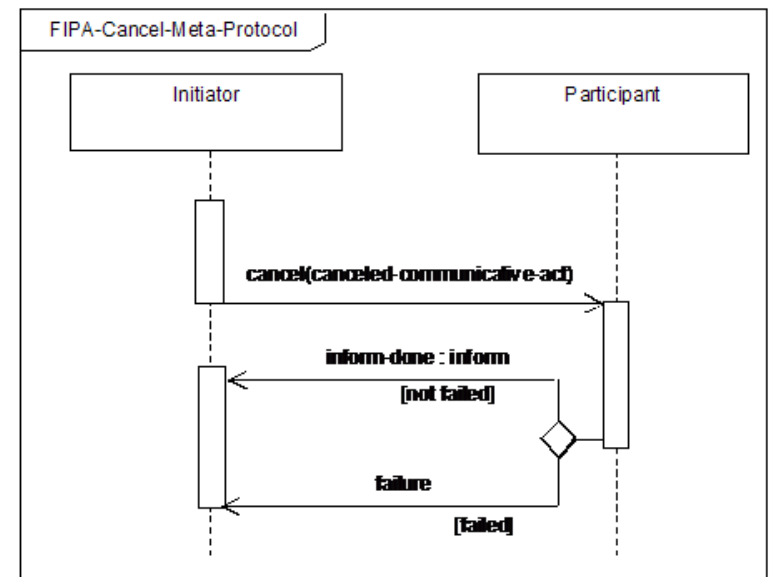
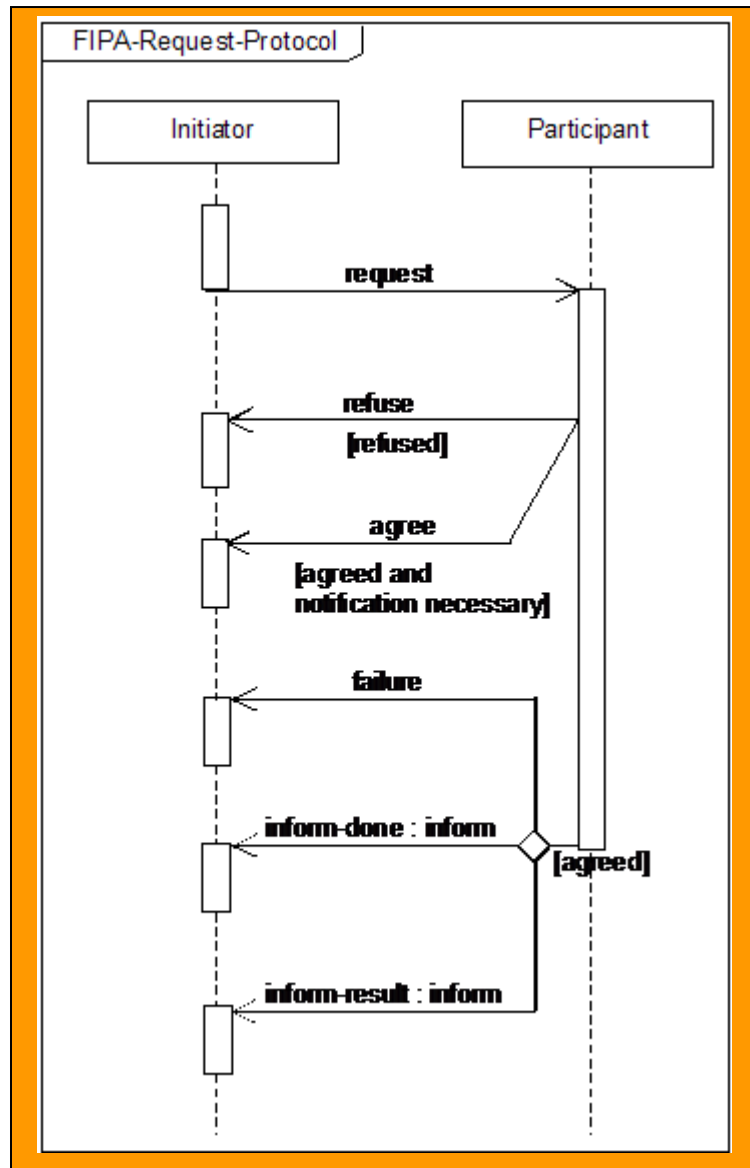


# FIPA - Protocoale

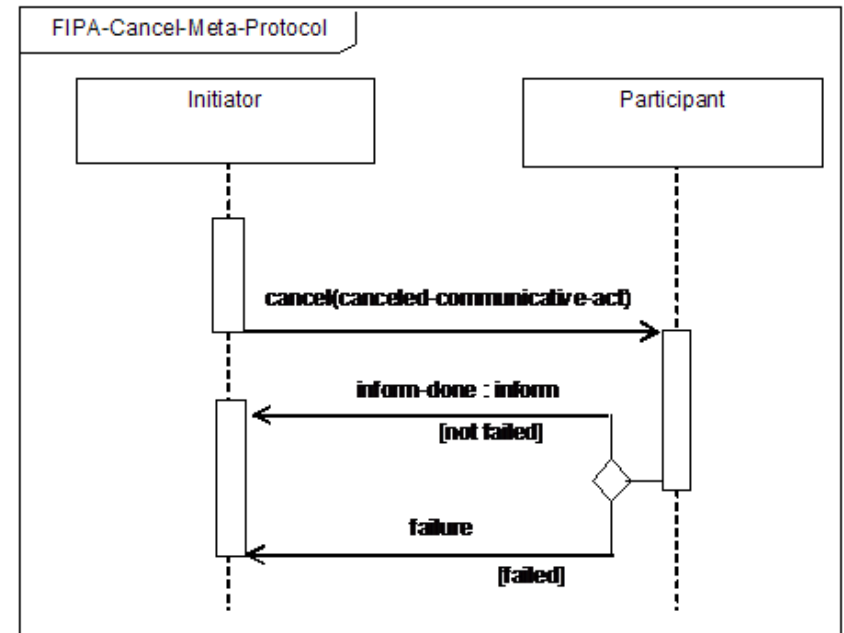
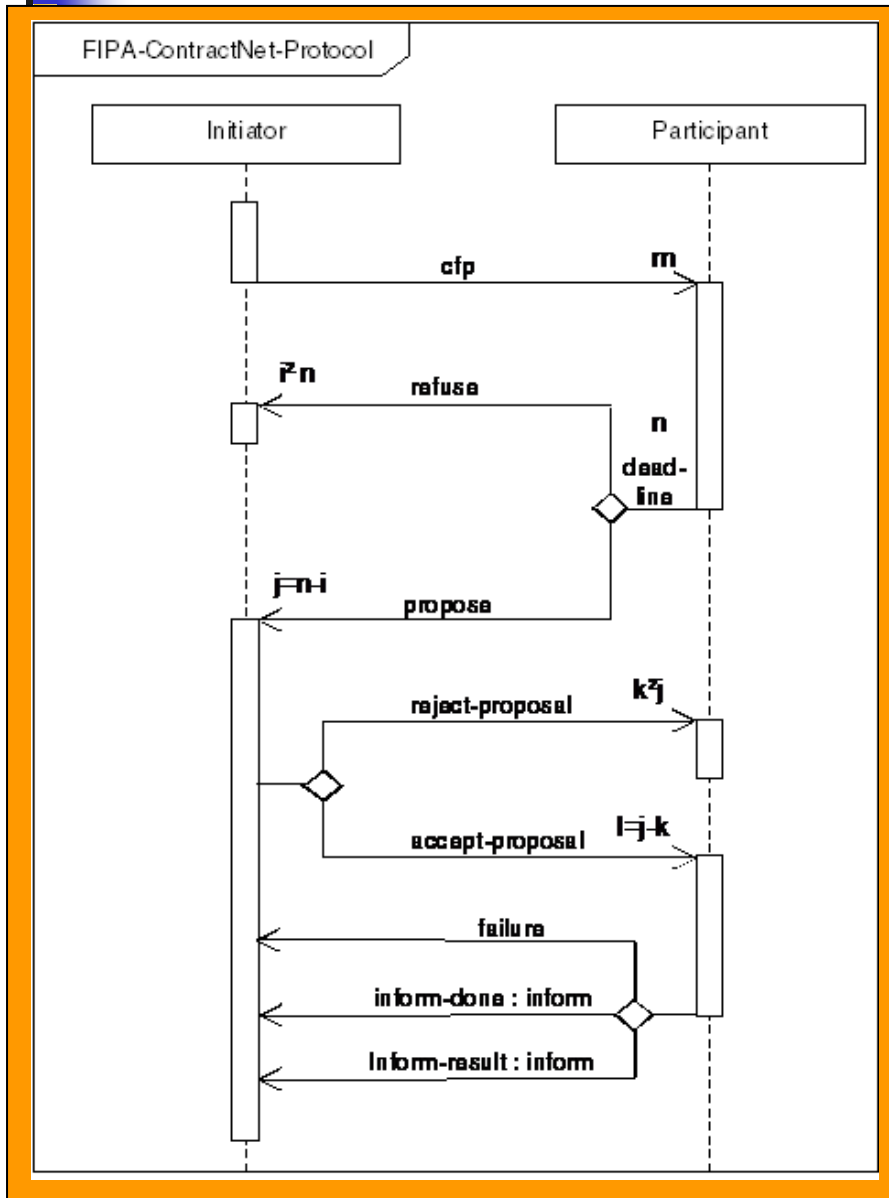
- Defineste o serie de protocoale standard
  - **FIPA-query, FIPA-request, FIPA-contract-net, ...**



# FIPA - Request



# FIPA - Contract net





# Limbaje pentru continut

- KIF
- Prolog
- Clips
- SQL
- FIPA-SL, FIPA-KIF



# FIPA Knowledge Interchange Format (KIF)

## ■ Facts

```
(salary 015-46-3946 john 72000)
(salary 026-40-9152 michael 36000)
(salary 415-32-4707 sam 42000)
```

## ■ Asserted relation

```
(> (* (width chip1) (length chip1))
    (* (width chip2) (length chip2)))
```

## ■ Rule

```
(=> (and (real-number ?x)
          (even-number ?n))
     (> (expt ?x ?n) 0))
```

## ■ Procedure

```
(progn      (fresh-line t)
             (print "Hello!")
             (fresh-line t))
```