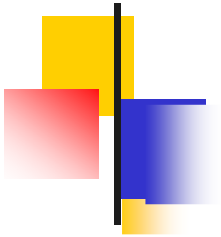
Inteligenta Artificiala

Universitatea Politehnica Bucuresti
Anul universitar 2010-2011

Adina Magda Florea

http://turing.cs.pub.ro/ia_10 si
curs.cs.pub.ro



Curs nr. 12

Agenti inteligenti

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 exsitente (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 etitate care functioneaza **permanent** si **autonom** intr-un mediu in care exsita 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 urmatoarele proprietati:

- **autonomie** – agentii opereaza fara interventai directa a utilizatorui 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

(Shoham, 1993)

- De ce se utilizeaza aceste notiuni?
- Comparatie cu IA

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”

Problema inteleptilor

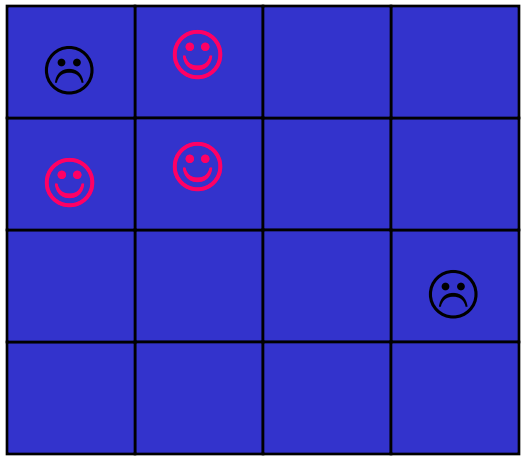
Regele picteaza cate o pata alba si spune ca cel putin o pata este alba

Dilema prizonierului

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

Player A / Player B	***Tradeaza***	***Coopereaza***
Tradeaza	**2 , 2**	**5 , 0**
Coopereaza	**0 , 5**	**3 , 3**

Problema prazilor si vanatorilor



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

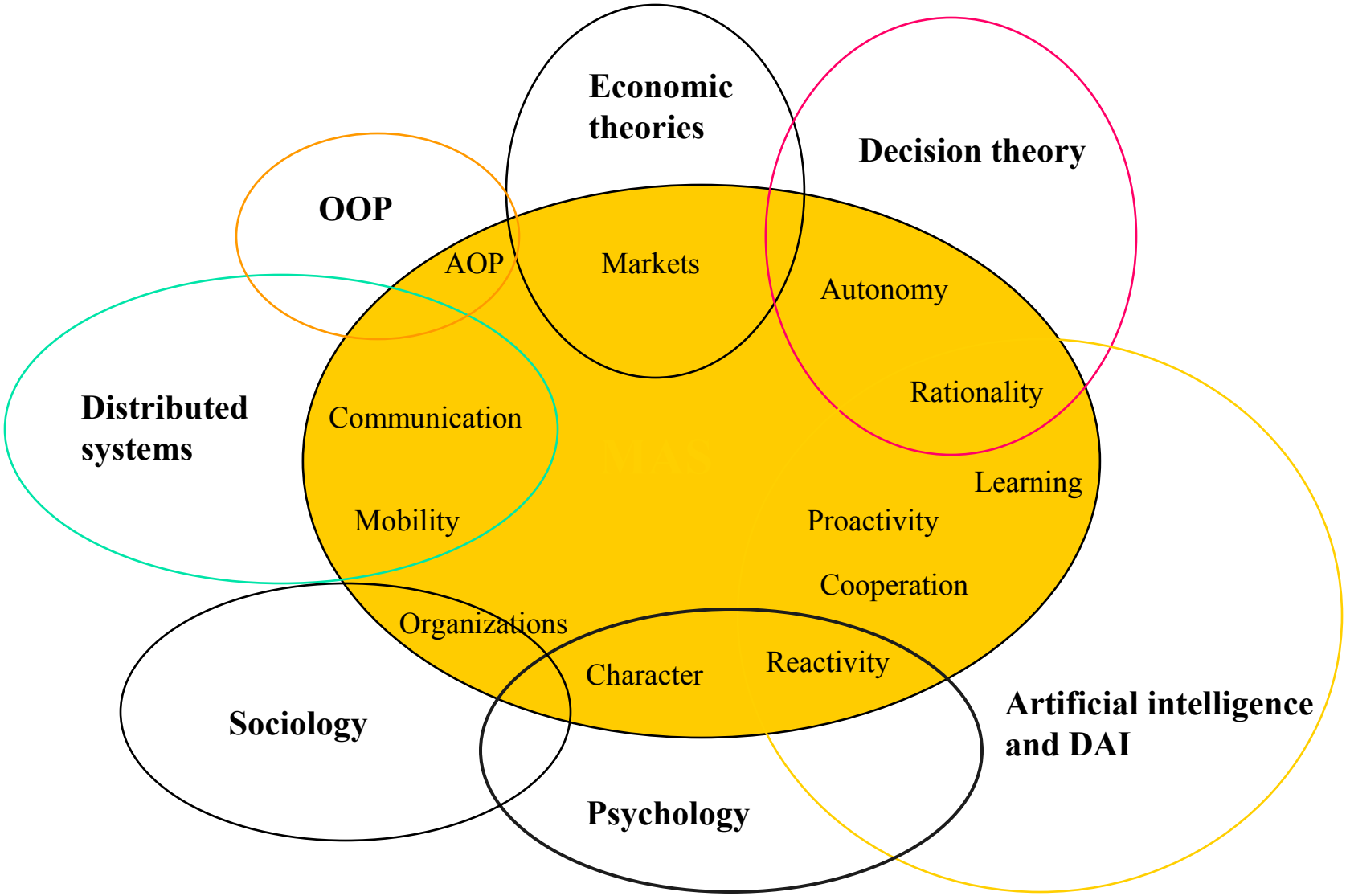
Abordare reactiva

- 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

3.3 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

Legaturi 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, piete de marfuri
- Invatare
- Structuri organizationale
- **Implementare:**
  - Programarea agentilor: paradigme, limbaje
  - Platforme multi-agent
  - Middleware, mobilitate, securitate

5. Modele arhitecturale de agenti

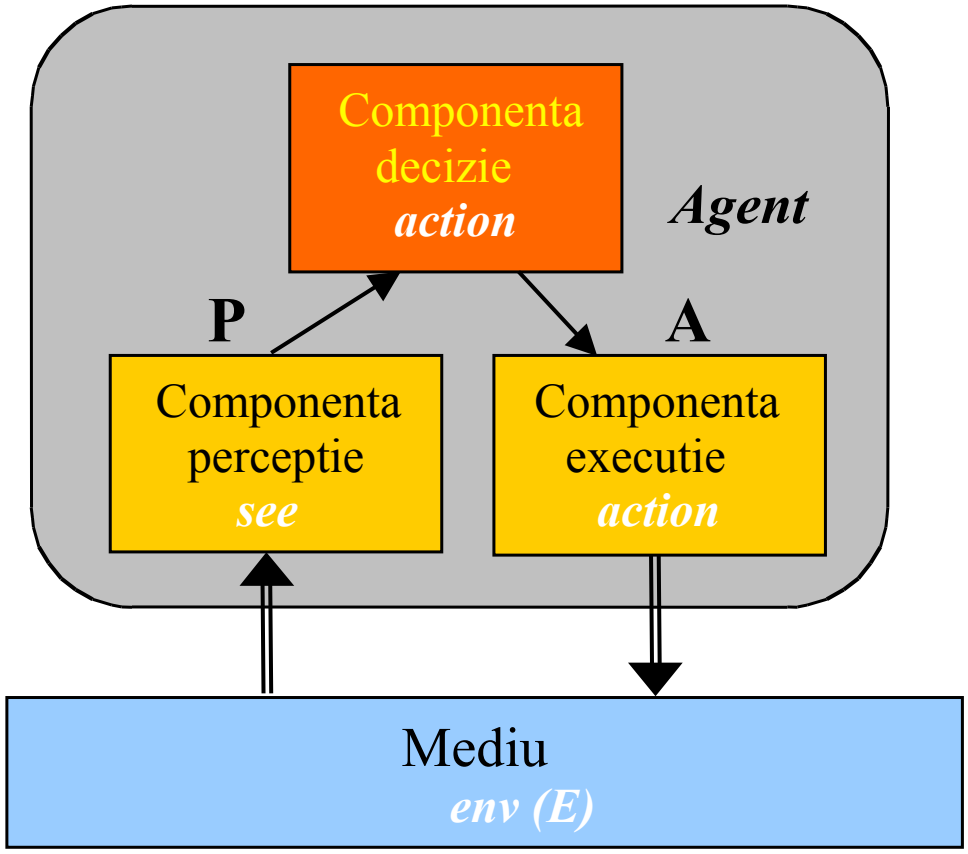
- Structura conceptuala a agentilor
- Arhitecturi de agenti cognitivi
- Arhitecturi de agenti reactivi

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

Modelare agent reactiv



$E = \{e_1, .., e, ..\}$
$P = \{p_1, .., p, ..\}$
$A = \{a_1, .., a, ..\}$

Agent reactiv

see **: E → P**

action **: P → A**

env **: E x A → E**

(env **: E x A →** ***P*****(E))**

Modelare agenti reactivi

Mai multi agenti reactivi

$see_i : E \rightarrow P_i$

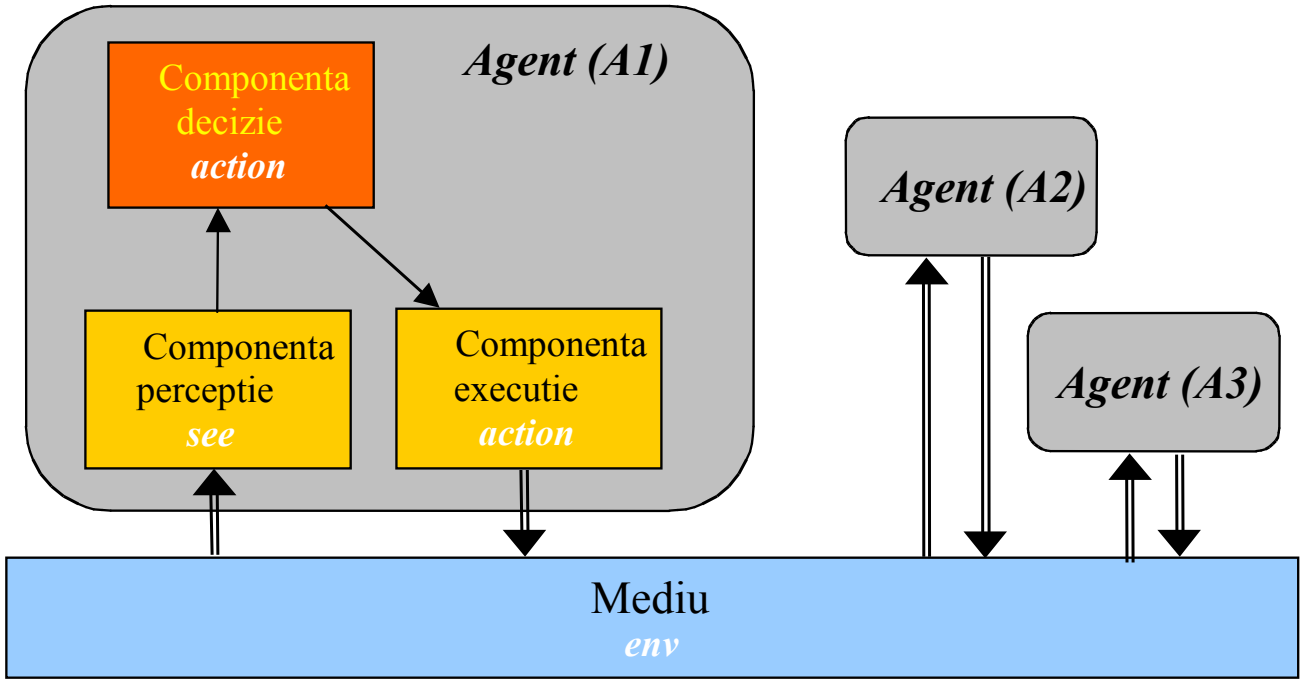
$action_i : P_i \rightarrow A_i$

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

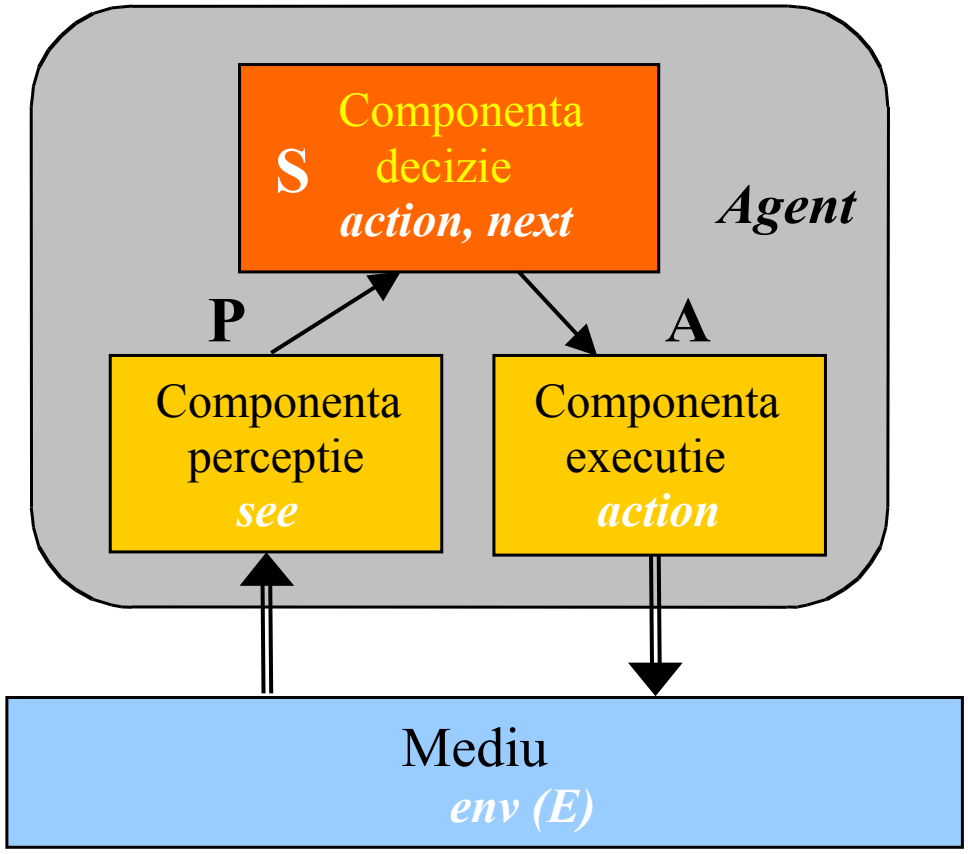
$A_1, \ldots, A_i, ..$

$P_1, \ldots, P_i, ..$

(de obicei identice)



Modelare agent cognitiv



$E = \{e_1, .., e, ..\}$
$P = \{p_1, .., p, ..\}$
$A = \{a_1, .., a, ..\}$
$S = \{s_1, .., s, ..\}$

Agent cu stare

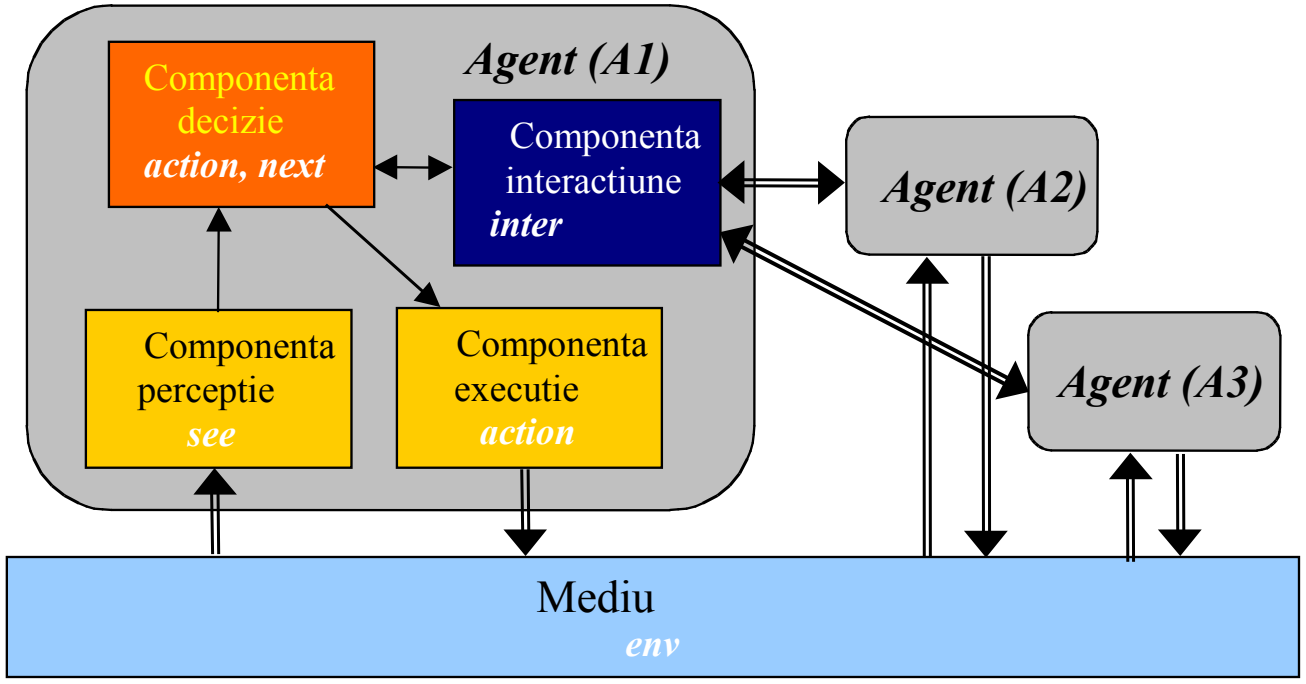
see **: E → P**
next **: S x P → S**
action **: S → A**
env **: E x A →** ***P*****(E)**

Modelare agenti cognitivi

S1,…, Si,..
A_1,…, A_i,..
P_1,…, P_i,..
(nu intotdeauna identice)
$I = \{i_1, .., i_k, \ldots\}$

Mai multi agenti cognitivi

$see_i : E \rightarrow P_i$
$next_i : S_i \times P \rightarrow S_i$
$action_i : S_i \times I \rightarrow A_i$
$inter_i : S_i \rightarrow I$
$env : E \times A_1 \times \ldots A_n \rightarrow P(E)$



Modelare agent cognitiv

Agenti cu stare si scopuri

goal **: E → {0, 1}**

Agenti cu utilitate

utility : E → R

Mediu nedeterminist

env **: E x A →** ***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

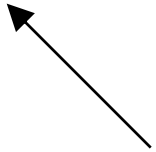
Agenti cu utilitate

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

Exemplu

Cum modelam?

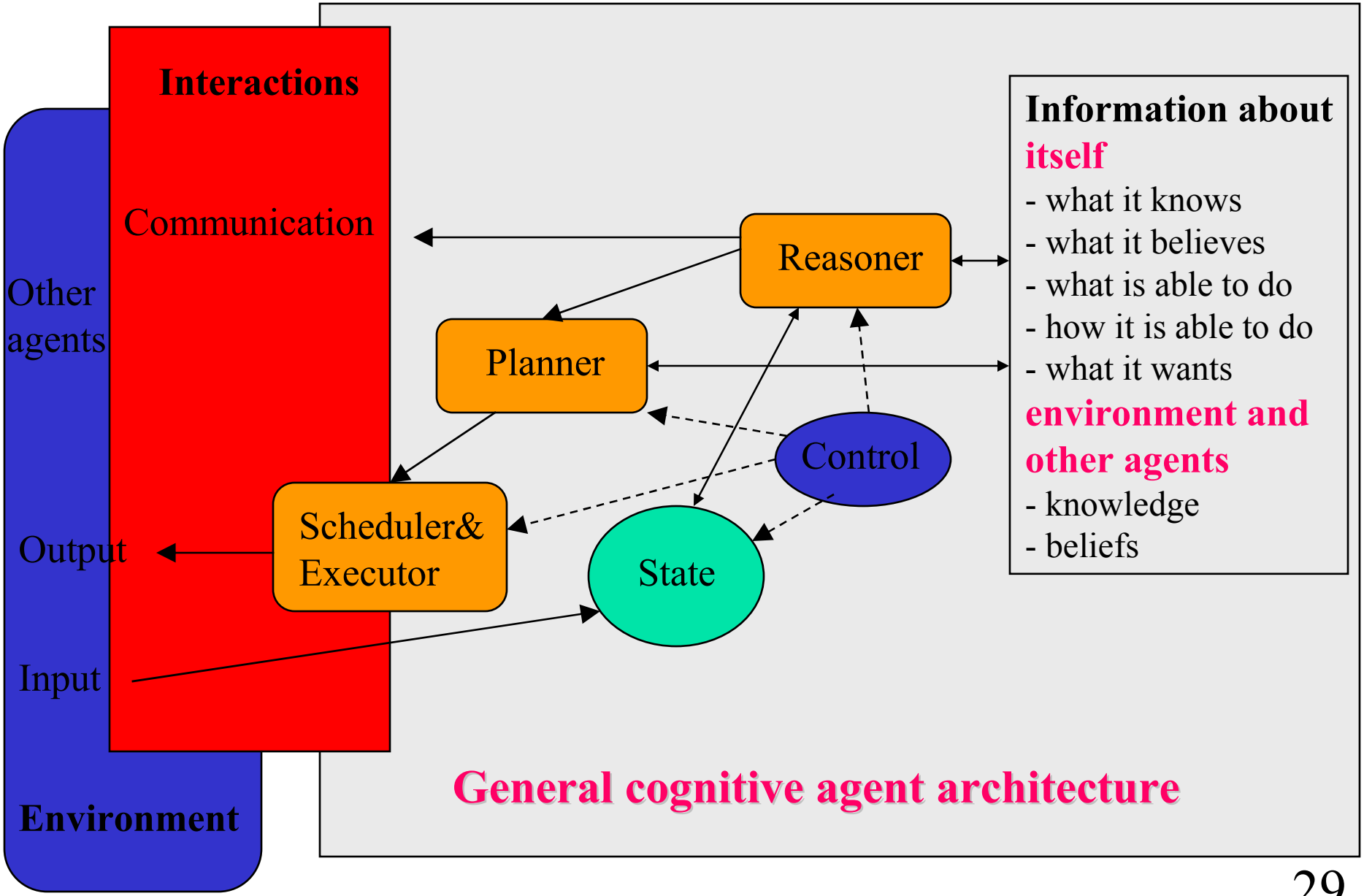
- Curatirea unei camere
  - Agent reactiv
  - Agent cognitiv
  - Agent cognitiv cu utilitate

5.2 Arhitecturi de agenti cognitivi

5.2.1 Comportare rationala

IA si Teoria deciziei

- IA
- Teoria deciziei
- Problema 1 = deliberare/decizie vs. actiune/proactivitate
- Problema 2 = limitarea resurselor



General cognitive agent architecture

5.2.2 Modele LPOI

- Reprezentare simbolica + inferente – demonstrarea teoremelor pt a afla ce actiuni va face agentul
- Abordare declarativa
- **(a)Reguli de deductie**

Predicate At(x,y), Free(x,y), Wall(x,y), Exit(dir), Do(action)

Fapte si axiome despre mediu

At(0,0)

Wall(1,1)

$\forall x \; \forall y \quad Wall(x,y) \rightarrow \neg Free(x,y)$

Reguli de deductie

$At(x,y) \wedge Free(x,y+1) \wedge Exit(east) \rightarrow Do(move\_east)$

Actualizare automata a starii curente si test pt starea scop

At(0,3)

Modele LPOI

(b) Utilizarea calculului situational = descrie schimbari utilizand formalismul logic

- **Situatie** = starea rezultata prin executarea unei actiuni

Result(Action,State) = NewState

At(location, situation)

$At((x,y), S_i) \wedge Free(x,y+1) \wedge Exit(east) \rightarrow$

$At((x,y+1), Result(move\_east,S_i))$

Scop **At((0,3), _)** + actiuni care au condus la scop

means-end analysis

Avantaje LPOI

Dezavantaje

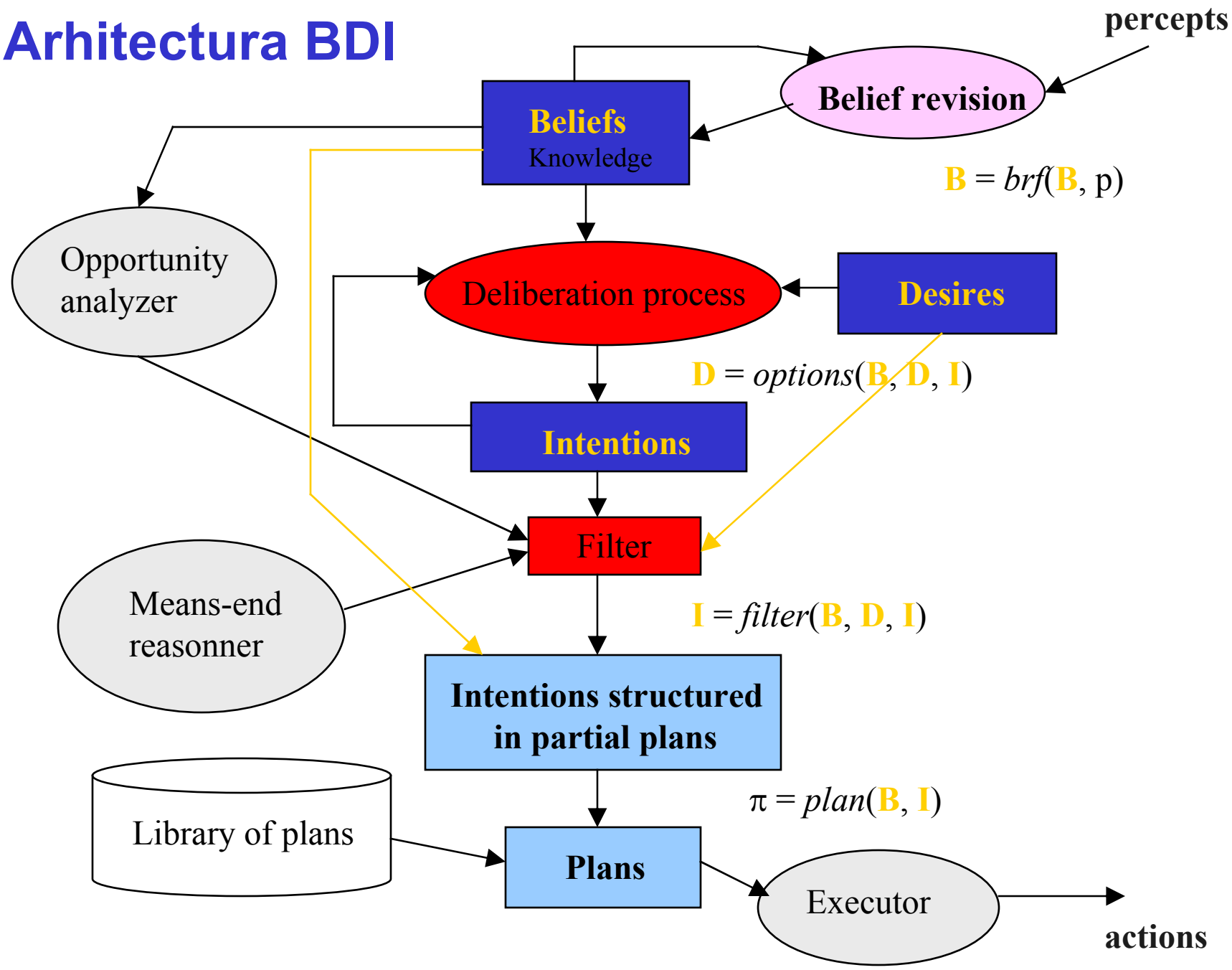
Avem nevoie de un alt model

5.2.3 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**

- Rolul intentiilor

Arhitectura BDI



Bucla de control a agentului

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

while true **do**

get next perceipt p

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

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

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

π = **plan**(**B**, **I**)

execute(π)

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 inetntie?

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

Bucla de control BDI

angajare oarba

```
B = B0
I = I0  D = D0
while  true  do
        get next perceipt p
        B = brf(B,p)
        D = options(B, D, I)
        I = filter(B, D, I)
        π = plan(B, I)
        while not (empty(π) or succeeded (I, B)) do
                α = head(π)
                execute(α)
                π = tail(π)
                get next perceipt p
                B = brf(B,p)
                if not sound(π, I, B) then
                        π = plan(B, I)   <-------  Reactivity, replan
        end while
end while
```

Bucla de control BDI

angajare limitata

```
B = B0
I = I0  D = D0
while  true  do
        get next perceipt p
        B = brf(B,p)
        D = options(B, D, I)
        I = filter(B, D, I)
        π = plan(B, I)
        while not (empty(π) or succeeded (I, B)  or impossible(I, B))  do
                α = head(π)
                execute(α)
                π = tail(π)
                get next perceipt p
                B = brf(B,p)
                if not sound(π, I, B) then
                        π = plan(B, I)
        end while
end while
```

Dropping intentions that are impossible or have succeeded

Reactivity, replan

Bucla de control BDI

angajare deschisa

```
B = B0
I = I0  D = D0
while true do
    get next perceipt p
    B = brf(B,p)
    D = options(B, D, I)
    I = filter(B, D, I)
    π = plan(B, I)
    while not (empty(π) or succeeded (I, B) or impossible(I, B)) do
        α = head(π)
        execute(α)
        π = tail(π)
        get next perceipt p
        B = brf(B,p)
        if reconsider(I, B) then
            D = options(B, D, I)
            I = filter(B, D, I)
        π = plan(B, 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

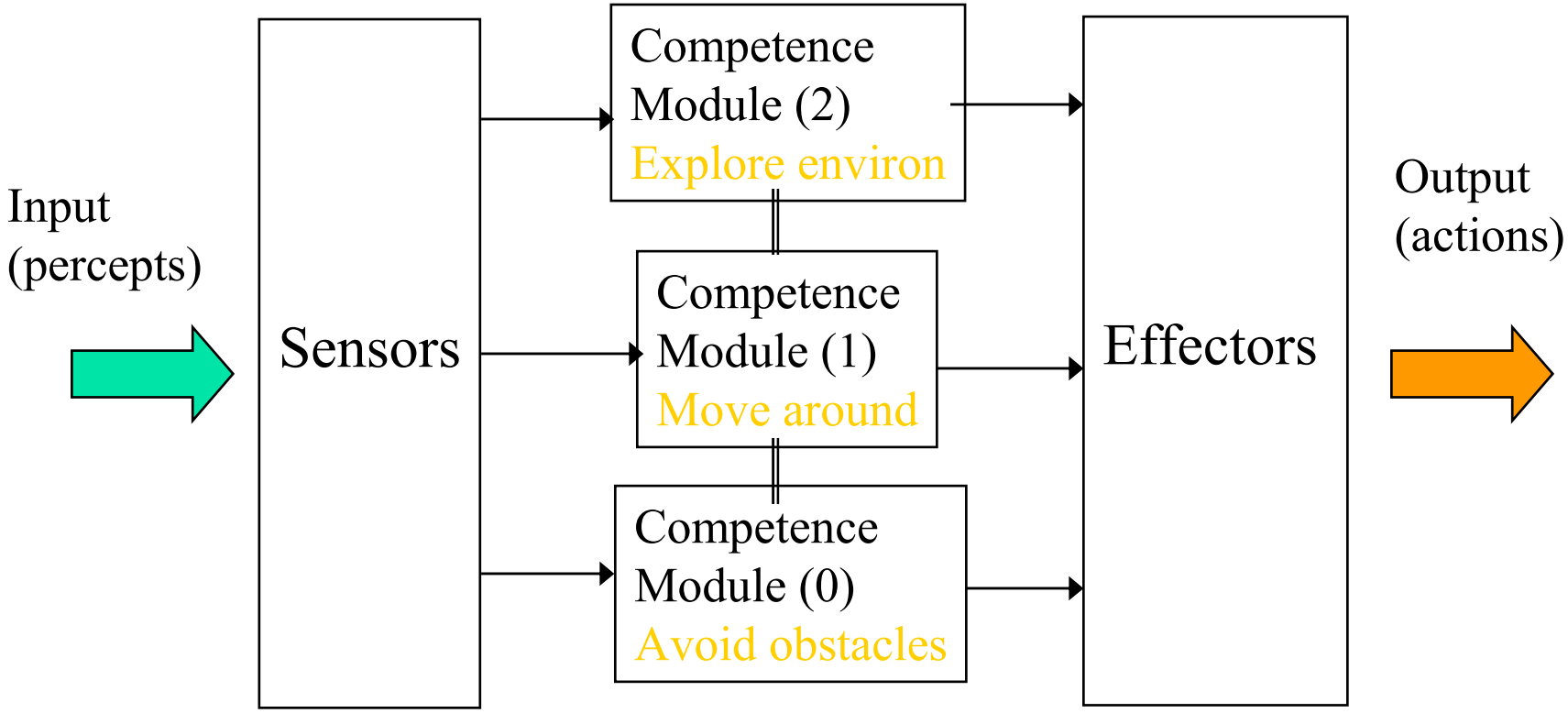
Arhitectura de subsumare - Brooks, 1986

- (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

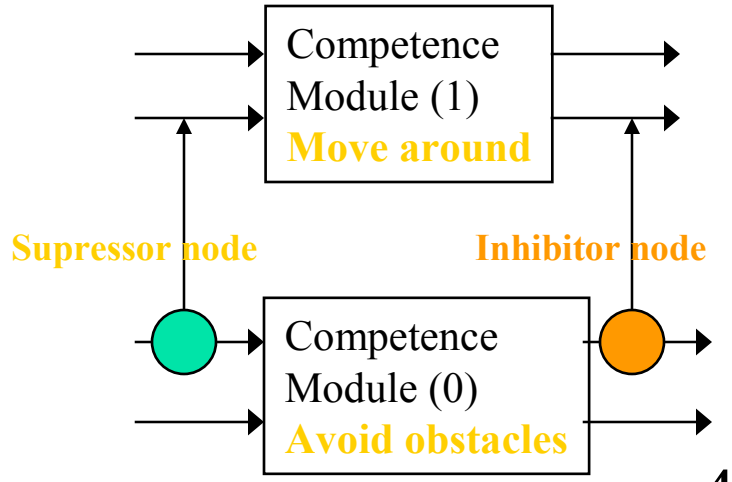
➔ ***subsumtion architecture***



M1 = **move around** while *avoiding obstacles* ⊃ M0

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

- Incoroprarea 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

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

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

function action(p: P)
var fired: P(R), selected: A
begin
    fired = {(c, a) | (c, a) $\in$ R and p $\in$ c}
    for each (c, a) $\in$ fired **do**
    if $\neg \exists$ (c', a') $\in$ 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. Exsita 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. 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

Protocoale de comunicare

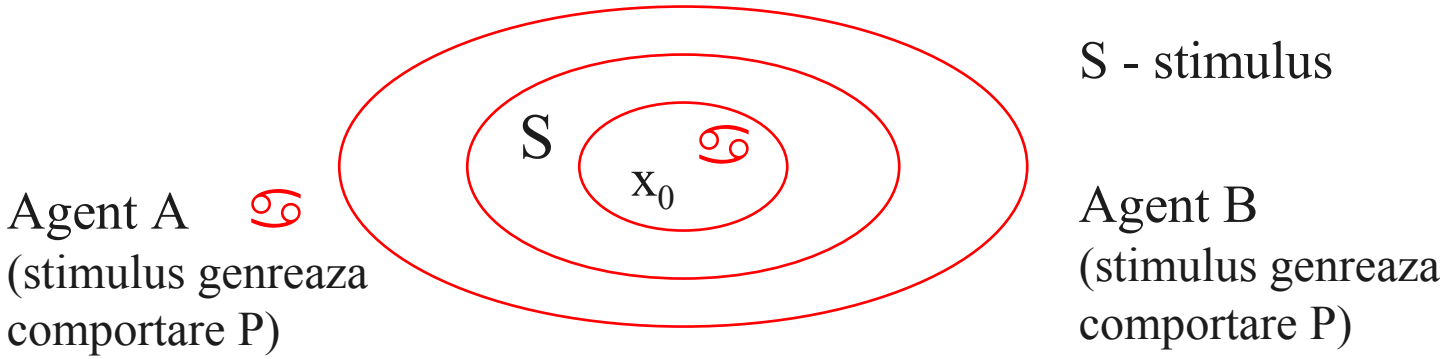
Protocoale de interactiune – **conversatii** = schimb structurat de mesaje

Scop ➔ comunicarea permiet agentilor:

- coordinarea actiunii si comportarii
- schimbarea starii altor agenti
- determina agenti sa faca actiuni

6.1 Comunicare indirecta

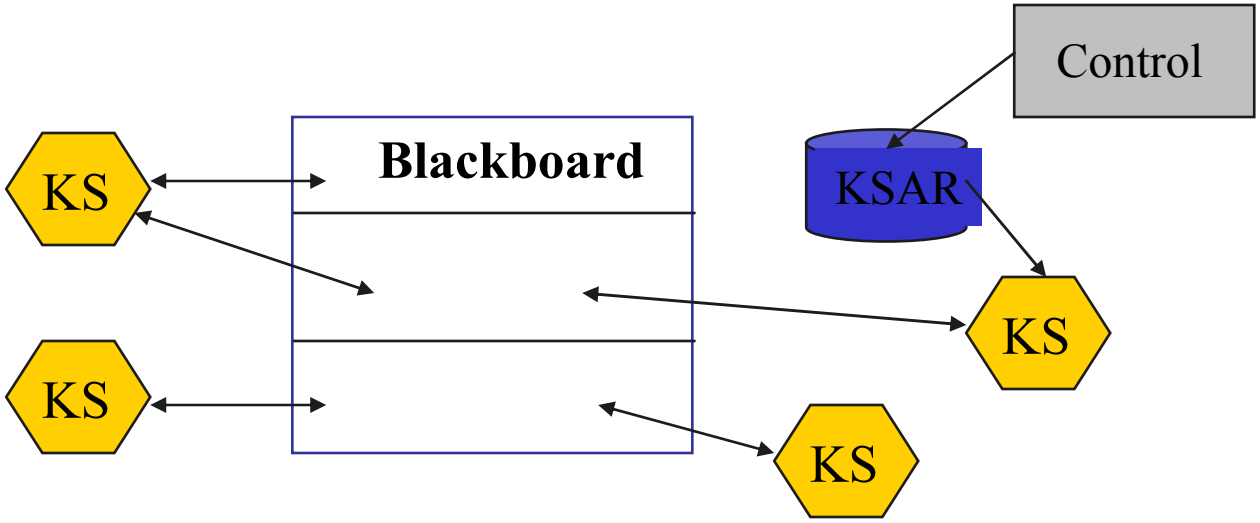
- In general pt agenti reactivi
- Comunicare prin semnale
- $V(x)=V(x_0)/dist(x,x_0)$



- Comunicare prin "urme" lasate in mediu

Comunicare indirecta

- Comunicare in sisteme tip "blackboard"



6.2 Comunicare directa

- SMA – limbaje de nivel inalt
- Presupun in general agenti BDI

- ACL = *Agent Communication Languages*
- Comunica cunostinte → rep. cunostinte
- Intelegerea mesajului in context → ontologii
- Comunicare vazuta ca o actiune – **acte de vorbire (de comunicare)**

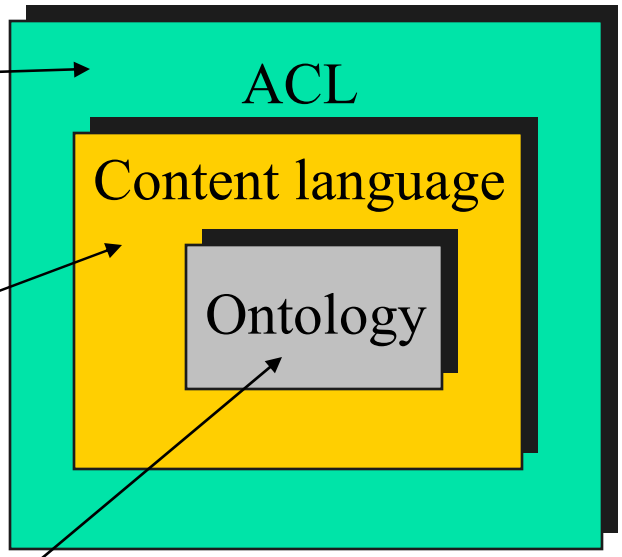
6.2.1 ACL

Concepts (distinguish ACLs from RPC, RMI or CORBA, ORB):

- An ACL message describes a desired state in a declarative language, rather than a procedure or method invocation
- ACLs handle propositions, rules, and actions instead of objects with no associated semantics - ***KR***
- ACLs are mainly based on BDI theories: BDI agents attempt to communicate their BDI states or attempt to alter interlocutor's BDI state – ***Cognitive Agents***
- ACLs are based on Speech Act Theory – ***Communicative Acts***
- ACLs refer to shared ***Ontologies***
- Agent behavior and strategy drive communication and lead to conversations - ***Protocols***

3 straturi ale comunicarii

- **Primitive si protocol**
  - KQML
  - FIPA
- **Limbaje pentru continut**
  - KIF
  - Prolog
  - Clips
  - SQL
  - DL
  - FIPA-SL, FIPA-KIF
- **Ontologii**
  - DAML
  - OWL



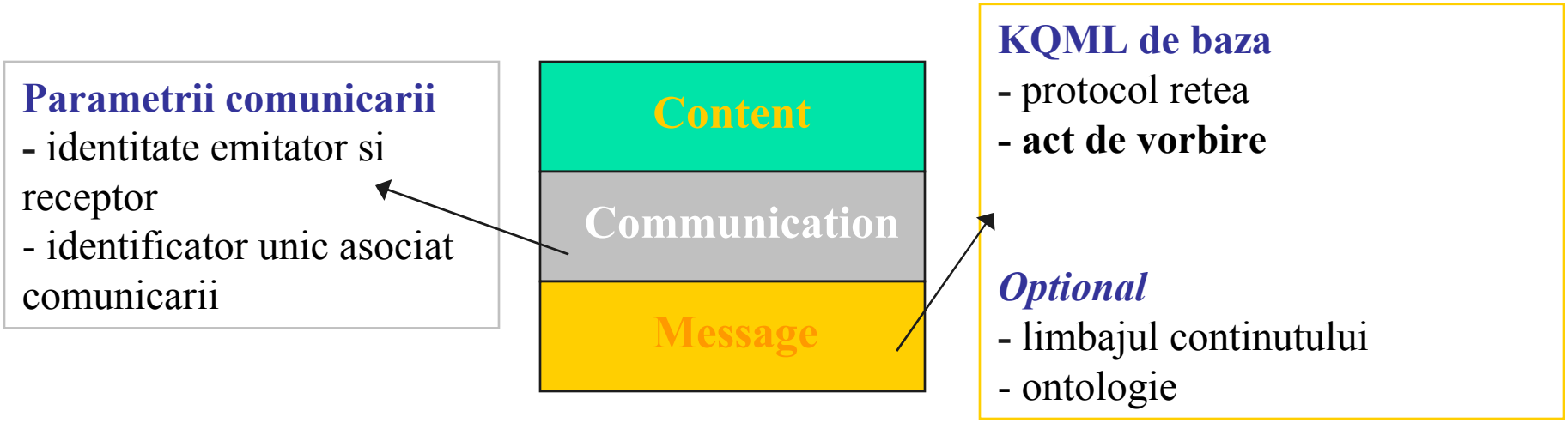
6.2.2 Primitive ACL

- 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 protocoloale

KQML



Tipuri de performative

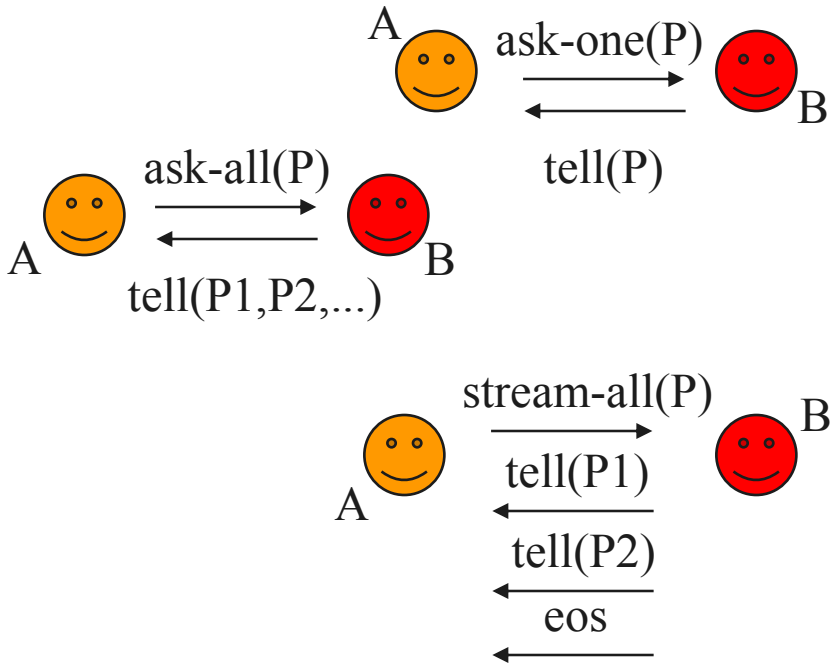
- **Queries** - **ask-one, ask-all, ask-if, stream-all,...**
- **Generative** - **standby, ready, next, rest, discard, generate,...**
- **Response** - **reply, sorry ...**
- **Informational** - **tell, untell, insert, delete, ...**
- **Capability definition** - **advartise, subscribe, recommend...**
- **Networking** - **register, unregister, forward, route, ...**

KQML - exemple

querie

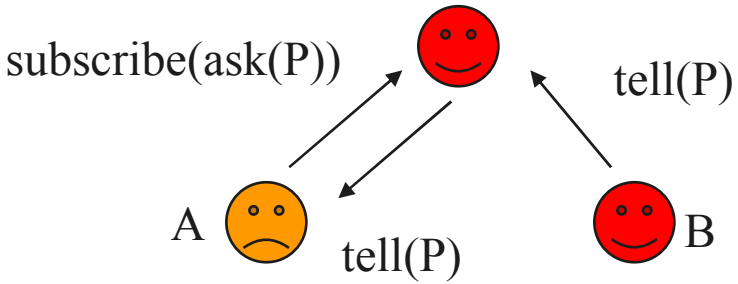
```
(ask-one :sender  joe
         :receiver  ibm-stock
         :reply-with  ibm-stock
         :language  PROLOG
         :ontology NYSE-TICKS
         :content (price  ibm  ?price) )

(tell    :sender  willie
         :receiver  joe
         :reply-with  block1
         :language  KIF
         :ontology BlockWorld
         :content (AND (Block A) (Block B) (On  A  B)) )
```

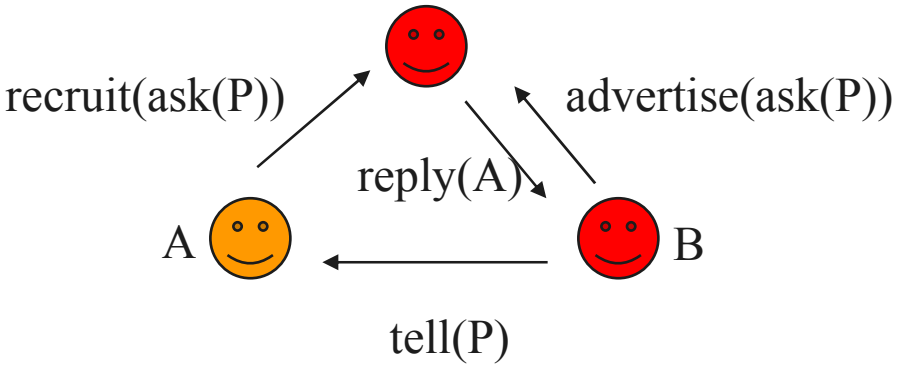
informational



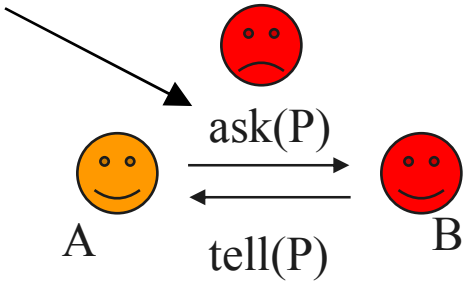
Agent facilitator

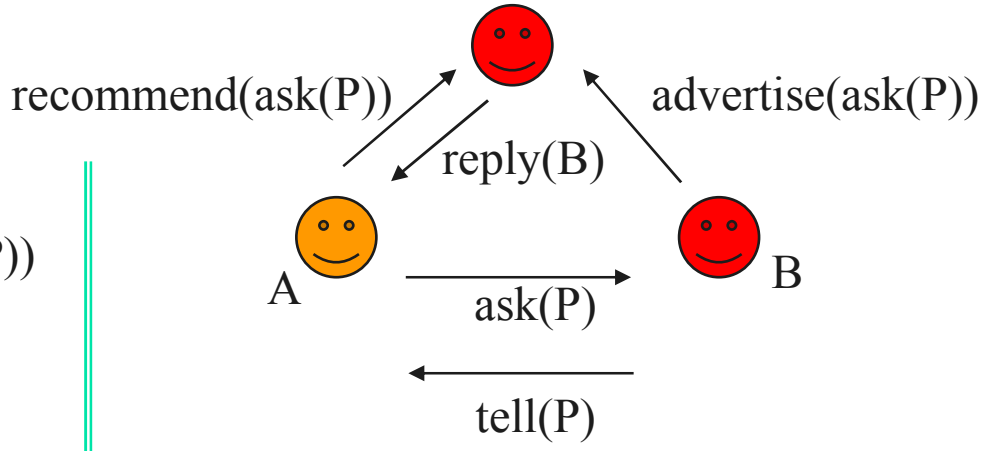
capability





querie point-to-point





FIPA ACL

- Asemanator cu KQML
- Primitive oarecum diferite
- Semantica semnificativ diferita

```
(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-identifer :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-identifer :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)

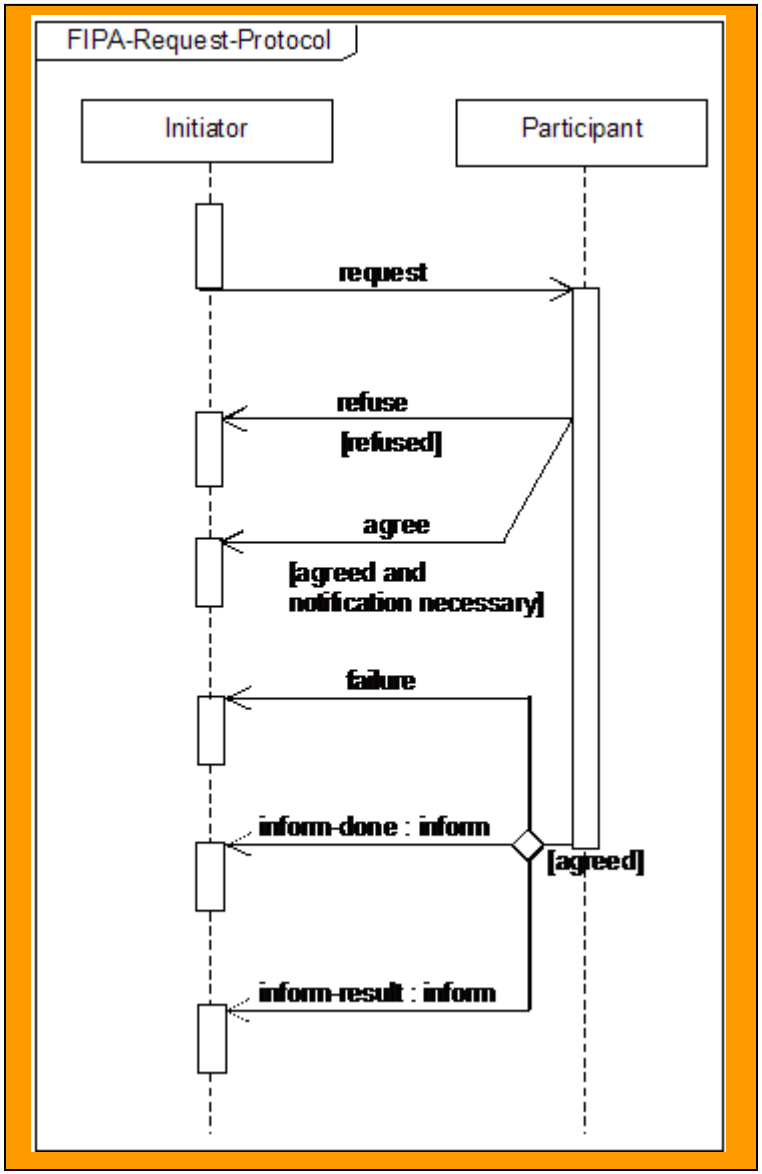
FIPA - primitive

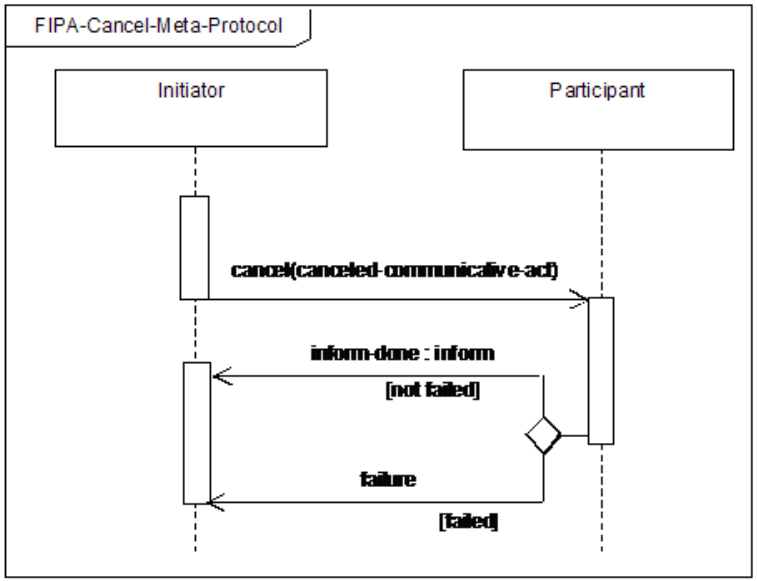
- **FIPA – acte de comunicare**
- **Informative**
  - **query_if, subscribe, inform, inform_if, confirm, disconfirm, not_understood**
- **Distributie taskuri**
  - **request, request_whenever, cancel, agree, refuse, failure**
- **Negociere**
  - **cfp, propose, accept_proposal, reject_proposal**

FIPA - Protocoale

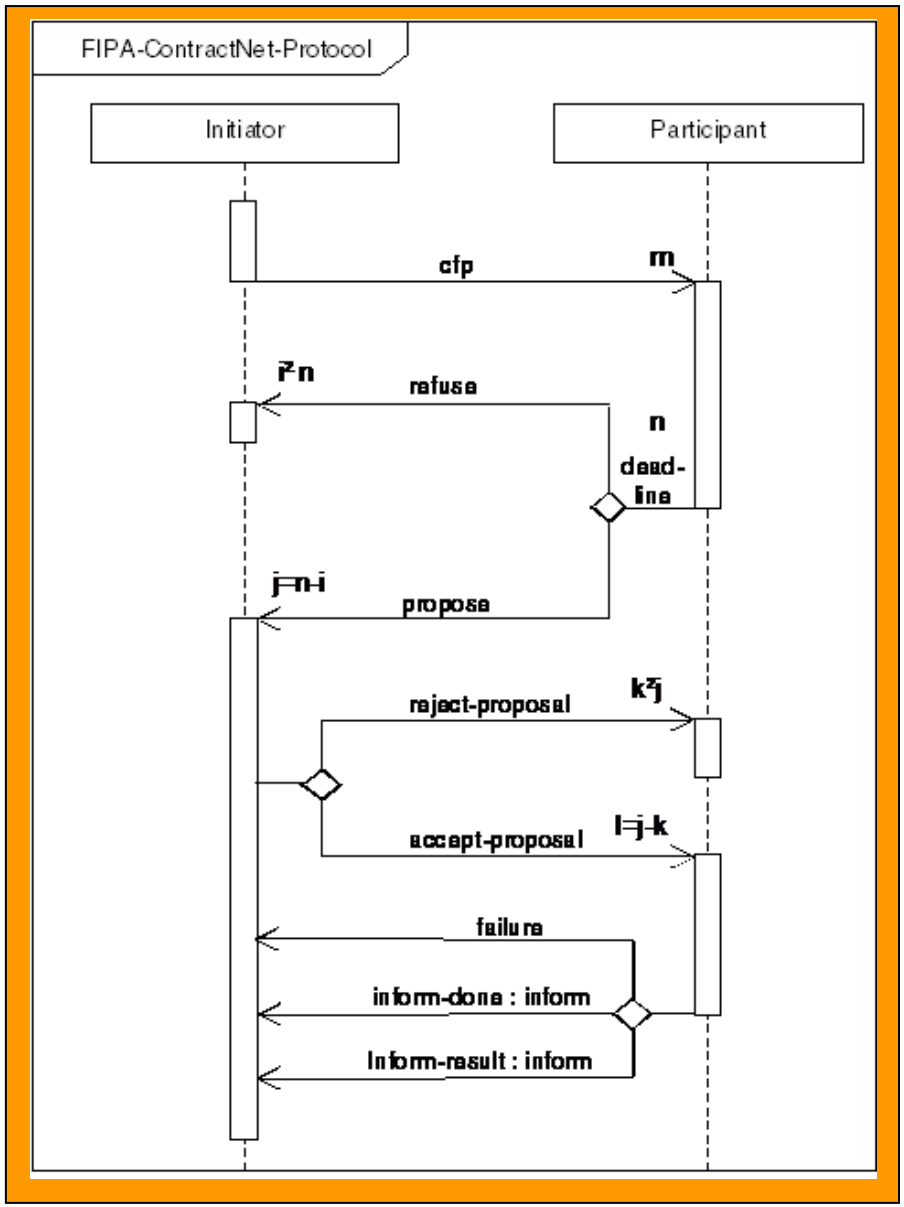
- Defineste o serie de protocoale standard
  - **FIPA-query, FIPA-request, FIPA-contract-net, ...**

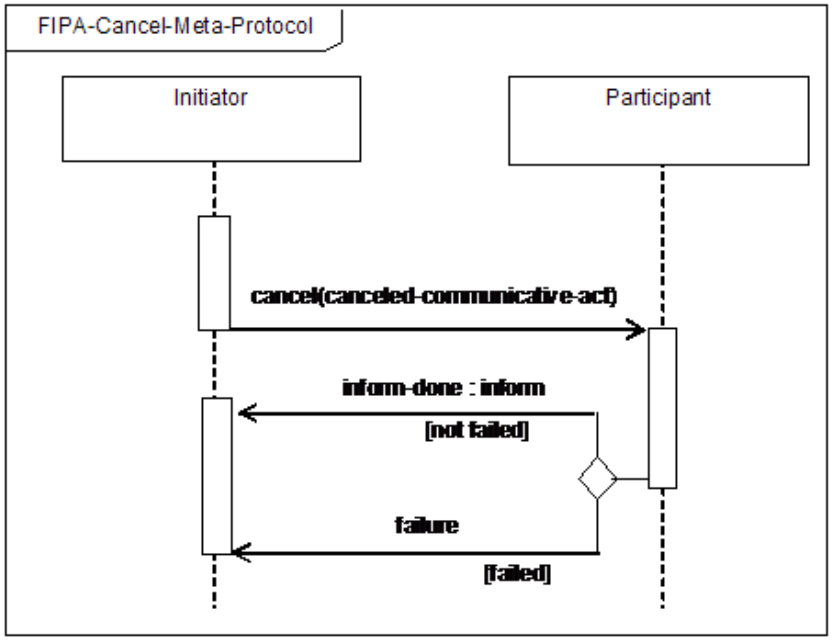
FIPA - Request





FIPA - Contract net





6.3.3 Limbaje pentru continut

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

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

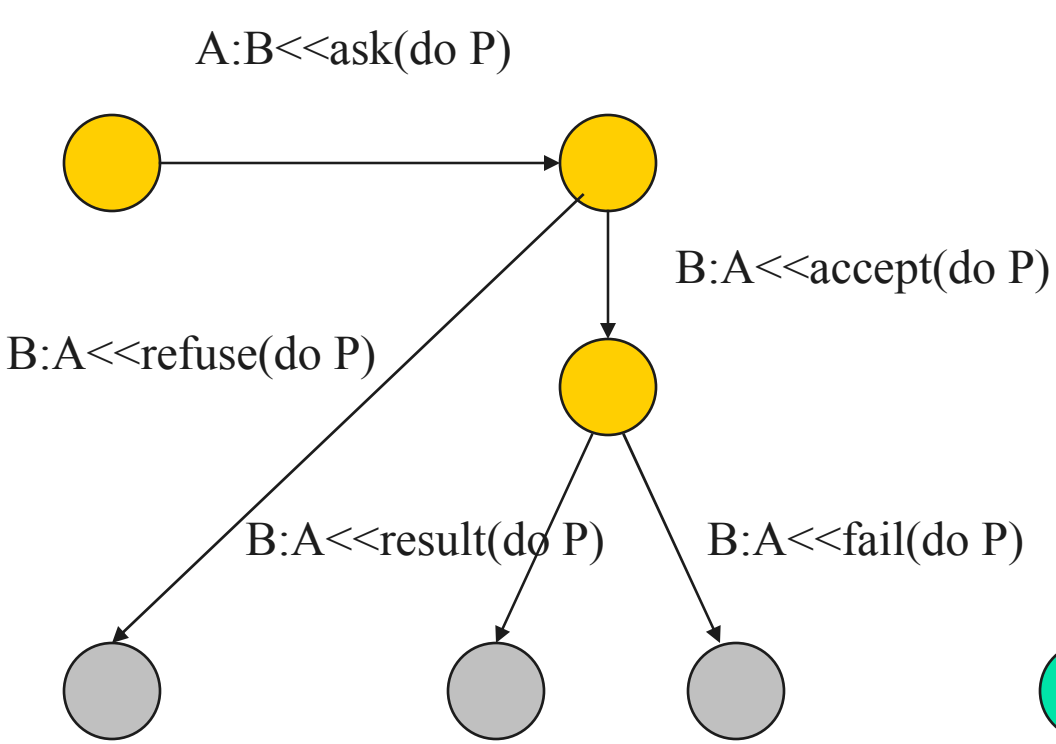
```
(progn (fresh-line t)
          (print "Hello!")
          (fresh-line t))
```

6.3.4 Protocoale de interactiune

Perimit agentilor realizarea de conversatii = schimburi structurate de mesaje

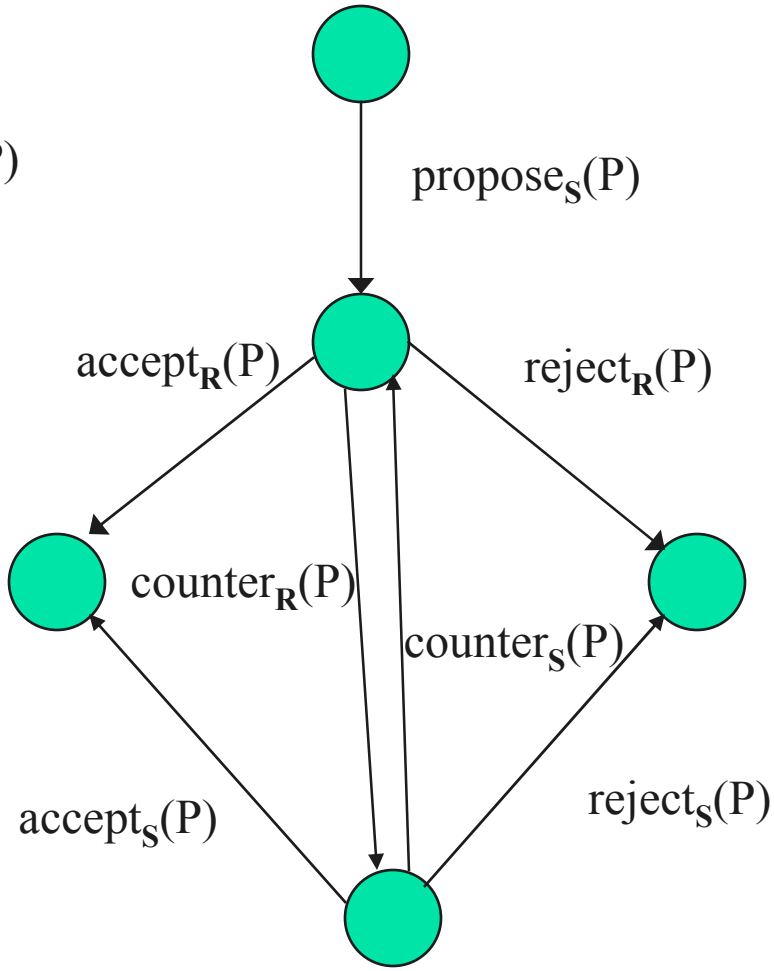
- **Finite automata**
- **Conversations in KQML**
- **Petri nets**

Automate finite



Winograd, Flores, 1986

COOL, Barbuceanu,95



Conversatii in KQML

Definite Clause Grammars

S → **s**(Conv, P, S, R, inR, Rw, IO, Content), {member(P, [advertise, ask-if]}

s(Conv, **ask-if**, S, R, inR, Rw, IO, Content) →

[**ask-if**, S, R, inR, Rw, IO, Content] |

[**ask-if**, S, R, inR, Rw, IO, Content], {OI is inv(IO)},

r(Conv, **ask-if**, S, R, _, Rw, OI, Content)

r(Conv, **ask-if**, R, S, _, inR, IO, Content) →

[**tell**, S, R, inR, Rw, IO, Content] |

problem(Conv, R, S, inR, _, IO)

Labrou, Finin, 1998

Retele Petri

Ferber, 1997