Multi-Agent Systems
Lecture 2

University "Politehnica" of Bucarest
2003 - 2004

Adina Magda Florea
adina@cs.pub.ro

http://turing.cs.pub.ro/blia_2004

Models of agency and architectures
Lecture outline

- Conceptual structures of agents
- Cognitive agent architectures
- Reactive agent architectures
- Layered architectures

1. Conceptual structures of agents

1.1 Agent rationality

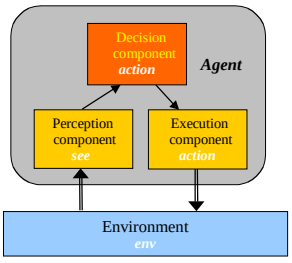
- An agent is said to be **rational** if it acts so as to obtain the best results when achieving the tasks assigned to it.
- *How can we measure the agent's rationality?*
- A **measure of performance**, an objective measure if possible, associated to the tasks the agent has to execute.

- An agent is **situated in an environment**
- An agent perceives its ***environment*** through **sensors** and acts upon the environment through **effectors**.

- **Aim:** design an **agent program** = a function that implements the agent mapping from percepts to actions.
  We assume that this program will run on some computing device which we will call the **architecture**.
  Agent = architecture + program

- The environment
  - accessible vs. inaccessible
  - deterministic vs. non-deterministic
  - static vs. dynamic
  - discrete vs. continue

1.2 Agent modeling



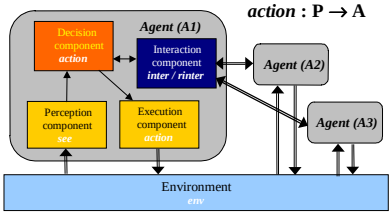
$E = \{e_1, .., e, ..\}$
$P = \{p_1, .., p, ..\}$
$A = \{a_1, .., a, ..\}$

Reflex agent
see **: E → P**
action **: P → A**
env **: E x A → P(E)**

Agent modeling

I = {i1,…,i,..}

Several reflex agents
see **: E → P**
env **: E x A_1 x … A_n → P(E)**
inter **: P → I**
rinter **: I → P**
action **: P → A**



Agent modeling

Cognitive agents

Agents with states S = {s1,…,s,…}

- ***action*** **: S → A_i**
- ***next*** **: S x P → S**
- ***inter*** **: S x P → I**
- ***rinter*** **: S x I → P**
- ***see*** **: E → P**
- ***env*** **: E x A_1 x … A_n → P(E)**

Agent modeling

Agents with states and goals

goal **: E → {0, 1}**

Agents with utility

utility : E → R

Environment non-deterministic

env **: E x A → P(E)**

The probability estimated by the agent that the result of an action (*a*) execution in state *e* will be the new state *e'*

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

Agent modeling

Agents with utility

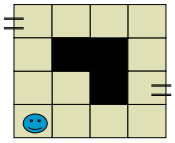
The ***expected utility*** of an action in *a* state *e*, from the agent's point of view

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

The principle of
Maximum Expected Utility (**MEU**) =
a rational agent must choose the action that will bring the maximum expected utility

How to model?

- Getting out of a maze
  - Reflex agent
  - Cognitive agent
  - Cognitive agent with utility



3 main problems:

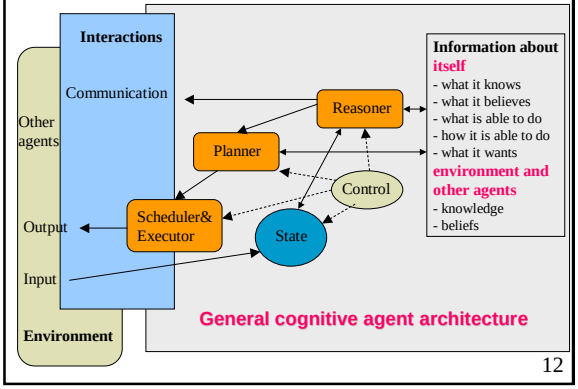
- what action to choose if several available
- what to do if the outcomes of an action are not known
- how to cope with changes in the environment

2. Cognitive agent architectures

2.1 Rational behaviour

AI and Decision theory

- AI = models of searching the space of possible actions to compute some sequence of actions that will achieve a particular goal
- Decision theory = competing alternatives are taken as given, and the problem is to weight these alternatives and decide on one of them (means-end analysis is implicit in the specification of competing alternatives)
- Problem 1 = deliberation/decision vs. action/proactivity
- Problem 2 = the agents are resource bounded



General cognitive agent architecture

2.2 FOPL models of agency

- Symbolic representation of knowledge + use inferences in FOPL - deduction or theorem proving to determine what actions to execute
- Declarative problem solving approach - agent behavior represented as a theory T which can be viewed as an executable specification

(a) Deduction rules

$At(0,0) \wedge Free(0,1) \wedge Exit(east) \rightarrow Do(move\_east)$

Facts and rules about the environment

$At(0,0)$

$Wall(1,1)$

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

Automatically update current state and test for the goal state **At(0,3)**

13

FOPL models of agency

(b) Use situation calculus = describe change in FOPL

Define a function *Result*(Action,State) = NewState

$At((0,0), S_0) \wedge Free(0,1) \wedge Exit(east) \rightarrow$
$At((0,1), Result(move\_east, S_0))$

Try to prove the goal **At((0,3), _)** and determine the actions that lead to it

- means-end analysis

14

Advantages of FOPL

- simple, elegant
- executable specifications

Disadvantages

- difficult to represent changes over time other logics
- decision making is deduction and selection of a strategy
- intractable
- semi-decidable

15

2.3 BDI architectures

- High-level specifications of a practical component of an architecture for a resource-bounded agent.
- It performs means-end analysis, weighting of competing alternatives and interactions between these two forms of reasoning
- **Beliefs = information the agent has about the world**
- **Desires = state of affairs that the agent would wish to bring about**
- **Intentions = desires (or actions) that the agent has committed to achieve**
- BDI - a theory of practical reasoning - Bratman, 1988
- intentions play a critical role in practical reasoning - limits options, DM simpler

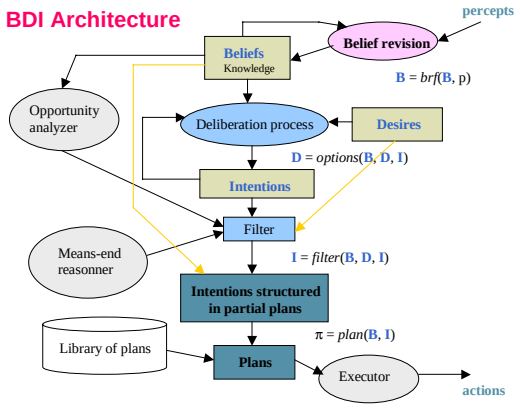
16

BDI particularly compelling because:

- **philosophical component** - based on a theory of rational actions in humans
- **software architecture** - it has been implemented and successfully used in a number of complex fielded applications
  - IRMA - Intelligent Resource-bounded Machine Architecture
  - PRS - Procedural Reasoning System
- **logical component** - the model has been rigorously formalized in a family of BDI logics
  - Rao & Georgeff, Wooldrige
  - $(Int\ A_i\ \varphi) \rightarrow \neg (Bel\ A_i\ \varphi)$

17

BDI Architecture



18

Roles and properties of intentions

- Intentions drive means-end analysis
- Intentions constraint future deliberation
- Intentions persist
- Intentions influence beliefs upon which future practical reasoning is based

Agent control loop

```
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)
        execute(π)
end while
```


Commitment strategies

- 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**
- Commitments 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?

- **Blind commitment**
- **Single minded commitment**
- **Open minded commitment**

Note that the agent is committed to both ends and means.

Revised BDI agent control loop

single-minded commitment

```
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

Revised BDI agent control loop

open-minded commitment

```
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)
        end while
end while
```

Replan

3. Reactive agent architectures

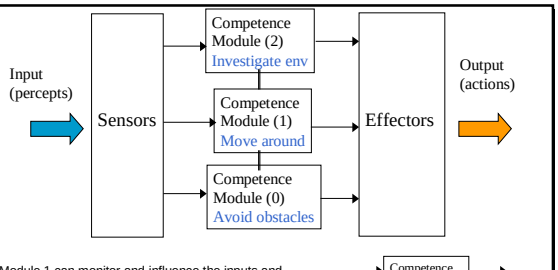
Subsumption architecture - Brooks, 1986

- (1) Decision making = {**Task Accomplishing Behaviours**}
  - Each behaviour = a function to perform an action
  - Brooks defines TAB as finite state machines
  - Many implementations: ***situation → action***
- (2) Many behaviours can fire simultaneously

Subsumption architecture

- A TAB is represented by a competence module (c.m.)
- Every c.m. is responsible for a clearly defined, but not particular complex task - concrete behavior
- The c.m. are operating in parallel
- Lower layers in the hierarchy have higher priority and are able to inhibit operations of higher layers
- c.m. at the lower end of the hierarchy - basic, primitive tasks;
- c.m. at higher levels - more complex patterns of behaviour and incorporate a subset of the tasks of the subordinate modules

➔ ***subsumtion architecture***

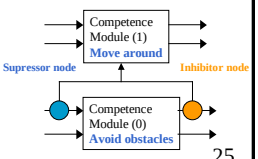


Module 1 can monitor and influence the inputs and outputs of Module 2

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

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

- Incorporating the functionality of a subordinated c.m. by a higher module is performed using **suppressors** (modify input signals) and **inhibitors** (inhibit output)



25

- More modules can be added:
  - Replenishing energy
  - Optimising paths
  - Making a map of territory
  - Pick up and put down objects

Behavior

(c, a) – pair of condition-action describing behavior

R = { (c, a) | c ⊆ P, a ∈ A} - set of behavior rules

∠ ⊆ R x R - binary inhibition relation on the set of behaviors, total ordering of R

function action(p: P)
var fired: P(R), selected: A
begin
fired = {(c, a) | (c, a) ∈ R and p ∈ c}
for each (c, a) ∈ fired **do**
if ¬ ∃ (c', a') ∈ fired such that (c', a') ∠ (c, a) **then return** a
return null
end

26

- Every **c.m.** is described using a subsumption language based on AFSM - Augmented Finite State Machines
- An AFSM initiates a response as soon as its input signal exceeds a specific threshold value.
- Every AFSM operates independently and asynchronously of other AFSMs and is in continuos competition with the other c.m. for the control of the agent - real distributed internal control
- 1990 - Brooks extends the architecture to cope with a large number of c.m. - Behavior Language

Other implementations of reactive architectures

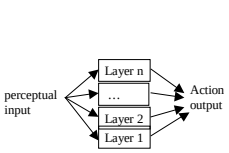
- Steels - indirect communication - takes into account the social feature of agents

- Advantages of reactive architectures
- Disadvantages

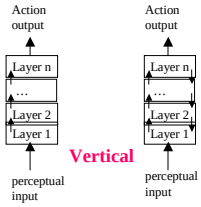
27

4. Layered agent architectures

- Combine reactive and pro-active behavior
- At least two layers, for each type of behavior
- **Horizontal layering** - i/o flows horizontally
- **Vertical layering** - i/o flows vertically



Horizontal



Vertical

28

TouringMachine

- **Horizontal layering - 3 activity producing layers,** each layer produces suggestions for actions to be performed
- **reactive layer** - set of situation-action rules, react to precepts from the environment
- **planning layer**
  - pro-active behavior
  - uses a library of plan skeletons called schemas
  - hierarchical structured plans refined in this layer
- **modeling layer**
  - represents the world, the agent and other agents
  - set up goals, predicts conflicts
  - goals are given to the planning layer to be achieved
- **Control subsystem**
  - centralized component, contains a set of control rules
  - the rules: suppress info from a lower layer to give control to a higher one
  - censor actions of layers, so as to control which layer will do the actions

29

InteRRaP

- Vertically layered two pass agent architecture
- Based on a BDI concept but concentrates on the dynamic control process of the agent

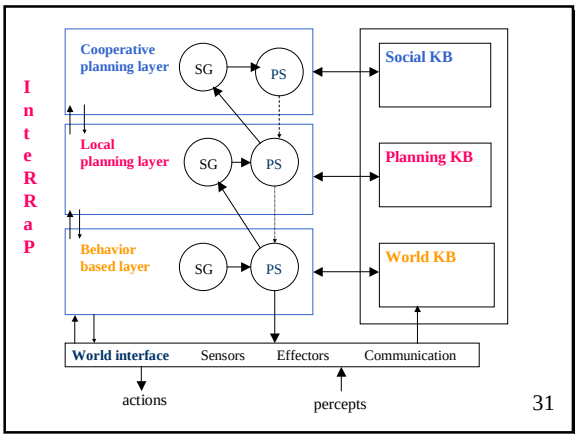
Design principles

- the three layered architecture describes the agent using various degrees of abstraction and complexity
- both the control process and the KBs are multi-layered
- the control process is bottom-up, that is a layer receives control over a process only when this exceeds the capabilities of the layer beyond
- every layer uses the operations primitives of the lower layer to achieve its goals

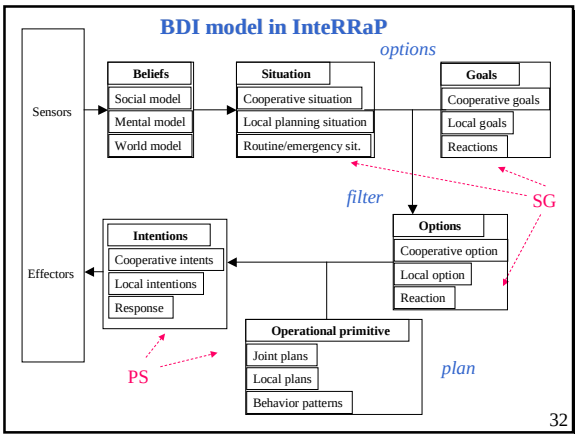
Every control layer consists of two modules:

- situation recognition / goal activation module (SG)
- planning / scheduling module (PS)

30



31



32

- Muller tested InteRRaP in a simulated loading area.
- A number of agents act as automatic fork-lifts that move in the loading area, remove and replace stock from various storage bays, and so compete with other agents for resources

33