A4M33MAS - Multiagent Systems
Intelligent agents architectures

Michal Pechoucek & Michal Jakob

Department of Computer Science
Czech Technical University in Prague

OTEVŘENÁ INFORMATIKA

In parts based on Michal Pechoucek: Multi-Agent Systems, Lecture course at State University of New York, University at Binghamton

Towards Architectures for IA

- Reactive Architectures
- Deliberative Architectures

Subsumption Architecture

best known reactive agents architecture developed by R. Brooks:

- Two key ideas:
  - situatedness and embodiment of intelligence: real intelligence is situated in the world it is not disembodied such as expert system or theorem prover
  - intelligence and emergence: intelligence arises from the agent x environment interaction, intelligence is not an isolated property
- Task accomplishing behavior: a function that maps the percept into action as there is no representation/reasoning the task accomplishing behaviors are implemented as rules:

$$\texttt{situation} \longrightarrow \texttt{action}$$

- the rules are fired simultaneously – the rules are structures into subsumption hierarchy: layers of different levels of abstraction of the behavior (where each layer inhibit the higher level layer)

Subsumption Architecture

01

- behavior:

$$\text{Beh} = \{\texttt{cond} \longrightarrow \texttt{act}, \text{ where } \texttt{cond} \subseteq \mathcal{P} \text{ and } \texttt{act} \subseteq \mathcal{A}c\}$$

- inhibition relation:
  - for rules in Beh, there is a binary relation – a strict total ordering on Beh (i.e. transitive, irreflexive and antisymetric)

$\texttt{fired} \longrightarrow \{\texttt{cond} \longrightarrow \texttt{act}, \text{ where } \texttt{cond} \longrightarrow \texttt{act} \in \mathcal{P} \text{ and } \texttt{act} \subseteq \mathcal{A}c\}$

  for each $\texttt{cond} \longrightarrow \texttt{act} \in \texttt{fired}$ do

    if $\nexists\ \texttt{cond}' \longrightarrow \texttt{act}' \in \texttt{fired}$ such that $\texttt{cond}' \longrightarrow \texttt{act}' \prec \texttt{cond} \longrightarrow \texttt{act}$

      then return act

    end-if

  end-for

return $null$

Subsumption Architecture

- Example of rock sample collecting robots (Steels), inspired by ants:
  - noncooperative agents

```
(cond (detect-an-obstacle #'change-direction)               r1
        ((and carrying-samples at-base) #'drop-samples)      r2
        ((and carrying-samples (not at-base)) #'travel-up)   r3
        ((detect-a-sample #'pick-up-sample)                  r4
        (t #move-randomly)                                   r5
    )
```

$$r1 \prec r2 \prec r3 \prec r4 \prec r5$$

Subsumption Architecture

- if samples in clusters, collaborative property is desirable
  - direct/indirect communication
- stigmergy: ant-inspired collaborative decision making

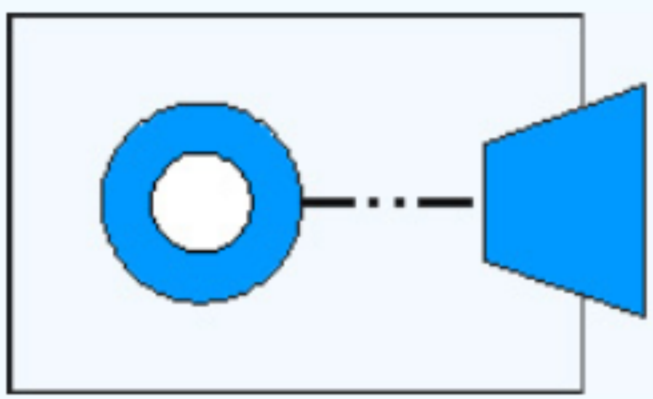
```
((and carrying-samples (not at-base))
       (and #drop-2-crumbs #'travel-up))                   r6
((detect-crambs (and #'pick-up-1-crumb #'travel-down)      r7
```

$$r1 \prec r2 \prec r6 \prec r4 \prec r7 \prec r5$$

Subsumption Architecture

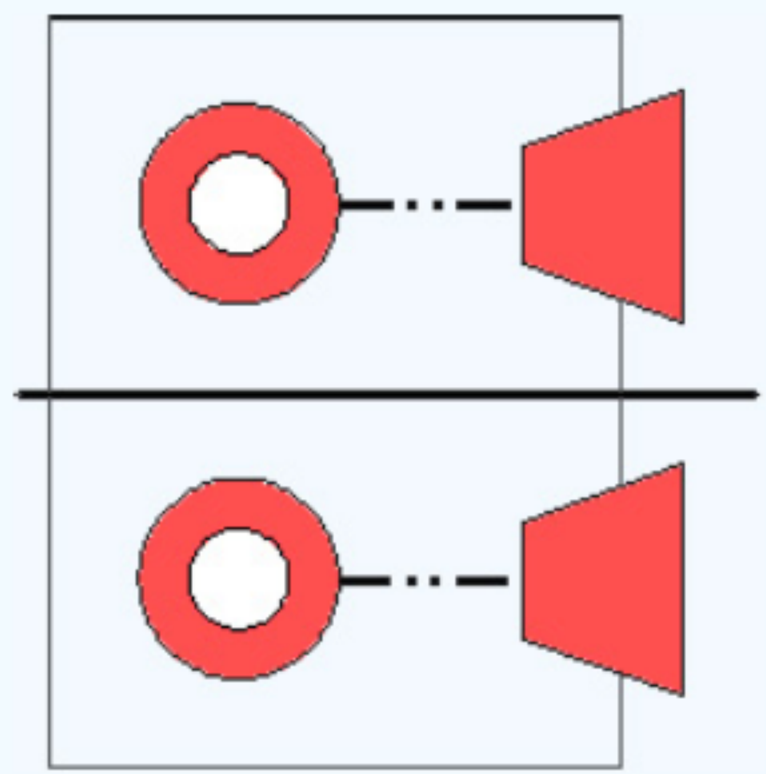
- S-agents

```
(cond
        (light-in-front-increasing #move-forward-faster)
)
```



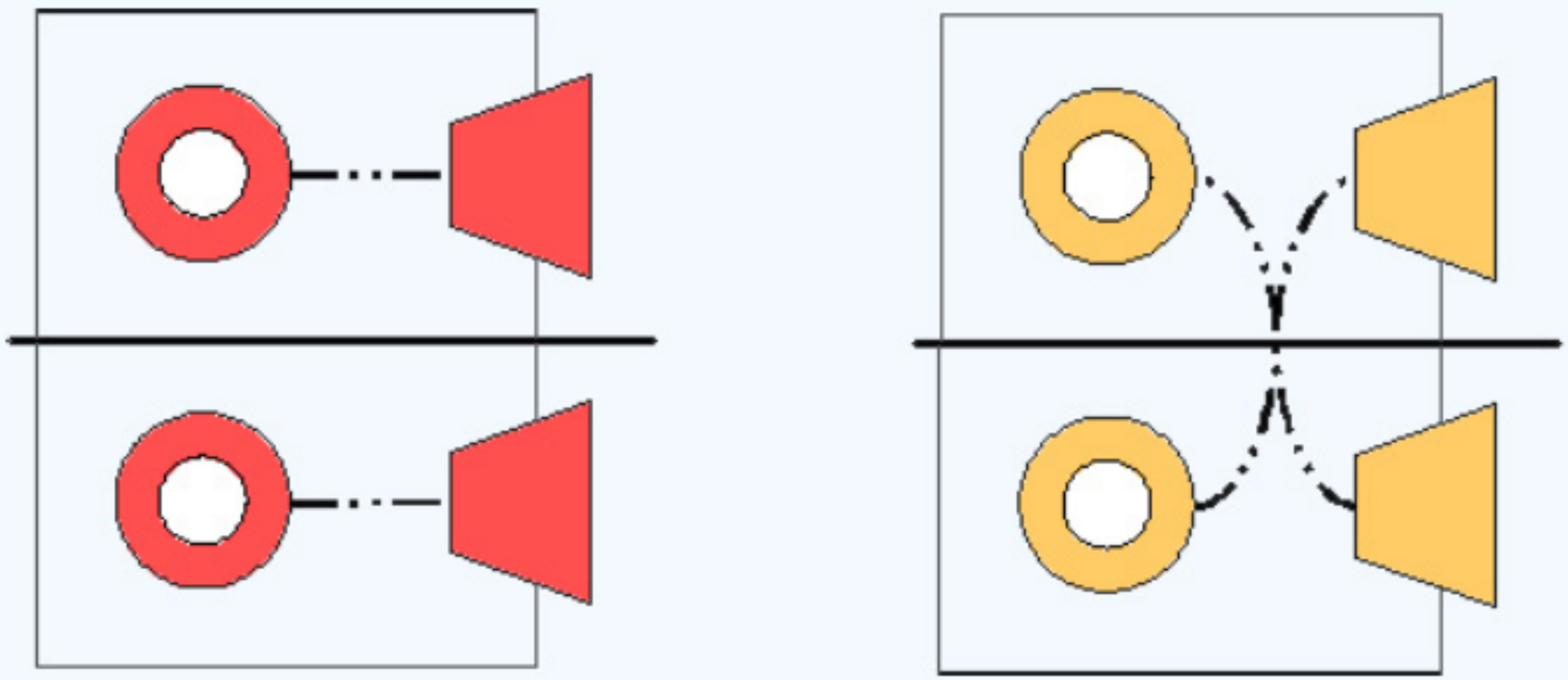
Subsumption Architecture

- S-agents: composition



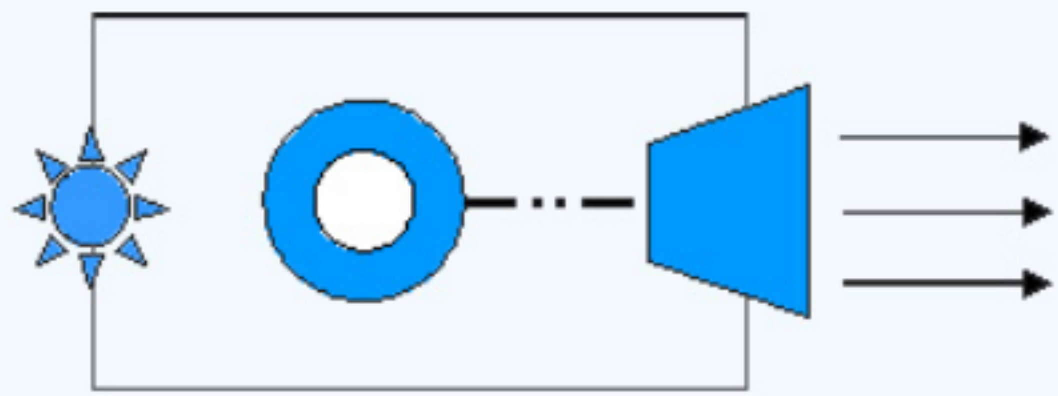
Subsumption Architecture

- S-agents: composition



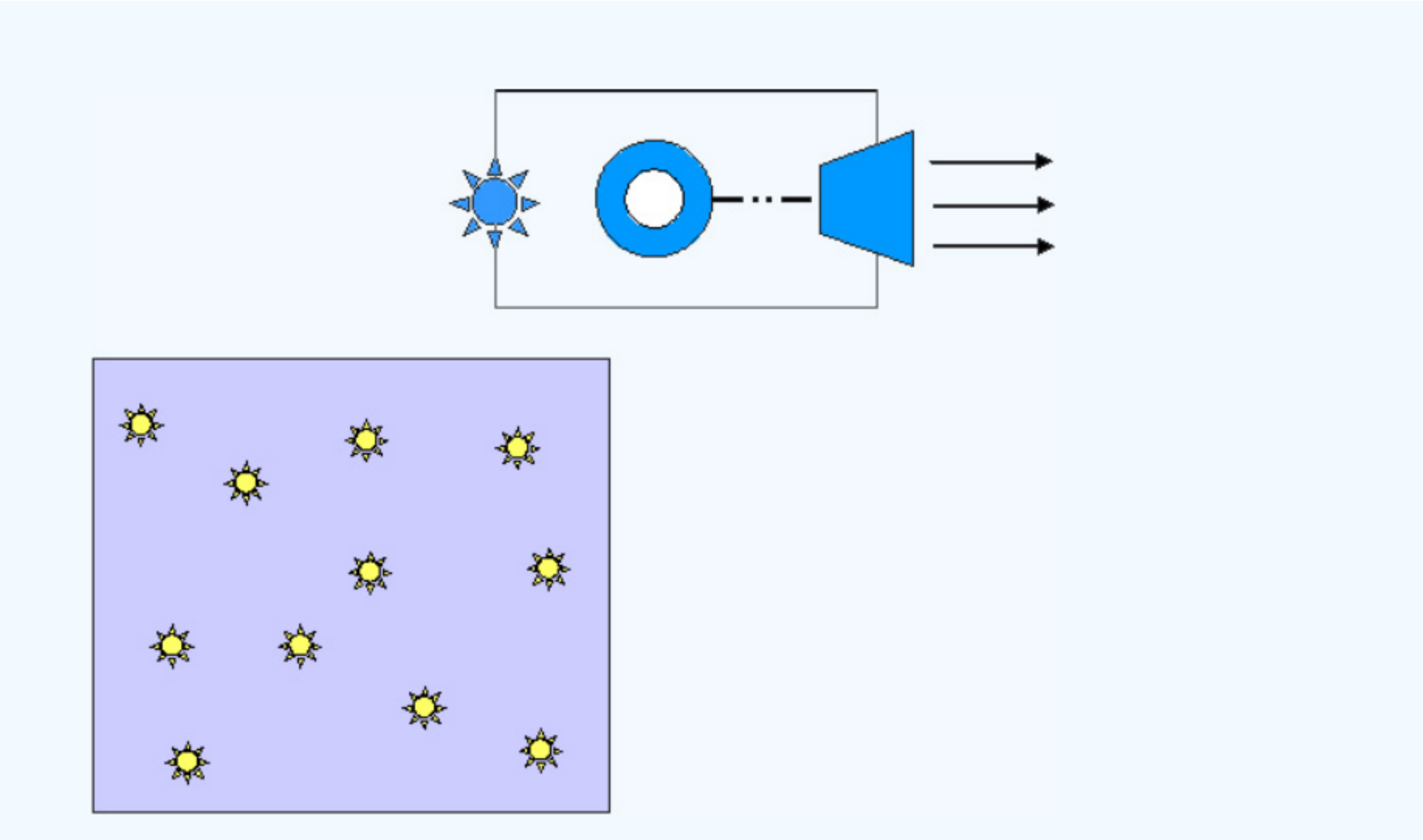
Subsumption Architecture

- S-agents: composition



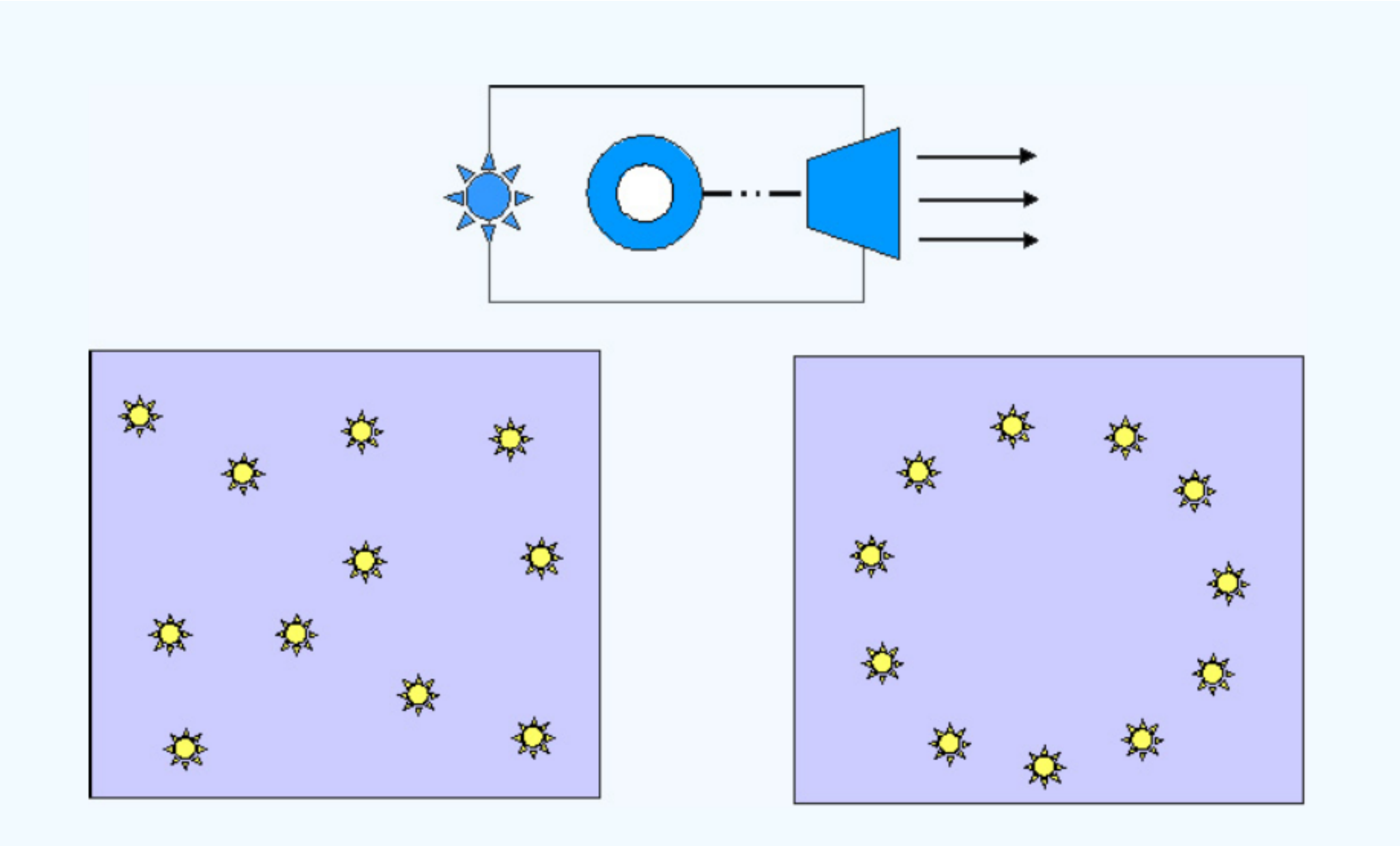
Subsumption Architecture

- S-agents: composition



Subsumption Architecture

- S-agents: composition



Deliberative architectures

1. Planning architectures
2. Deductive architectures
3. Models of practical reasoning

Planning Agents

Planning Agents

- Basic architecture:
  1. Plan(s_0, s_g, O) → π
  2. Execute (π) → sg

Planning Agents

01

- Basic architecture:
  1. Plan(s_0, s_g, O) → π
  2. Execute (π) → sg

- Reactive planning architecture:
  1. s_0 → s_n
  2. Plan(s_n, s_g, O) → π
  3. Read(Execute(*Head*(π))) → s_n
  4. if $s_n \neq s_g$ *GOTO* 2.

Planning Agents

- Basic architecture:
  1. Plan(s_0, s_g, O) → π
  2. Execute (π) → sg

- Reactive planning architecture:
  1. $s_0 = s_n$
  2. Plan(s_n, s_g, O) → π
  3. Read(Execute(*Head*(π))) → s_n
  4. if $s_n \neq s_g$ *GOTO* 2.

∗ *For Plan see to A4M36PAH or related*

Deductive Agents

01

- Models of deductive reasoning:

– constantly running a **reasoning loop** that is try to prove that there is an action that shall, or at least is allowed to be carried out – given agent's knowledge Δ, and its goal $\mathcal{G}$ we shall instantiate a variable α in a predicate $\mathtt{do}(\alpha)$ when proving $_{\Delta,\mathcal{G}} \vdash \exists\alpha\ \mathtt{do}(\alpha)$ – where $\mathtt{shall\text{-}be\text{-}done}(\alpha) \vee \mathtt{may\text{-}be\text{-}done}(\alpha) \Rightarrow \mathtt{do}(\alpha)$

– or prove what is the **best reaction** to a new piece of information $\mathcal{P}$ sensed from the environment $_{\Delta,\mathcal{G}} \vdash \exists\alpha\ \mathtt{react}(\mathcal{P},\alpha)$

Reasoning rules are here the rules of mathematical logic. The agent's knowledge Δ contain agents' model of the environment and its ($\mathtt{if\text{-}then}$) decision making rules.

Deductive Agents

01

- Implementation of deductive agents:

:: **Firstly,**

we shall find the proper logical language (e.g. FOPL) for representing agents model of its environment and than design a solid system that will represent the environmental objects that the agent may model, relation between them etc.

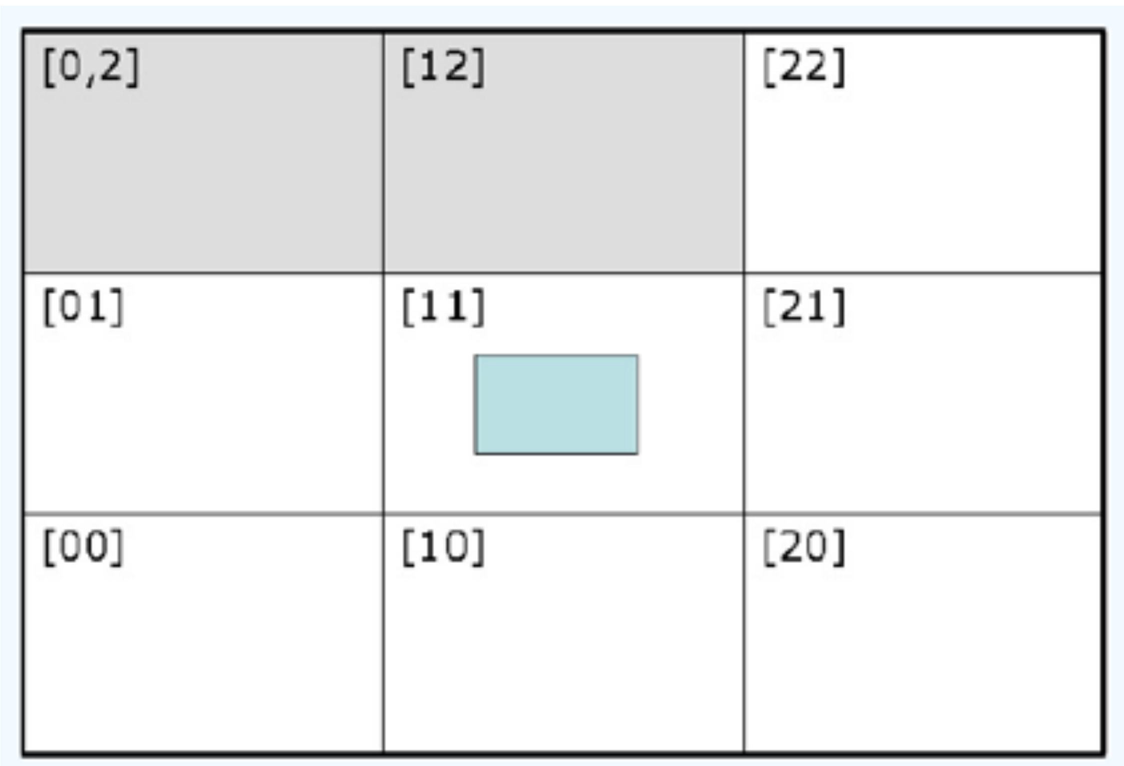
:: **Secondly,**

you have to design and implement/integrate an inference engine that will try compute (and instantiate) the logical consequence of the model in the $\mathtt{do}(\alpha)$ sense.

We will need a theorem proving system, e.g. based on resolution technique (implemented in Prolog, Otter, etc.)

Deductive Agents

- Example



We use 3 domain predicates in this exercise:

- $\texttt{In}(x, y)$ agent is at x, y
- $\texttt{Dirt}(x, y)$ there is dirt at x, y
- $\texttt{Facing}(d)$ the agent is facing direction d

Possible actions:

- $\texttt{Ac} = \{\texttt{turn}, \texttt{forward}, \texttt{suck}\}$

Deductive Agents

- Example

$$\texttt{In}(x,y) \wedge \texttt{dirt}(x,y) \rightarrow \texttt{do}(suck)$$

$$\texttt{In}(x,0) \wedge \texttt{Facing}(north) \wedge \neg\texttt{dirt}(x,0) \rightarrow \texttt{do}(forward)$$

$$\texttt{In}(x,0) \wedge \neg\texttt{Facing}(north) \wedge \neg\texttt{dirt}(x,0) \rightarrow \texttt{do}(turn) \wedge \texttt{do}(forward)$$

$$\texttt{In}(x,y) \wedge \neg\texttt{dirt}(x,y) \rightarrow \texttt{do}(forward)$$

$$\texttt{In}(x,2) \wedge \texttt{Facing}(south) \wedge \neg\texttt{dirt}(x,2) \rightarrow \texttt{do}(forward)$$

$$\texttt{In}(x,2) \wedge \neg\texttt{Facing}(south) \wedge \neg\texttt{dirt}(x,2) \rightarrow \texttt{do}(turn) \wedge \texttt{do}(forward)$$

$$\texttt{In}(2,2) \wedge \texttt{do}(finish)$$

Problems with Deductive Agents

:: calculative rationality (CR) requirements

– an agent comply with calculative rationality requirements provided:

$$\forall \Delta_{1,2} :_{\Delta_1} \vdash \mathtt{do}(\alpha_1) \wedge_{\Delta_2} \vdash \mathtt{do}(\alpha_2) \Rightarrow \mathtt{time}(\Delta_1 \leadsto \Delta_2) > \mathtt{time}(\Delta_1 \vdash \mathtt{do}(\alpha_1))$$

:: first order logic is not expressive enough

– we need mechanisms for expressing functions, effects and dynamics of actions, higher level modalities such as time, obligation, knowledge and agents mutual mental positions

– all this can be done in different logics (e.g. temporal, deontic, dynamic) but automated theorem proving in such systems is very, very complex and CR requirements are likely to fail

Problems with Deductive Agents

01

:: calculative rationality (CR) requirements

– an agent comply with calculative rationality requirements provided:

$$\forall \Delta_{1,2} :_{\Delta_1} \vdash \mathtt{do}(\alpha_1) \wedge_{\Delta_2} \vdash \mathtt{do}(\alpha_2) \Rightarrow \mathtt{time}(\Delta_1 \rightsquigarrow \Delta_2) > \mathtt{time}(\Delta_1 \vdash \mathtt{do}(\alpha_1))$$

:: first order logic is not expressive enough

– we need mechanisms for expressing functions, effects and dynamics of actions, higher level modalities such as time, obligation, knowledge and agents mutual mental positions

– all this can be done in different logics (e.g. temporal, deontic, dynamic) but automated theorem proving in such systems is very, very complex and CR requirements are likely to fail

* *but we have modal logic*