# Agent Architectures

# Agent Architectures

1. What is Agent Architecture?
2. Abstract agent architectures.
3. Deliberative Architectures
4. Reactive Architectures
5. Hybrid Architectures
6. Discussion

# References - Curriculum

- Wooldridge: ”Introduction to MAS”,
  - Chapters 2,3,4,5

# What is Agent Architecture? 1

- We want to build agents that enjoy the properties of autonomy, reactiveness, proactiveness and social ability.
- How do we construct computer systems that satisfy properties specified by agent theorists?

➢ the area of agent architectures.

# History of Agent Architecture

- Originally (1956-1985), all agents designed within AI were **symbolic reasoning** agents.
  - Agents use explicit reasoning in order to decide what to do.
- Problems with symbolic reasoning led to a reaction against this – the so-called **reactive agents** movement, 1985 onwards.
- From 1990-present, a number of alternatives proposed: **hybrid architectures**, which attempt to combine the best of reasoning and reactive architectures.

# Abstract Architectures for Agents 1

- Assume the environment may be in any of a finite set *E* of discrete, instantaneous states:

  $E = \{e_1, e_2, \ldots\}$

- Agents are assumed to have a repertoire of possible actions available to them, which transform the state of the environment.

  $A = \{a_1, a_2, \ldots\}$

# Abstract Architectures 2

- The interaction of agents and environment can be represented as a history (run):

$$r: \; e_0 \xrightarrow{a_0} e_1 \xrightarrow{a_1} e_2 \xrightarrow{a_2} \ldots\ldots \xrightarrow{a_{u-1}} e_u \xrightarrow{a_u} \ldots$$

- Two important points:

1. Environments are assumed to be **history dependent**, i.e. the next state of an environment may not solely determined by the action performed by the agent and the current state of the environment.
2. Possible **non-determinism** in the environment, i.e. there is uncertainty about the result of performing an action in some state.

# Abstract Architectures 3

$$r:\ e_0 \xrightarrow{a_0} e_1 \xrightarrow{a_1} e_2 \xrightarrow{a_2} \ldots\ldots \xrightarrow{a_{u-1}} e_u \xrightarrow{a_u} \ldots$$

- The (non-deterministic) behaviour of an environment can be modeled as a state transformer function:

$$\tau: r(\text{ended with } a) \rightarrow \rho(E),$$

which takes a run and maps it to a set of environment states $\tau(r)$ – those that could result from performing action $a$ in state $e$.

  - **non-determistic** if $\rho(E) = \{e_x, e_y\}$
  - **determinstic** if $\rho(E) = \{e_x\}$

# Abstract Architectures

$$r:\ e_0 \xrightarrow{a_0} e_1 \xrightarrow{a_1} e_2 \xrightarrow{a_2} \ldots\ldots \xrightarrow{a_{u-1}} e_u \xrightarrow{a_u} \ldots$$

- Agents can be viewed as a function:

  Ag: *r(ended with e)* → *A*

- The characteristic behavior of an *agent* in an *environment* is the set of all runs.

*R(Ag, Env)* - the set of all histories/runs of *agent* in *environment (Env={E, e0, τ})*

- If some property holds of all these histories, this property can be regarded as an invariant property of the agent in the environment.

# Abstract Architectures

Two agents are behaviorally equivalent wrt environment *Env* iff *R(Ag1,Env)=R(Ag2,Env)*

Two agents are simply behaviorally equivalent iff they are behaviorally equivalent wrt to all environments

# Abstract Architectures
## Purely Reactive Agents

- Certain types of agents decide what to do without reference to their history.
  - Decision making based entirely on the present.
- They are agents without internal states.
- Behaviour of purely reactive agents can be represented as a function:

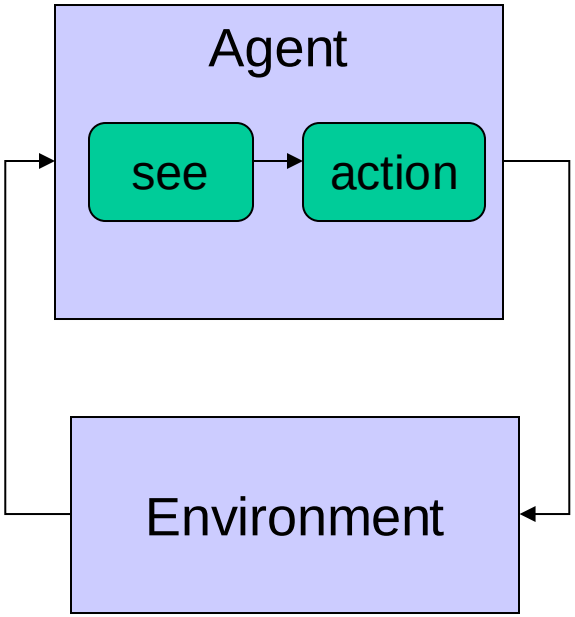
  Ag: $E \rightarrow A$

- A thermostat is a purely reactive agent:

$$\text{action}(s) = \begin{cases} \text{Off} & \text{if e = temperature OK} \\ \text{On} & \text{otherwise} \end{cases}$$

# Abstract Architectures

## Perception




- The **see** function is the agent's ability to observe its environment.
- The **action** function represents an agent's decision making process.
- The output of a see function is a **percept**:
  see: *E* → *Per*
  which maps environment states to percepts, and action is now a function:
  action: *Per* → *A*
  which maps sequences of percepts to actions.

# Abstract Architectures
## Example

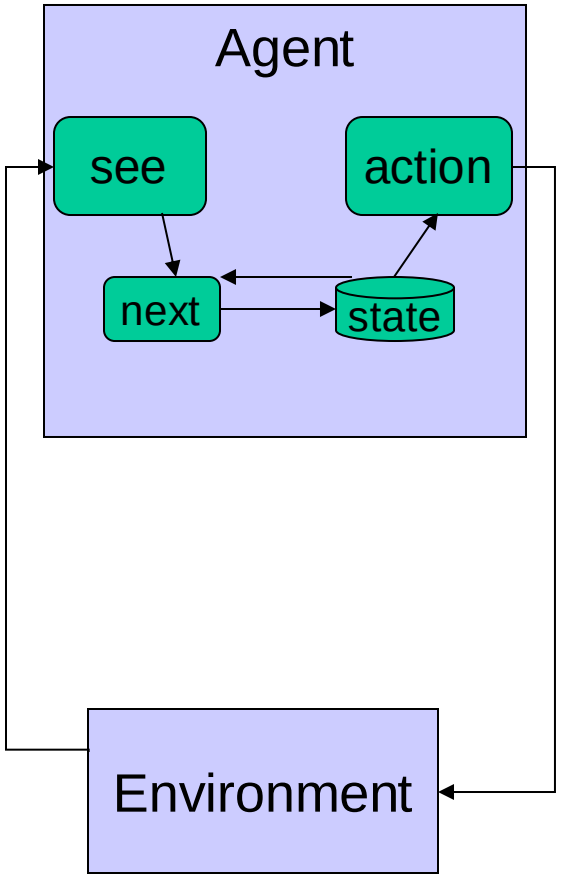
- Suppose we have 2 environment states $e_1 \in E$ and $e_2 \in E$ such that:
  - $e_1 \neq e_2$, but see($e_1$) = see($e_2$)
  - $e_1$ and $e_2$, are indistinguishable!
- Example: let $x$="the room temperature is OK" and $y$="Today is Wednesday".
- The set of environment states: $E = \{\underbrace{\{\neg x, \neg y\}}_{e_1}, \underbrace{\{\neg x, y\}}_{e_2}, \underbrace{\{x, \neg y\}}_{e_3}, \underbrace{\{x, y\}}_{e_4}\}$

$$see(e) = \begin{cases} p_1, & \text{if } e=e_1 \text{ or } e=e_2 \\ p_2, & \text{if } e=e_3 \text{ or } e=e_4 \end{cases}$$

- An agent has a perfect perception if it can distinguish every environment state.

# Abstract Architectures

## Agents with State 1




- The agents have some internal data structure, which is typically used to record information about the environment state and history.
- Let *I* be the set of all **internal states** of the agent.
- Perception function is unchanged: see: *E* → *Per*
- The action-selection function action is now defined as a mapping from internal states to actions.

  action: *I* → *A*
- An new function **next** maps an internal state and percept to an internal state

  next: *I* x *Per* → *I*

# Abstract Architectures
## Agents with State 2

- Behaviour of an agent:

1. Starts in initial state $i=i_0$.
2. Observes its environment state *e* and generates a percept *see*(*e*).
3. The internal state of the agent is then updated via the next function,

   *next(i , see(e))*
4. The action selected by the agent is then

   Ag = *action(next(i , see(e)))*
5. The action is performed and the agent enters a new cycle (i.e. goes to 2).

# Practical Reasoning 1

- Distinction between practical reasoning and theoretical reasoning:
  - Theoretical reasoning is directed towards beliefs.
  - Practical reasoning is directed towards actions
- **Practical reasoning directed towards action** – the process of figuring out what to do:
  - Practical reasoning is a matter of weighing conflicting considerations for and against competing options, where the relevant considerations are provided by what the agent desires/values/cares about and what the agent believes. (Bratman)

# Practical Reasoning 2

- Human practical reasoning consists of 2 activities:

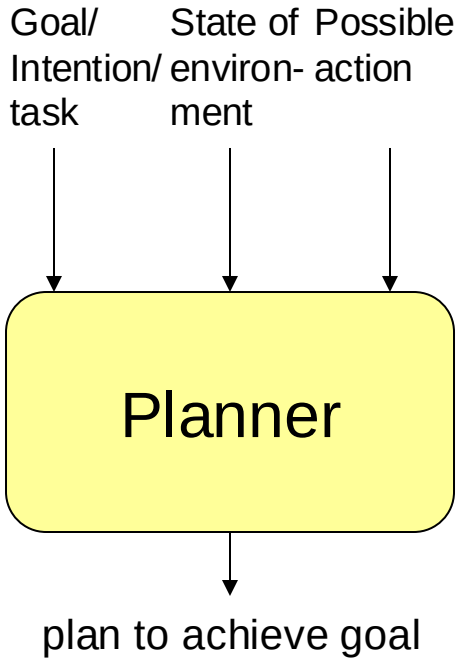
  1. **Deliberation**: deciding what state of affairs we want to achieve.
  2. **Means-end reasoning**: deciding how to achieve these states of affairs.

- The output of deliberations are intentions.
- The output of **means-end reasoning** are plans.

# What is Means-end Reasoning?

- Means-end reasoning is a process of deciding how to achieve an end using available means (it is also known in AI as planning)
- Basic idea is to give an agent the following and have it generate a plan to achieve the goal:
  - Representation of goal/intention to achieve.
  - Representation of actions it can perform.
  - Representation of the environment.

➢ **Automatic programming**



# Deliberation

How does an agent deliberate?

- **Option generation**: trying to understand what the available options are
  - Agent generates a set of alternatives (goals/desires)
- **Filtering**: choosing between options and committing to one.
  - Agent chooses between competing alternatives and commits to achieving them.

➢ Chosen options are then intentions.

# Symbolic Reasoning Agents (Deliberative Agent)

- The classical approach to building agents is to view them as a particular type of knowledge-based system and bring all associated methodologies of such systems.
- The paradigm is known as **symbolic AI**.
- We define a **deliberative agent** or agent architecture to be one that:
  - Contains an explicitly represented, symbolic model of the world.
  - Makes decisions (e.g. about what actions to perform) via symbolic reasoning.

# Deliberative Architectures

- Early systems were planning systems, e.g. STRIPS
  - Takes a symbolic description of the world, a desired goal state and a set of action descriptions (pre and post-conditions are associated)
  - Find a sequence of actions that will achieve the goal (matching post-conditions and goals)
- Very simple planning algorithms
- Very inefficient planning

# Symbolic Reasoning Agents

- Two key problems:

1. **Transduction**:

   Translating the real world into accurate, adequate symbolic description, in time for that description to be useful. (…vision, speech understanding, learning)

2. **Representation/reasoning**:

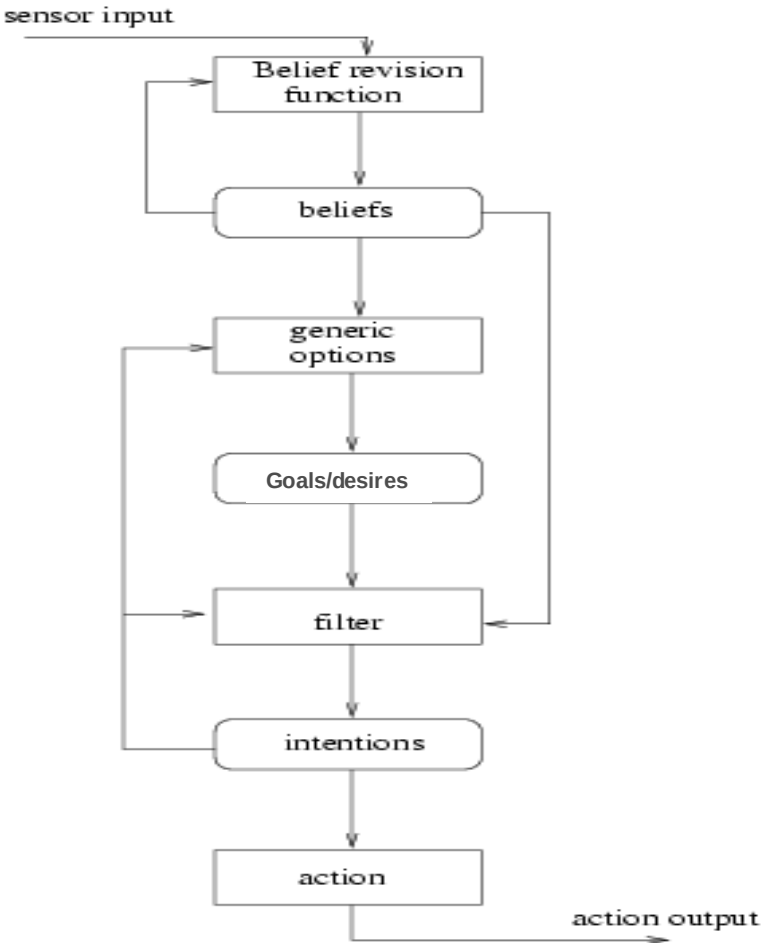
   How to symbolically represent information about complex real-world entities and processes and how to get agents to reason with this information in time for the results to be useful. (…knowledge representation, automated reasoning, automatic planning)

# Deliberative Architectures for Agents
## Overview

1. BDI – Belief, Desire, Intentions
2. PRS – Procedural Reasoning Systems
3. IRMA – Intelligent Resource-bounded Machine Architecture

# Deliberative Architectures

## Belief Desire Intention (BDI) Architectures



# Deliberative Architectures

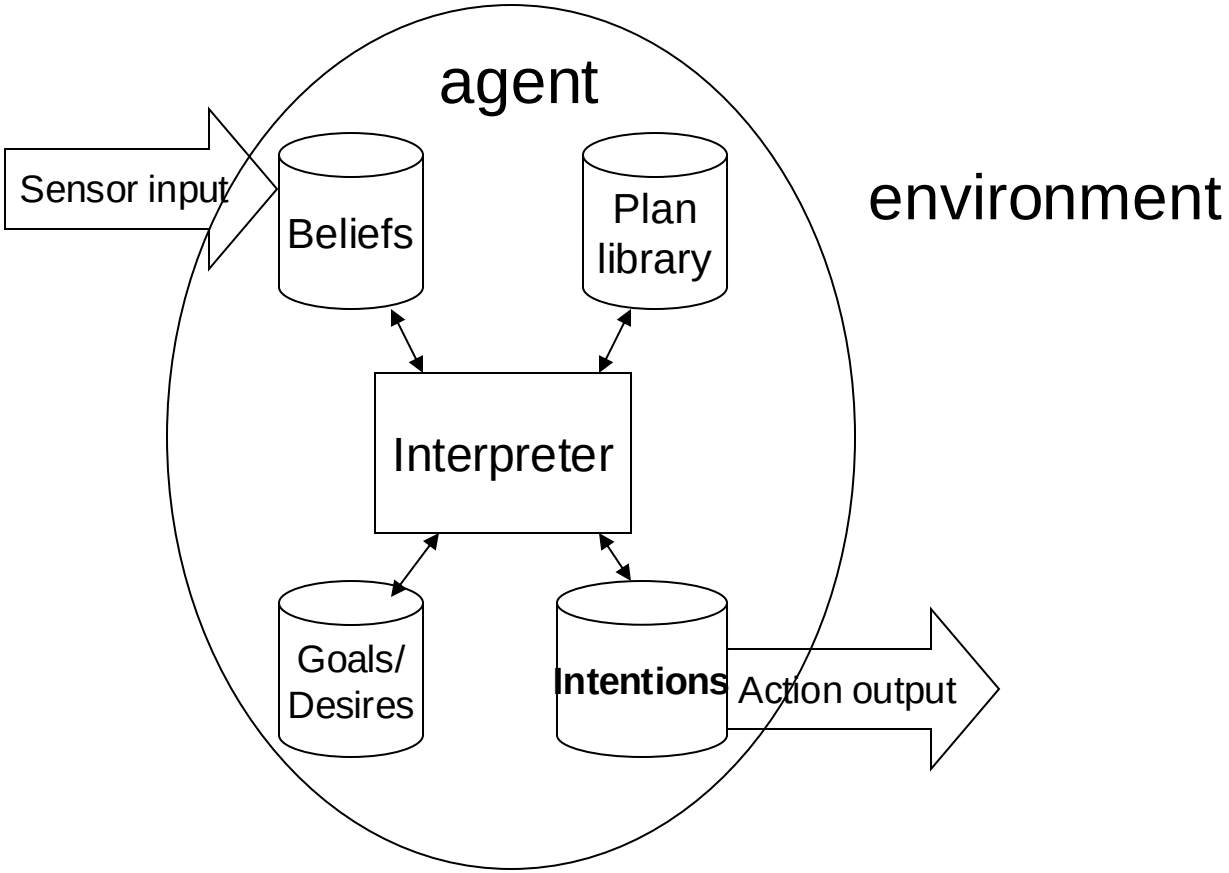
## BDI architectures

```
function action(p : E) : A
begin
   B := brf(B,p)
   D := options(B,I)
   I := filter(B,D,I)
   return execute(I)
end function action
```

# Deliberative Architectures

## PRS

- PRS (Procedural Reasoning Systems)



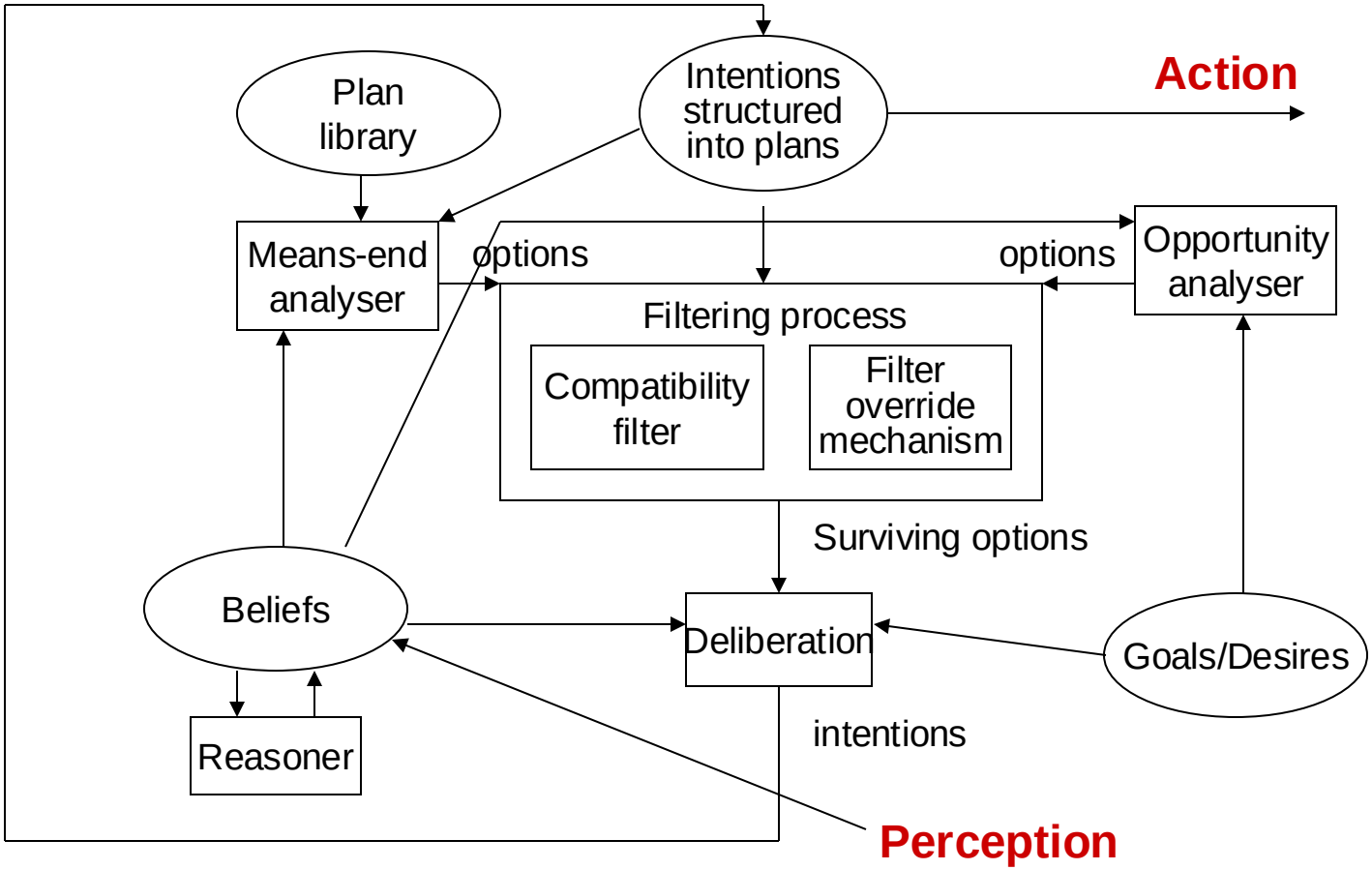
# Deliberative Architectures
## PRS

- In PRS, each agent is equipped with a plan library, representing the agent's procedural knowledge: knowledge about the mechanisms that can be used by the agent in order to realize its intentions.
- The options available to an agent are directly determined by the plans an agent has: an agent with no plans has no options.
- PRS agents have explicit representations of beliefs, desires (goals) and intentions.

# Deliberative Architectures

**BDI Architectures**:

IRMA (Intelligent Resource Bounded Machine Architecture)



# Deliberative Architectures
## IRMA

- IRMA has 4 key symbolic data structures:
  1. A plan library
  2. Beliefs: information available to the agent.
  3. Goals/Desires: those things that the agent would like to make true, (e.g. tasks that the agent has been allocated.)
  4. Intentions: goals/desires that the agent has chosen and committed to.

# Deliberative Architectures
## IRMA

- Additionally, the archirecture has:
  - A reasoner: for reasoning about the world, an inference engine.
  - A means-end analyser: determines which plans might be used to achieve the intentions.
  - An opportunity analyser: monitors the environment, and as a result of changes, generates new options.
  - A filtering process: determines which options are compatible with current intentions.
  - A deliberation process: responsible for deciding upon the ‘best’ intentions to adopt.

# Reactive Architectures
## Overview

1. Brook's subsumption Architecture
2. Steel's Mars Explorer

# Reactive Architectures

- There are many unsolved (perhaps insoluble) problems associated with symbolic AI, which has led researchers to develop reactive architectures.
- They use many different techniques.
- Most vocal critic: Rodney Brooks.

# Reactive Architectures

## Brooks

- Brooks has put forward 3 theses:

1. Intelligent behavior can be generated without explicit representations of the kind that symbolic AI processes.
2. Intelligent behavior can be generated without explicit reasoning of the kind that symbolic AI processes.
3. Intelligence is an emergent property of certain complex systems.

# Reactive Architectures

## Brooks

- 2 key areas that have informed his research:

1. **Situatedness and embodiment**: “real” intelligence is situated in the world, not in disembodied systems such as theorem provers and expert systems.
2. **Intelligence and emergence**: “intelligent” behaviour arises as a result of an agent’s interaction with its environment. Also, intelligence is “in the eye of the beholder”; it is not an innate, isolated property.

# Reactive Architectures

- Reactivity is a behavior-based model of activity as opposed to the symbol manipulation model used in planning.
  - A **stimulus-response** model.
- Situated rules
  - If <perceived situation> then <specifications>
- Reactive agents are **situated**: they do not take past events into account and cannot foresee the future.
- Do not have to revise their world model when perturbations change the world in unexpected ways.
- Considered as very flexible and adaptive because they can manage their resource abilities in unpredictable worlds.

# Reactive Architectures

- Can describe very simple behavior, but hardly applicable for more complex actions.
- Cannot plan ahead.
- Actions come only from perceptions.
- Assumes mutually exclusive rules and no rule conflicts.
- Reactive agents are difficult to predict

# Reactive Architectures

Situated rules

if <perceived situation>

then <specifications>

Rule take-hammer

if I don't carry anything and I am not at the workshop location

then take a hammer

# Reactive Architectures

## Situated automata (Rosenchein & Kälbling )

- agent is specified in declarative terms of two components: perception and action
- the logic used to specify an agent is essentially modal logic of beliefs
- perception is specified by three components:
  - semantics of agent's input: "whenever bit 1 is on, it is raining"
  - a set of static facts: "whenever it is raining, the ground is wet"
  - a specification of state transitions: "if the ground is wet, it stays wet until the sun comes out"
- action is specified by semantics of output: "if this bit is on, the ground is wet"
- the compiler syntheses a circuit whose output will have the correct semantics
- the generated circuit does not represent or manipulate symbolic expressions
- all symbolic manipulation is done at compile time

# Reactive Architectures

## Brooks, Subsumption Architecture

- To illustrate his ideas, Brooks built some robots based on his subsumption architecture.
- A **subsumption architecture** is a hierarchy of task-accomplishing behaviours.
- Each behaviour is a rather simple rule-like structure.
- Each behaviour ‘competes’ with others to exercise control over the others.

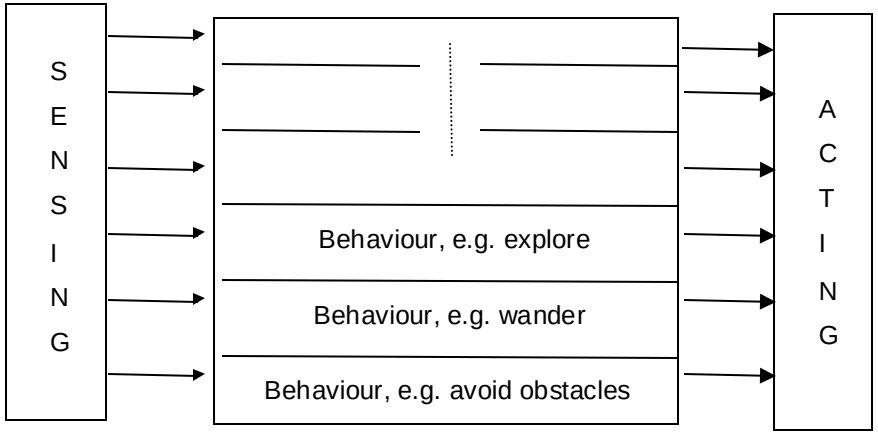
# Reactive Architectures
## Brooks, Subsumption Architecture

- it is made of a set of modules, represented as augmented finite state machines (AFSM)
- AFSM is triggered if its input signal exceeds some threshold
- modules are linked through master/slave relationship of inhibition
- modules are grouped and placed into layers which work asynchronously
- modules in a lower level can inhibit those in higher layers

# Reactive Architectures

## Brooks, Subsumption Architecture

- Lower layers represent more primitive kinds of behaviour (such as avoiding obstacles), and have precedence over layers further up the hierarchy.



# Reactive Architectures

## Steel, Mars Explorer System

- Uses subsumption architecture.
- Achieves near-optimal cooperative performance in simulated “rock gathering on Mars” domain.
- The **objective** is to explore  a distant planet, and in particular, to collect samples of a precious rock. The location of the samples is not known in advance, but it is known that they tend to be clustered.

# Reactive Architectures

## Steel, Mars Explorer System

- For individual (non-cooperative agents), the lowest level behaviour, (and hence the behaviour with the highest priority) is obstacle avoidance:

  *if* detect an obstacle *then* change direction. (1)

- Any samples carried by agents are dropped back at the mother-ship:

  *if* carrying samples *and* at the base *then* drop samples. (2)

- Agents carrying samples will return to the mother-ship:

  *if* carrying samples *and not* at the base *then* travel up gradient. (3)

# Reactive Architectures

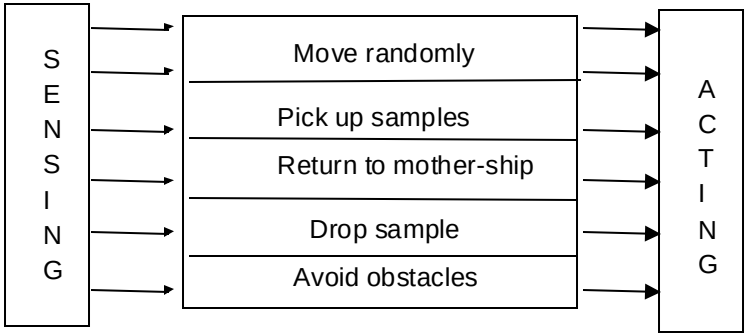
## Steel, Mars Explorer System

- Agents will collect samples they find:

  *if* detect a sample *then* pick sample up. (4)

- An agent with “nothing better to do” will explore randomly:

  *if* true *then* move randomly. (5)



# Reactive Architectures

function action(p : P) : A
begin
    fired:={$(c_i,a)$ | $(c_i,a) \in R$ and $p \in c_i$}
    for each $(c_i,a) \in$ fired do
        if $\neg(\exists(c_{i-1},a') \in$ fired such that $(c_{i-1},a') < (c_i,a)$) then
            return a
        end-if
    end-for
    return null
end function action

(c,a) - condition-action pair
R - set of condition-action pairs
$c_1 < c_2$ is read as "$c_1$ is lower in hierarchy than $c_2$"

# Hybrid Architectures
## Overview

1. Touring Machine
2. RoboSwarm
3. InteRRaP

# Hybrid Architectures

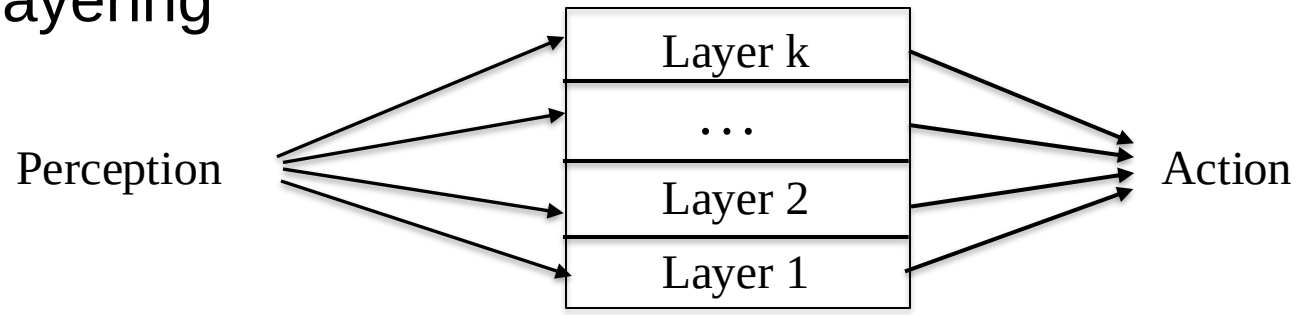
- Many researchers have argued that neither a completely deliberative nor completely reactive approach is suitable for building agents.
- They have suggested using **hybrid systems**, which marry deliberative and reactive systems.
- Approach: build an agent out of 2 subsystems:
  1. A deliberative one: containing a symbolic world model which develops plans and makes decisions in the way proposed by symbolic AI.
  2. A reactive one which is capable of reacting to events without complex reasoning.

# Hybrid Architectures

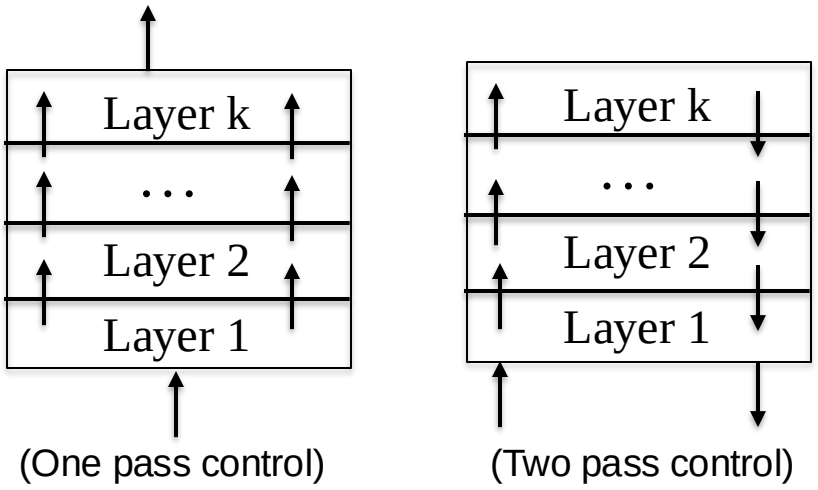
- A key problem in such architectures is what kind of control framework to embed the agent's subsystems in, to manage the interactions between the layers.
- **Horizontal layering**:
  - Layers are each directly connected to the sensory input and action output.
- **Vertical layering**:
  - Sensory input and action output are each dealt with by at most one layer.

# Hybrid Architectures
## Layered Architectures

Horizontal layering

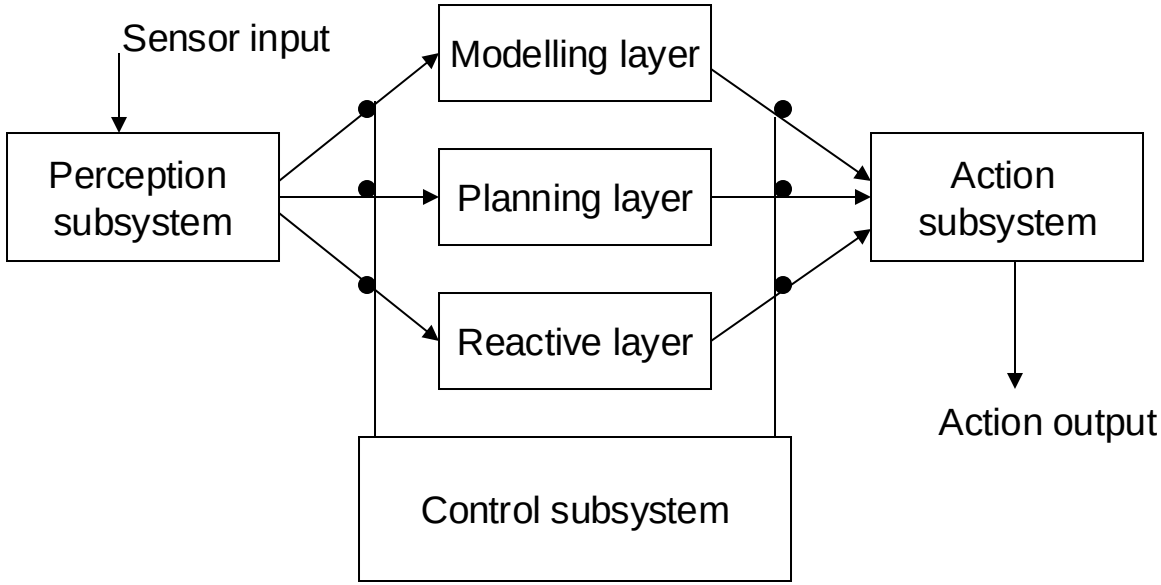



Vertical layering



# Hybrid Architectures
## Ferguson, Touring Machines

- The **Touring Machines** architecture consists of perception and action subsystems, which interface directly with the agent's environment, and control layers embedded in a control framework, which mediates between the layers.



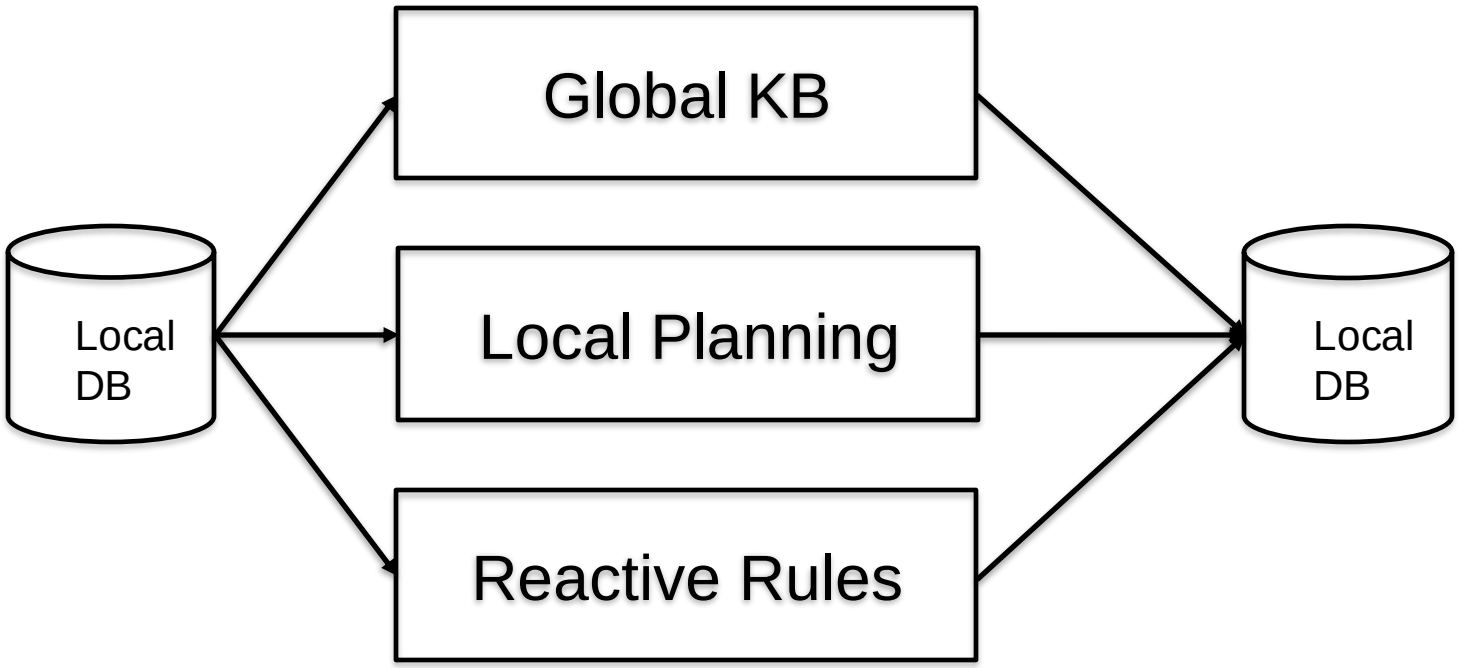
# Hybrid Architectures
## Ferguson, Touring Machines

- **Reactive layer**: immediate response.
- **Planning layer**: “day-to-day” running under normal circumstances.
- **Modelling layer**: predicts conflicts and generate goals to be achieved in order to solve these conflicts.
- **Control subsystem**: decided which of the layers should take control over the agent.

❖ Different from Brook’s subsumption architecture is that layers may contain explicit representations (e.g. beliefs, desires, intentions).

# ROBOSWARM project

# ROBOSWARM Architecture



# ROBOSWARM
# Employing Global Knowledge Base

# ROBOSWARM
# Task Allocation and Composition

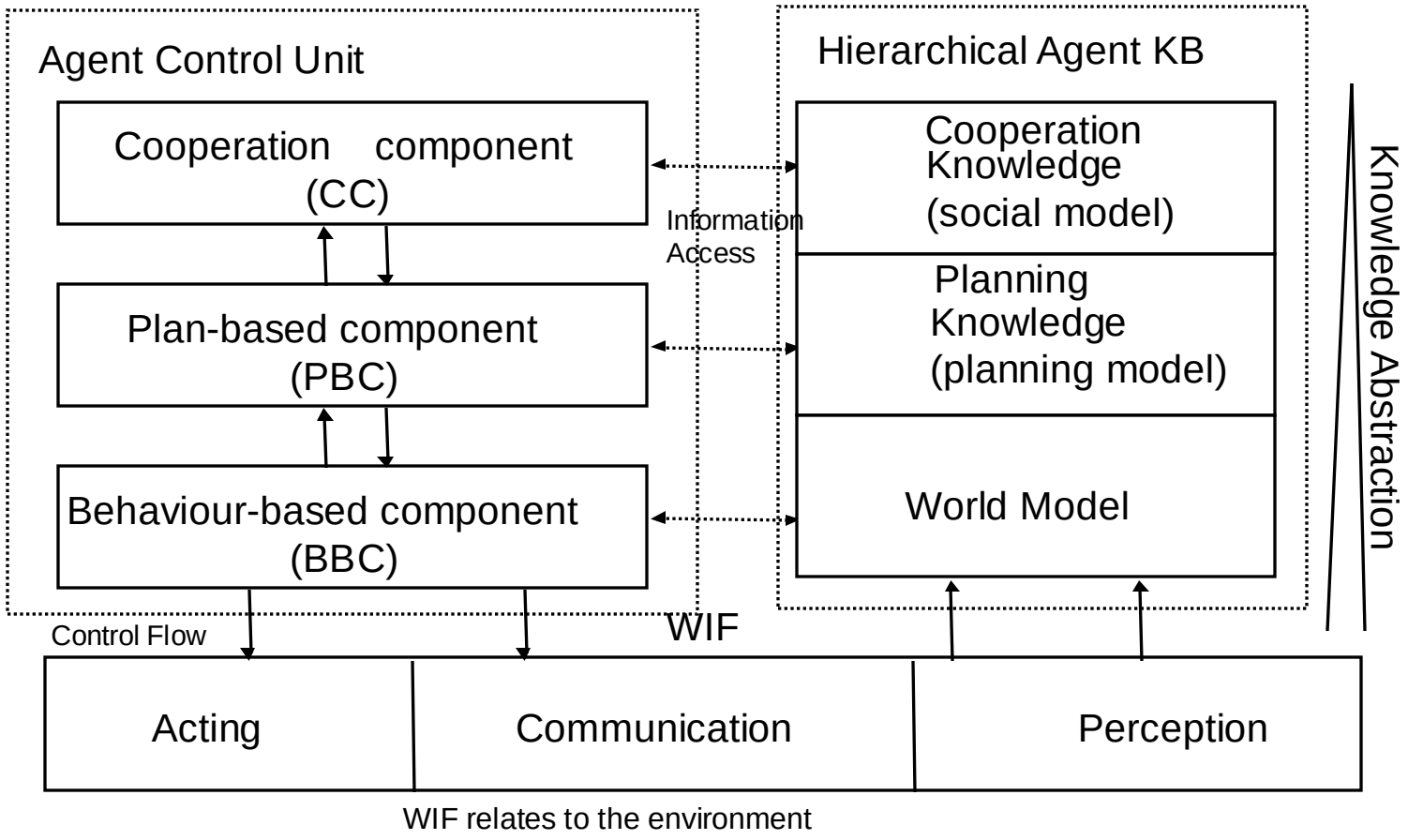
# Hybrid Architectures

## Muller et. al., InteRRaP

- Vertically layered architecture.
- Layers, divided into **knowledge bases** and **control components**.
- Lowest layer represents reactive components and highest layers deliberative components.
- Control is both data and goal driven.
  1. Reactive
  2. Local planning
  3. Cooperative planning

# Hybrid Architectures

## Muller et. al., InteRRaP



# Hybrid Architectures

## Muller et. al., InteRRaP

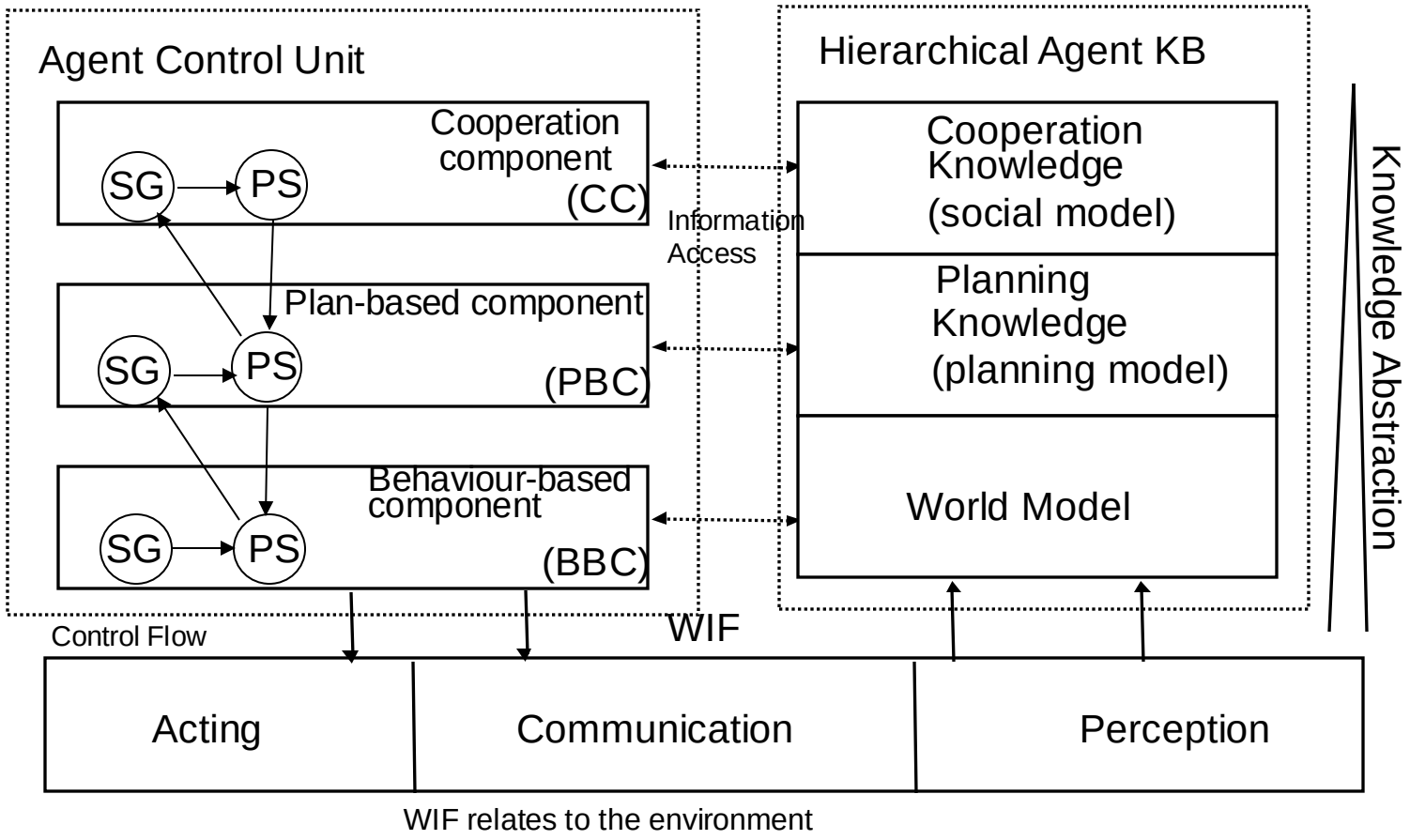
| Control Component | Corresponding Knowledge base | Function |
|---|---|---|
| Cooperation | Cooperation knowledge | Generate joint plans that satisfy the goals of a number of agents, in response to request from the plan-based component. |
| Plan-based | Planning knowledge + plan library | Generate single-agent plans to requests from the behaviour-based component. |
| Behaviour-based | World Model | Implement and control the basic reactive capability of the agent. (Call on the world interface or a higher-level layer to generate a plan). |
| World Interface | World Model | Manages the interface between the agent and its environment. |

# Hybrid Architectures
## (Muller et. al., InteRRaP)

- B - belief, G - goal, I - intention, P - set of perceived propositions
- BR(B,P) = B belief revision and knowledge abstraction function
- SG(B,G) = G' situation recognition and goal activation function
- PS(B,G,I) = I' planning and scheduling function

| Layer<br>Function | BBC | PBC | CC |
|---|---|---|---|
| BR | Generation and revision of beliefs (world model) | Abstraction of local beliefs (mental model) | Maintaining models of other agents (social model) |
| SG | Activation of reactor patterns | Recognition of situations requiring local planning | Recognition of situations requiring cooperative planning |
| PS | Reactor: direct link from situations to action sequences | Modifying local intentions; local planning | Modifying joint intentions; cooperative planning |

# Hybrid Architectures (Muller et. al., InteRRaP)



# Discussion/Summary 1

- Deliberative approach is currently the dominant approach in (D)AI:
  - Technology is familiar
  - Clear methodology
  - Lots of nice theory
  - ❖ But tends not to work well in highly dynamic domains.

# Discussion/Summary 2

- Reactive architectures are currently the most practical approach:
  - No technology consensus
  - No methodology
  - Only isolated islands of theory
  - Only simple behaviour
  - ❖ Works best in unknown environments.

# Discussion/Summary 3

- Hybrid architectures are a major force in current work on agents:
  - A pragmatic solution
  - But, an ad hoc one!
  - Still no clear consensus, but a lot of similarity.
  - No real methodology
  - No real theory.