LECTURE 2: AGENT ARCHITECTURES

Artificial Intelligence II – Multi-Agent Systems

Introduction to Multi-Agent Systems

URV, Winter - Spring 2010

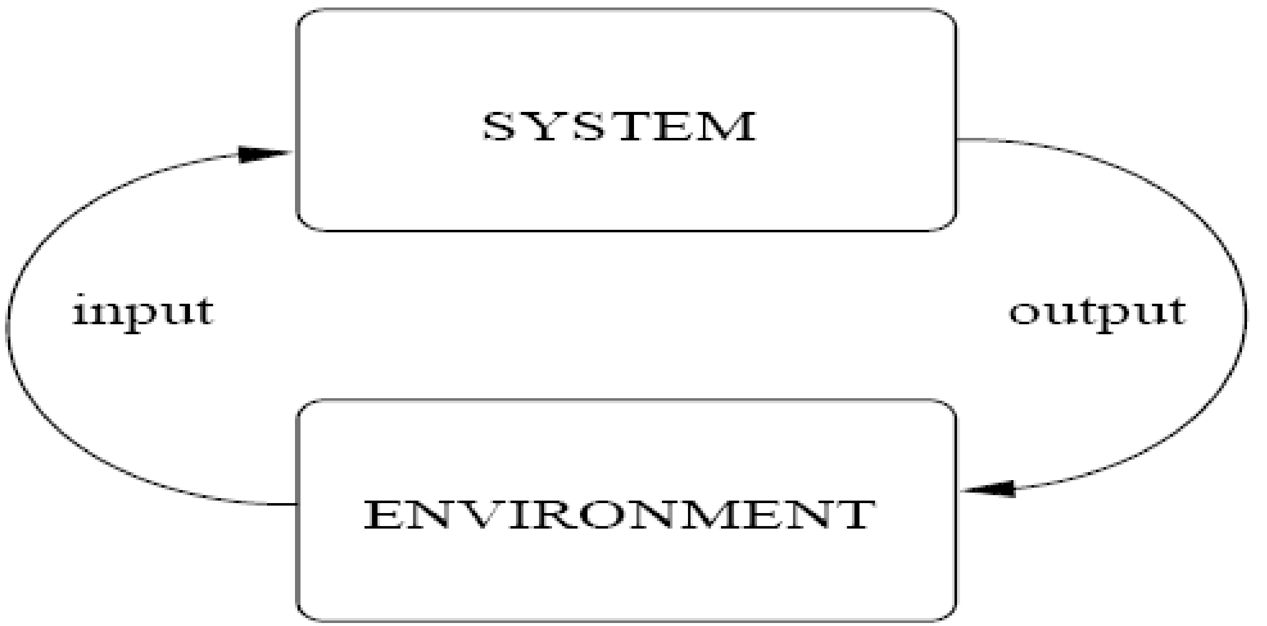
Outline of the talk

- Intelligent agent – reactivity
- Environments
- Agent architectures
  - Reactive
  - Deliberative
  - Hybrid

Agent definitions

- "An agent is anything that can be viewed as perceiving its environment through sensors and acting upon that environment through effectors."
- "Autonomous agents are computational systems that inhabit some complex dynamic environment, sense and act autonomously in this environment, and by doing so realize a set of goals or tasks for which they are designed."
- "An autonomous agent is a system situated within and a part of an environment that senses that environment and acts on it, over time, in pursuit of its own agenda and so as to affect what it senses in the future."

Basic abstract view of an agent



Reactivity

- An agent has to be able to react [adapt its behaviour] in an appropriate way to the dynamic changes in its "environment"
  - Other computational agents
  - Human agents/users
  - External information sources (e.g. sensors)
  - Physical objects (e.g. robots)
  - Internet
  - ...
- This is one of several properties that an intelligent agent should have ... [more on that next week]

Kinds of environments (I)

- Accessible vs inaccessible
  - An accessible environment is one in which the agent can obtain complete, accurate, up-to-date information about the environment's state.
  - Most moderately complex environments (including, for example, the everyday physical world and the Internet) are inaccessible.
  - The more accessible an environment is, the simpler it is to build agents to operate in it.

Kinds of environments (II)

- Deterministic vs non-deterministic
  - A deterministic environment is one in which any action has a single guaranteed effect — there is no uncertainty about the state that will result from performing an action.
  - The physical world can to all intents and purposes be regarded as non-deterministic.
  - Non-deterministic environments present greater problems for the agent designer.

Kinds of environments (III)

- Episodic vs non-episodic
  - In an episodic environment, the performance of an agent is dependent on a number of discrete episodes, with no link between the performance of an agent in different scenarios.
  - Episodic environments are simpler from the agent developer's perspective because the agent can decide what action to perform based only on the current episode — it need not reason about the interactions between this and future episodes.

Kinds of environments (IV)

- Static vs dynamic
  - A static environment is one that can be assumed to remain unchanged except by the performance of actions by the agent.
  - A dynamic environment is one that has other processes operating on it, and which hence changes in ways beyond the agent’s control.
  - The physical world is a highly dynamic environment.

Kinds of environments (V)

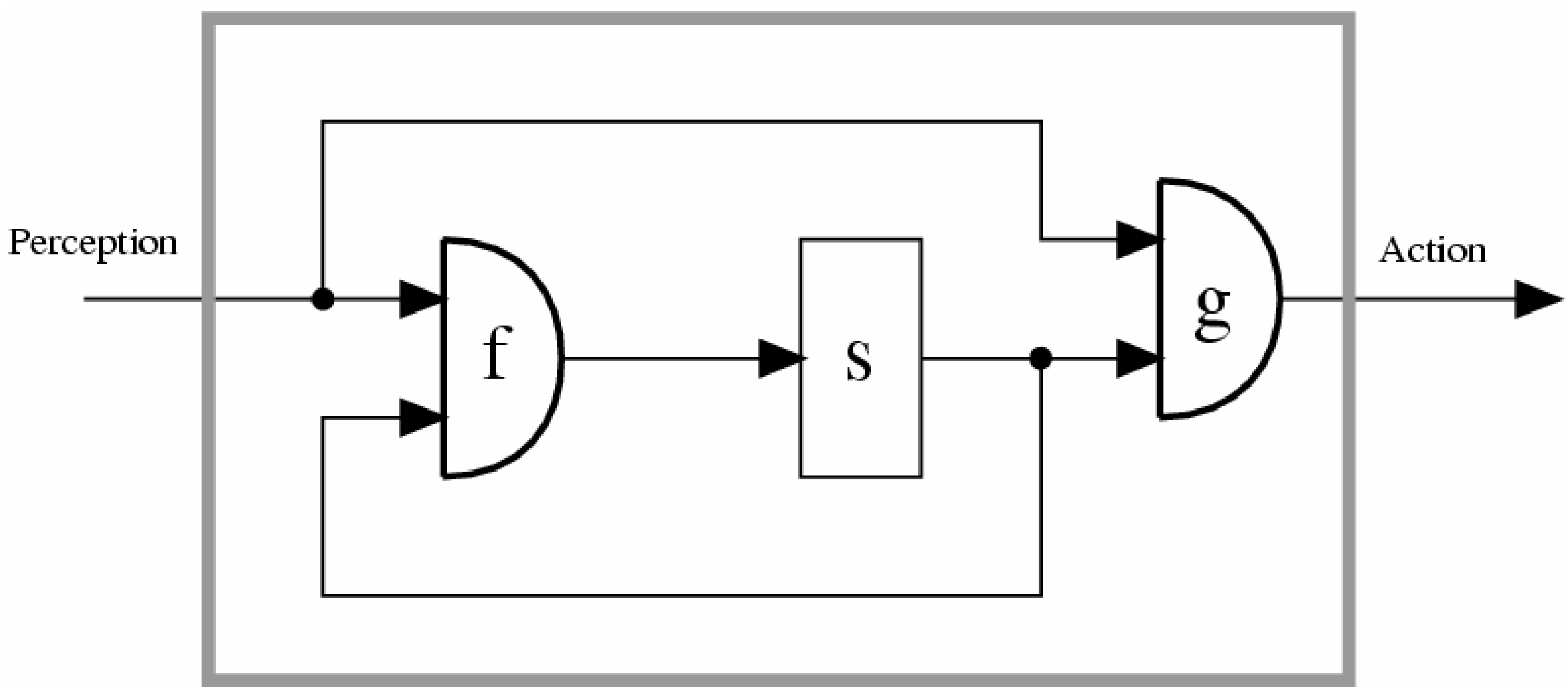
- Discrete vs continuous
  - An environment is discrete if there are a fixed, finite number of actions and percepts in it.
  - The real world is a continuous environment.

Practical exercise, step 1: think about the kind of environment

Agent architectures

- An architecture proposes a particular methodology for building an autonomous agent
  - How the construction of the agent can be decomposed into the construction of a set of component modules
  - How these modules should be made to interact
  - These two aspects define how the sensor data and the current internal state of the agent determine the actions (effector outputs) and future internal state of the agent

From perception to action



f = state update function
s = internal state
g = output function

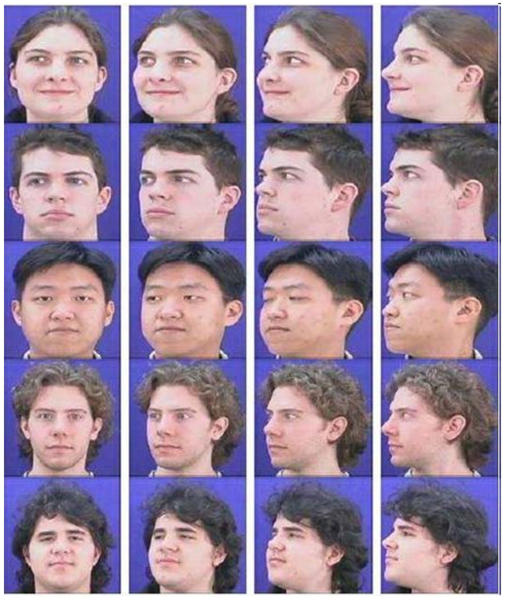
Main kinds of agent architectures

- Reactive architectures
  - Focused on fast reactions/responses to changes detected in the environment
- Deliberative architectures (symbolic)
  - Focused on long-term planning of actions, centred on a set of basic goals
- Hybrid architectures
  - Combining a reactive side and a deliberative side

Reactive vs Deliberative: example

- Robot that has to reach a certain point
  - Reactive
    - Sensor in the front of the robot
    - Change movement right/left when sensor detects obstacle
      - Minimal computation based on current location and destination point
  - Deliberative
    - Explicit representation of the environment (map)
    - Planning procedure that finds the minimal route between the current position and the destination
      - High computational cost
      - Possible dynamic re-plannings needed

Reactive Architectures



- There are many unsolved (some would say insoluble) problems associated with symbolic AI
  - Computational cost, brute search
  - Problems below the 100 ms threshold
    - For example, face recognition
- These problems have led some researchers to question the viability of the whole paradigm, and to the development of *reactive* architectures
- Although united by a belief that the assumptions underpinning mainstream AI are in some sense wrong, reactive agent researchers use many different techniques

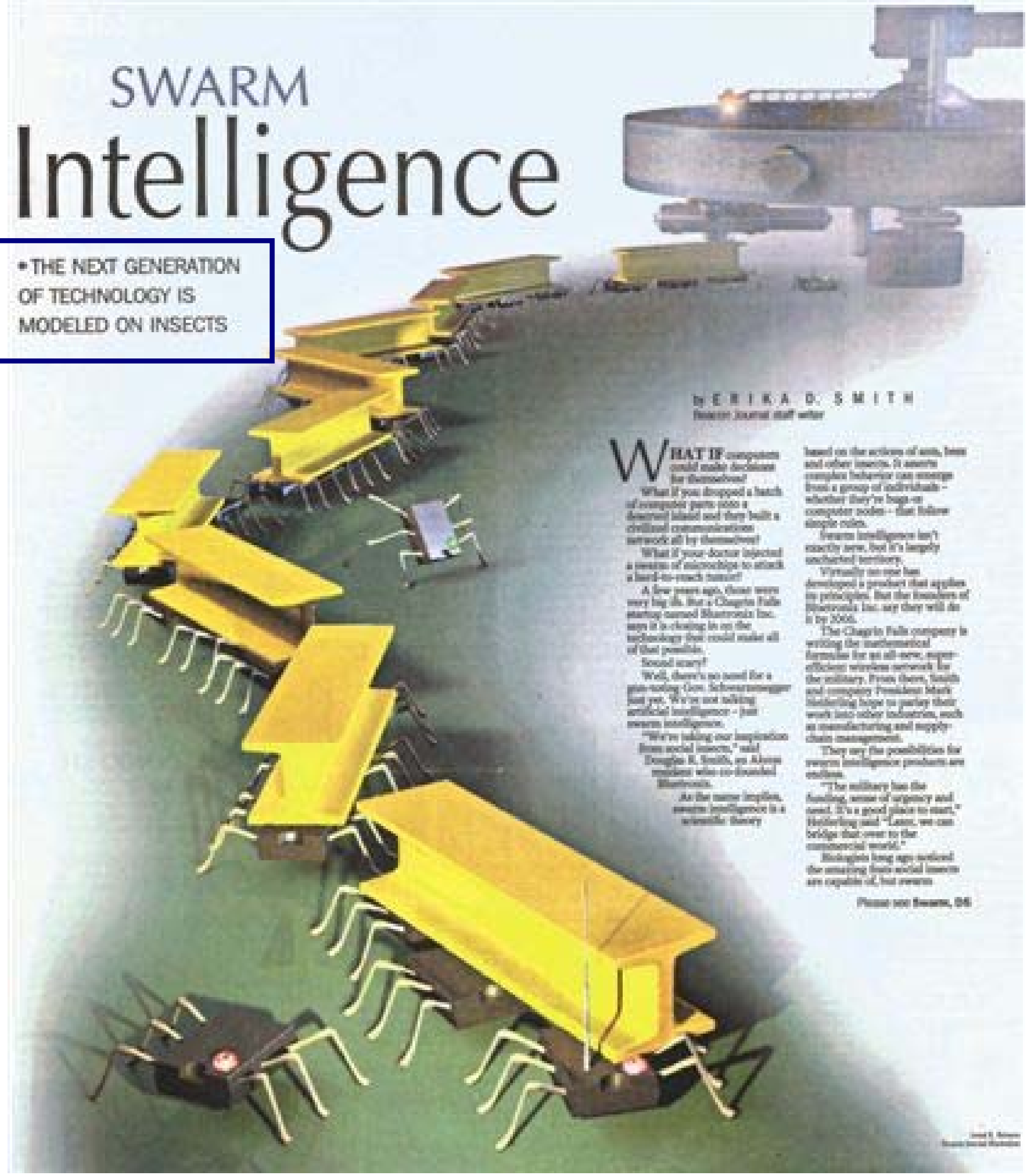
Reactive agents – basic ideas

- Reactive agents have
  - at most a very simple internal representation of the world,
  - but provide tight coupling of perception and action
- Behaviour-based paradigm
- Intelligence is a product of the interaction between an agent and its environment

Classic example: ant colony

- A single ant has very little intelligence, computing power or reasoning abilities
- The union of a set of ants and the interaction between them allows the formation of a highly complex, structured and efficient system.



SWARM Intelligence

▪ THE NEXT GENERATION OF TECHNOLOGY IS MODELED ON INSECTS

by ERIKA D. SMITH
Beacon Journal staff writer

Main characteristics (I)

- Emergent functionality
  - Simple agents
  - Simple interaction
  - Complex behaviour patterns appear as a result of the dynamic interactions
  - The global behaviour of the system is not specified a priori
    - Dynamic movement of robots, depending on obstacles

Main characteristics (II)

- Task decomposition
  - Agents composed of autonomous modules
  - Each module manages a given task
    - Sensor, control, computations
  - Minimal, low-level communication between modules
  - There isn’t any world global model
  - There isn’t any “planning/controller agent”

Main characteristics (III)

- Raw data
  - Basic data from sensors
  - There isn't any complex symbolic management of data as in classical AI
    - Refusal of the *Hypothesis of the physic symbols system* [basic pillar of symbolic AI]
      - "Intelligent behaviour can only be obtained in symbol-processing systems"

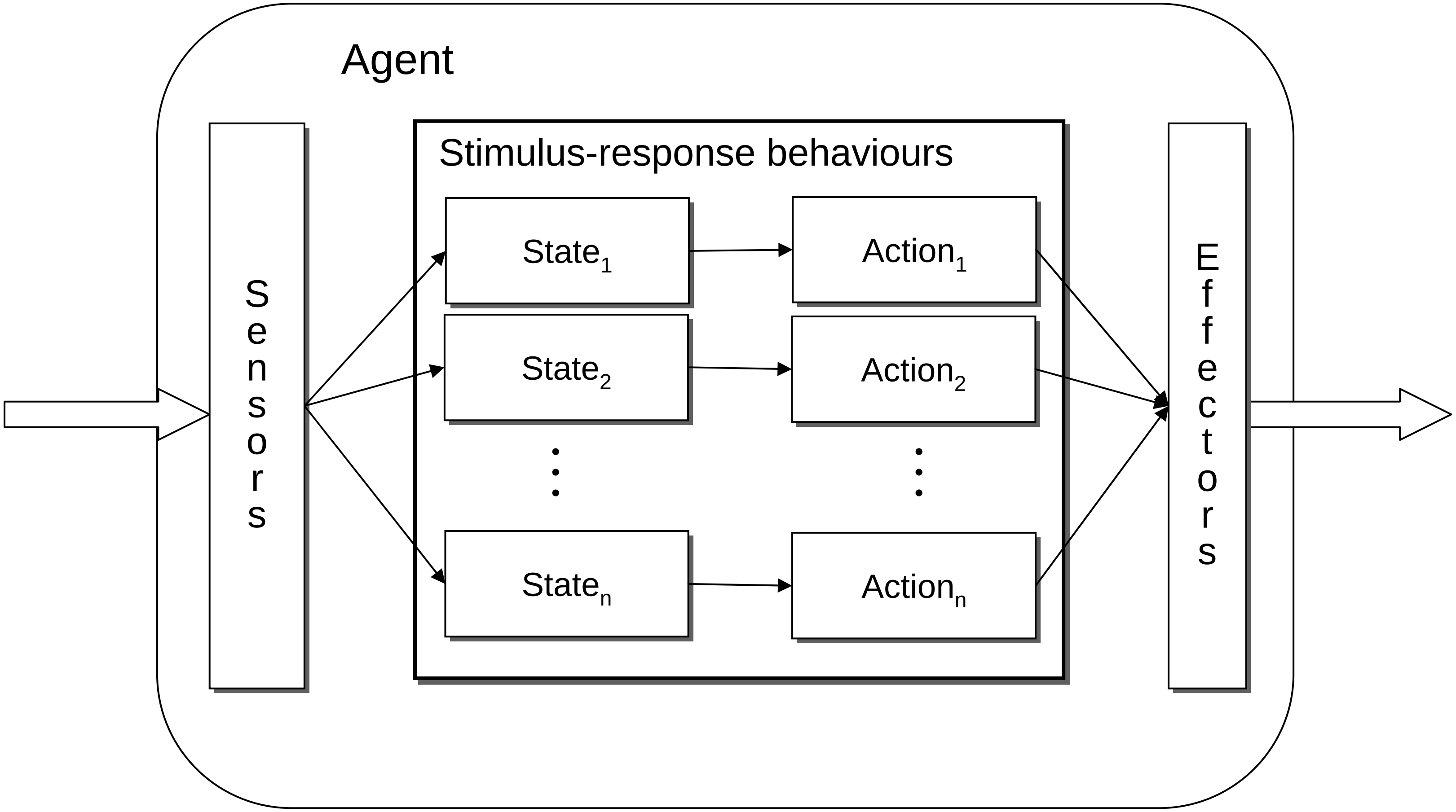
Basic concept

- Each behaviour continually maps perceptual input to action output
- Reactive behaviour: action rules: $S \rightarrow A$

  where S denotes the states of the environment, and A the primitive actions the agent is capable of performing.
- Example:

$$\text{action}(s) = \begin{cases} \textit{Heater off}, & \text{if temperature is OK in state s} \\ \textit{Heater on}, & \text{otherwise} \end{cases}$$

Basic schema of reactive architecture



Rodney Brooks

Director of the Computer Science and Artificial Intelligence Lab (MIT)

1997-2007

Brooks refutal of symbolic AI

- Brooks has put forward three theses:
  1. Intelligent behaviour can be generated *without* explicit representations of the kind that symbolic AI proposes
  2. Intelligent behaviour can be generated *without* explicit abstract reasoning of the kind that symbolic AI proposes
     - Reduced computation on sensor-like data
  3. Intelligence is an *emergent* property of certain complex systems

Brooks – key ideas (I)

- Situatedness: ‘Real’ intelligence is situated in the world
  - The world is its best model
  - The world is always up-to-date
  - A model is an abstraction, a simplification of the world, considering a particular set of characteristics and disregarding others

Brooks – key ideas (II)

- Embodiment: ‘Real’ intelligence requires a physical body, and cannot be found in disembodied systems such as theorem provers or expert systems
  - Physical robots
- Intelligence and emergence: ‘Intelligent’ behavior 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

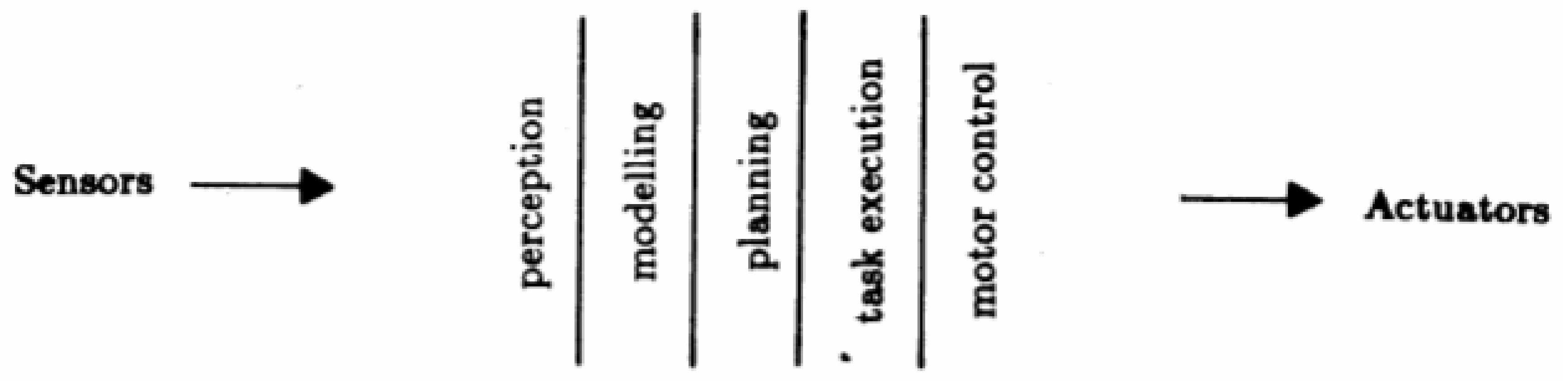
Brooks – behaviour languages

- To illustrate his ideas, Brooks built some systems 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 agent, as different behaviours may be applicable at the same time

Behaviour layers

- Lower layers represent more primitive kinds of behaviour (such as avoiding obstacles)
- Higher layers represent more complex behaviours (e.g. identifying an object)
- Lower layers have precedence over layers further up the hierarchy
- The resulting systems are, in terms of the amount of computation they do, *extremely* simple
- Some of the robots do tasks that would be impressive if they were accomplished by symbolic AI systems

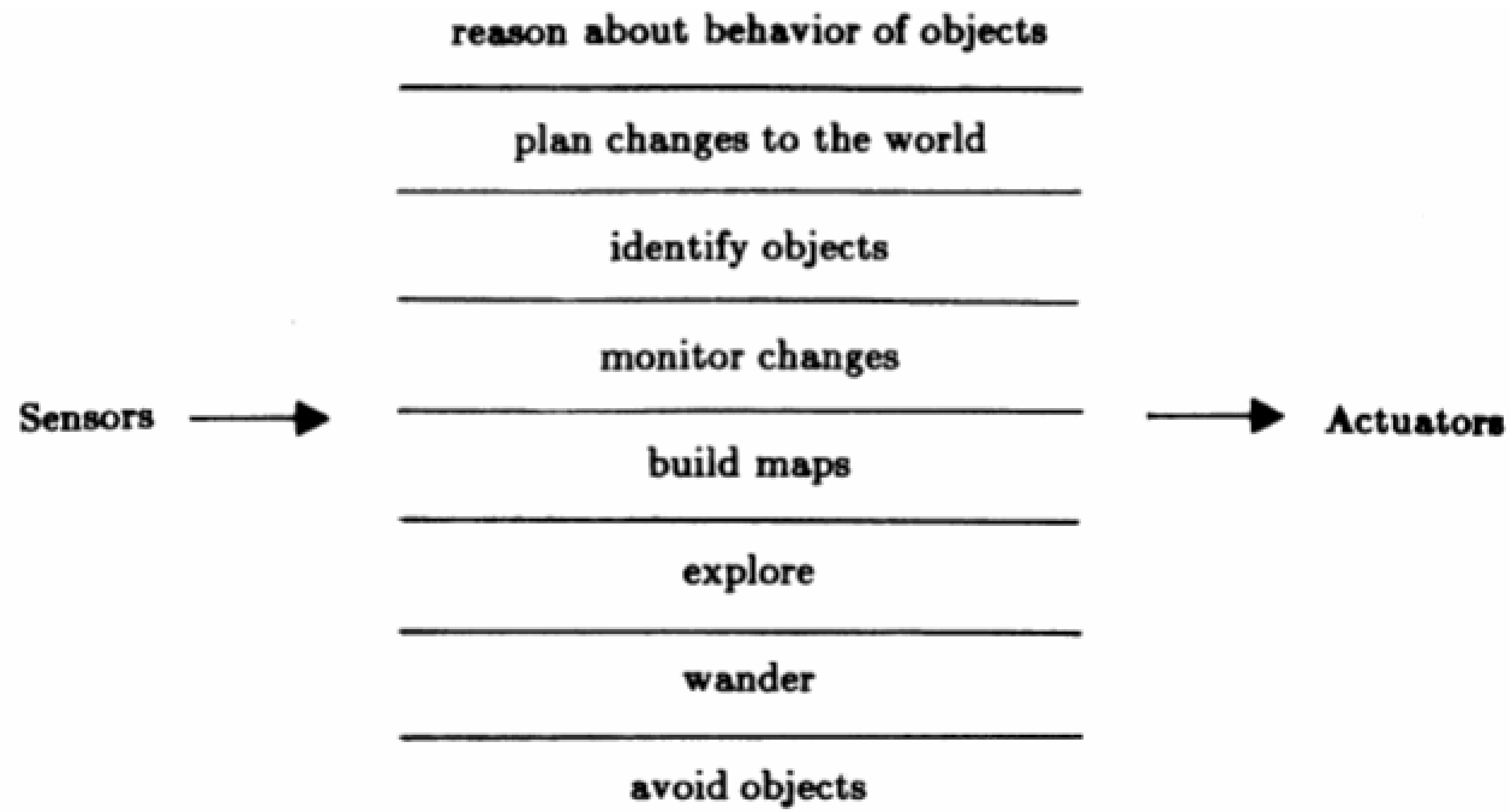
Decomposition of a Mobile Robot Control System into Functional Modules



Modules in Classical AI systems

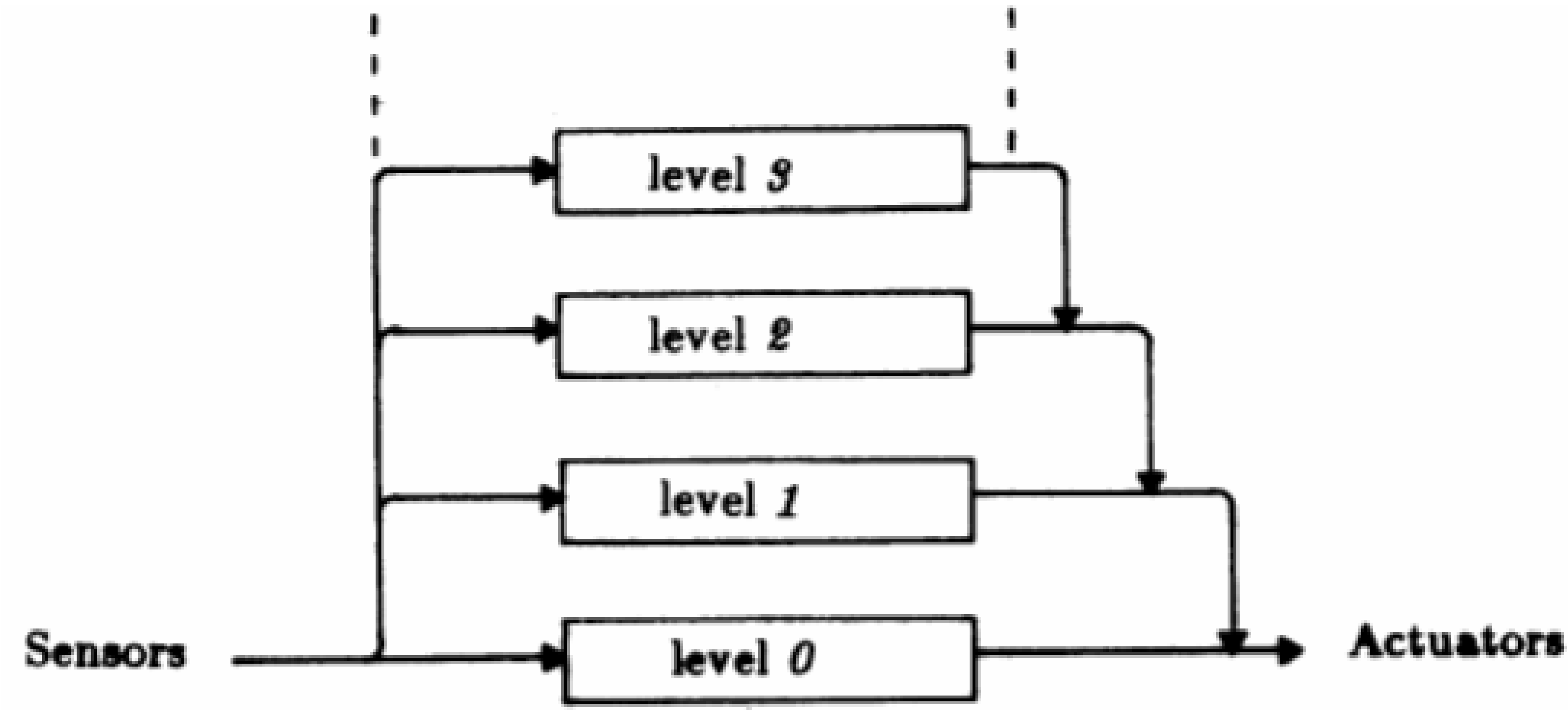
From Brooks, “A Robust Layered Control System for a Mobile Robot”, 1985

Decomposition Based on Task Achieving Behaviours



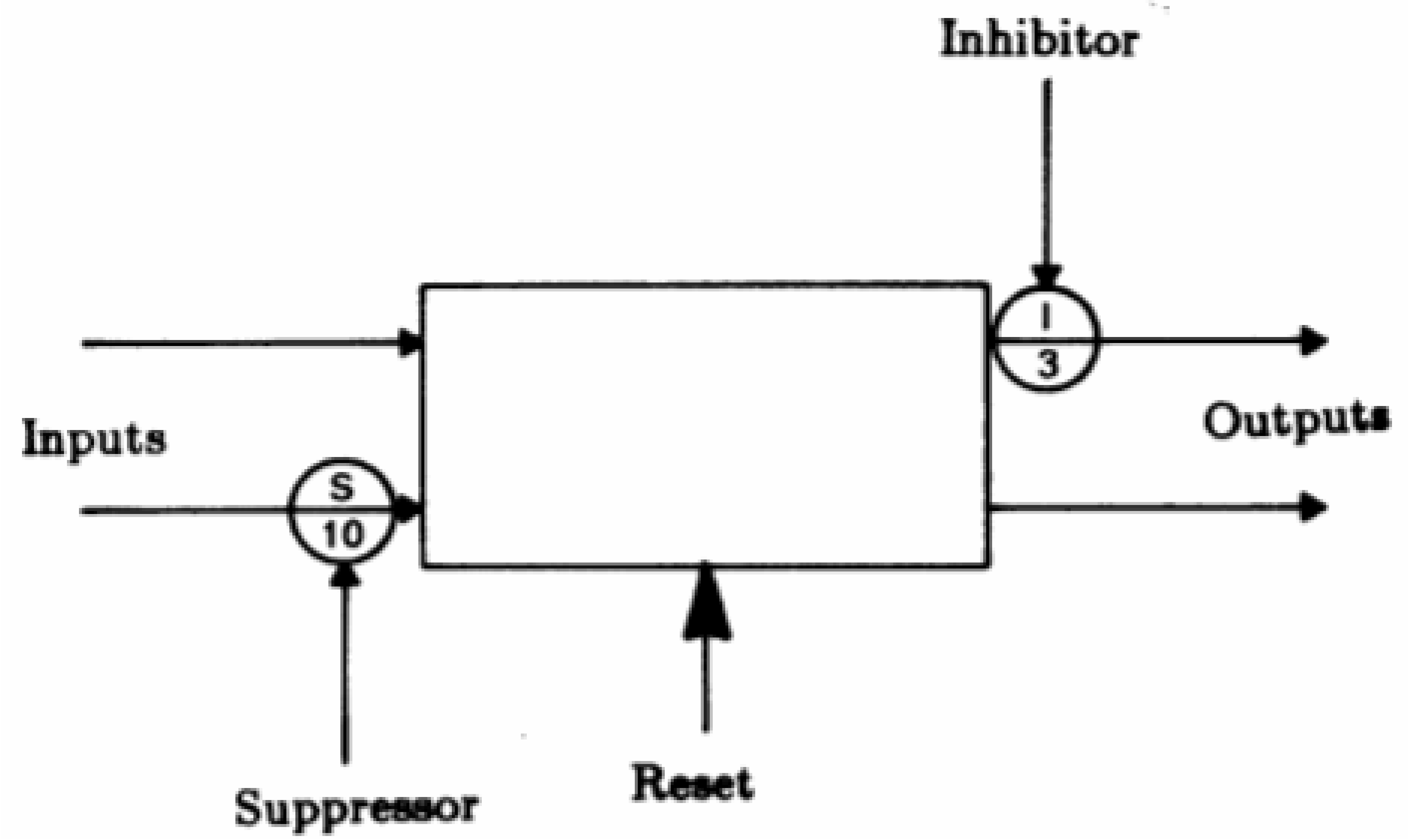
From Brooks, “A Robust Layered Control System for a Mobile Robot”, 1985

Layered Control in the Subsumption Architecture



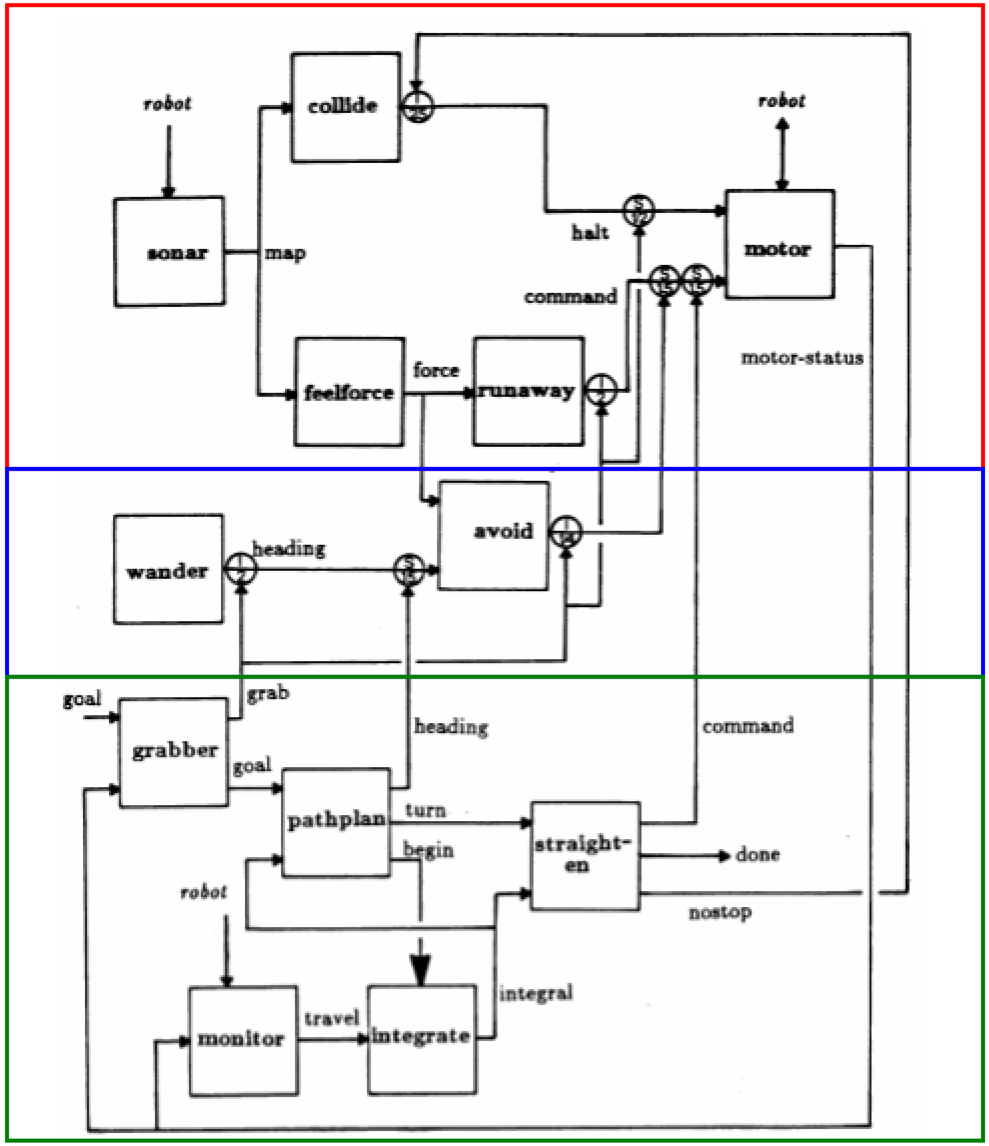
From Brooks, “A Robust Layered Control System for a Mobile Robot”, 1985

Schematic of a Module



From Brooks, “A Robust Layered Control System for a Mobile Robot”, 1985

Levels 0, 1, and 2 Control Systems

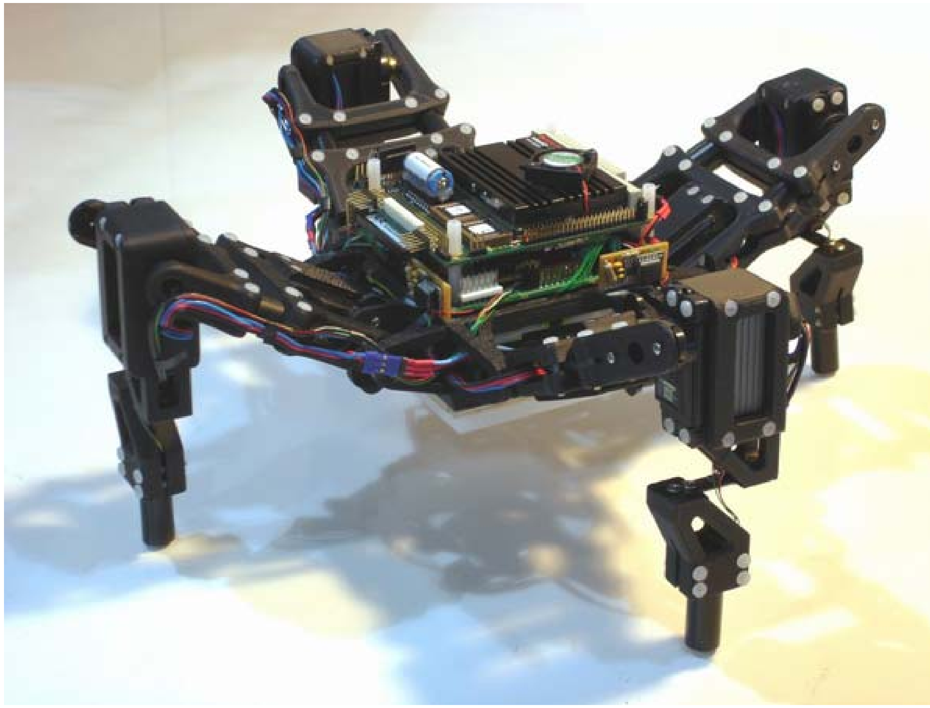


Avoid collisions

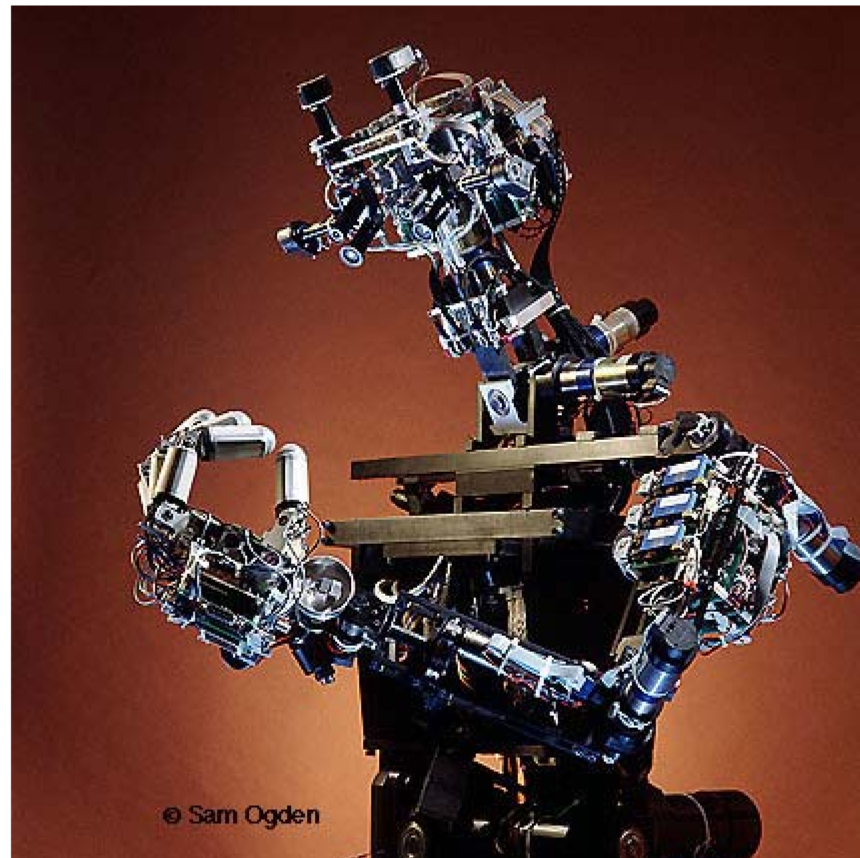
Random movement

Pick up objects, monitor, explore

From Brooks, “A Robust Layered Control System for a Mobile Robot”, 1985




ATTILA


© Sam Ogden

Steels' Mars Explorer

- Steels' Mars explorer system, using the 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 sample of a precious rock. The location of the samples is not known in advance, but it is *known that they tend to be clustered.*

Steels' Mars Explorer Rules (I)

- 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)

Steels' Mars Explorer Rules (II)

- 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)

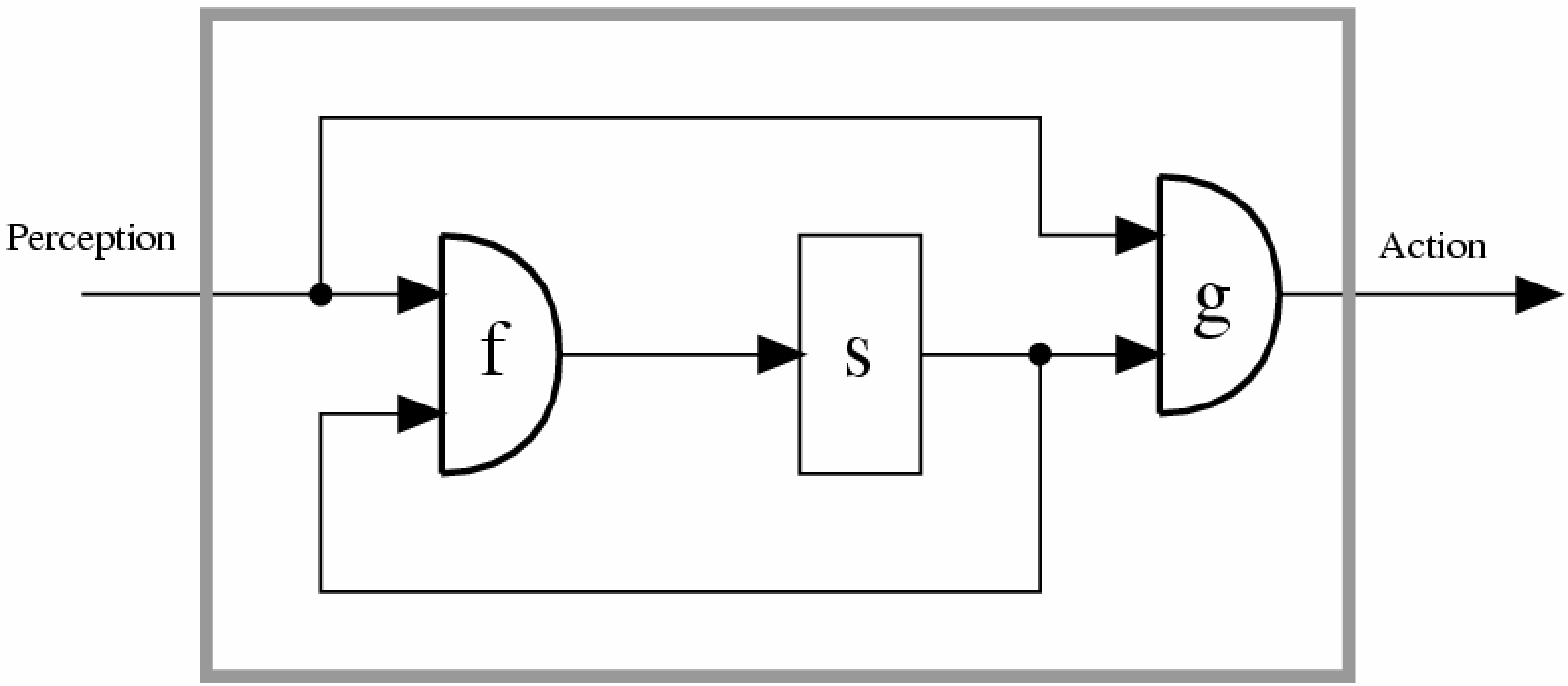
Situated Automata

- A sophisticated theoretical approach is that of Rosenschein and Kaelbling
- In their *situated automata* paradigm, an agent is specified in a rule-like (declarative) language, and this specification is then compiled down to a digital machine, which satisfies the declarative specification
- This digital machine can operate in a *provable time bound*
- Reasoning is done *off line*, at *compile time*, rather than *online* at *run time*

Situated Automata components

- An agent is specified in terms of two components: perception and action
- Two programs are then used to synthesize agents
  - RULER is used to specify the perception component of an agent
  - GAPPS is used to specify the action component

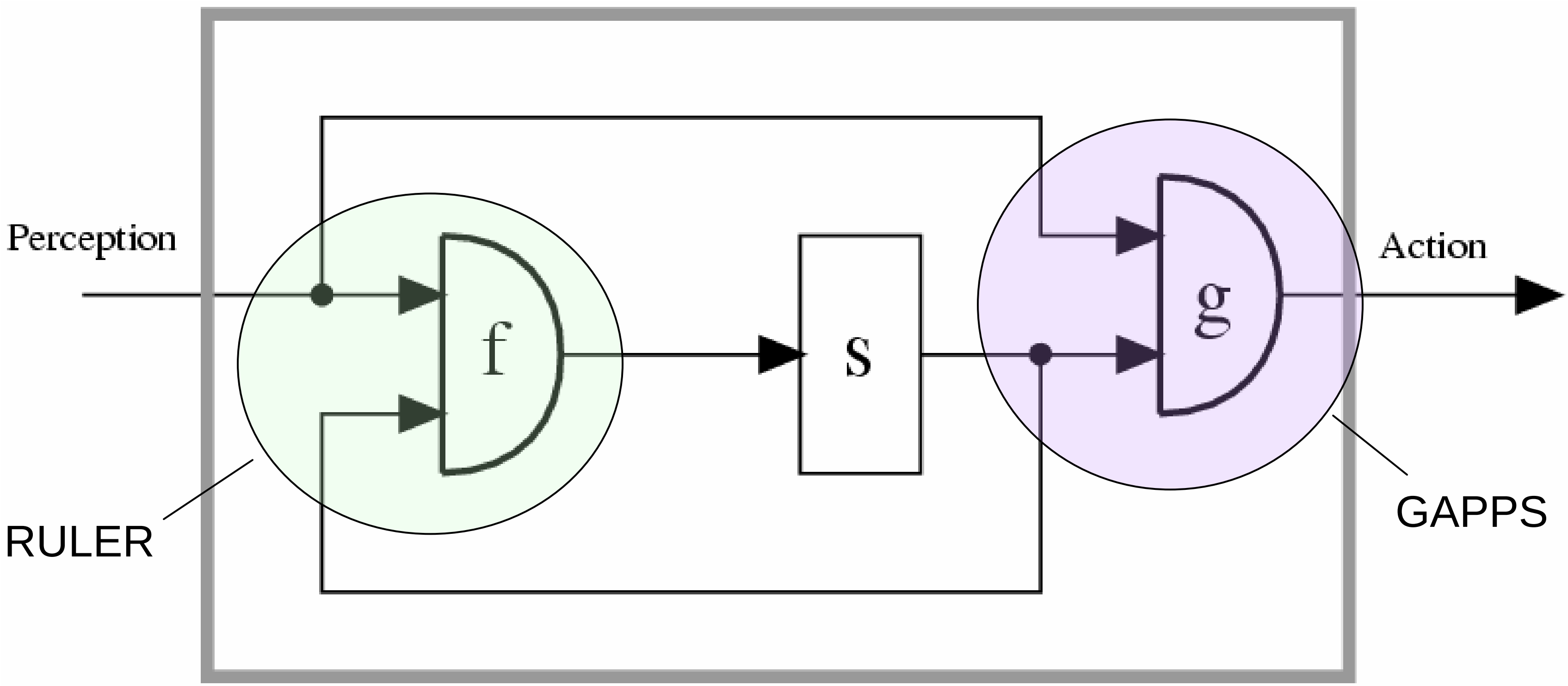
Circuit Model of a Finite-State Machine



f = state update function
s = internal state
g = output function

From Rosenschein and Kaelbling,
“A Situated View of Representation and Control”, 1994

Circuit Model of a Finite-State Machine



From Rosenschein and Kaelbling,
“A Situated View of Representation and Control”, 1994

RULER – Situated Automata

- RULER takes as its input three components
  - The semantics of the agent's inputs (‘whenever bit 1 is on, it is raining’)
  - A set of static facts (‘whenever it is raining, the ground is wet’)
  - A specification of the state transitions of the world (‘if the ground is wet, it stays wet until the sun comes out’).
- The programmer then specifies the desired semantics for the output (‘if this bit is on, the ground is wet’)
- The compiler designs a circuit whose output will have the correct semantics

GAPPS – Situated Automata

- The GAPPS program takes as its input
  - A set of *goal reduction rules*,
    - Rules that encode information about how goals can be achieved in a given state
  - A top level goal
- Then it generates a program that can be translated into a digital circuit in order to realize the goal
- The generated circuit does not represent or manipulate symbolic expressions; all symbolic manipulation is done at compile time

Advantages of Reactive Agents

- Simplicity of individual agents
- Flexibility, adaptability
  - Ideal in very dynamic and unpredictable environments
- Computational tractability
  - Avoiding complex planning/reasoning procedures
  - Avoiding continuous model update
- Robustness against failure
  - No central planning component (e.g. ant colony)
- Elegance

Limitations of Reactive Agents (I)

- Agents without environment models must have sufficient information available from local environment
- If decisions are based on *local* environment, how can we take into account *non-local* information?
  - “Short-term” view
- No long-term planning capabilities
- Limited applicability
  - Games, simulations, basic robots (insects)

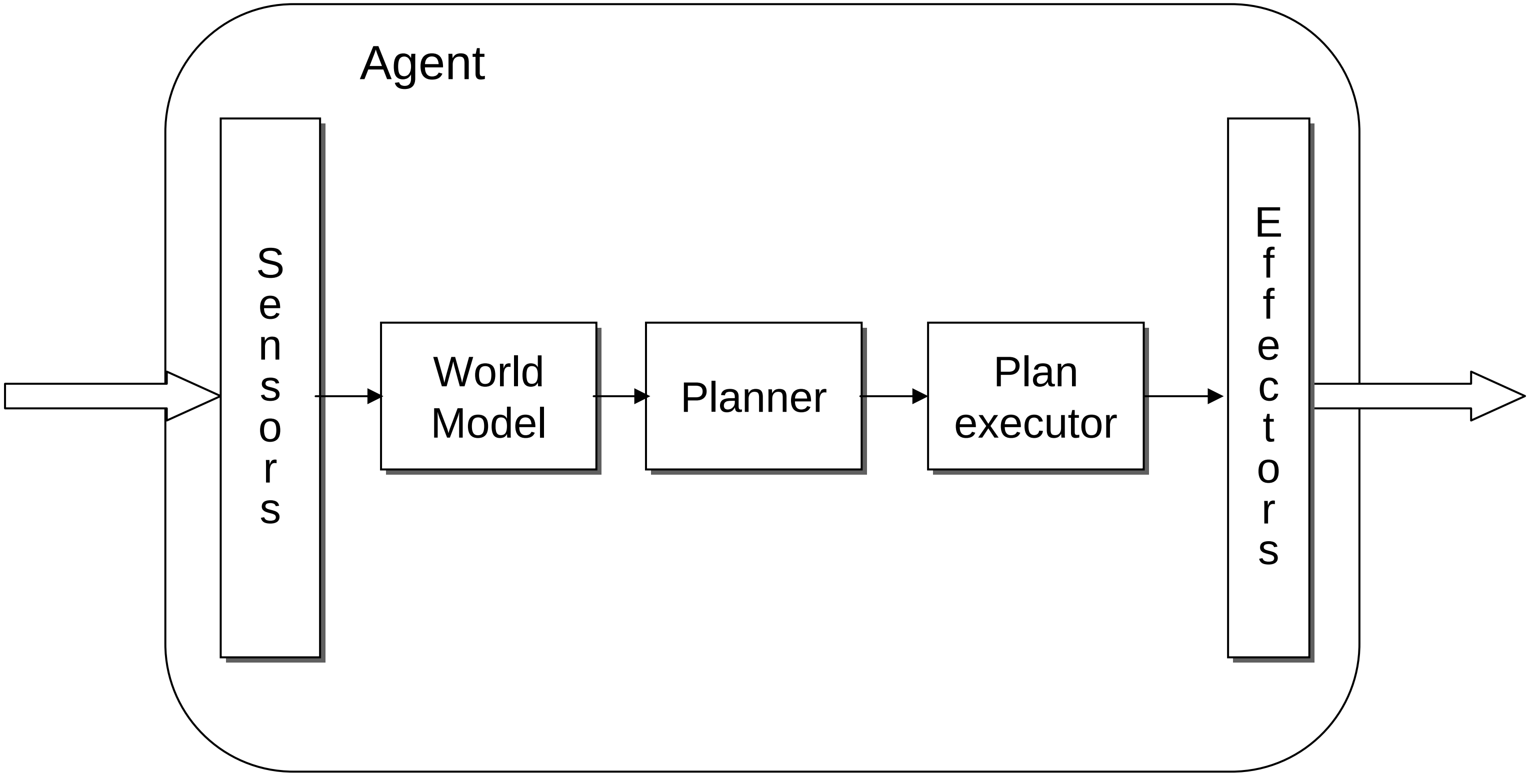
Limitations of Reactive Agents (II)

- Difficult to make reactive agents that learn
  - Dynamic evolution of rules?
- Since behaviour emerges from component interactions plus environment, it is hard to see how to *engineer* specific agents (no principled methodology exists)
- It is hard to engineer agents with large numbers of behaviours (dynamics of interactions become too complex to understand)

Deliberative agent architecture

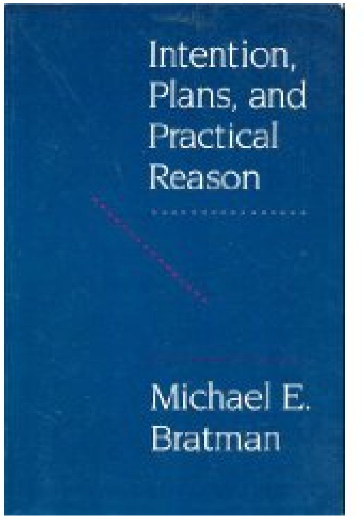
- Explicit symbolic model of the world
- Decisions are made via logical reasoning, based on pattern matching and symbolic manipulation
- Sense-plan-act problem-solving paradigm of classical AI planning systems

Basic deliberative architecture



Belief-Desire-Intention (BDI) model

- A theory of practical reasoning.
- Originally developed by Michael E. Bratman in his book ”*Intentions, Plans, and Practical Reason*”, (1987).
- Concentrates in the roles of the intentions in practical reasoning.



Practical reasoning

- Reasoning directed towards actions — 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 and what the agent believes.” (Bratman)

  ”We deliberate not about ends, but about means. We assume the end and consider how and by what means it is attained.” (Aristotle)

Human practical reasoning

- Human practical reasoning consists of two activities:
  - *Deliberation*, deciding *what* state of affairs we want to achieve
    - the outputs of deliberation are *intentions*
  - *Means-ends reasoning*, deciding *how* to achieve these states of affairs
    - the outputs of means-ends reasoning are *plans*

Belief-Desire-Intention paradigm

- Beliefs:
  - Agent’s view of the environment/world.

- Desires:
  - Follow from the beliefs. Desires can be unrealistic and inconsistent.
- Goals:
  - A subset of the desires. Realistic and consistent.

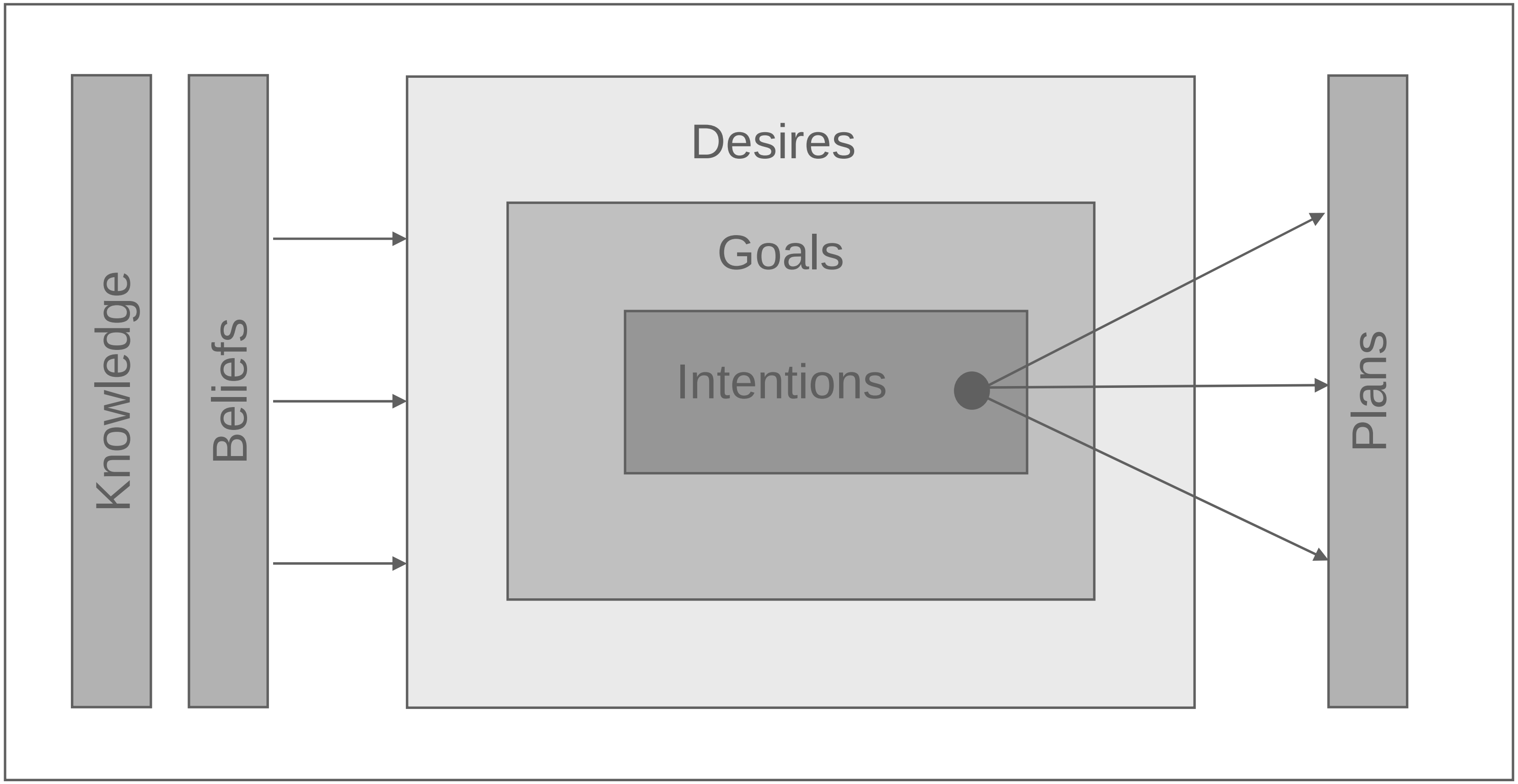
    Determine potential processing.
- Intentions:
  - A subset of the goals. A goal becomes an intention when an agent decides to commit to it (e.g. by assigning priorities to goals)

- Plans:
  - Sequences of actions that are needed to achieve the intentions, given the agent’s beliefs

BDI plans

- In BDI implementations plans usually have:
  - a *name*
  - a *goal*
    invocation condition that is the triggering event for the plan
  - a *pre-condition list*
    list of facts which must be true for plan to be executed
  - a *delete list*
    list of facts that are no longer true after plan is performed
  - an *add list*
    list of facts made true by executing the actions of the plan
  - a *body*
    list of actions

Belief-Desire-Intention architecture



Intention is choice with commitment (Cohen & Levesque)

- There should be ”rational balance” among the beliefs, goals, plans, intentions, commitments and actions of autonomous agents.
- Intentions play a big role in maintaining ”rational balance”
- An autonomous agent should act on its intentions, not in spite of them
  - adopt intentions that are feasible
  - drop the ones that are not feasible
  - keep (or commit to) intentions, but not forever
  - discharge those intentions believed to have been satisfied
  - alter intentions when relevant beliefs change

Using plans to constrain reasoning

- An agent's plans serve to frame its subsequent reasoning problems so as to constrain the amount of resources needed to solve them
  - Agents *commit* to their plans
  - Their plans tell them *what to* reason about, and *what not* to reason about
  - Plans can help reasoning in differents levels of abstraction

Intention reconsideration

- Intentions (plans) enable the agent to be goal-driven rather than event-driven.
- By committing to intentions the agent can pursue long-term goals.
- However, it is necessary for a BDI agent to reconsider its intentions from time to time:
  - The agent should drop intentions that are no longer achievable.
  - The agent should adopt new intentions that are enabled by opportunities.

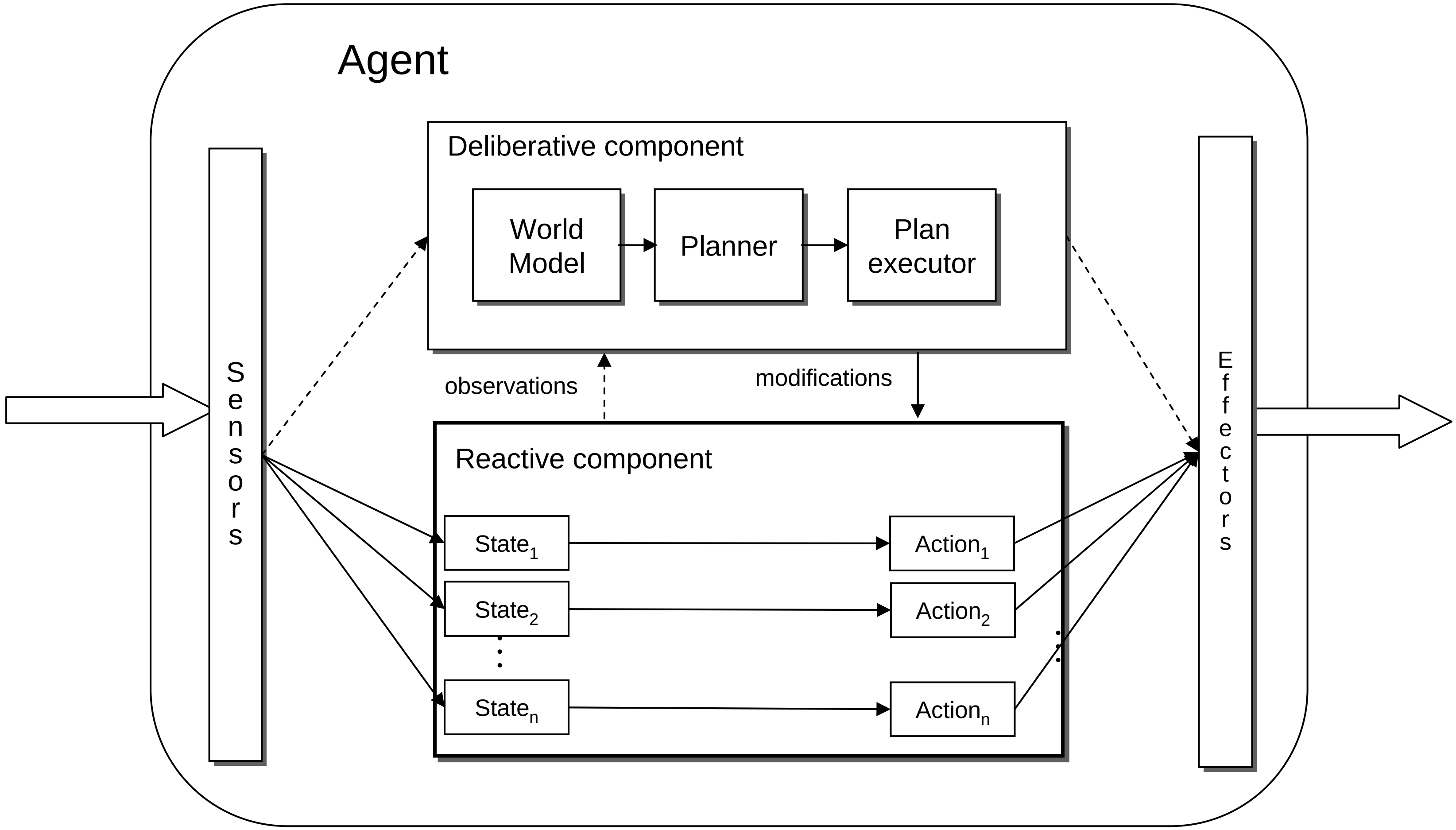
Problems in the deliberative approach

- Dynamic world
  - Update symbolic world model
  - World changes while planning is being done
- Representation language
  - Expressive enough to be useful in any domain
  - Limited enough to be computationally tractable
- Classical planning => complete, optimal solutions
  - High computational cost
  - Sometimes a sub-optimal low-cost fast reaction can be effective

Hybrid Approaches

- Many researchers have argued that neither a completely deliberative nor a completely reactive approach are suitable for building agents
- They have suggested using *hybrid* systems, which attempt to marry classical and alternative approaches
- An obvious approach is to build an agent out of two (or more) subsystems:
  - a *deliberative* one, containing a symbolic world model, which develops plans and makes decisions in the way proposed by symbolic AI
  - a *reactive* one, which is capable of reacting quickly to events without complex reasoning

Hybrid agent architecture



Layered Architectures

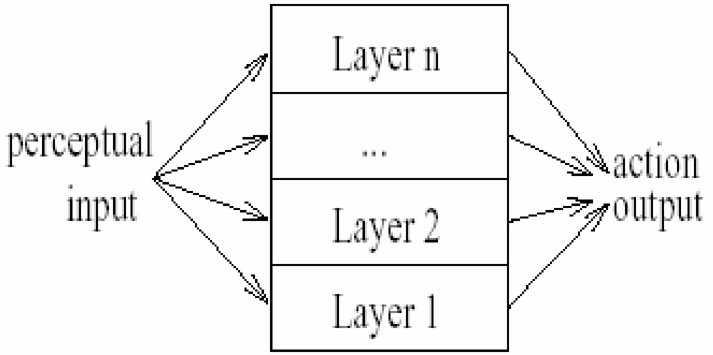
- Often, the reactive component is given some kind of precedence over the deliberative one
- This kind of structuring leads naturally to the idea of a *layered* architecture, of which TOURINGMACHINES and INTERRAP are examples
- In such an architecture, an agent's control subsystems are arranged into a hierarchy, with higher layers dealing with information at increasing levels of abstraction

Layering techniques

- 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 various layers.
- *Horizontal layering*
  Each layer is directly connected to the sensory input and action output.
  In effect, each layer itself acts like an agent, producing suggestions as to what action to perform.
- *Vertical layering*
  Sensory input and action output are dealt with by at most one layer each.

Horizontal layering

m possible actions suggested by each layer, *n* layers



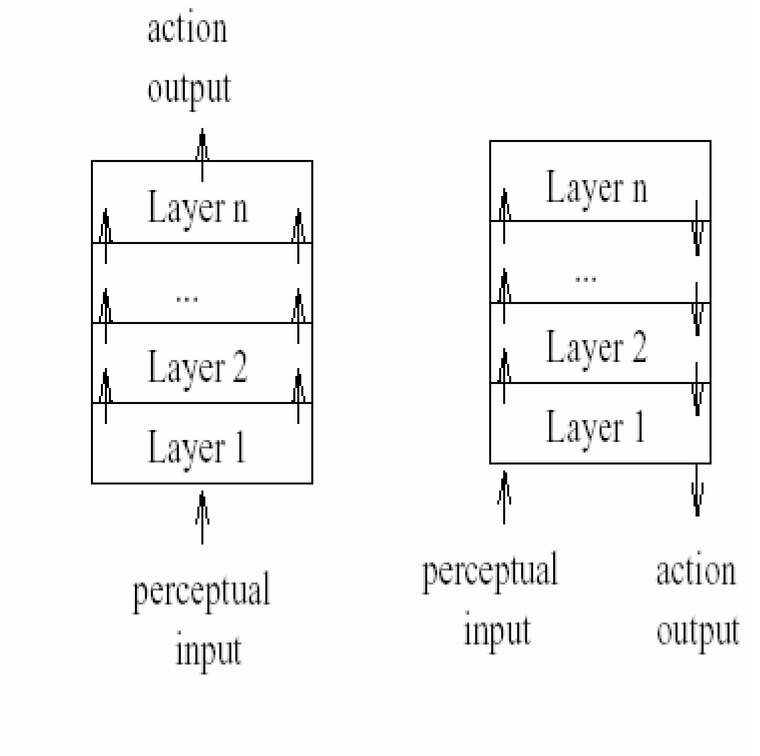
(a) Horizontal layering

$O(m^n)$ possible options to be considered

Introduces bottleneck in central control system

Vertical layering

m possible actions suggested by each layer, *n* layers



(b) Vertical layering (One pass control)

(c) Vertical layering (Two pass control)

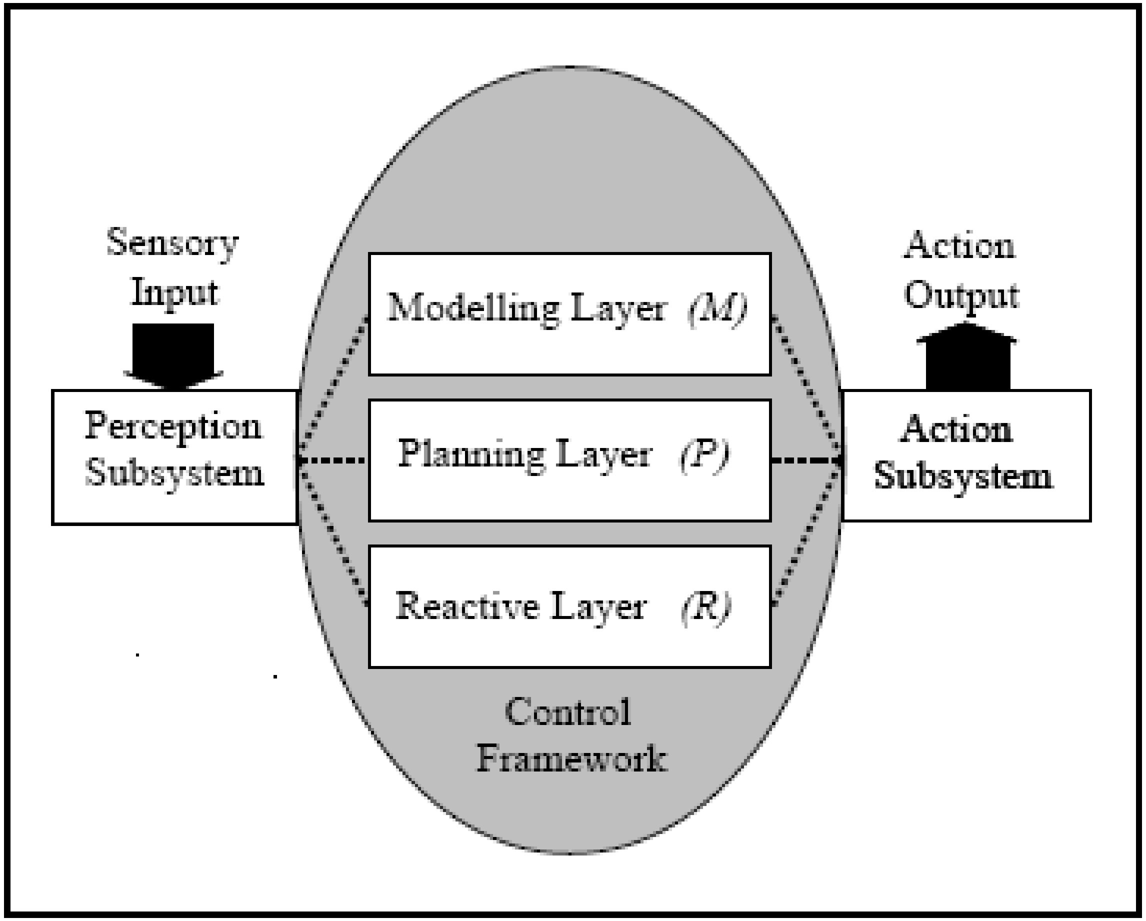
$O(mn)$ interactions between layers

Not fault tolerant to layer failure

Ferguson – TOURINGMACHINES

- The TOURINGMACHINES architecture consists of *perception* and *action* subsystems, which interface directly with the agent’s environment, and three *control layers*, embedded in a *control framework*, which mediates between the layers

Ferguson – TOURINGMACHINES



Ferguson – TOURINGMACHINES

- The *reactive layer* is implemented as a set of situation-action rules, *a la* subsumption architecture

```
rule-1: obstacle-avoidance
    if
          is-in-front(Obstacle, Observer) and
          speed(Observer) > 0 and
          separation(Obstacle, Observer) < ThreshHold
    then
          change-orientation(ObstacleAvoidanceAngle)
```

- The *planning layer* constructs plans and selects actions to execute in order to achieve the agent's goals

Ferguson – TOURINGMACHINES

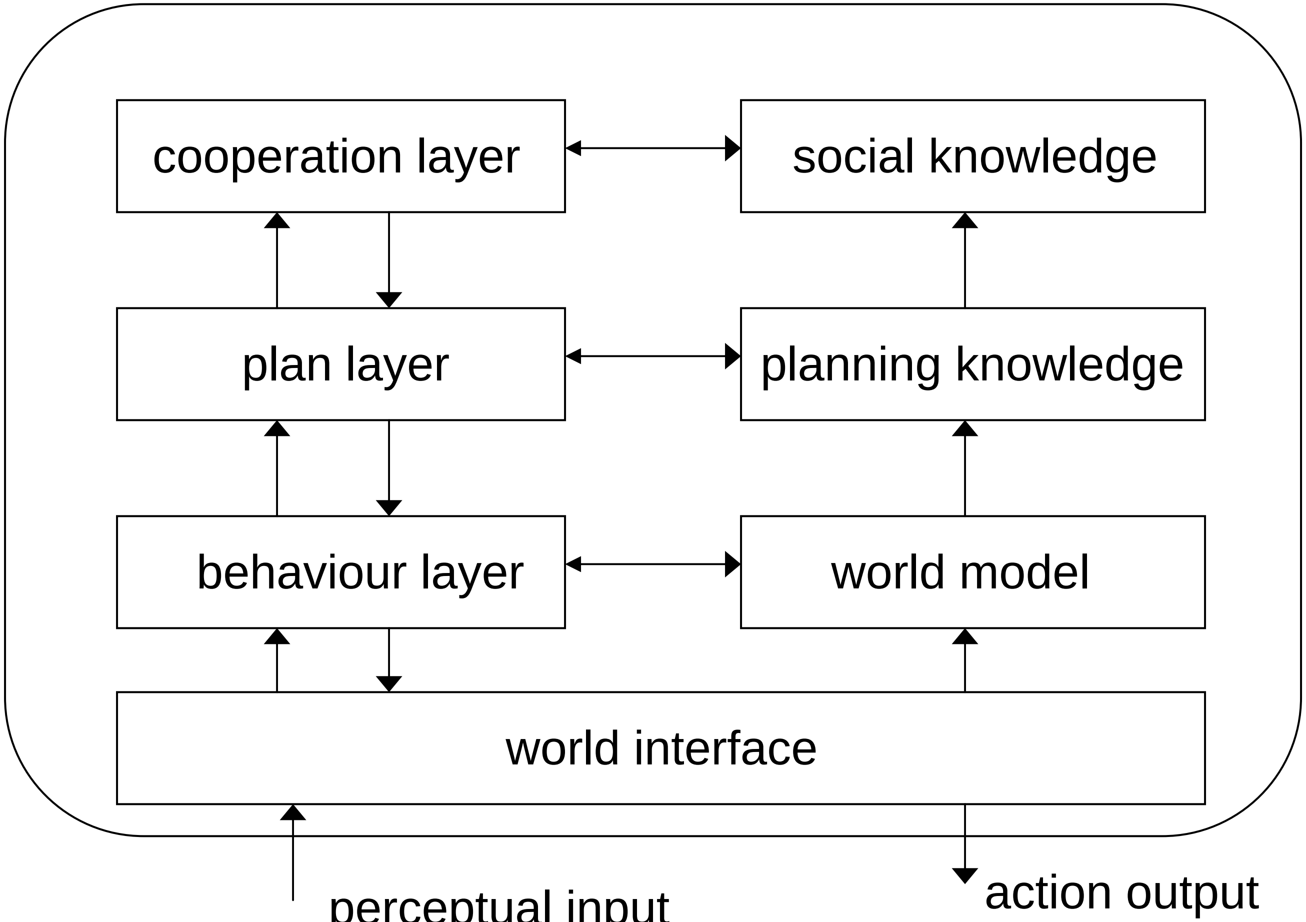
- The *modeling* *layer* contains symbolic representations of the ‘cognitive state’ of other entities in the agent’s environment
- The three layers communicate with each other and are embedded in a control framework, which use *control rules*

```
censor-rule-1:
     if entity(obstacle-6) in perception-buffer
     then remove-sensory-record(layer-R, entity(obstacle-6))
```

[Prevents the reactive layer from reacting in front of obstacle-6]

Müller –InteRRaP

- Vertically layered, two-pass architecture



Behaviour layer

- Reactive part of the architecture
- Works with the world model (beliefs on the world state)
- Only level that interacts with the real world
- Has a set of “situation $\rightarrow$ action” rules
  - Fast recognition of situations that deserve a quick reaction
- Makes routine tasks efficiently, without complex symbolic planning

Planning layer

- Works with the mental model (beliefs on the own agent)
- Standard deliberative level
- Implements local behaviour guided towards certain goals

Cooperative planning layer

- Works with the social model (beliefs on other agents of the system)
- Allows planning and cooperation with other agents
  - Global plans of action
  - Conflict resolution

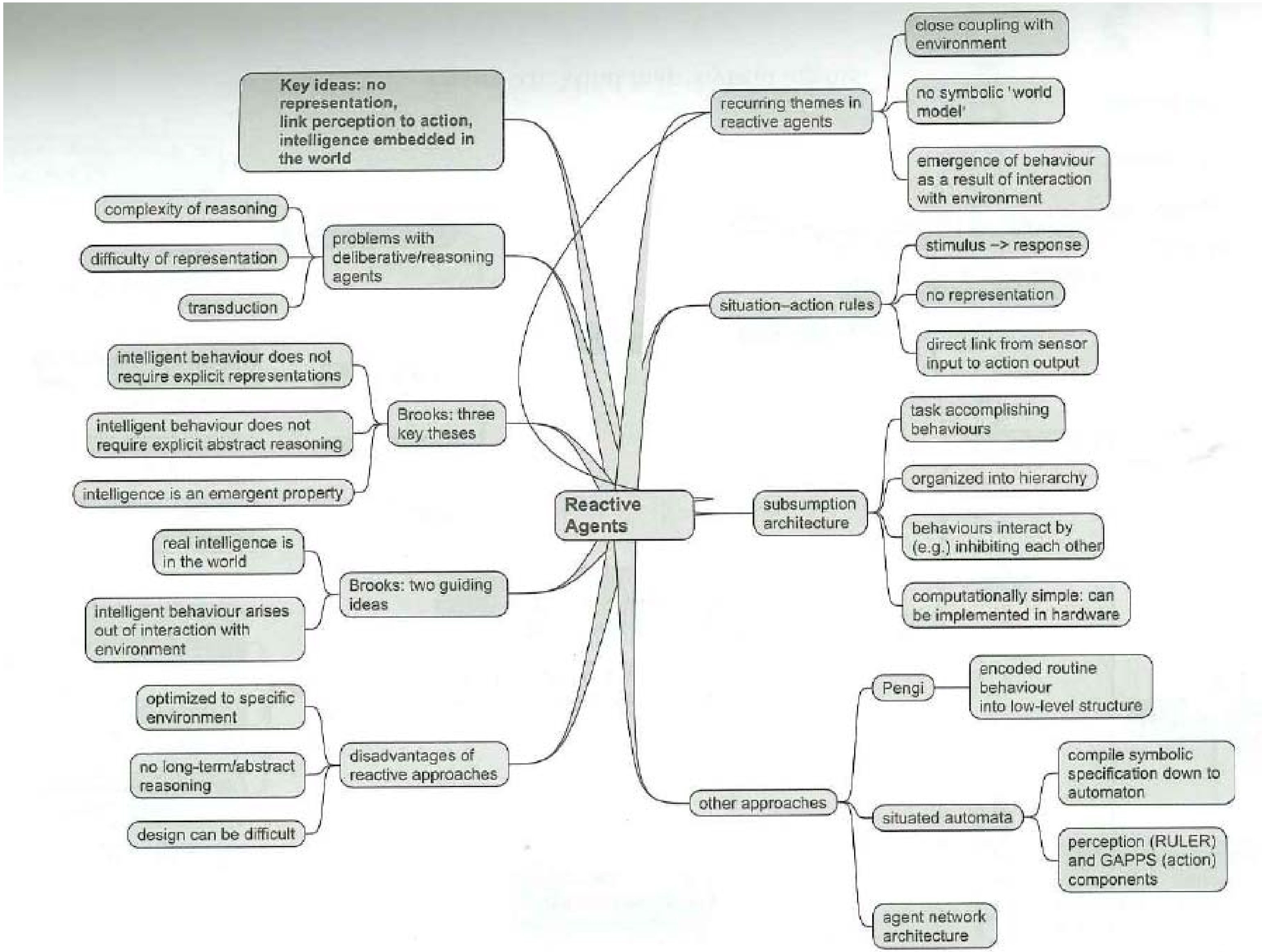
Critiques to hybrid architectures

- Lack of general design guiding methodologies
- Very specific, application dependent
- Unsupported by formal theories

Readings for this week

- M.Wooldridge: *An introduction to MultiAgent Systems* – *chapter 5 (reactive, hybrid)*
- A.Mas: *Agentes software y sistemas multi-agente: conceptos, arquitecturas y aplicaciones* – *chapter 2*
- Articles/book chapters on Moodle web site

Reactive agents, mind map (Wooldridge)



Hybrid agents, mind map (Wooldridge)