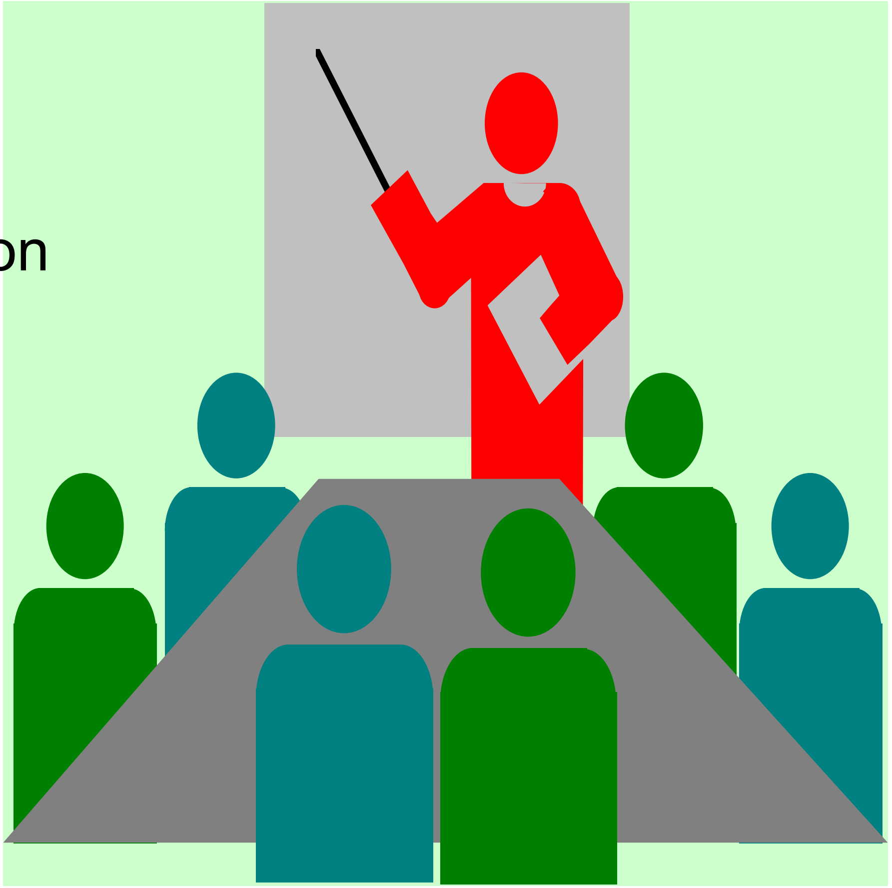
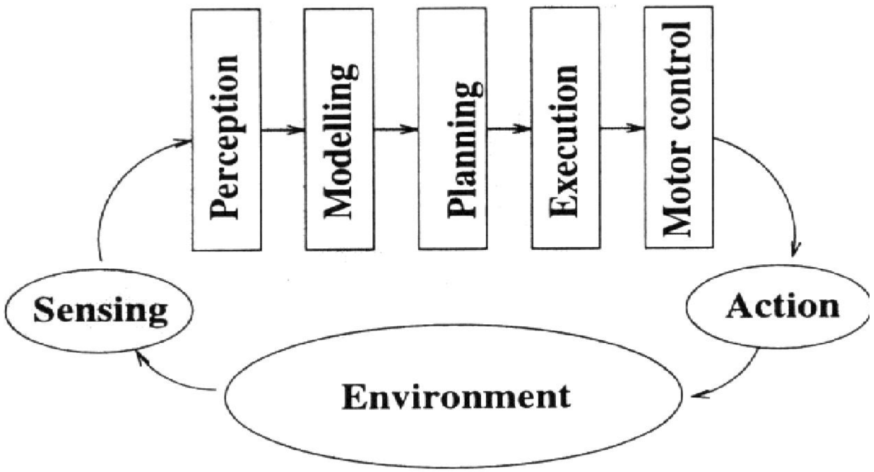
# Topic 10: Reactive & Hybrid Architectures





# 10.1 Introduction

## Deliberative Architecture

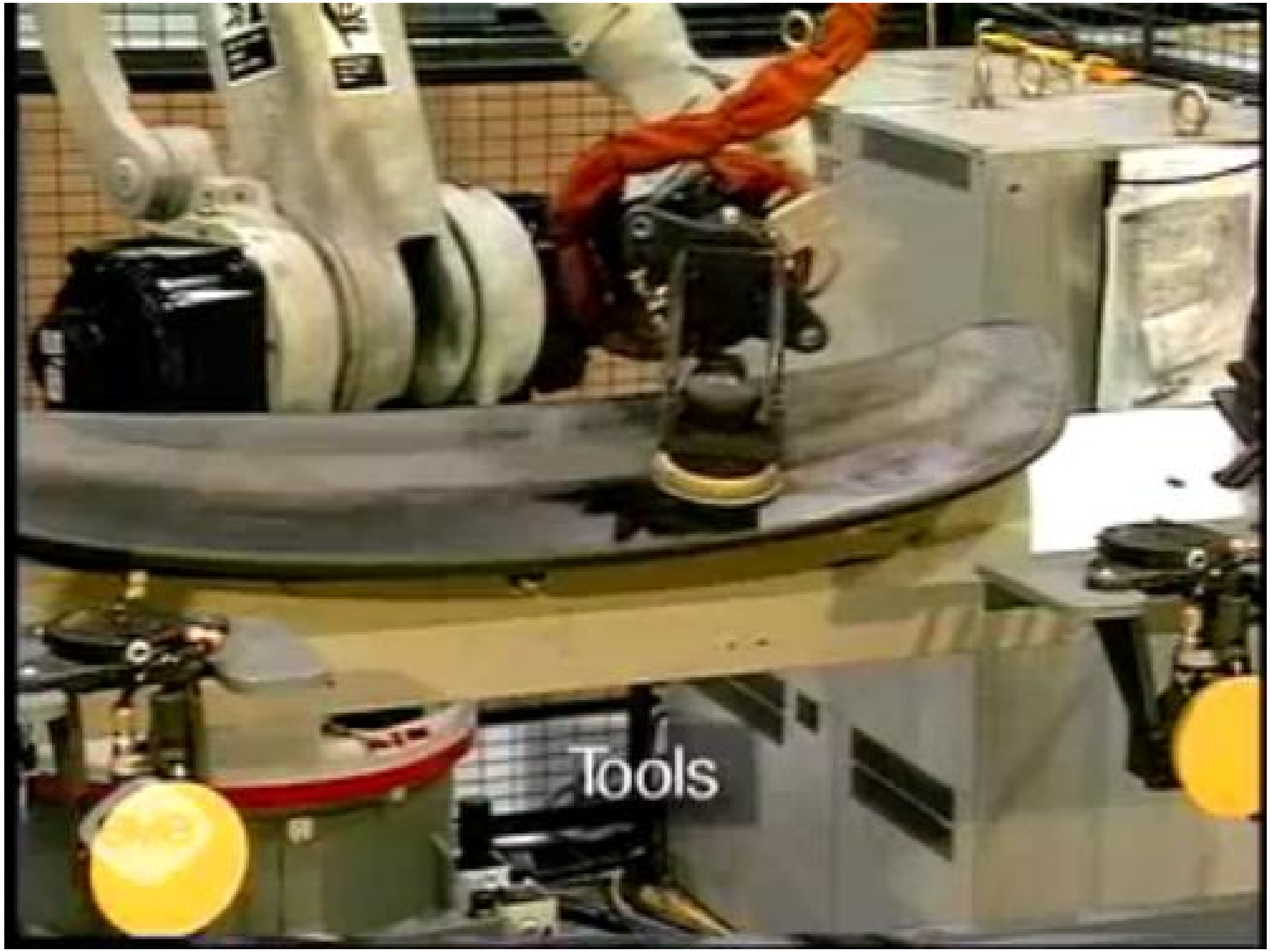



**Users**

| | | |
|---|---|---|
| Level 2 *Management* | • user interface (programming support / graphics)<br>• operating system environment<br>• resource allocation<br>• coordination | |
| Level 1 *Reasoning* | • decision making<br>• causal, temporal & geometric reasoning<br>• mission & path planning<br>• world modelling | |
| Level 0 *Device Interaction* | • multisensor fusion<br>• interpretation<br>• feature extraction<br>• pre-processing | • adaptive control<br>• force compliance<br>• robot dynamics<br>• robot kinematics |
| | **sensors** | **actuators** |

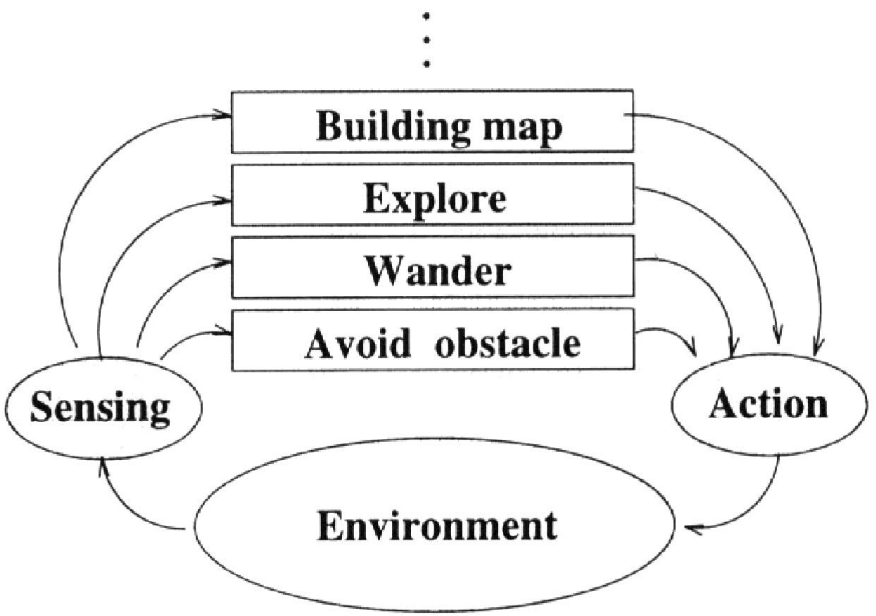
# 10.1 Introduction

Deliberative Architecture



# 10.1 Introduction

## Reactive/Behaviour Architecture




**Two types of Reactive architectures:**

➢ ***Subsumption***

*coordination:* competitive

encoding: discrete

➢ ***Schema-based***

coordination: cooperative

encoding: continuous

# 10.1 Introduction

Reactive/Behaviour Architecture

This robot detects a light and moves towards it. However, it detects an object on the way and then detour it.

# 10.2 Subsumption Architectures

**Key aspects in design**:

- Situatedness -- refers to the robot's ability to sense its environment and avoid the use of abstract representation.
- Embodiment -- insists that the robots be physical creatures and thus experience the real world rather than through simulation.

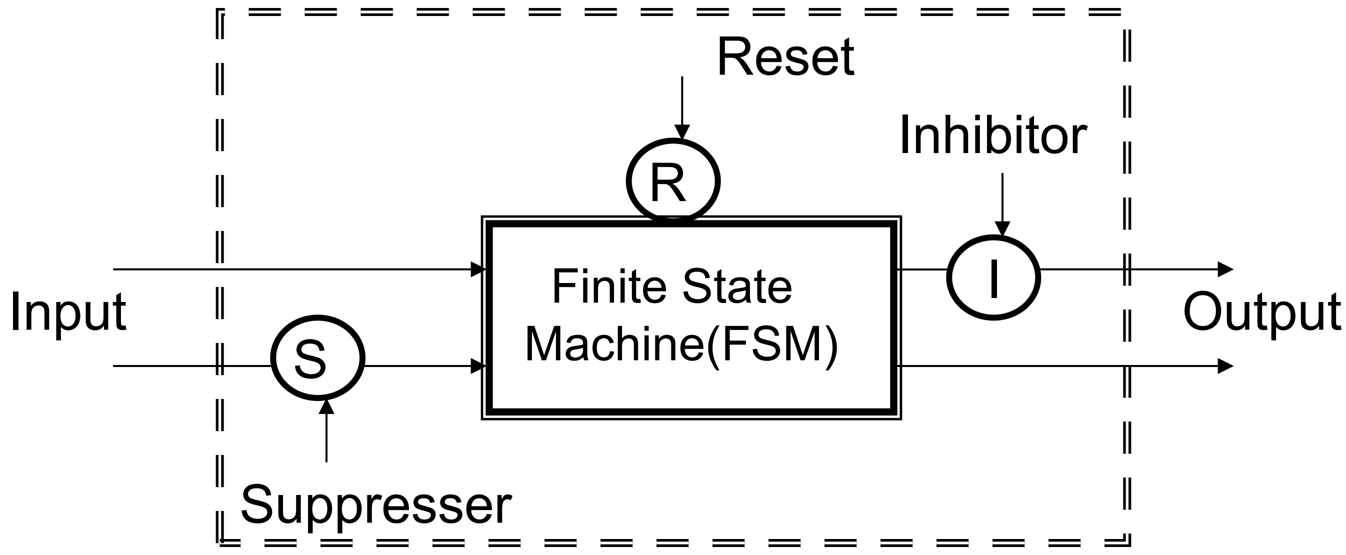
**Basic procedures**:

*Step 1*: Qualitatively specify the behaviour needed for the task.
*Step 2*: Decompose & specify the robot's independent behaviours as a set of observable disjoint actions.
*Step 3*: Determine the behavioural granularity and ground the resulting low-level behaviours onto sensors and actuators.

# 10.2.1 Behaviours in Subsumption

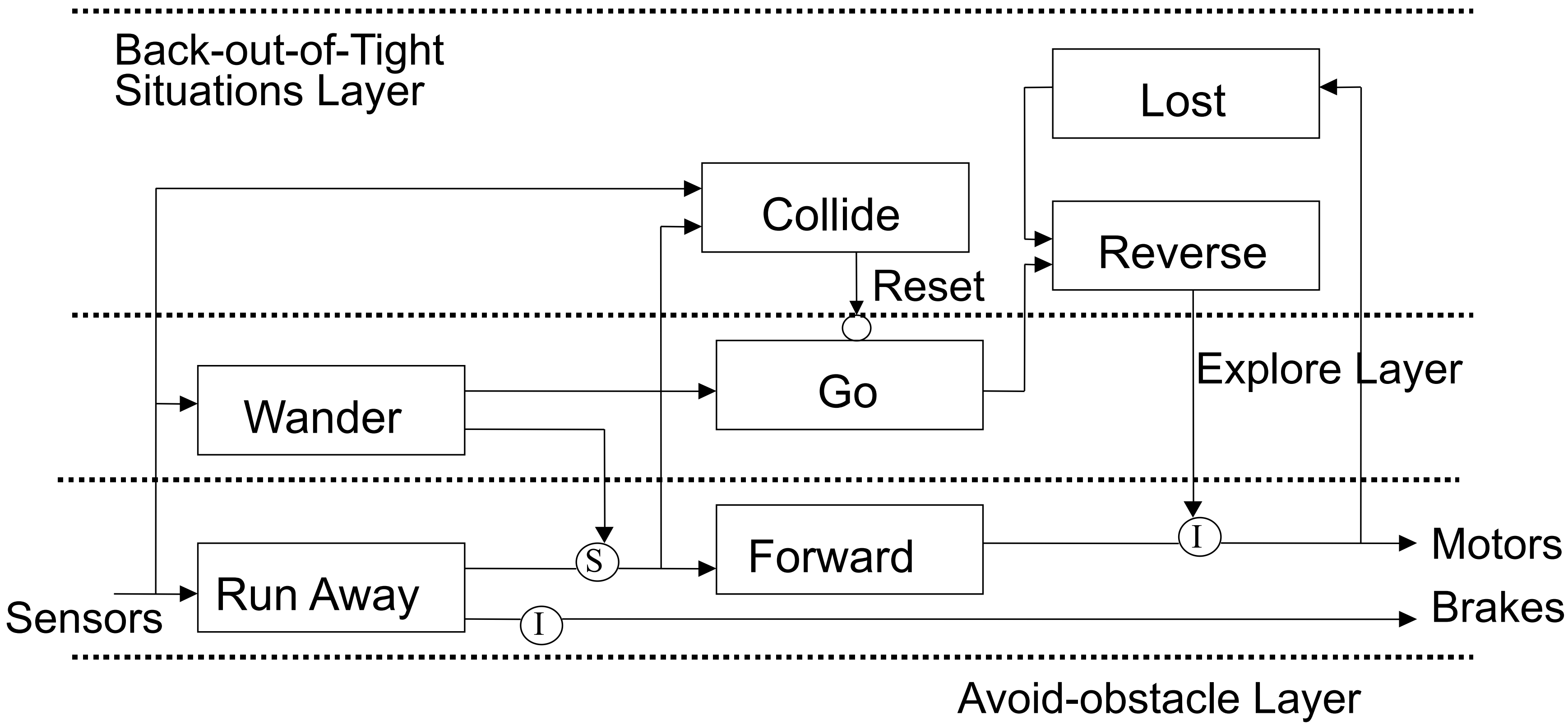
- Each behaviour is represented using an augmented finite state machine (*AFSM*).
- *AFSM* encapsulates a particular behavioural transformation function $\beta_i$
- All layers work on individual goals concurrently and asynchronously.

*An **A**ugmented **F**inite **S**tate **M**achine*



# 10.2.1 Behaviours in Subsumption

**AFSMs for a simple 3-layered robot (Brooks87)**



# 10.2.1 Behaviours in Subsumption

**Main features:**

- *Top layer* -- enables the robot reverse direction in particular tight situation where simpler avoidance and exploration behaviours fail to extricate the robot.
- *Middle layer* -- permits the robot to move in the absence of obstacles and cover large area.
- *Bottom layer* -- either halts or turns away from an obstacle, depending upon the input from the robot's infrared proximity sensors.

**Note**:

- Each *AFSM* performs an action and is responsible for its own perception of the world.
- There is no global memory, bus or clock.
- Each behavioural layer can be mapped onto its own processor.

# 10.2.2 Coordination in Subsumption

**Two coordination mechanisms:**

- Inhibition: used to prevent a signal being transmitted along an *AFSM* wire from reaching the actuators.
- Suppression: prevents the current signal from being transmitted and replaces that signal with the suppression message.

**Main features**:

- A priority-based arbitration is enforced.
- Communication is restricted: low baud rate & no handshaking.
- Message passing is implemented via machine registers.
- Inhibition prevents transmission.
- Suppression replaces message with suppressing one.
- Reset signal restores behaviour to original state.

# 10.2.3 Behaviour Language in Subsumption

- The behaviour language groups multiple processes into behaviours, shown in (a) below.

```
1. (whenever (received? Mess)
           (output out (+3 mess)))

2. (whenever (delay 0.5)
           (ping-sonar)

3. (whenever (< (infrared 3) (meters 1.5))
           (output velocity (meters 1.0))
           (delay 0.5)
           (output halt 't))
```

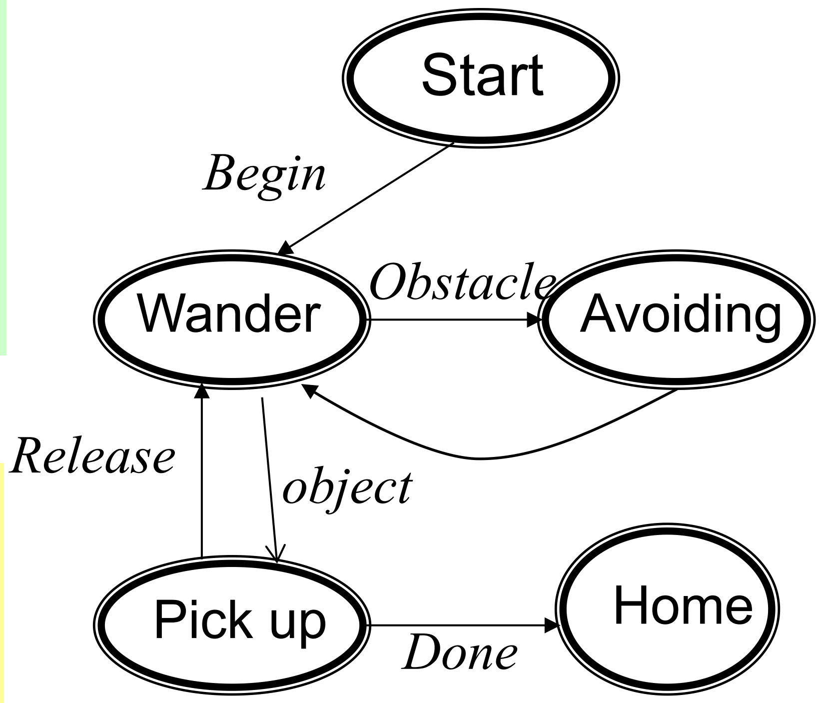
(a) *whenever* clause

- The building blocks for specifying a process are condition-action rules, called ***whenever*** clause in the behaviour language.
- There are message passing, suppression and inhibition both between processes within a behaviour and between behaviours.

# 10.2.3 Behaviour Language in Subsumption

***A Forage Example*** This task includes a mobile robot

- move away from a start position
- wonder & look for attractive objects
- move to attractive object & pick it up
- avoid obstacle if detected.
- return to the home base

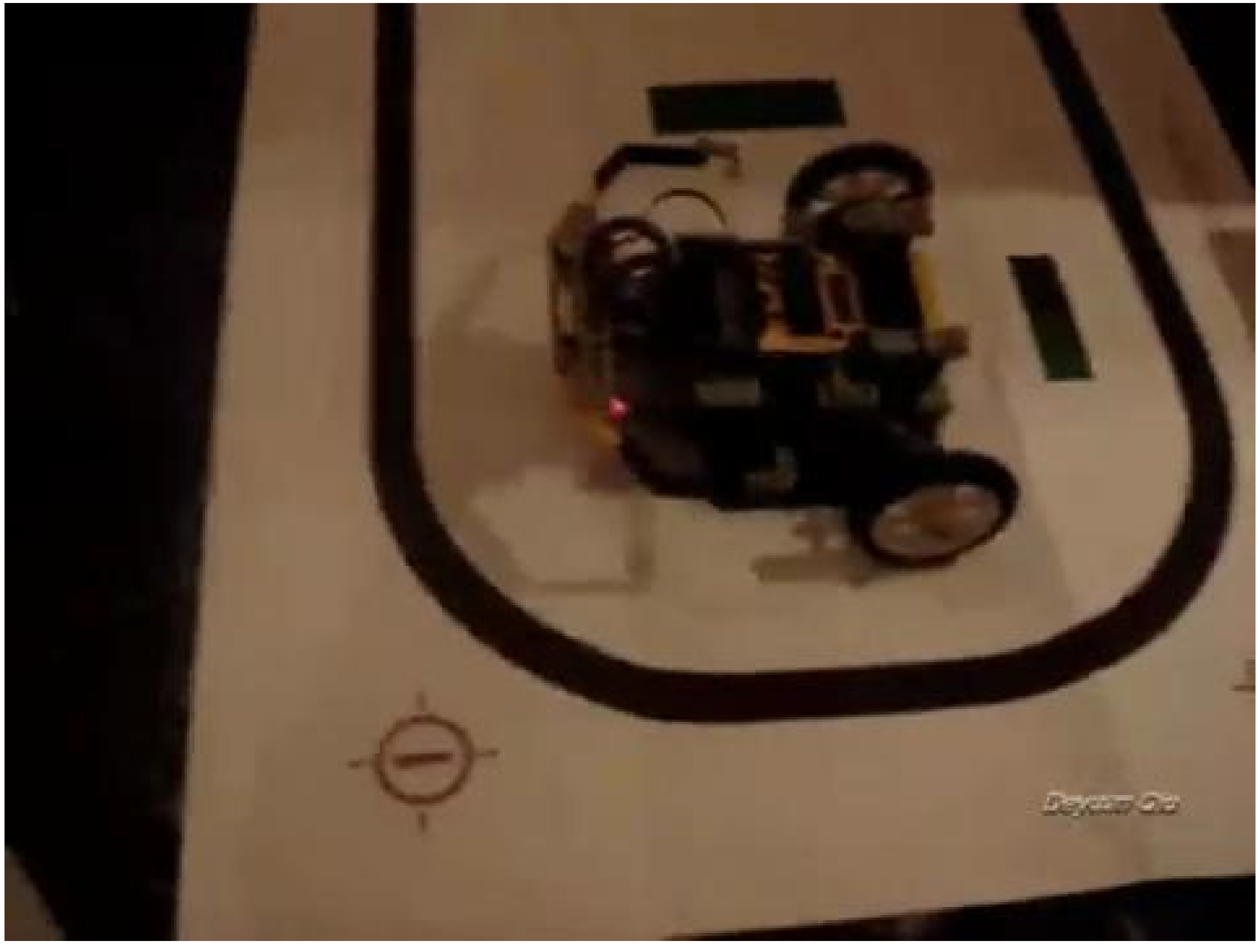



High-level behaviours

- *Wander* -- move around to search
- *Avoiding* – to avoid any obstacle
- *Pick up* – to catch and release the object;
  return to home if done; or
  wander again if it is wrong

# 10.2.3 Behaviour Language in Subsumption

Line following robot



# 10.3 Schema-based Architectures

**Main features:**

- Schema expresses the relationship between sensing and motor action.
- A schema stores both how to react and the way reaction can be realised.
- Schema provides a behaviour primitive that is a basic building block for a more complex behaviour.
- Schemas act concurrently as individual distributed agents.

**Comparing with Subsumption:**

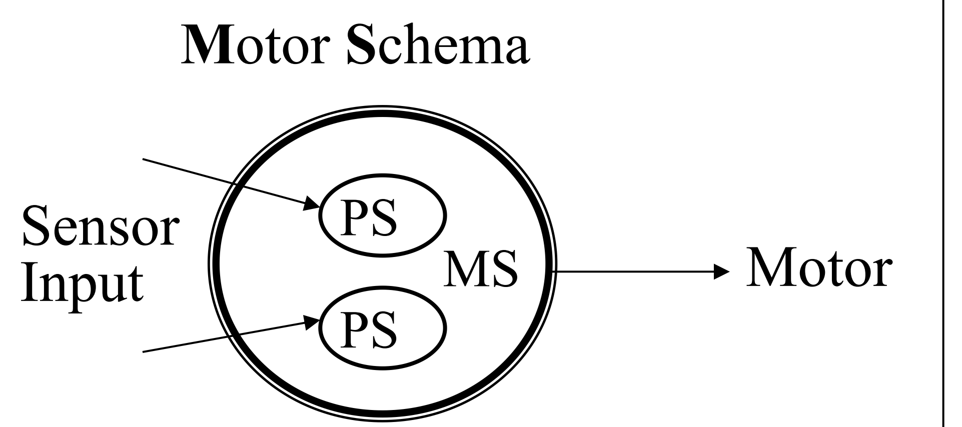
- No predefined hierarchy exists for coordination.
- Behaviour responses are all represented in a single uniform format: *vectors* generated by potential fields, a continuous response encoding.
- Coordination is achieved through cooperative means by vector addition.

# 10.3.1 Perception-action schemas

- Schema-based behaviours are relative large grain decomposition.
- A perception schema is embedded within each motor schema, providing suitable stimuli, i.e. *action-oriented perception.*
- Perception schemas can be recursively defined for a meaningful unit.

**Typical motor schemas:**

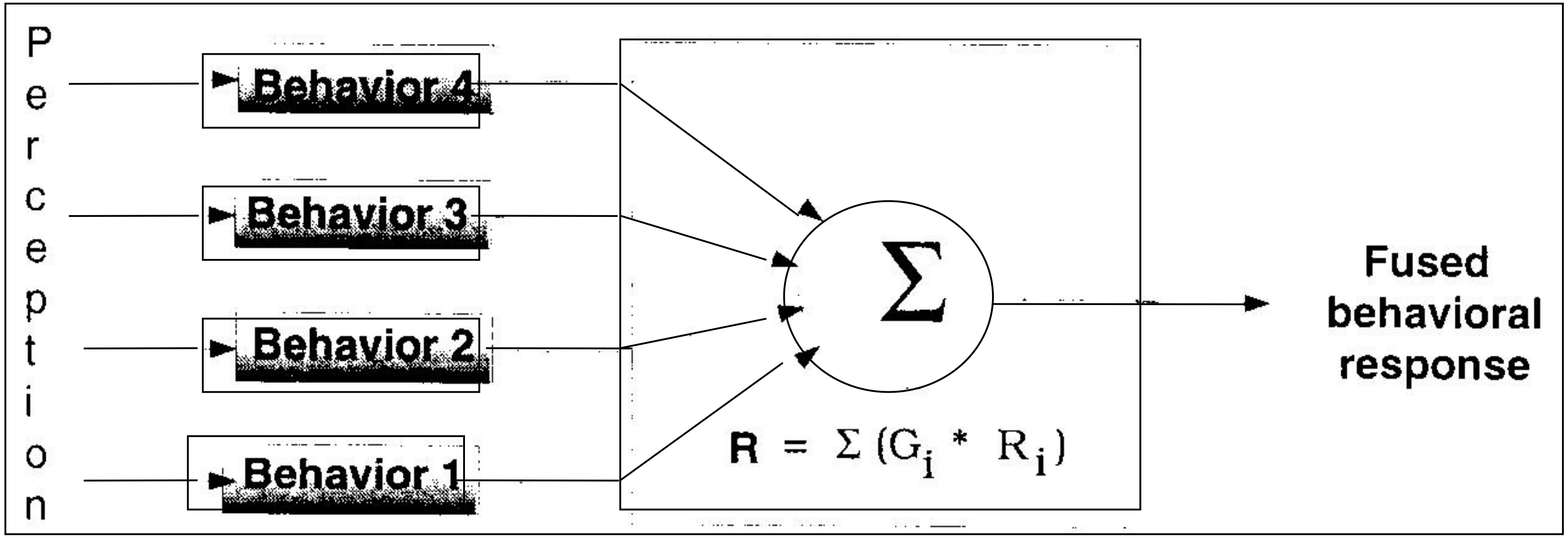
- Move-ahead
- Move-to-goal
- Stay-on-path
- Move-up/down
- Fellow-the-leader
- Avoid-obstacle
- Dodge
- Escape
- Probe
- Dock
- Noise
- Avoid-past




PS - *Perception Schema*

# 10.3.2 Schema-based Coordination

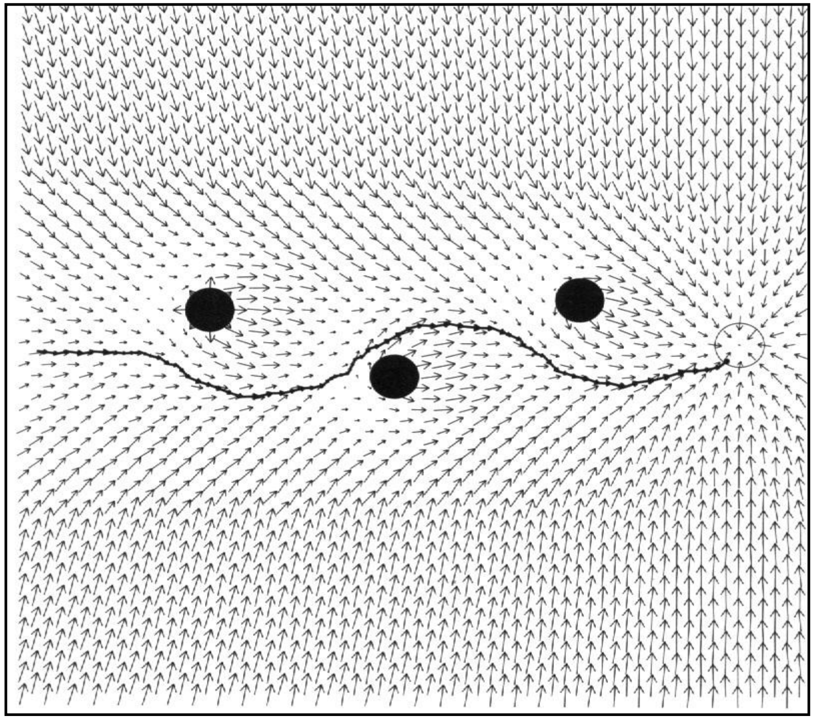
- All active behaviours contribute to the robot action according to the gain **G**.
- The gain remains constant in a non learning & adaptation system.
- A summation vector is sent to the robot for execution.



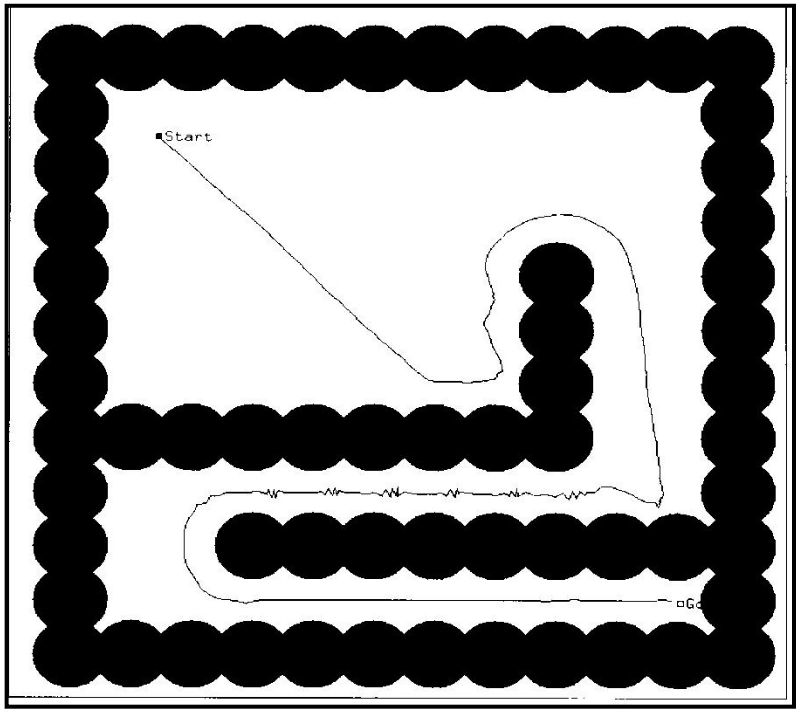

*Vector summation*

**Examples for vector summation:**

(a) Coordination between 4 schemas



(b) Navigation in a maze with *avoid-past*



# 10.4 Hybrid Architectures

Advantages of reactive & deliberative systems

- ***Reactive*** behaviour-based robotic control can effectively produce robust performance in complex & dynamic environments.
- ***Deliberative*** systems permit representation knowledge to be used for planning when their environment has restricted uncertainty.

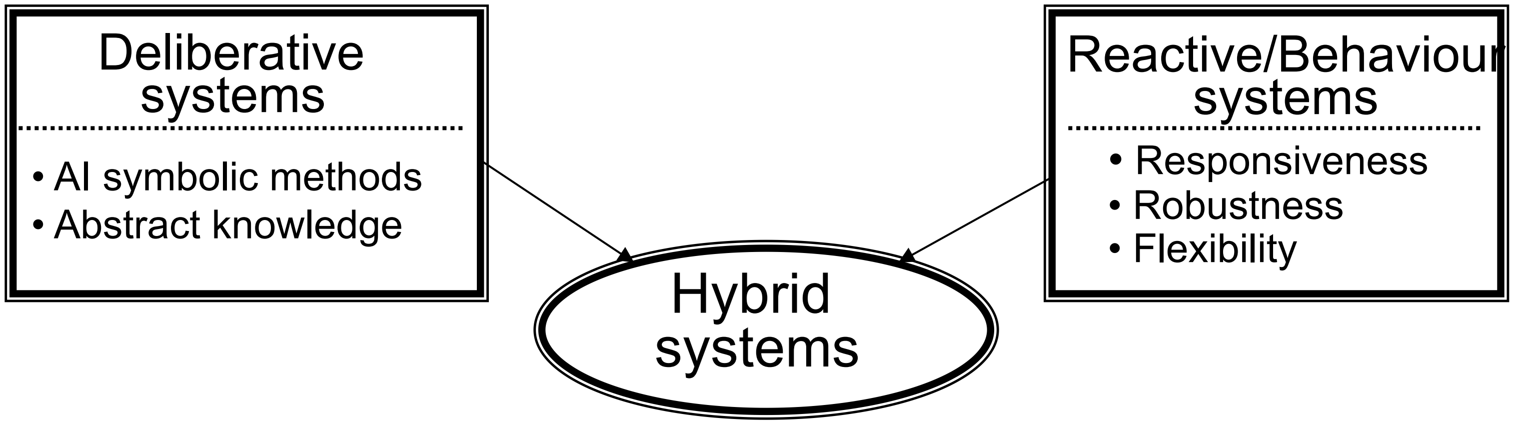
Limitations of reactive & deliberative systems:

- ***Purely reactive*** robotic systems are not adequate for all real world applications, i.e. no accurate positioning & world knowledge.
- ***Purely deliberative*** robotic systems are unable to operate in a dynamic and unknown environment effectively.

Solution:

- ***Hybrid system*** = ***reactive system*** + ***deliberative system***

# 10.4.1 Key factors to hybrid architecture




Benefit:

- Behavioural and perceptual strategies can be represented as modules and configured to match various missions & environments.
- A priori world knowledge, when available & stable, can be used to configure or reconfigure these behaviours effectively.
- Dynamically acquired world models can used to prevent certain pitfalls to which behaviour-based systems are subject.

# 10.4.2 Interface Strategy

**Interface strategy for hybrid architectures**

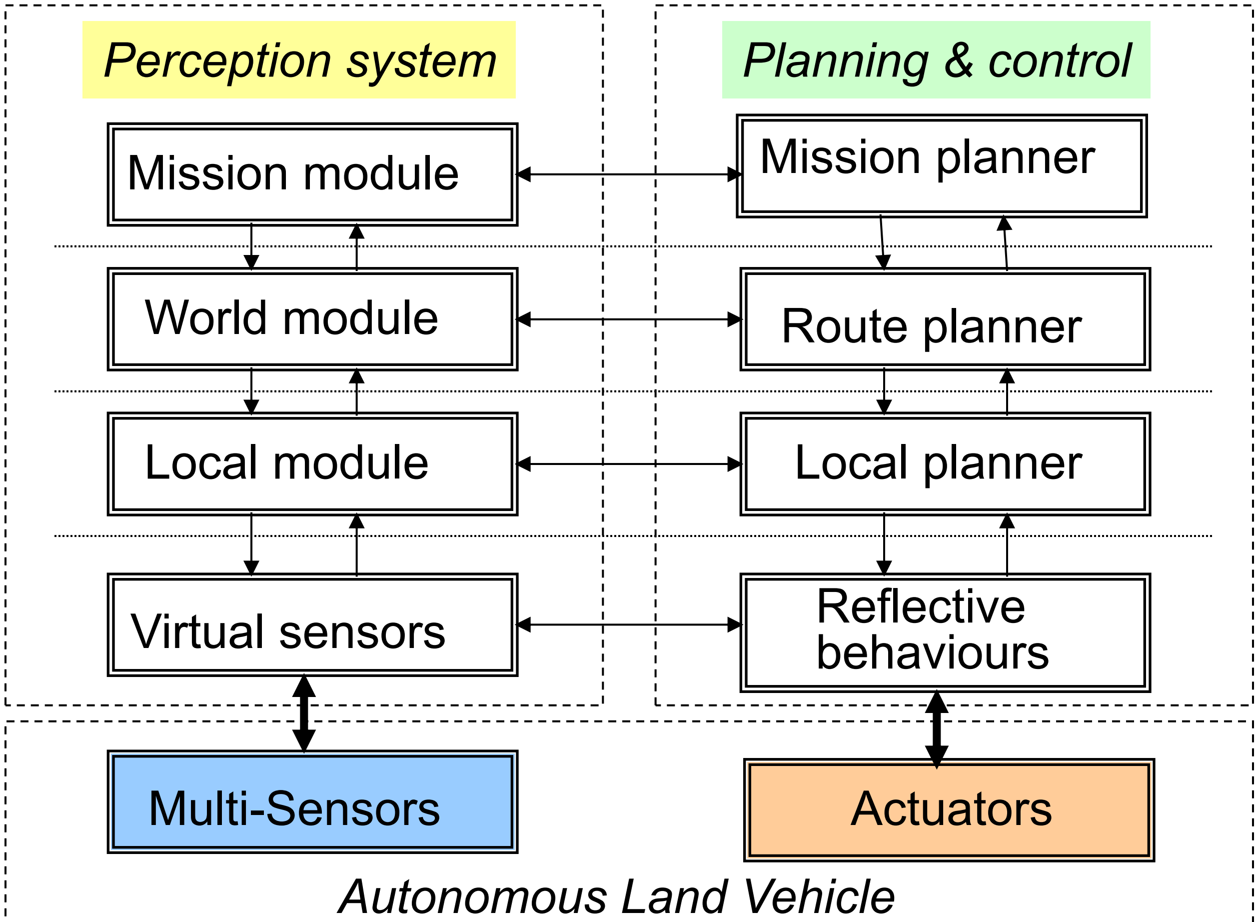
- **Selection**: Planning is viewed as configuration.
  The planner determines the behaviour composition and parameters used during execution.
- **Advising**: Planning is viewed as advice giving.
  The planner advises courses of actions but the reactive agent determines whether it may or may not use.
- **Adaptation**: Planning is viewed as adaptation.
  The planner continuously alters the ongoing reactive component in light of changing conditions within the world and task requirements.
- **Postponing**: Planning is viewed as a least commitment process.
  The planner defers making decisions on actions until as late as possible. This enables recent sensor data to provide a more effective course of action.

# 10.4.3 Typical Hybrid Architectures

**Hybrid architecture for robot navigation**

# 10.4.3 Typical Hybrid Architectures

**ALV: the Autonomous Land Vehicle** *(DARPA, USA, 1987-1990)*



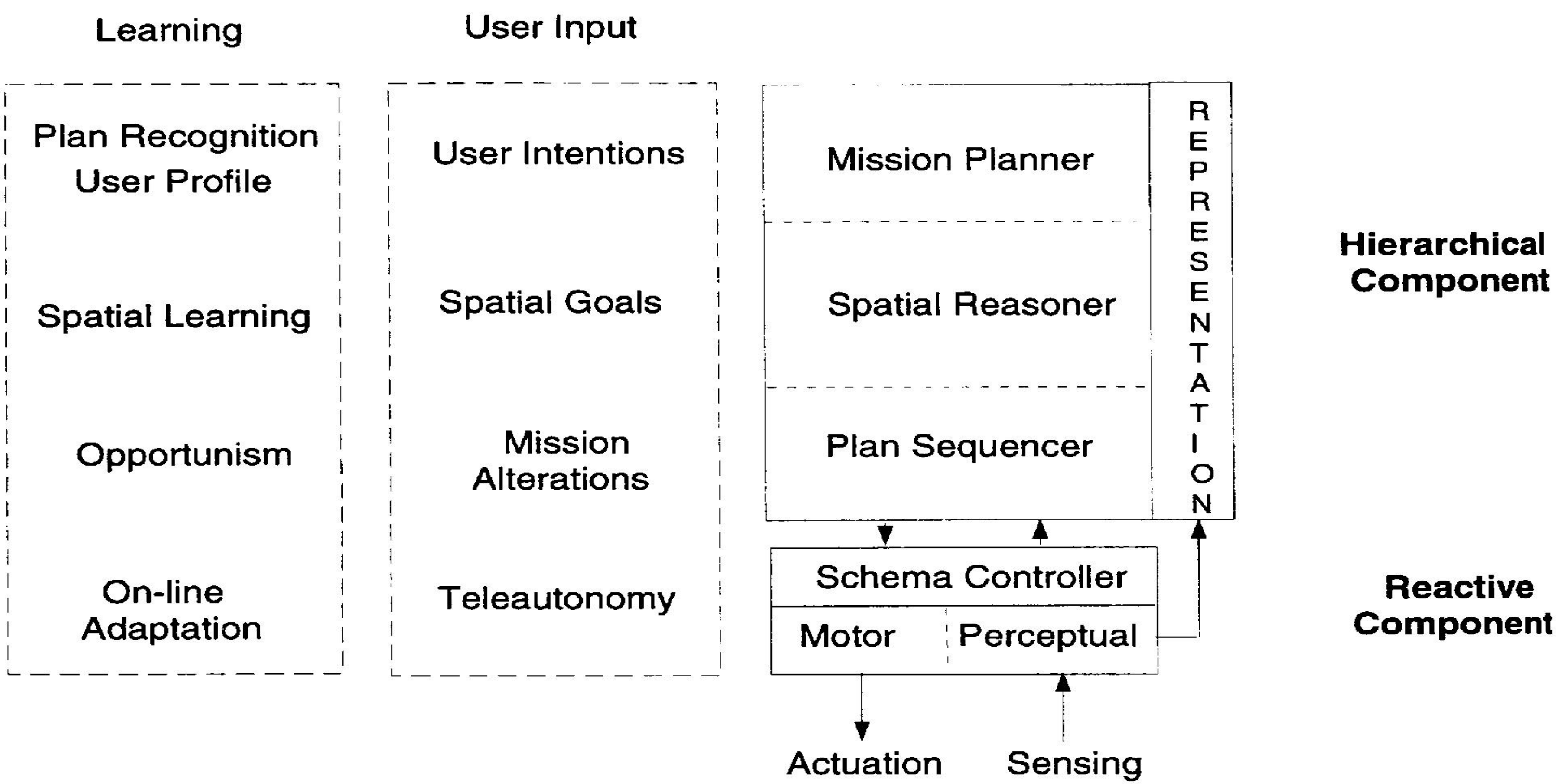
# 10.4.3 Typical Hybrid Architectures

**Main features:**

- At the highest level, the mission planner defines mission goals & constraints.
  - -- Mission goals & constraints are passed down to the second level.
  - -- The mission module configures sensors & predicts the system performance.
- At the second level, the map-based planner uses digital maps to plan the best route which satisfies the mission constraints.
  - -- The planned route is passed down to the local planner in a symbolic form.
  - -- The world module predicts landmarks to confirm the route & update.
- At the third level, the local planner executes the symbolic route plan.
  - -- It selects and monitors behaviour activities in the lower level.
  - -- The local module for perception performs data fusion to detect objects.
- At the lowest level, the knowledge assimilation is minimised in order to provide the fastest possible vehicle response.
  - -- Virtual sensors detect very specialised environment features.
  - -- Reflexive behaviours use virtual sensor data to provide real-time control.

# 10.4.3 Typical Hybrid Architectures

**Aura (Autonomous Robot Architecture)** (Arkin 1986)



# 10.4.3 Typical Hybrid Architectures

**Main features of Aura:**

- A traditional hierarchical planning system

  -- *Mission planner*: establishing high-level goals and the constraints.
  -- *Spatial reasoner*: uses carto-graphic knowledge in LTM to construct a sequence of navigation path legs towards the goal.
  -- *Plan sequencer*: translates each path leg into a set of motor behaviours for execution. (Rule-based & finite state)

- A behaviour-based reactive system

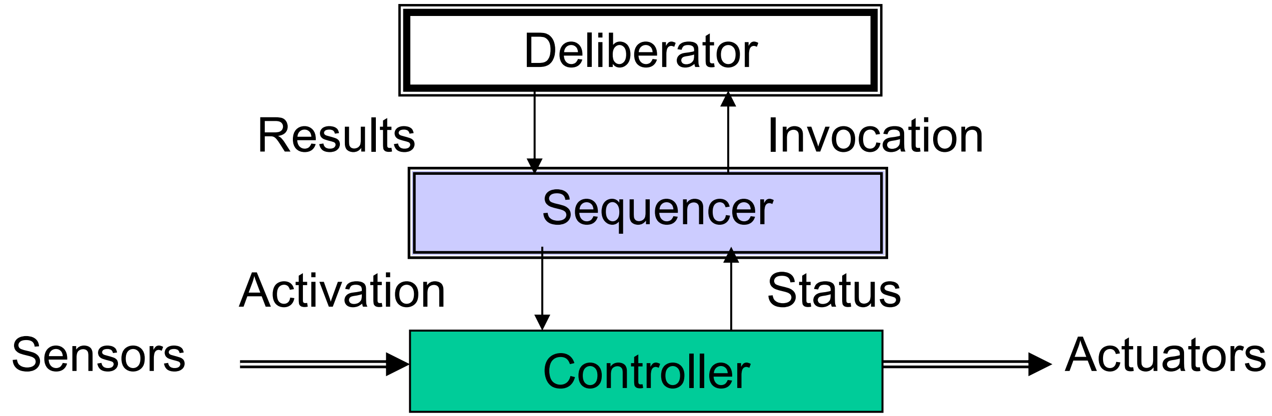
  -- *Schema manager*: responsible for controlling & monitoring the behavioural processes at run time.
  -- *Each motor schema* is associated with a perceptual schema.
  -- *The schemas* operate synchronously, transmitting their results to a process (move-robot) for summation, then to low-level control system for execution.

# 10.4.3 Typical Hybrid Architectures

**Staminode:**

# 10.4.3 Typical Hybrid Architectures

**Atlantis for advising** (Gat 1991, at JPL)




- A 3-layer hybrid system

-- ***Deliberator*** handles planning (search-based algorithms) and world modelling. It produces plans for the sequencer to execute, and responses specific queries from it.
-- ***Sequencer*** deals with initiation & termination of low-level control activities, supplies parameters for the behaviour, and addresses reactive-system failures.
-- ***Controller*** is in charge with collecting primitive actions, tightly coupling sensors to actuators for real-time control.

# 10.5 Learning in Behaviour-based Robotics

## Learning methods

- Numeric or symbolic:
  - -- Symbolic learning includes symbolic structures such as logical assertions, production rules and semantic networks.
  - -- Numeric learning manipulates numeric quantities such as Neural Network and statistical methods.

- Inductive or deductive:
  - -- Inductive learning generalises learning results gathered from experiments. *(No prior belief/knowledge*)
  - -- Deductive learning produces a more efficient concept from an initial one originally provided to the robot.

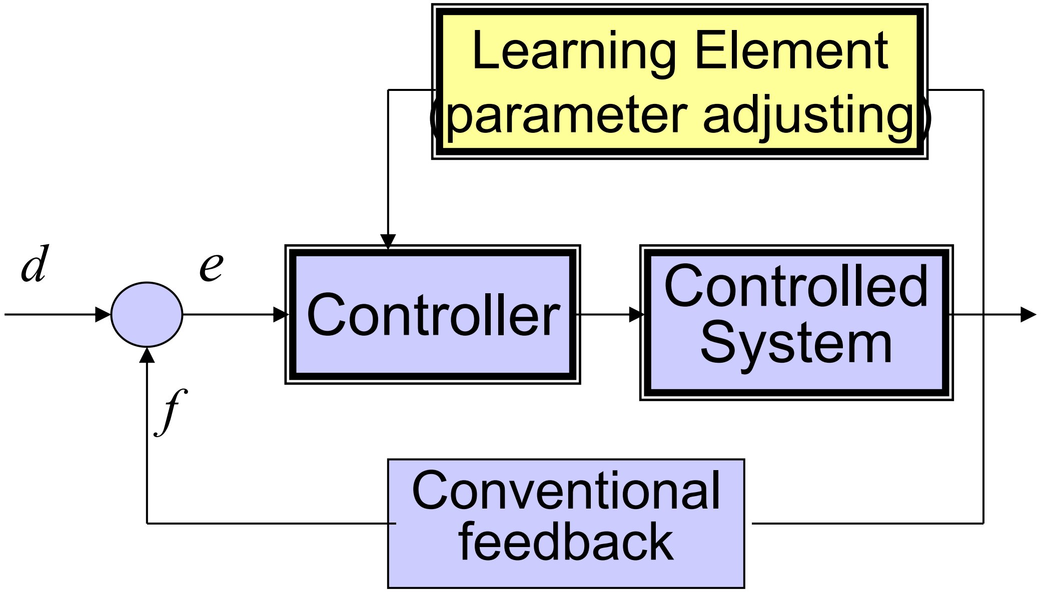
- Recursive or batch:
  - -- Recursive learning is conducted continuously & on-line when sensor information is made available.
  - -- Batch learning is implemented after a large body of experience or data.

# 10.5 Learning in Behaviour-based Robotics

**Learning** enables a robot system to achieve adaptive behaviour in the real world.

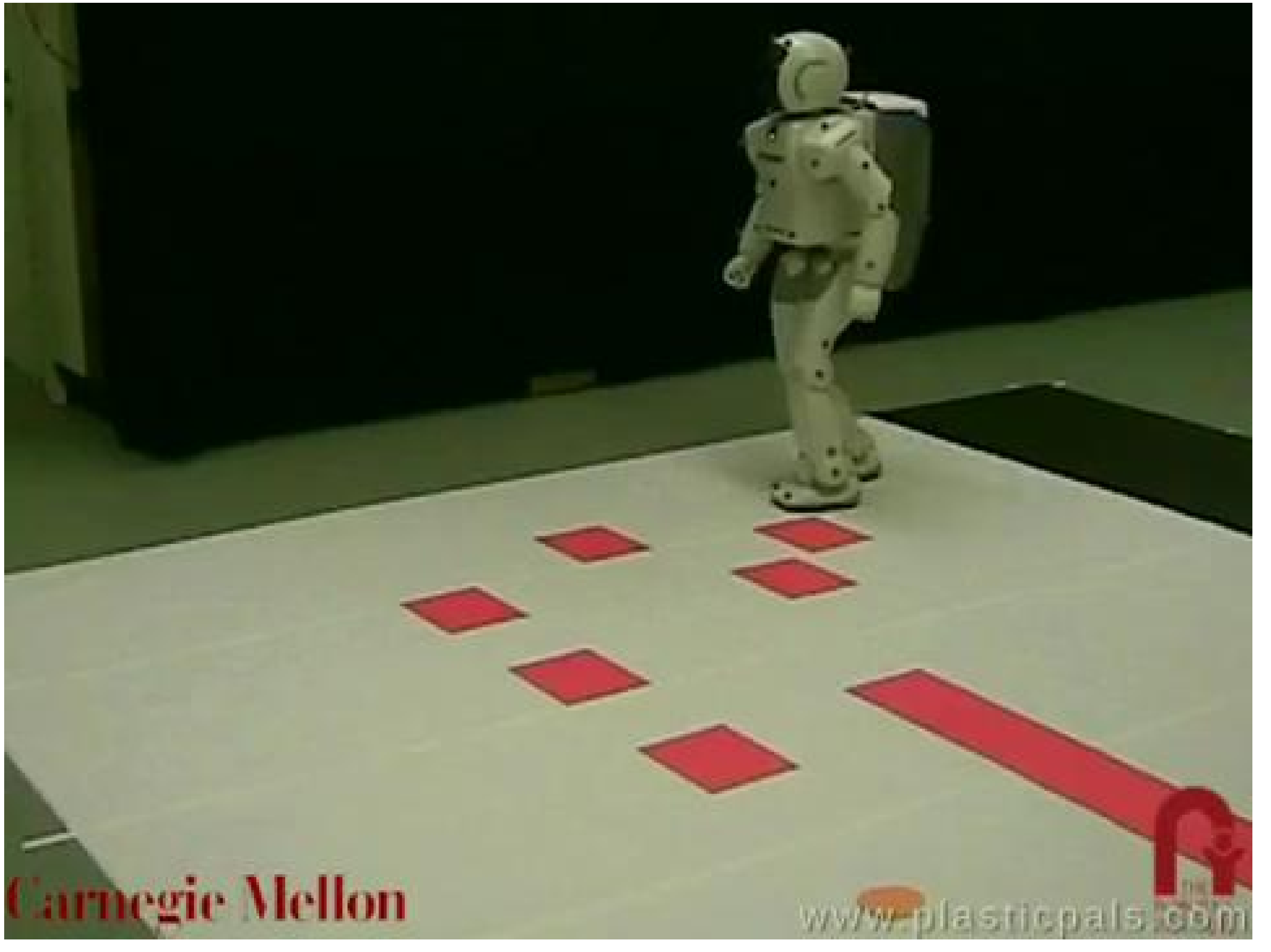
**Types of adaptation**

- *Behaviour adaptation* -- Robot's individual behaviours are adjusted relative to one another.
- *Evolutionary adaptation* -- Descendants change over long time based on success/failure of their ancestors in the environment.
- *Sensor adaptation* -- Robot's perceptual system becomes more attuned to its environment.



# 10.5 Learning in Behaviour-based Robotics

**ASIMO** learns to avoid moving obstacles



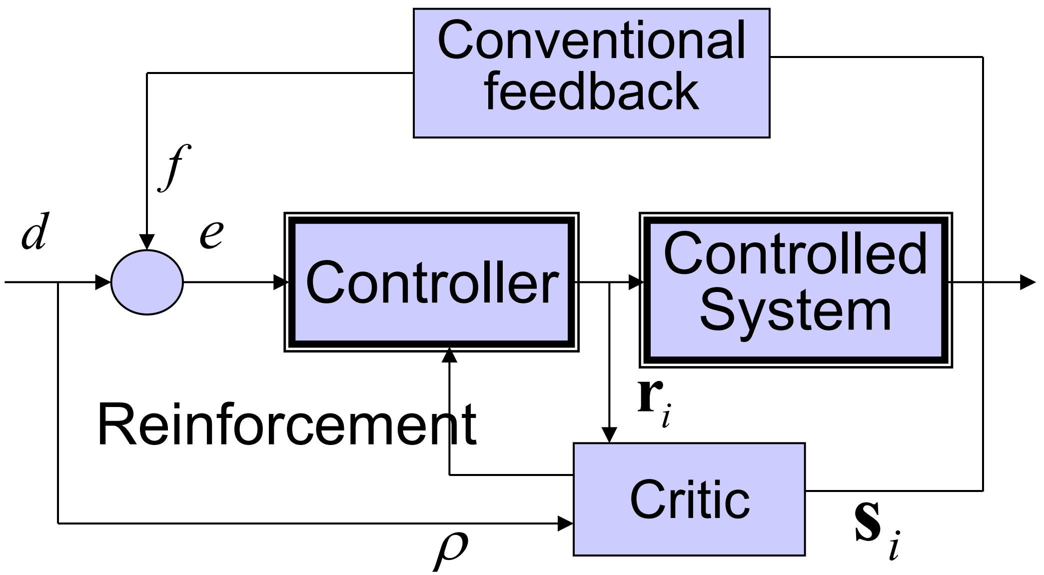
# 10.5 Learning in Behaviour-based Robotics

Reinforcement learning

It is a numeric, inductive and recursive learning process.

Main features

- **Critic** applies reinforcement to the control system in light of its evaluation.
- A reward or punishment is applied based on the results evaluated by Critic module.
- The reward/punishment is either logic (pass/fail) or numeric (complex real value), according to the quality of the behaviour response.




*Reinforcement learning system*

# 10.5 Learning in Behaviour-based Robotics

**<u>Real robot learning</u>**

# 10.5 Learning in Behaviour-based Robotics

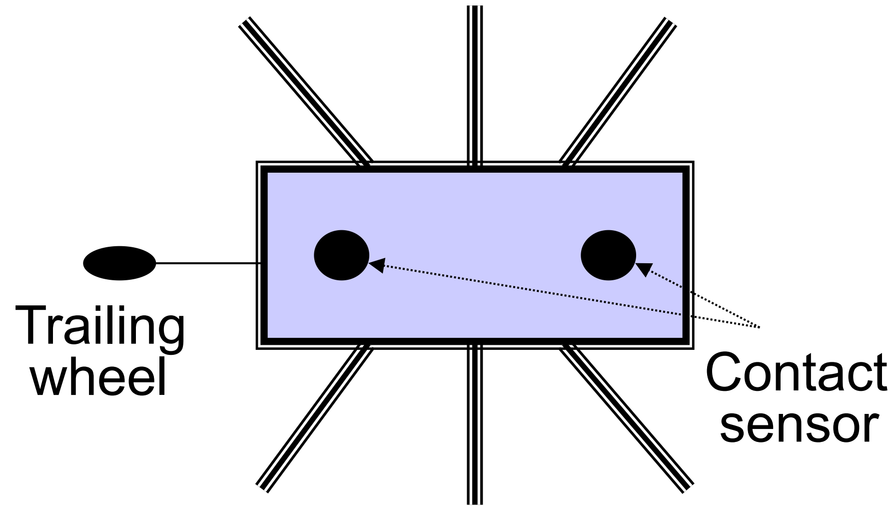
**A robot hexapod: Genghis**

- A rule-based Subsumption architecture for the robot controller.
- 13 high-level behaviours: 6 swing forward, 6 swing backward, 1 balance
- 2 contact sensors for feedback (binary)
- A trailing wheel to measure distance

Task -- learn to move forward

Feedback

- Negative feedback -- one of touch sensors makes contact with ground.
- Positive feedback -- its measurement wheel indicates the forward motion.




***A hexapod***

Reinforcement

- to alter the precondition list of Subsumption behaviours on the criteria of their being: *relevant* or *reliable*.

# 10.5 Learning in Behaviour-based Robotics

***Principle:***

- If the behaviour is relevant but not reliable, a different perceptual condition is monitored to see if it is responsible for un-reliability.
- If yes, the behaviour rule's pre-condition list will be altered, and new statistics are gathered correlating the robot's performance.

***The results:***

- Using only negative feedback with the balance and 6 swing forward behaviours, the robot learned to adopt a stable tripod stance, keeping 3 legs on the ground all the time (the middle leg from one side). No forward movement was achieved.

***however,***

- Using both positive and negative feedback resulted in robot's walking using the tripod gait, in which it alternatively swings two sets of 3 legs forward as the robot moves.

# 10.5 Learning in Behaviour-based Robotics

***ASIMO is learning***

# 10.6 Conclusions

- Behaviour-based architectures emphasis on a tight coupling between sensing and action, but differ in behaviour decomposition, coordination, response encoding.

***Subsumption design:***

- Multiple behaviours can run asynchronously and in parallel.
- Custom behaviours can be created for specific task-environment pairs.
- Less flexibility and non-modularity

***Motor Schemas design***:

- Motor schemas are a software-oriented dynamic reactive architecture.
- Vectors serve as the continuous coordination strategy.
- Action-selection are dynamic and arbitration-based.

- Hybrid architectures include hierarchical integration, planning to guide reaction, and coupled planning and reacting. Three layers are very popular.
- Learning plays an important role for robots to achieve adaptive behaviours.