

# Robot Architectures

Benjamin N. Passow

[benpassow@dmu.ac.uk](mailto:benpassow@dmu.ac.uk)  
De Montfort University, UK

# Overview

- Architectures
  - Reactive Control
  - Deliberative
  - Motor Schema
- Examples:
  - Finite state machines
  - Subsumption Architecture

# Architectures



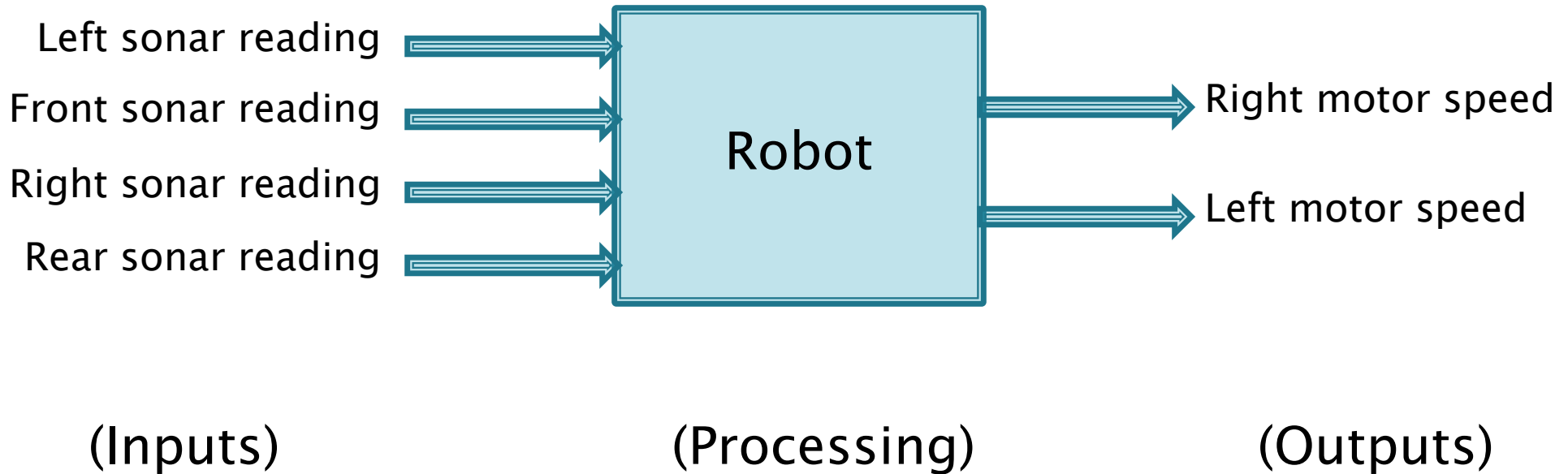
# Architectures

- A single robot is a highly complex system
  - Multitude of sensor and actuators
  - Dynamic nature – events dictate actions
  - Multiple, possibly conflicting task competing for attention
- Architectures are needed to co-ordinate a complete robotic system
  - Give more rational behaviour
  - Prevent a robot contradicting itself

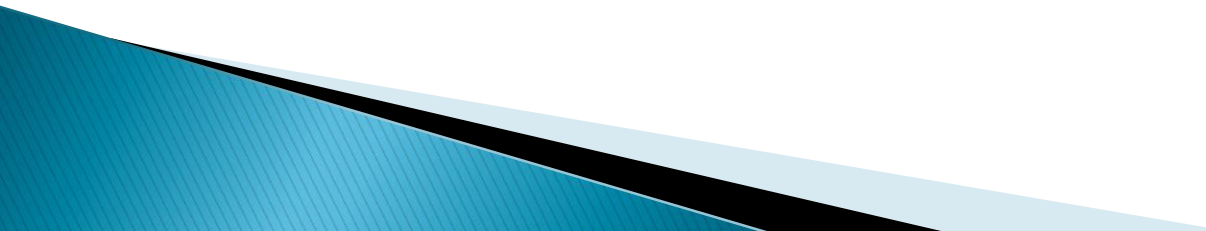
# Reactive Control

- A robot is complex system
- $M$  inputs
- $N$  outputs
- Complex non-linear mapping between input and output
- Additional temporal dimension
- Very fast – near instantaneous
- Problems:
  - Tasks lack depth
  - Robot has zero knowledge of what it has been doing over time

# Reactive Control

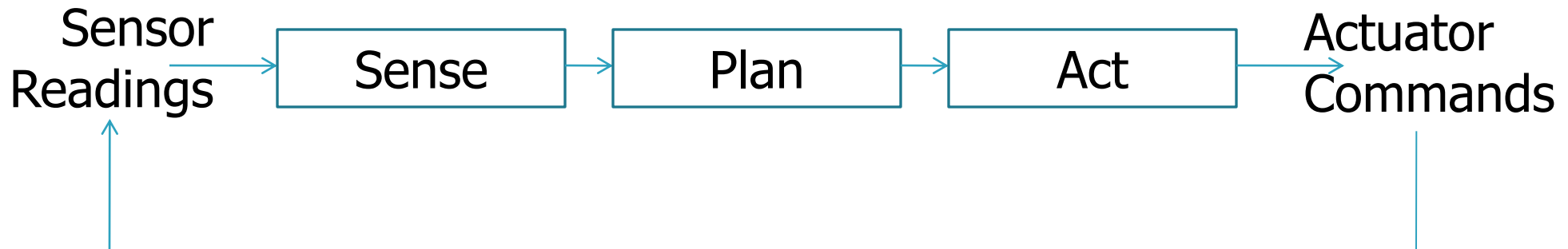


# Reactive Control

- A few processing options:
    - PID
    - Rule-Based Fuzzy System
    - Neural Network
- 

# Deliberative Control

- Sometimes called model-based architectures
- At the heart of deliberative control is the notion of a sense-plan-act structure



# Deliberative

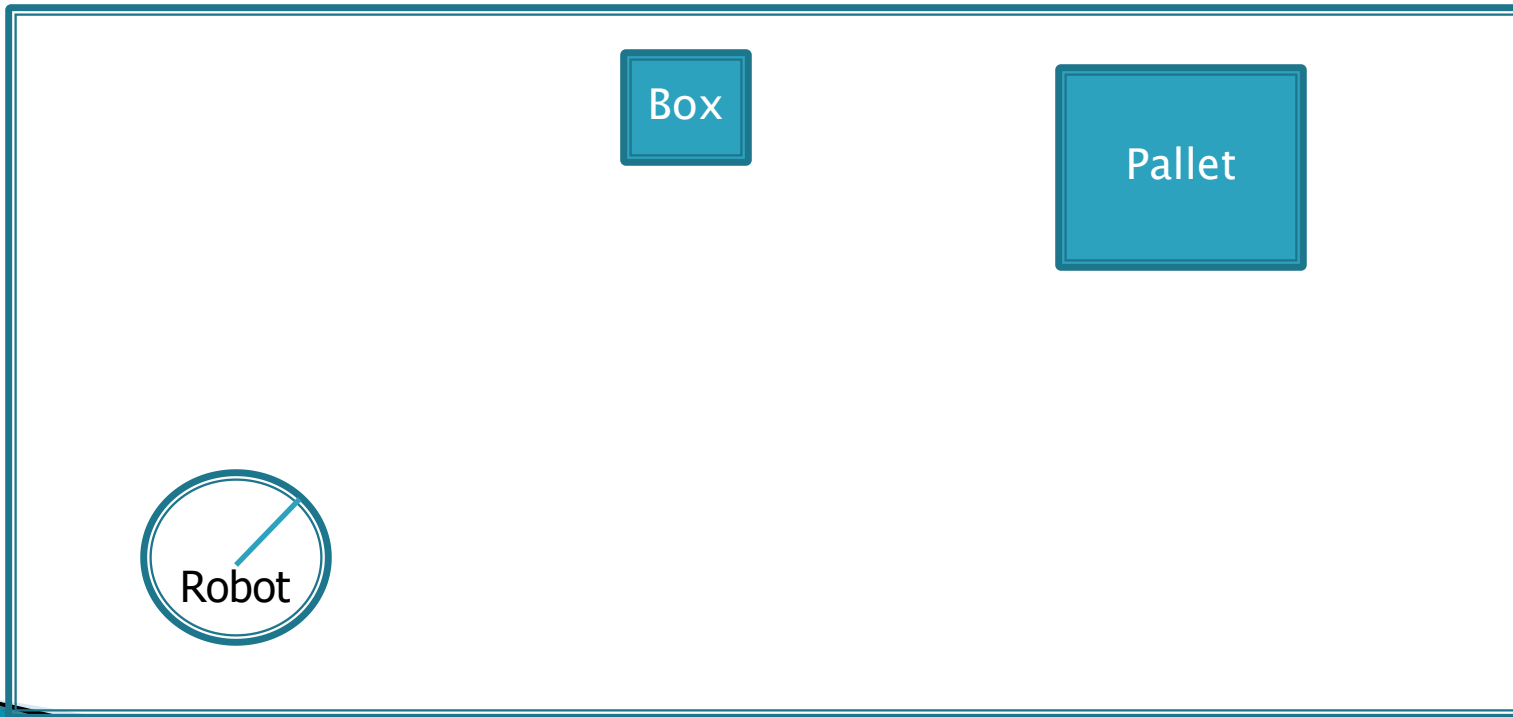
- Sense:
  - Get a set of sensor readings
  - Use these readings to perceive the environment
  - Update the environment model
- Plan:
  - Try out scenarios according to the environment model
  - i.e. Search for a solution
  - Plan out the actuators commands to realise this solution
- Act:
  - Realise the actuator commands that have been planned

# Deliberative

- ▶ **Advantages**
  - Reasons about contingencies
  - Computes solutions to the given task
  - Goal-directed
- ▶ **Problems**
  - Perception is itself a highly challenging task
  - Solutions fail in the presence of uncertainty
  - Reacts slowly, if at all, to dynamic events

# Deliberative Example

- ▶ Pick up a box and place it on a pallet:



# Deliberative Example

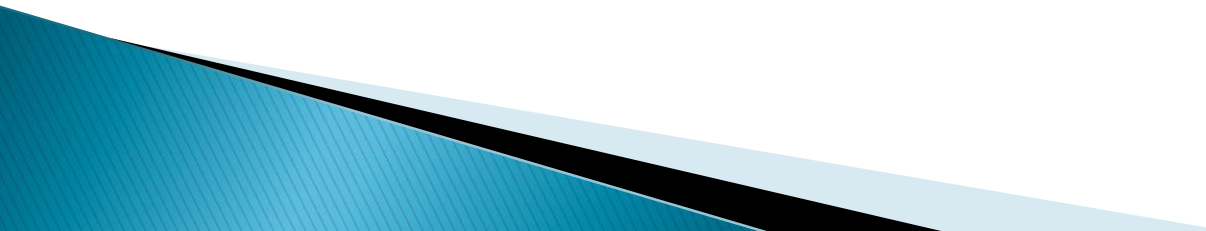
- ▶ **Sense:**
  - Localization – where am I?
  - Perception of the box and pallet – where are they?
- ▶ **Plan**
  - How do I get from here to the box?
  - How do I pick up the box?
  - How do I get to the pallet?
  - How do I put the box down?
- ▶ **Act**
  - Action my plan

# Deliberative Example

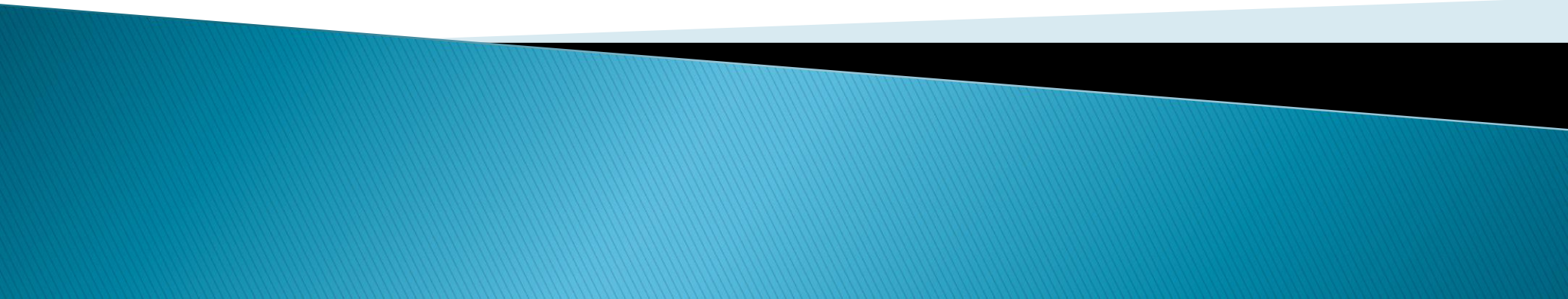
- ▶ Advantages:

- Path planning is fairly easy

- ▶ Problems:

- Tasks aren't decomposed enough
  - Accurate localization on one set of readings is near impossible
  - Perception of the box and pallet is extremely difficult
  - Any slight change or error in the environment model can cause failure
- 

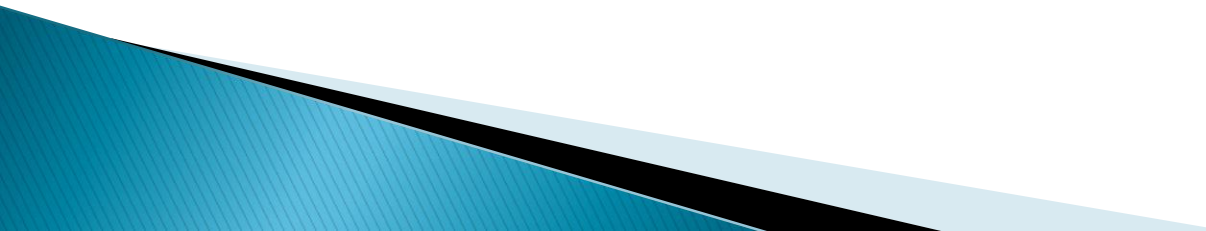
# The Motor Schema Architecture



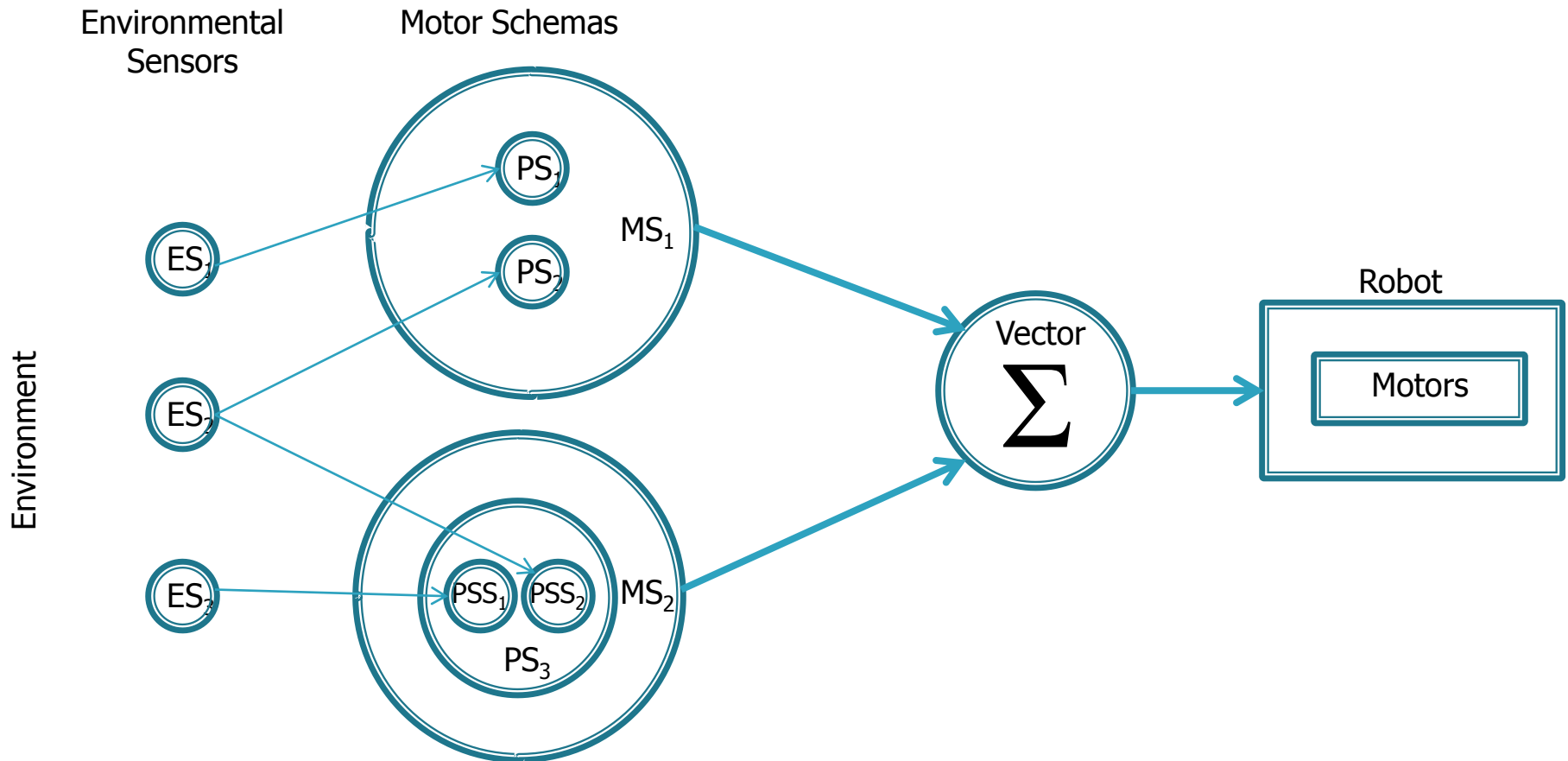
# Motor Schema

- ▶ Proposed by Ronald Arkin, Georgia Tech in 1987
- ▶ Motivation:
  - Biologically inspired
  - Massively parallel
  - Mimics (sort of) muscle memory

# Motor Schema

- ▶ Scaled decomposition of a task
  - ▶ Each motor schema represents a behaviour
  - ▶ Inputs stimulate the motor schemas
  - ▶ Outputs are aggregated
- 

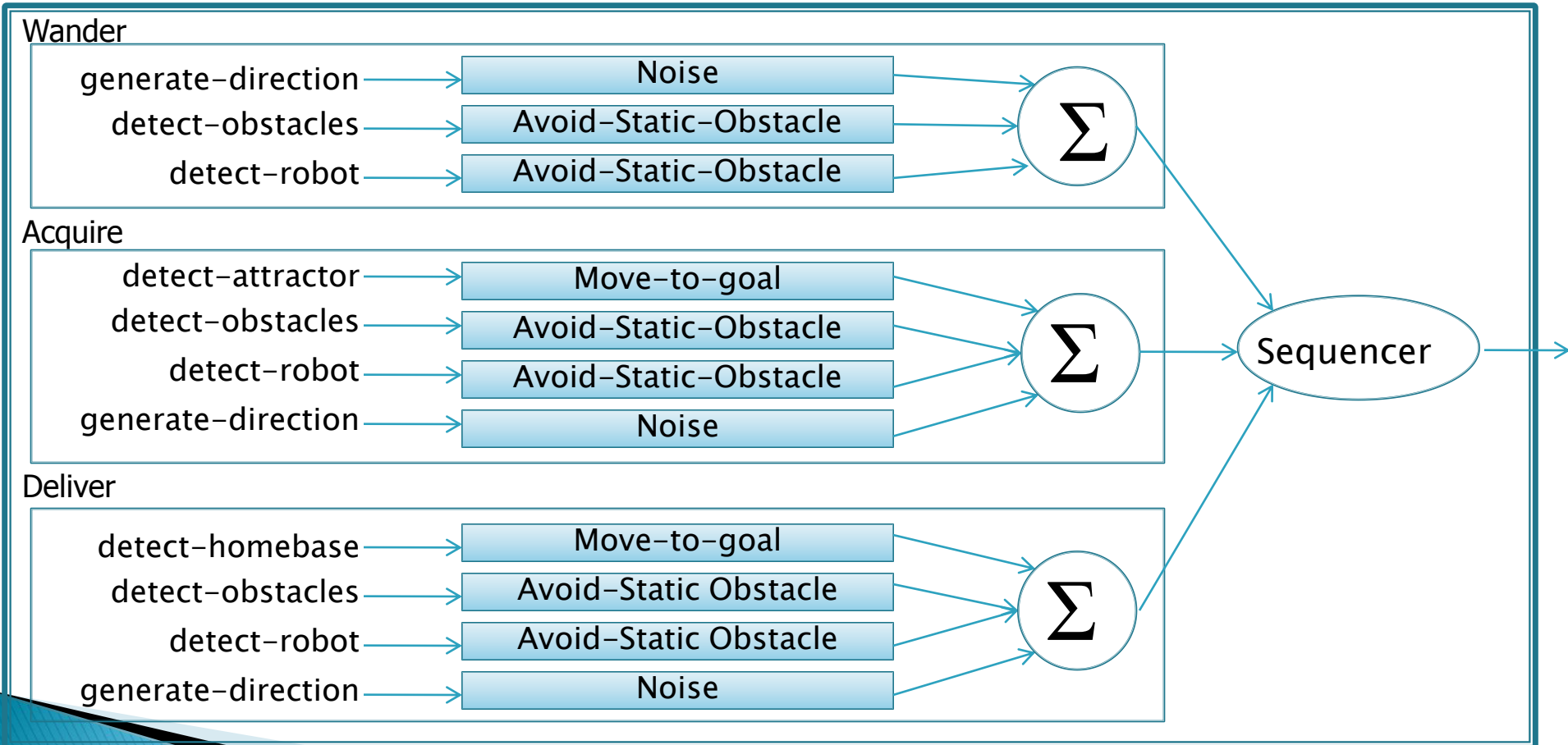
# Motor Schema



► Diagram from Arkin's book (p 144)

# Motor Schema Example

Forage



# Motor Schema

## ▶ Pros

- Scalable solution
- Includes planning
- Incremental development

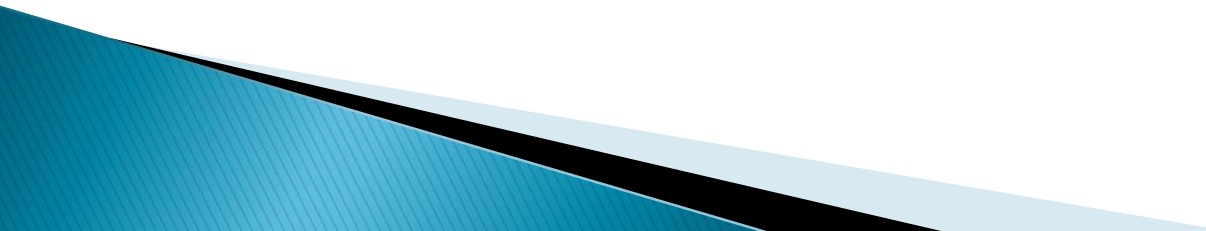
## ▶ Cons

- Does not solve the problem of perception i.e. How exactly do I build an environmental sensor which detects other robots
- Sequencing can be highly complex

# Finite State Machines

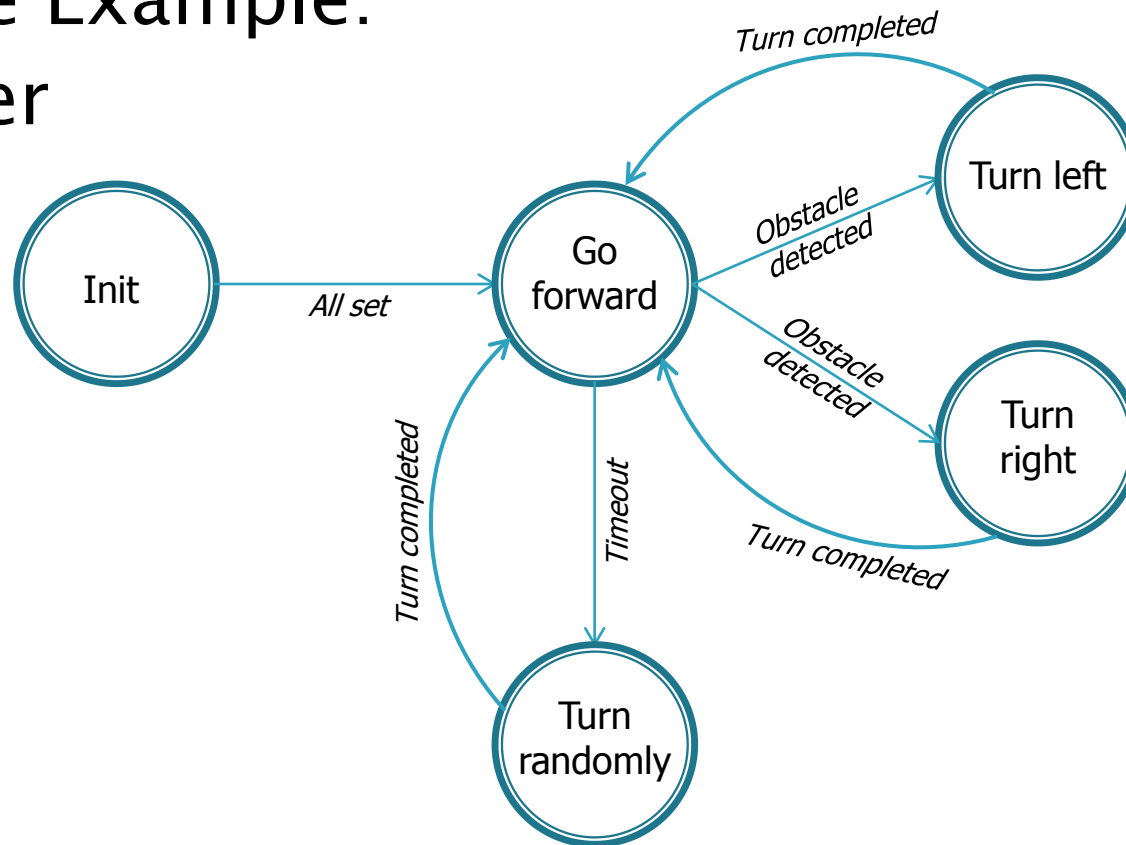


# Finite State Machines

- ▶ Abstract mathematical technique
  - ▶ Can be used to order simpler behaviours synchronously
    - Each state represents a behaviour
    - State transition = behavioural transition
    - Transitions are based on some criteria
- 

# Finite State Machines

- ▶ Simple Example:
- ▶ Wander



# Finite State Machines

## ▶ Pros

- Simple
- Powerful
- Clean arbitration

## ▶ Cons

- Not scalable to very large systems
- Most code is not running at any one time

# The Subsumption Architecture

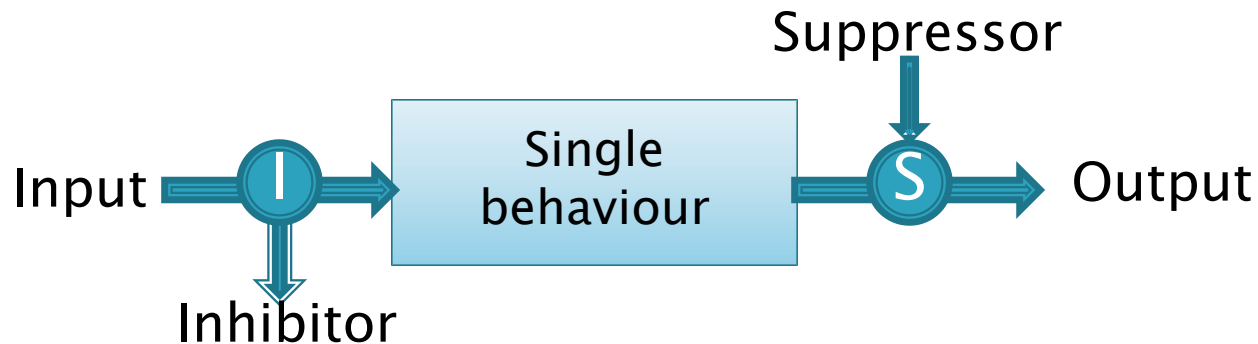


# Subsumption

- ▶ Proposed by Rodney Brooks, MIT in 1986
- ▶ Motivation:
  - Complex behaviour does not need a complex controller
  - The world is it's own best model
  - Systems should be built incrementally
  - Robustness is a design goal

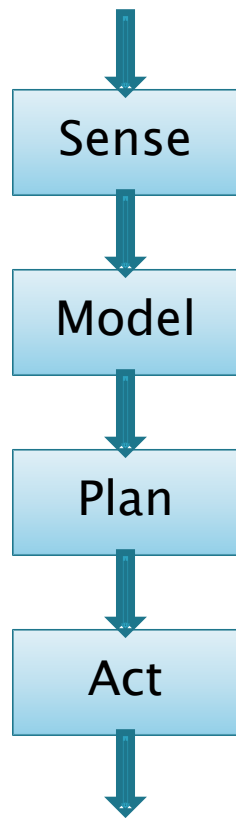
# Subsumption

- ▶ Vertical decomposition of task
- ▶ Each vertical unit is a behaviour
- ▶ Each behaviour can:
  - Inhibit other behaviours
  - Be suppressed by other behaviours:

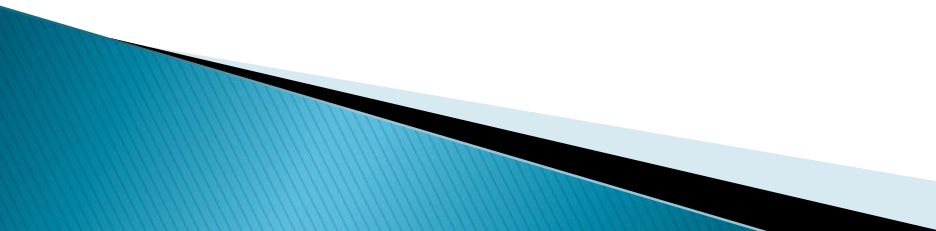


# Subsumption

- ▶ Comparison with sense-plan-act



# Subsumption Example

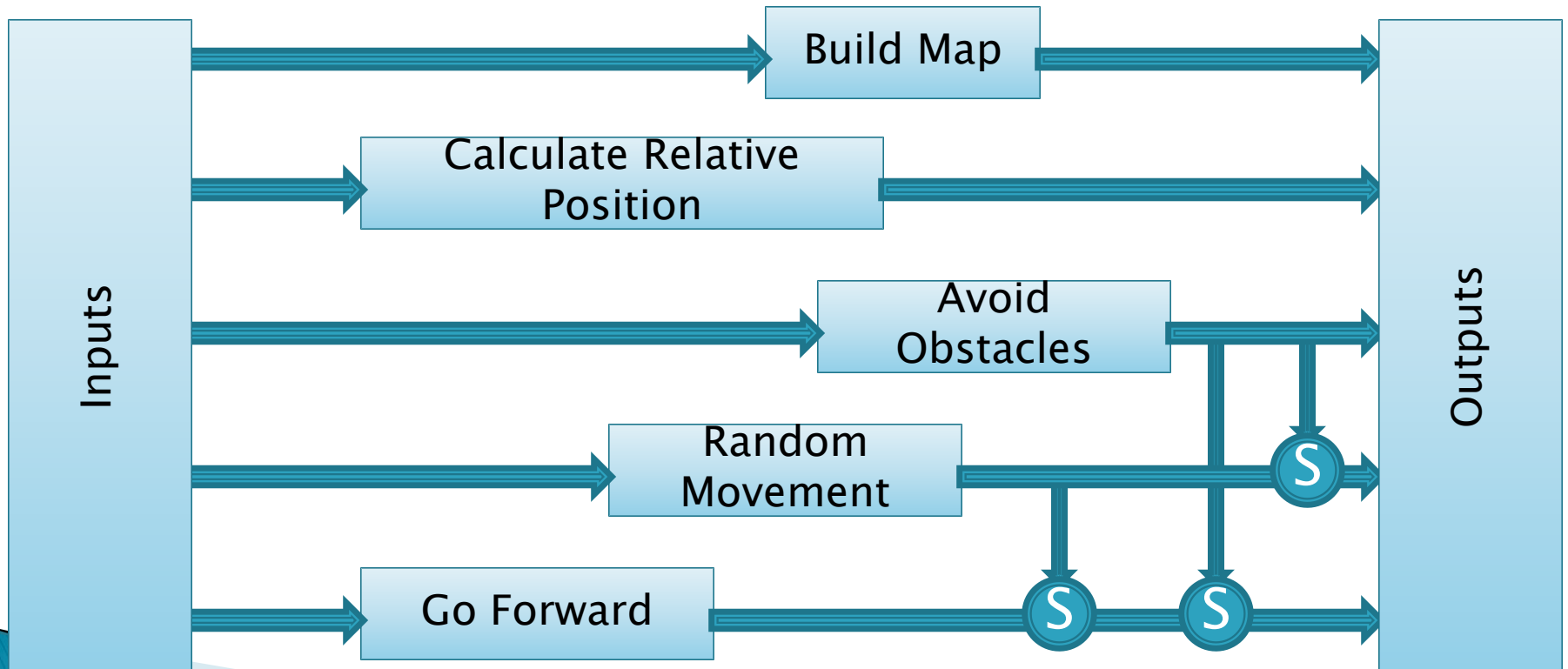
- ▶ Simple Mapping Robot
  - ▶ Task:
    - Map the current environment
  - ▶ Actuators
    - Two independent motorised wheels
  - ▶ Sensors
    - Array of eight sonar sensors
    - Two shaft encoders, one per wheel
- 

# Subsumption Example

- ▶ Simple Mapping Robot
- ▶ Five behaviours:
  - Avoid obstacles – use sonar readings to ensure a safe distance is maintained from any obstacle, simple reactive controller
  - Go forward – travel forward in a straight line
  - Random movement – Every 10 seconds turn the robot to a random heading
  - Calculate relative position
  - Build map

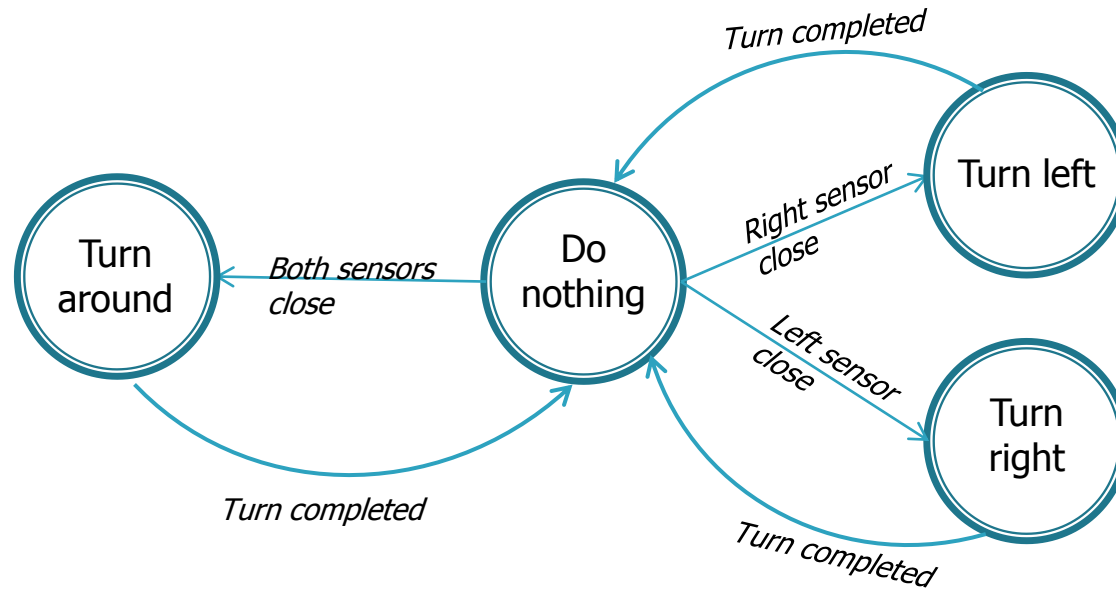
# Subsumption Example

- ▶ Simple Mapping Robot



# Subsumption

- ▶ Individual behaviours are often FSMs:
- ▶ Avoid obstacles



# Subsumption

## ▶ Pros

- Simple
- Robust
- Incremental development

## ▶ Cons

- Planning isn't taken care of
  - Lacks scalability
- 