Behaviour Based Robot Control Architectures
3 – Behaviour Design & Behaviour Arbitration

Embodiment

- The fact of having a body, and its associated properties, e.g., sensors and motors (effectors)
- Physical Grounding Hypothesis: “to build a system that is intelligent it is necessary to have its representations grounded in the physical world” (Brooks, 1999, p. 114).
- Searle argued that the main reason for the failure of strong (traditional) AI was that it is concerned with computer programs, but ‘has nothing to tell us about machines’ (Searle, 1980), i.e. physical systems situated in and causally connected to their environments.” (Ziemke, 2001)
- *Grounding*: connecting an AI system to the world using a sensory-motor system

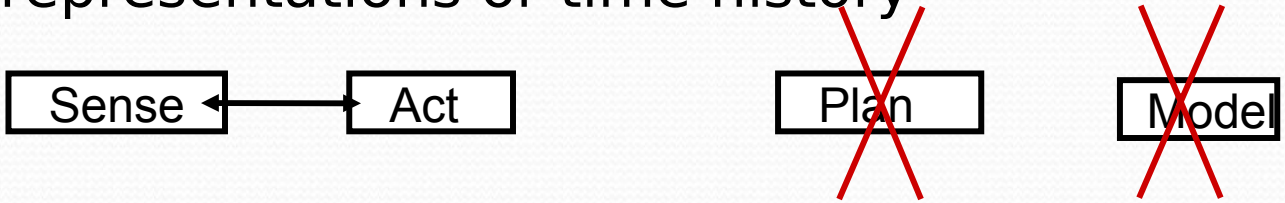
Behaviour-based, Reactive Systems

- Behavior = Mapping of sensory inputs to a pattern of motor actions that are used to achieve a task

  or (equivalently) = Reaction to a stimulus

  or (equivalently) = a control law providing rapid closed-loop feedback control between the robot's sensing and acting, and embodying some competence in the designer's perspective (*obstacle-avoidance, follow-wall).*

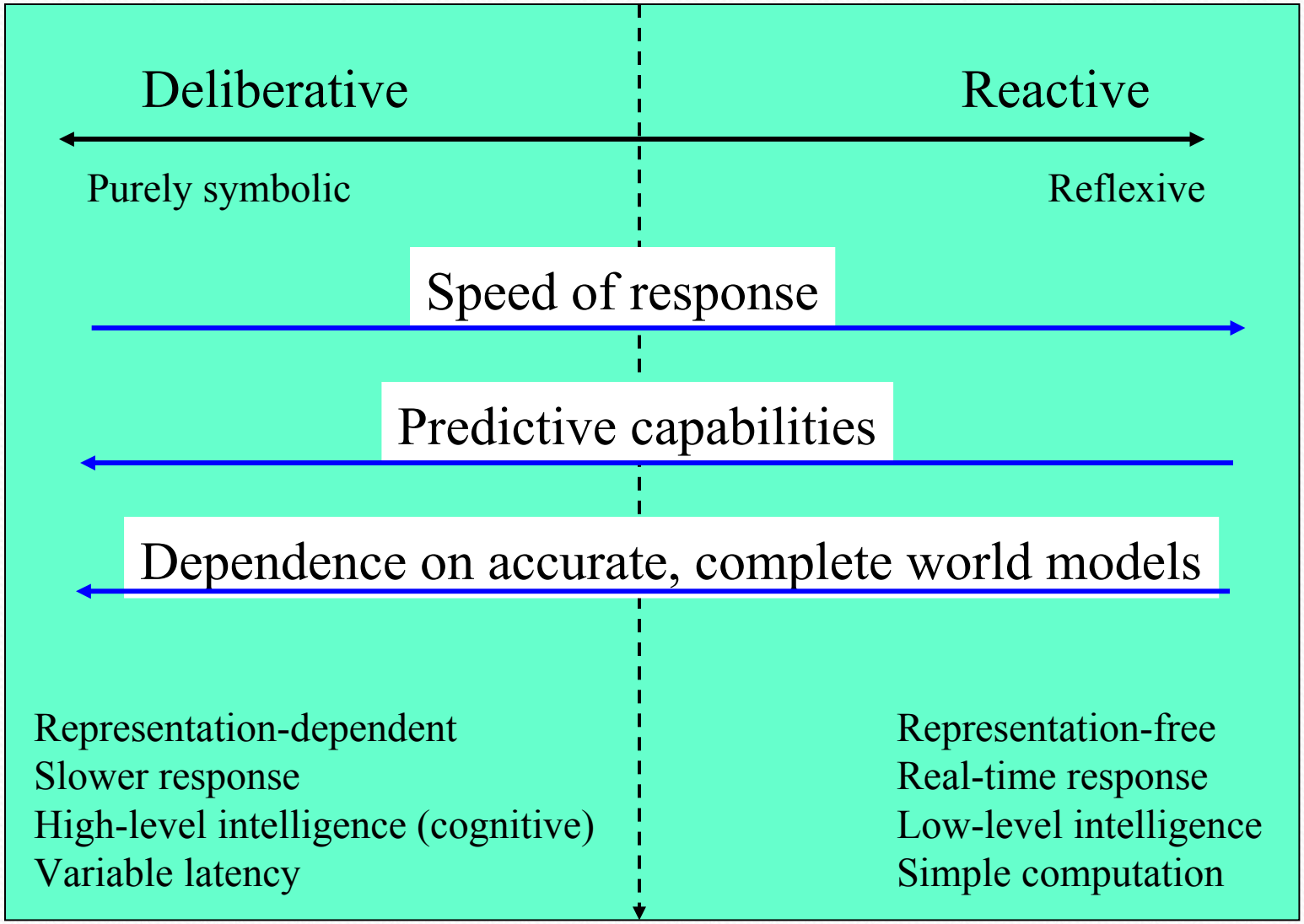
- A reactive robotic system tightly couples perception to action without the use of intervening abstract / complex representations or time history



Characteristics of Reactive Robotic Systems

- Behaviors serve as basic building blocks for robotic actions
  - Typically, behaviors are simple sensorimotor pairs
- Use of explicit abstract representational knowledge is avoided in the generation of a response
- Animal models of behavior often serve as a basis for these systems
- These systems are inherently modular from a software design perspective

Recall


Deliberative
Reactive
Purely symbolic
Reflexive
Speed of response
Predictive capabilities
Dependence on accurate, complete world models
Representation-dependent
Slower response
High-level intelligence (cognitive)
Variable latency
Representation-free
Real-time response
Low-level intelligence
Simple computation

Basis for Robotic Behavior

- Key questions:
  - What are the right behavioral building blocks for robotic systems?
  - What really is a primitive behavior?
  - How are these behaviors effectively coordinated?
  - How are these behaviors grounded to sensors and actuators?
- No universally agreed-upon answers
- Ultimate judge: appropriateness of the robotic response to a given task and environment

Three common methods for specifying and designing robotic behaviors

1. Ethologically guided/constrained design (lecture B-1)

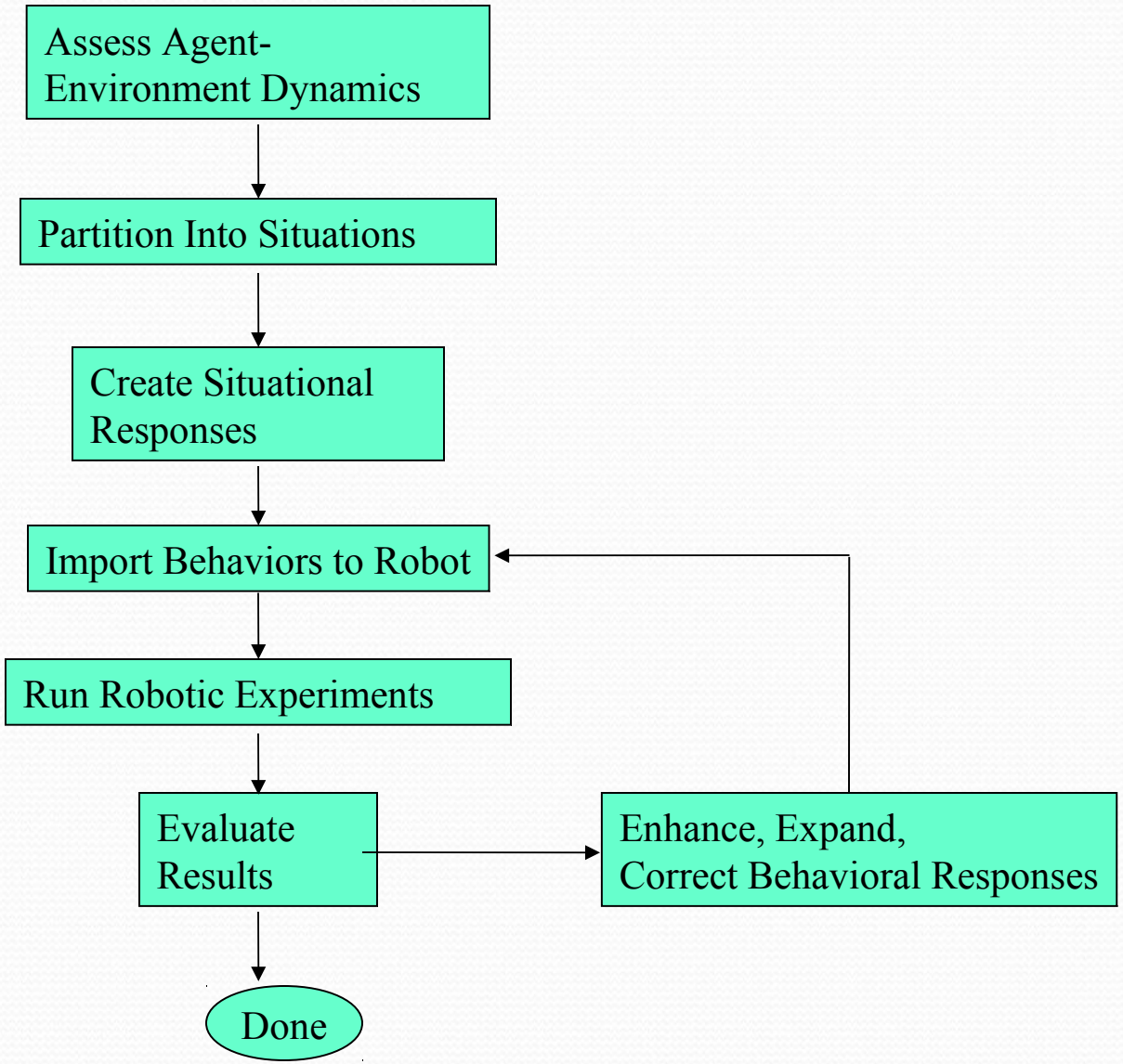
1. Situated activity-based design

1. Experimentally driven design

Situated Activity-Based Design

- *Situated* activity: robot's actions are predicated upon the situations in which the robot finds itself
- Therefore, the perception problem is reduced to recognizing what situation(s) the robot is in and then choosing one action (out of perhaps many) to undertake.
- When robot finds itself in new situation, it selects more appropriate action
- Using this design methodology: requires solid understanding of relationship between robotic agent and its environment

Design Methodology for Situated Activity



Here, Situations Don't Need Biological Basis

- Arbitrarily complex situations can be created and specified
- Approach by Agre and Chapman (1987)

with Pengi system:

- Situations characterized by their indexical-functional aspects
- Indexical → what makes the circumstances unique
- Functional → robotic agent's intended outcome or purpose in a given situation

Examples of indexical-functional characterization of situations:

- Situation 1: the-block-I-need-to-kick-at-the-enemy-is-behind-me
- Situation 2: I've-run-into-the-edge-of-the-wall

Pros/Cons of “Situations”

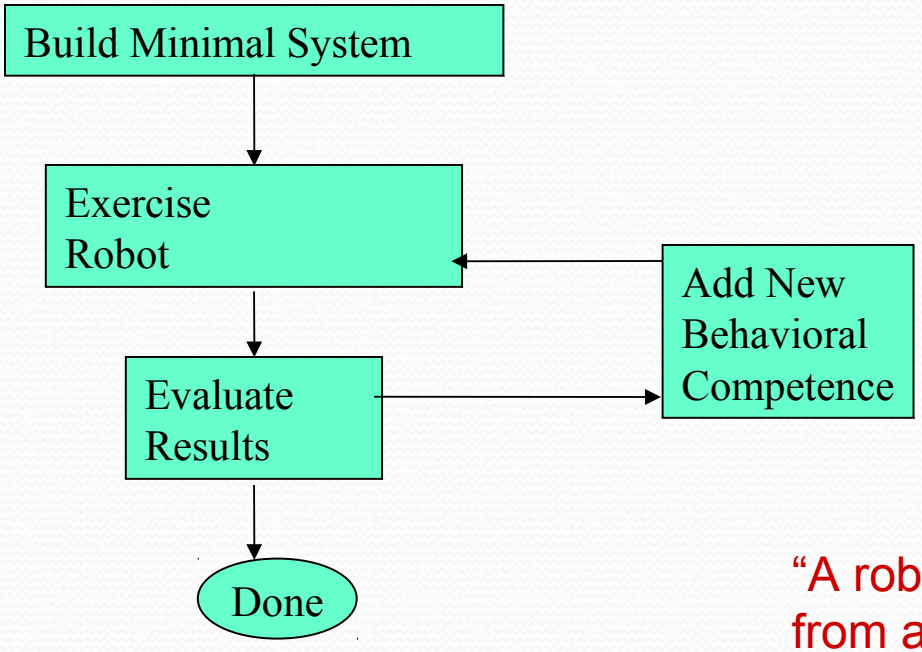
- Situations can be highly artificial
- Situations can be arbitrarily large in number
- Multiple candidate actions may be in conflict – no planning to project consequences of actions

More expansive version of theory of situated activity: **Universal plans/Reactive plans**

- Cover entire domain of interaction
- But, enumeration of every possible situation is impractical and mathematically intractable

Experimentally Driven Design

- Created in a bottom-up manner
  (Stand up, walk, prowling…)
- Iterative design



"A robot that walks; emergent behaviors from a carefully evolved network", by Brooks, MIT AI Lab memo, 1989.

Behavioral Encoding

Behavioral encoding: creates functional mapping from the stimulus plane to the motor plane

- Behavior expressed as a triple: **(*S, R, β*)**

  where:

- *S* denotes domain of all interpretable stimuli, or percept ***s*** **(s ∈ S)** represented as tuple **(*p*, λ)**

  where:

  p = perceptual class (e.g. obstacle, goal)

  λ = property of strength

  R denotes range of possible responses (e.g. v ,w)

  β denotes mapping ***β: S→ R***

Two Categories of Behavioral Mappings

1. Discrete: stimulus produces a response from an enumerable set ***R*** of prescribed choices
   - E.g.: turn-right, go-straight, stop, travel-at-speed-5

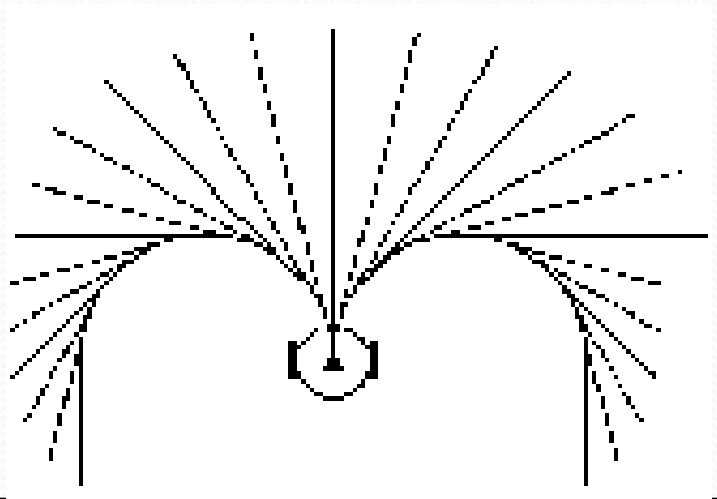
1. Continuous: stimulus produces a motor response that is continuous over ***R***'s range

Discrete Encoding

- Example: Use of rule-based systems using IF-THEN rules
  - Randomwalk

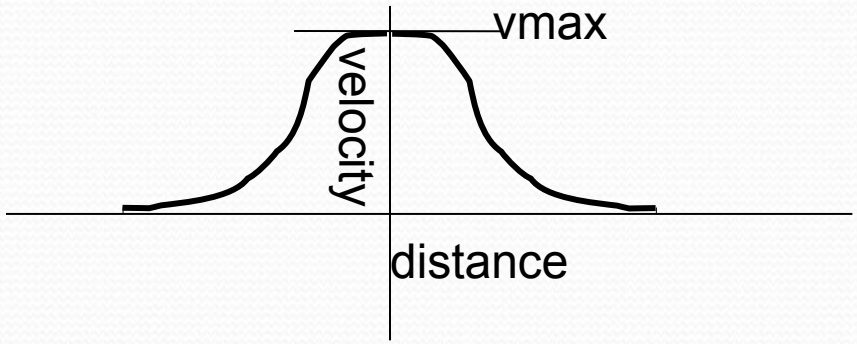
```
IF (shortest-front == 0)
       vm(vmax, 0, 0)
ELSE { IF (shortest-front < 5)
           recompute shortest-front
        ELSE IF (shortest-front > 12)
                 recompute shortest-front
        computed-angle = compute angle from shortest-front
        vm(0, computed-angle, computed-angle)
       }
```

  - DAMS, Dynamic Window Approach (DWA), VFH, VFH+…



Continuous Functional Encoding

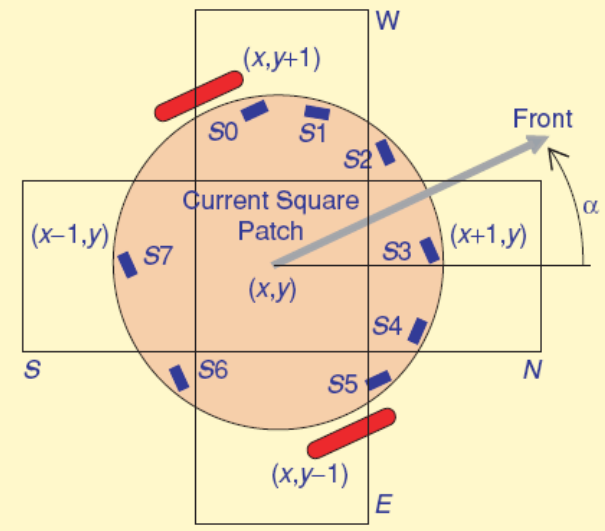
- No longer use enumerated set of responses
- Instead, mathematical function transforms sensory input into behavioral reaction
- Example: Velocity inversely proportional to distance from obstacle

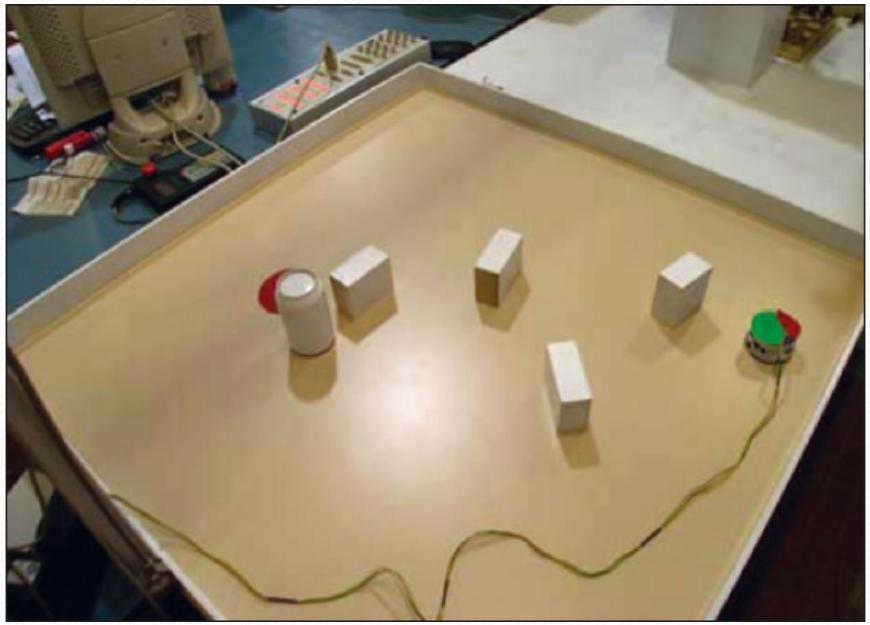


How do we write behaviours?

1. Hand-coded rules, Fuzzy Rules
2. Dynamic Control System Approach
3. Learning Solutions

Example Hand-Coded rules





Obstacle avoidance behaviour:

IF S0>0.6 and S1 > 0.3 and S1 < 0.6 => V=(+0.5, -2)

IF S5>0.6 and S4 > 0.3 and S4 < 0.6 => V=(-2, +0.5)

...

ELSE V=(+2,+2)

Hand-Coded Rules

- ‘Complicated’ function
- Not easy to modify
- Not intuitive
- Many hard-coded parameters
- Not easy to understand

The Fuzzy Approach

What we want to express is:

If S0 is close and S1 is approaching => turn fast right

IF S5 is close and S4 is approaching => turn fast left

...

We have just defined the rules for a fuzzy logic system

The Fuzzy Approach

- Before we have a fuzzy solution we need to find out

a) how to define terms such as close, *approaching, fast.*
b) how to combine terms using AND, OR and other connectives
c) how to combine all the rules into one final output

Fuzzy sets

Boolean/Crisp set A is a mapping for the elements of S to the set {0, 1}, i.e., A: S → {0, 1}

Characteristic function:

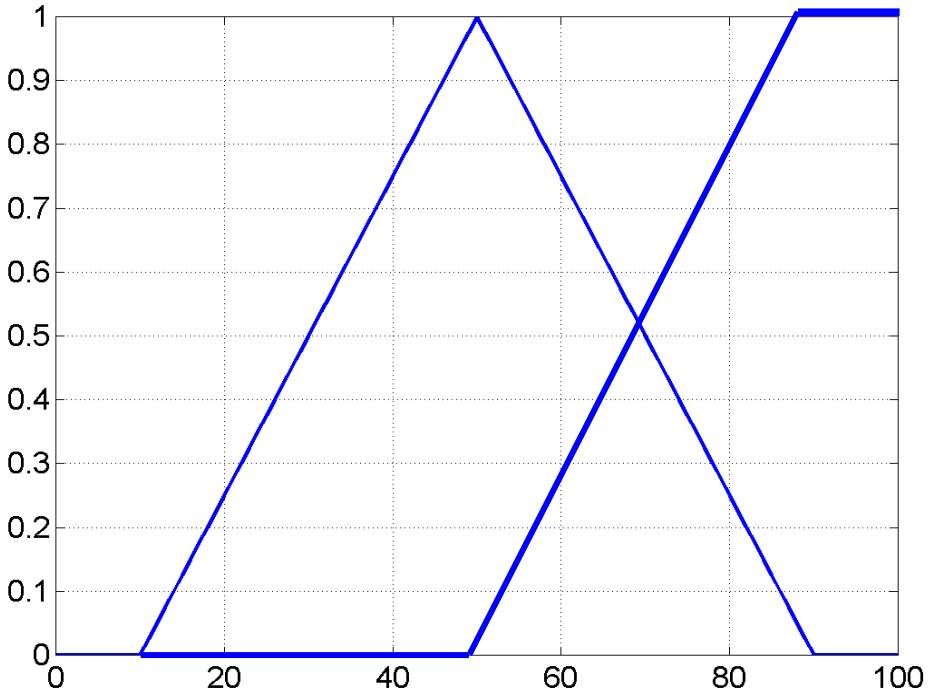
$$\mu_A(x) = \begin{cases} 1 & \text{if x is an element of set A} \\ 0 & \text{if x is not an element of set A} \end{cases}$$

- **Fuzzy set F** is a mapping for the elements of S to the interval [0, 1], i.e., F: S → [0, 1]
- Characteristic function: $0 \leq \mu_F(x) \leq 1$
- 1 means full membership, 0 means no membership and anything in between, e.g., 0.5 is called **graded membership**

Simple membership functions

Piecewise linear: triangular etc.

Easier to represent and calculate ⇒ saves computat

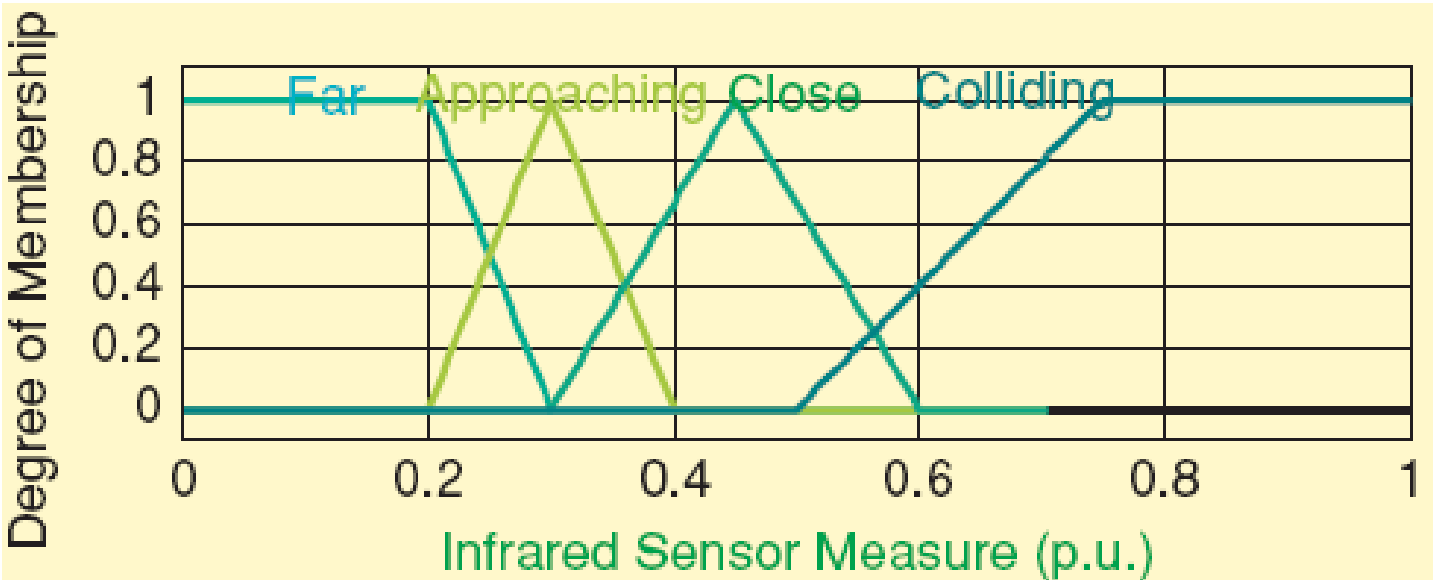


Using Fuzzy Logic

fuzzy set A

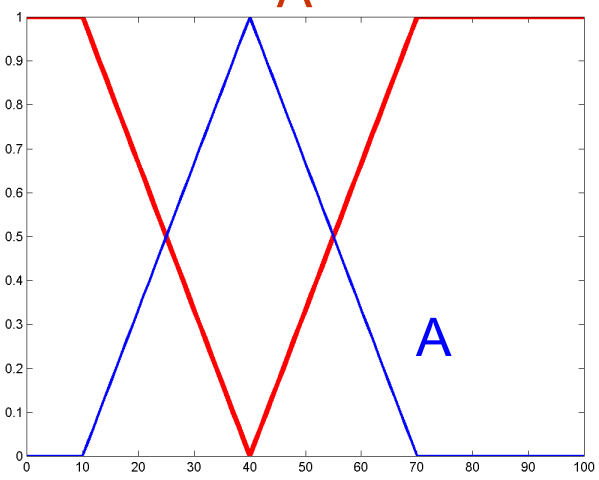
$A = \{(x, \mu_A(x)) | x \in X\}$ where $\mu_A(x)$ is called the membership function for the fuzzy set A. *X* is referred to as the universe of discourse.

The membership function associates each element x Є X with a value in the interval [0,1].

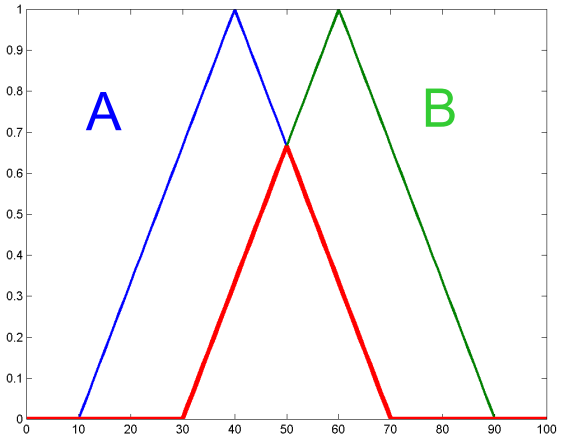


Fuzzy set operations

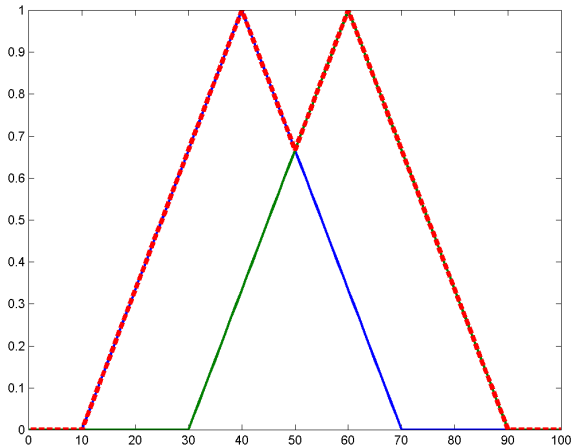
A’



A ∩ B



A ∪ B



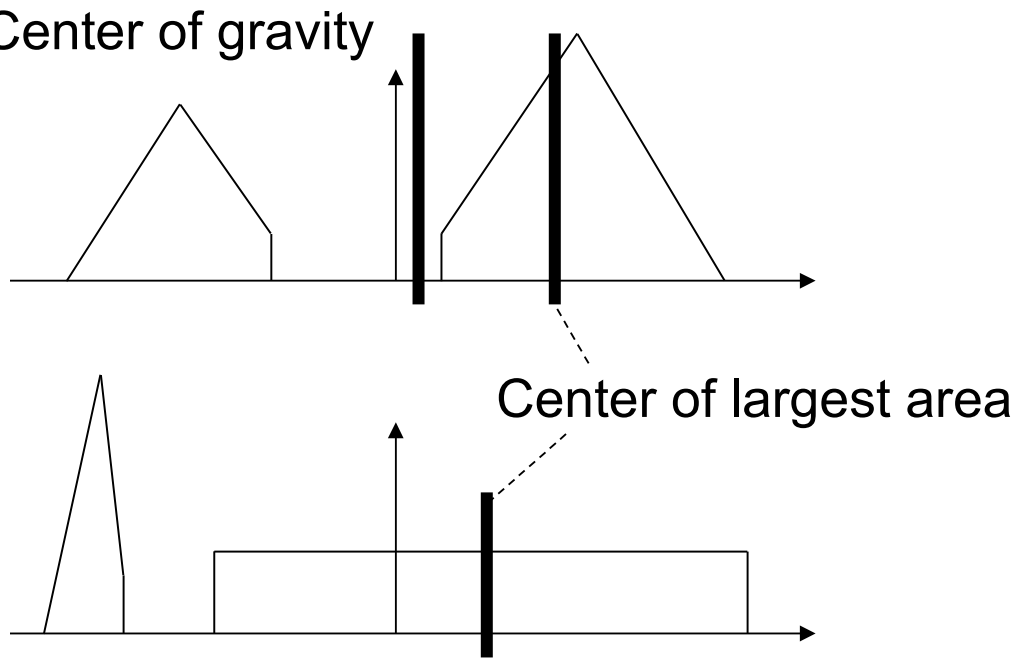
Defuzzify the output

Take a fuzzy set and produce a single crisp number that represents the set.

Practical when making a decision, taking an action etc.

$\Sigma \mu_i x$

$\Sigma \mu_i$



Obstacle Avoidance Example

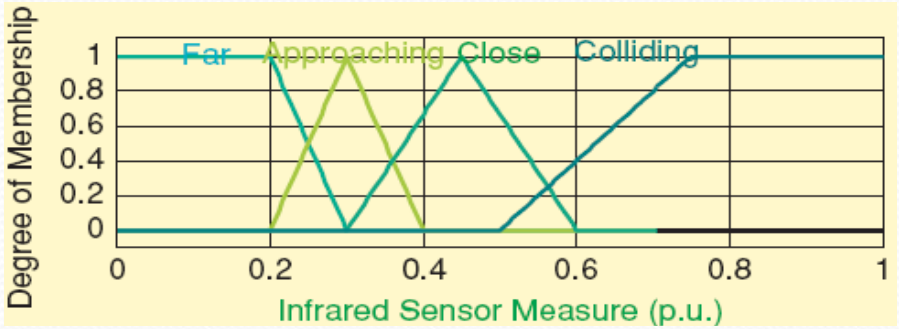
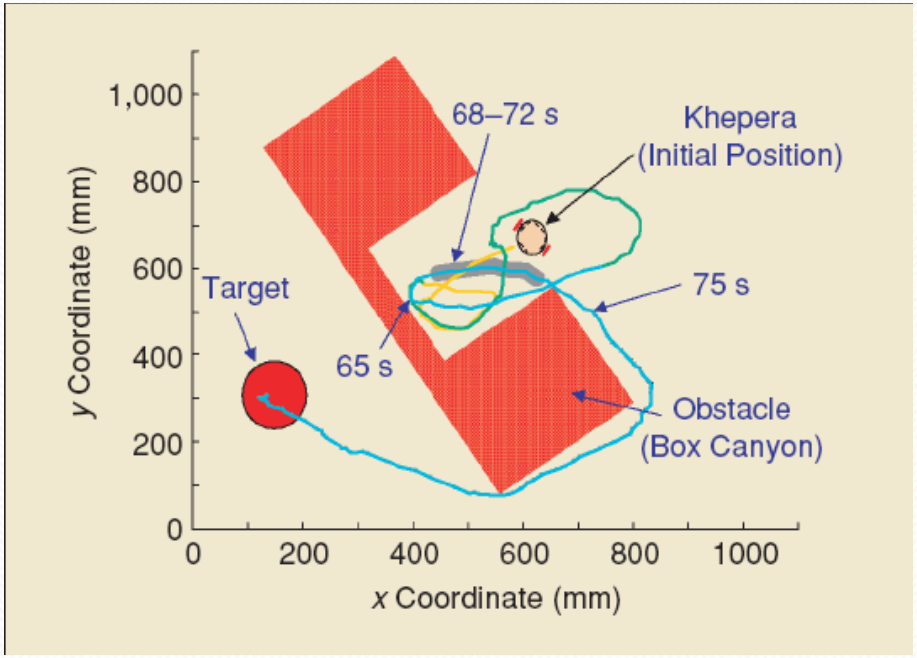


Table 2. FLC2 rule table.

INPUTS (logic OR)						OUTPUTS	
S0	S1	S2	S3	S4	S5	Right speed	Left speed
Coll	Coll	Coll	Any	Any	Any	Negative fast	Aero
Any	Any	Any	Coll	Coll	Coll	Zero	Negative fast
Close	Appr	Any	Any	Any	Any	Negative fast	Positive slow
Any	Any	Any	Any	Appr	Close	Positive slow	Negative fast
Far	Far	Far	Far	Far	Far	Positive fast	Positive fast



Why use fuzzy logic?

Pros:

Conceptually easy to understand w/ “natural” maths

Tolerant of imprecise data

Universal approximation: can model arbitrary nonlinear functions

Offers an automatic way to combine multiple rules

Intuitive

Based on linguistic terms

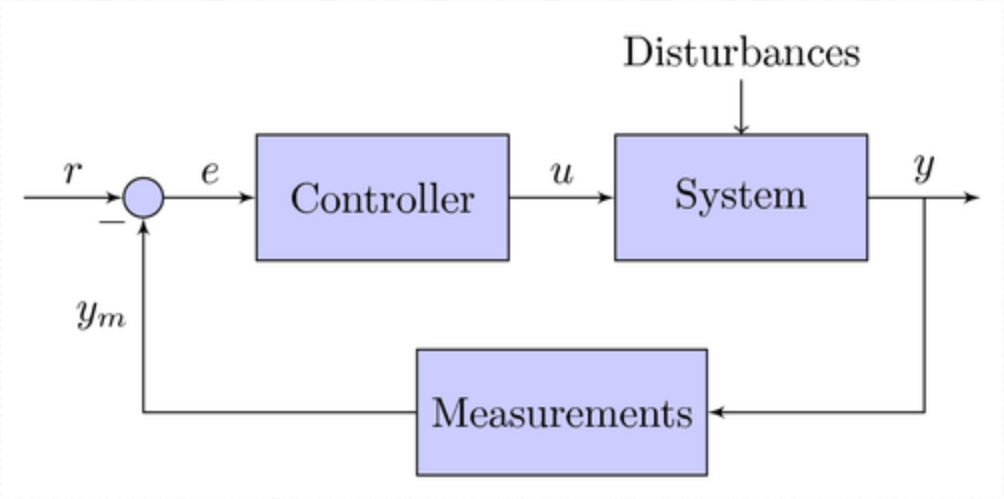
Convenient way to express expert and common sense knowledge

Cons:

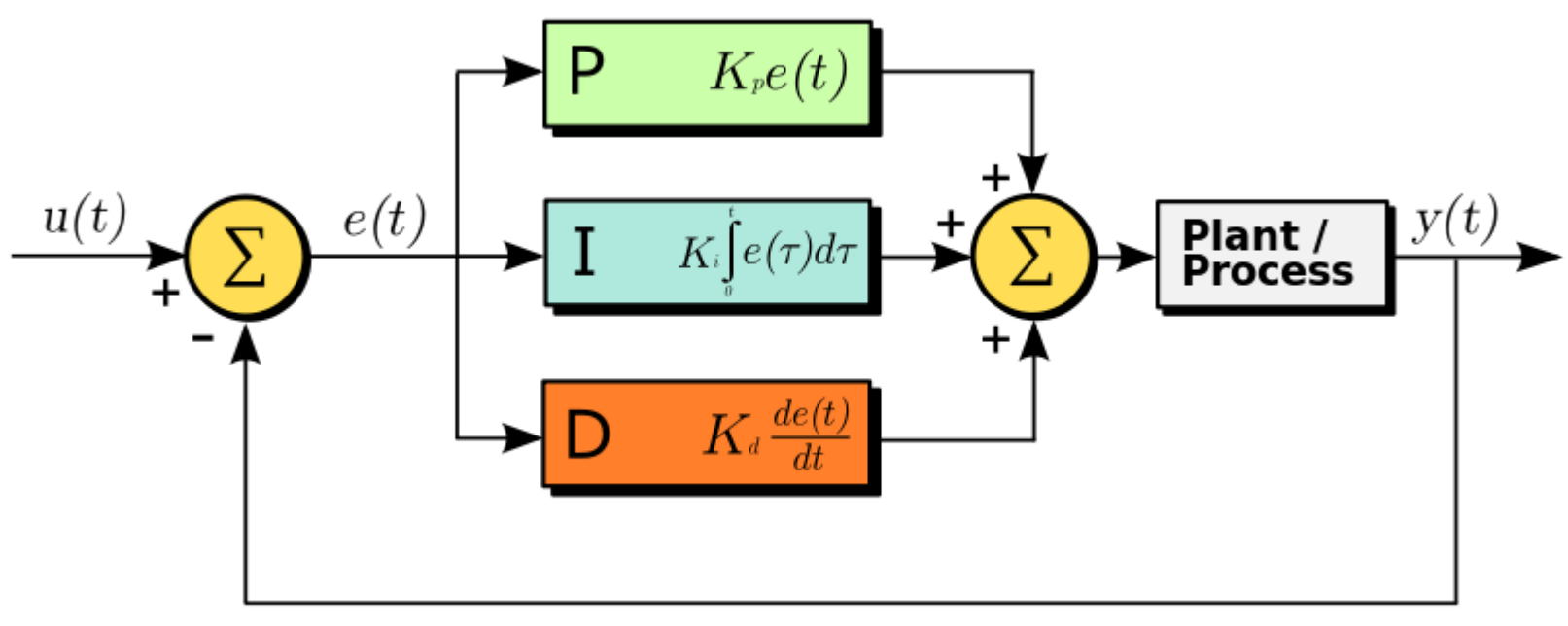
Not a cure-all

Crisp/precise models can be more efficient and even

Control System

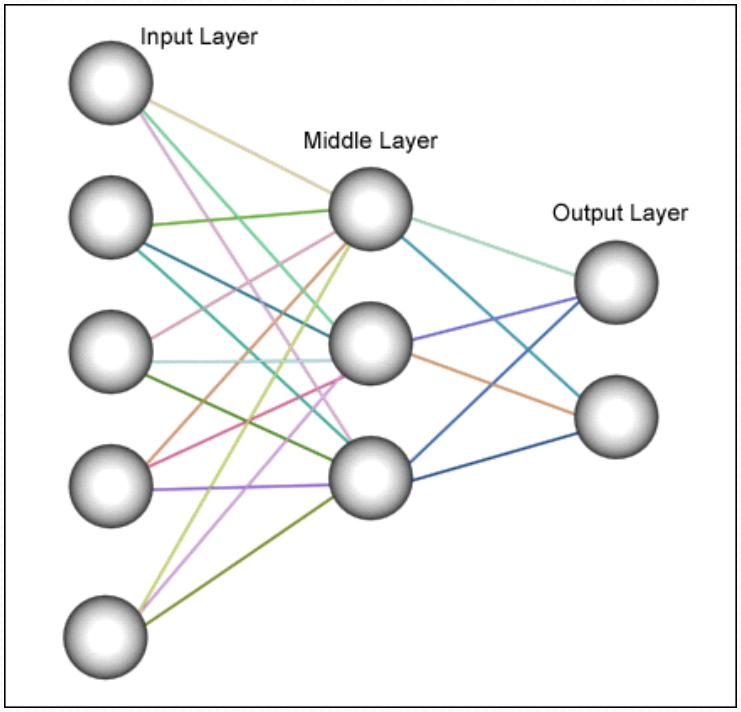


Example: PID Controller



Proportional-Integral- Differential Controller

Learning



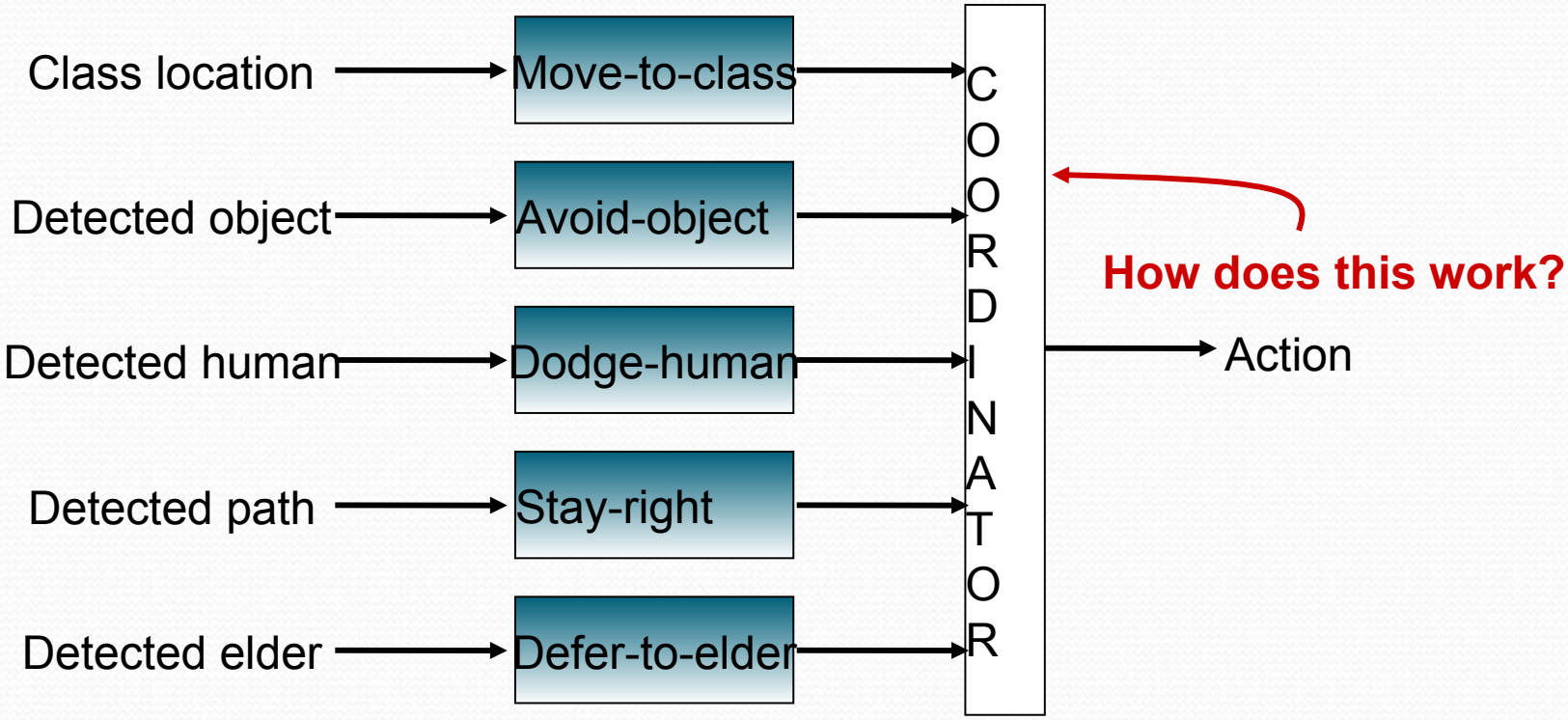
Deals with
- Lack or imprecision of sensor, actuator & behavioral models
- Noisy sensor data
- Uncertainty in sensing & acting process

Learning the best response given the inputs

- Supervised Learning - Needs training set with teaching signal (e.g. learn from demonstration)
- Unsupervised Learning, e.g. Reinforcement Learning (usually uses offline learning phase in simulation)

Assembling Behaviors

- Why: To achieve more complex tasks
- Issue: When have multiple behaviors, how do we combine them?
- Think about in terms of Navigation example



Competitive Methods

- Provide a means of coordinating behavioral response for conflict resolution / sequencing the achievement of different goals at different times
- Can be viewed as “winner take all” arbitration

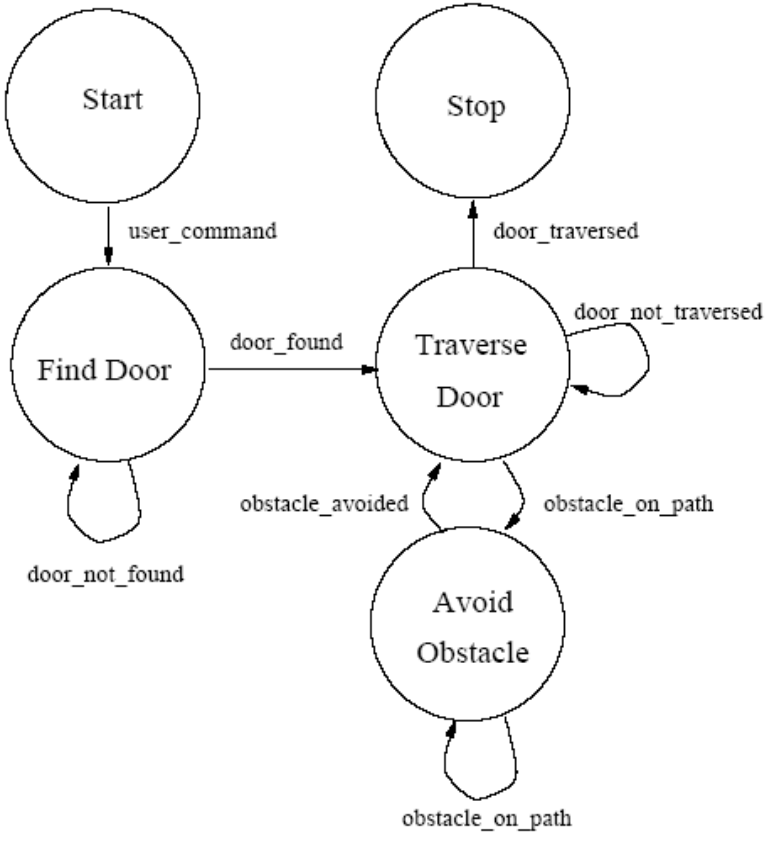
- Arbitration can be:
  - Sequencing
  - Prioritization

Competitive Methods

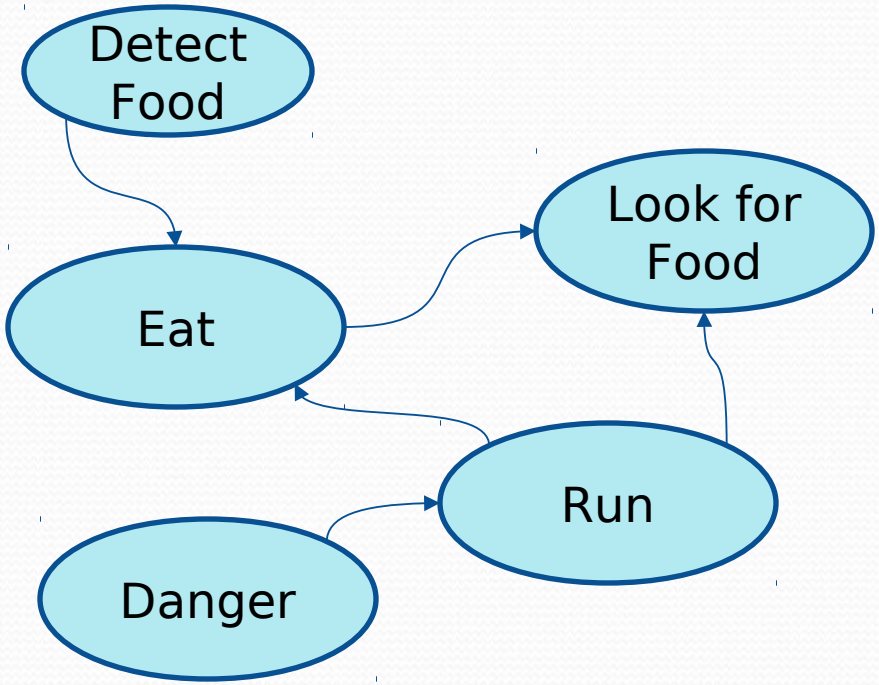
- Provide a means of coordinating behavioral response for conflict resolution / sequencing the achievement of different goals at different times
- Can be viewed as “winner take all” arbitration

- Arbitration can be:
  - Sequencing
  - Prioritization

Competitive Methods: Sequencing

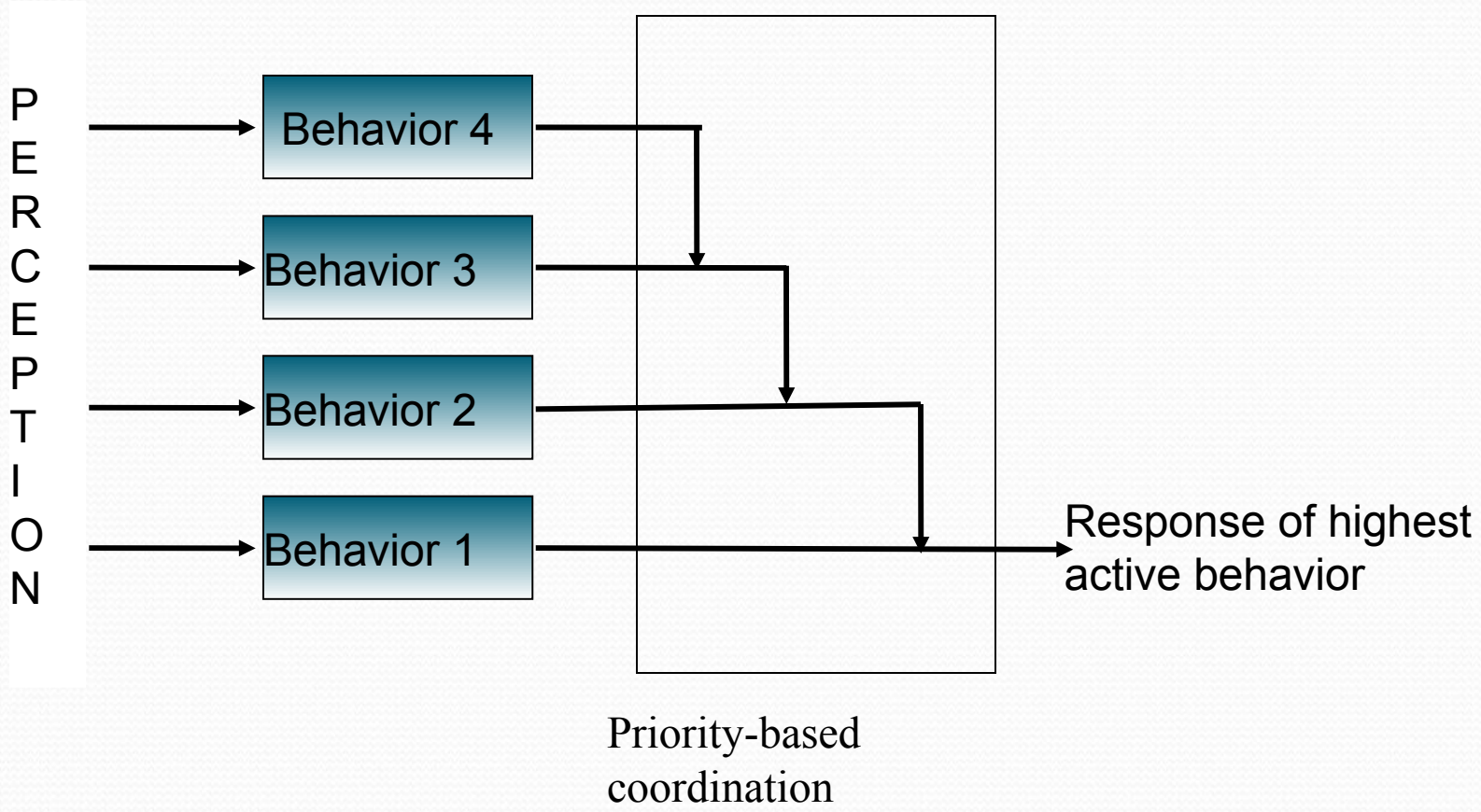


Finite State Machine (FSM)
- Centralized Sequencer
- Switching Conditions



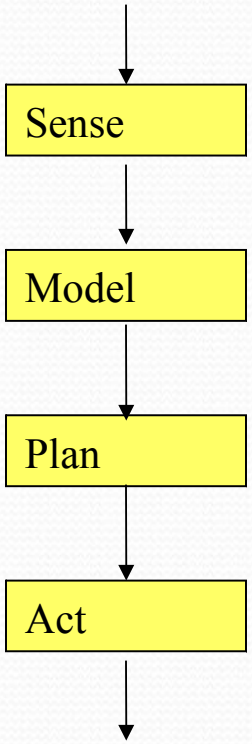
Activation Network (e.g. Maes, Tyrrell…)
- Decentralized Mechanism
- Parallelism

Competitive Methods: Prioritization

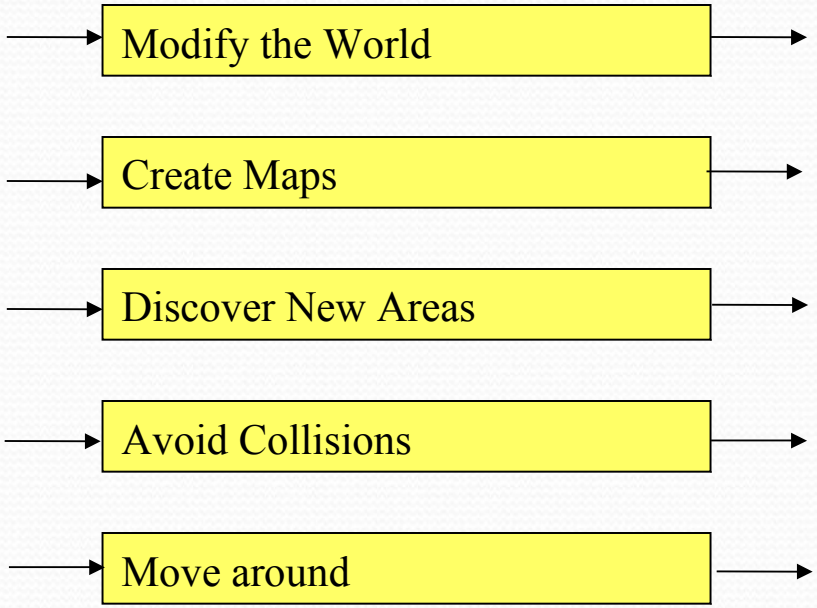


Example: Subsumption Architecture

- Developed in mid-1980s by Rodney Brooks, MIT



Old Sense-plan-act model



New subsumption model

Subsumption Robots

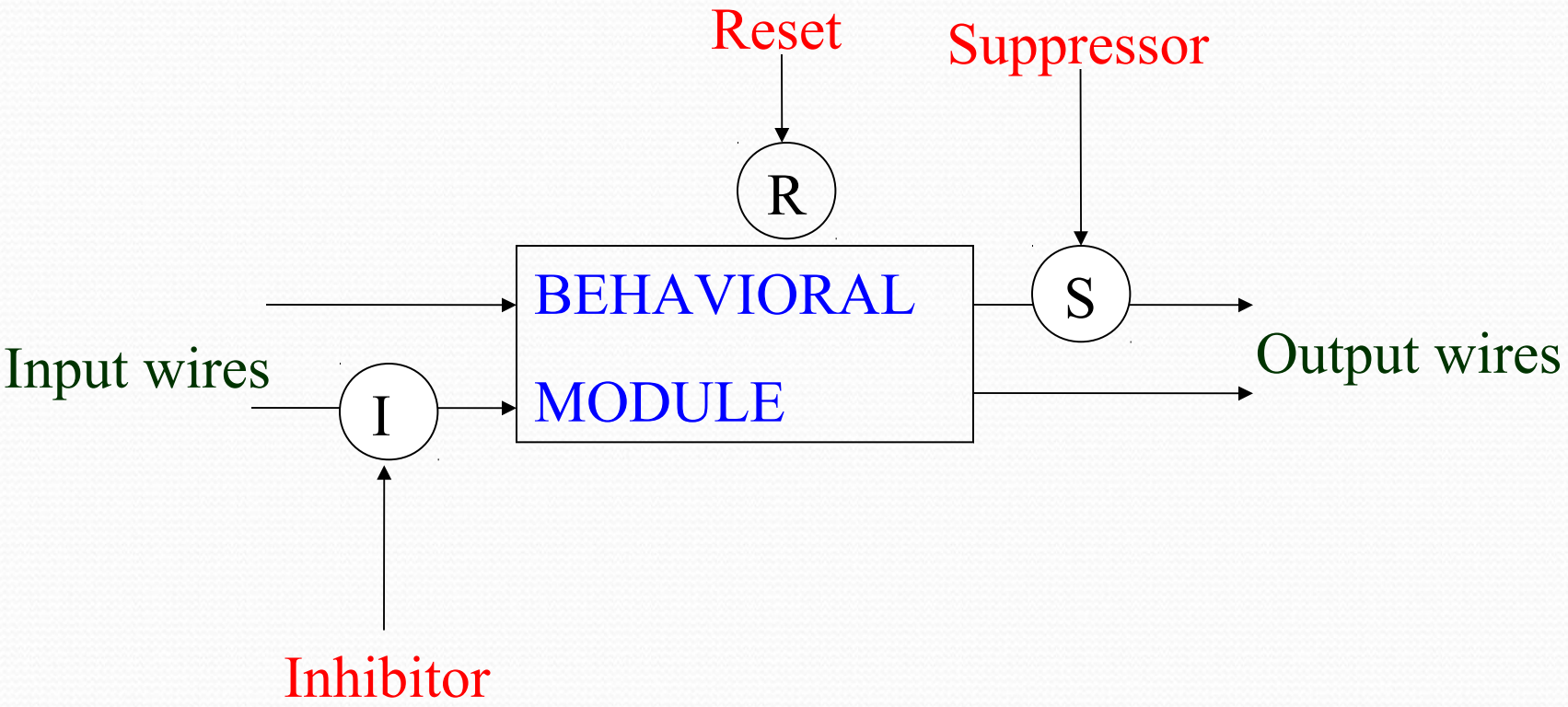
- Allen
- Tom and Jerry
- Genghis and A
- Squirt
- Toto
- Seymour
- Tito
- Polly
- Cog

Coordination in Subsumption

- "Subsumption" comes from coordination process used between layered behaviors of architecture
- Complex actions subsume simpler behaviors
- Fixed priority hierarchy defines topology
- Lower levels of architecture have no "awareness" of upper levels

- Coordination has two 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 suppressing message

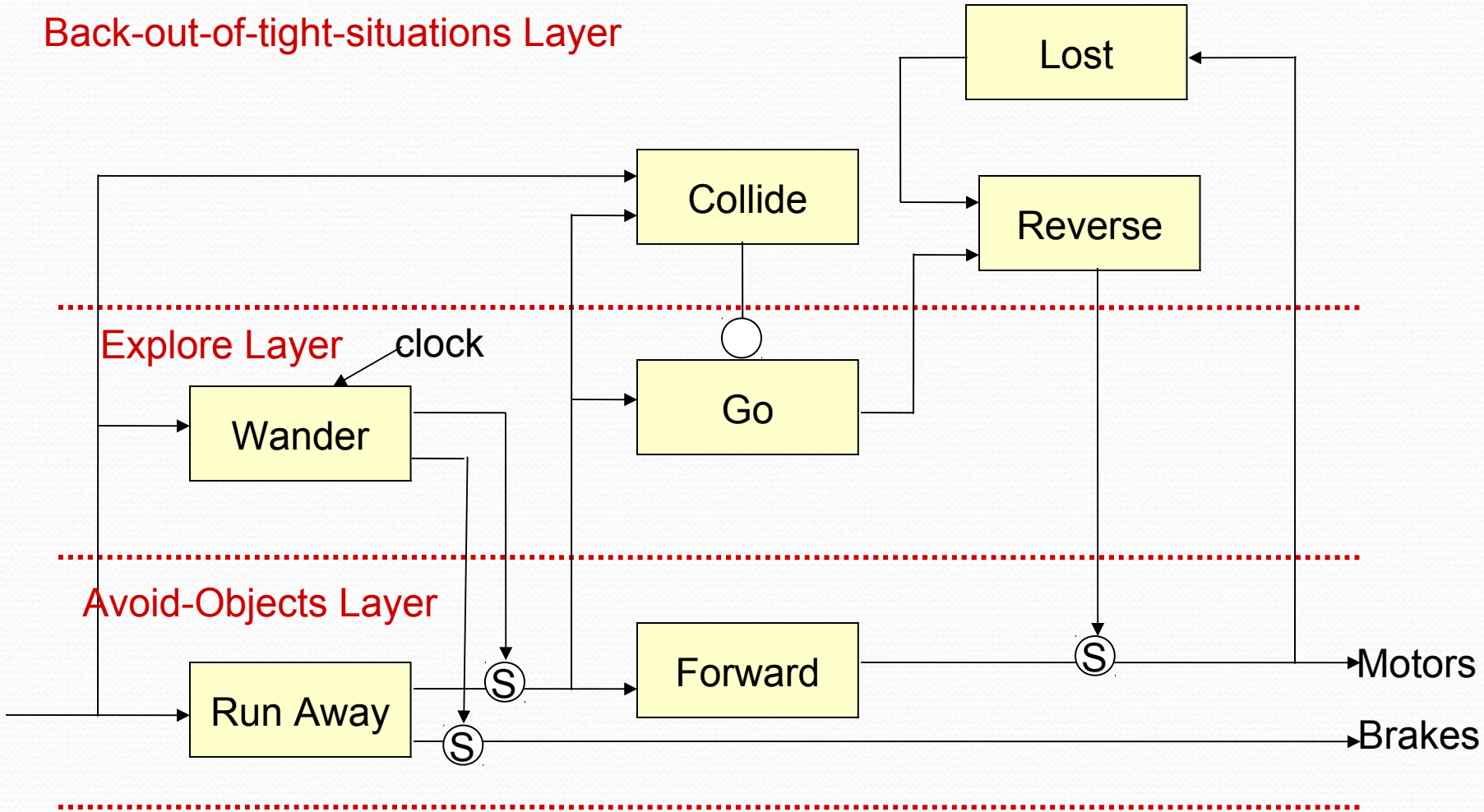
Subsumption Based on Augmented Finite State Machines (AFSM)



Characteristics of Subsumption

- AFSM encapsulates a particular behavioral transformation function β_i
- Stimulus or response signals can be suppressed or inhibited by other active behaviors
- Each AFSM performs its own action and is responsible for its own world perception
- No global memory, bus, or clock
- No central world models
- No global sensor representations
- Each behavior can be (but is not required to be) mapped onto its own processor
- Later versions of subsumption:
  - Use Behavior Language, which provides higher abstraction independent of AFSM, using a single rule set to encode each behavior
  - High-level language is then compiled to the intermediate AFSM, which can then be further compiled to run on a range of target processors

Example


Back-out-of-tight-situations Layer
Lost
Collide
Reverse
Explore Layer
clock
Wander
Go
SENSORS
Avoid-Objects Layer
Run Away
S
S
Forward
S
Motors
Brakes

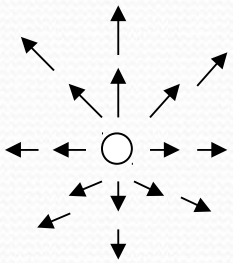
Cooperative Methods

- Provide a means for satisfying multiple objectives at the same time

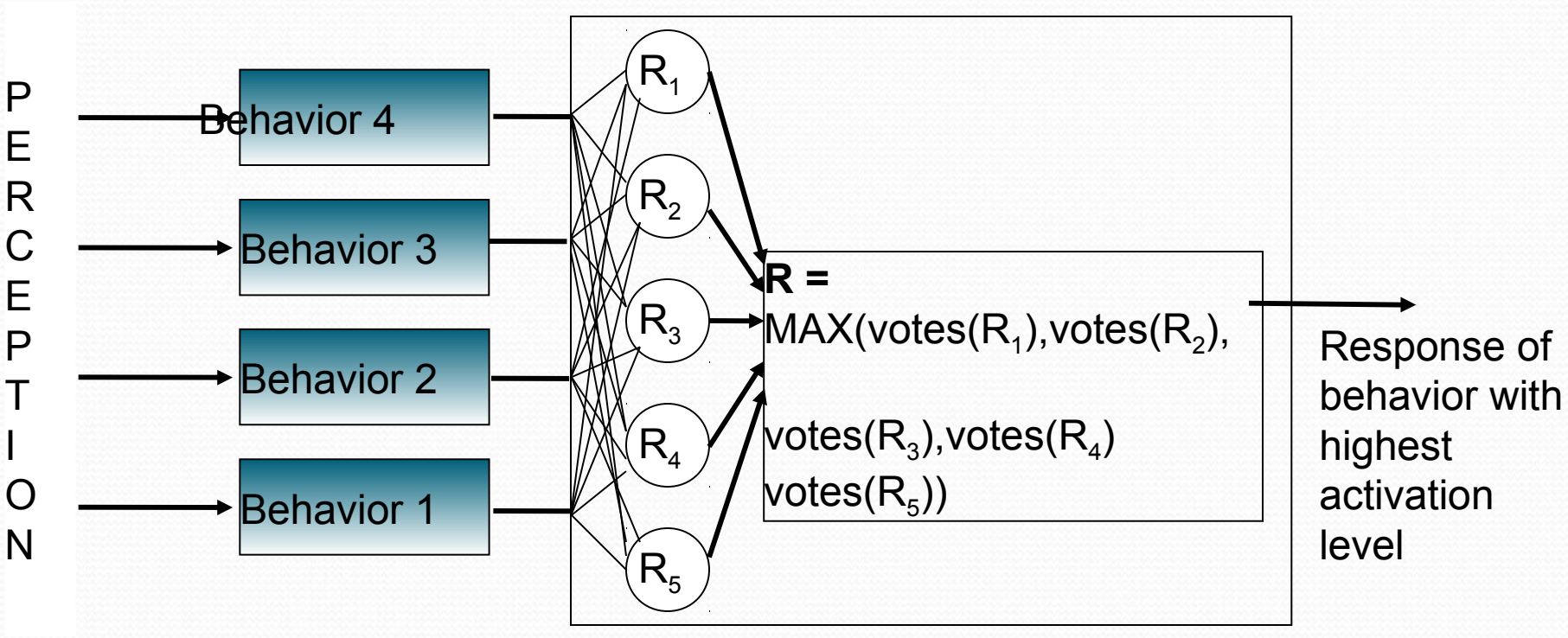
- Arbitration can be:
  - Behaviour Fusion
  - Voting

Cooperative Methods: Behaviour Fusion

- Behavioral fusion provides ability to concurrently use the output of more than one behavior at a time
- Central issue: finding representation amenable to fusion
- Common method:
  - Vector addition using potential fields
  - Example potential field (Architecture B2 lecture):



Cooperative Method: Voting

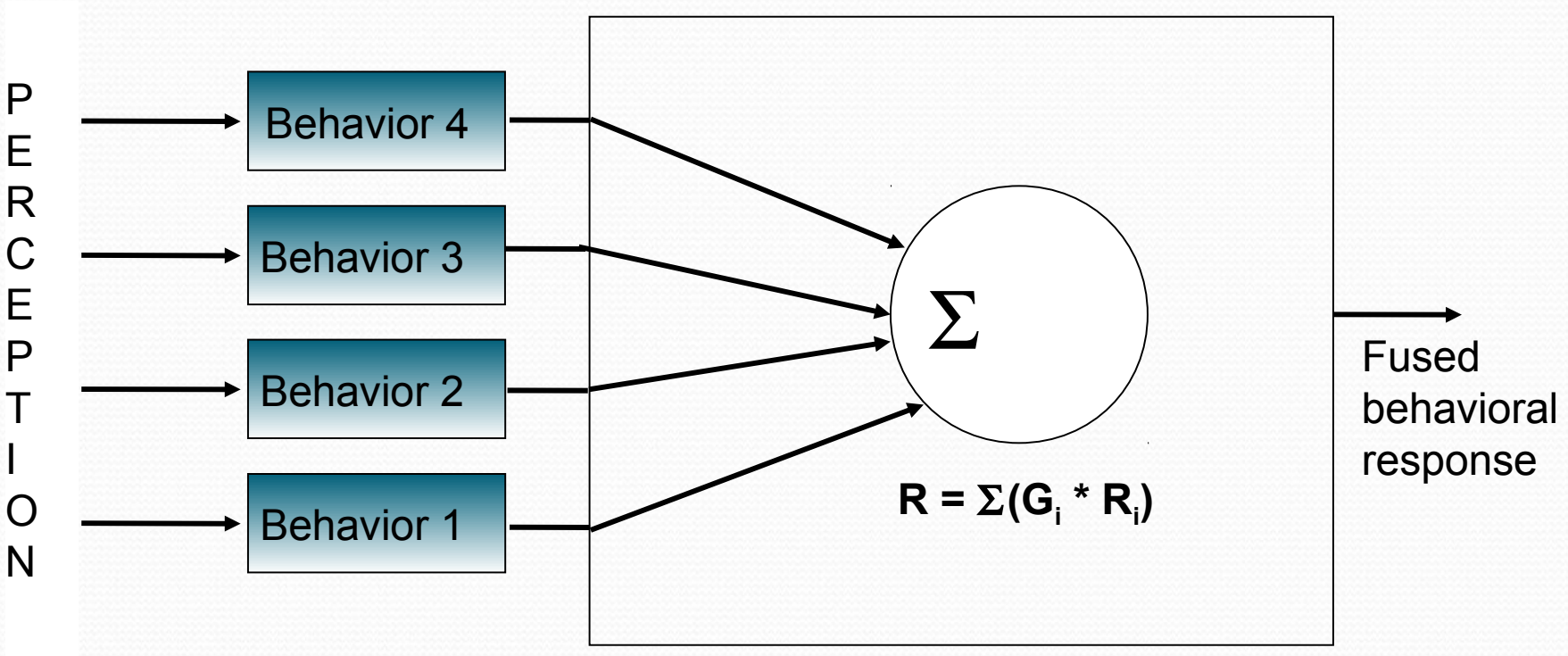


Voting-based coordination

- Pre-defined set of motor responses;
- Each behavior allocates votes (in some distribution) to each motor response
- Motor response with most votes is executed

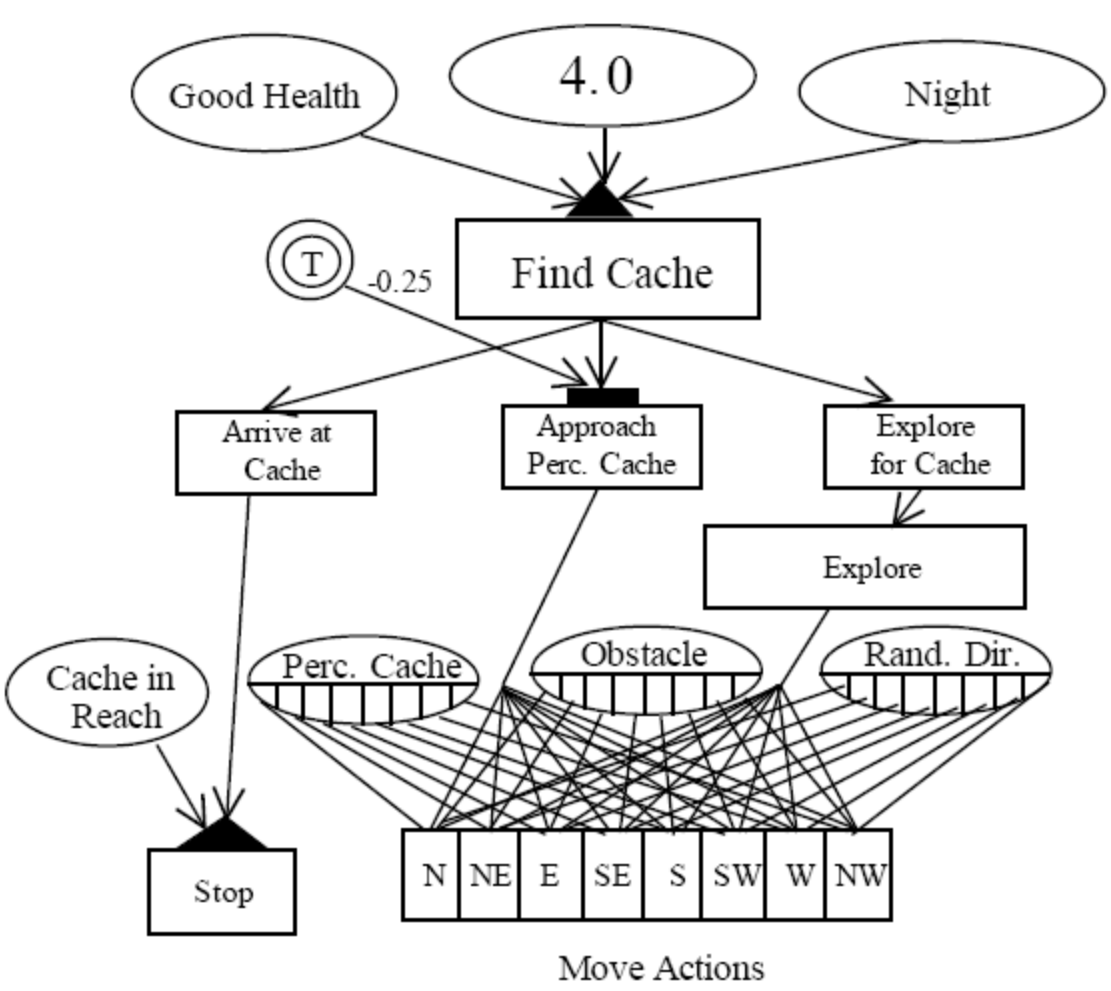
Priorities may vary during mission

Cooperative Method: Behavioral Fusion via Vector Summation



Behavioral fusion

Mixed & Free Flow



BISMARC (Huntsberger & Rose 1998)

Summary

- Competitive Behaviour Arbitration (sequencing, activation networks, prioritization…)
- Competitive Behaviour Arbitration (voting, behaviour fusion)
- Free Flow (e.g. BISMARC, CAMPOUT…)

Suggested Readings

- Brooks, R. A. *"A Robust Layered Control System for a Mobile Robot"*, IEEE Journal of Robotics and Automation, Vol. 2, No. 1, March 1986, pp. 14-23; also MIT AI Memo 864, September 1985.
- Robot Programming: A Practical Guide to Behavior-based Robotics, Joseph L. Jones, McGraw-Hill, 2004.
- Motor Schema Based Navigation for a Mobile Robot: An Approach to Programming by Behavior, Ron Arkin, Proc of ICRA, 1987, pp 265-271.
- Behavior-based control: Main properties and Implications, Maja Mataric, *Proceedings, IEEE International Conference on Robotics and Automation, Workshop on Architectures for Intelligent Control Systems*, Nice, France, May 1992, 46-54.
- Pirjanian, P. (1999) *Behaviour coordination mechanisms - state-of-the-art. Technical Report IRIS-99-375, Institute for Robotics and Intelligent Systems, School of Engineering, University of Southern California.*
- Bonissone, P. (1996) *Fuzzy Logic and Fuzzy Logic Controllers: An Introduction, P. Bonissone, in Genetic Algorithms and Soft Computing, F. Herrera and J.L. Verdegay (eds.), pp. 30--47, Physica-Verlag (Heidelberg, Germany), 1996.*
- Ulrich, I.; Borenstein, J. (1998). "VFH+: reliable obstacle avoidance for fast mobile robots". *Robotics and Automation, 1998. Proceedings. 1998 IEEE International Conference on*.
- Schoppers, M. J. 1987. Universal plans for reactive robots in unpredictable environments. In Proceedings of the Tenth International Joint Conference on Artificial Intelligence, pp. 1039-1046 Milan. IJCAI.