Toward Process Architectures for Behavioural Robotics

Jon Simpson
J.Simpson@kent.ac.uk
School of Computing, University of Kent.

How should we structure process-oriented behavioural control programs?

Architectures for behavioural robotics in the context of process-oriented programming.

Distil design principles and components for behavioural control via process architectures.

Behavioural robotics.

Behaviour-based AI uses a modular decomposition of the system’s intelligence into, generally, a three-layer system.

Three-layer systems.

Deliberative & reactive components, co-ordinated by and co-existing with a support layer.

Behavioural robotics requires concurrency.

Use a concurrent language for implementation.

occam-pi

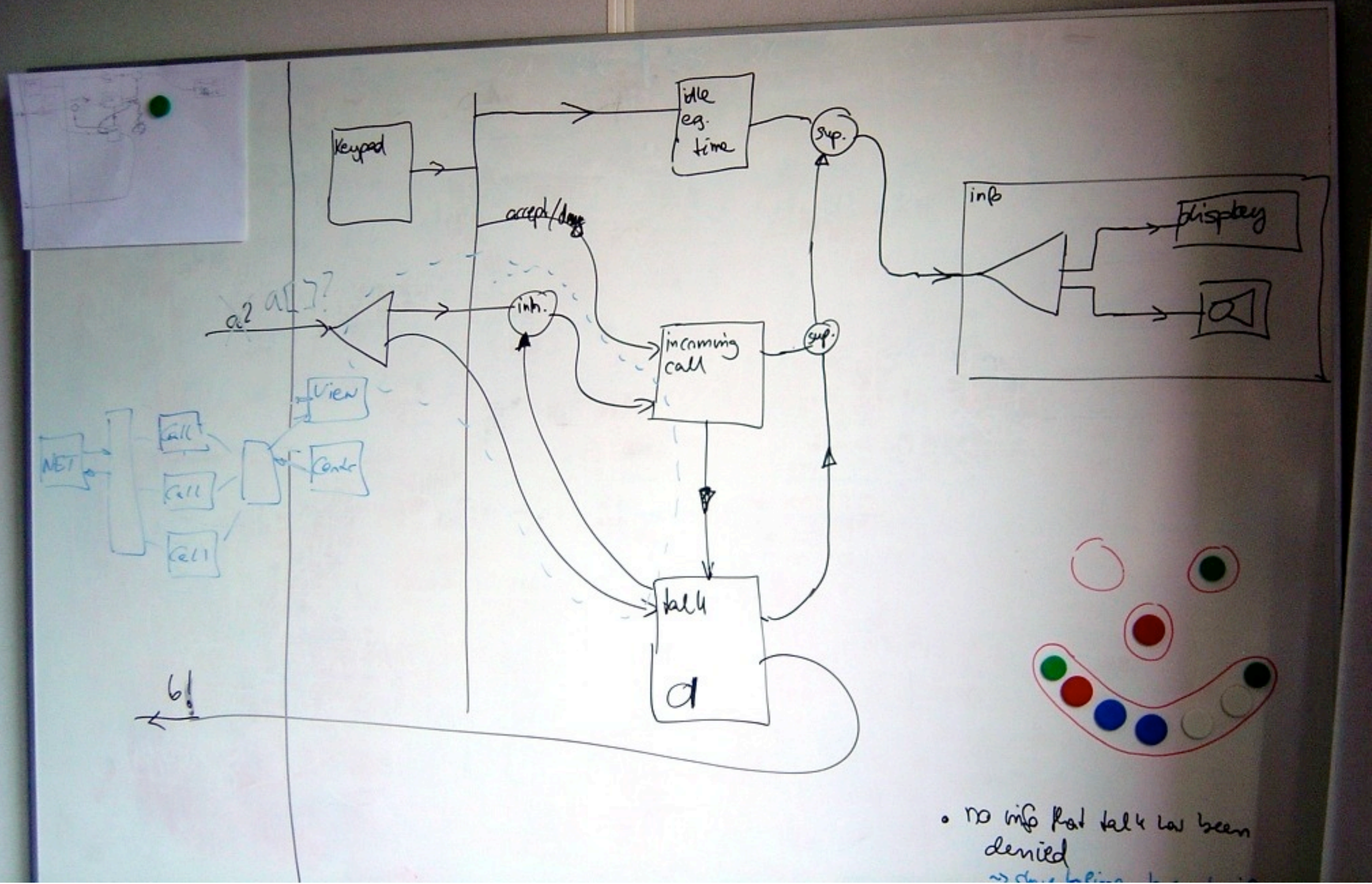
Process-oriented programming language.

Runtime support for small robotics platforms.

Surveyor SRV-1, Mindstorms RCX, IPRE Scribbler, Pioneer 3-DX.

Process-oriented programming.

Networks of concurrent processes communicating synchronously via channels.



- no info that talk has been denied

Architectures

Principles and methodologies for complexity management in system development.

Previous Work

- Subsumption Architecture in occam-pi (2006)
- Visual Process-oriented Programming for Robotics (2008)

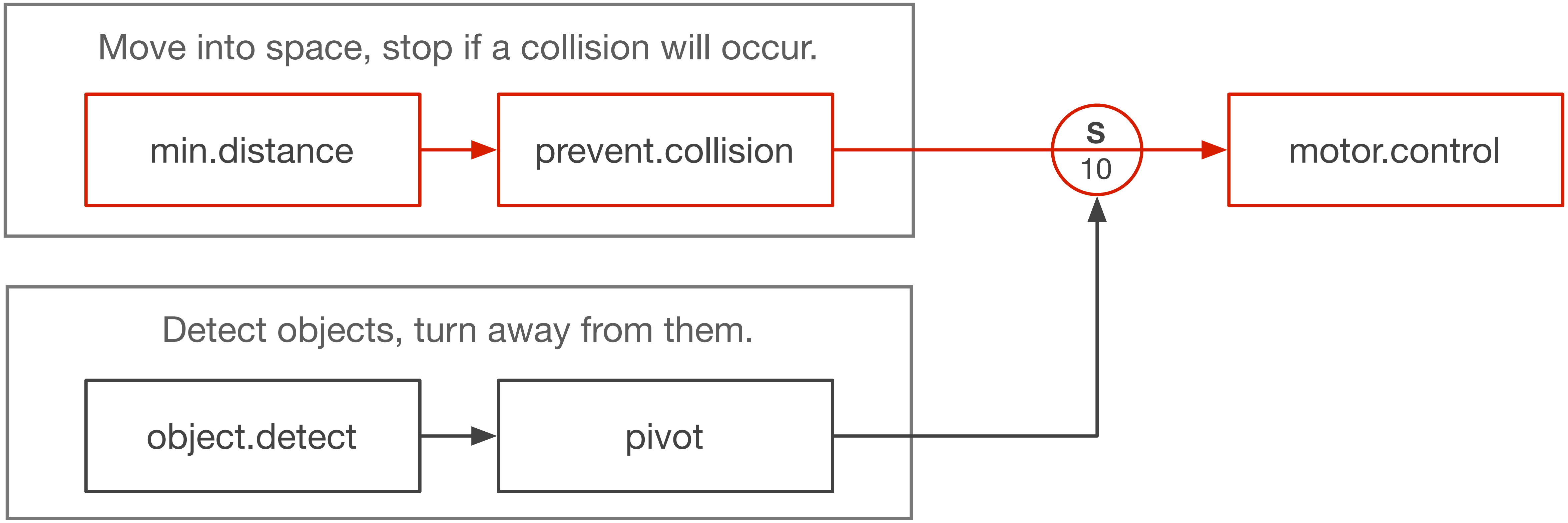
Subsumption Architecture

Brooks (1984, 1986)

Subsumption Architecture

Communicating modules form ‘levels of competence.’

Suppression of inputs and inhibition of outputs.



Move into space, stop if a collision will occur.

min.distance

prevent.collision

S

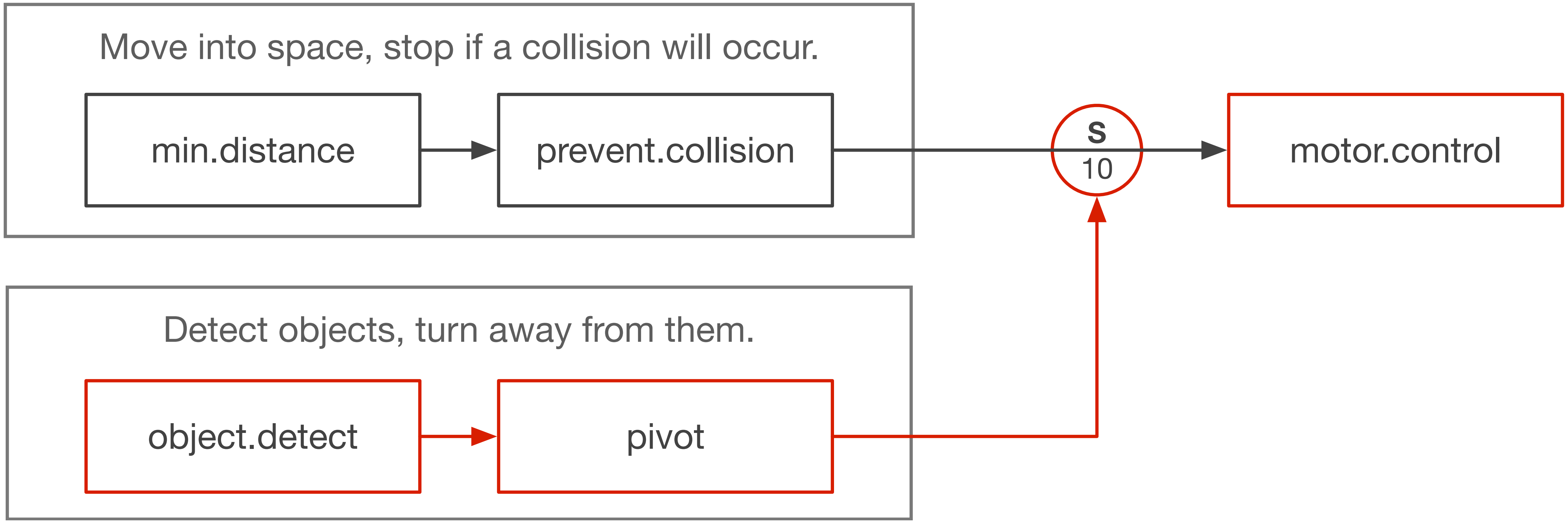
10

motor.control

Detect objects, turn away from them.

object.detect

pivot



Move into space, stop if a collision will occur.

min.distance

prevent.collision

S

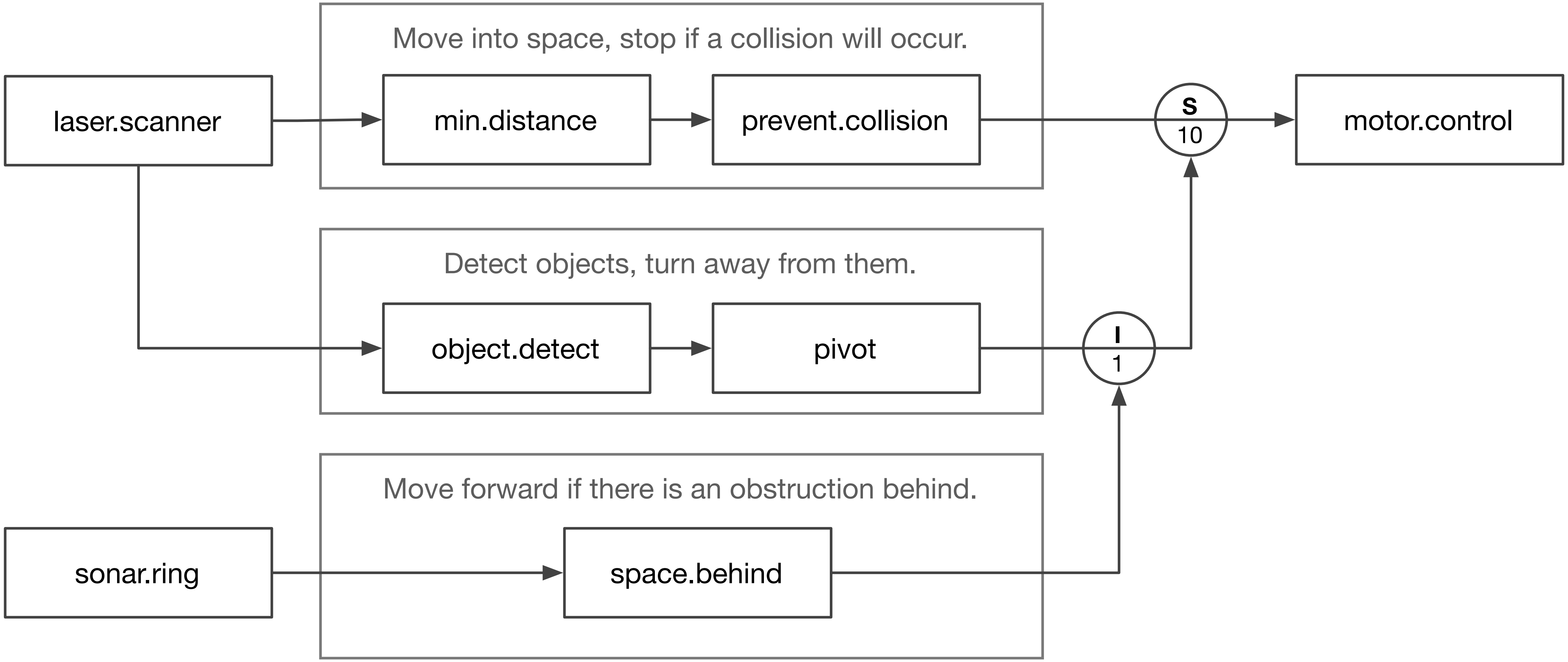
10

motor.control

Detect objects, turn away from them.

object.detect

pivot



Move into space, stop if a collision will occur.

laser.scanner

min.distance

prevent.collision

S

10

motor.control

Detect objects, turn away from them.

object.detect

pivot

I

1

Move forward if there is an obstruction behind.

sonar.ring

space.behind

Benefits

Inhibitor and suppressor primitives.

Structural layers of behaviour modules.

Problems

Tight layer interactions negatively effect scalability.

Modules spy on lower modules, creating hidden dependencies.

Rich process implementation versus state machines.

Colony Architecture

Connell (1987)

Colony Architecture

Refinement of the subsumption architecture.

Layers not strictly hierarchical.

No inhibition, only suppression at behaviour edges.

Colony Architecture

Improves scalability, allowing only behaviour interactions.

Changes fed back into a later revision of Subsumption.

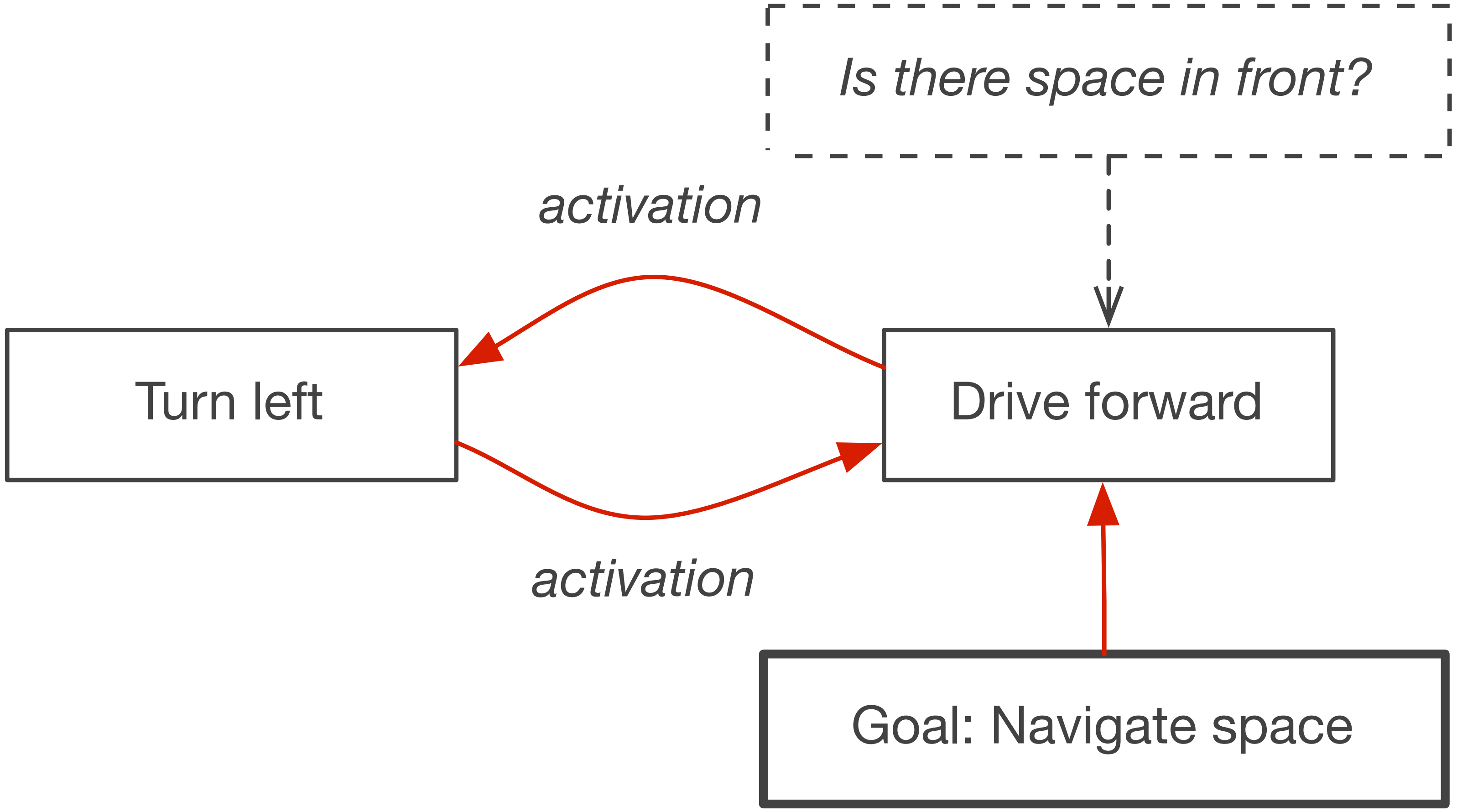
Action-Selection

Maes (1989)

Action Selection

Modules controlled via activation levels.

Primed depending on action and the environment.



Is there space in front?

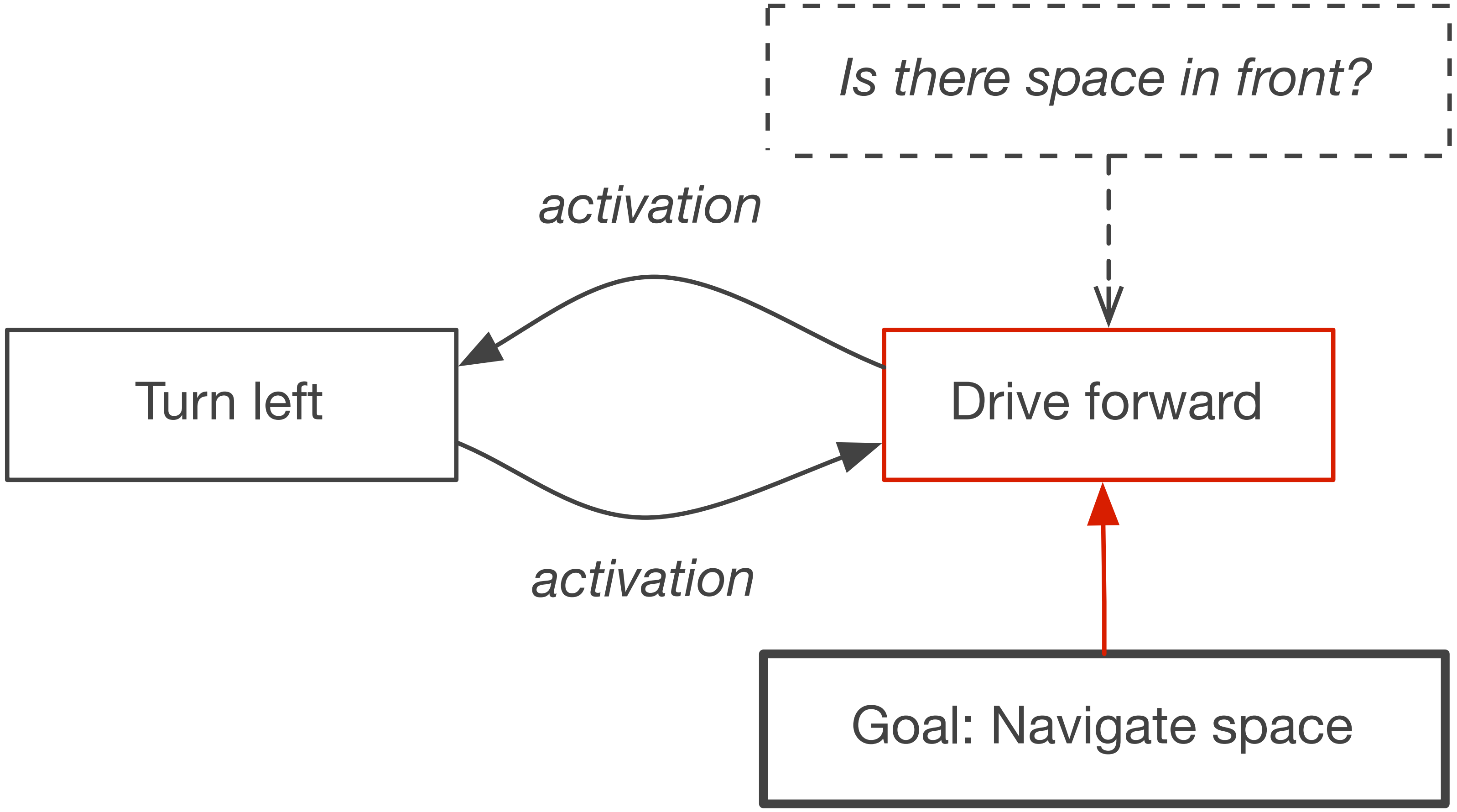
activation

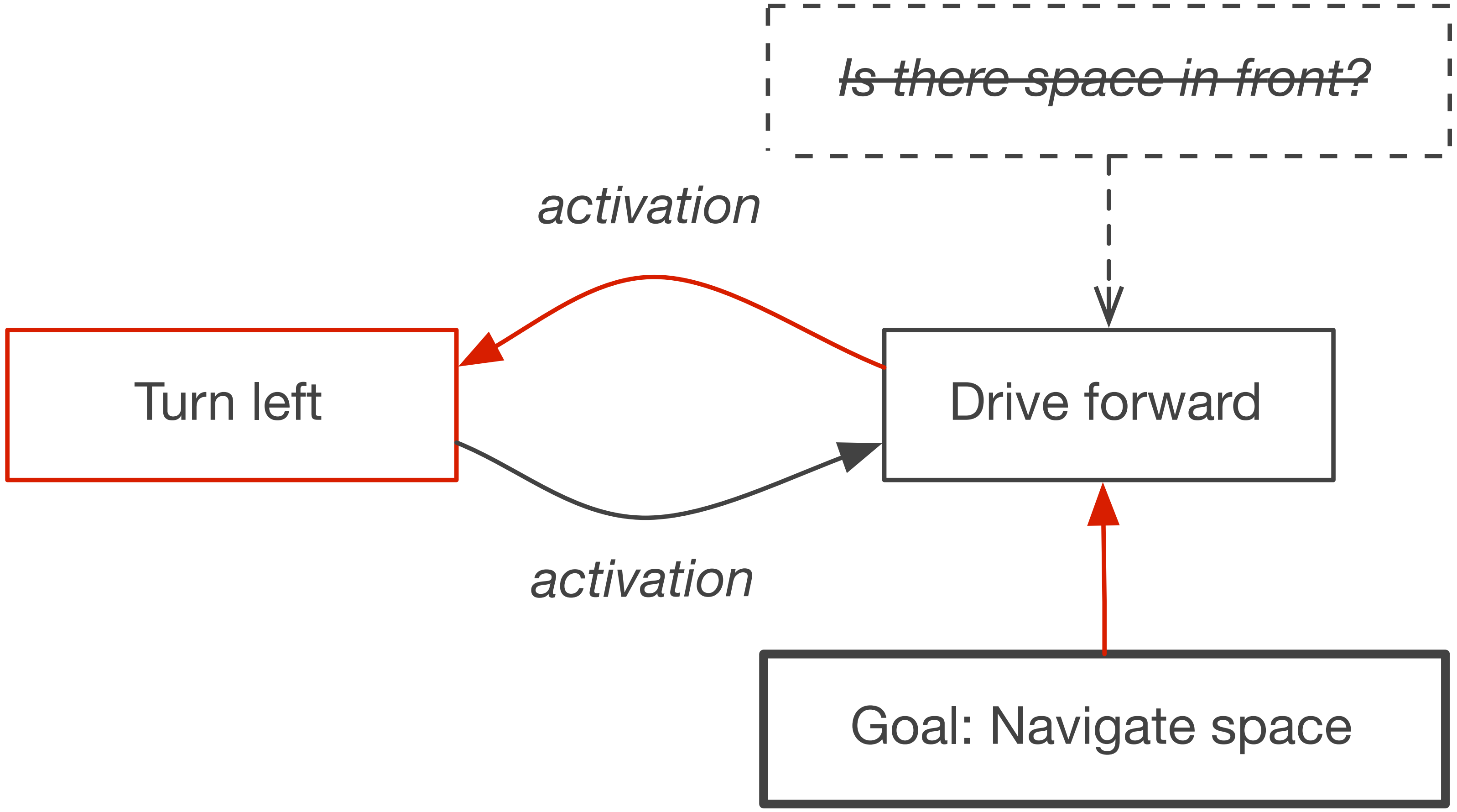
Turn left

Drive forward

activation

Goal: Navigate space


Is there space in front?
activation
Turn left
Drive forward
activation
Goal: Navigate space



~~*Is there space in front?*~~

activation

Turn left

Drive forward

activation

Goal: Navigate space

Action Selection

Interesting, but leads to highly connected networks.

Simplest implementation uses a separate decision network.

Motor Schema

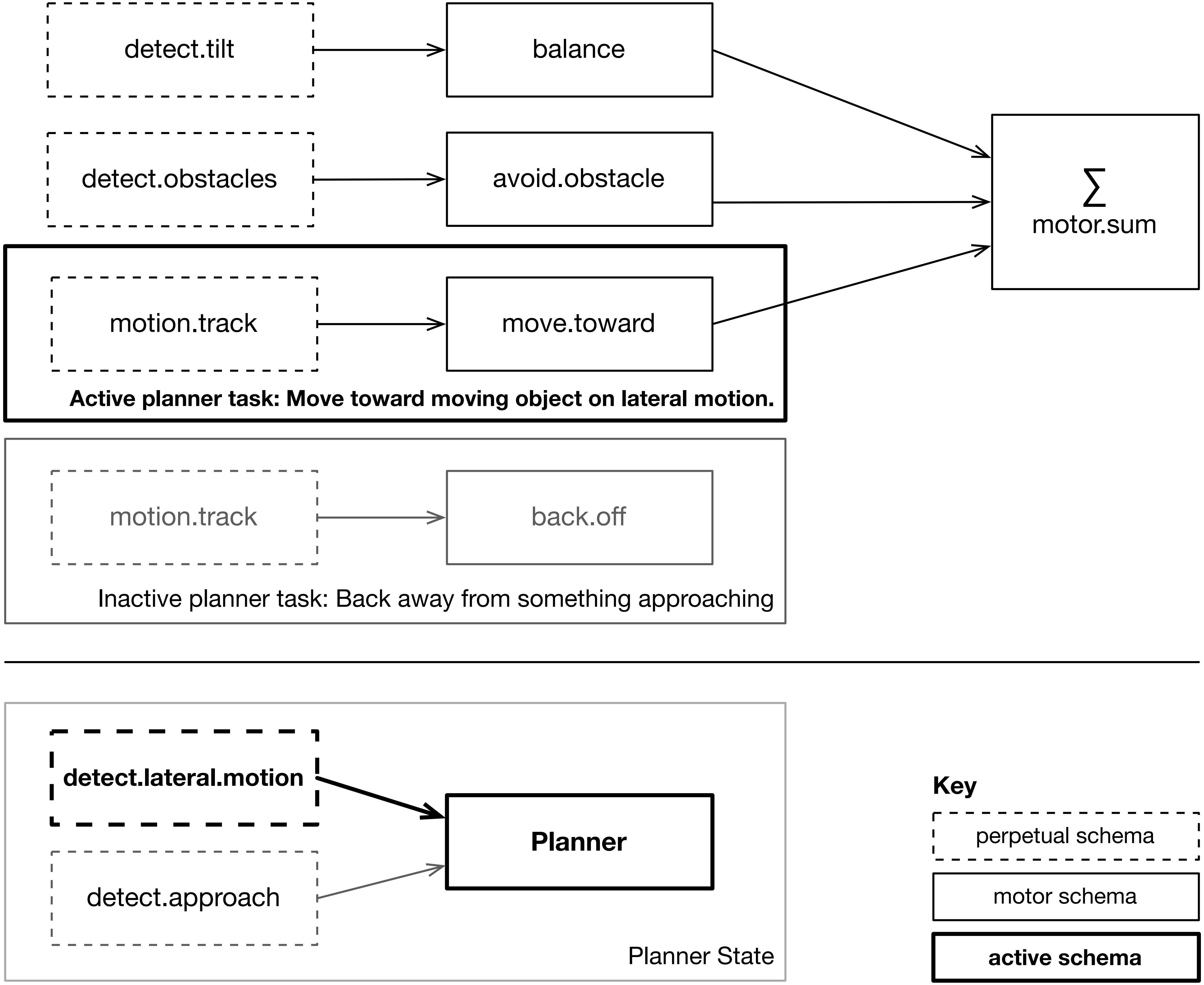
Arkin (1987)

Motor Schema

Perceptual schemata identify features and conditions in the environment, providing data to motor schema.

Motor schemata control the motion or activity of the robot.

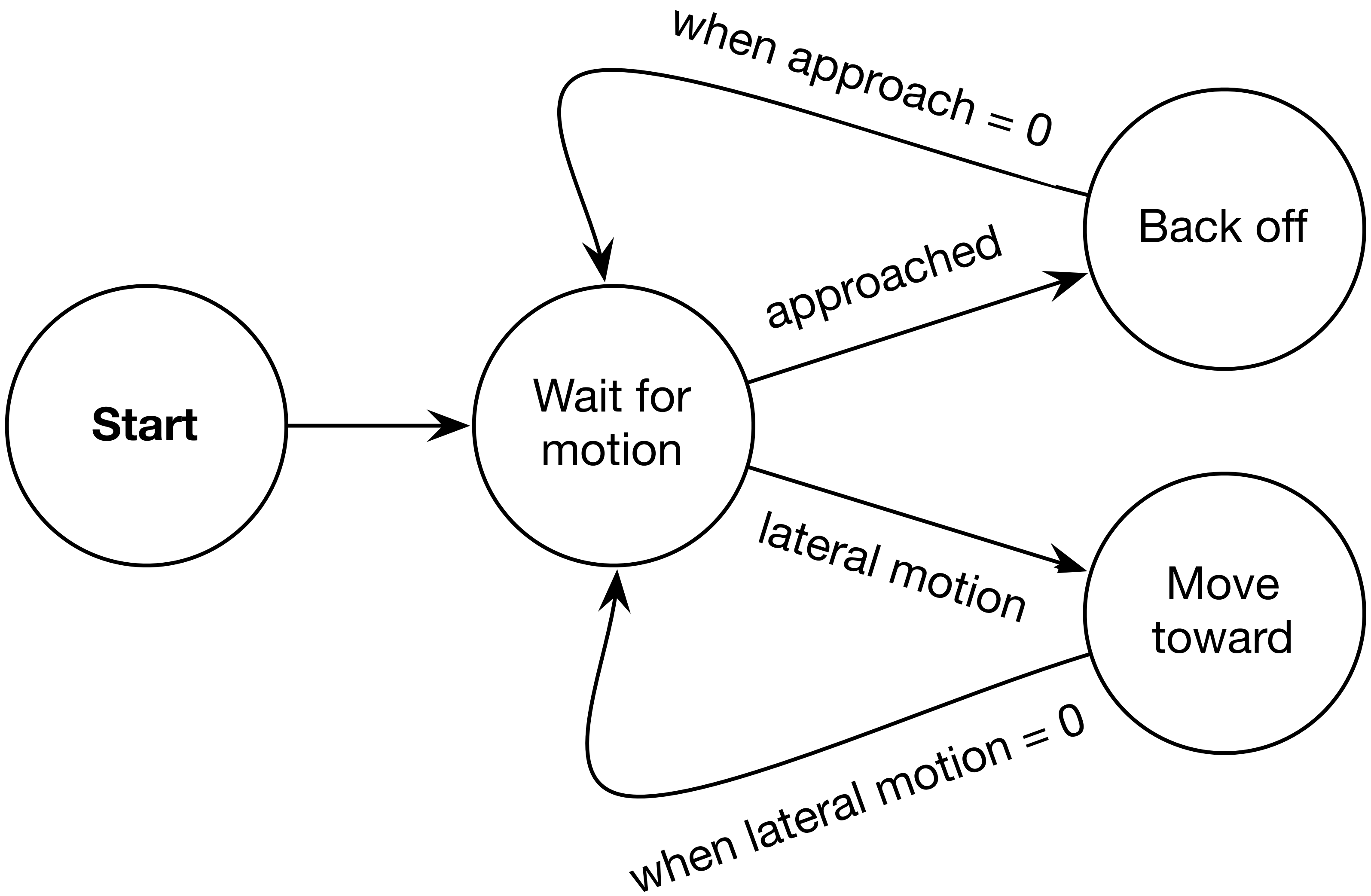
Groupings of the two are known as assemblages.


detect.tilt
balance
detect.obstacles
avoid.obstacle
∑
motor.sum
motion.track
move.toward
Active planner task: Move toward moving object on lateral motion.
motion.track
back.off
Inactive planner task: Back away from something approaching
detect.lateral.motion
Planner
detect.approach
Planner State
Key
perpetual schema
motor schema
active schema

Planner

Perceptual schemas build a state machine planner.

Planner can load assemblages based on conditions.


Start
Wait for motion
Back off
Move toward
approached
lateral motion
when approach = 0
when lateral motion = 0

Motor Schemas

Design rules plus a ‘vector.sum’ primitive.

Separation between planning and sensing/acting.

State machines produce simple reasoning.

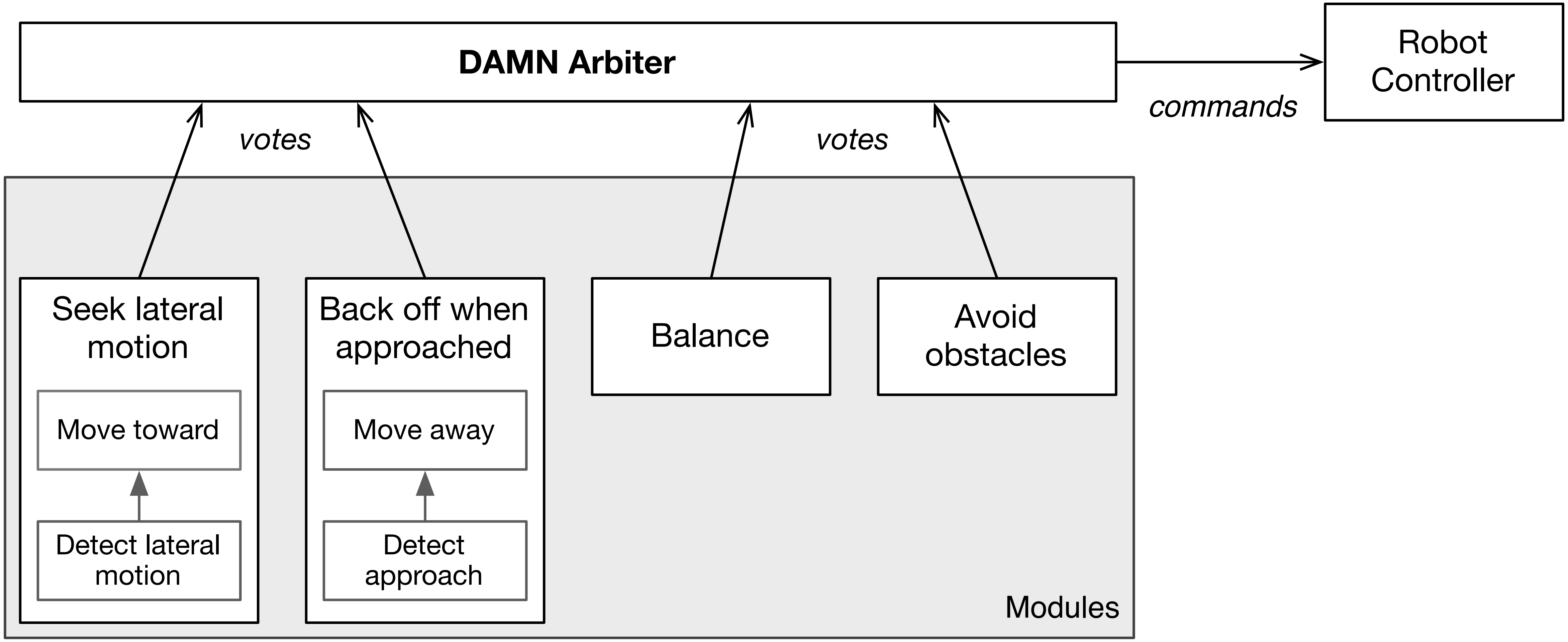
Distributed Architecture for Mobile Navigation

Rosenblatt (1995)

DAMN

Arbiters using voting to perform command fusion.

Weighting of votes can be fixed or altered using a mode manager for sequential action.


DAMN Arbiter
Robot Controller
commands
votes
votes
Seek lateral motion
Move toward
Detect lateral motion
Back off when approached
Move away
Detect approach
Balance
Avoid obstacles
Modules

DAMN

Re-use from Motor Schemas of sensing and activity.

Connected between behaviours and arbiter.

Arbiter and control process take advantage of expressive processes.

Quantitative Evaluation

- Code metrics for solving specific tasks.
  - Processes or LOC
  - Good measures of complexity?

The Road Ahead

More behavioural architectures.

ATLANTIS (Gatt 1992), SSS (Connell 1992).

From process-orientation toward robot architectures.

Structures & tooling for visual programming.

Thank you.

Questions welcome.