Aalto University
School of Electrical
Engineering

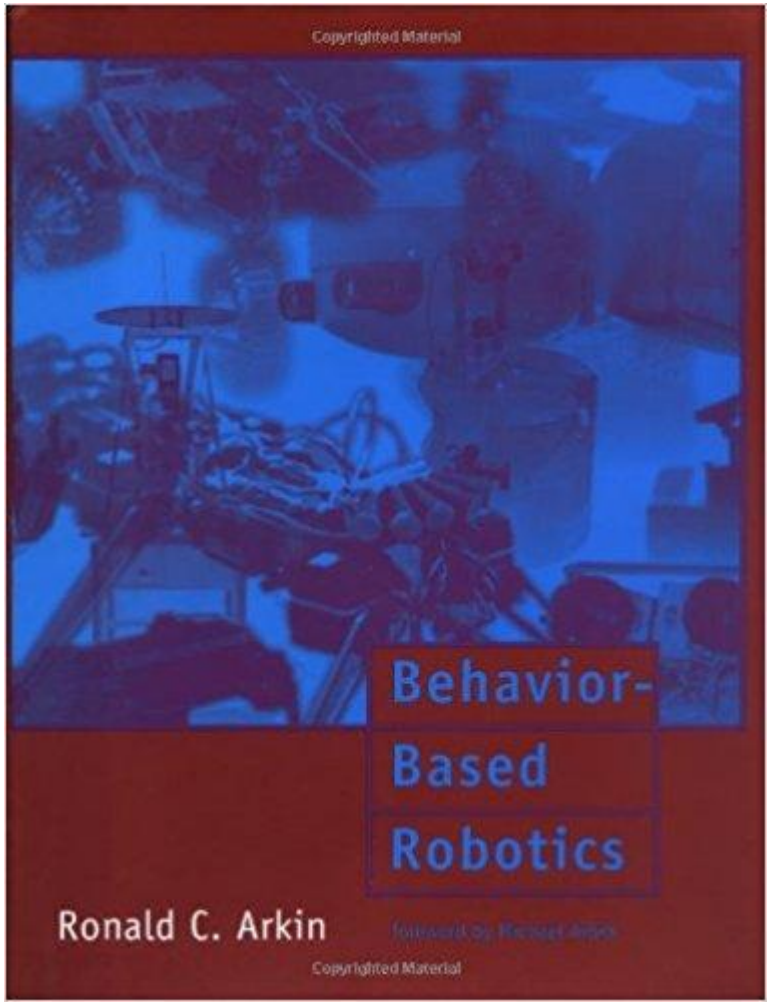
ELEC-E8111 – Autonomous Mobile Robots

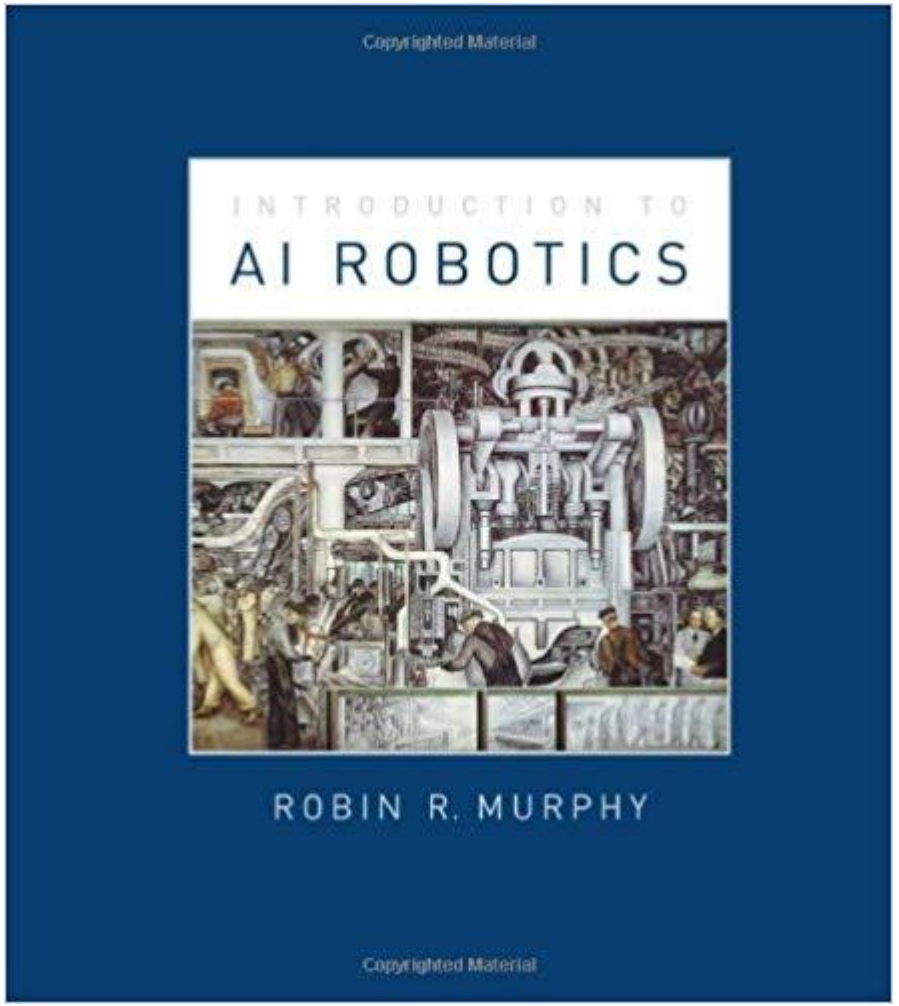
CONTROL ARCHITECTURES

Mika Vainio, Research Fellow, Aalto

Contents of the lecture

- Motivation and state-of-the-art issues
- Short introduction to control paradigms
- The Hierarchical Paradigm
- Biological foundations for the reactive paradigm
- The Reactive Paradigm
- The Hybrid Paradigm
- AuRA Architecture (example of Hybrid solutions)





MAINLY BASED ON THESE TWO BOOKS BY RONALD ARKIN AND ROBIN MURPHY

Robotics (along with information technology and biotechnology) is identified as one of the key technologies of the future (the new millennium)

CEO Honda

SMACK IN THE MIDDLE

Big Data

Data Mining

Virtual Reality

Machine Learning

Deep Learning

ROBOTICS

Blockchain

Augmented reality

Artificial Intelligence

What is a robot?

- A robot is a re-programmable, multi-functional, manipulator designed to move material, parts, or specialized devices though variable programmed motions for the performance of a task (Robotics Industry Association)

- An intelligent robot is a machine able to extract information from its environment and use knowledge about its world to move safely in a meaningful and purposeful manner. (Arkin)

- A robot is a system which exists in the physical world and autonomously senses its environment and acts in it. (Mataric)

What makes a MOBILE robot?

- sensors
- effectors/actuators
- locomotion system
- on-board computer system

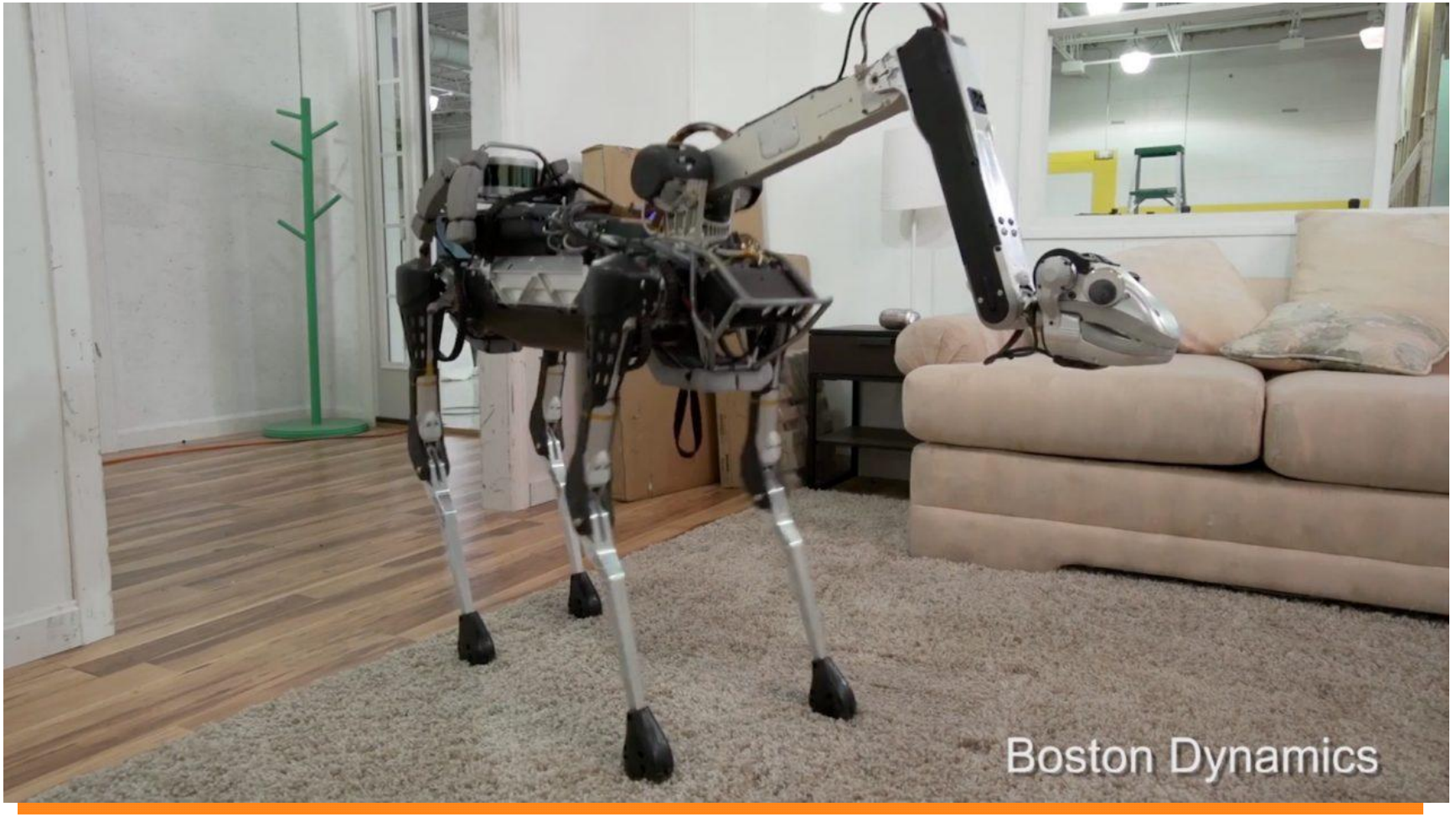
Why is Robotics so hard in reality?

- Sensors are limited and crude
- Effectors are limited and crude
- State (internal and external, but mostly external) is partially-observable
- Environment is dynamic (changing over time)
- Environment is full of potentially-useful information

Key Issues

- Grounding in reality: not just planning in an abstract world
- Situatedness: tight connection with the environment
- Embodiment: having a body
- Emergent behavior: interaction with the environment
- Scalability: increasing task and environment complexity

REAL WORLD TESTING - SpotMini by Boston Dynamics



What is autonomy?

- the ability to make one's own decisions and act on them

- for robots, the ability to sense the situation and act on it appropriately

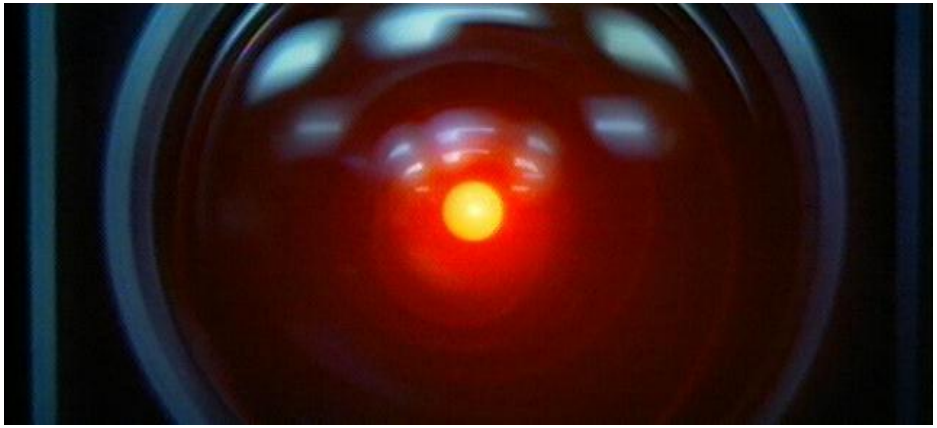
…Autonomy

- Autonomy can be complete, as in autonomous robots, or partial, as in tele-operated robots.

- examples of autonomous robots: R2D2

- examples of tele-operated robots: NASA's robots before Pathfinder

REALLY SHORT HISTORY



FILM by S. Kubrick

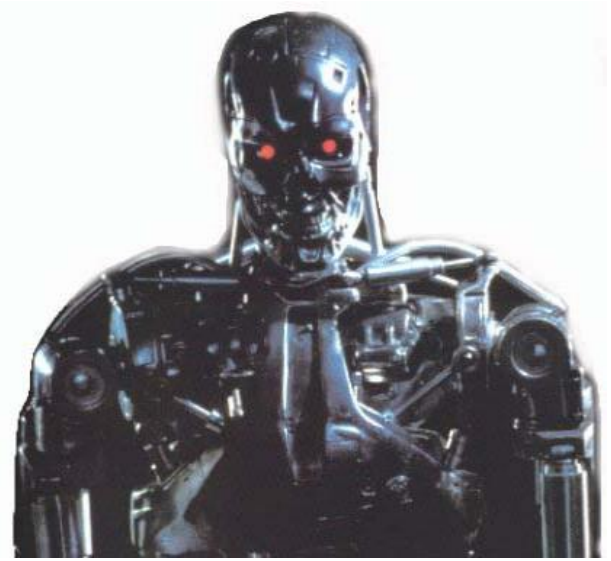
2001: A Space Odyssey

(1968)

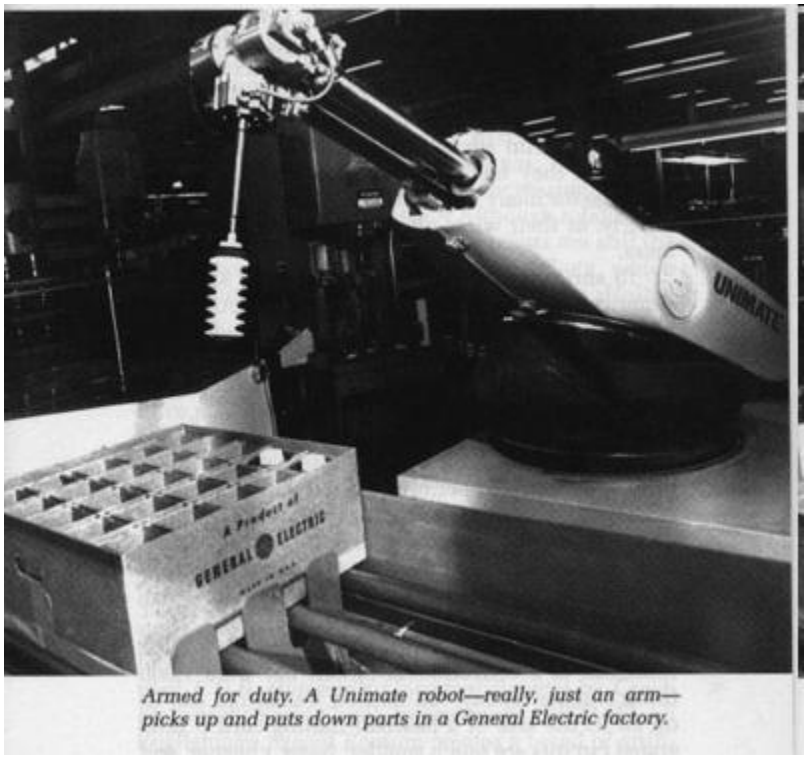
From scifi...

FILM by J. Cameron

The Terminator (1985)



...to factories



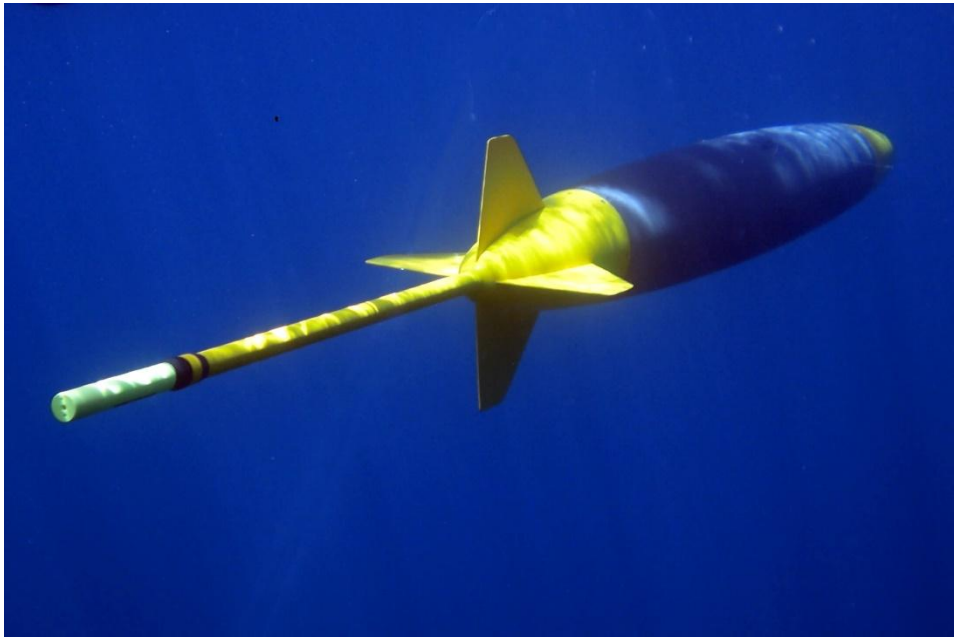
Armed for duty. A Unimate robot—really, just an arm—picks up and puts down parts in a General Electric factory.

In 1961 the first industrial robot, Unimate, joined the assembly line at a General Motors plant.

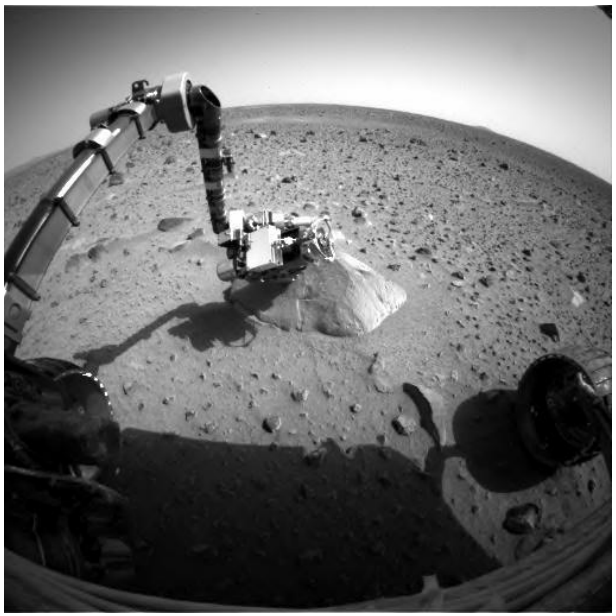
ABB Industrial manipulator robot

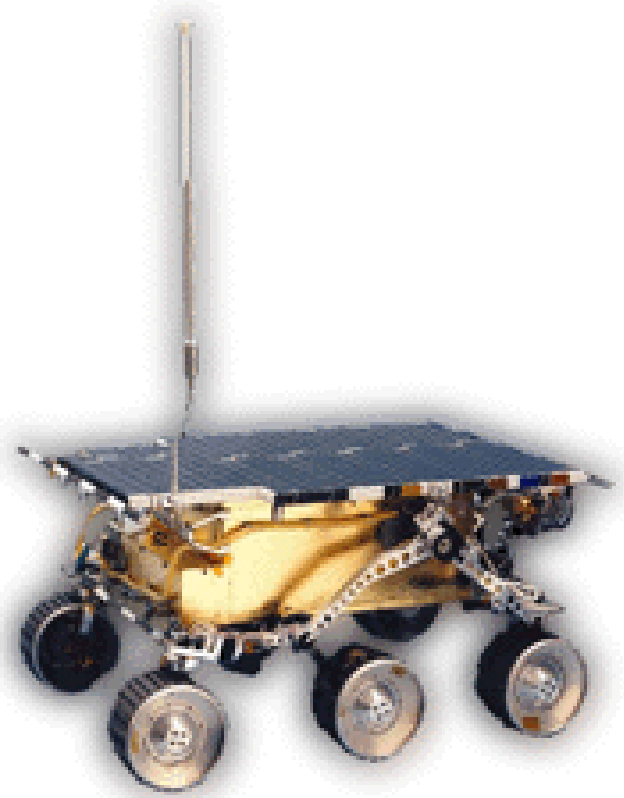
Automatically Guided Vehicles

From volcanoes down to abyss



To other planets...

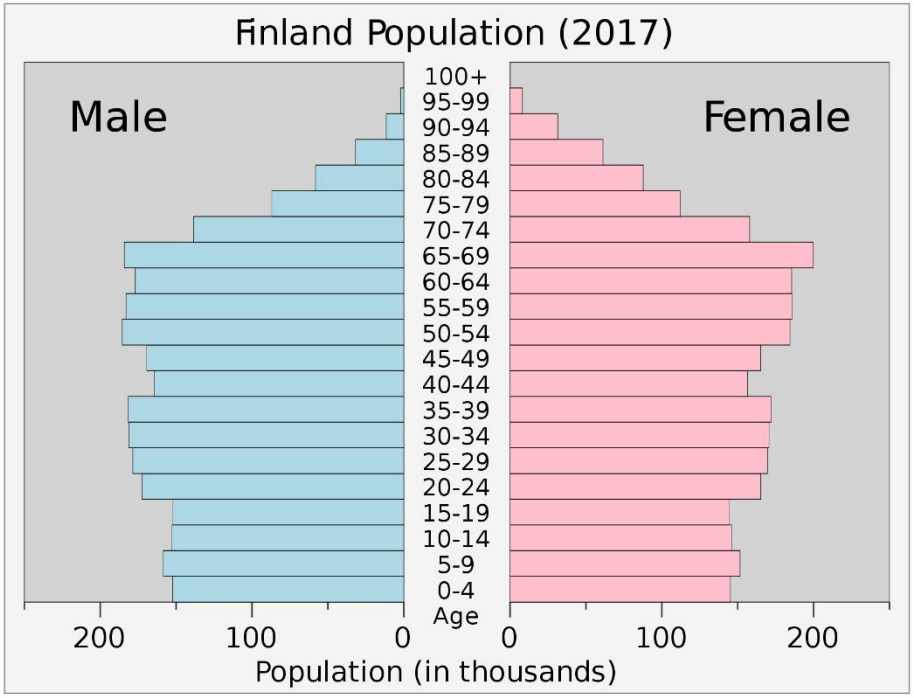




and even to our homes.

THE (fantastic) FUTURE:
4 focal areas

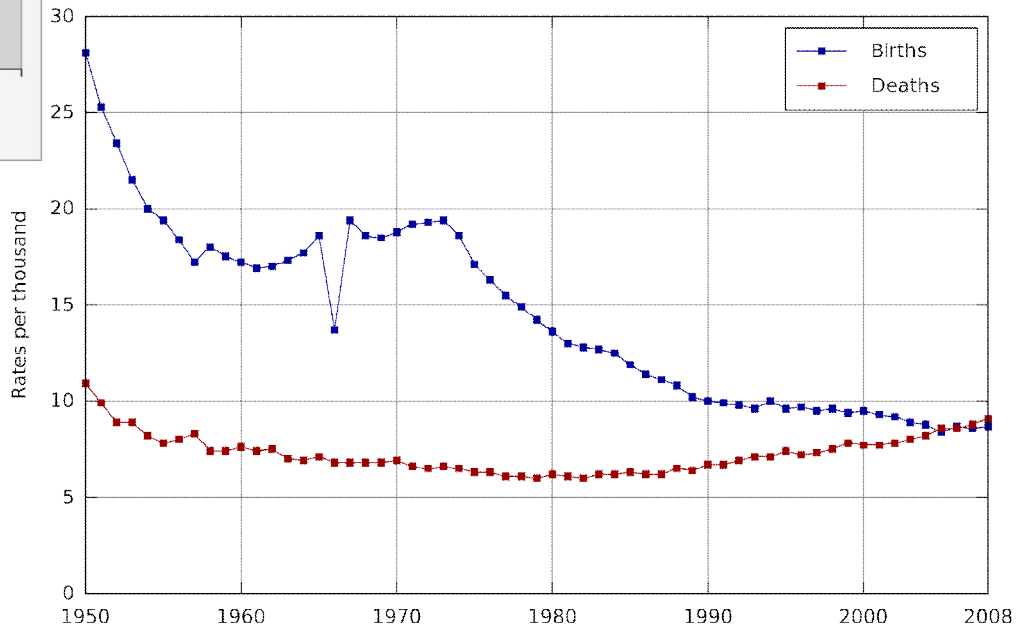
Military robotics. Scary stuff. Skipping this one.



Finnish population pyramid in 2017

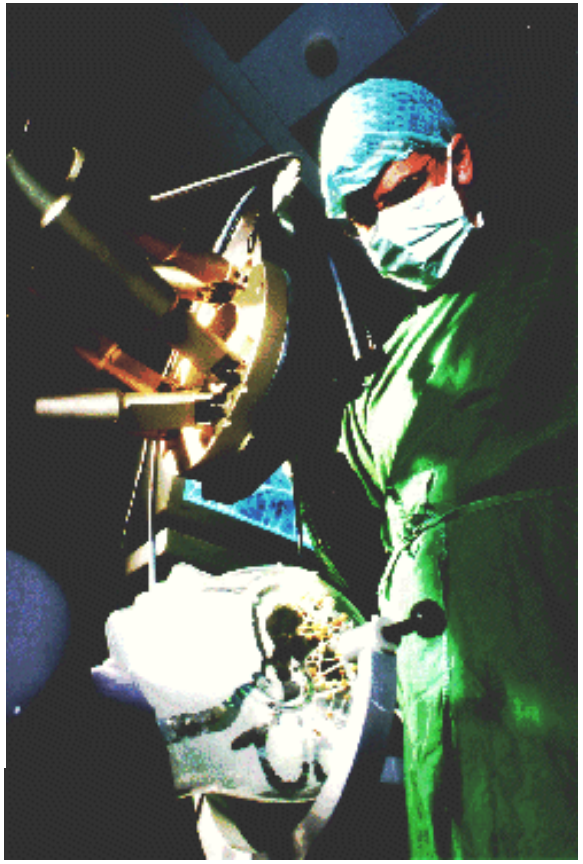
FOCUS 1

Japanese birth and death rates



AGING POPULATIONS

In hospital, elderly and disabled care



Social aspects, logistics, surgery, rehabilition,etc.

Left: Paro Therapeutic Robot by AIST
Middle: HelpMate robotic materials transport system by HelpMate Robotics, Inc
Right: da Vinci Surgical System by Intuitive Surgical

SELF-DRIVING CARS

FOUNDATION – NAVLAB project

Started 1984
The Robotics Institute at the School of Computer Science, Carnegie Mellon University

In July 1995, the team took Navlab5 from Pittsburgh to San Diego on a proof-of-concept trip, dubbed "No Hands Across America", with the system navigating for all but 50 of the 2850 miles, averaging over 60 MPH.

QUANTUM LEAP- DARPA Grand Challenge

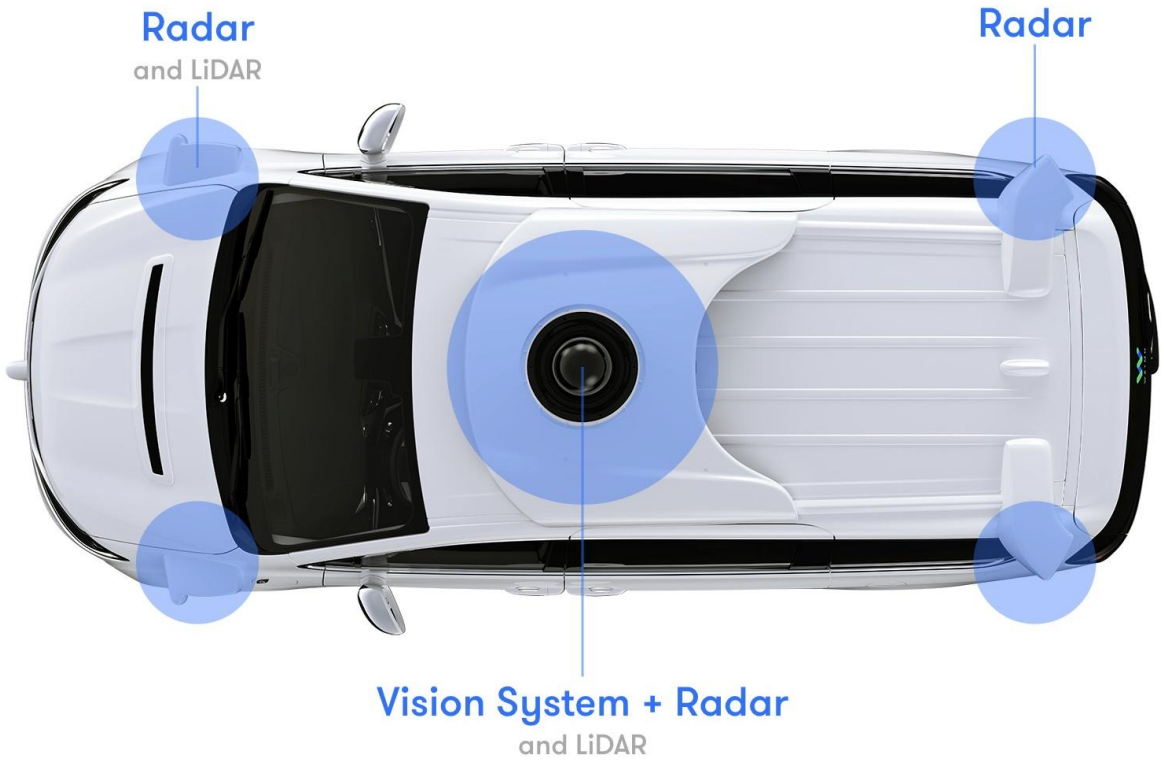
The first competition of the DARPA Grand Challenge was held on March 13, 2004 in the Mojave Desert region of the United States, along a 150-mile (240 km) route. None of the robot vehicles finished the route. No winner was declared, and the cash prize was not given.

The second competition of the DARPA Grand Challenge began at 6:40am on October 8, 2005. All but one of the 23 finalists in the 2005 race surpassed the 11.78 km (7.32 mi) distance completed by the best vehicle in the 2004 race. Five vehicles successfully completed the 212 km (132 mi) course.

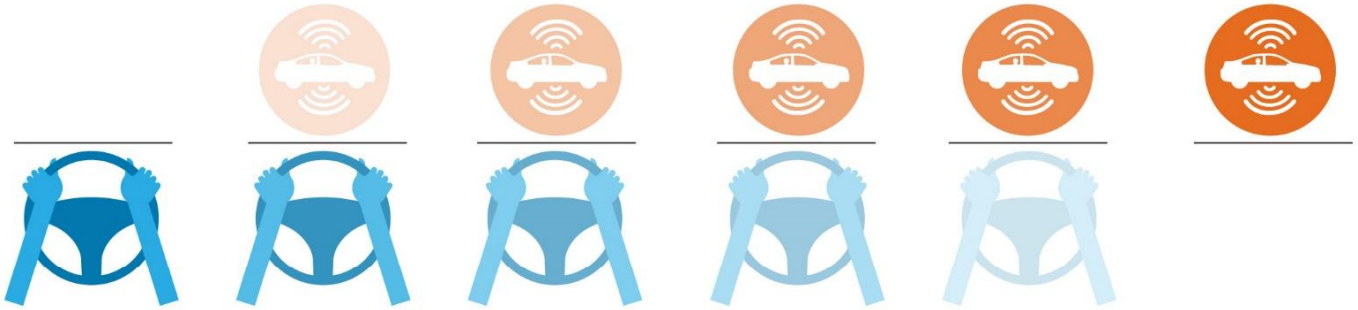

cruise


cruise
JOIN THE DRIVERLESS REVOLUTION


WAYMO


Radar
and LiDAR
Radar
Vision System + Radar
and LiDAR

Five Levels of Vehicle Autonomy



Level 0

No automation: the driver is in complete control of the vehicle at all times.

Level 1

Driver assistance: the vehicle can assist the driver or take control of either the vehicle's speed, through cruise control, or its lane position, through lane guidance.

Level 2

Occasional self-driving: the vehicle can take control of both the vehicle's speed and lane position in some situations, for example on limited-access freeways.

Level 3

Limited self-driving: the vehicle is in full control in some situations, monitors the road and traffic, and will inform the driver when he or she must take control.

Level 4

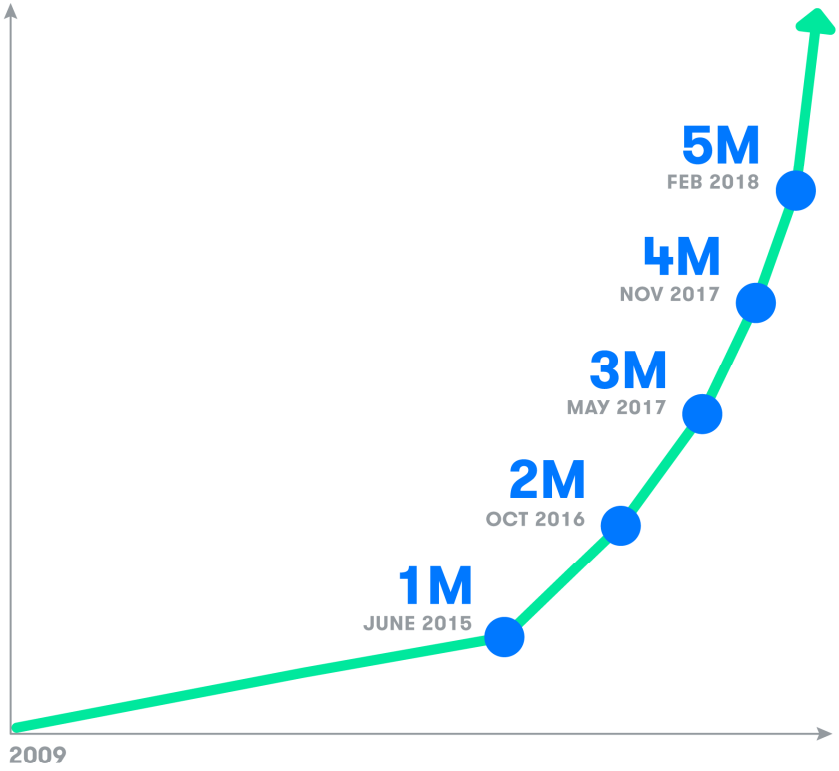
Full self-driving under certain conditions: the vehicle is in full control for the entire trip in these conditions, such as urban ride-sharing.

Level 5

Full self-driving under all conditions: the vehicle can operate without a human driver or occupants.

Source: SAE & NHTSA

ENTENSIVE TESTING



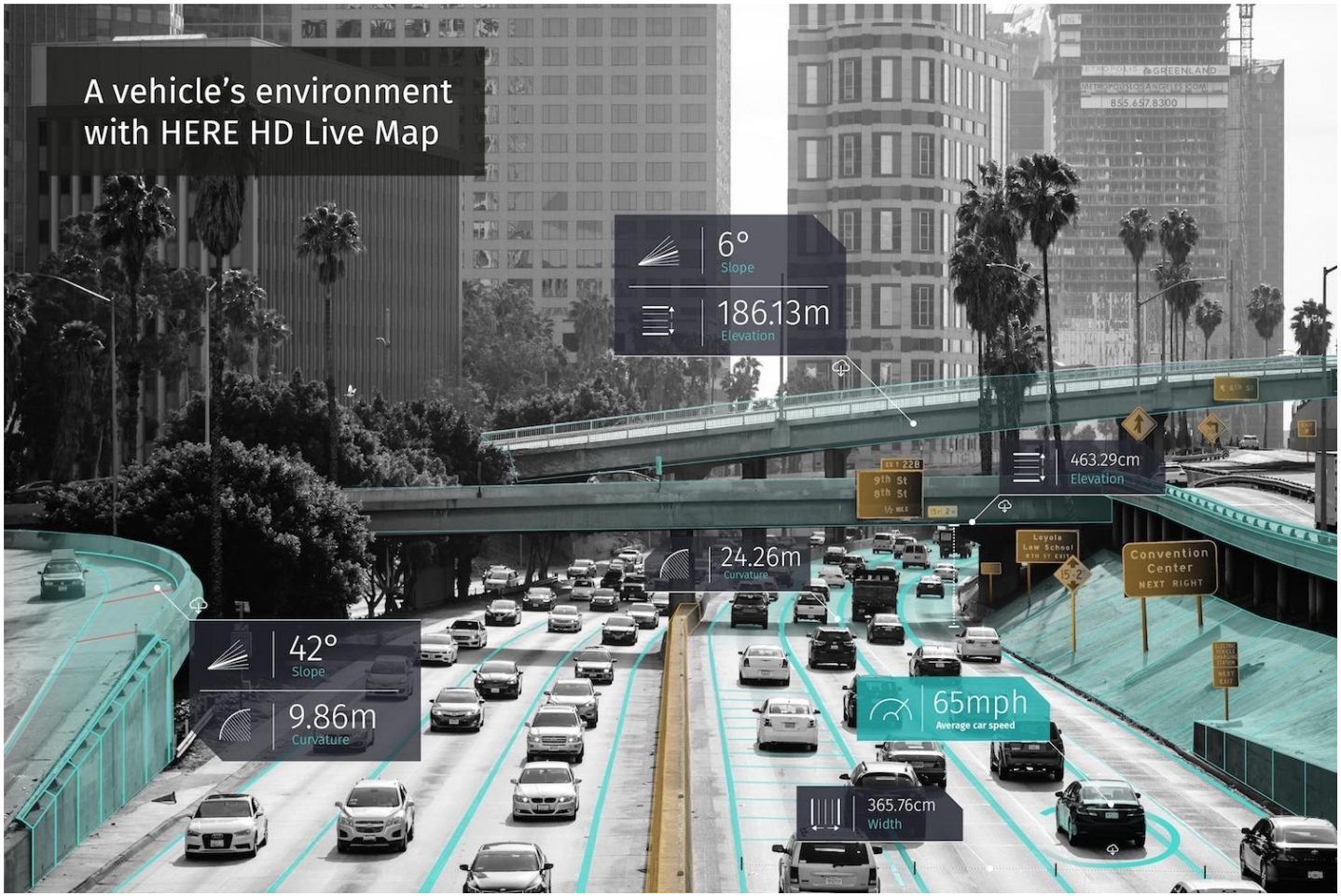
WAYMO

5,000,000+

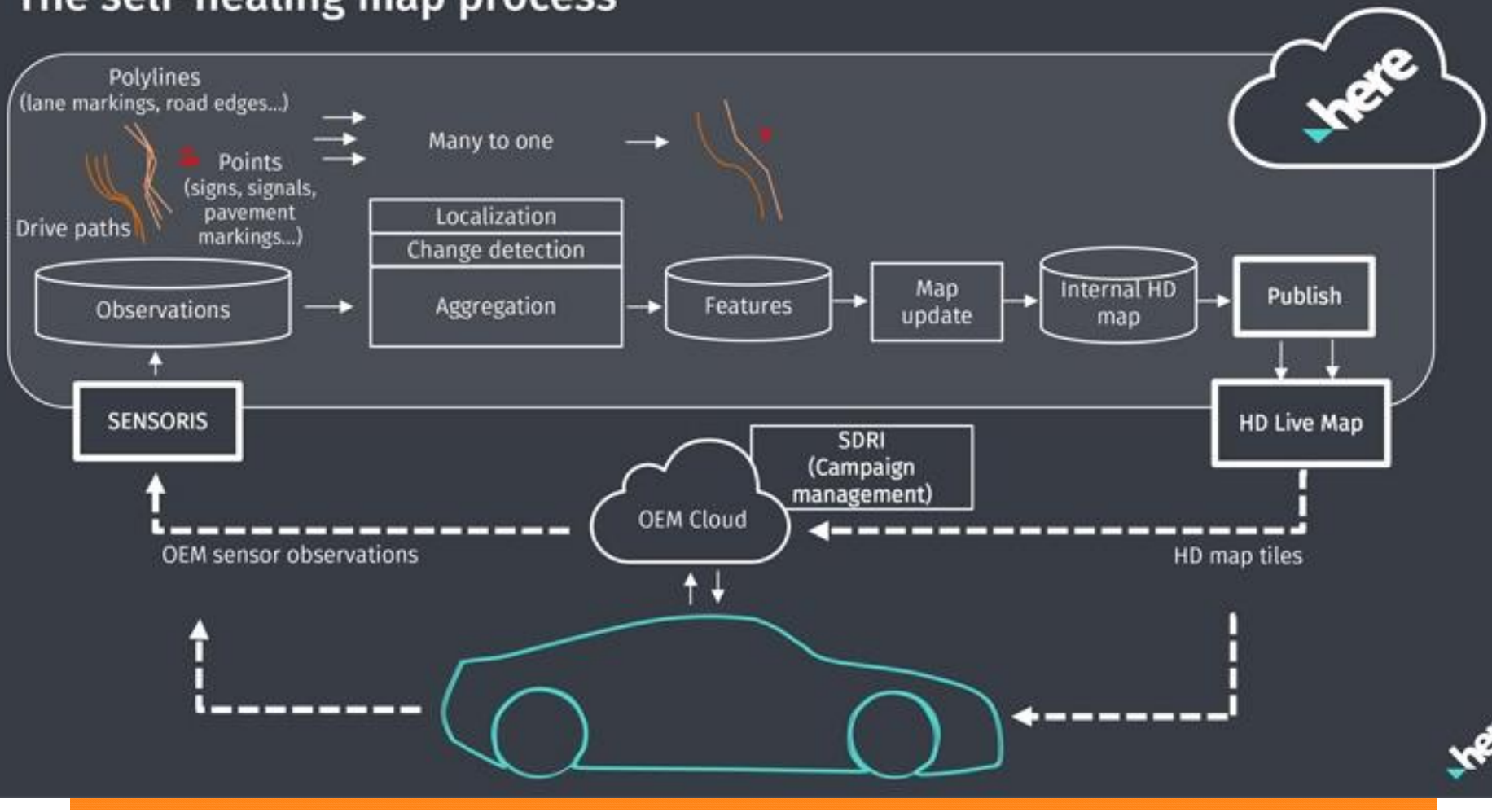
MILES AND COUNTING

Supportive infra: Self-healing maps

(HERE)



The self-healing map process



SELF-
DRIVING
EVERY-
WHERE

SENSIBLE 4

MADE IN FINLAND

Aalto University
School of Electrical
Engineering


Sensible 4

Amazon Prime

FOCUS 3

Drones – the last mile problem

Zipline

By Roksenhorn - Own work, CC BY-SA 4.0

ATLAS by Boston Dynamics

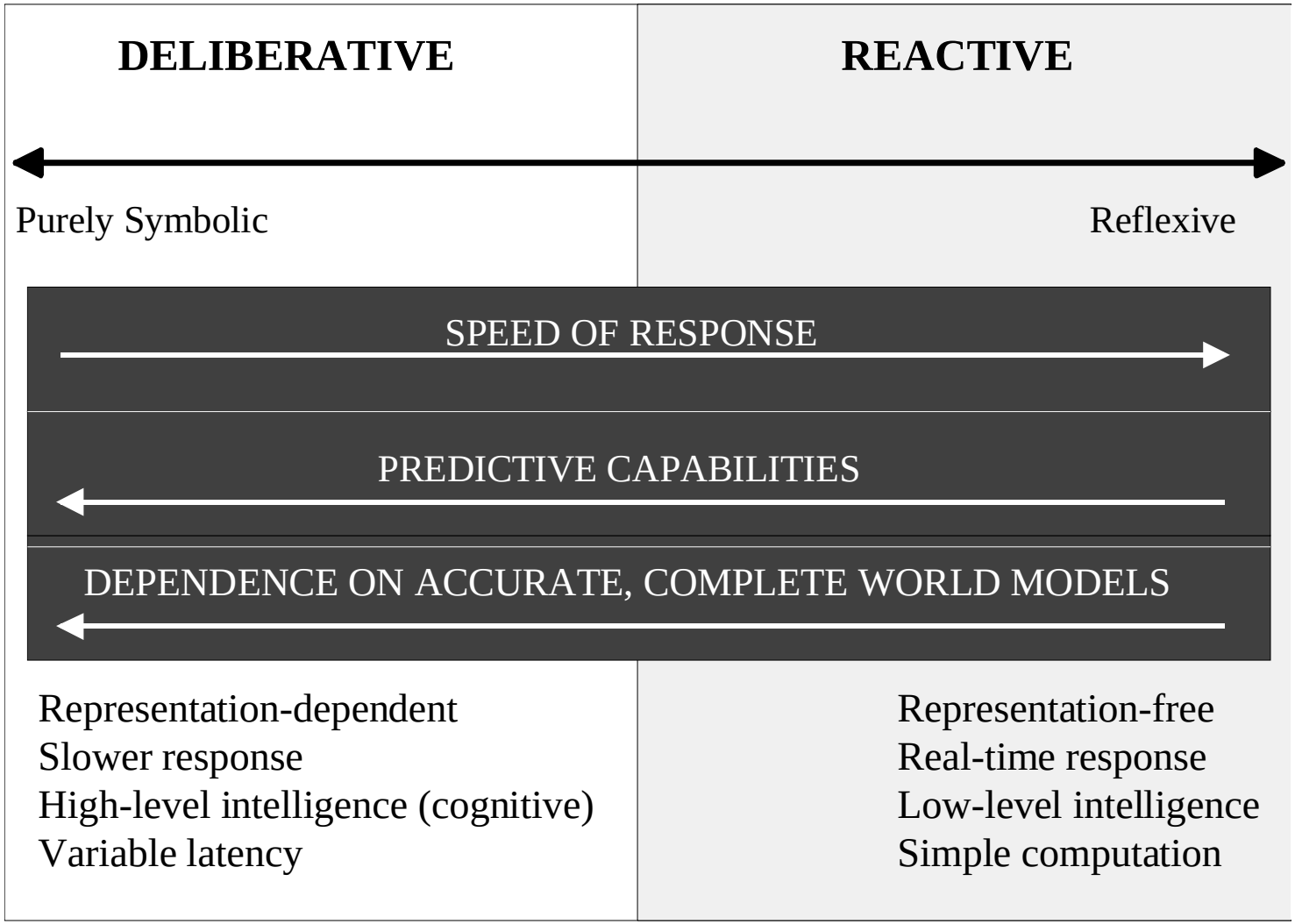
FOCUS 4
Humanoids

CONTROL ARCHITECTURES

THE REAL DEAL

Control

Robot control refers to the way in which the sensing and action of a robot are coordinated. The many different ways in which robots can be controlled all fall along a well-defined spectrum of control.


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

Architecture

- provides a principled way of organizing a control system. However, in addition to providing structure, it imposes constraints on the way the control problem can be solved [Mataric]

- describes a set of architectural components and how they interact [Dean & Wellman]

Evaluating an Architecture

- ***support for modularity*:** does it show good software engineering principles?
- ***niche targetability*:** how well does it work for the intended application?
- ***ease of portability to other domains*:** how well would it work for other applications or other robots?
- ***robustness***: where is the system vulnerable, and how does it try to reduce that vulnerability?

Control Approaches:

- Reactive Control : Don’t think, (re)act.
- Deliberative Control : Think hard, act later.
- Hybrid Control : Think and act independently, in parallel.

Control Trade-offs:

- Thinking is slow.
- Reaction must be fast.
- Thinking enables looking head (planning) to avoid bad solutions.
- Thinking too long can be dangerous (e.g., falling off a cliff, being run over).
- To think, the robot needs (a lot of) accurate information => world models.

Reactive Systems:

- Don’t think, react!
- Reactive control is a technique for tightly coupling perception (sensing) and action, to produce timely robotic response in dynamic and unstructured worlds.
- Think of it as "stimulus-response".
- A powerful method: many animals are largely reactive.

Limitations:

- Minimal (if any) state.
- No memory.
- No learning.
- No internal models / representations of the world.

Deliberative Systems

- Based on the sense->plan->act model
- Inherently sequential
- Planning requires search, which is slow
- Search requires a world model
- World models become outdated
- Search and planning takes too long

Hybrid Systems

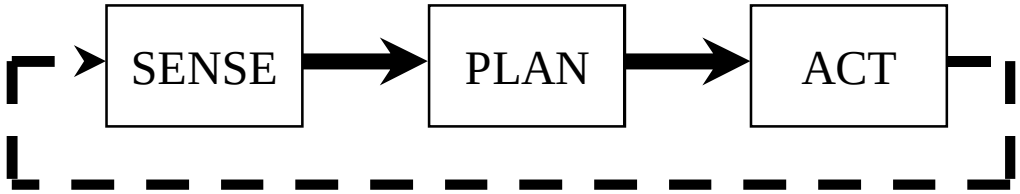
- Combine the two extremes
  - reactive system on the bottom
  - deliberative system on the top
  - connected by some intermediate layer
- Often called 3-layer systems
- Layers must operate concurrently
- Different representations and time-scales between the layers
- The best of both worlds?

The Hierarchical (aka deliberative) Paradigm

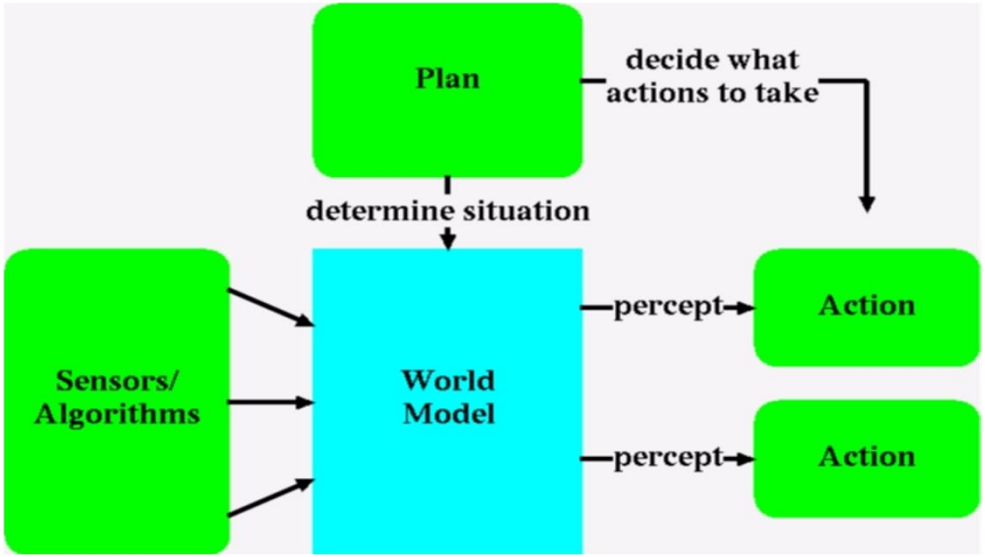
Hierarchical Paradigm…

- Top-down:
  plan, plan, plan
- Control-theoretic:
  must measure error in order to control device
- Planning means:
  dependence on world models

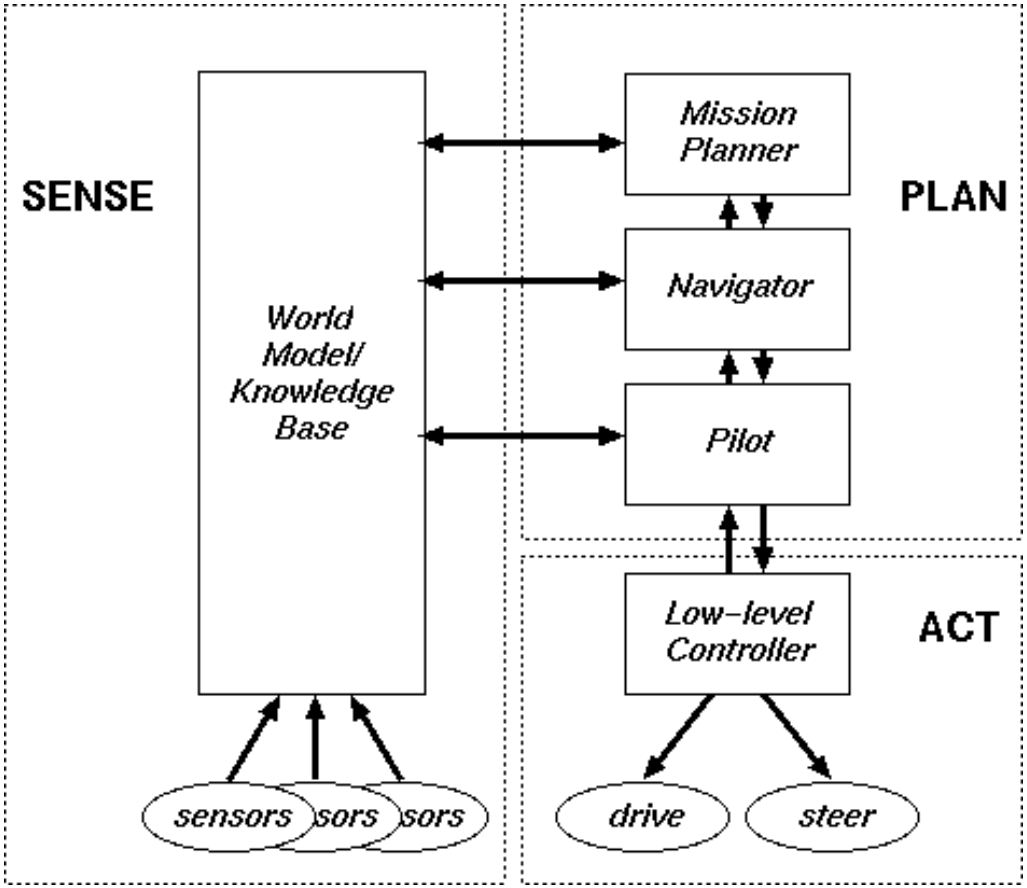
Organization



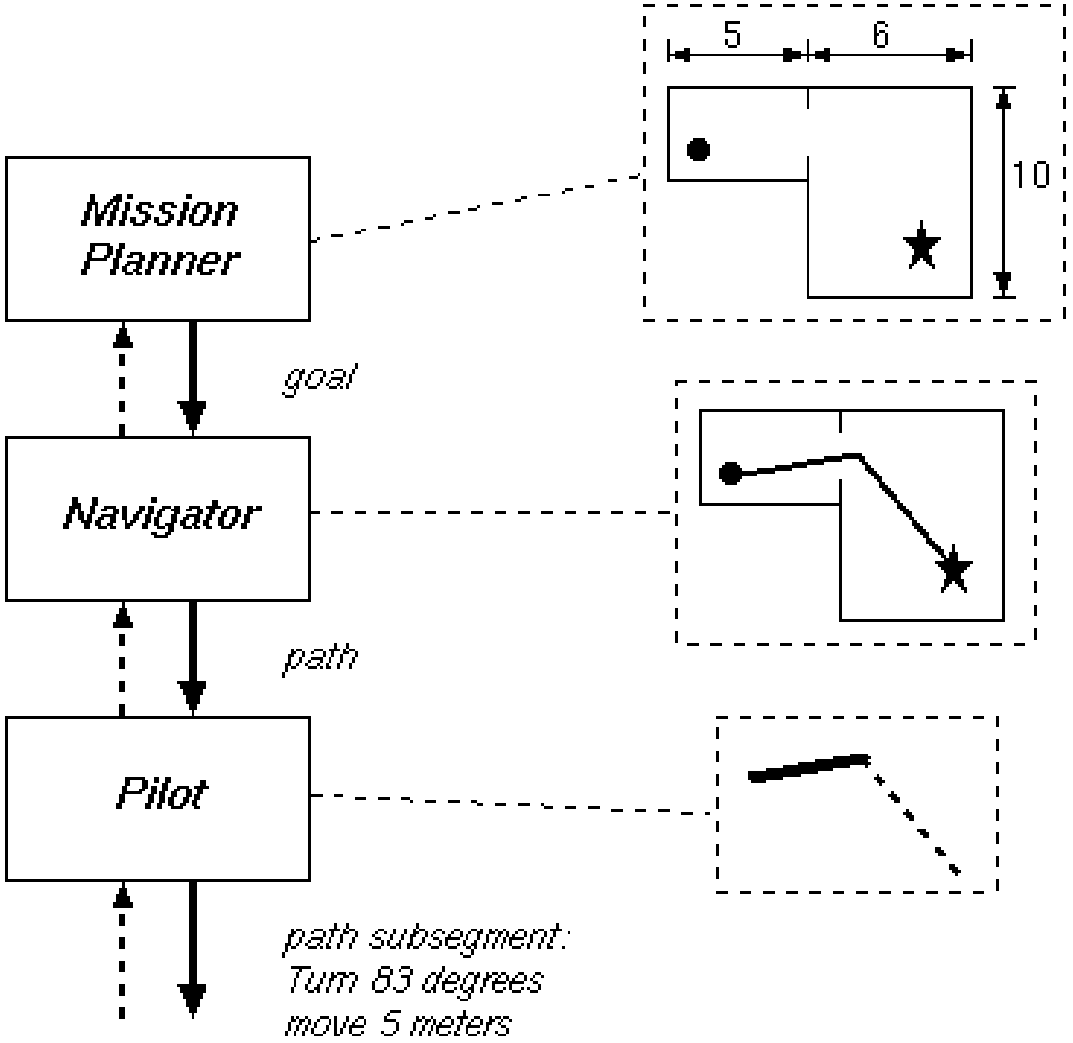
World model:
1. A priori rep
2. Sensed info
3. Cognitive



Nested Hierarchical Controller (Meystel)



NHC Planner


Mission Planner
goal
Navigator
path
Pilot
path subsegment:
Turn 83 degrees
move 5 meters
5
6
10

SPA = Planner-based

SPA has serious drawbacks

- Problem 1: Time-Scale
- Problem 2: Space
- Problem 3: Information
- Problem 4: Use of Plans

Problem 1: Time-Scale

- It takes a very (prohibitively) long time to search in a real robot's state space, as that space is typically very large
- Real robots may have collections of simple digital sensors (e.g., switches, IRs), a few more complex ones (e.g., cameras), or analog sensors (e.g., encoders, gauges, etc.) => "too much information" => Generating a plan is slow.

Problem 2: Space

- It takes a lot of space (memory) to represent and manipulate the robot's state space representation.
- The representation must contain all information needed for planning. => Generating a plan can be large.
- Space is not nearly as much of a problem as time, in practice.

Problem 3: Information

- The planner assumes that the representation of the state space is accurate and up-to-date => The representation must be constantly updated and checked
- The more information, the better => "too little information"

Problem 4: Use of Plans

The resulting plan is only useful if:

a) the environment does not change during the execution of a plan in a way that affects the plan

b) the representation was accurate enough to generate a correct plan

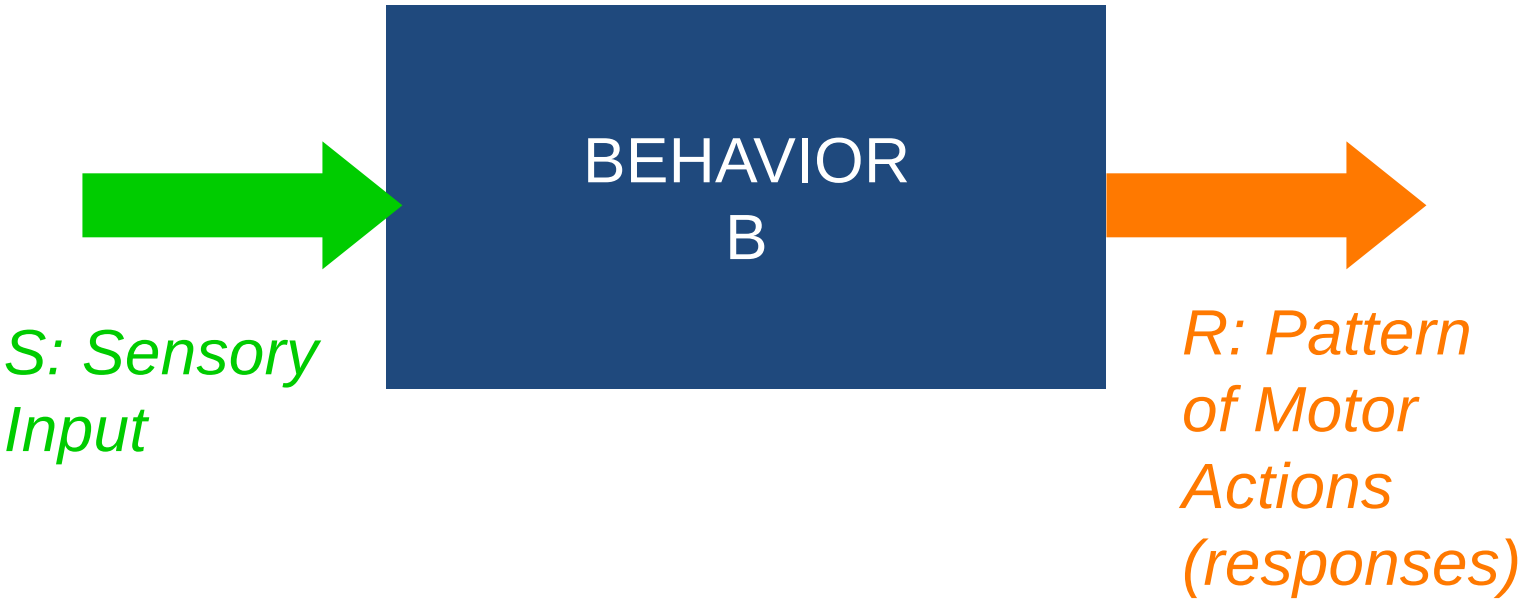
c) the robot's effectors are accurate enough to perfectly execute each step of the plan in order to make the next step possible.

Planners Live On in Robotics

- The SPA approach has not been abandoned, it has been expanded
- Given the two fundamental problems with purely deliberative approaches, we can augment them:
  - search/planning is slow, so save/cache important and/or urgent decisions;
  - be ready to respond or re-plan when the plan fails.

Biological Foundations of the Reactive Paradigm

Behavior Definition



A behavior is a mapping of sensory inputs to a pattern of motor actions which are then used to achieve a task.

Notation: B(S)=R

Types of Behaviors

- **Reflexive**
  - stimulus-response, often abbreviated S-R
- **Reactive**
  - learned or “muscle memory”
- **Conscious**
  - deliberately stringing together

WARNING Overloaded terms:
Roboticists often use “reactive behavior” to mean purely reflexive,
And refer to reactive behaviors as “skills”

Ethology: Coordination and Control of Behaviors

Nobel 1973 in
-von Frisch
-Lorenz

INNATE RELEASING MECHANISMS

www.nobel.se

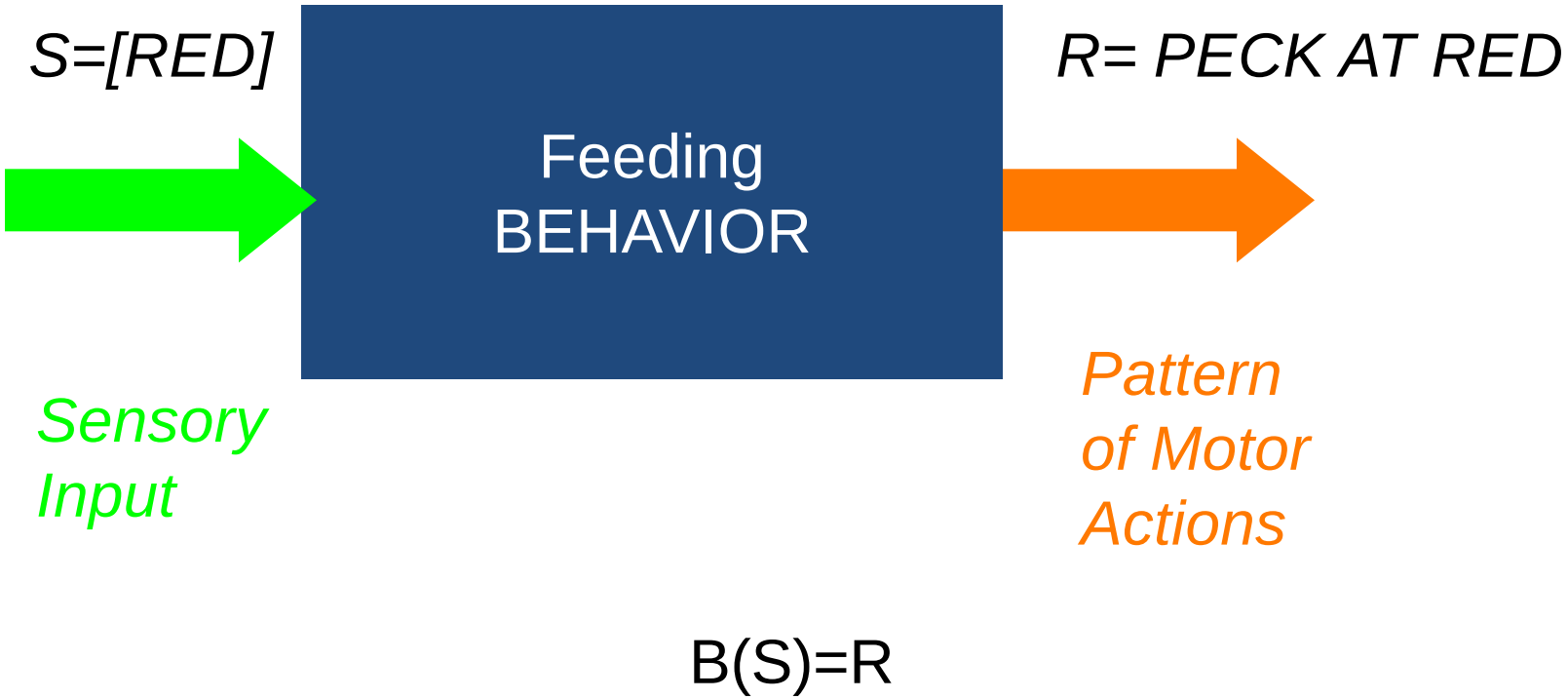
Motivating Example: Arctic Terns

- Arctic terns live in Arctic (black, white, gray environment, some grass) but adults have a red beak

- When hungry, baby pecks at parent's beak, who regurgitates food for baby to eat
- How does it know its parent?
  - It doesn't, it just goes for the largest red spot in its field of view (e.g., ethology grad student with construction paper)
  - **Only red thing should be an adult tern**
  - **Closer = large red**

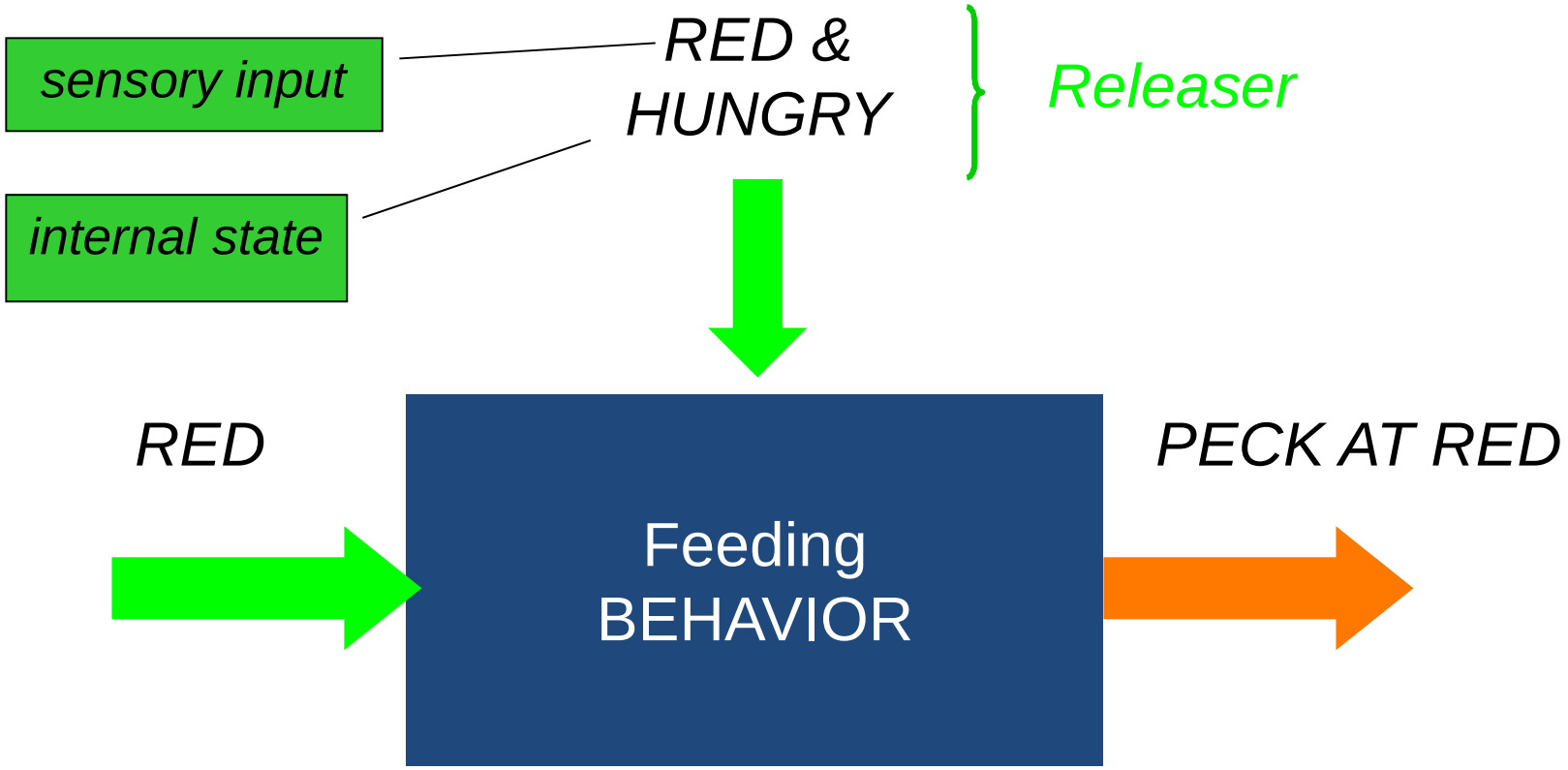
Behavior template

“the feeding behavior”



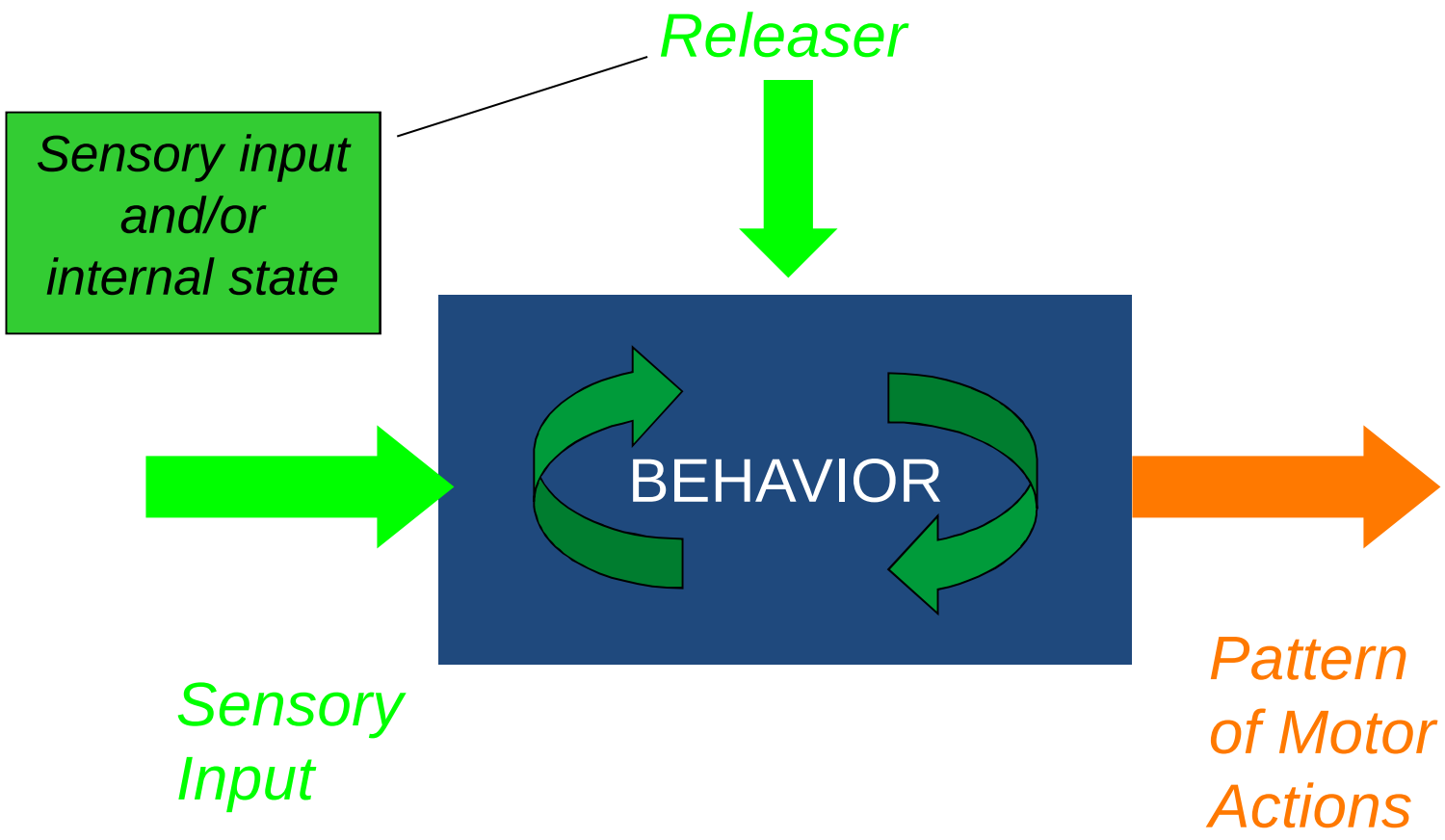
B(S)=R

“the feeding releaser”



$$S = [RED, HUNGRY], R = [PECK\text{-}AT(RED)], B = [b_{feeding}]$$

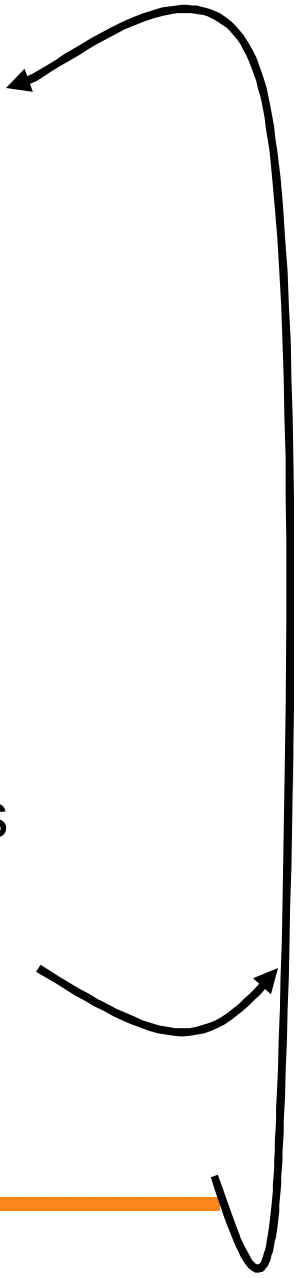
Innate Releasing Mechanisms



Example: Cockroach Hide

- light goes on, the cockroach turns and runs
- when it gets to a wall, it follows it
- when it finds a hiding place (thigmotrophic), goes in and faces outward
- waits, then comes out
- *even if the lights are turned back off earlier*

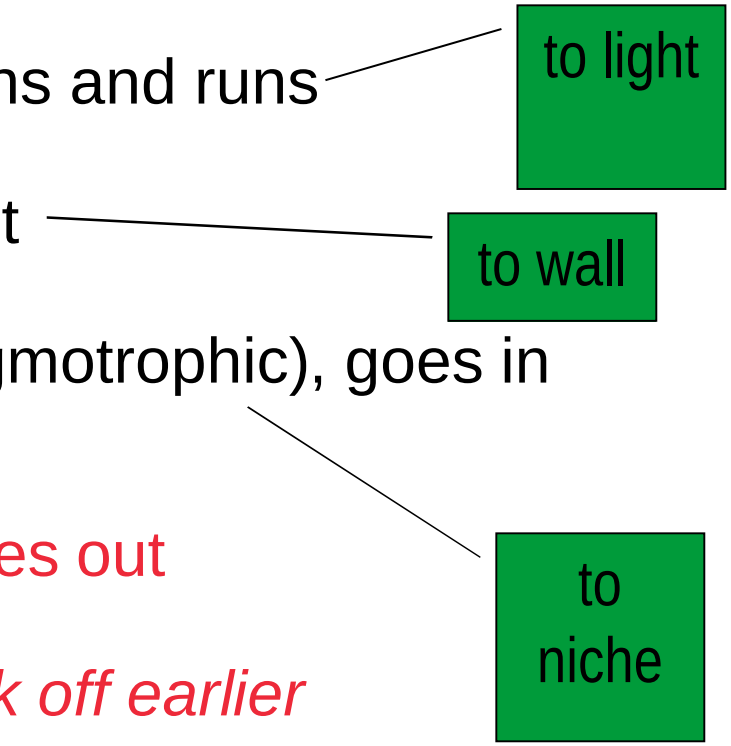
Observation 1: Fixed Pattern Action



- light goes on, the cockroach turns and runs
- when it gets to a wall, it follows it
- when it finds a hiding place, goes in and faces outward
- waits, then comes out
- *even if the lights are turned back off earlier*

Observation 2: Exhibits Taxis

- light goes on, the cockroach turns and runs
- when it gets to a wall, it follows it
- when it finds a hiding place (thigmotrophic), goes in and faces outward
- waits until not scared, then comes out
- *even if the lights are turned back off earlier*



Break into Behaviors

Flee

- b_{flee} light goes on, the cockroach turns and runs

Follow-wall

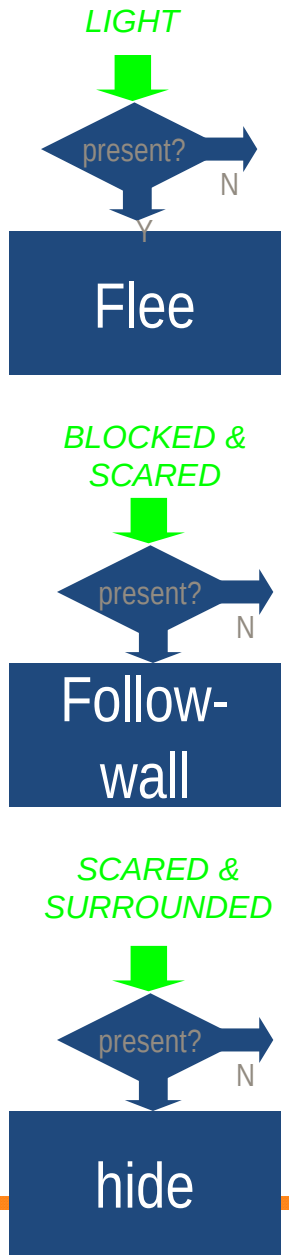
- $b_{follow\text{-}wall}$ when it gets to a wall, it follows it on right

hide

- b_{hide} when it finds a hiding place, goes in and faces outward
- waits until not scared, then comes out

Cockroach Hiding: the behaviors

$$B = \begin{bmatrix} b_{flee} \\ b_{follow\text{-}wall} \\ b_{hide} \end{bmatrix}$$



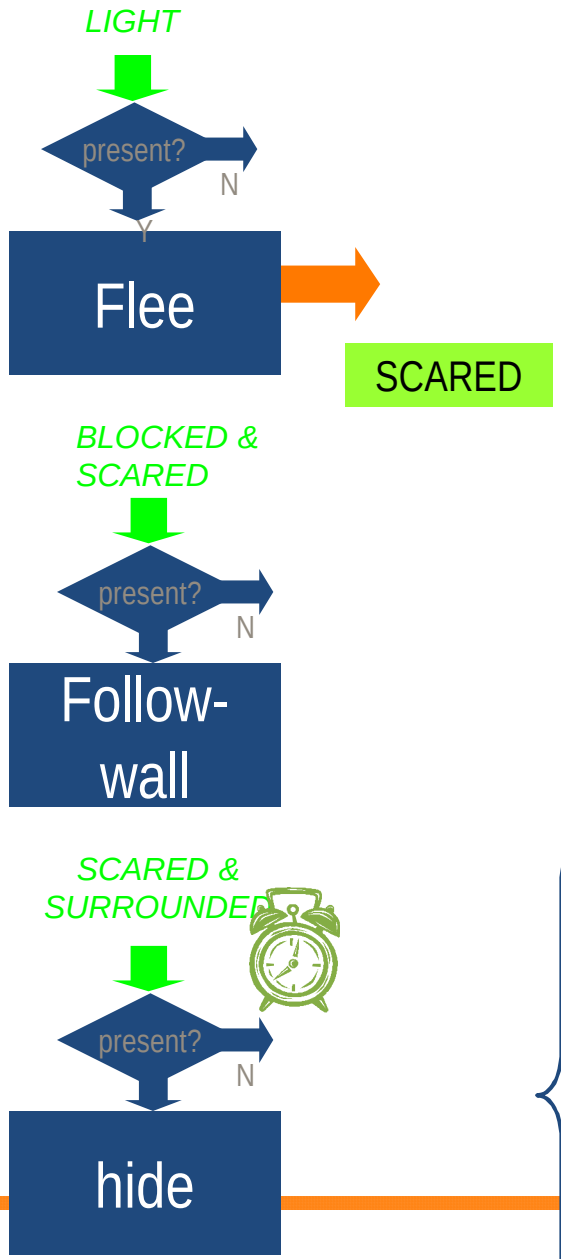
Find Releasers

- **light goes on**, the cockroach turns and runs

Ooops, need internal state: Scared

ollows

- **when it finds a hiding place**, goes in and faces outward
- **waits until not scared**, then comes out



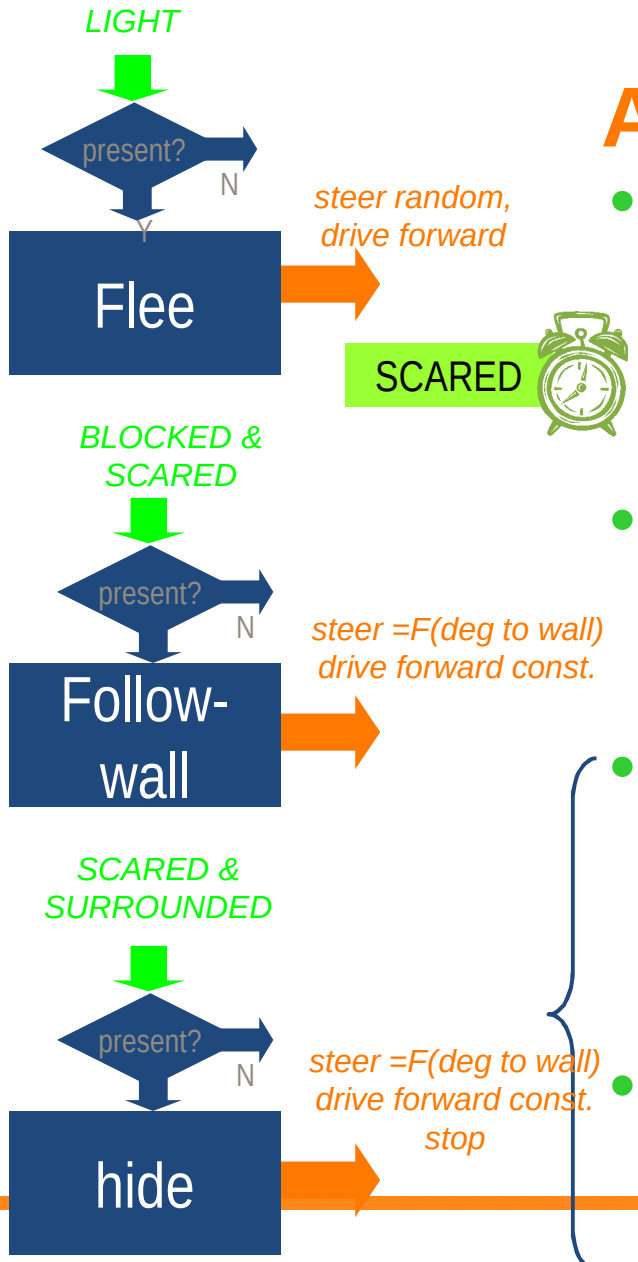
Internal State Set

- **light goes on**, the cockroach is **scared** and turns and runs
- **when it gets to a wall**, it follows it on right
- **when it finds a hiding place**, goes in and faces outward
- **waits until not scared**, then comes out

Cockroach Hiding

$$B = \begin{bmatrix} b_{flee} \\ b_{follow\text{-}wall} \\ b_{hide} \end{bmatrix}$$

$$S = \begin{bmatrix} LIGHT \\ (BLOCKED\text{-}ON\text{-}RIGHT, l_{angle}) \,\&\, SCARED) \\ (SURROUNDED \,\&\, SCARED) \end{bmatrix}$$



Action (Responses)

- **light goes on**, the cockroach gets scared, turns and runs
- **when it gets to a wall**, it follows it
- **when it finds a hiding place**, goes in and faces outward
- **waits until not scared**, then comes out

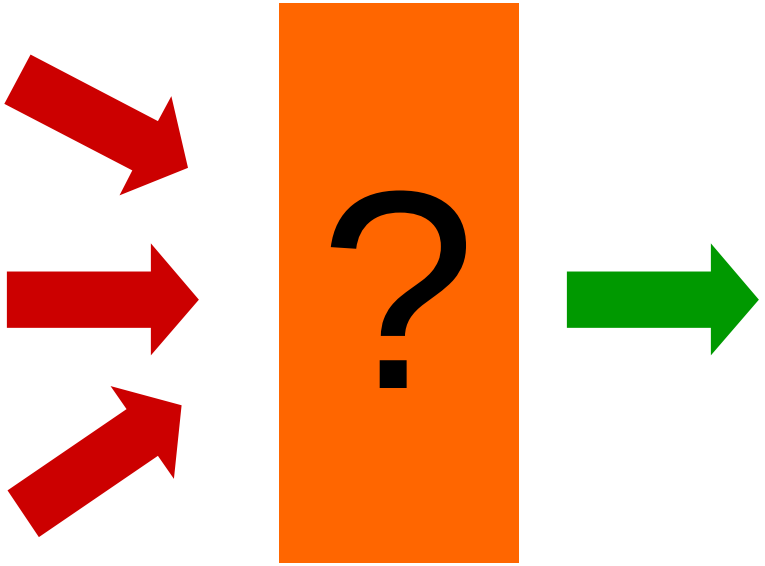
Cockroach Hiding

$$B = \begin{bmatrix} b_{flee} \\ b_{follow\text{-}wall} \\ b_{hide} \end{bmatrix} \qquad S = \begin{bmatrix} LIGHT \\ (BLOCKED\text{-}ON\text{-}RIGHT, l_{angle}) \,\&\, SCARED) \\ (SURROUNDED \,\&\, SCARED) \end{bmatrix}$$

$$R = \begin{bmatrix} r(steer = random, drive = f(SCARED)) \\ r(steer = f(\%\, blocked), drive = f(SCARED)) \\ r(steer = f(\%\, surrounded), drive = slow) \end{bmatrix}$$

$$R = \begin{bmatrix} r_{flee} \\ r_{follow\text{-}wall} \\ r_{hide} \end{bmatrix}$$

What happens when there's a conflict from concurrent behaviors?



$$r = C(G * R)$$

You need arbitration

- If the rules are not triggered by mutually-exclusive conditions, more than one rule can be triggered in parallel, resulting in two or more different actions being output by the system.
- Deciding among multiple actions or behaviors is called arbitration, and is in general a difficult problem.

Arbitration can be done based on:

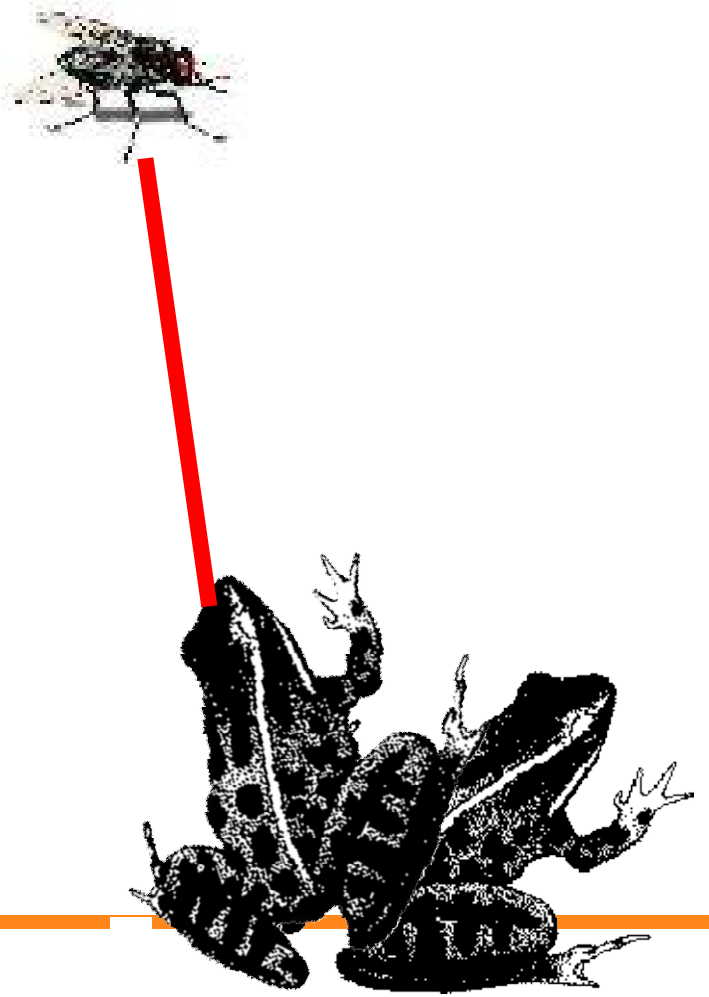
- a fixed priority hierarchy (processes have pre-assigned priorities)
- a dynamic hierarchy (process priorities change at run-time)
- learning (process priorities may be initialized or not, and are learned at run-time, once or repeatedly/dynamically)

Schema Theory

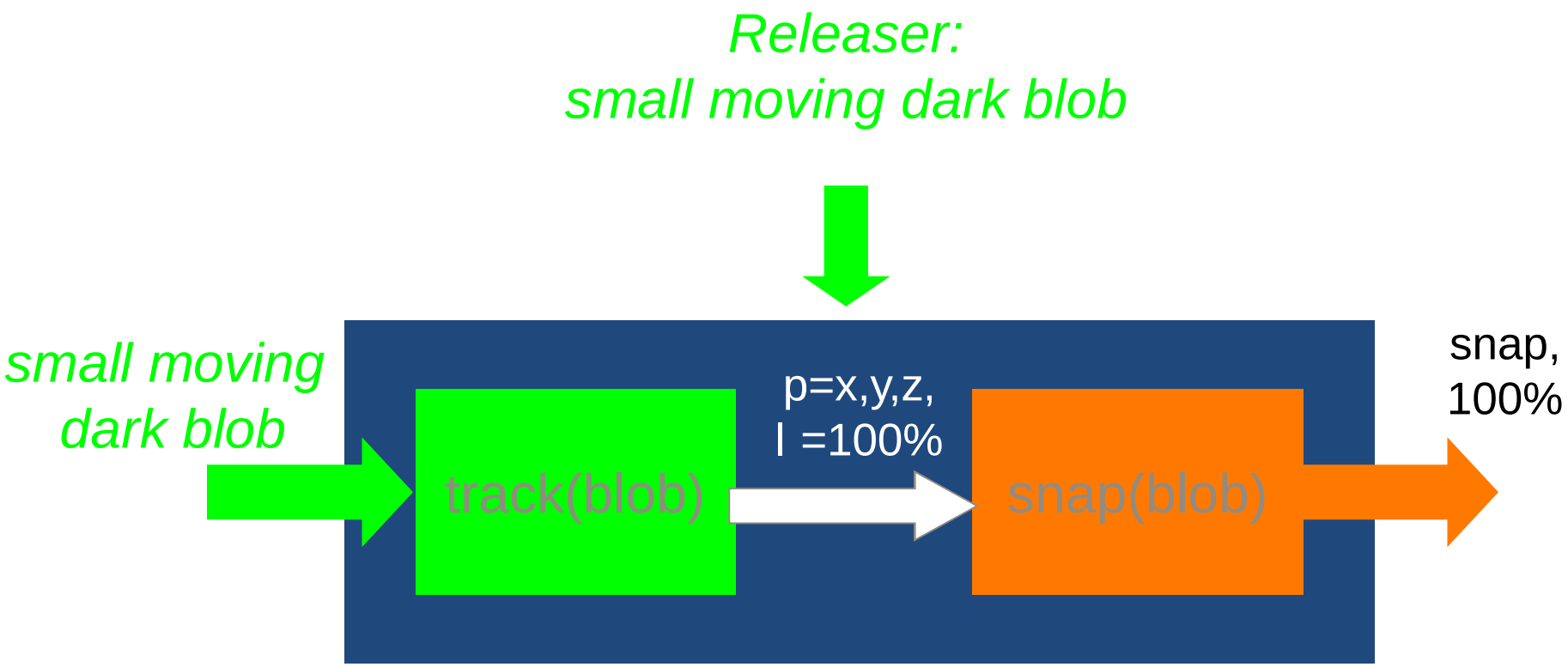
SCHEMA is used in cognitive science and ethology to refer to a particular organized way of perceiving cognitively and responding to a complex situation or set of stimuli

- a behavior is a schema, consists of
  - perceptual schema
  - motor schema
  - other behaviors

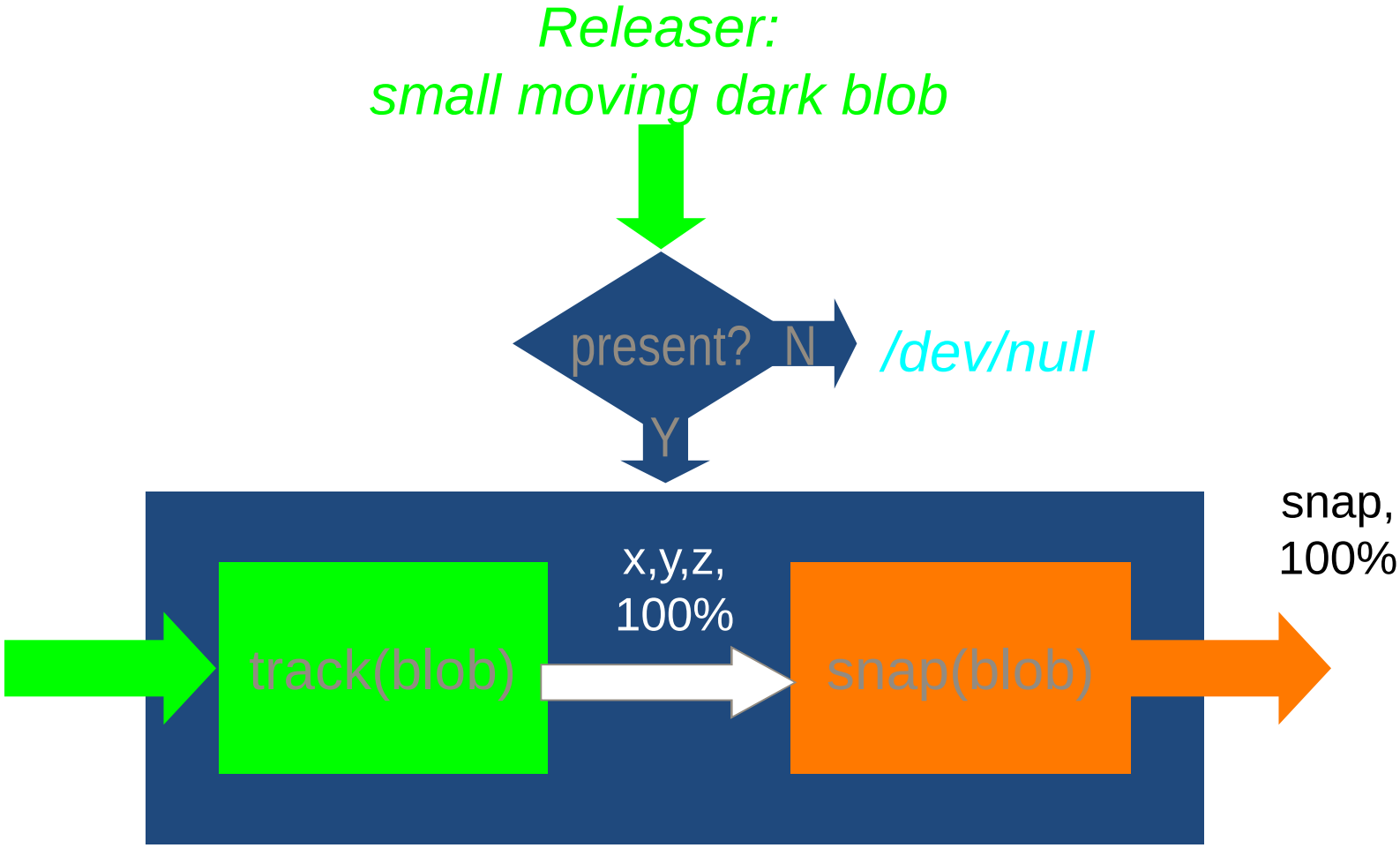
Fly Snapping



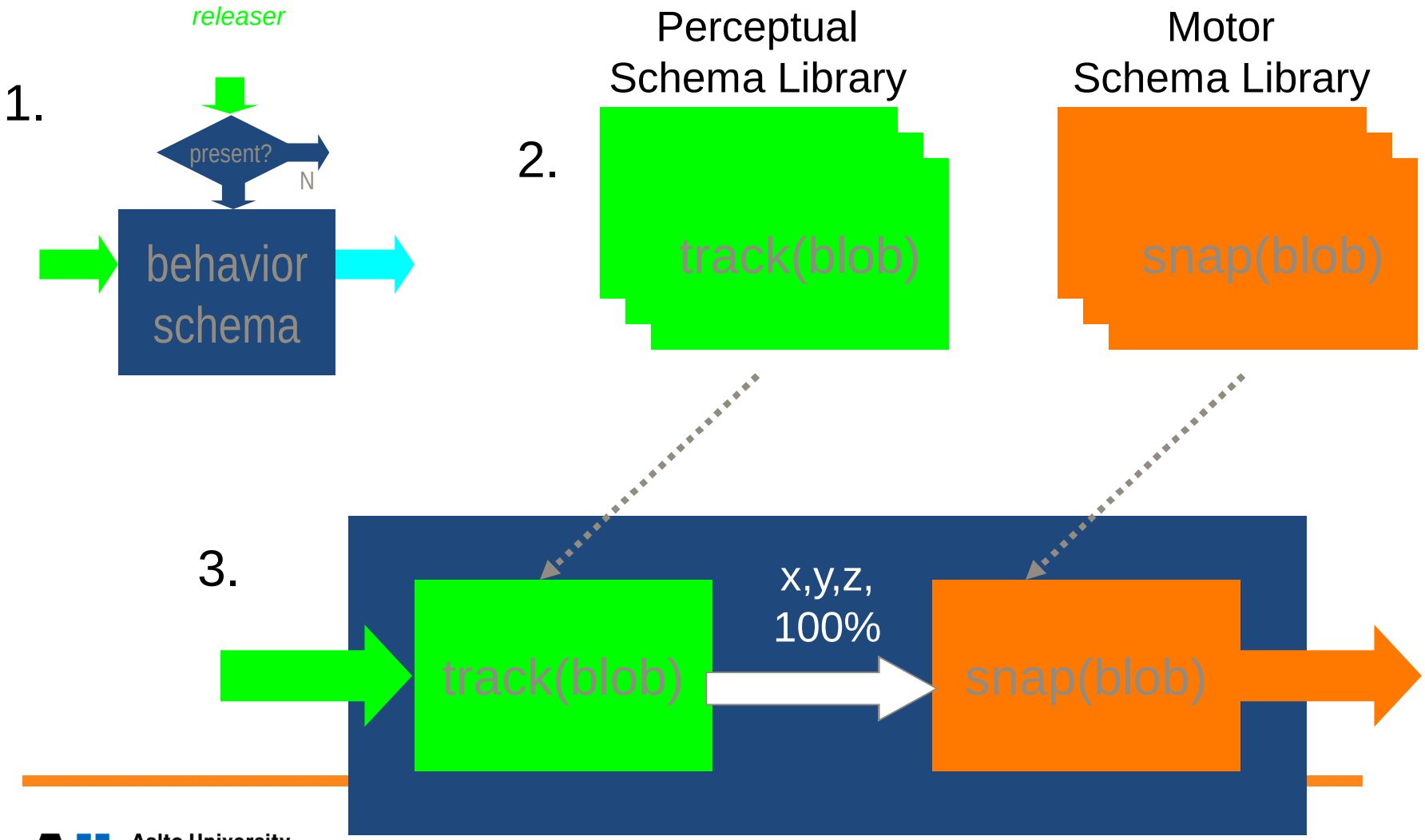
Ex. Fly Snapping Behavior IRM



Schema Instantiation (SI)



Schema/Schema Instantiation

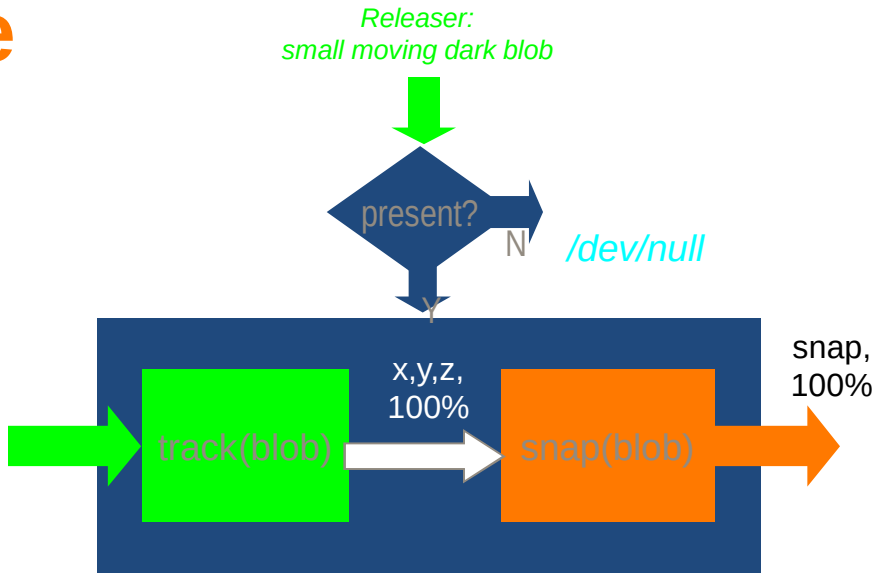


Advantages

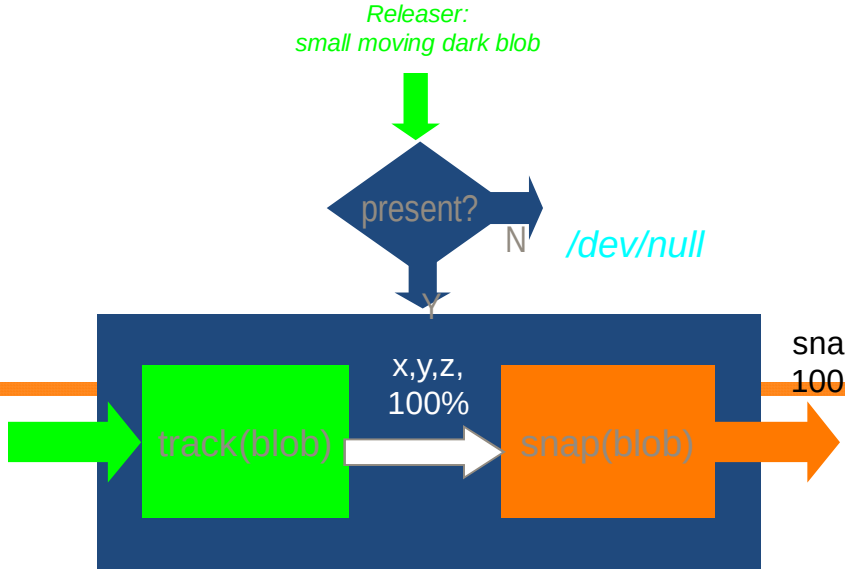
- modular
- can assemble new behaviors from existing schemas
  - learning by experimentation
- can substitute alternatives
  - reroute nerves

Back to Toads and Frogs: Instantiation for each eye

Left eye



Right eye



Snap at vector sum (middle)

General Principles

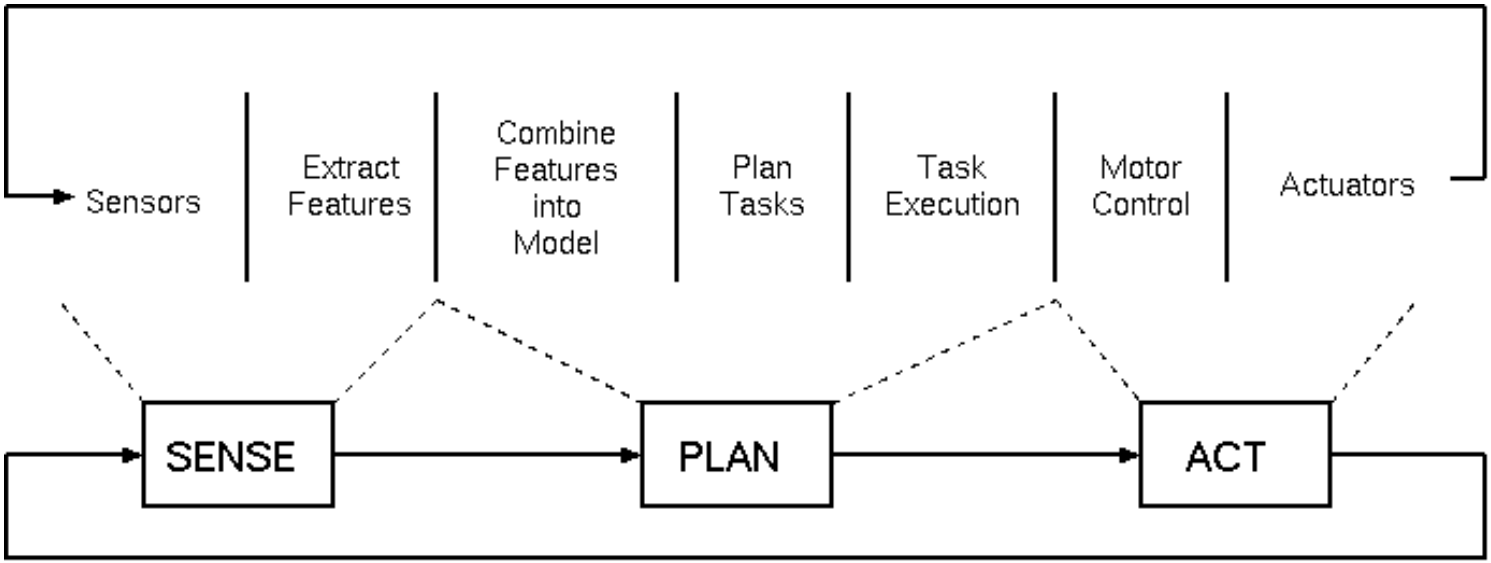
- All animals possess a set of behaviors
- Releasers for these behaviors rely on both internal state and external stimulus
- Perception is filtered; perceive what is relevant to the task

The Reactive Paradigm

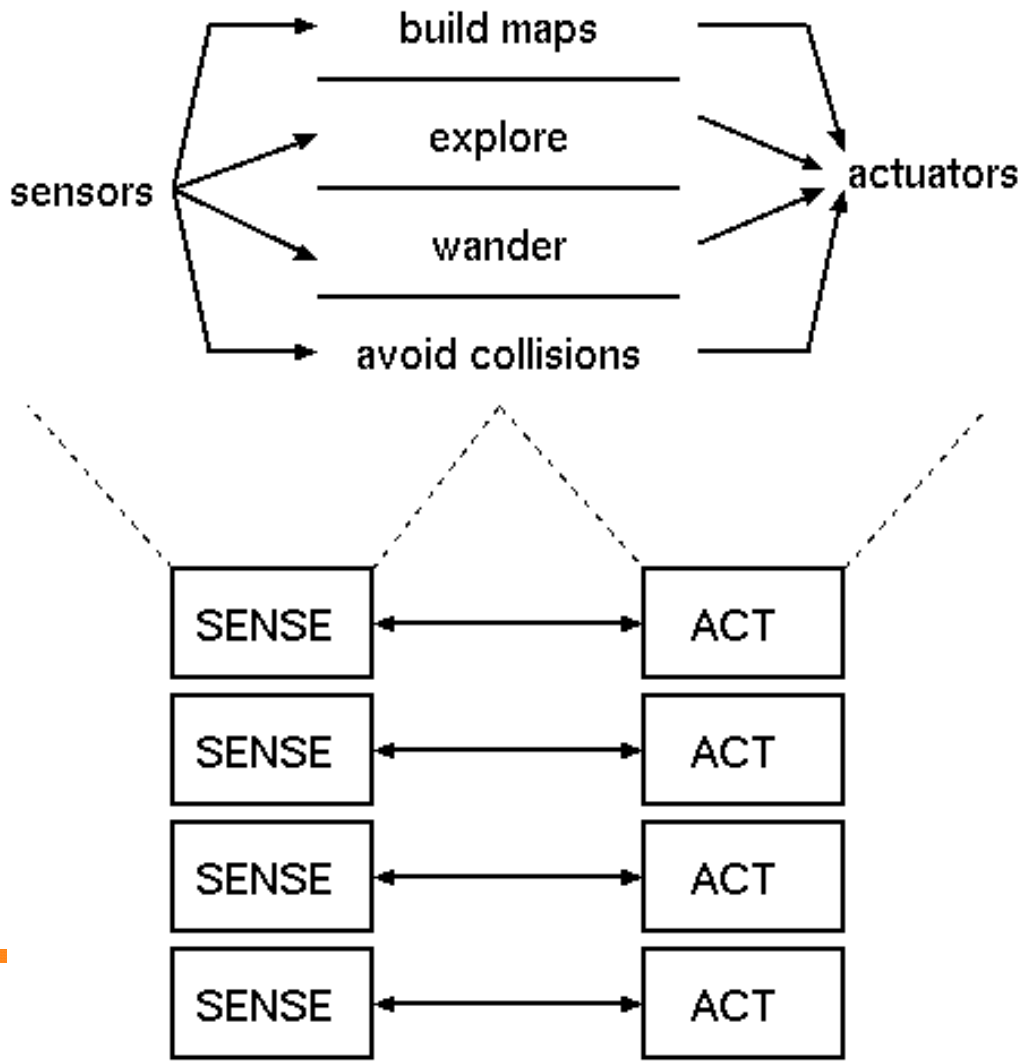
Review: Lessons from Biology

- Programs should decompose complex actions into behaviors. Complexity emerges from concurrent behaviors acting independently
- Agents should rely on straightforward activation mechanisms such as IRM
- Perception filters sensing and considers only what is relevant to the task (action-oriented perception)
- Behaviors are independent but the output may be used in many ways including: combined with others to produce a resultant output or to inhibit others

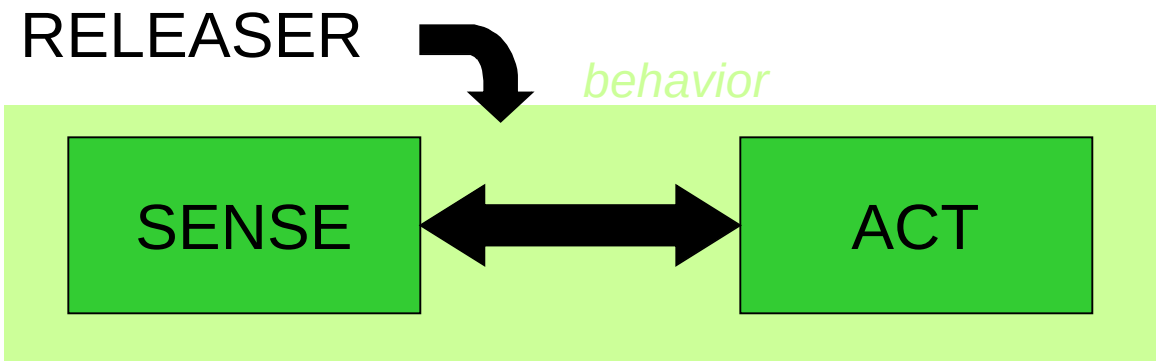
Hierarchical Organization is "Horizontal"



More Biological is “Vertical”



Reactive Robots



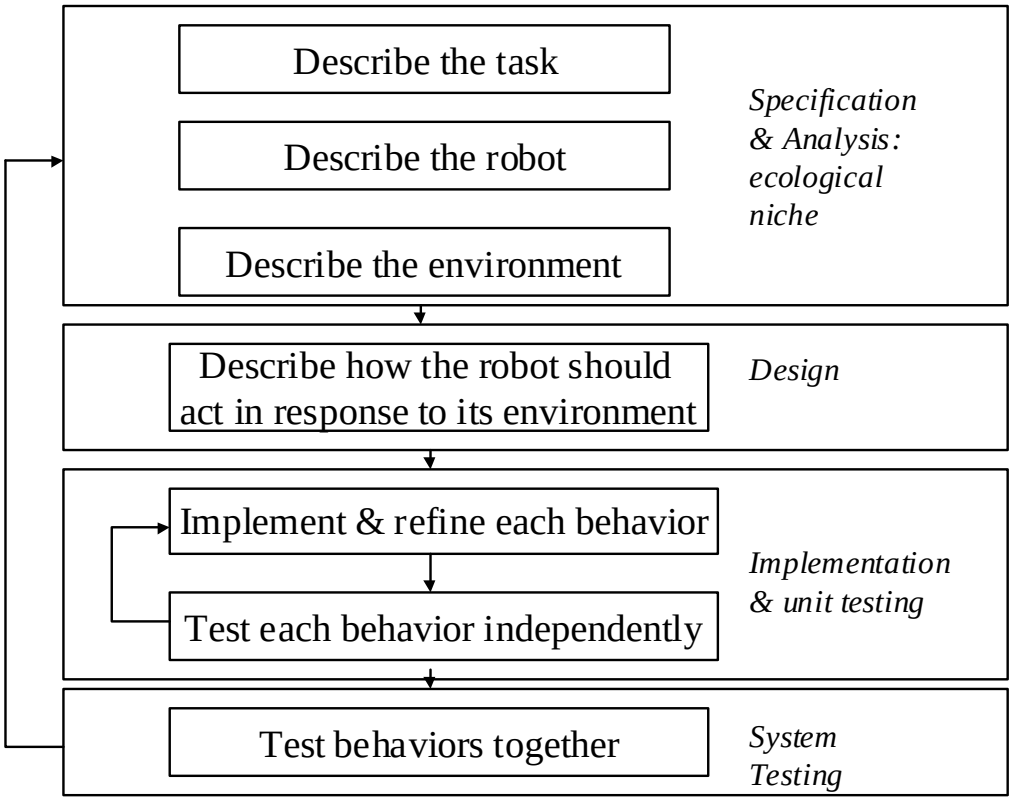
- Most apps are programmed with this paradigm
- Biologically based:
  - Behaviors (independent processes), released by perceptual or internal events (state)
  - No world models or long term memory
  - Highly modular, generic
  - Overall behavior *emerges*

Example 1: Robomow

- Behaviors?
- Random
- Avoid
  - Avoid(bump=obstacle)
  - Avoid(wire=boundary)
- Stop
  - Stop(tilt=ON)
- All active

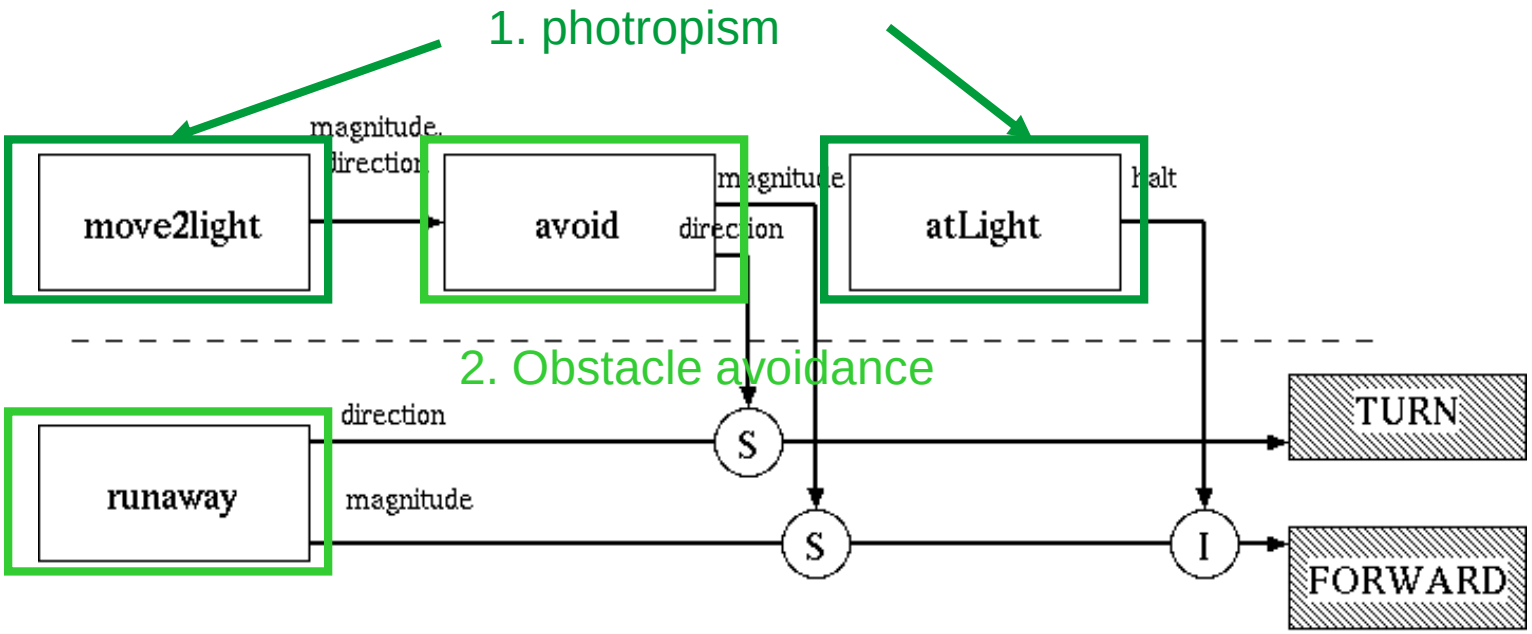
www.friendlymachines.com

Steps in Designing a Reactive Behavioral System



Design Tool: Behavior Table

An agent is attracted to light. If it sees light, it heads in that direction. If it encounters an obstacle, it turns either left or right, favoring the direction of the light. If there is no light, it sits and waits. But if an obstacle appears, the agent runs away.



notice: unlike examples in book, robot doesn't have to go a fixed speed

Behavior Table

An agent is attracted to light. If it sees light, it heads in that direction. If it encounters an obstacle, it turns either left or right, favoring the direction of the light. If there is no light, it sits and waits. But if an obstacle appears, the agent runs away.

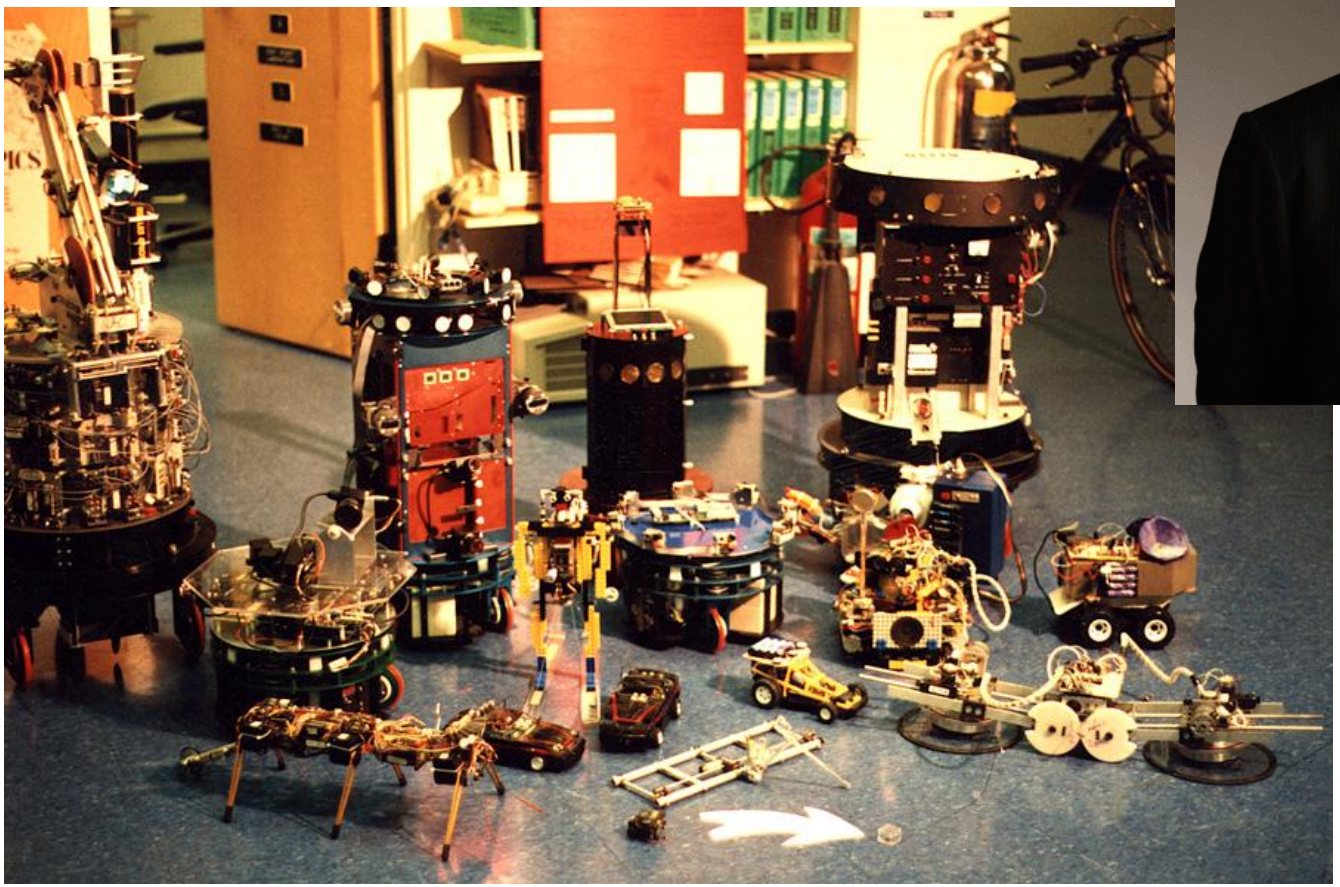
Releaser	**Behavior**	**Motor Schema**	**Percept**	**Perceptual Schema**
Light	**phototropism**	move2Light() :Attraction	Light: direction & strength	Brightest(dir), atLight()
Range <tasked level>	**Obstacle avoidance**	avoid(): turn left or right; runaway()	proximity	Obstacle()

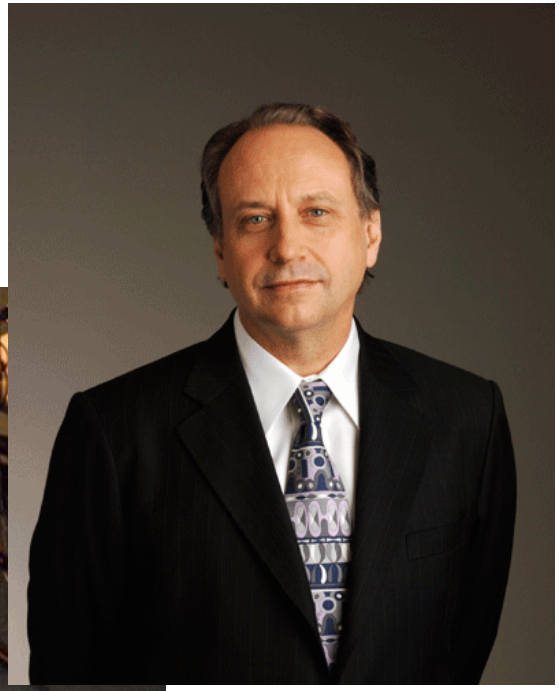
Reactive: 2 main styles

- Historically, there are two main styles of creating a reactive system
  - Subsumption architecture
    - Layers of behavioral competence
    - How to control relationships
  - Potential fields
    - Concurrent behaviors
    - How to navigate

- They are equivalent in power

Subsumption: Rodney Brooks

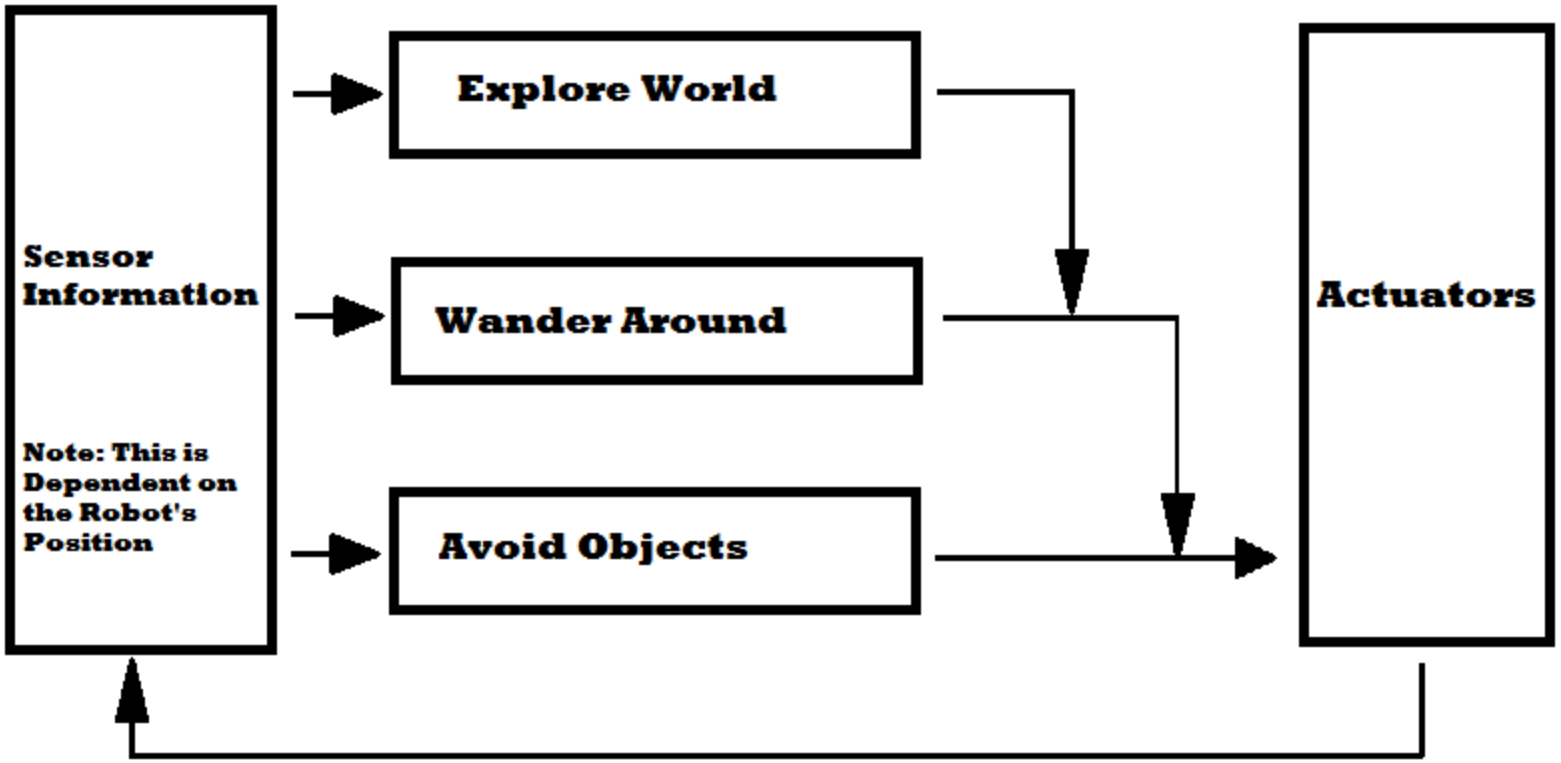




Panasonic Professor of Robotics (emeritus) at MIT.

https://people.csail.mit.edu/brooks/

The Subsumption Architecture (Brooks 1985)



The Subsumption Architecture

- systems are built from the bottom up
- components are task-achieving actions/behaviors (not functional modules)
- components can be executed in parallel
- components are organized in layers, from the bottom up
- lowest layers handle most basic tasks
- newly added components and layers exploit the existing ones
- each component provides and does not disrupt a tight coupling between sensing and action
- there is no need for internal models: "the world is its own best model"

- Subsumption systems grow from the bottom up, and layers can keep being added, depending on the tasks of the robot.
- How exactly layers are split up depends on the specifics of the robot, the environment, and the task.
- There is no strict recipe, but some solutions are better than others, and most are derived empirically.

- The inspiration behind the Subsumption Architecture is the evolutionary process, which introduces new competencies based on the existing ones.
- Complete creatures are not thrown out and new ones created from scratch; instead, solid, useful substrates are used to build up to more complex capabilities.

- The original Subsumption Architecture was implemented using a language based on finite state machines (FSMs) augmented with a very small amount of state (AFSMs), themselves implemented in Lisp.
- An AFSM can be in one state at a time, can receive one or more inputs, and send one or more outputs AFSMs are connected by communication wires, which pass input and output messages between them.

Situated Automata

- A formal notion of finite state machines whose inputs are connected to sensors and whose outputs are connected to effectors are called situated automata.
- Situated means existing in and interacting with a complex world, and automata is the formal name for FSMs (formally: finite state automata).
- Situated automata are used to create reactive principled control systems.

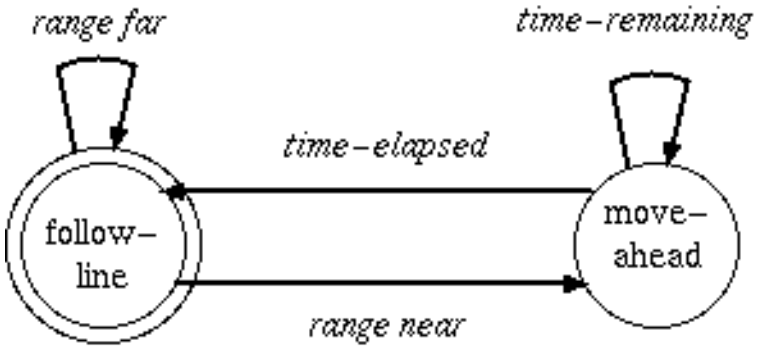
FSA Diagrams

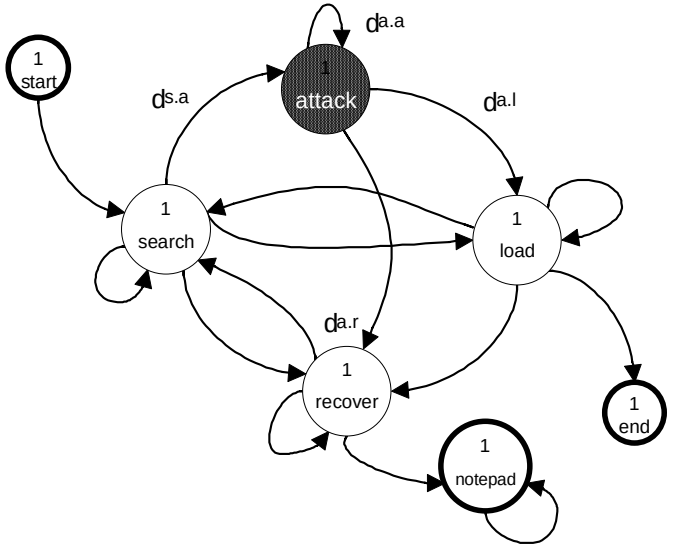
- States and state transitions are most easily encoded in finite state automata and drawn as finite state diagrams
- The states of the diagrams can also be behaviors, so the diagrams show sequences of behavior transitions
- These are called finite state acceptors
- Acceptor M is a quadruple (Q,d,q0,F)

Finite State Acceptors

- M (Q, d,q0,F):
  - Q is the set of legal behavioral states
  - d is a transition function from a state and an input to the next state (can be represented as a table)
  - q0 is the starting behavioral configuration
  - F is the set of accepting states (a subset of Q, indicates completion)

FSA: M={Q,d,q_0,F}

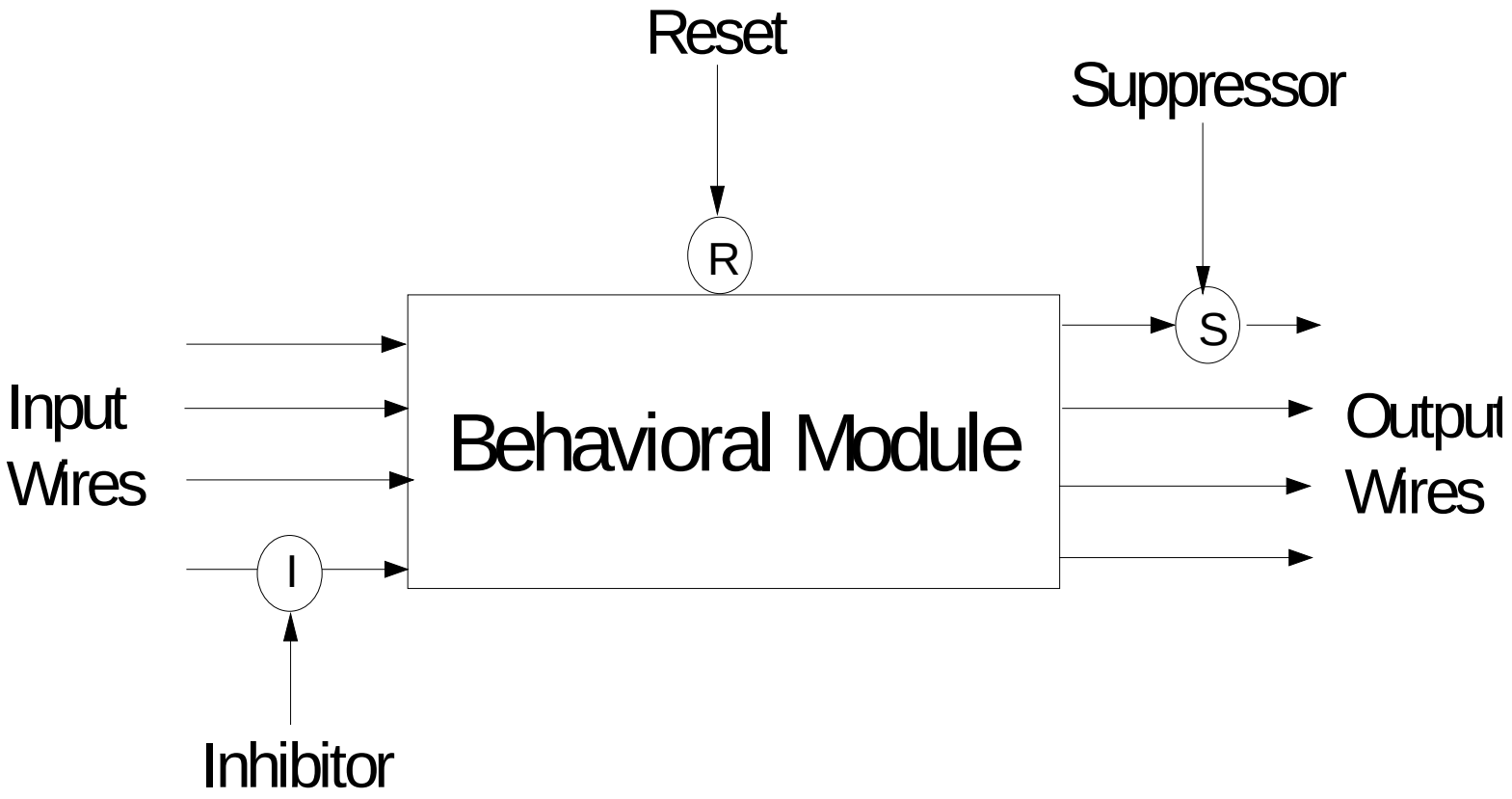




- Q: all the states, each is “q”- behaviors
- d: transition function, d(q,s)= new behavior
- q_0: Start state(s)- part of Q
- F: Terminating state(s)- part of Q

Arbitration in Subsumption

- Arbitration: deciding who has control
- Inhibition: prevents output signals from reaching effectors
- Suppression: replaces input signal with the suppressing message
- The above two are the only mechanisms for coordination => Results in priority-based arbitration, the rule or layer with higher priority takes over, i.e., has control of the AFSM

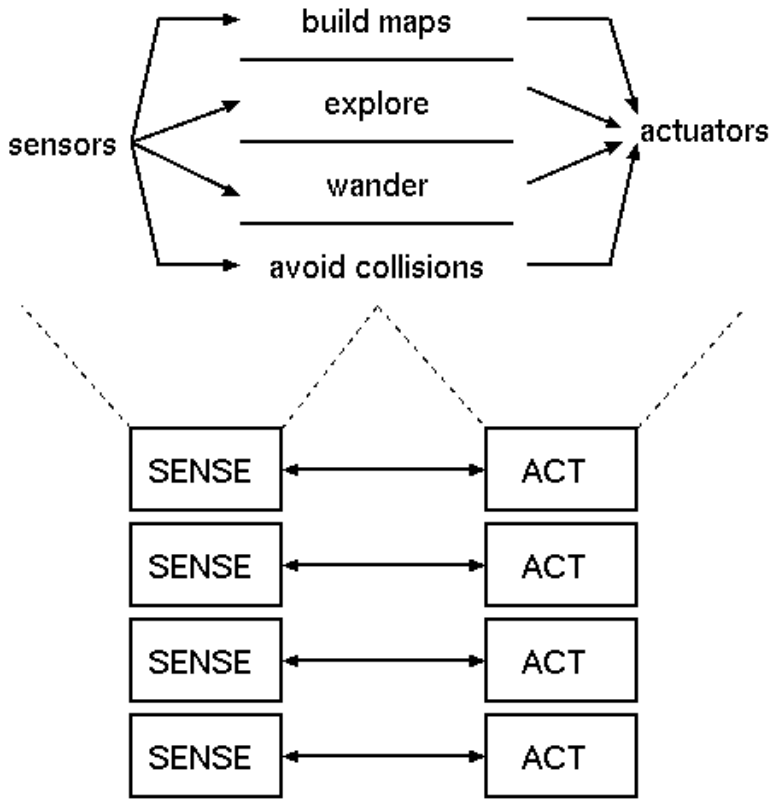

Reset
Suppressor
R
S
Input
Wires
Behavioral Module
Output
Wires
I
Inhibitor

Designing in Subsumption

- Qualitatively specify the overall behavior needed for the task
- Decompose that into specific and independent behaviors (layers)
- The layers should be bottom-up and consisting of disjoint actions
- Ground low-level behaviors in the robot's sensors and effectors
- Incrementally build, test, and add

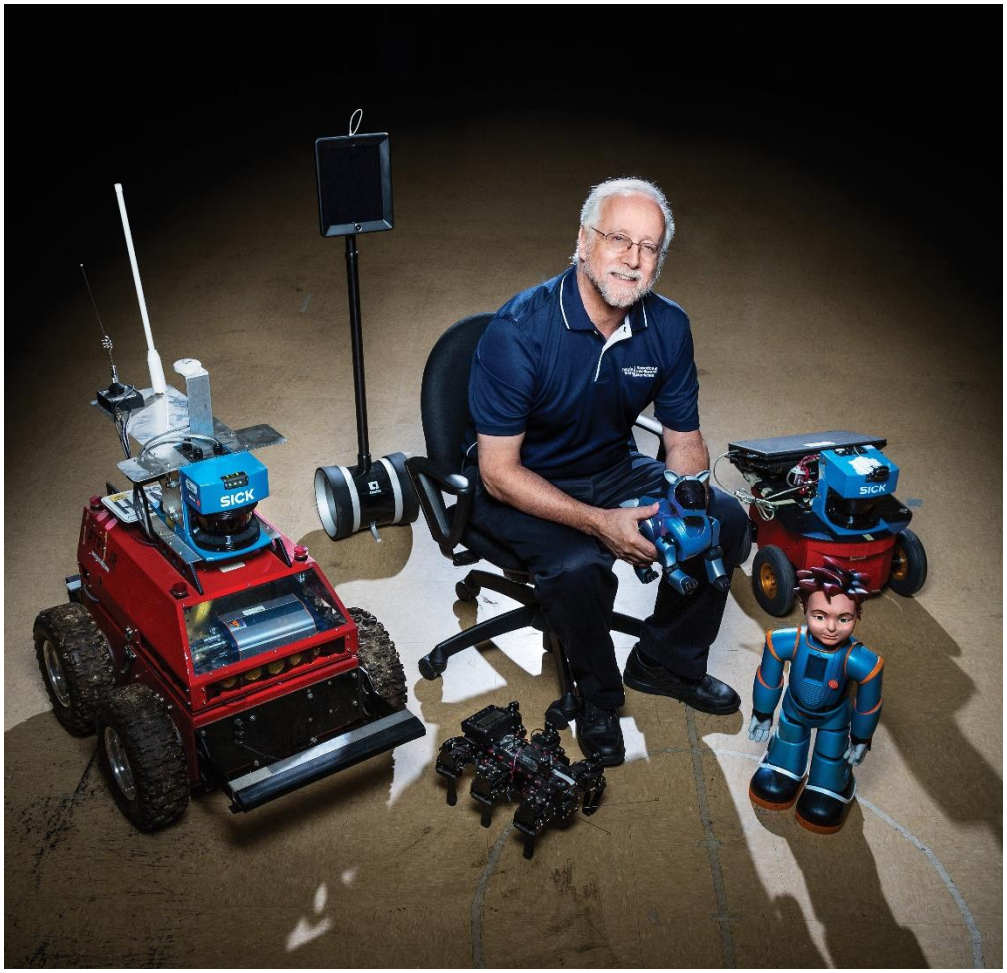
Subsumption Philosophy

- Modules should be grouped into *layers of competence*
- Modules in a higher lever can override or *subsume* behaviors in the next lower level
  - Suppression: substitute input going to a module
  - Inhibit: turn off output from a module
- ***No internal state*** in the sense of a local, persistent representation similar to a world model.



Subsumption Evaluation

- Strengths:
  - Reactivity (speed, real-time nature)
  - Parallelism
  - Incremental design => robustness
  - Generality
- Weaknesses:
  - Inflexibility at run-time
  - Expertise needed in design



Potential Fields: Ronald Arkin

Regents' Professor, Director of Mobile Robot Laboratory School of Interactive Computing, College of Computing, Georgia Tech

https://www.cc.gatech.edu/aimosaic/faculty/arkin/index.html

Potential Fields Philosophy

- The motor schema component of a behavior can be expressed with a potential fields methodology
  - A potential field can be a “primitive” or constructed from primitives which are summed together
  - The output of behaviors are combined using vector summation
- From each behavior, the robot “feels” a vector or force
  - Magnitude = force, strength of stimulus, or *velocity*
  - Direction
- But we visualize the “force” as a field, where every point in space represents the vector that it would feel if it were at that point

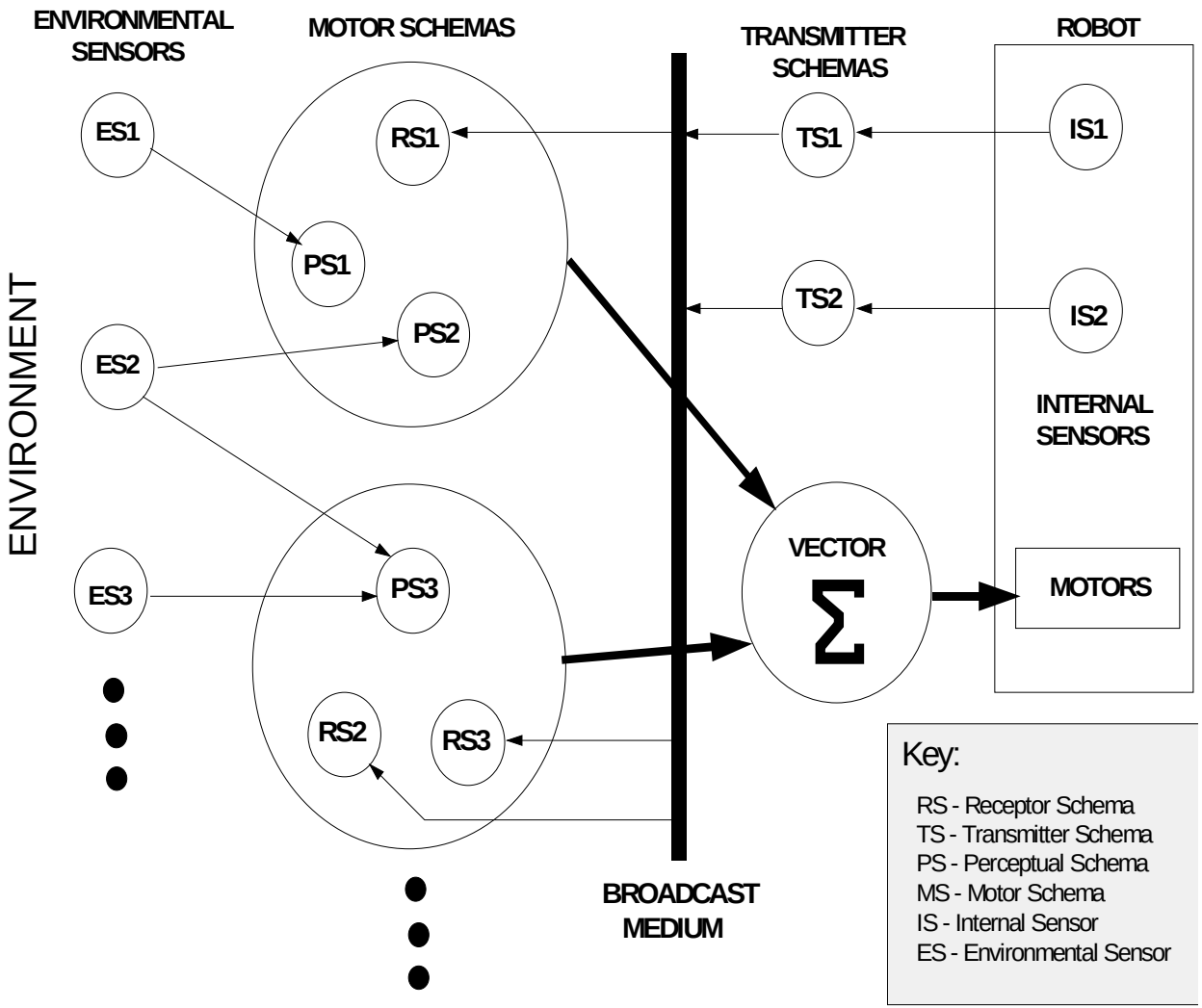
Schema Theory (as presented earlier)

SCHEMA is used in cognitive science and ethology to refer to a particular organized way of perceiving cognitively and responding to a complex situation or set of stimuli

- a behavior is a schema, consists of
  - perceptual schema
  - motor schema
  - other behaviors

Motor Schemas

- Motor schemas are a type of behavior encoding: They are based on schema theory (Arbib); Provide large grain modularity; Distributed concurrent schemas used; Based on neuroscience and cognitive sci.
- Represented as vector fields
- Composed into “assemblages” by fusion (not competition)


ENVIRONMENTAL SENSORS
MOTOR SCHEMAS
TRANSMITTER SCHEMAS
ROBOT
ENVIRONMENT
ES1
ES2
ES3
RS1
PS1
PS2
PS3
RS2
RS3
TS1
TS2
IS1
IS2
INTERNAL SENSORS
VECTOR
Σ
MOTORS
BROADCAST MEDIUM
Key:
RS - Receptor Schema
TS - Transmitter Schema
PS - Perceptual Schema
MS - Motor Schema
IS - Internal Sensor
ES - Environmental Sensor

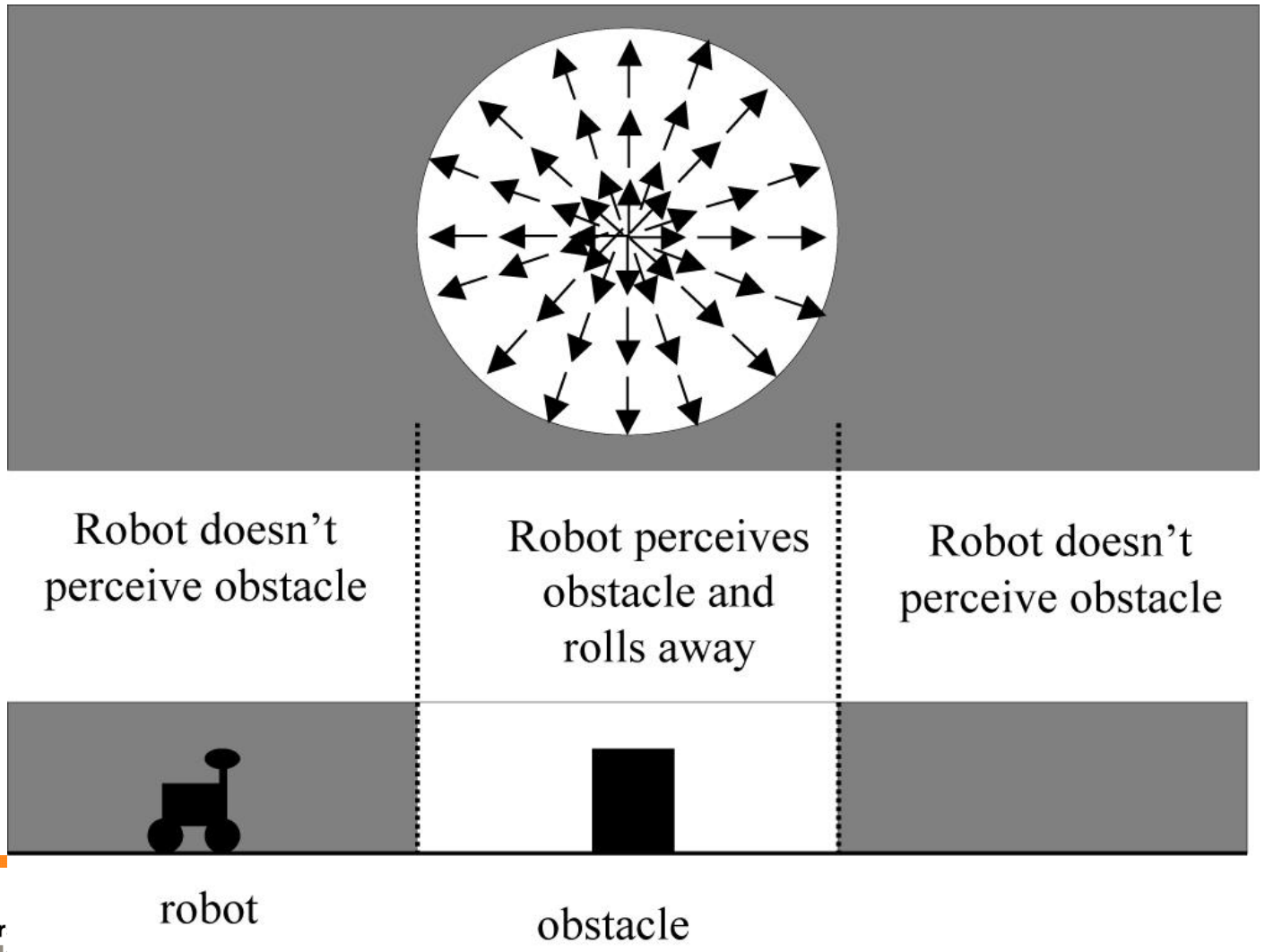
Schema Representation

- Responses represented in uniform vector format
- Combination through cooperative coordination via vector summation
- No predefined schema hierarchy
- Arbitration is not used - gain values control behavioral strengths

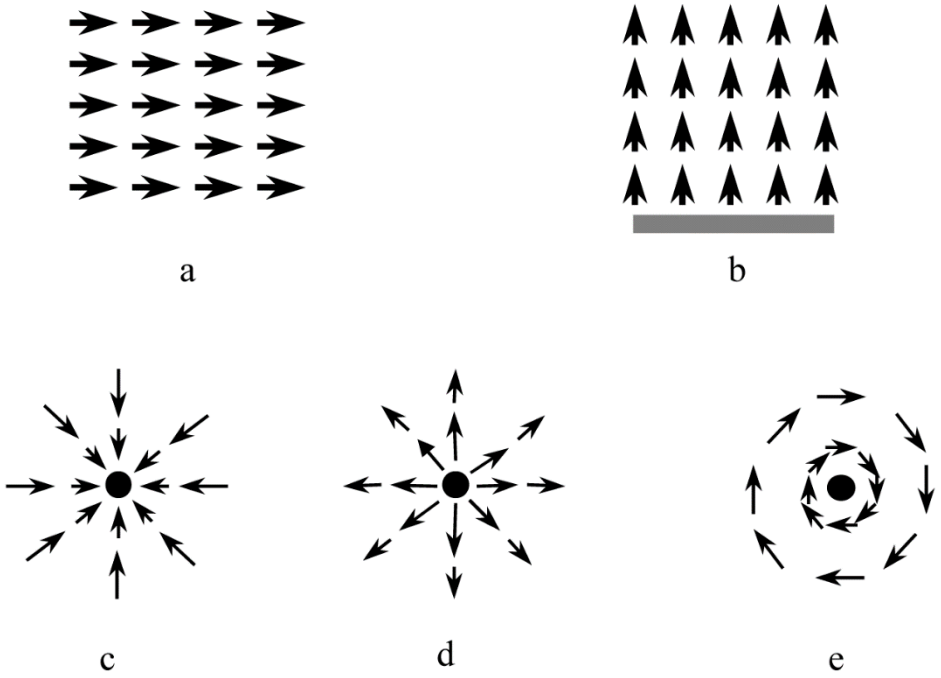
Designing with Schemas

- Characterize motor behaviors needed
- Decompose to most primitive level, use biological guidelines where appropriate
- Develop formulas to express reaction
- Conduct simple simulations
- Determine perceptual needs to satisfy motor schema inputs
- Design specific perceptual algorithms
- Integrate/test/evaluate/iterate

Example: Run Away via Repulsion

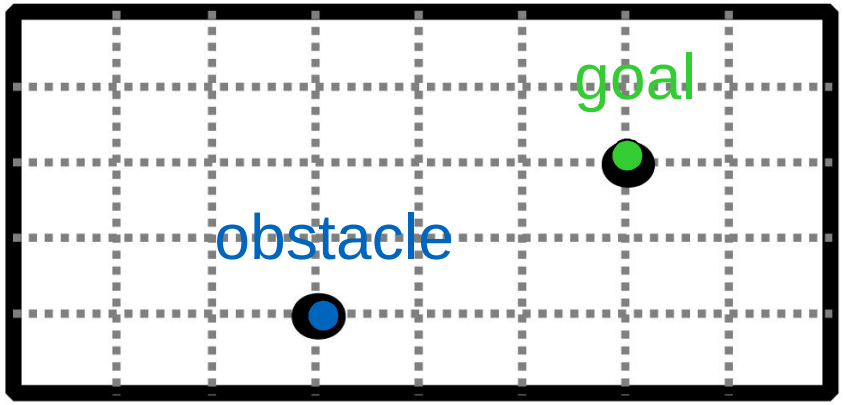


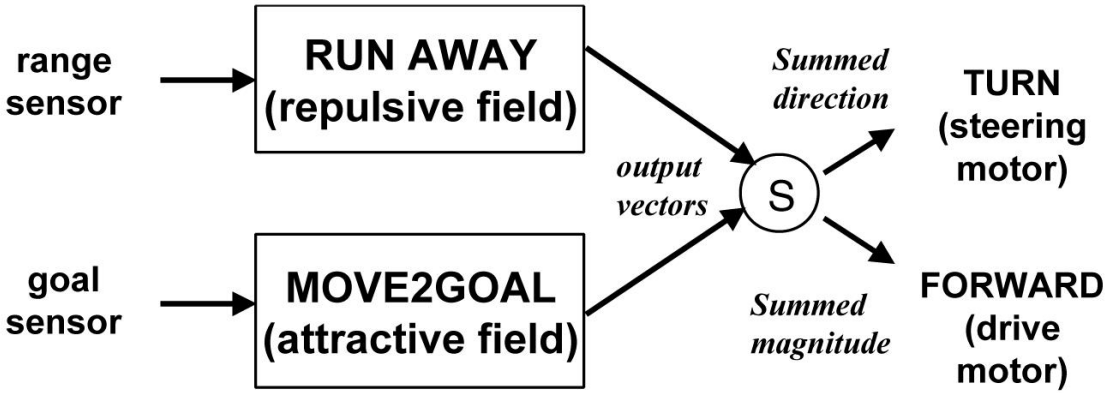
5 Primitive Potential Fields



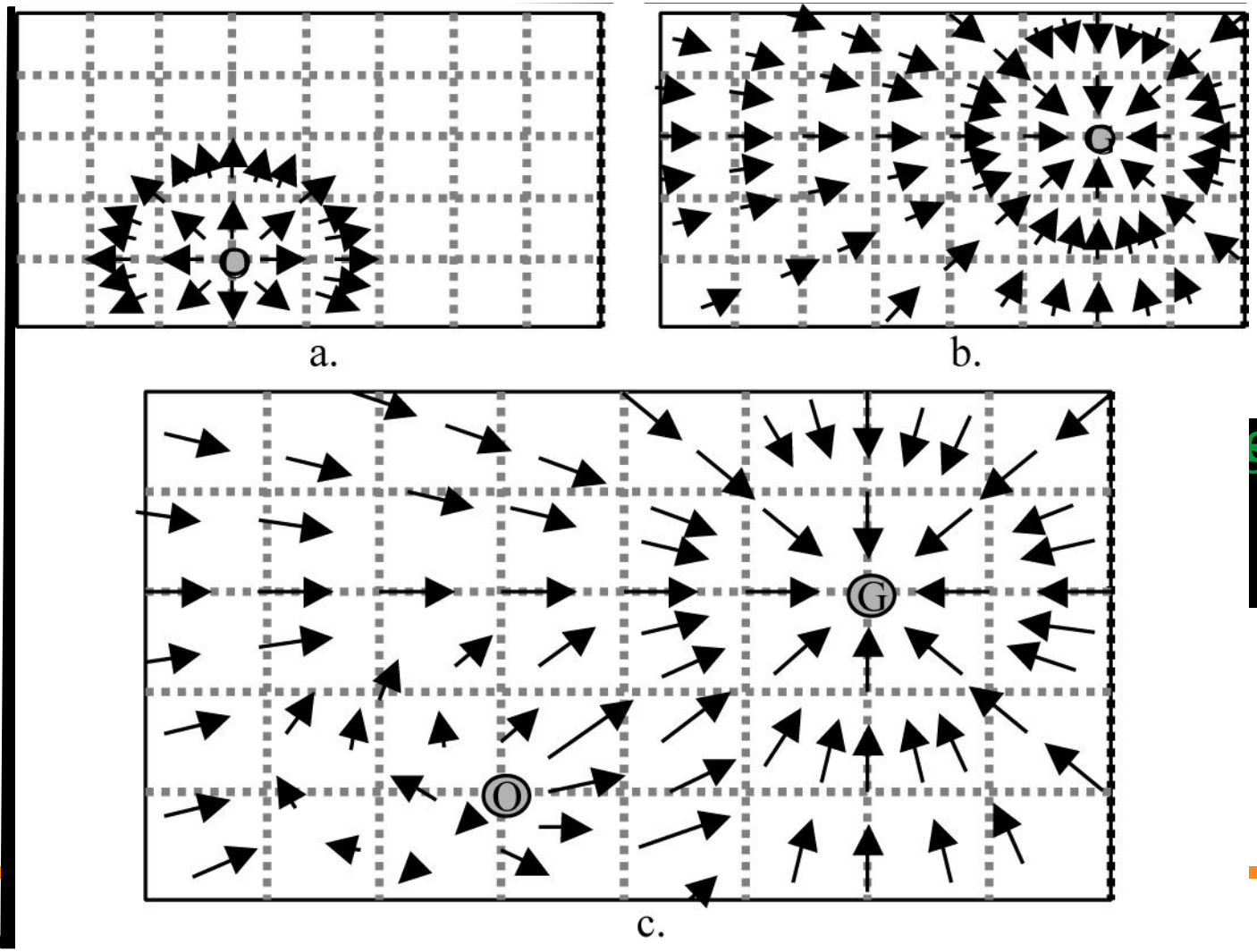
a) Uniform b) Perpendicular c) Attractive d) Repulsive e) Tangential

Combining Fields for Emergent Behavior



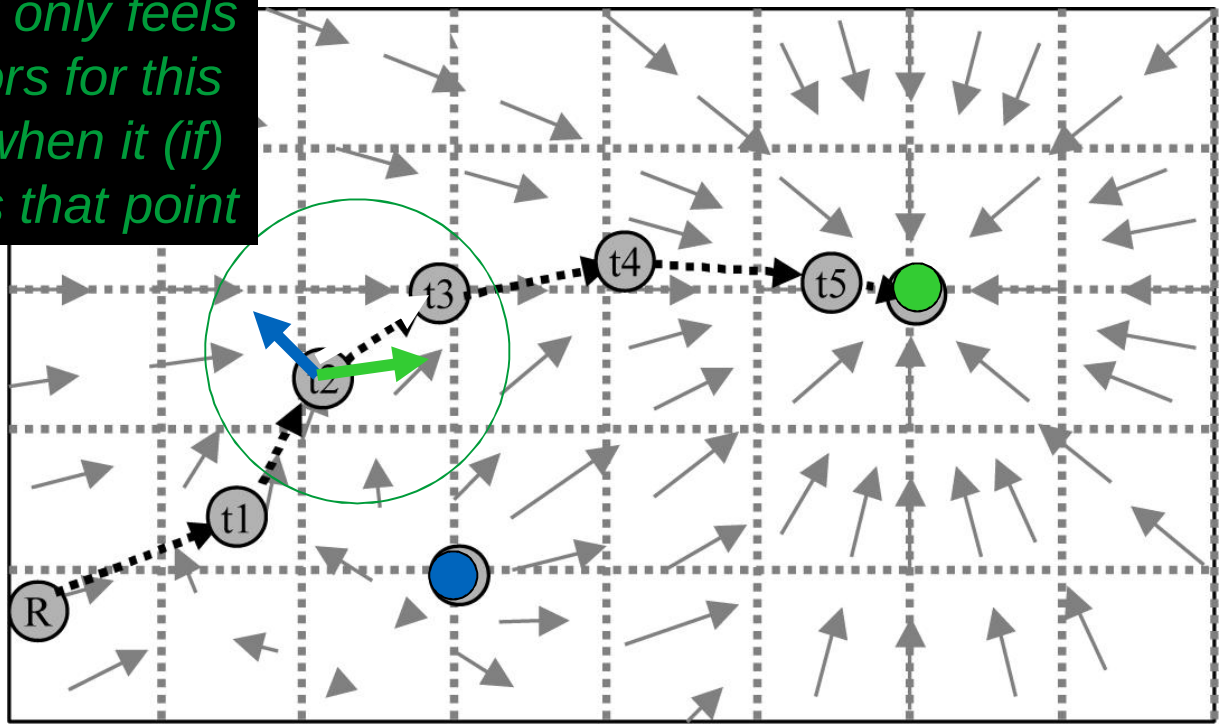


Fields and Their Combination



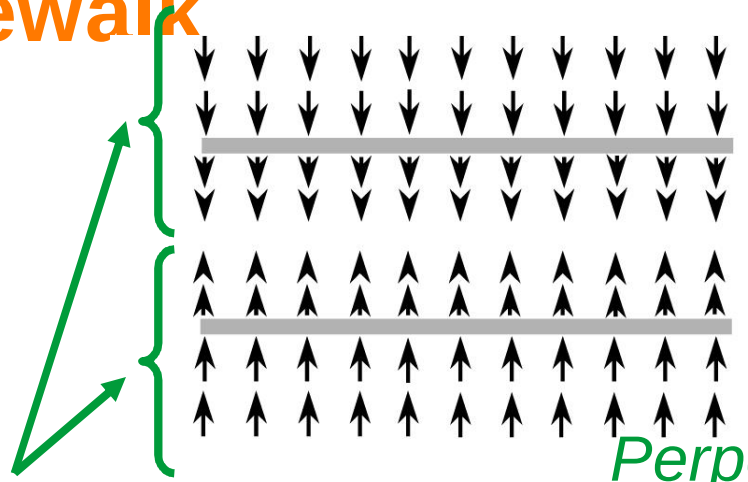
Path Taken

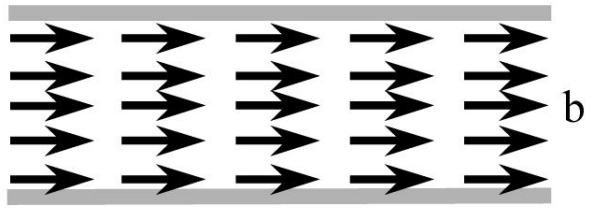
Robot only feels vectors for this point when it (if) reaches that point



- If robot started at this location, it would take the following path
- It would only “feel”the vector for the location, then move accordingly, “feel” the next vector, move, etc.
- Pfield visualization allows us to see the vectors at all points, but robot never computes the “field of vectors” just the local vector

Example: follow-corridor or follow-sidewalk

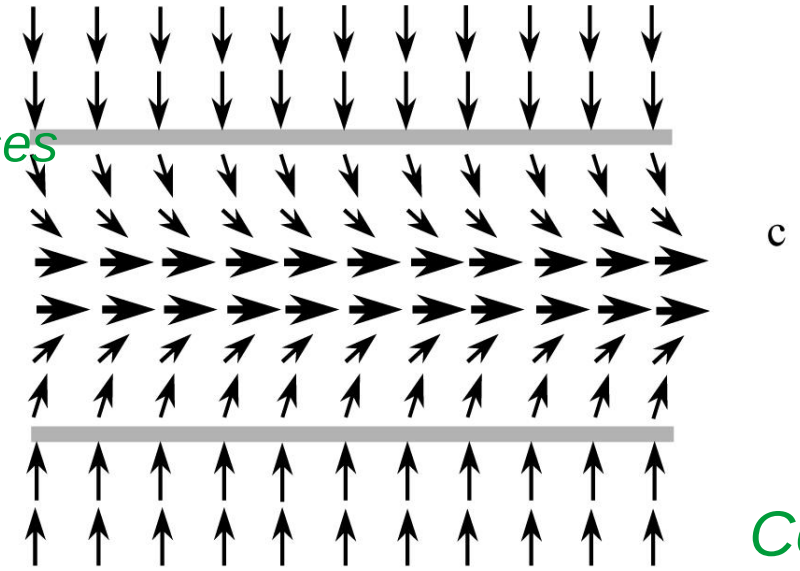




Perpendicular

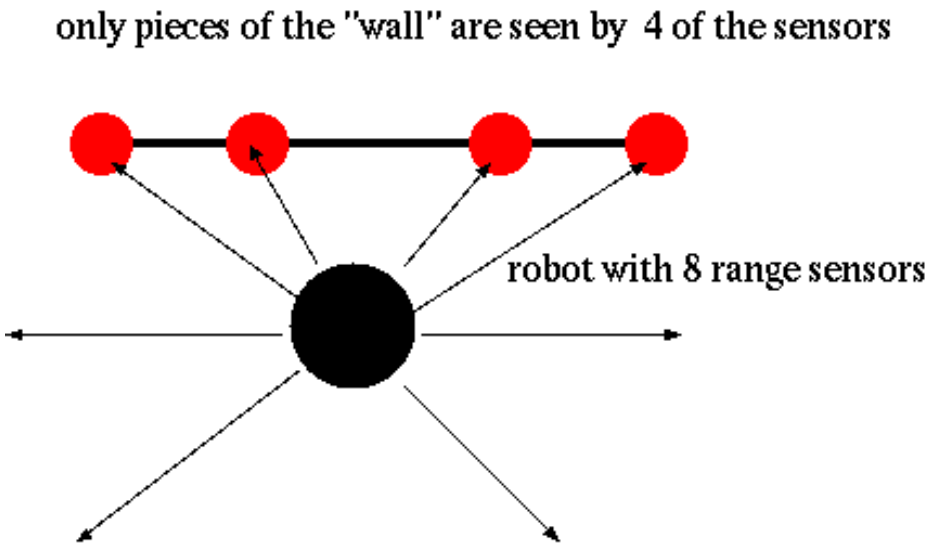
Uniform

Note use of Magnitude profiles: Perpendicular decreases

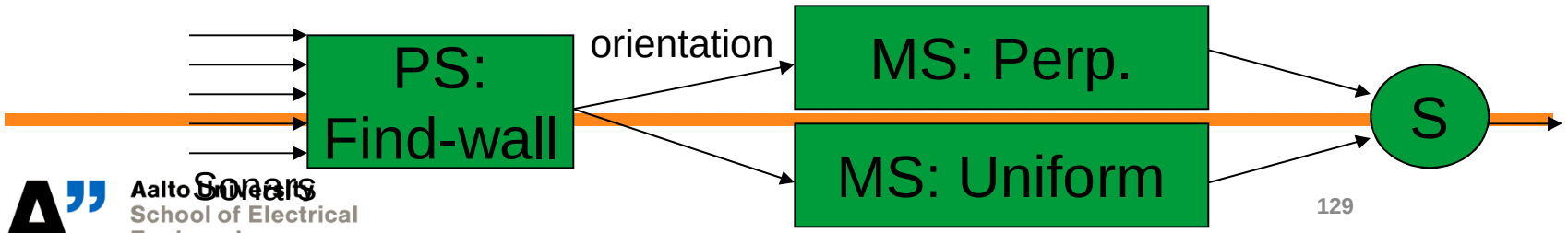


Combined

But how does the robot see a wall without reasoning or intermediate representations?



- Perceptual schema “connects the dots”, returns relative orientation



Strengths and Weaknesses

- Advantages
  - Easy to visualize
  - Easy to build up software libraries
  - Fields can be parameterized
  - Combination mechanism is fixed, tweaked with gains
  - Support for modularity and parallelism
  - Run-time flexibility
- Disadvantages
  - Local minima problem (sum to magnitude=0)
    - Box canyon problem
  - Jerky motion

HYBRID SOLUTIONS

Inventing Hybrid Control

- The basic idea is simple: we want the best of both worlds (if possible)
- That means to combine reactive and deliberative control
- This implies combining the different time-scales and representations
- This mix is called hybrid control

Key Questions

- How does the architecture distinguish between reaction and deliberation?
- How does it organize responsibilities in the deliberative portion?
- How does overall behavior emerge?

Common Functionalities

- Mission planner
- Cartographer
- Sequencer
- Behavioral (resource) manager
- *Performance monitor/problem solving agent*

Mission planner

Mission planner interacts withe the human operator, operationalizes the commands into robot terms and creates the actual mission plan.

Cartographer

Cartographer is responsible for creating, storing and maintaining map (or any relevant spatial information) plus methods and routines to access the data. It often contains a global word model and relevant knowledge representations.

Sequencer

A sequencer agent generates the set of behaviors to accomplish a subtask and determines the needed sequences and activation conditions. A sequence is often represented as a finite state machine and the sequencer should either generate it or be able to dynamically modify it accordingly.

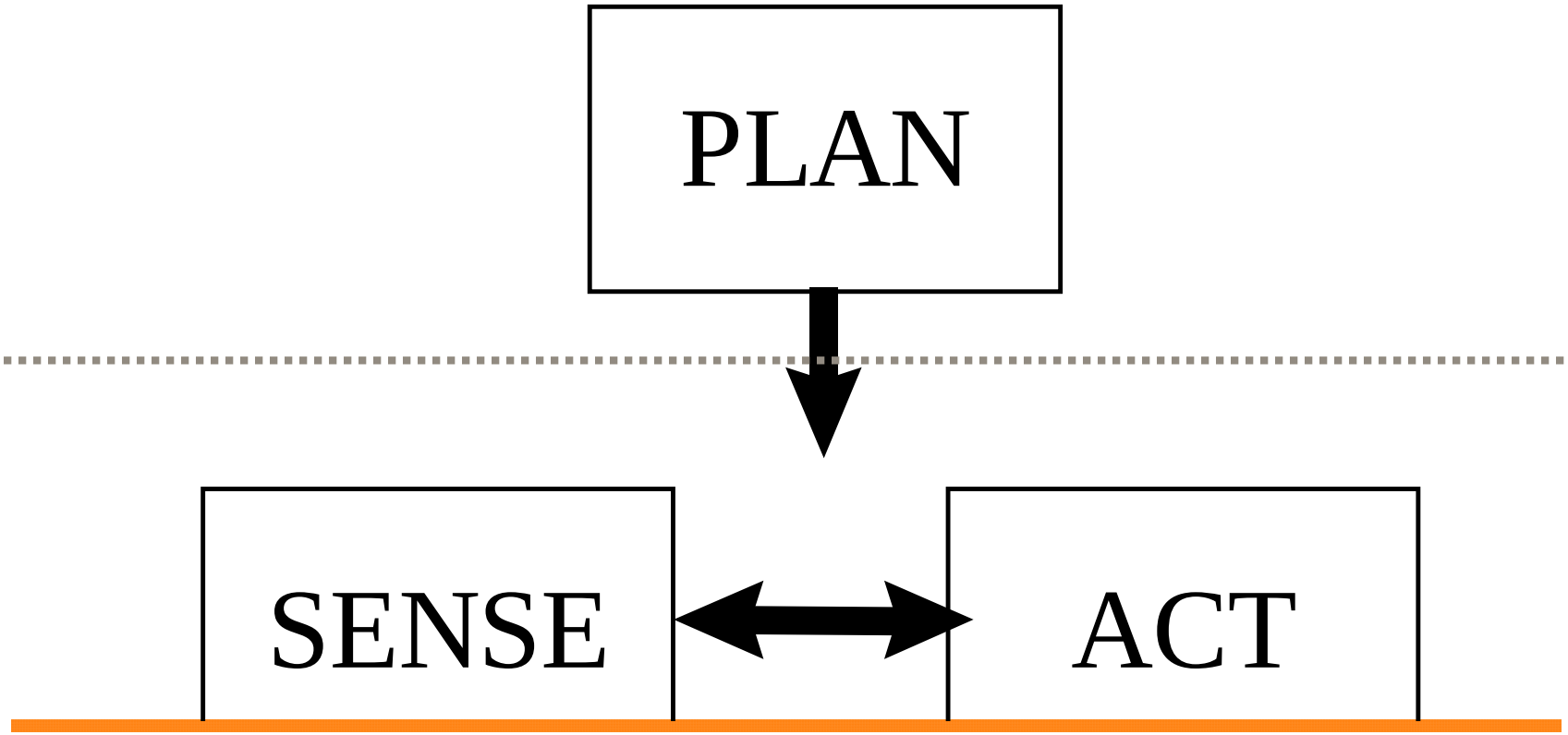
Behavioral (resource) manager

This manager allocates resources to planned behaviors, mainly by selecting them from libraries of schemas. Its job includes for example selecting a suitable active sensor set for task at hand. Remember that you want to have redundancy to secure the perception, but you don't want to overdo it for several reason (e.g., time, energy, CPU usage etc.) Less is often more.

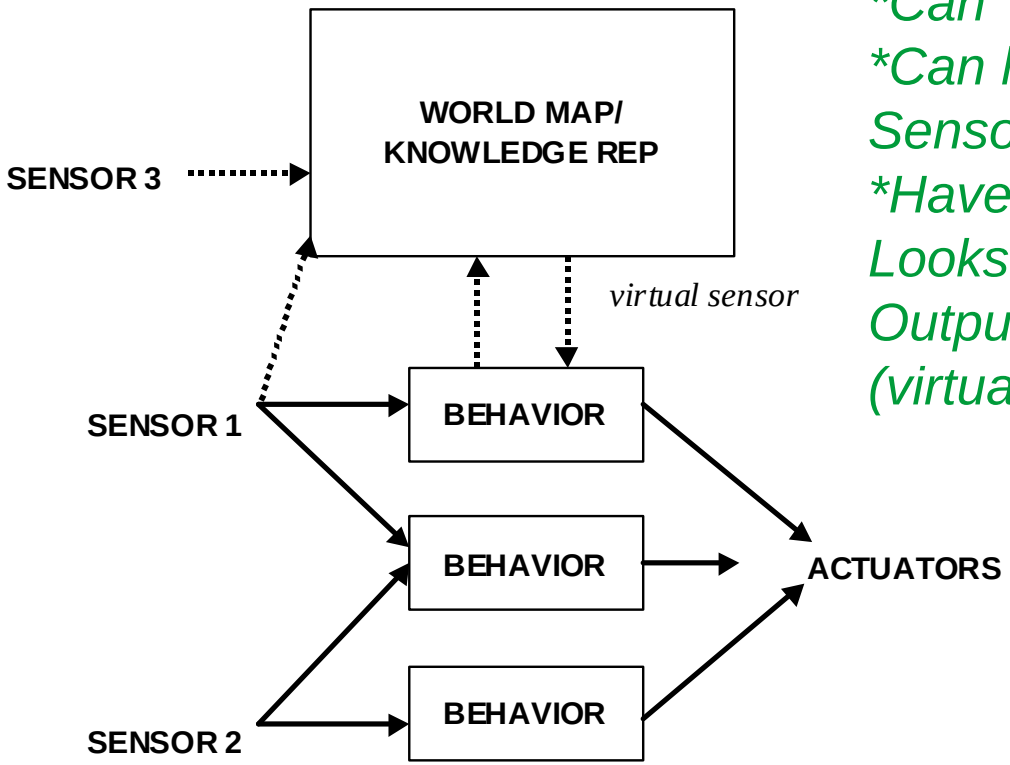
Performance monitor/problem solving agent

This agent tracks how well the robot is doing its job, i.e. how well (and fast) it is achieving the mission objectives. It might include mechanisms to detect and identify various problems and provide a set of solutions for them. For example by monitoring the energy level and proposing when and where to go when the level goes below some safety threshold.

Organization: Plan, Sense-Act



Sensing Organization



Deliberative functions
**Can "eavesdrop"*
**Can have their own Sensors*
**Have output which Looks like a sensor Output to a behavior (virtual sensor)*

Organizing Hybrid Systems

- A hybrid system typically consists of three components:
  - a reactive layer
  - a planner
  - a layer that puts the two together
- Hybrid architectures are often called three-layer architectures
- The planner and the reactive system are both standard, as we have covered them so far

The Magic Middle

- The middle layer has a hard job:
  - 1) compensate for the limitations of both the planner and the reactive system
  - 2) reconcile their different time-scales
  - 3) deal with their different representations
  - 4) reconcile any contradictory commands between the two
- This is the challenge of hybrid systems => achieving the right compromise between the two ends

Dynamic Re-planning

- Reaction can influence planning
- Any "important" changes discovered by the low-level controller are passed back to the planner in a way that the planner can use to re-plan
- The planner is interrupted when even a partial answer is needed in real-time
- The reactive controller (and thus the robot) is stopped if it must wait for the planner to tell it where to go.

Planner-Driven Reaction

- Planning can influence reaction
- Any "important" optimizations the planner discovers are passed down to the reactive controller
- The planner’s suggestions are used if they are possible and safe => Who has priority, planner or reactor?
- It depends, as we will see...

Universal Plans

- Suppose for a given problem, all possible plans are generated for all possible situations in advance, and stored (e.g., automated tic-tack-toe)
- If for each situation a robot has a pre-existing optimal plan, it can react optimally, be reactive and optimal, it has a universal plan (These are complete reactive mappings)

Viability of Universal Plans

- A system with a universal plan is reactive; the planning is done at compile-time, not at run-time
- Universal plans are not viable in most domains, because:
  - the world must be deterministic
  - the world must not change
  - the goals must not change
  - the world is too complex (state space is too large)

Hybrid Summary

- P,S-A, deliberation uses global world models, reactive uses behavior-specific or virtual sensors
- Architectures generally have modules for mission planner, sequencer, behavioral mgr, cartographer, and performance monitoring
- Deliberative component is often divided into sub-layers (sequencer/mission planner or managers/mission planner)
- Reactive component tends to use assemblages of behaviors

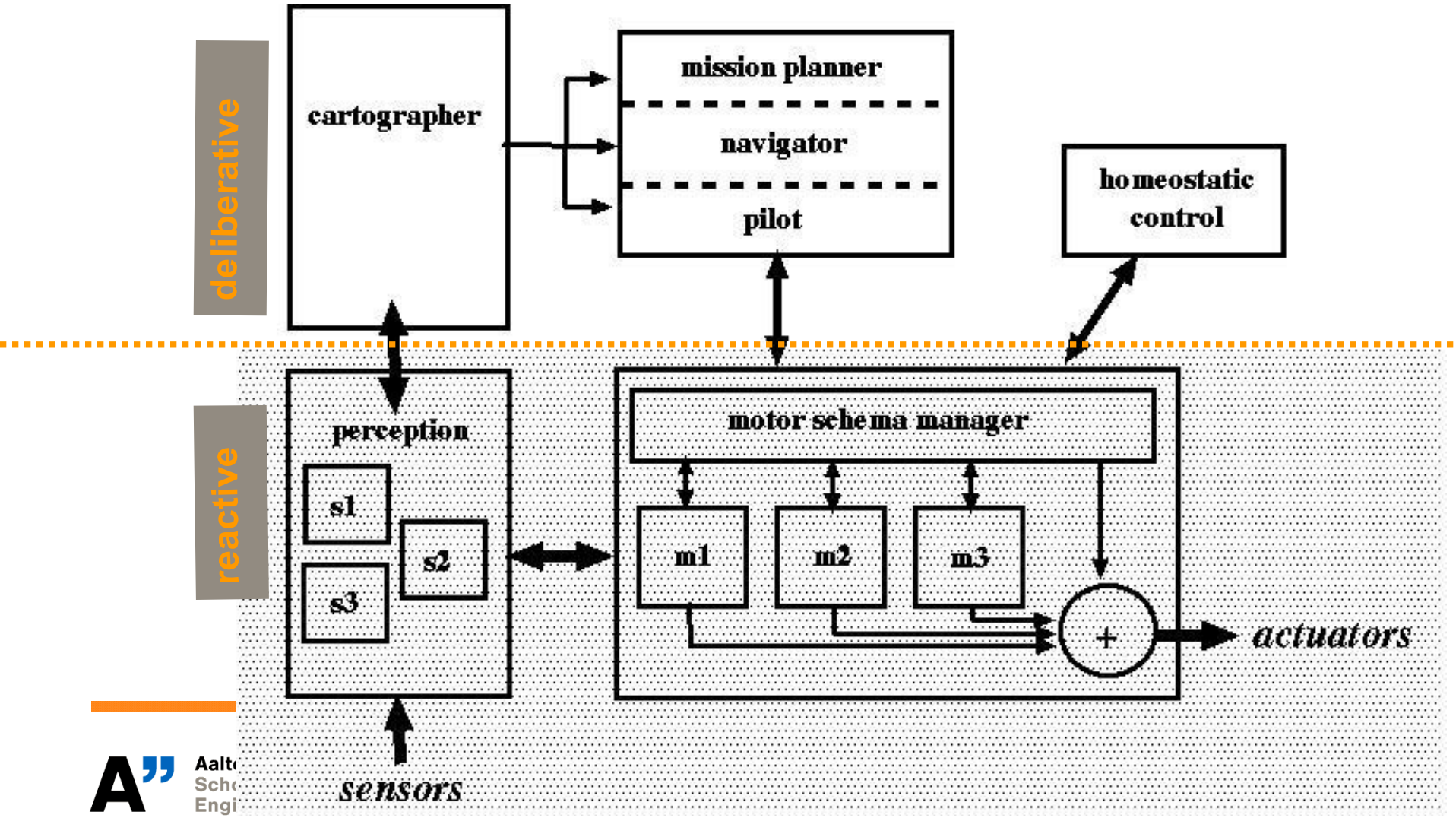
Relative Strengths
(Hybrid Control)

- Deliberative planners
  - Rely heavily on world models
  - Can readily integrate world knowledge
  - Have broader perspective and scope
- Reactive & behavior-based systems
  - Afford modular development
  - Provide real-time robust performance in dynamic world
  - Provide for incremental growth
  - Tightly coupled to incoming sensory data

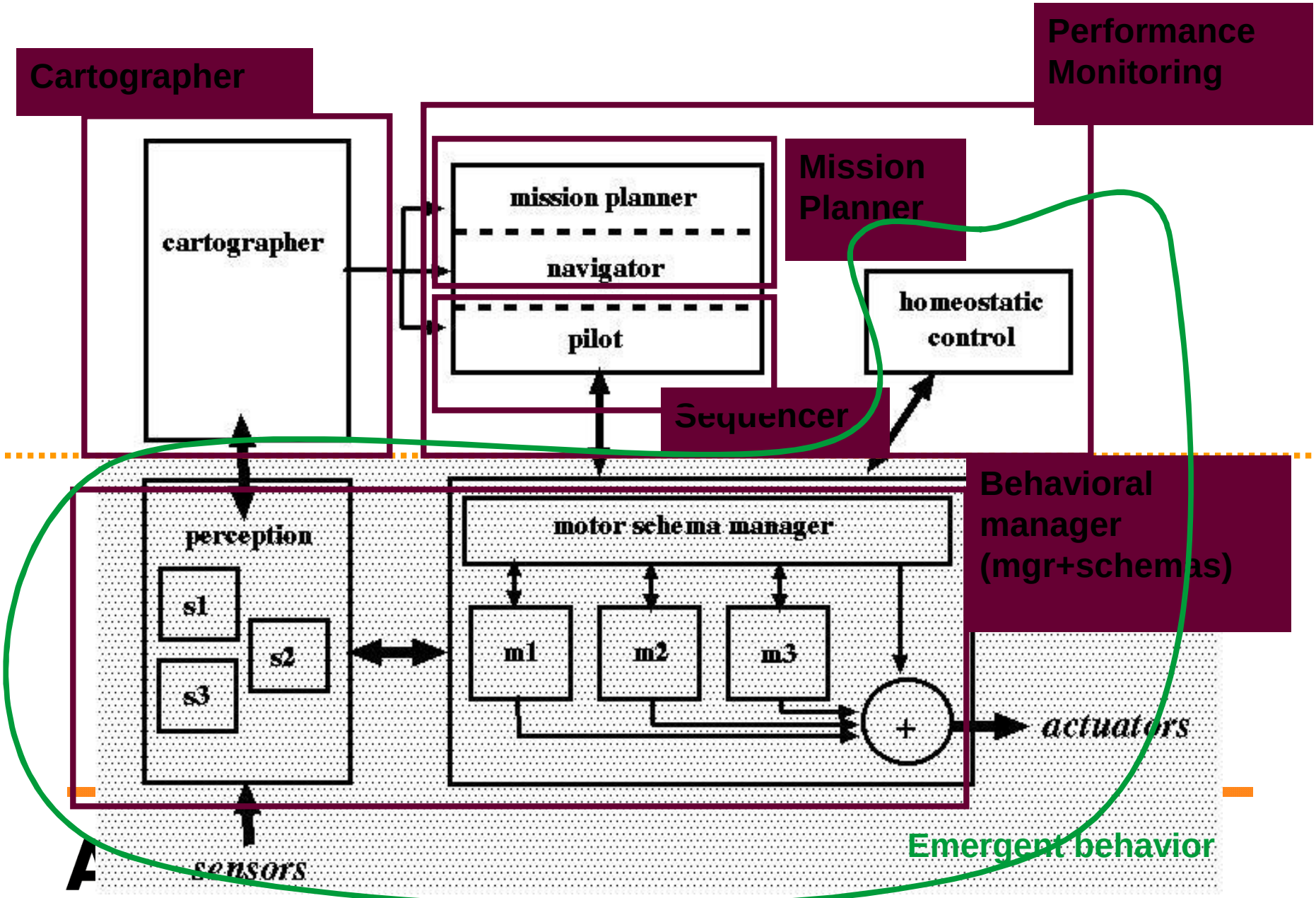
Example: AuRA

- Roland C. Arkin (1986)
- Planning is viewed as configuration
- Initial A* planner integrated with schema-based controller
- Provides modularity, flexibility, and adaptability

AuRA Architectural Layout



AuRA Architectural Layout



READ THE ATTACHED JOURNAL PAPER

Ronald C. Arkin & Tucker Balch (1997) AuRA: principles and practice in review, Journal of Experimental & Theoretical Artificial Intelligence, 9:2-3, 175-189, DOI: 10.1080/095281397147068

Robot control refers to the way in which the sensing and action of a robot are coordinated. The many different ways how robots can be controlled all fall along a well-defined spectrum of control ranging from purely symbolic, detailed representation dependent planning based systems all the way to reactive, representation-free real-time solutions.

React when you can, plan when you must.

-Tom Mitchell, CMU

Robot architecture provides a structured and principled way of organizing a control system. It defines the components, their interactions, prevailing constraints and operations how to survive and achieve the given mission objectives.