EGN 3060c: Introduction to Robotics

Lecture 4:

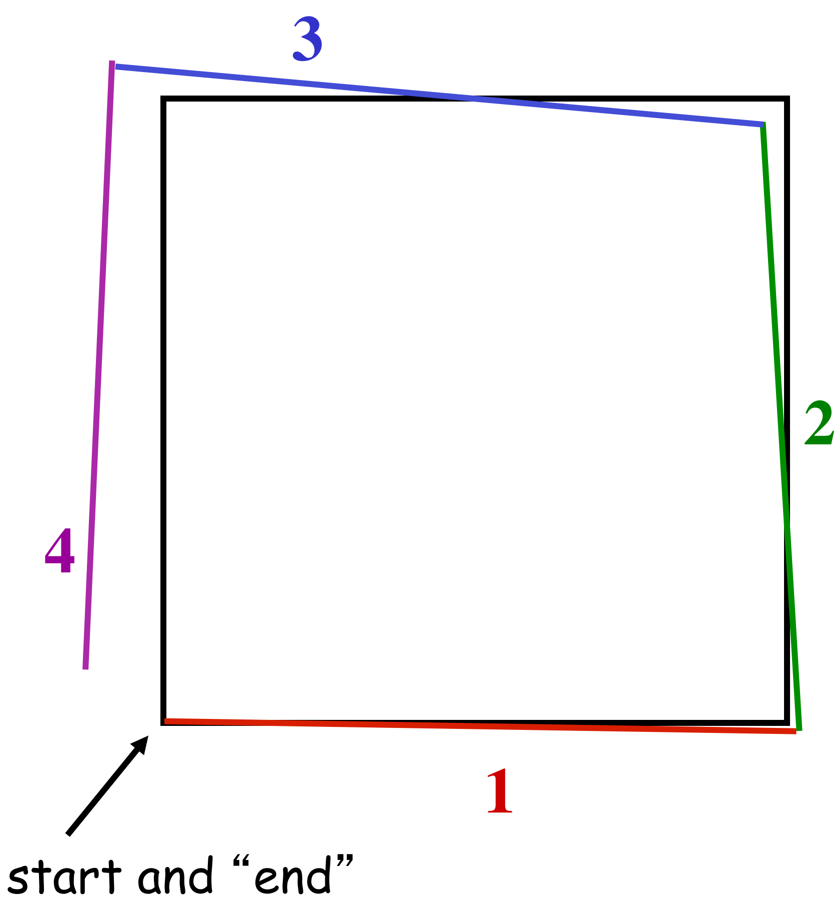
Robot Architectures: Deliberative and Reactive

Instructor: Dr. Gita Sukthankar
Email: gitars@eecs.ucf.edu

Announcements

- Webcourses should be updated with the background material and reading.
- Please make sure that you have formed groups; I will try to add the groups into webcourses.
- HW1 due today; please doublecheck it to make sure that it is coherently written.
- No class on Monday (Labor Day)
- Lab 2: robot's movement and odometry
- Next assignment due: Sept 17th (Lab 2 report)
- Please review the terk javadocs under Lab directory

Lab 2: Robot Movement



- Create a program that will run your robot in a square (~2m to a side), pausing after each side before turning and proceeding.
- For 4 runs, collect both the odometric estimates of where the robot thinks it is and where the robot ***actually is*** after each side.
- You should end up with two sets of angle measurements and length measurements: one set from odometry and one from "ground-truth."
- Find the **mean** and the **standard deviation** of the ***differences*** between odometry and ground truth for the angles and for the lengths – this is the robot's ***motion uncertainty model***.

This provides a *probabilistic kinematic* model.

Useful Robot Functions

Purpose	Functions

User input	`waitforPlay()`
	`isPlaying()`
	`getTextFieldValueAsInt()`

Sensors	`bumpRight()` `bumpLeft()`
Movement	`moveMotors()` `moveDistance()`
Turning	`moveAngle()`
Stopping	`stopMoving()`
Program Output	`writeToTextField()`
Other	`dockRobot()` `unDockRobot()`

Lab Report

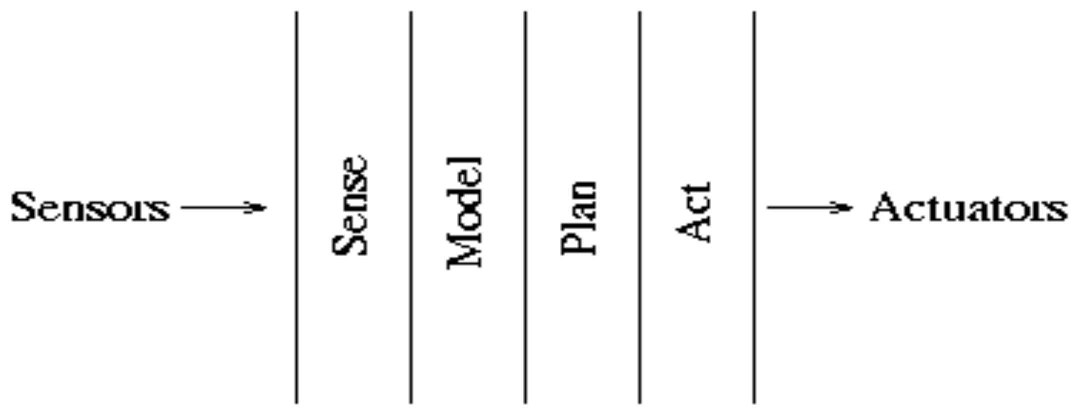
- First one due Sept 17^{th}
- Between 1-2 page writeup describing your robot code
- Include any other pictures or results specifically asked for
- Useful styles
  - Pseudocode
  - Javadoc style comments about key methods
  - List of steps describing operation of robot
  - Free form text does **not** usually work well
- Accompanying java files for the main section of program and any other new classes and methods introduced (CreateMove.java)
- Graded on: clarity, detail, and correctness (1-3 pts)

Robotics Research

- Complete system vs. partial system?
- No hardware—simulation only, computer vision?
- Dirty, dull, or dangerous application areas?
- Service robotics: devices to help humans in their activities of daily living
- General research areas:
  - Manipulation/locomotion (actuation)
  - Computer vision (perception)
  - Machine learning or planning (cognition)
  - Human-robot interaction
  - Mobile robotics
  - Humanoid robotics
  - UAVs

Deliberative Architecture

- Maps, lots of state
- Look-ahead

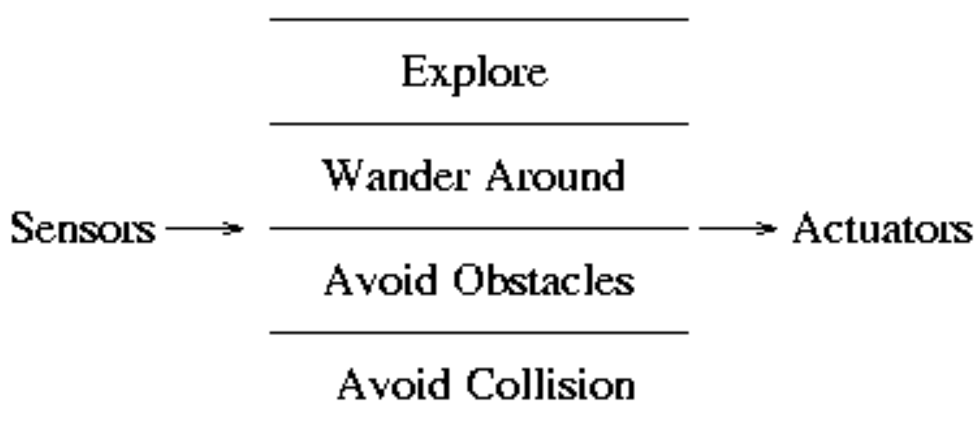


Reactive Architecture

- No maps, no state
- No look ahead

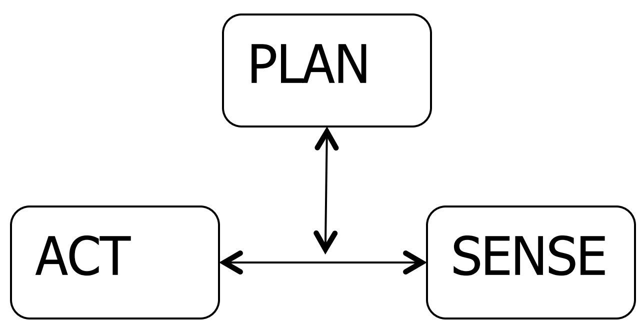
Behavior-based Architecture

- Reactive + state information
- State information allows robot to retain memory of previous actions
- Easily implemented



Hybrid architectures

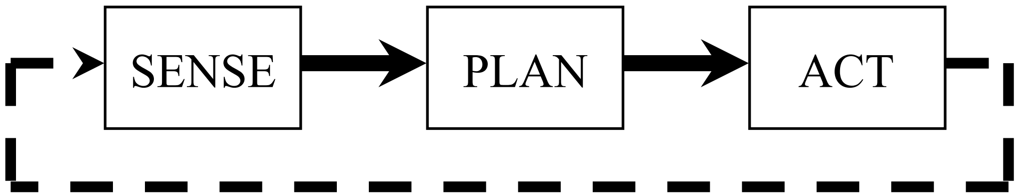
- Information flows in multiple directions
- Implementation is often multi-threaded
- Look ahead but continue to react to incoming sensory information
- Combines long and short time scales
- Used in most real-world robotic systems



Criteria For Selection

	deliberative	reactive	behavior
Task and environment	structured	unstructured	both
Run-time constraints	“thinking time”	“reflex”	mix
Correctness/ Completeness	provable	hard	really hard
Hardware	Sensors, processor	Lots of sensors	mix

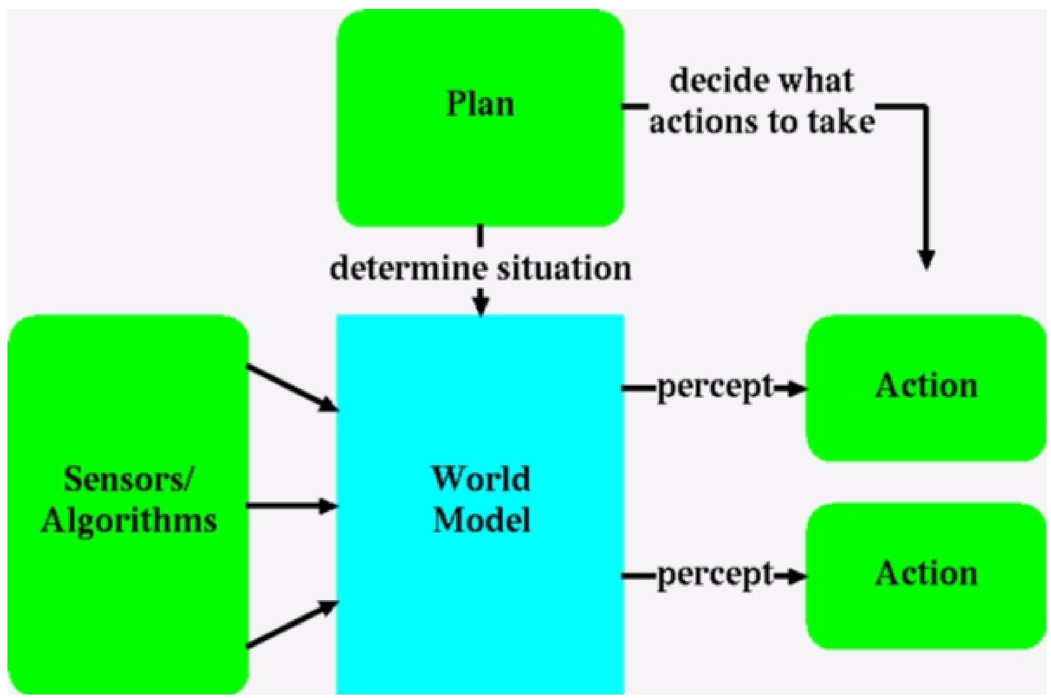
Deliberative Architecture



▪Robot builds a model of the world, and deliberates over the model before acting.

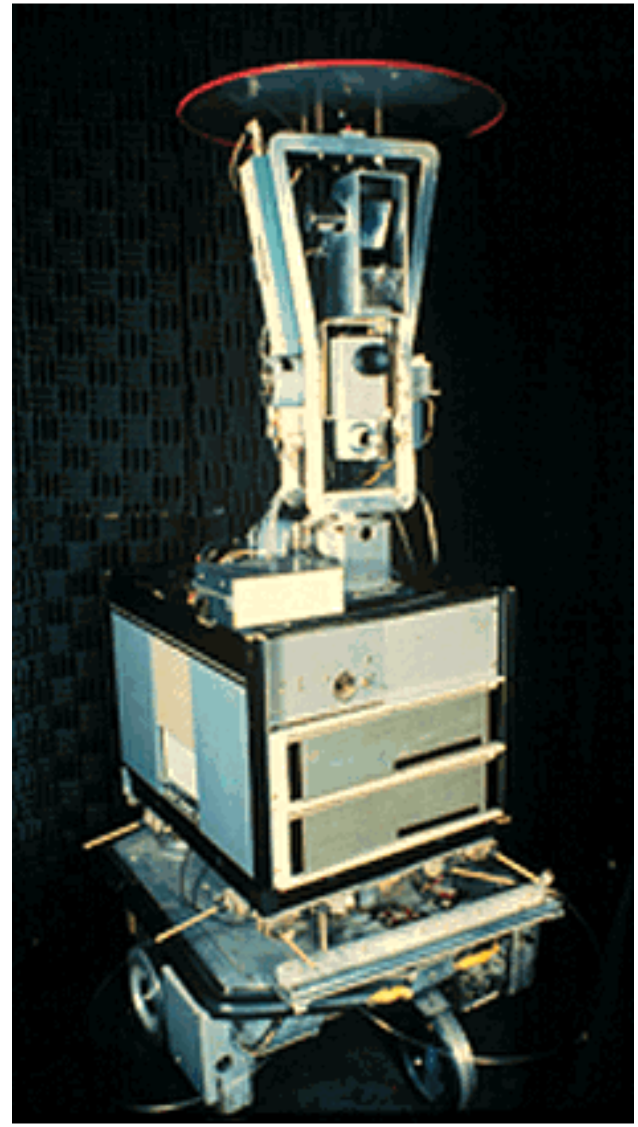
World model:

1. A priori rep
2. Sensed info
3. Cognitive



Shakey

- Early research robot platform
- Built by SRI (Stanford Research Institute) for DARPA 1967-9
- Used ***STRIPS*** as main algorithm for controlling what to do
- (**ST**anford **R**esearch **I**nstitute **P**roblem **S**olver)
- STRIPS uses a propositional logic representation of the world.
- 50th anniversary of Shakey!



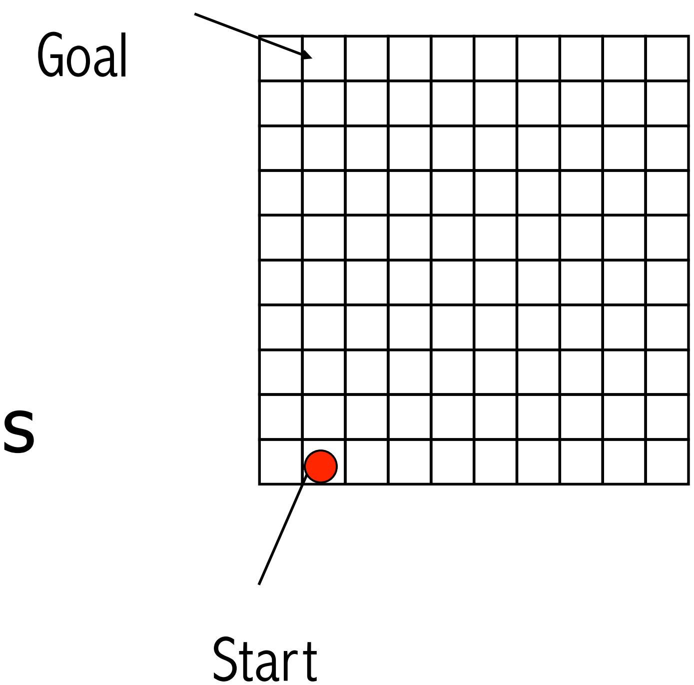
General Approach to Planning

- Define
  - Possible states (e.g. situations)
  - Operators (actions) that move the robot from one state to another
  - Operator costs
- Problem
  - Find some sequence of operators that move robot from start state to goal state
- Optimize
  - Search to find operator sequence with minimum cost

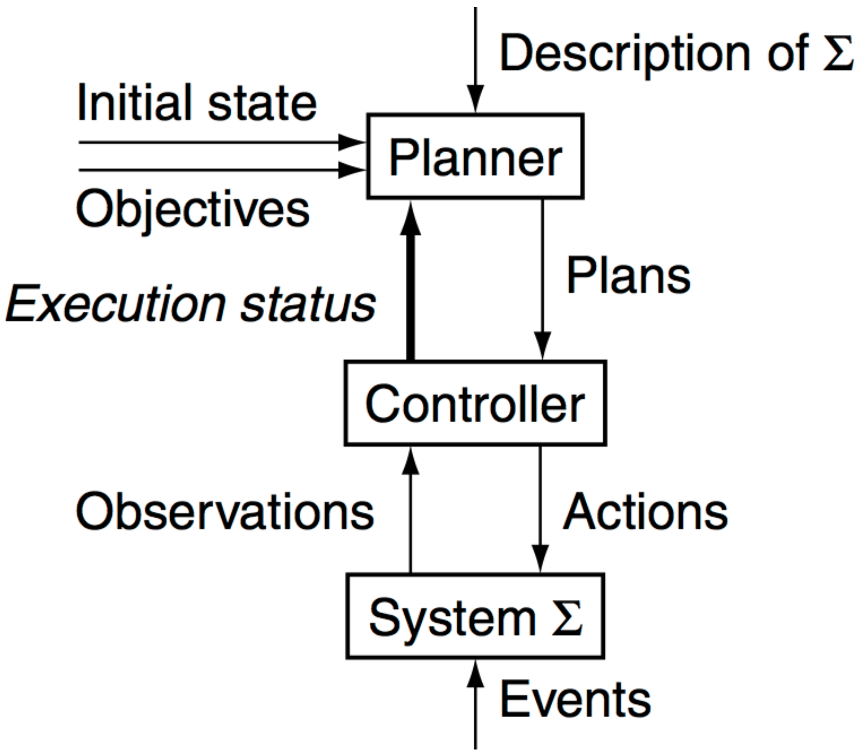
Example: Navigation

- State: location (x,y)
- Operators: move N, S, E, W
- Costs: 1 per move

- Start
- Goal
- But planning can be applied to many problems beyond navigation



AI Planning



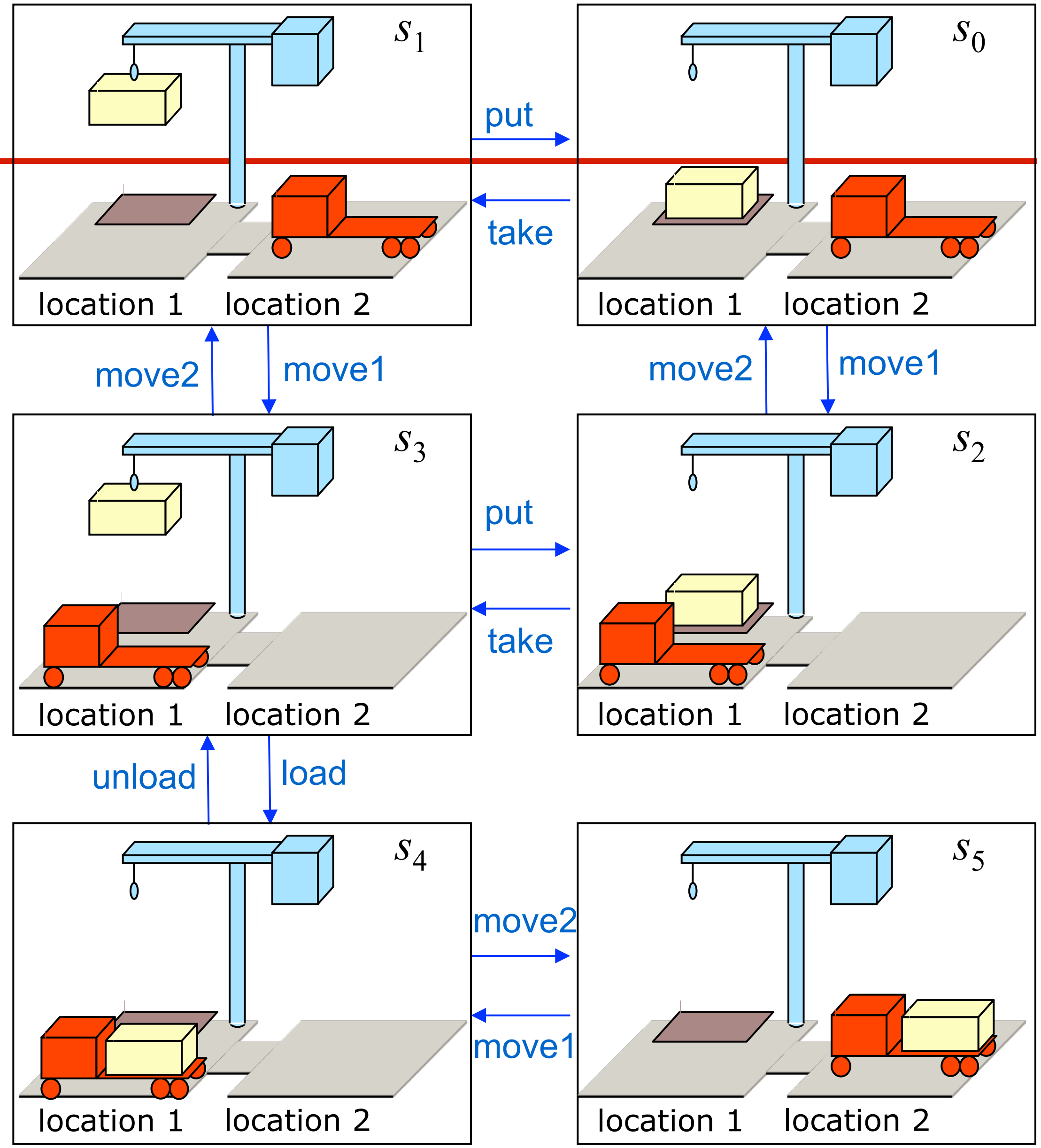
- **Planner** operates on a the (symbolic) model of world state
- Using search and optimization techniques the planner constructs a procedure for the agent to follow
- **Controller** takes plan and translates it into executable commands.

State Transition Model

$\Sigma = (S, A, E, \gamma)$

- S = {states}
- A = {actions}
- E = {exogenous events}
- State-transition function
  γ: $S \times (A \cup E)$

  - $S = \{s_0, \ldots, s_5\}$
  - A = {move1, move2, put, take, load, unload}
  - $E = \{\}$
  - γ: see the arrows



The Dock Worker Robots (DWR) domain

Problem Description

Description of Σ

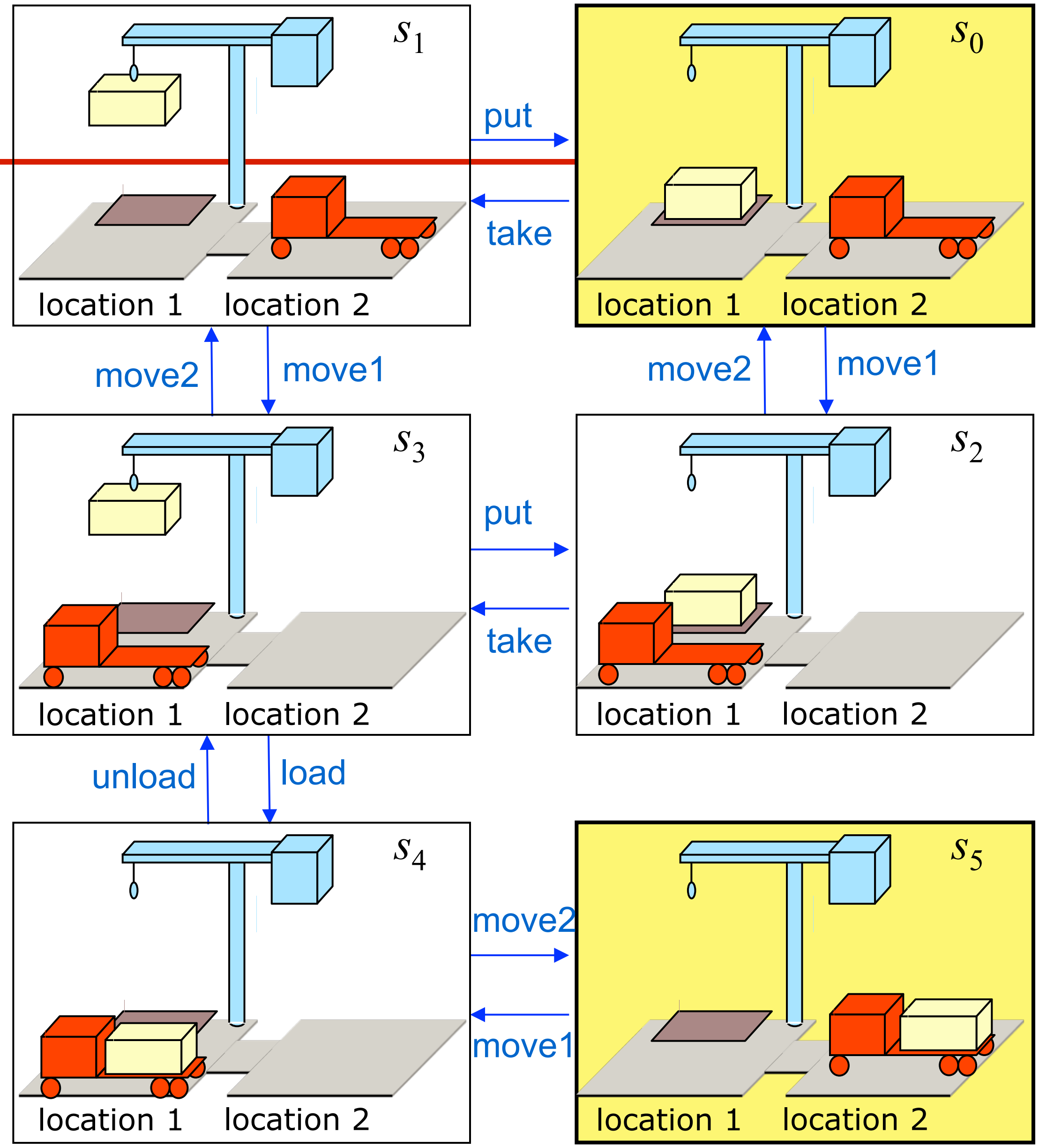
Initial state or set of states

Initial state = s_0

Objective

Goal state, set of goal states, set of tasks, "trajectory" of states, objective function, …

Goal state = s_5



The Dock Worker Robots (DWR) domain

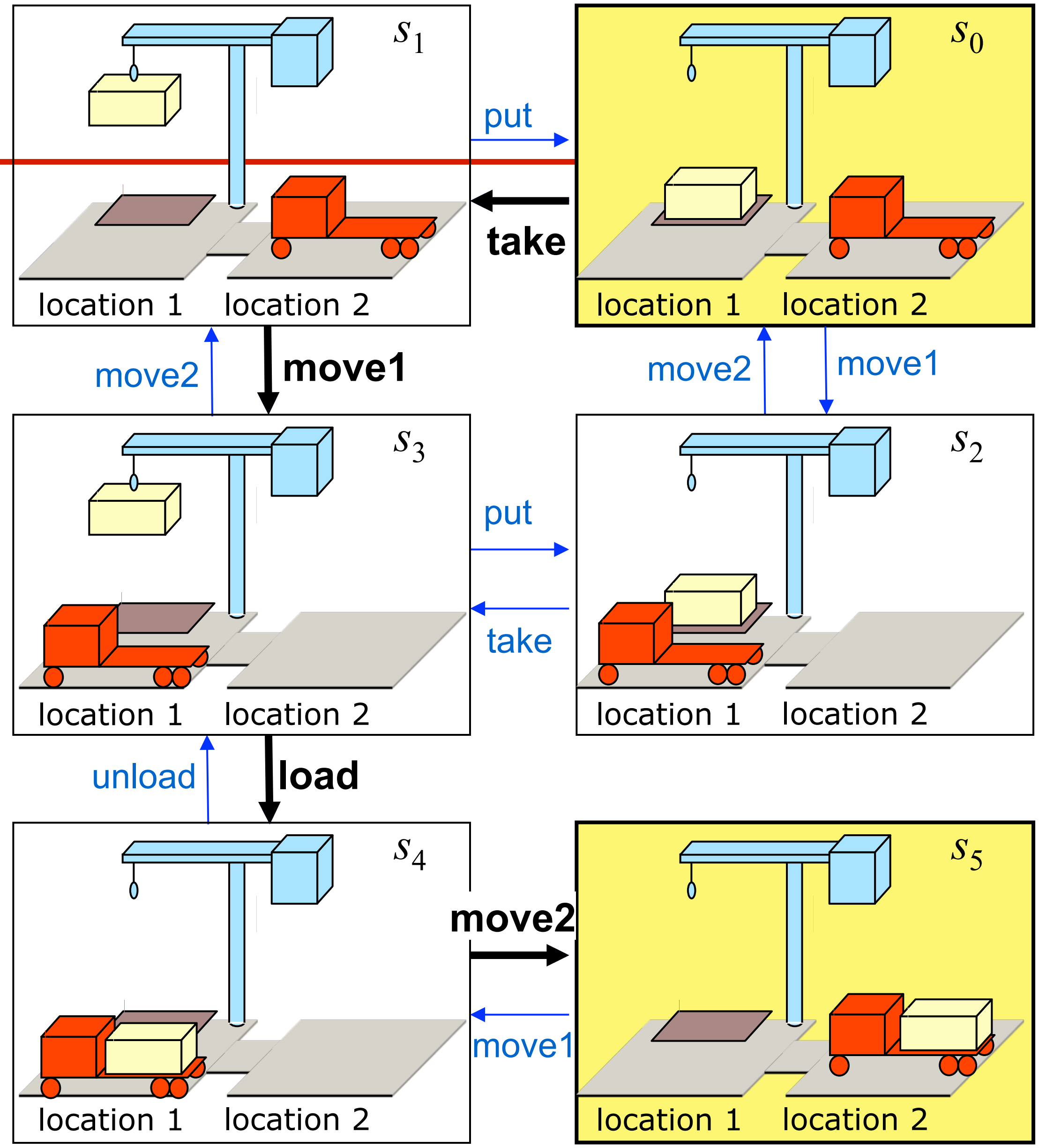
Plans

Classical plan: a sequence of actions

$\langle$take, move1, load, move2$\rangle$

Policy: partial function from S into A

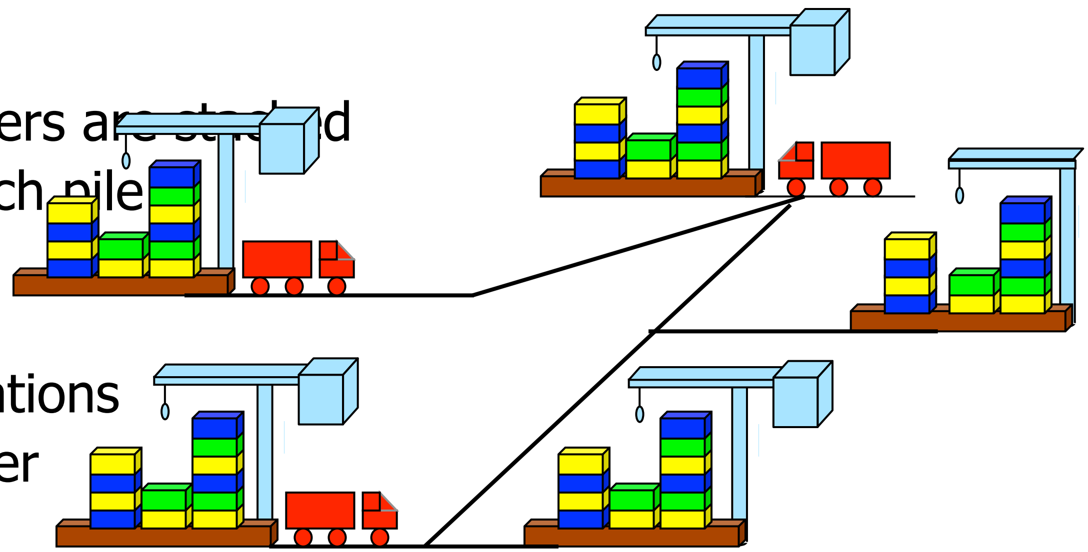
$\{(s_0, \text{take}),$
$(s_1, \text{move1}),$
$(s_3, \text{load}),$
$(s_4, \text{move2})\}$



The Dock Worker Robots (DWR) domain

World Model

- **Locations**: l1, l2, …
- **Containers**: c1, c2, …
  - can be stacked in piles, loaded onto robots, or held by cranes
- **Piles**: p1, p2, …
  - fixed areas where containers are stacked
  - pallet at the bottom of each pile
- **Robot carts**: r1, r2, …
  - can move to adjacent locations
  - carry at most one container
- **Cranes**: k1, k2, …
  - each belongs to a single location
  - move containers between piles and robots
  - if there is a pile at a location, there must also be a crane there



How to construct a plan?

- Search through the list of possible actions (and parameters) until you find a valid sequence that changes the world state in a desired way

- Research issues:
  - Representing world state
  - Making the search fast/tractable
  - Choosing good order for selecting operators
  - Handling conflicting subgoals
- We'll discuss this in more detail when we talk about motion planning.

STRIPS: Means-ends analysis

"Go to Stanford AI Lab"

INITIAL STATE:	Tampa, Florida (0,0)
GOAL STATE:	Stanford, California (1000,200)
Difference:	1020 miles

Use difference to determine which operator is relevant in a situation

States are represented by sets of propositions and a truth value

Difference Table

Distance (difference)	**Mode of transportation (OPERATOR)**
d<=200 miles	FLY
100<d<200	TRAIN
d<=100	DRIVE
d<1	WALK

mode=difference_table(INITIAL STATE, GOAL STATE, difference)

1. Look up what to do: FLY
2. Not at Stanford, so repeat
3. Look up what to do: DRIVE

Preconditions

difference	OPERATOR	PRECONDITIONS
d<=200 miles	FLY	
100<d<200	TRAIN	
d<=100	DRIVE (rental)	at airport
	DRIVE (personal car)	at home
d<1	WALK	

In order for the operator to be applied the precondition must be true.

Updating World Model

distance	OPERATOR	PRECONDITIONS	ADD-LIST	DELETE-LIST
d<=200 miles	FLY		at city Y at airport	at city X
100<d<200	TRAIN		at city Y at train station	at city X
d<=100	DRIVE (rental)	at airport		
	DRIVE (personal)	at home		
d<1	WALK			

STRIPS Summary

- Designer must set up
  - World model representation
  - Difference table with operators, preconditions, add & delete lists
  - Difference evaluator
- Pros
  - Relatively easy to define new domains
  - Handles many domains quite well (blocks-world, transportation planning)
- Cons
  - ?

STRIPS Summary

- Assumes ***closed world***
  - Closed world: world model contains everything needed for robot (implication is that it doesn't change)
  - Open world: world is dynamic and world model may not be complete
- Suffers from ***frame problem***
  - How to represent effects of actions without explicitly having to enumerate intuitively obvious side effects
  - Representation grows too large to reasonably operate over
- Simplistic representation (doesn't handle planning about resources, metacontrol, uncertainty)

Shakey Video

- https://www.youtube.com/watch?v=qXdn6ynwpiI

Docking using Deliberation

- How does the robot find home base and dock using a planner and deliberative architecture?

Summary

- Deliberative control architectures maintain a world model and lookahead to possible future states.
- STRIPS/Shakey is one of the earliest examples of coupling a robot with a planner.
- Main problem with deliberative planners is that dynamic environments quickly render their plans obsolete
- In practice, deliberative planners are usually coupled with a lower layer of reactive control primitives within a hybrid architecture.

Control Architecture Types

- Deliberative control
- Reactive control
- Hybrid control
- Behavior-based control

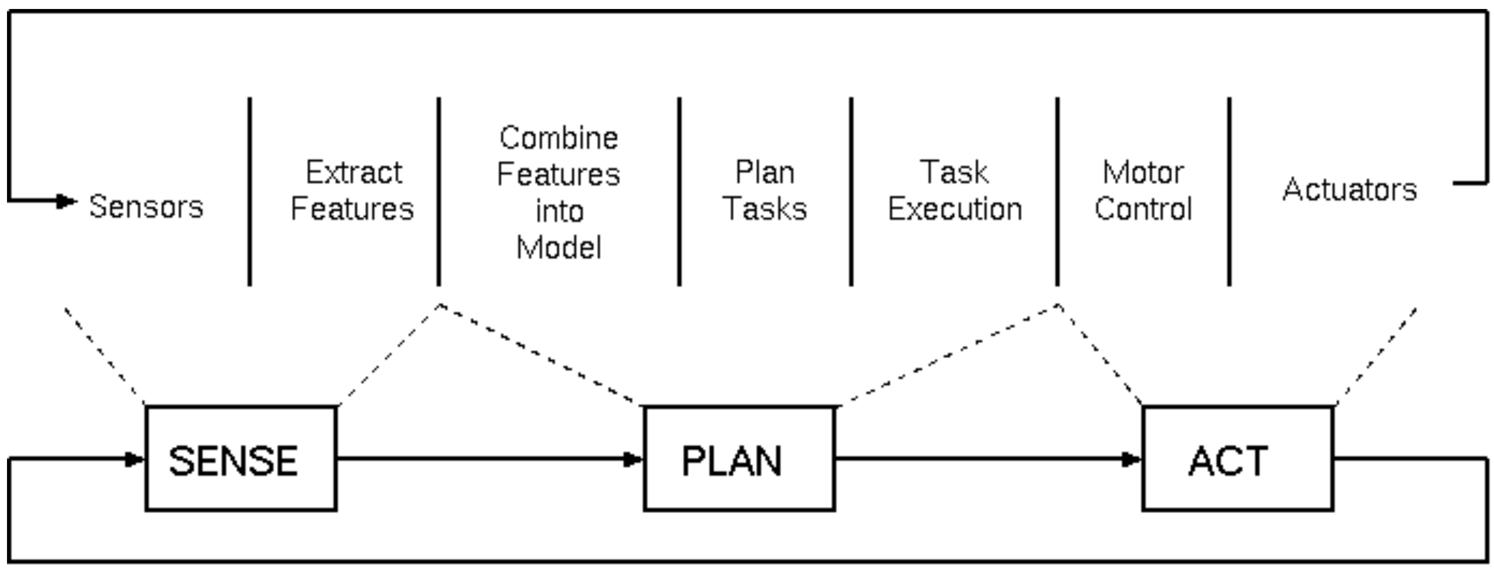
Reactive Architecture

- No maps, no state
- No look ahead
- No planner, no need for fancy search techniques
- Biologically inspired by S-R behaviors in animals

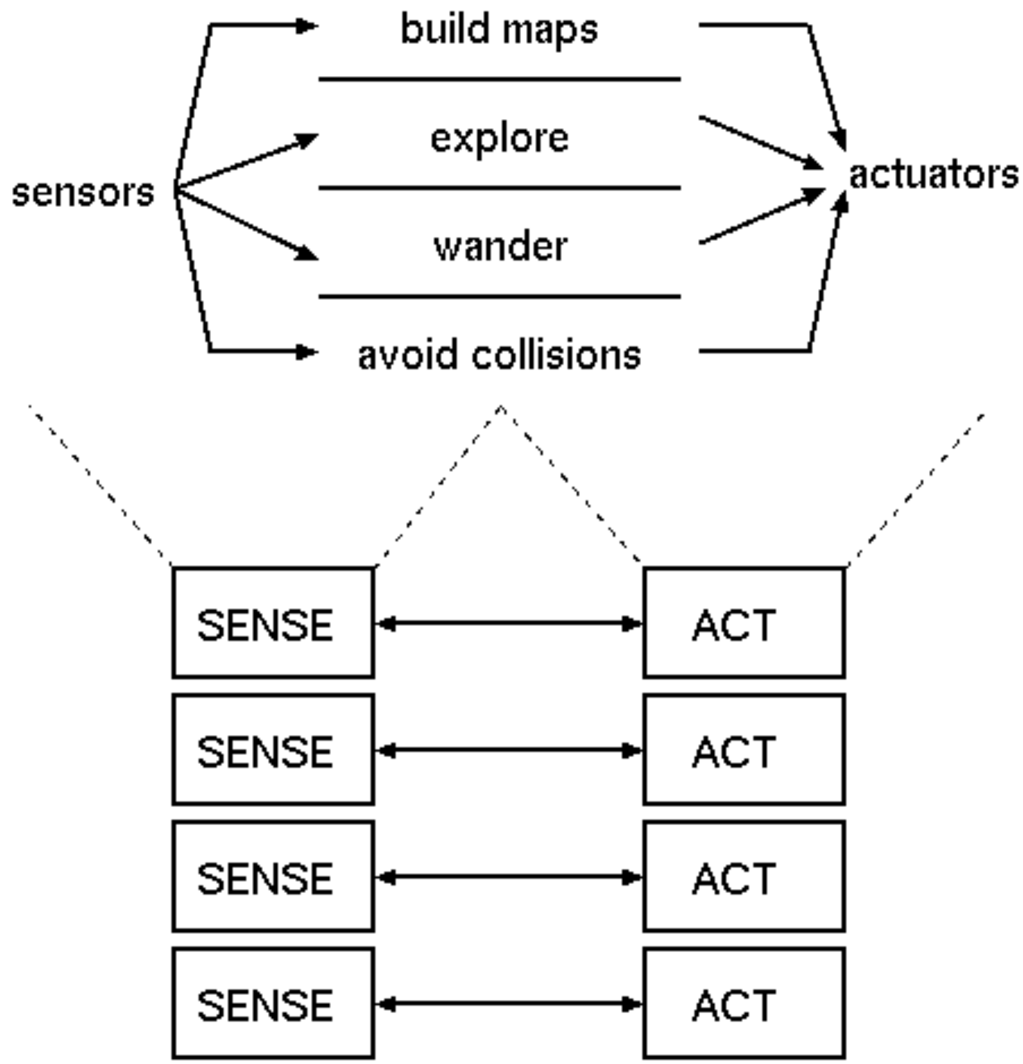
Animal Behavior

- Some robots model the form of animals (Sony Aibo).
- Many roboticists have been inspired to mimic and model animal behaviors.
  - fleeing
  - foraging
  - taxes (movement towards a particular orientation)
  - homing
- Reactive architectures have been used to model many types of animal behaviors.

Deliberative Architectures are "Horizontal"

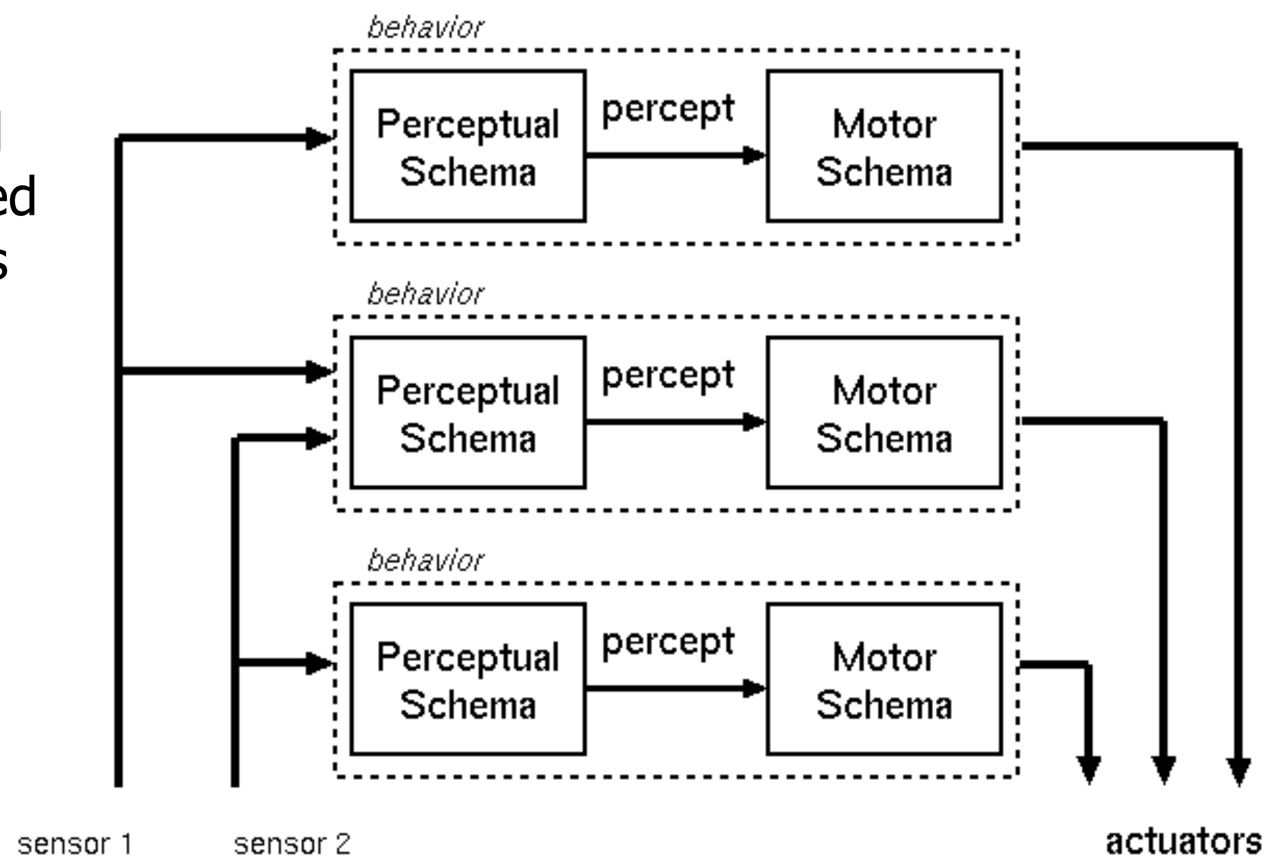


More Biological is "Vertical"



Sensing is Behavior-Specific or Local

No central world model maintained across behaviors

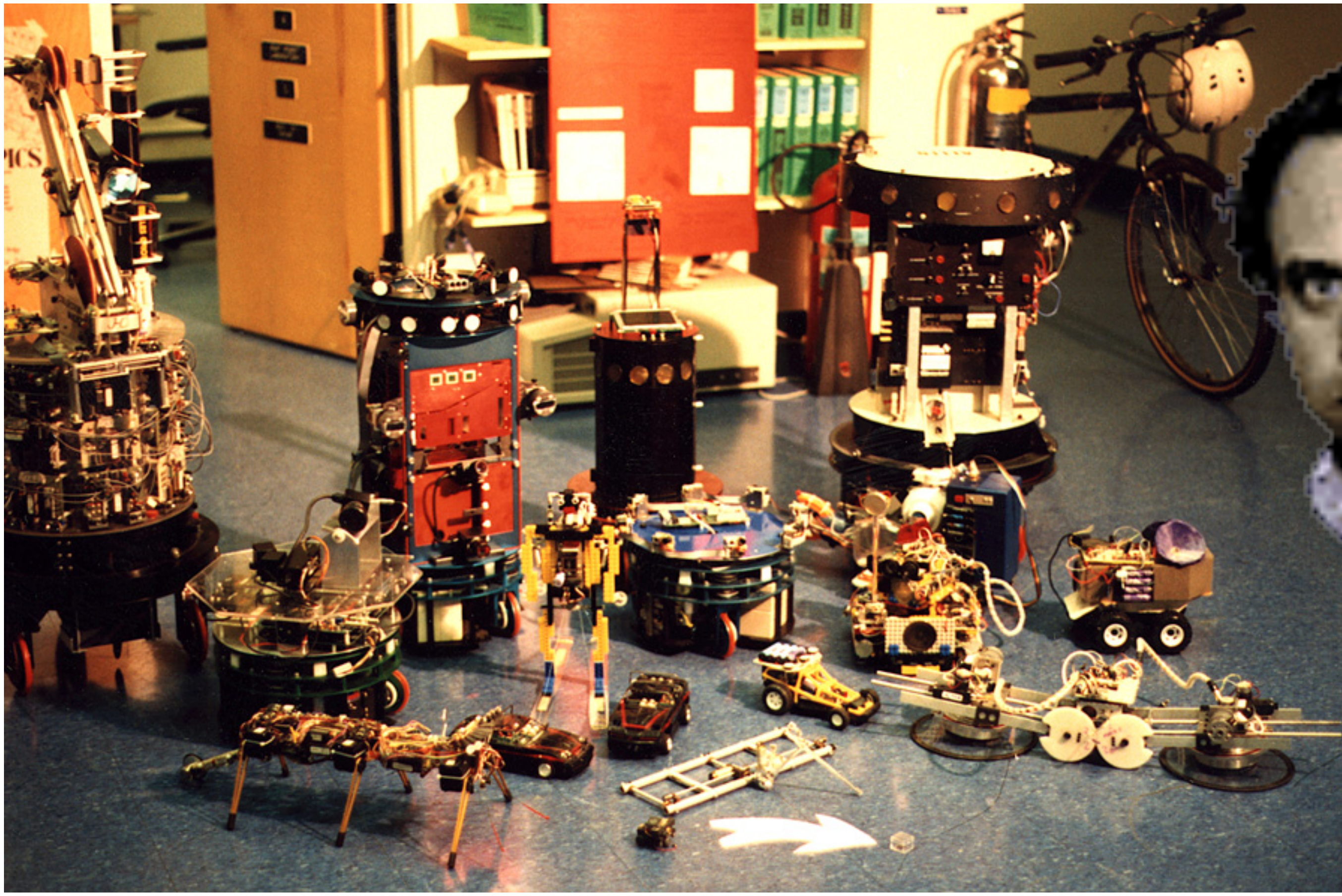


Reactive

- 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; the main difference is in how the behaviors are combined.

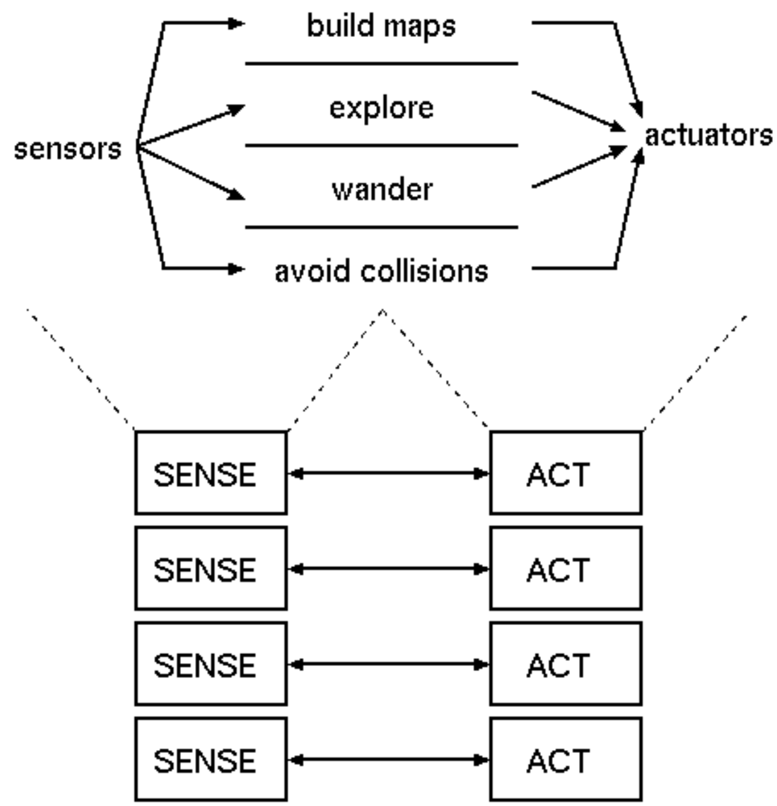
Subsumption (Brooks, MIT)



http://www.youtube.com/watch?v=K2xUHYFcYKI

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.
- Architecture should be *taskable*: accomplished by a higher level turning on/ off lower layers

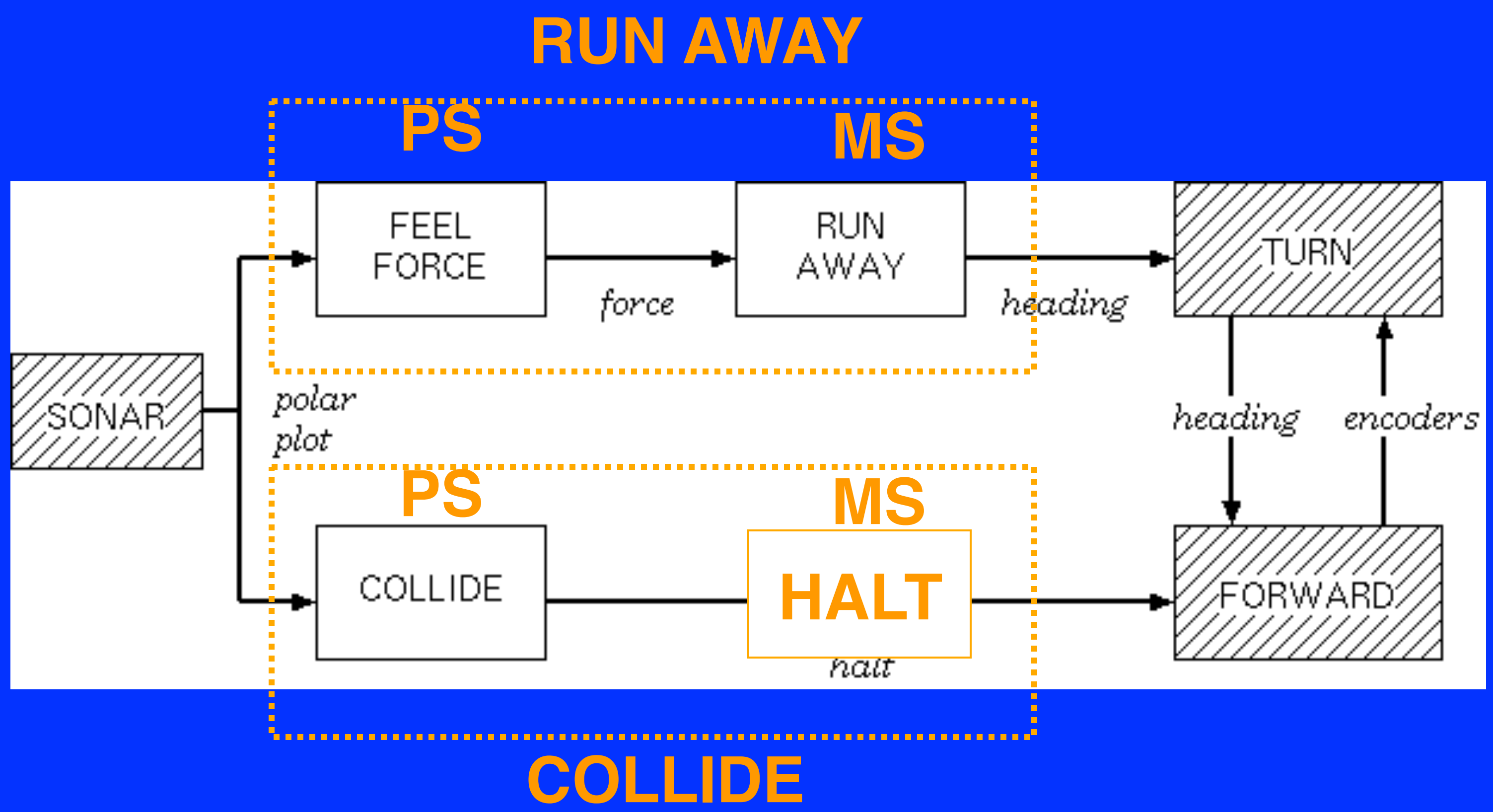


Level 0: Runaway

follow-corridor 2

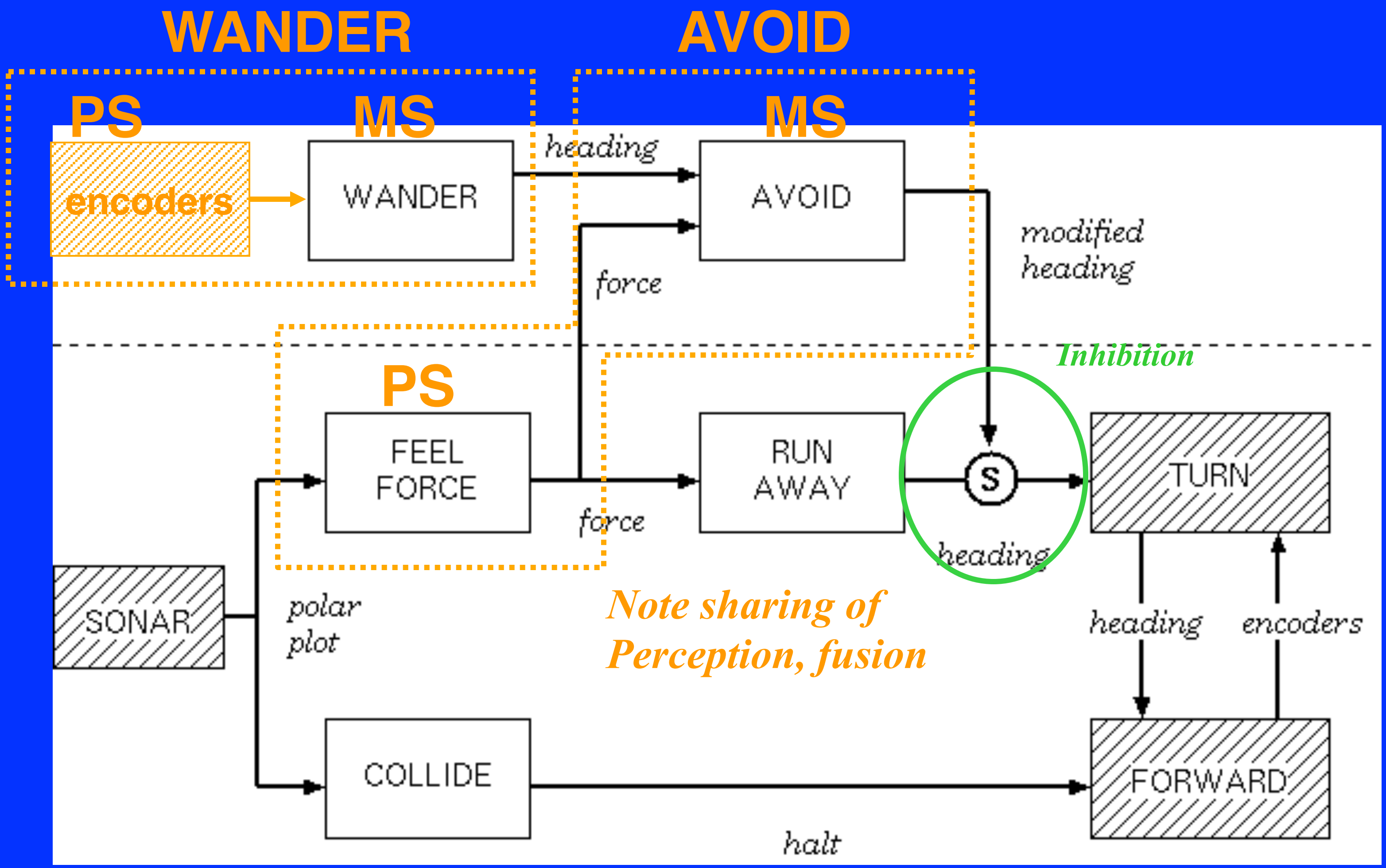
wander 1

runaway 0



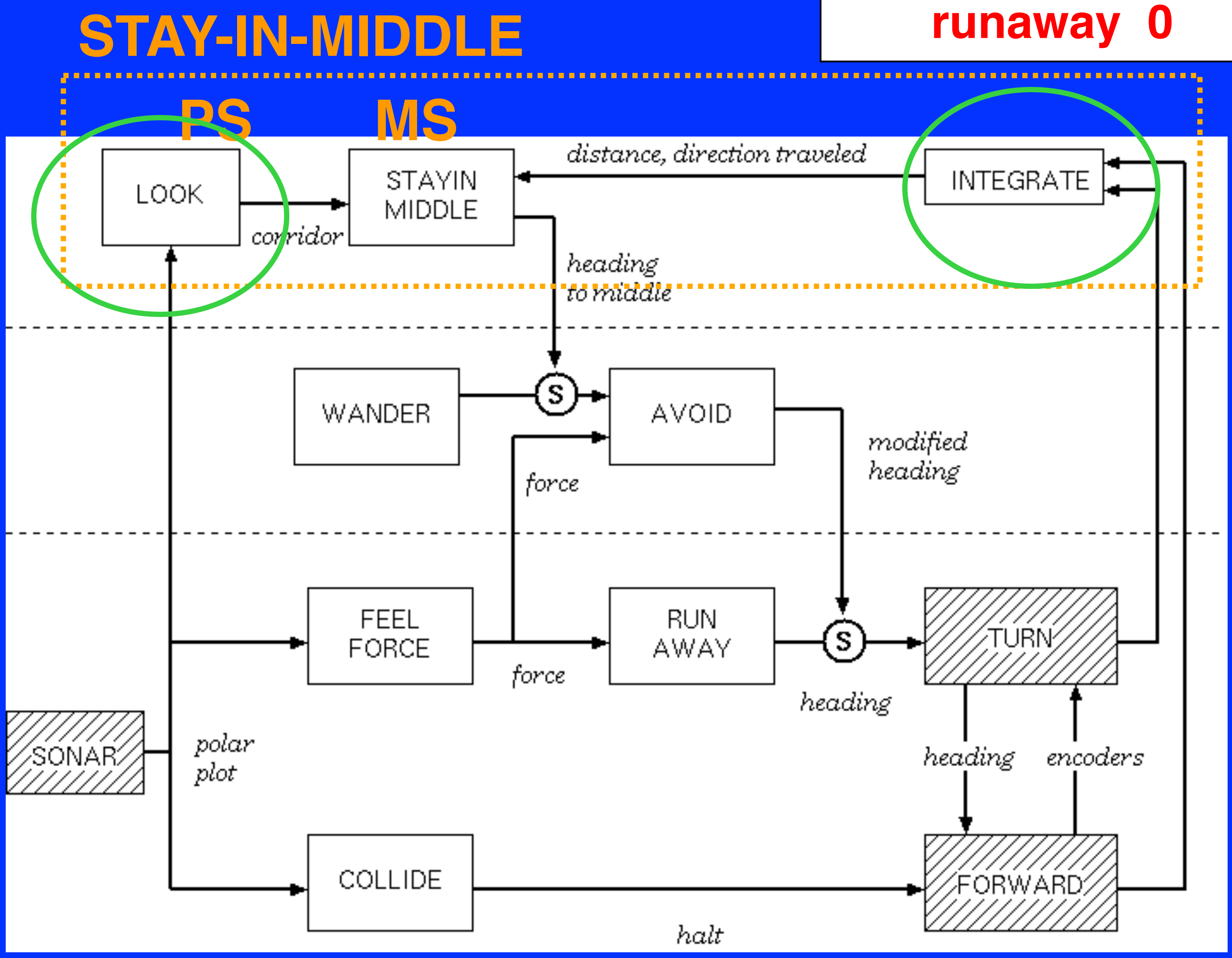
Level 1: Wander

follow-corridor 2
wander 1
runaway 0



Level 2: Follow-Corridors

follow-corridor 2
wander 1
runaway 0



Subsumption Summary

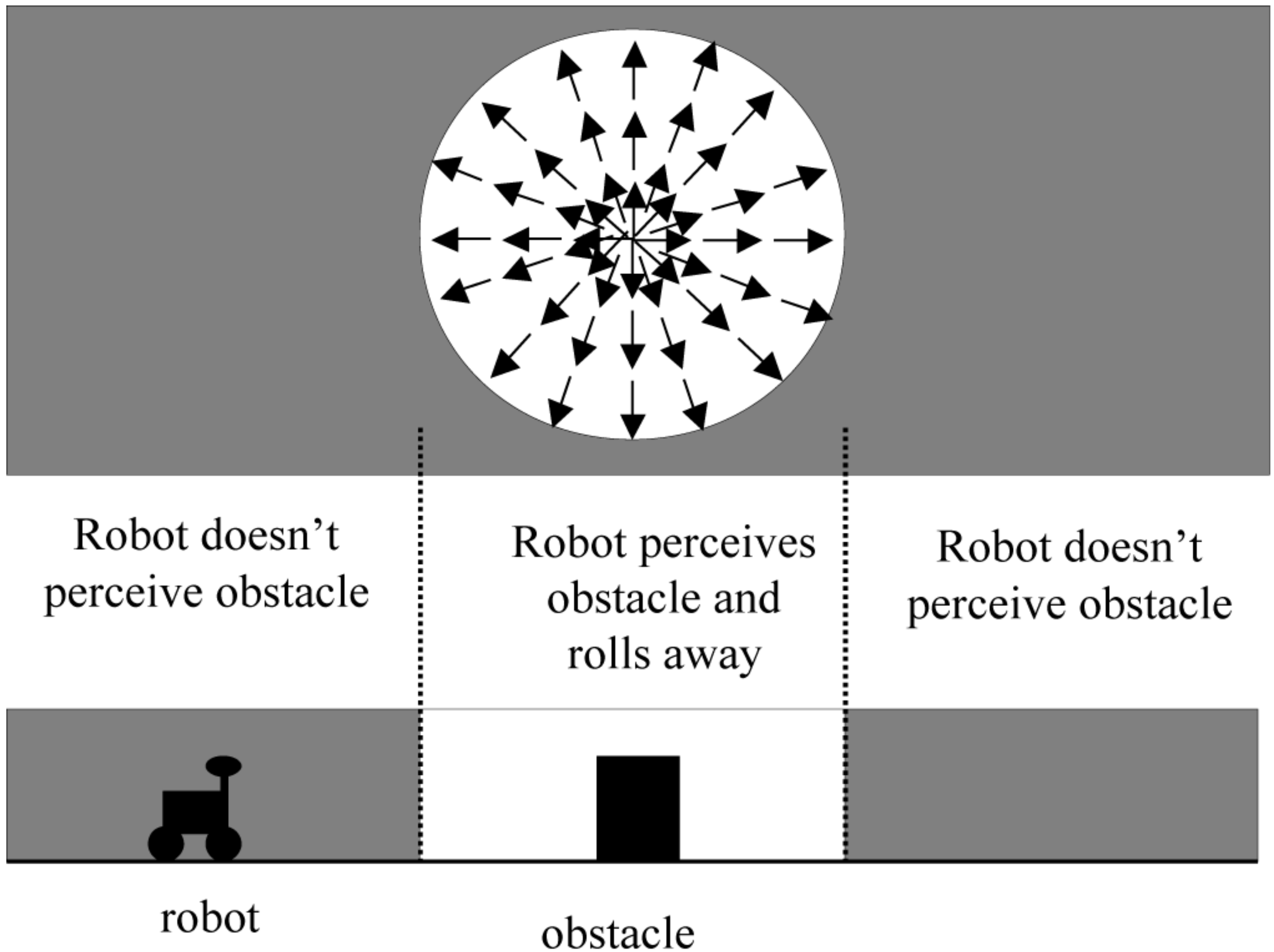
- Many modules operating concurrently at different layers of competence.
- Modules from higher layers of competence can inhibit or suppress other lower level modules.
- Higher level modules can be added to the system without removing or modifying lower-level modules.
- No single world model is maintained; each module can draw from the outputs of different sensors and modules.

Potential Fields: R. Arkin (G. Tech)

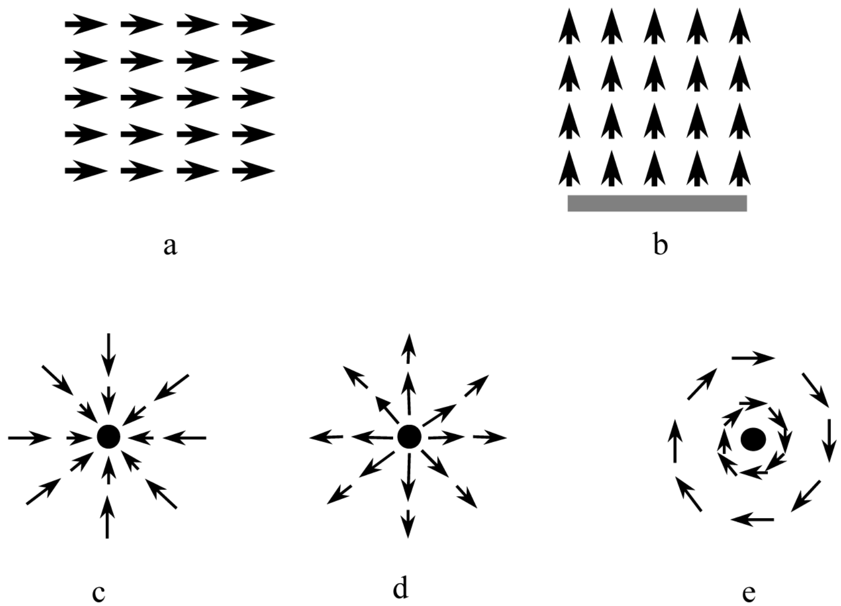
Potential Fields Philosophy

- 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

Run Away via Repulsion

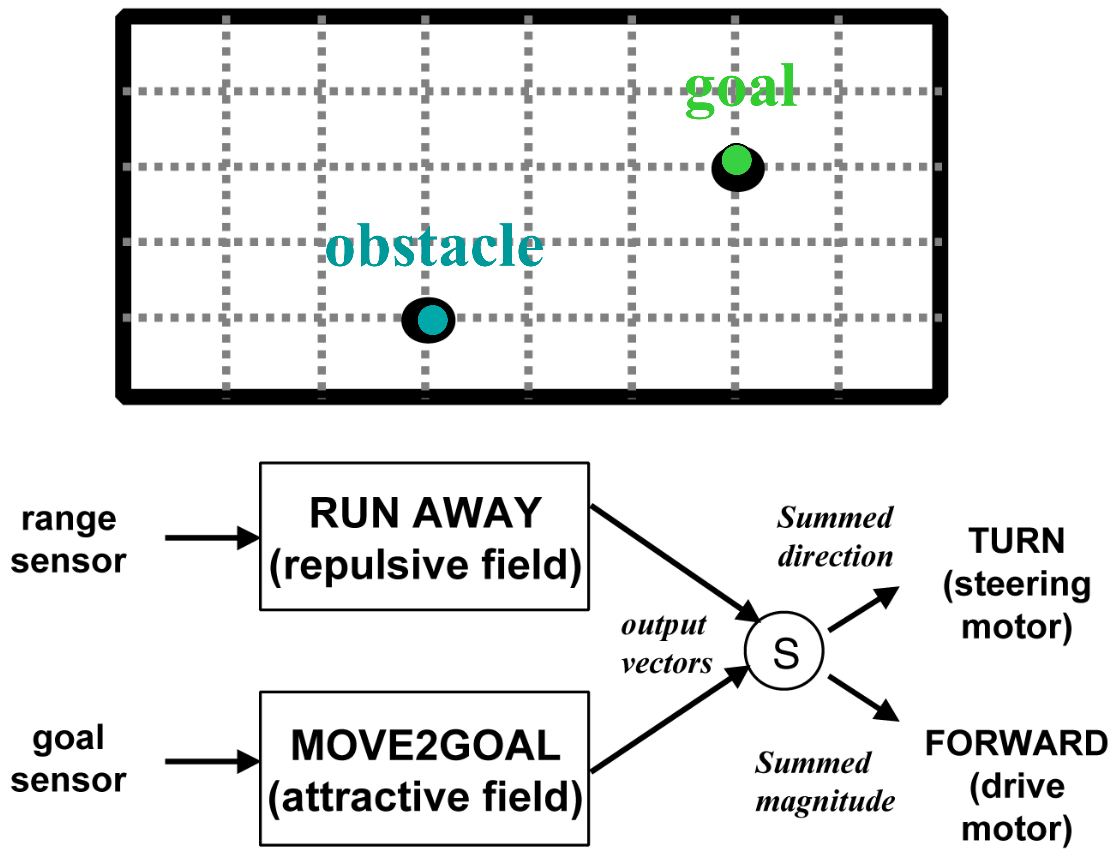


5 Primitive Potential Fields

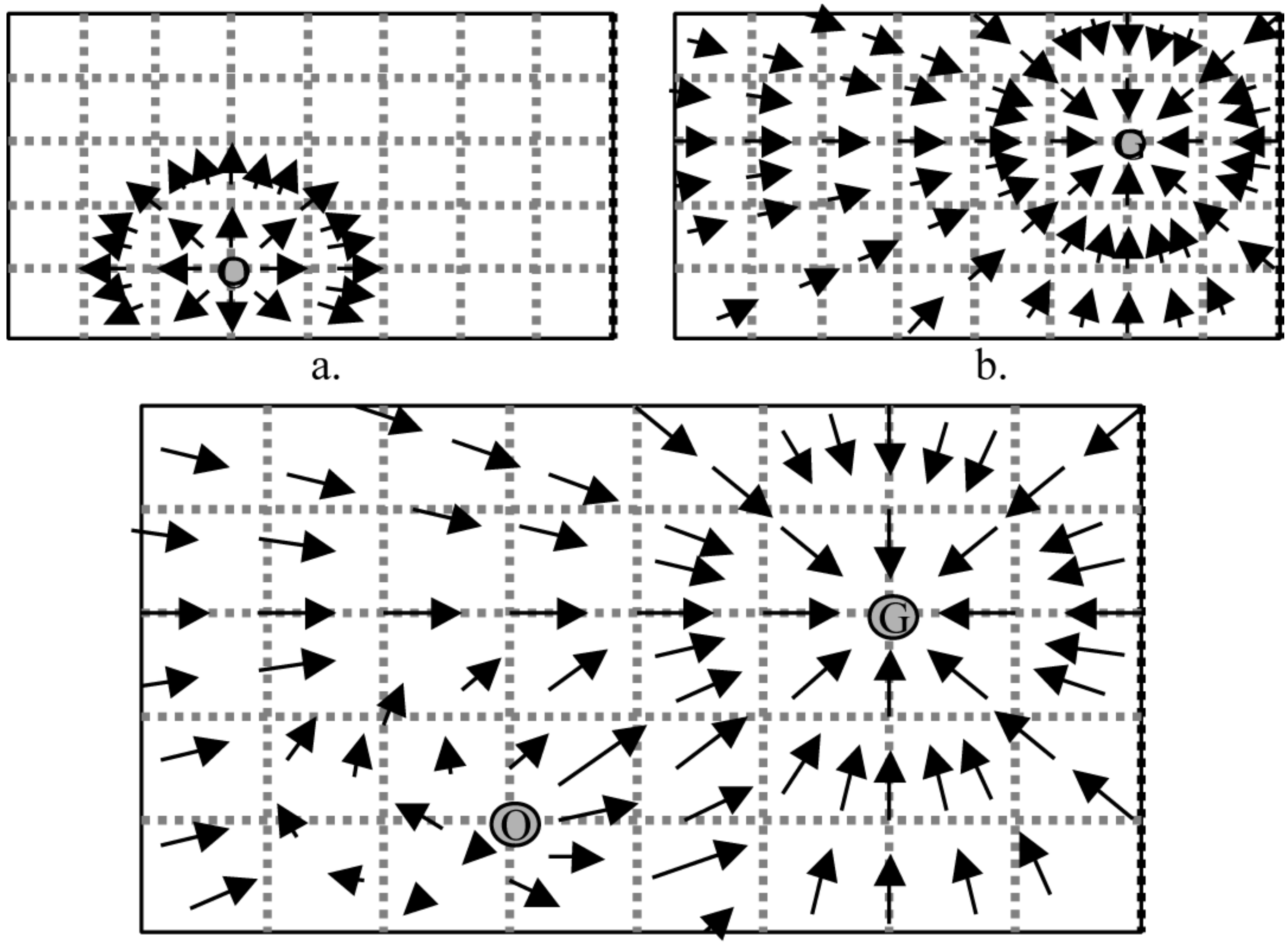


- Uniform
  - Move in a particular direction, corridor following
- Perpendicular
  - Corridor following
- Repulsion
  - Runaway (obstacle avoidance)
- Attraction
  - Move to goal
- Tangential
  - Move through door, docking

Combining Fields for Emergent Behavior



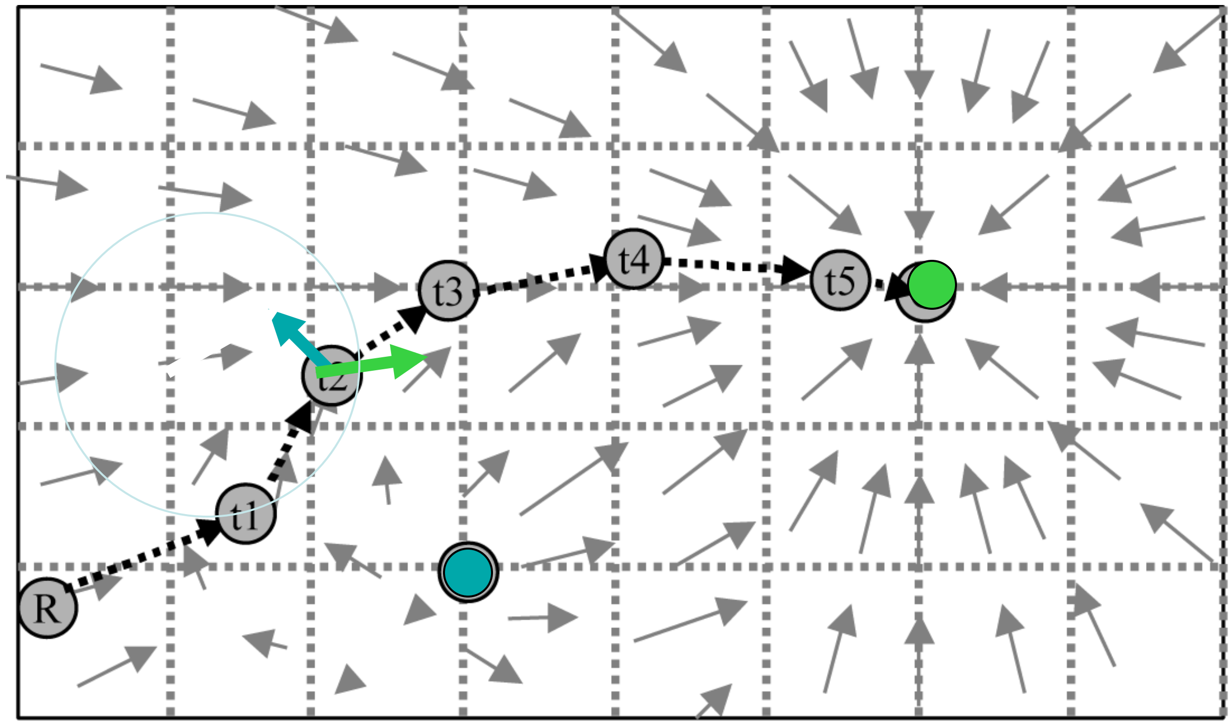
Fields and Their Combination



a.   b.   c.

Path Taken

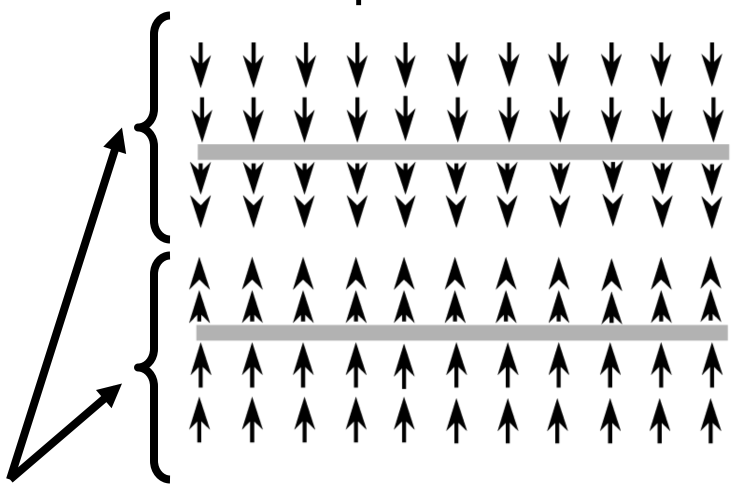
Robot only feels vectors for this point when it (if) reaches that point



- 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

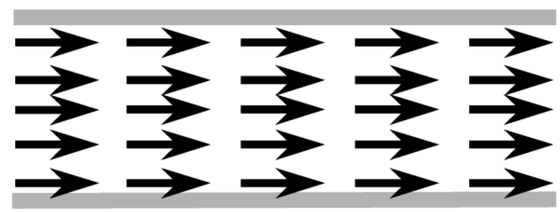
Example: follow-corridor or follow-sidewalk

Perpendicular



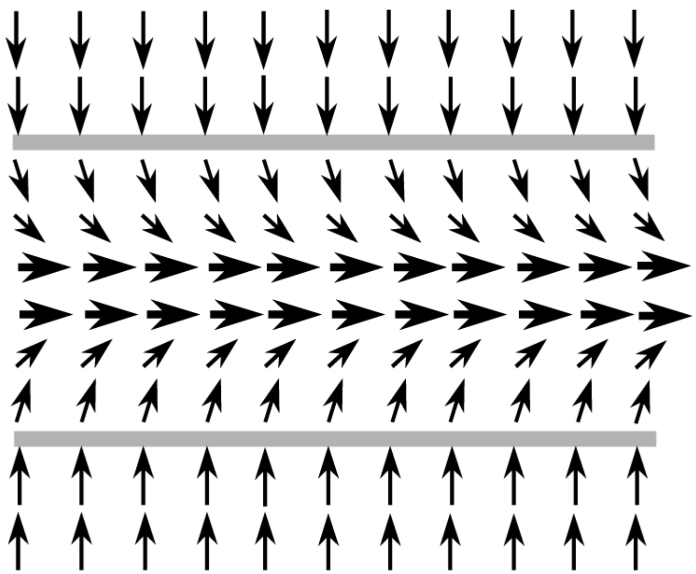
a

Uniform



b

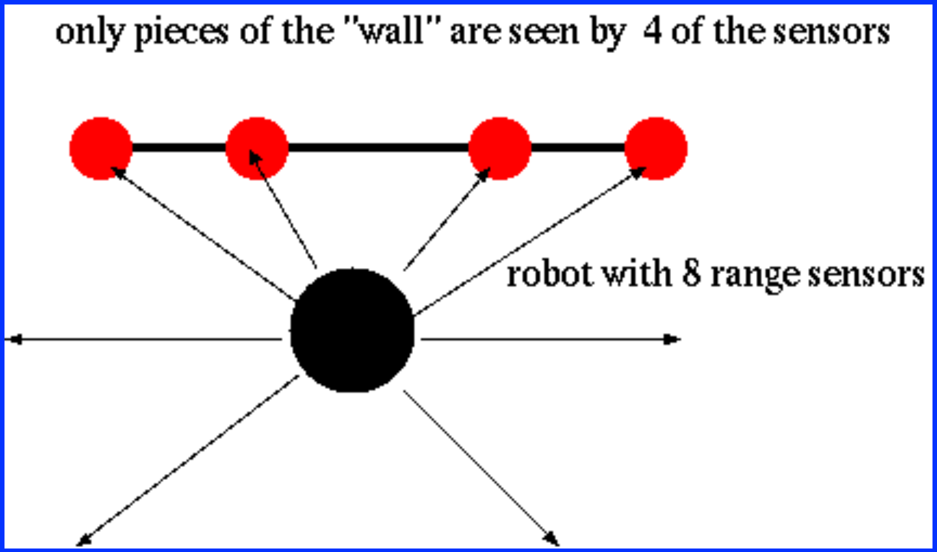
Note that a magnitude profile can be used to modify the vectors at different points



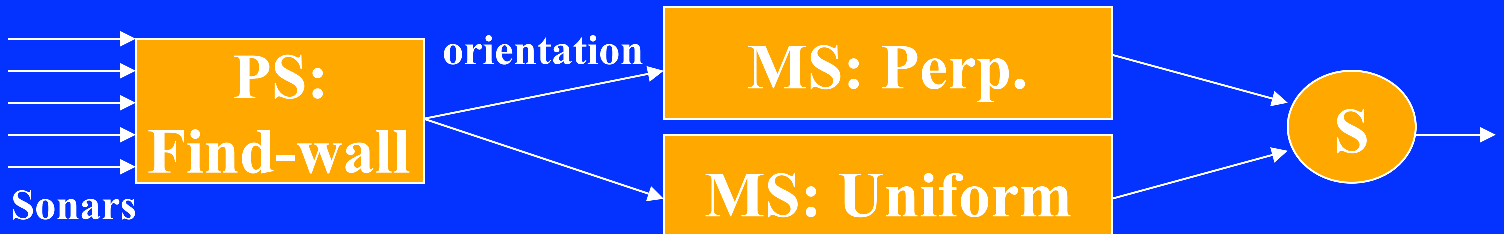
c

Combined

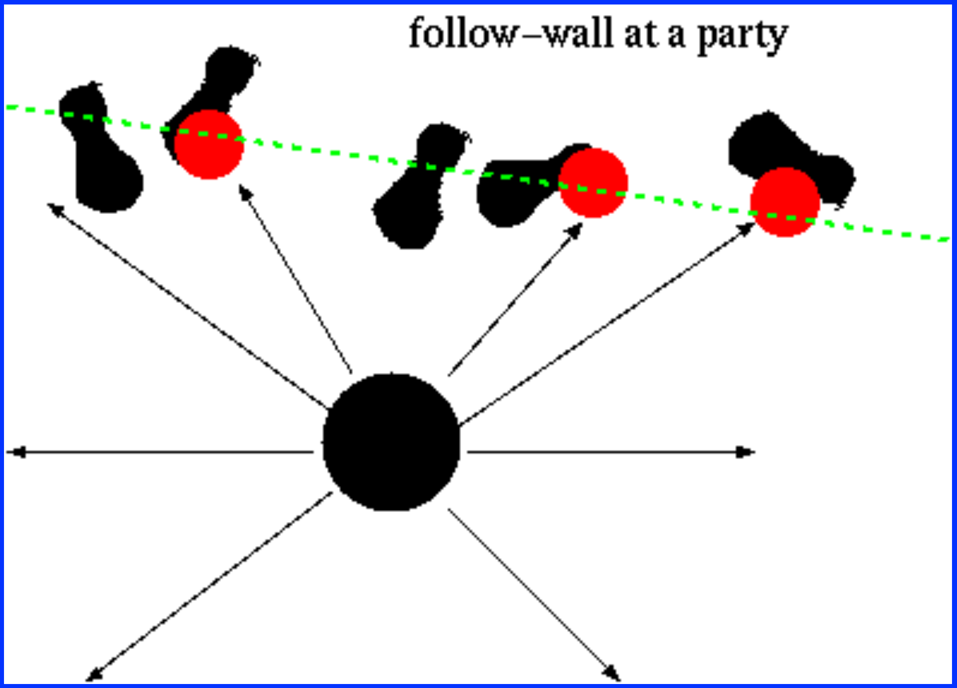
But how does the robot see a wall without reasoning or intermediate representations?



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

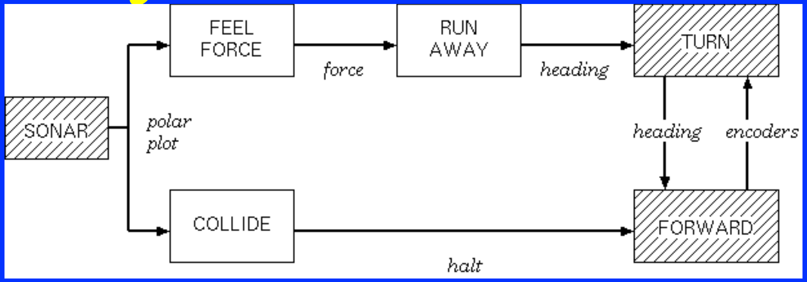


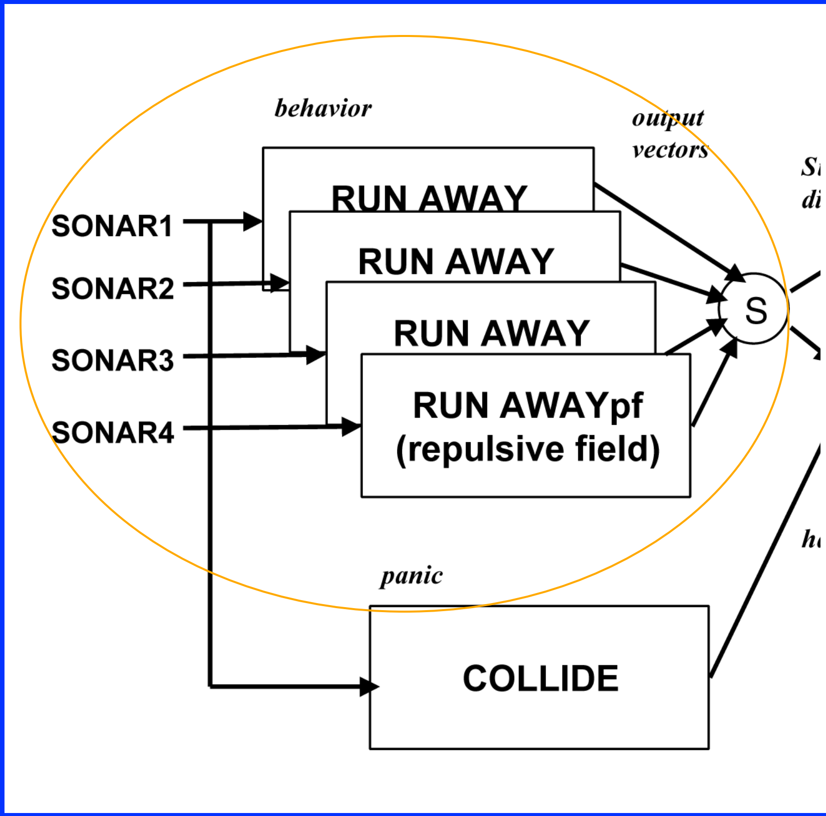
OK, But why isn’t that a representation of a wall?



- It’s not really *reasoning* that it’s a wall, rather it is reacting to the stimulus which happens to be smoothed (common in neighboring neurons)

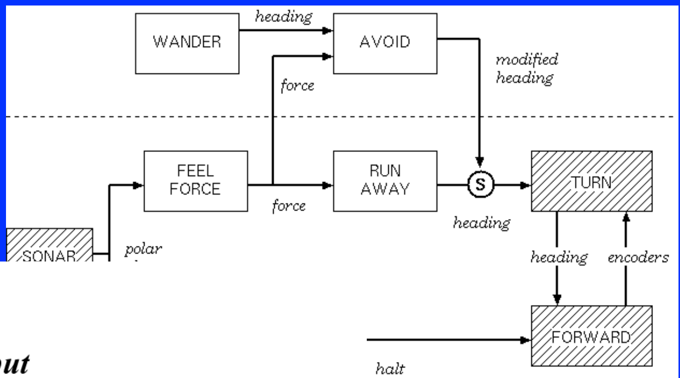
Level 0: Runaway

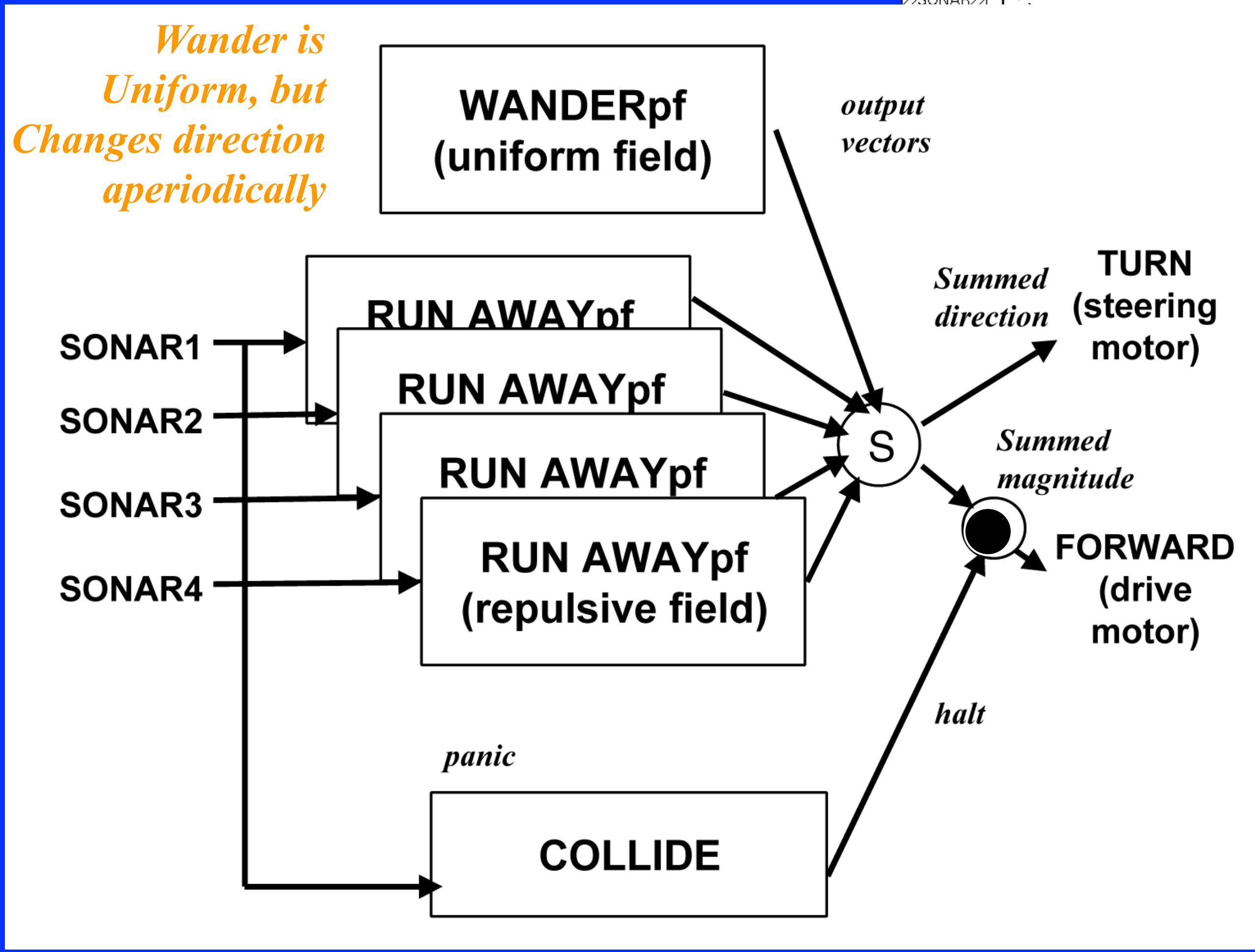




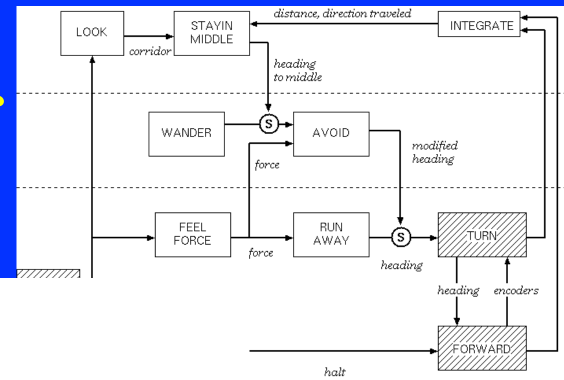
Note: multiple instances of a behavior vs. 1:
Could just have 1
Instance of RUN AWAY,
Which picks nearest reading;
Doesn't matter, but this
Allows addition of another
Sonar without changing the
RUN AWAY behavior

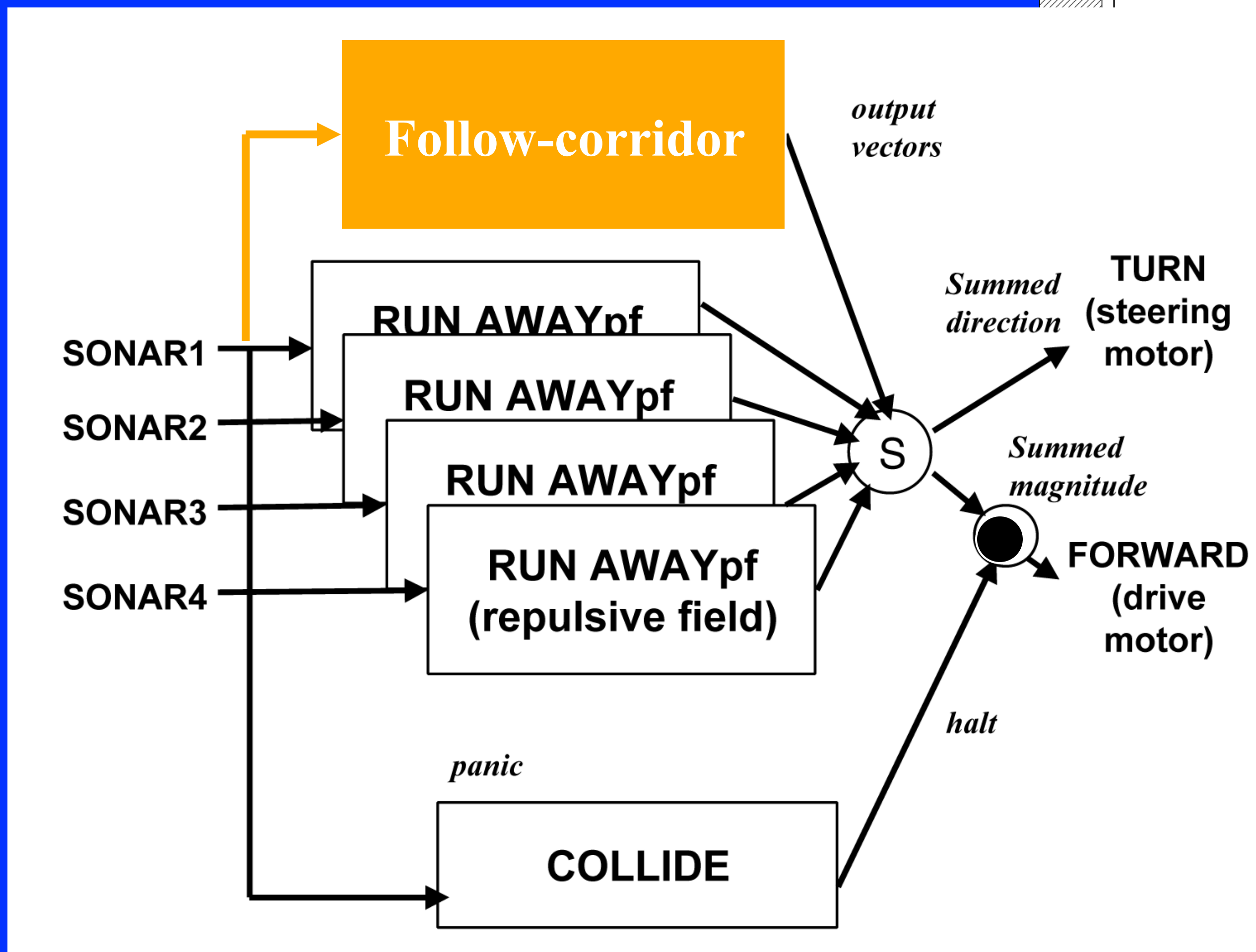
Level 1: Wander





Level 2: Follow Corridor





Potential Fields Summary

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

Summary

- Reactive Paradigm: SA, sensing is local
  - Solves the *Open World problem* by emulating biology
  - Eliminates the *frame problem* by not using any global or persistent representation
  - Perception is direct, ego-centric, and distributed
- Two architectural styles are: *subsumption* and *pfields*
- Behaviors in pfield methodologies are a tight coupling of sensing to acting; modules are mapped to schemas conceptually
- Potential fields and subsumption are logically equivalent but different implementations
- Pfield problems include
  - local minima (ways around this)
  - jerky motion
  - bit of an art