behavior-based systems.

- control
- behavior-based systems
- expressing behaviors
- behavioral encoding
- representations
- example: Toto
- behavior coordination
- emergent behavior

models.

we like to make a distinction between

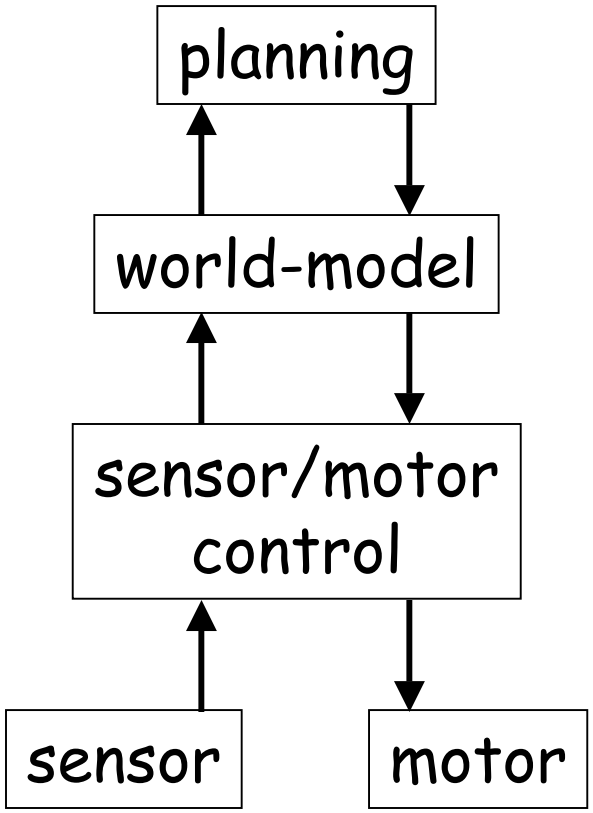
- classic "model-based" AI
  - *symbolic representations*
- neo "behavior-based" AI
  - *numeric representations*

classic models.

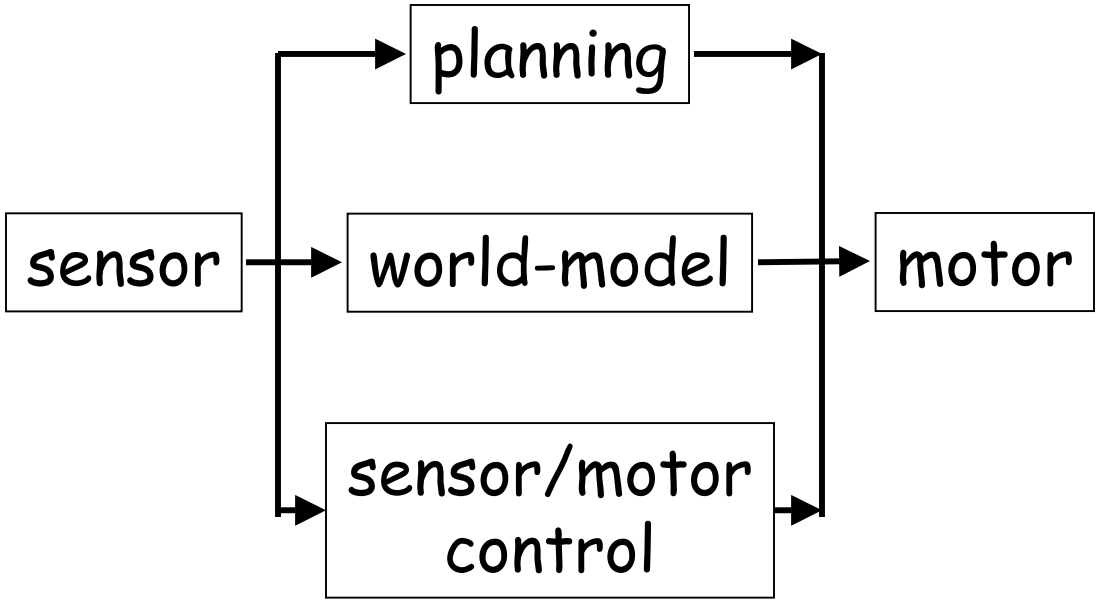
- deliberative… SPA (sense,plan,act)
  - *functional decomposition*
    - systems consist of sequential modules achieving independent functions
      - sense world
      - generate plan
      - translate plan into actions
- reactive architectures
  - *task-oriented decomposition*
    - systems consist of concurrently executed modules achieving specific tasks
      - avoid obstacle
      - follow wall

two orthogonal flows.

vertical



horizontal



distinctions.

- key issues that distinguish architectures
  - *time scale*
  - *looking ahead*
  - *modularity*
    - the way in which the architecture decomposes into components

behavior-based systems.

- BBS
- behaviors are the underlying module of the system
- behavioral decomposition
  - *systems consist of sequential modules achieving independent functions*

robotic behavior.

- generate a motor response from a given perceptual stimulus
- basis in biological studies
  - *serves as inspiration for design*

behavior vs action.

- behavior
  - *based on dynamic process*
    - operating in parallel
    - lack of a central control
    - fast couplings between sensors and motors
  - *exploiting emergence*
    - side-effects from combined processes
    - using properties of the environment
  - *reactive*
- action
  - *discrete in time*
    - well-defined start and end points
    - allows pre- and post-conditions
  - *avoidance of side-effects*
    - only one action or few actions at a time
    - conflicts are undesired and avoided
  - *deliberative*

behavior based systems… are reactive systems.

- behaviors serve as building blocks for actions
- abstract representation avoided
- often modeled after animal behaviors
- inherently modular

stimuli.

- presence of stimulus is necessary but not sufficient in behavior-based robot
- stimulus must reach threshold value before response is generated

representation.

- behaviors can be represented/stored in a network, with relationships between them
- strength multiplier, or gain, can turn off behaviors or increase response

properties.

- feedback controllers
- achieve specific tasks/goals
  - *avoid others, find friend, go home*
- typically execute concurrently
- can store state and be used to construct world models/representations
- can directly connect sensors to effectors
- can take inputs from other behaviors and send outputs to other behaviors ($\Rightarrow$ networks)

properties, 2.

- typically higher-level than actions (go home, not turn left 45º)
- typically closed loop, but extended in time
- when assembled into distributed representations, behaviors can be used to look ahead but at a time-scale comparable with the rest of the system

key properties.

- ability to act in real time
- ability to use representations to generate efficient (not only reactive) behavior
- ability to use a uniform structure and representation throughout the system (so no intermediate layer)

key challenge.

- how can representation be effectively *distributed* over the behavior structure?
  - *time scale must be similar to that of real-time components of the system*
  - *representation must use same underlying behavior structure for all components of the system*

challenges, 2.

- some components may be reactive
- not every component is involved with representational computation
- some systems use a simple representation
- as long as the basis is in behaviors and not rules, the system is a BBS

what are behaviors?

- *behavior:* anything observable that the system/robot does
- how do we distinguish internal behaviors (components of a BBS) and externally observable behaviors?
- should we distinguish?

external vs internal behavior

- reactive robots display desired external behaviors
  - *avoiding*
  - *collecting cans*
  - *walking*
- but controller consists of a collection of rules, possibly in layers
- BBS actually consist and are programmed in the behaviors, which are higher granularity, extended in time, capable of representation

expressing behaviors.

- behaviors can be expressed with various representations
- when a control system is being designed, the task is broken down into desired external behaviors
- those can be expressed with
  - *functional notation*
  - *stimulus response (SR) diagrams*
  - *finite state machines/automata (FSA)*
  - *schema*

design paradigms.

- ethological guided/constrained
- situated activity
- experimentally driven

functional notation.

- mathematical model:
  - *represented as triples (S,R,β)*

    S = stimulus

    R = range of response

    β = behavioral mapping between S and R
- easily convert to functional languages like LISP

functional example.

```
coordinate-behaviors [
   move-to-classroom (detect-classroom-location),
   avoid-objects (detect-objects),
   dodge-students ( detect-students ),
   stay-to-right-on-path ( detect-path ),
   defer-to-elders ( detect-elders )
] = motor-response
```

FSA diagrams.

- states and transitions are most easily encoded in finite state automata and drawn as finite state diagrams
- states of the diagram can also be called behaviors
- diagrams show sequences of behavior transitions

formal methods.

- used to verify intentions of designer (not the same as correctness of controller)
- enable automatic generation of code
- use a common language
- support formal analysis
- support high-level programming
- example: situated automata

situated automata.

- formalism for specifying FSA's that are *situated* [Kaelbling & Rosenschein, 1991]
- task described in high-level logic expressions, as a set of goals and a set of operators that achieve (`ach`) and maintain (`maint`) the goals
- once defined, tasks can be compiled into circuits (using special purpose languages), which are reactive

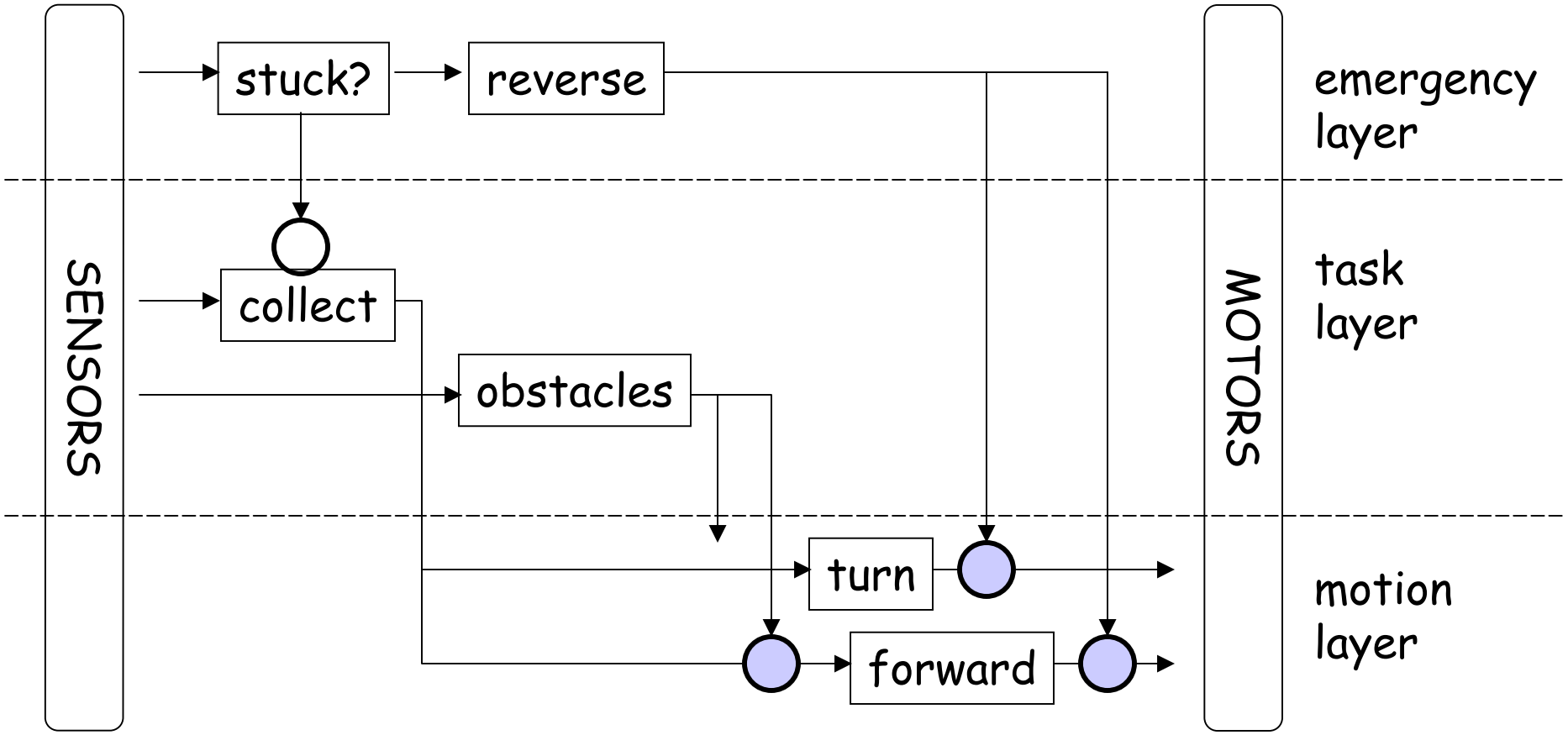
situated automata example.

```
(defgoalr (ach in-classroom)
   (if (not startOup)
      (maint (and (maint move-to-classroom)
                  (maint avoid-objects)
                  (maint dodge-students)
                  (maint stay-to-right-on-path)
                  (maint defer-to-elders)))))
```

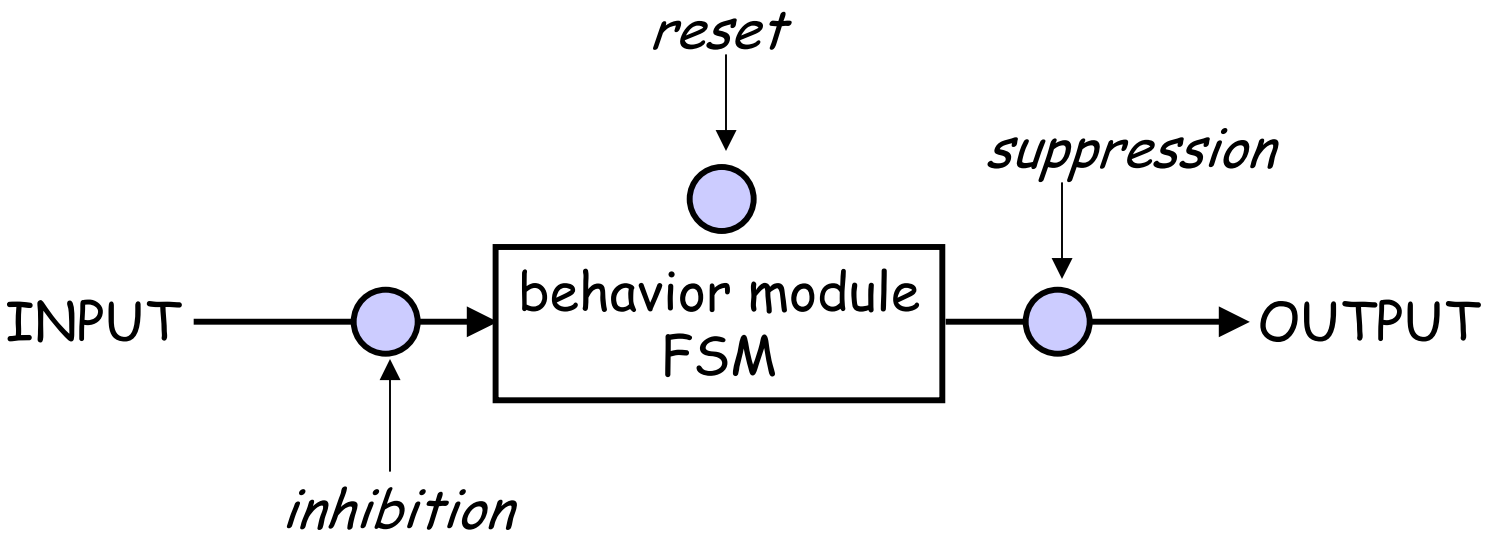
subsumption architecture.

- Rodney Brooks, 1986
- MIT AI lab
- reactive elements
- behavior-based elements
- layered approach based on *levels of competence*

subsumption architecture, 2.



expressing behaviors: augmented finite state machine (AFSM).



behavioral encoding.

- behavioral response in physical space has a strength and an orientation
- expressed as (S,R,β)
- S = stimulus, necessary but not sufficient condition to evoke a response (R); internal state can also be used
- β = behavioral mapping categories
  - *null*
  - *discrete*
  - *continuous*

discrete encoding.

- expressed as a finite set of situation-response pairs/mappings
- mappings often include rule-based form IF-THEN
- examples:
  - *Gapps* *[Kaelbling & Rosenschein]*
  - *subsumption language* *[Brooks]*

continuous encoding.

- instead of discretizing the input and output, a continuous mathematical function describes the input-output mapping
- can be simple, time-varying, harmonic
- examples:
  - *potential field*
  - *schema*
- problems with local minima, maxima, oscillatory behavior

motor schemas.

- type of behavior encoding
- based on schema theory
- Ron Arbib, Georgia Tech
- provide large grain modularity
- distributed, concurrent schemas used
- based on neuroscience and cognitive science
- represented as vector fields
- decomposed into “assemblages” by fusion, not competition

assemblages.

- recursively defined aggregations of behaviors or other assemblages
- important abstractions for constructing behavior-based robots

schema representation.

- responses represented in uniform vector format
- combination through cooperative coordination via vector summation
- no predefined schema hierarchy
- arbitration 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 reactions
- conduct simple simulations
- determine perceptual needs to satisfy motor schema inputs
- design specific perpetual algorithms
- integrate/test/evaluate/iterate

strengths and weaknesses.

- strengths
  - *support for parallelism*
  - *run-time flexibility*
  - *timeliness for development*
  - *support for modularity*
- weaknesses
  - *niche targetability*
  - *hardware retargetability*
  - *combination pitfalls (local minima, oscillations)*

representation.

- necessary in order to get benefits of BBS
- BBS representations are
  - *distributed*
  - *matched to time-scale of the system*
- constructed out of basic components of BBS system: behaviors

behavior.

- direct feedback loops/control laws: mapping sensors to effectors
- schemas: sensory or motor
- procedures: any combination
  - *sensory*
  - *motor*
  - *sensory to motor*
  - *representational*

example task: mapping.

- task: create a robot that can
  - *move around safely*
  - *make a map of its environment*
  - *use the map to find paths to specific locations*
- most common and useful mobile robot task
- many applications!

representing a map.

- representation needs to be
  - *distributed*
  - *on a short time-scale*
  - *cannot be a traditional CAD-CAM centralized map*

idea: distribute maps.

- distribute parts of map over different behaviors
- connect parts of the map that are adjacent in the physical world so that they are also adjacent in the map
- result is a network of behaviors representing the map
- map is topological

example: Toto.

- Maja Mataric, MIT now USC
- behavior-based robot
- first BBS robot to have a distributed representation
- control system consisted of a collection of behaviors
- lowest levels responsible for safe movement of robot (avoiding collisions)
- next levels responsible for keeping robot near walls, boundaries

landmarks.

- walls, corridors, messy irregular areas
- robot detected landmarks
- each landmark stored as a behavior
  - *landmark type*
  - *compass heading*
  - *approximate length/size*
- when a new landmark is found, a new behavior is added

landmarks, 2.

- adjacent landmarks connected by communication wires
- result is a topological representation of the environment
- also used for path finding

active map.

- whenever Toto visited a particular landmark, its associated map behavior would become activated
- if no behavior was activated, then the landmark was new, so a new behavior was created
- if an existing behavior was activated, it inhibited all other behaviors
- localization was based on which behavior was active

message passing.

- once Toto had a map, it could find paths from one landmark to another
- the goal behavior/landmark would send messages (send activation) to its neighbors, they would pass it on, etc
- eventually it would reach the current landmark (Toto's current position)

continuous map following.

- resulting string of behaviors is a path (or a plan) to the goal
- Toto did not store a string
- messages were passed continuously
- at each behavior in the map, Toto would decide where to go next
- goal reached one behavior at a time

path optimization.

- at a junction, how did Toto decide where to go?
- path length computed based on landmark size and number of landmarks from goal to current landmark
- Toto chose shortest path

behavior coordination.

- BBS consist of collection of behaviors
- execution must be coordinated in a consistent fashion
- coordination can be
  - *competitive*
  - *cooperative*
  - *combination of the two*

deciding what to do next.

- action-selection problem
- behavior-arbitration problem

behavior arbitration.

- summing up
- average
- winner takes all

competitive coordination.

- perform arbitration (selecting one behavior among a set of candidates)
  - *priority-based: subsumption*
  - *state-based: discrete event systems, Bayesian decision theory*
  - *function-based: spreading of activation action selection*

cooperative coordination.

- perform command fusion (combine outputs of multiple behaviors)
- voting
- fuzzy (formalized voting)
- superposition (linear combinations)
  - *potential fields*
  - *motor schemas*
  - *dynamical systems*

emergent behavior.

- important but not well-understood phenomenon
- often found in behavior-based robotics
- robot behaviors "emerge" from
  - *interactions of rules*
  - *interactions of behaviors*
  - *interactions of either with environment*

distinction.

- coded behavior
  - *in the programming scheme*
- observed behavior
  - *in the eyes of the observer*
  - *emergence*
- there is no one-to-one mapping between the two!

is it magic?

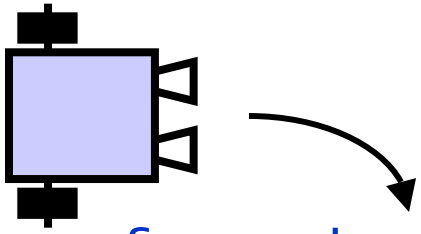
- sum is greater than the parts
- emergent behavior is more than the controller that produces it

interaction and emergence.

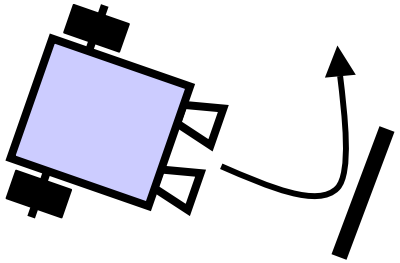
- interactions between rules, behaviors and environment
- source of expressive power for a designer
- i.e., systems can be designed to take advantage of emergent behavior

Example: wall following.

coded behaviors

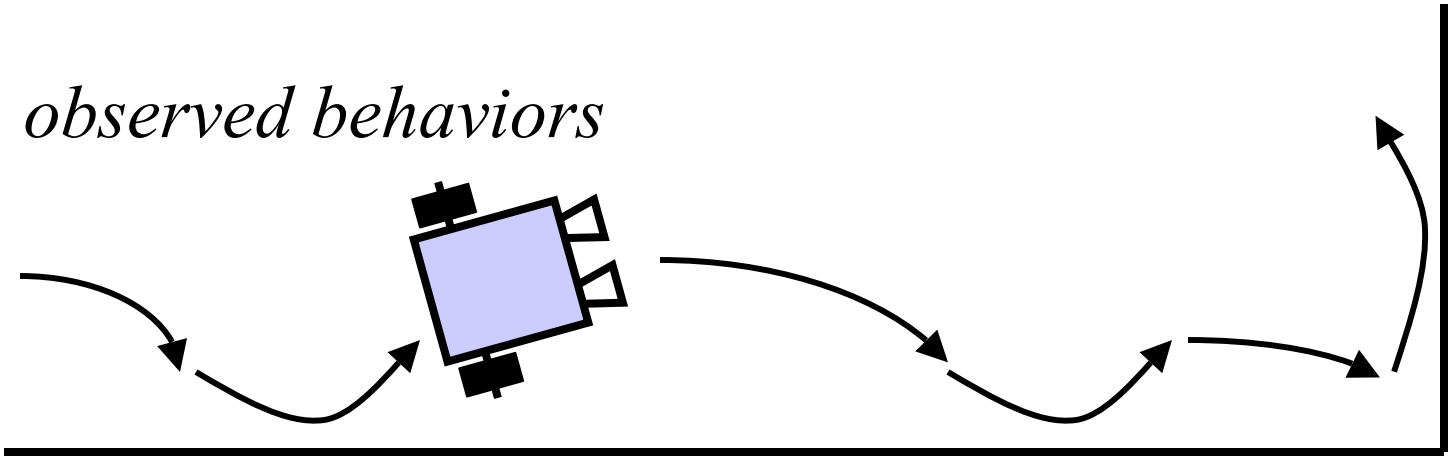


forward motion,
with slight turn right



obstacle
avoidance

observed behaviors



wall
following

example: wall following, 2.

- can be implemented with these rules:
  - *if too far, move closer*
  - *if too close, move away*
  - *otherwise, keep on*
- over time, in an environment with walls, this will result in wall-following
- is this emergent behavior?

emergent wall following.

- it is argued yes because
  - *robot itself is not aware of a wall, it only reacts to distance readings*
  - *concepts of "wall" and "following" are not stored in the robot's controller*
  - *the system is just a collection of rules*

emergent flocking.

- program multiple robots:
  - *don't run into any other robot*
  - *don't get too far from other robots*
  - *keep moving if you can*
- when run in parallel on many robots, the result is flocking

necessary conditions.

- notion of emergence depends on two aspects:
  - *existence of an external observer, to observe and describe the behavior of the system*
  - *access to the internals of the controller itself, to verify that the behavior is not explicitly specified naywhere in the system*

unexpected vs emergent.

- some researchers say the above is not enough for behavior to be emergent, because above is programmed into the system and the "emergence" is a matter of semantics
- so emergence must imply something unexpected, something "surreptitiously discovered" by observing the system

subjective expectations.

- "unexpected" is highly subjective, because it depends on what the observer was expecting
- naïve observers are often surprised!
- informed observers are rarely surprised

observation and emergence.

- once a behavior is observed, it is no longer unexpected
- is new behavior then "predictable"?

formalization.

- look for behaviors that are not apparent at system level (robot's controller) but are apparent at observer's level

execution and emergence.

- so now even simple wall following example given can be called “emergent”
- this means system has to execute in order for behavior to emerge

uncertainty and emergence.

- not difficult to achieve -- environment is uncertain, so exact behavior of a system is very hard to predict!
- if behavior contains novel and rich patterns, then it is “emergent”
- if world were completely predictable, then we’d remove “emergent behaviors” by this definition

emergence is unavoidable.

- some interesting and unexpected behaviors will always emerge in systems that interact with the physical world
- some may be labeled “emergent”
- some are not desirable!

emergent bugs.

- unexpected, emergent behavior that is undesirable
- e.g., oscillations
- try to avoid them, but still want to exploit desirable, unexpected behaviors
- system needs to know how to distinguish between the two

sequential vs parallel.

- emergent behaviors can arise from parallel execution of multiple behaviors (e.g., flocking)
- also from sequential interaction with an interesting environment

architectures and emergence.

- different architectures affect the likelihood of generating and using emergent behaviors
- deliberative: always aim to eliminate them
- reactive: aim to exploit them
- hybrid: typically aim to eliminate them
- BBS: aim to exploit them

modularity and emergence.

- modularity directly effects emergence
- reactive and BBS employ parallel rules and behaviors which interact with each other and the environment, thus directly producing and exploiting emergent behavior

avoiding & exploiting.

- hybrid systems follow deliberative model in attempt.
- aim to:
  - *produce a coherent, uniform output of the system,*
  - *minimize interactions and emergence*

Summary

- This lecture has introduced:
  - *Behavior-based robotics; and*
  - *Emergent behavior*
- We also discussed methods for:
  - *Encoding behavior*
  - *Controlling interactions between behaviors*
- This approach is rather different from the more "classical" approach we will study in the rest of the course.

reading.

- *Behavior-Based Robotics*, chapter 3, by Ron Arkin.
- *Behavior-based Robotics, its scope and its prospects*, by Andreas Birk.