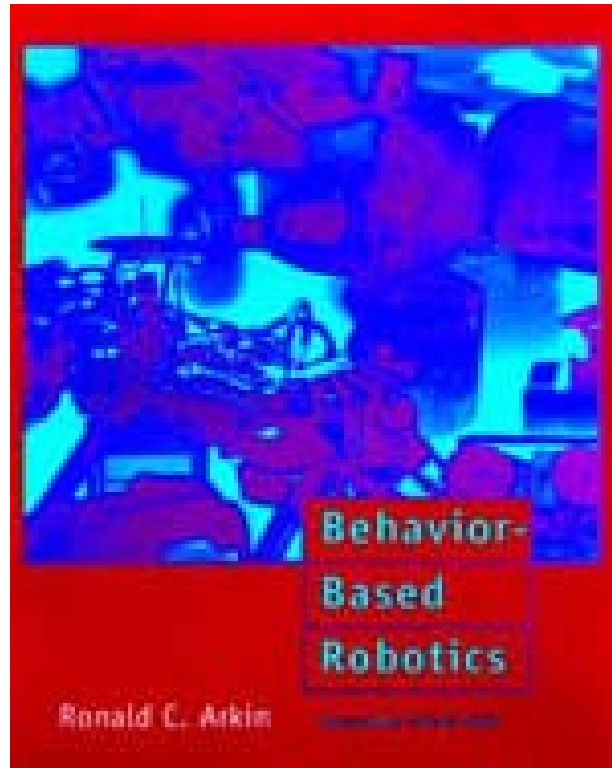


behavior-based systems.

- this lecture will cover behavior-based systems
 - *control*
 - *behavior-based systems*
 - *expressing behaviors*
 - *representations*
 - *behavior coordination*
 - *emergent behavior*
- this approach is not covered in the textbook

Source

- Much of this material is based on:



Behavior-Based Robotics
R. C. Arkin
MIT Press, 0-262-01165-4

models.

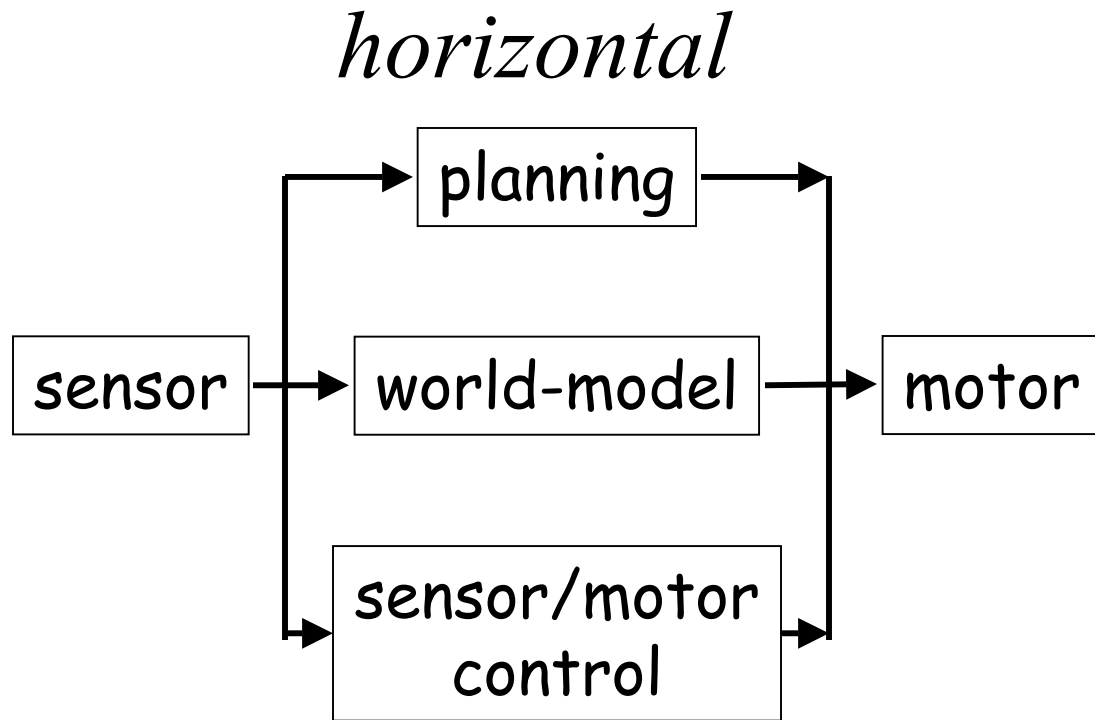
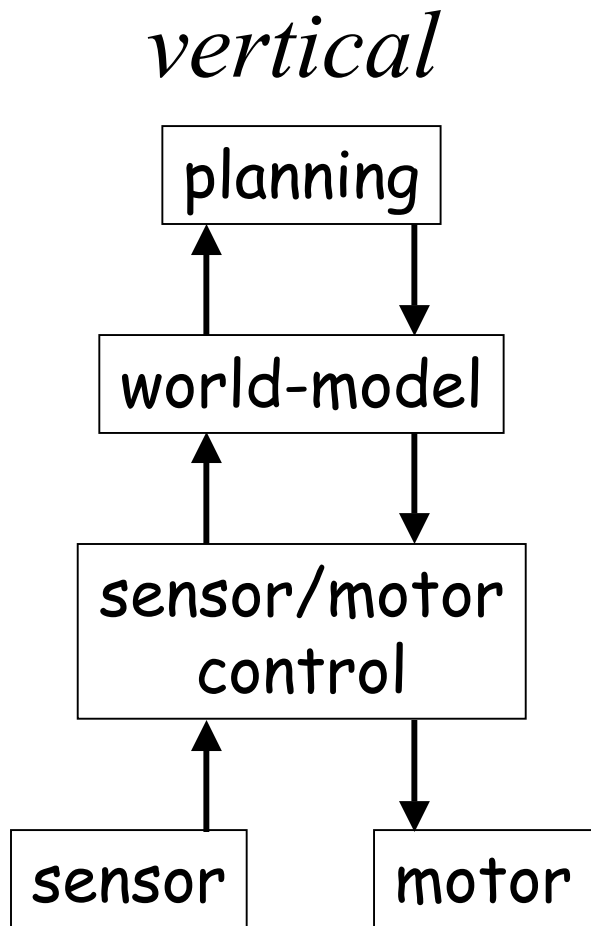
we like to make a distinction between

- classic “model-based” AI
 - *symbolic representations*
 - *explicit reasoning about the world*
- neo “behavior-based” AI
 - *numeric representations*
 - *little explicit reasoning about the world*

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
- behavior-based architectures
 - *task-oriented decomposition*
 - *systems consist of concurrent modules achieving specific tasks*
 - avoid obstacle
 - follow wall

two orthogonal flows.



distinctions.

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

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
- 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

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
 - *use biological models as inspiration*
- situated activity
 - *environment driven*
- experimentally driven
 - *bottom-up, iteratively refine controller*

Arkin, p69-75

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 or languages with functions like C

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

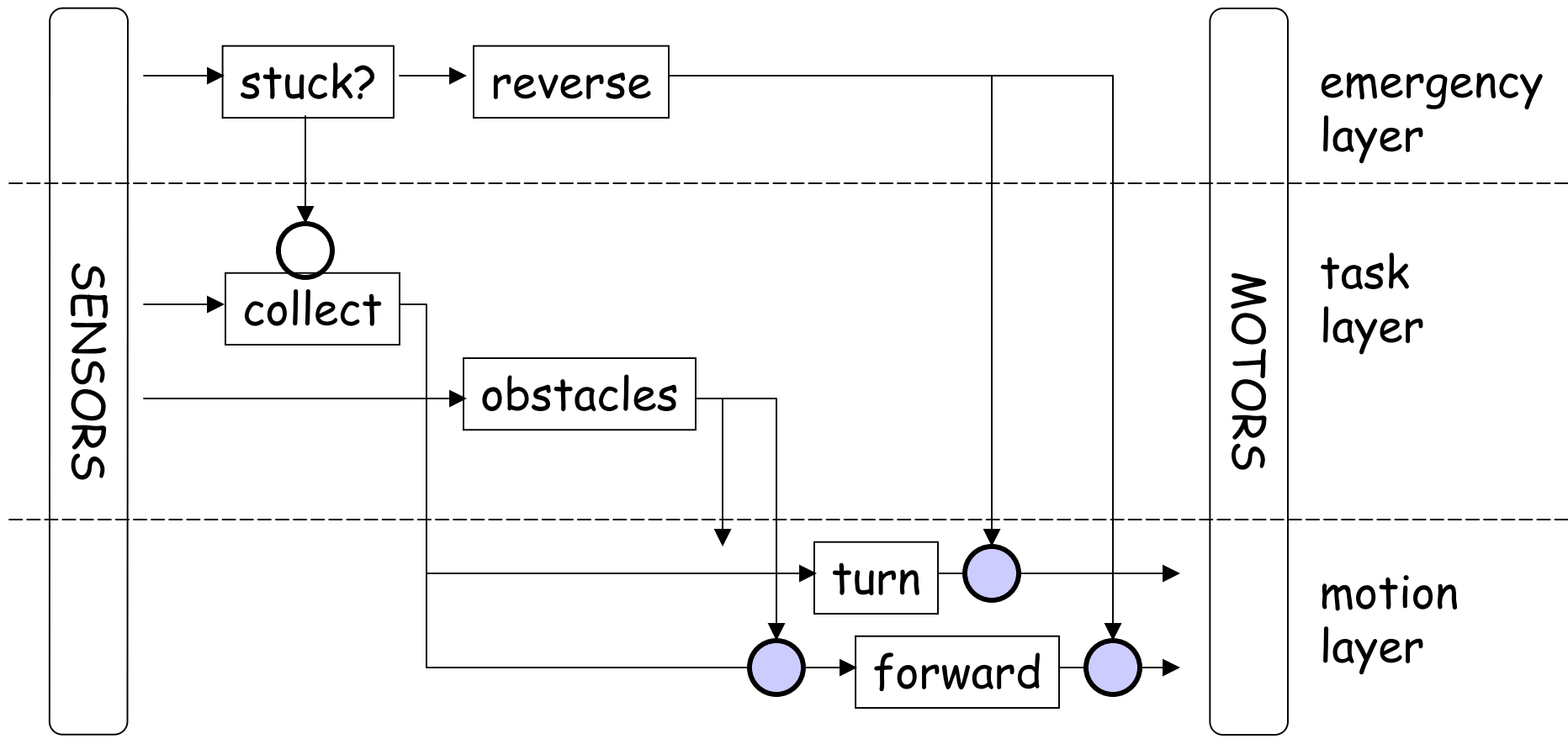
situated automata example.

```
(defgoalr (ach in-classroom)
  (if (not start0up)
      (maint (and (maint move-to-classroom)
                  (maint avoid-objects)
                  (maint dodge-students)
                  (maint stay-to-right-on-path)
                  (maint defer-to-elders))))))
```

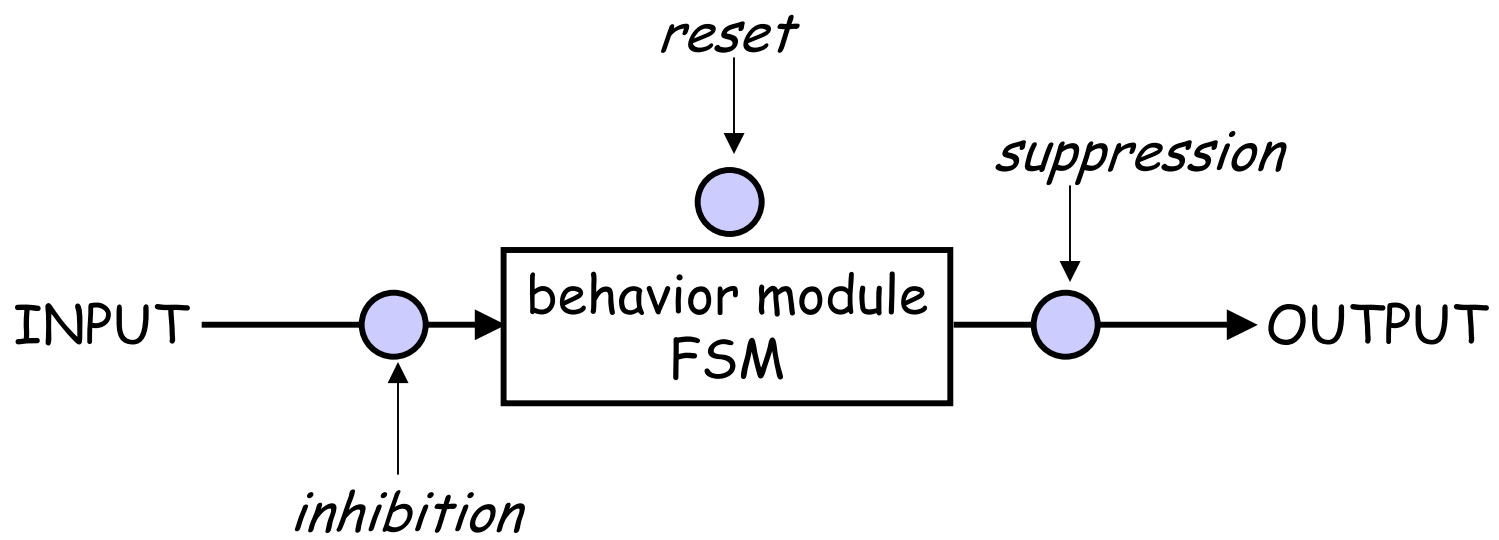
subsumption architecture.

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

subsumption architecture, 2.



augmented finite state machine



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

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!
- representation cannot be a traditional 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
 - 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 linked – a topological map

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*

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.

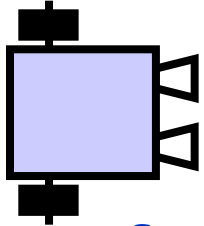
- important but not well-understood phenomenon
- often found in behavior-based robotics
- robot behaviors “emerge” from
 - *interactions of rules*
 - *interactions of behaviors*
 - *interactions of either with environment*

distinction.

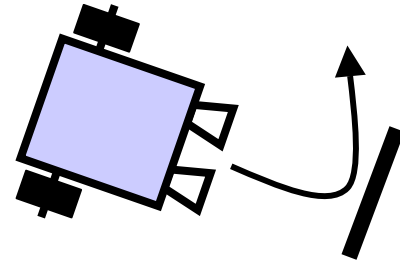
- coded behavior
 - *in the programming scheme*
- observed behavior
 - *in the eyes of the observer*
 - *emergence*
- there is no one-to-one mapping between the two!

Example: wall following.

coded behaviors

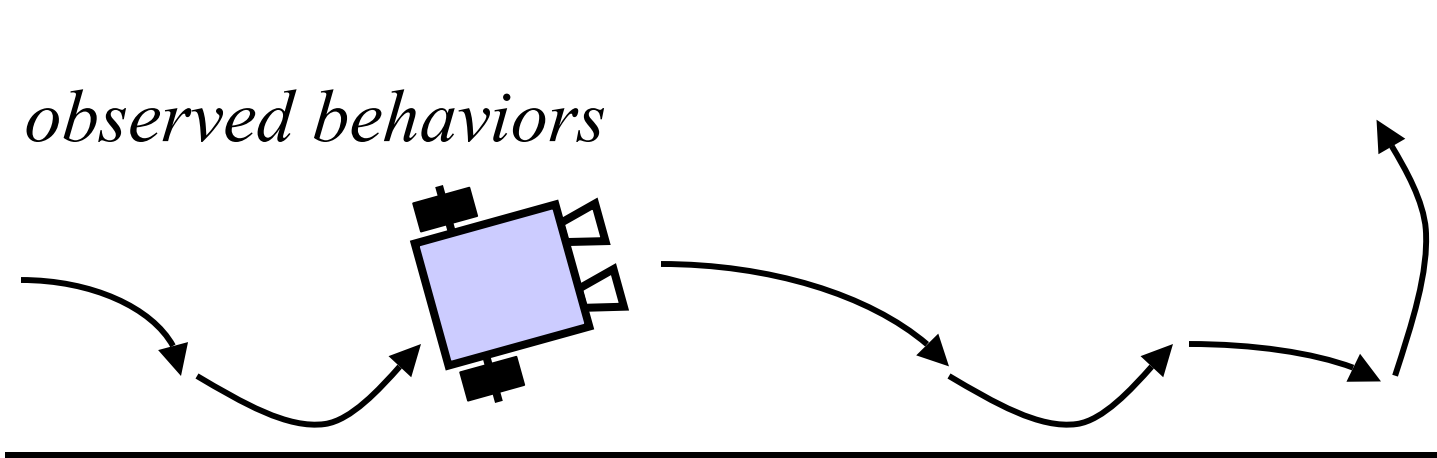


forward motion,
with slight turn right



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*
- but, once we know it is emergent we aren’t surprised that it happens!

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

architectures and emergence.

- different architectures affect emergence
- deliberative
 - *always aim to eliminate it*
- reactive
 - *aim to exploit it*
- hybrid
 - *typically aim to eliminate it*
- behavior-based systems:
 - *aim to exploit it*

modularity and emergence.

- modularity directly effects emergence
- reactive and BBS employ parallel rules and behaviors
 - *these interact with each other and the environment...*
 - *... directly producing and exploiting emergent behavior*

Summary

- this lecture has introduced:
 - *behavior-based robotics; and*
 - *emergent behavior*
- we also discussed methods for:
 - *describing behavior*
 - *controlling interactions between behaviors*
- this approach is rather different from the more “classical” approach we will study in the rest of the course.