Control architectures of a cognitive robot

Václav Hlaváč

Czech Technical University in Prague

Czech Institute of Informatics, Robotics and Cybernetics

160 00 Prague 6, Jugoslávských partyzánů 1580/3, Czech Republic

http://people.ciirc.cvut.cz/hlavac, vaclav.hlavac@cvut.cz

also Center for Machine Perception, http://cmp.felk.cvut.cz

Courtesy: Several lecture authors from the web.

Outline of the talk:

- Controller, brains of a robot.
- Control architecture.
- Behavior coordination.
- Deliberative control.
- Behavior-based control.
- Hybrid control.

Beyond cognitive control architectures

- Robots teleoperated by humans.

- Passive robots which do not need any explicit control.

  Courtesy: Steven H. Collins, CMU, 2005.

Controller ≈ brains of a robot

- The controller in a wider sense allows a robot to achieve its goals autonomously.
- The feedback control is an excellent tool for performing a single behavior (as avoiding obstacles, following the wall for a mobile robot).

- Autonomous robots are striving at achieving several goals simultaneously ranging
  - from simple survival behaviors (like not running out of power)
  - to complex activities (like acting in a robot-soccer field).

Just program the robot!?
There are several reasons why not.

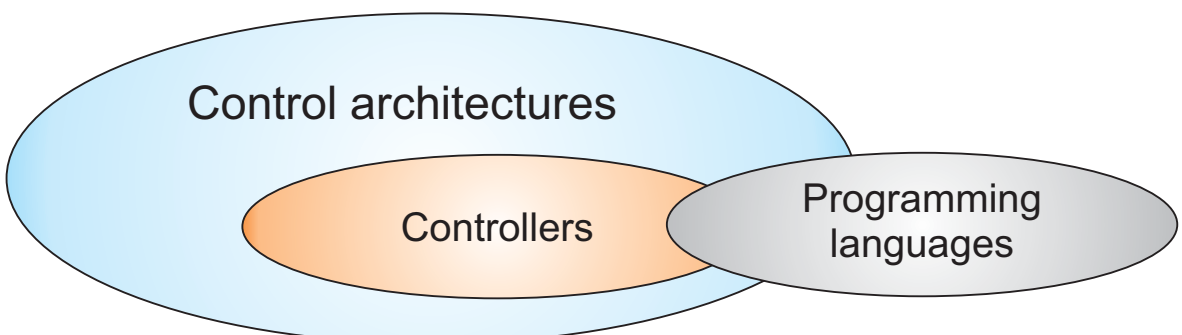
- What if more than one feedback control loop is needed?
- How to put multiple controllers together?
- How to decide what part of the control system to use?
- In what situation? For how long?
- What priorities are assigned to individual tasks?

- Just putting rules and programs together may be fun for a student task with small robots, which are not dangerous.

- Such a strategy does not bring a good result in general.
- *There is a need for guiding principles.*

The control architecture

- The control architecture provides guiding principles and constraints for organizing robot control system.
- There is an analogy with the computer or software architecture, i.e. it allows to construct the system from well understood building blocks.
- There is more in software than hardware. Q: Why? A: Because of flexibility.
- Not a single program.
  Q: Why?
  - A1: Robustness to failure.
  - A2: Exchangeability of modules.



Robot control — trade-offs

- Thinking is slow.
- Reaction must be fast.

- Thinking enables looking ahead (planning) to avoid bad situations and solutions.
- Thinking too long can be dangerous (e.g., falling off a cliff, being run over).
- To think, the robot needs (a lot of) rather accurate pieces of information, which has to be acquired in the noisy world.

  $\Longrightarrow$ *World models are needed.*

Levels of robot control tasks

Multiple control problems at different levels.

- Low-level control:

  Example: where to place a leg as robot takes its next step.

  - Continuous-valued problems in most cases.
  - A short time scale (under a second); high frequency loop; e.g. 1 kHz in haptics.

- Intermediate-level control:

  Example: Navigating to a destination, or picking up an object.

  - Continuous or discrete valued tasks.
  - A middle-time scale.
    E.g., a few seconds.

- High-level control:

  Example: Planning of a mission.

  - Discrete-valued tasks.
  - A long time scale.
    E.g., minutes.

Main control architectures

Robot architectures differ in dealing with:

- Time.
- Modularity.
- Robot world representation.

Common main control architectures:

- Deliberative control.
- Reactive control.
- Behavior-based control.

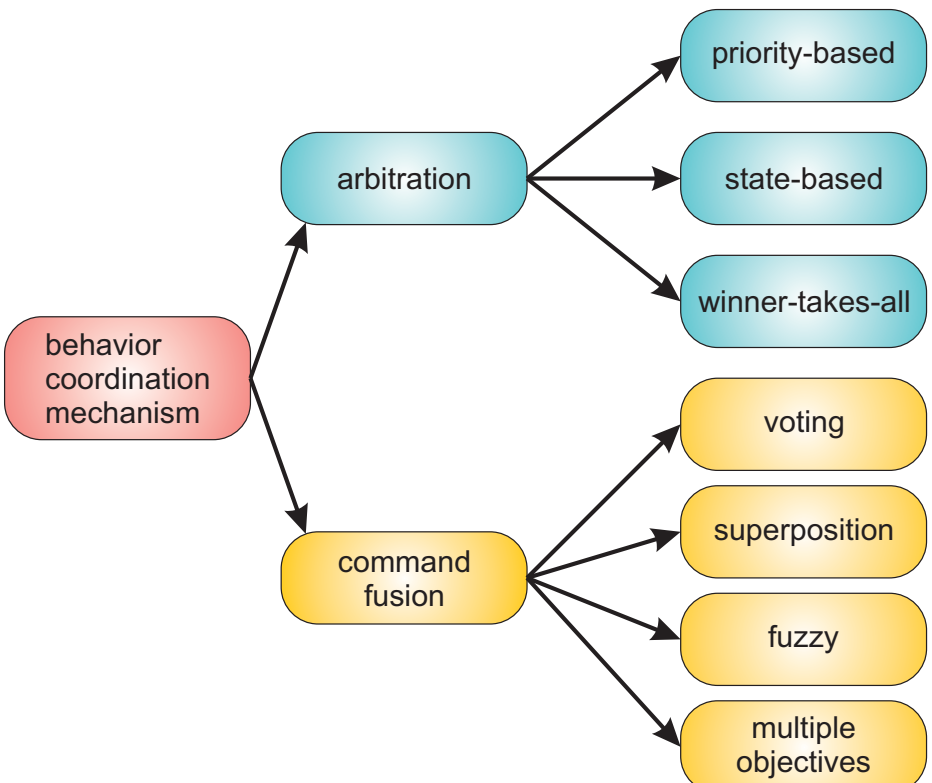
  The subsumption architecture in an often used case.
- Hybrid control.

Comparison of control architectures

Architecture	*Properties*
Deliberative	Think hard, act later. Lots of states. Maps of the robot environment. Look ahead.
Reactive	Do not think, react. No states. No maps. No look ahead.
Behavior-based	Think the way you act. Some states. Look ahead only while acting. Reactive + state.
Hybrid	Think and act independently, in parallel. States. Look ahead in parallel to acting. Combines long and short time scales.

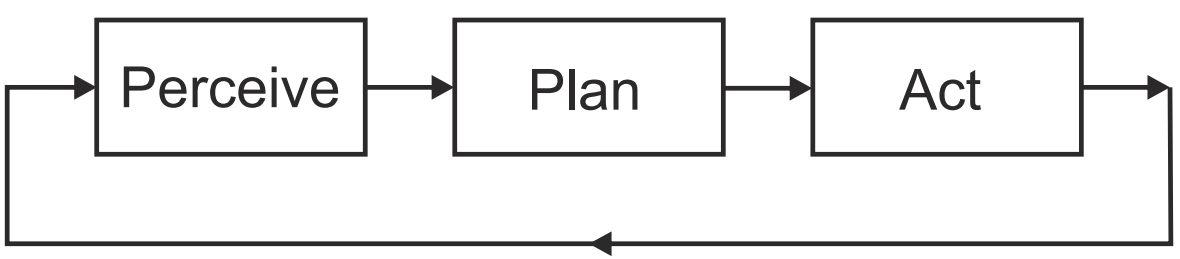
Behavior coordination

- Issue: When we have multiple behaviors, how do we combine them?
- Competitive (arbitration)
  - Example: pure arbitration, where only one behavior's output is selected.
- Cooperative (command fusion)
  - Blend outputs of multiple behaviors consistently with overall goals.
  - Example: vector addition in potential fields.



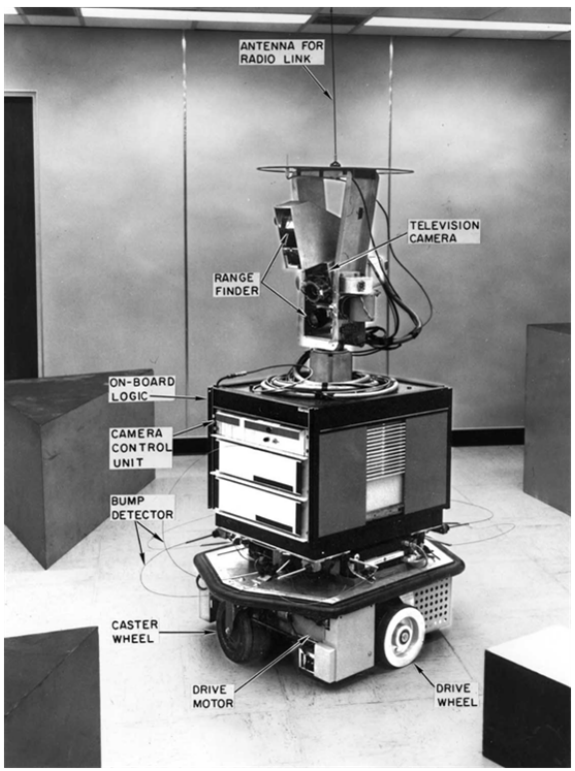
Deliberative control architecture

- University lab robotics was important playfield for starting artificial intelligence, $\sim$ 1960.
- Inherently serial (sequential).
- Planning requires search, which is both slow and memory hungry.
- Requires a (precise) world model.



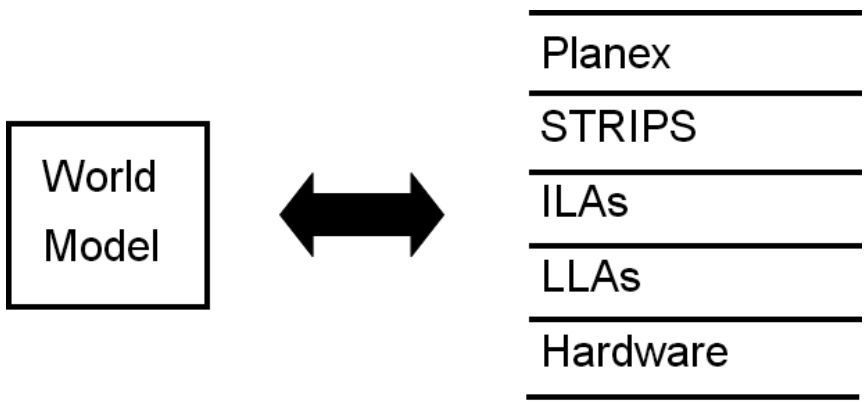
Deliberative control architecture
Shakey robot (1)

- Focus on automated reasoning and knowledge representation.
- STRIPS - Stanford Research Institute Problem Solver
  - Based on 1st order predicate logic reasoning.
  - Perfect world model.
  - Closed world assumption.
- Shakey robot (SRI 1969): pushing boxes.



Shakey robot (2)

- Central representation.
- Mathematical logic-based.
- Error recovery at several levels.
- Communication through a model.



Note:

LLAs = Low-level actions; ILAs = Intermediate-level actions.

PLANEX = plan execution monitoring and error-recovery system.

Shakey: key ingredients

- A mix of planning using logic and planning using geometric information.
- ILAs did simple error recovery internally (reactive controllers), e.g. push(box1, (14.1 22.3)).
- Major error recovery done by updating the world model, e.g. if the robot is uncertain about its position it takes a camera fix and updates the world model.
- The world model was based on the First Order Predicate Logic.

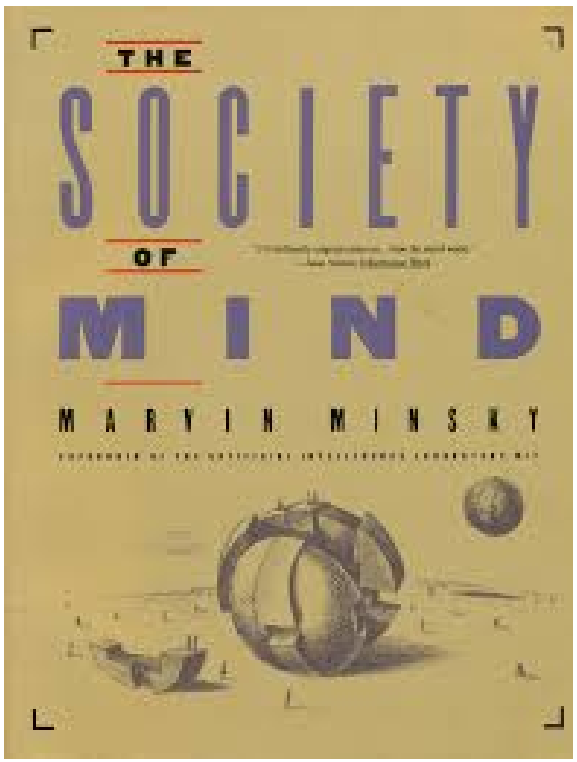
Drawbacks of the deliberative architecture

- Modelling the world is too hard and slow when a large state space is involved. Memory hungry.
- Non-linear planning is intractable (NP-complete).
- Feedback through the world model is cumbersome.
- A single chain maps sensing to action.
- A very general approach $\Longrightarrow$ poor at lots of specific tasks.
- Passing representations between modules is slow.

Other things are difficult for robots than for humans

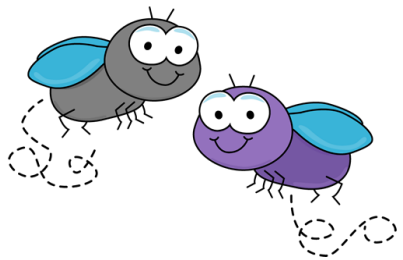
Marvin Minsky's book: The Society of Mind, Simon & Schuster Paperbacks, 1985, section 2.5:

"In attempting to make our robot work, we found that many everyday problems were much more complicated than the sorts of problems, puzzles, and games adults consider hard."



A house fly, the unlikely architecture

- Forms 3D surface descriptions of objects.
- Reasons about the threat of a human with a fly swatter, in particular about the human's beliefs, goals, or plans.
- Makes analogies concerning the suitability for egg laying between dead pigs.
- Constructs naïve physics theories of how to land on the ceiling.



A house fly, the likely architecture

- Has close connection of sensors to actuators.
- Has pre-wired patterns of behavior.
- Has simple navigation techniques.
- Functions almost as a deterministic machine.
- And yet, a house fly is much more successful in the real world than our attempts at artificial intelligence.

Flies' compound eye.

- Thousands of visual receptors, ommatidia, each of them is functioning eye, no focusing, short-sighted.
- Each ommatidium is connected to the brain.
- Much faster than human vision.

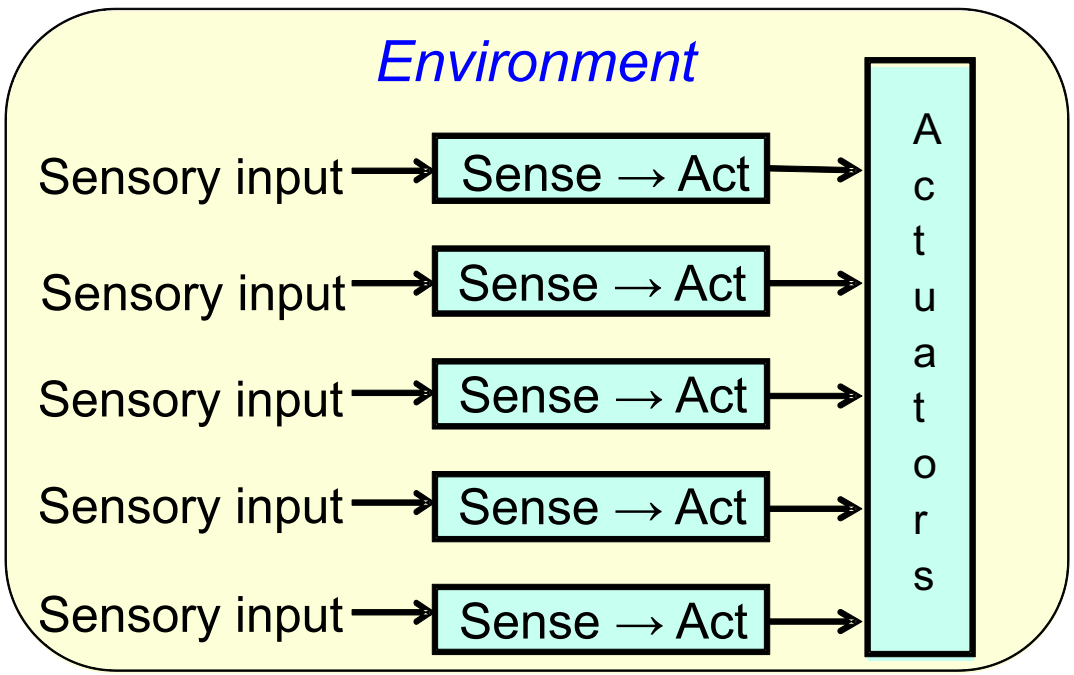
Reactive control (1)

- Turn the problem on its head.
- There are no general purpose animals . . .
  Why should there be general purpose robots?
- Do not build world models.
- Do not plan.
- Use short feedback loops.
- Create many chains that map sensing to action.

Reactive control (2)

- Collections of sense-act (stimulus-response) rules.
- Rules implemented as assembly code, C++ code, EPLD combinational logic, FPGA state machine, state machine with stacks (memory), etc.
- Inherently concurrent (parallel).
- Very specific $\Longrightarrow$ good at one or two tasks.
- Representations should not be passed between modules.

Reactive architecture diagrammatically



Concurent task-achieving modules

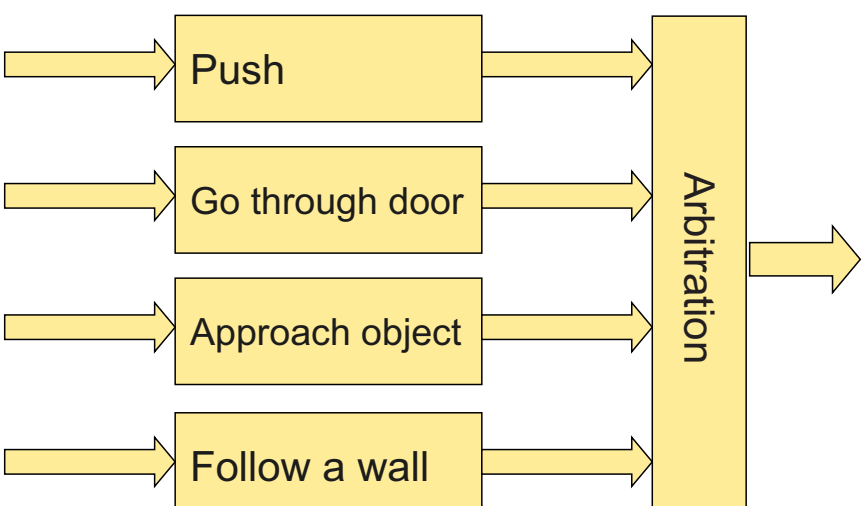
Example: The wall following (a)

Left, right whisker – on/off.

Algorithm

1. If the left whisker is bent then turn right.
2. If the right whisker is bent then turn left.
3. If both whiskers are bent then back up and turn left.
4. Otherwise, keep going.

Deliberative vs. reactive architectures; A comparison



Limitations of the reactive control

- Minimal representation of states, if any.
- No memory.
- No learning.
- No internal models / representations of the world.

Example: The wall following (b)

Robot got stuck in the corner. Oscillations appeared.

What to do about oscillations?

1. Explore randomness a little.
   It could take a lot of time to get out of the corner.
2. Keep a bit of history.
   E.g., remember where the robot turned last time = 1 bit of memory.

Behavior-based control

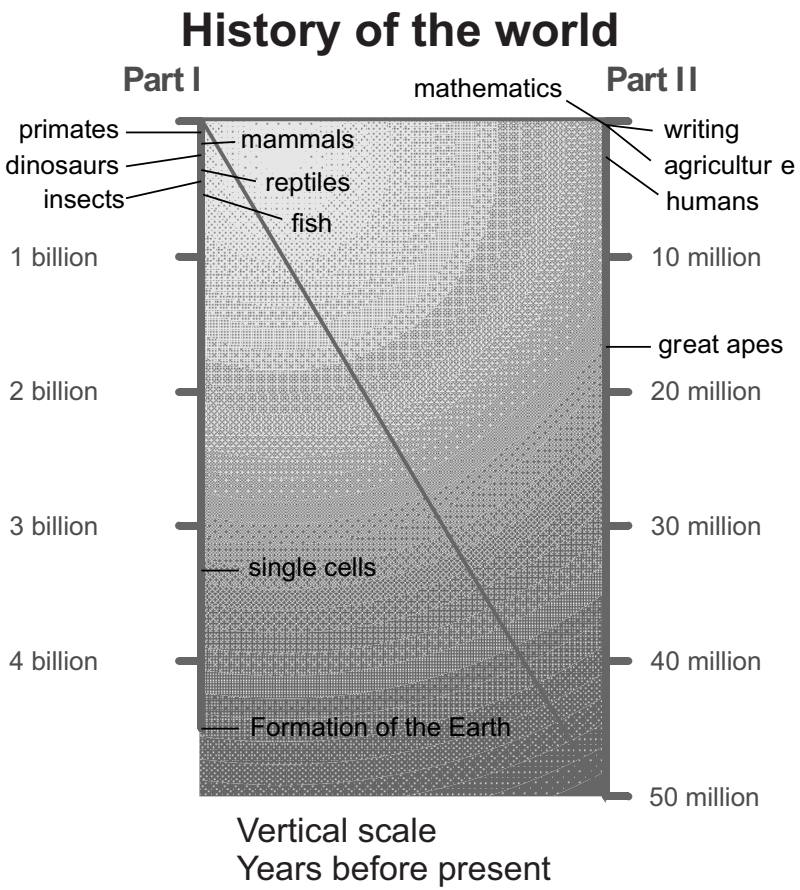
- The ability to act reactively, the ability to act deliberatively. No intermediate level *(Note: This is a differentiation property with respect to hybrid control).*
- A unified, consistent representation is used in the whole system. That resolves issues of time-scale.
- Overall controller composed of two parts
  - Task achieving controllers.
  - Arbitrating controller (also task specific).
- Controllers
  - Are reactive. They map perception to action.
  - Have no models. There is no planning.
  - Are only capable of performing one task each.

Biological inspiration

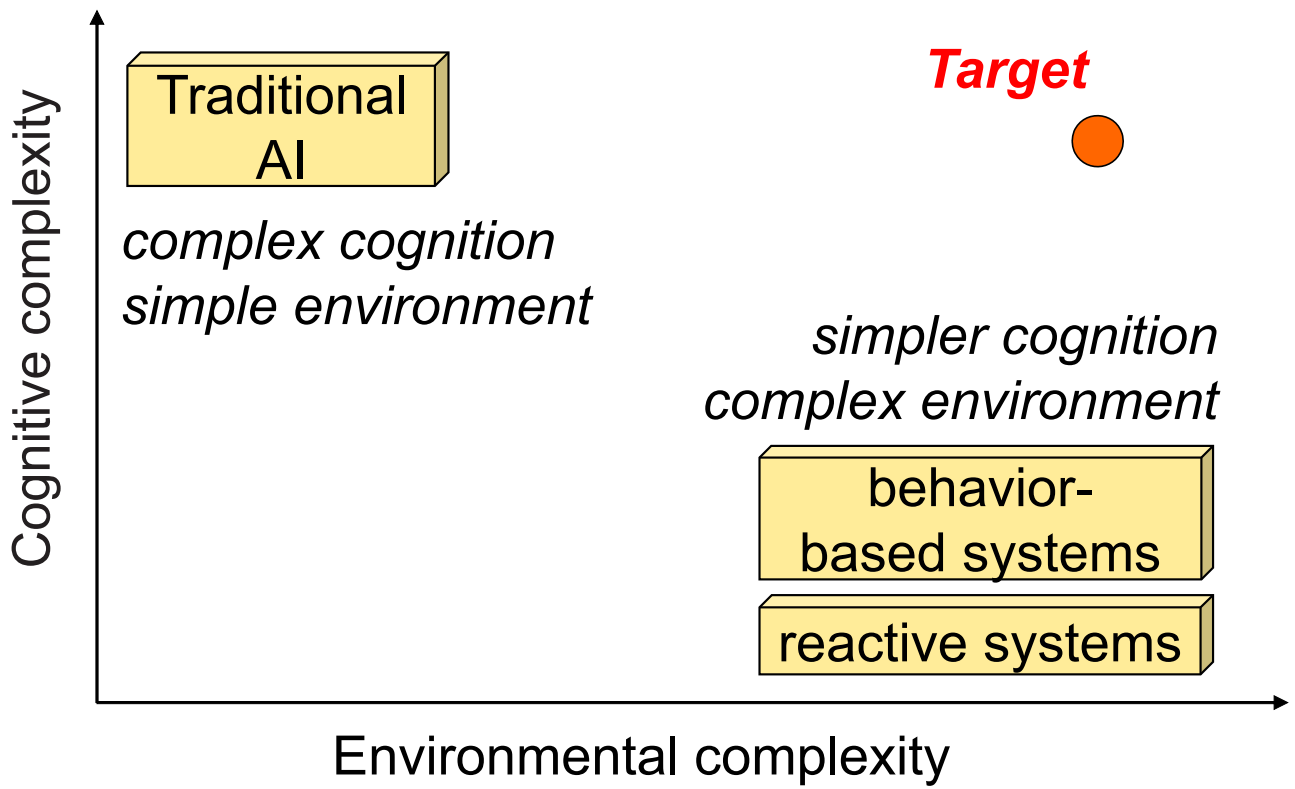
- The inspiration behind the Subsumption Architecture is the evolutionary process:
  New competencies are introduced based on existing ones.

- 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.

Where did evolution spent its time?



Cognitive (task) / environment complexity



Eyes, brain, hand: Rules for Eyes

Eyes:

Shout "Stop!" if Hand is in danger. (i.e. Hand might trip up or hit something). Otherwise you can answer:

1. Yes/No questions, e.g. *Q: Is there a box on the floor?*
2. How many? e.g. *Q: How many boxes are in the room?*
3. Spatial relationship questions, e.g. *Q: Where is the nearest box to the hand?*
   A: 3 metres in front of the hand.

Eye, brain, hand: Rules for Brain, Hand

Brain:

Build a tower out of three boxes on the table at the front of the lecture room. You may:

1. Give instructions to Hand.
2. Ask Eyes questions (yes/no, how many, how far, where).

Hand:

If Eyes shout "Stop!" then stop: something dangerous is about to happen.

- Otherwise do what brain tells you.

Behavior-based system, rules Agent 1

Agent 1: (Box finder)

Rule 1 If another agent is holding a box then look for boxes on the table.

Rule 2 If no other agent is holding a box look for boxes on the floor.

Rule 3 Go to the box you are looking at and stand next to it. Raise your hand if it is on the floor.

Rule 4 When there are no boxes left to look for return to your seat.

Behavior-based system, rules Agent 2

Agent 2: (Box getter)

Rule 1 Find and follow agent 1.

Rule 2 If agent 1 is stationary with hand raised, pick up the object near agent 1's feet.

Rule 3 If agent 1 is stationary and hand is down, hold out the object you are carrying to the other agent (not 1) nearby.

Rule 4 If agent 1 sits down return to your seat.

Behavior-based system, rules Agent 3

Agent 3: (Box Stacker)

Rule 1 Stand by the table until the other agents are seated.

Rule 2 When the other agents are seated return to your seat.

Rule 3 If another agent offers you an object take it.

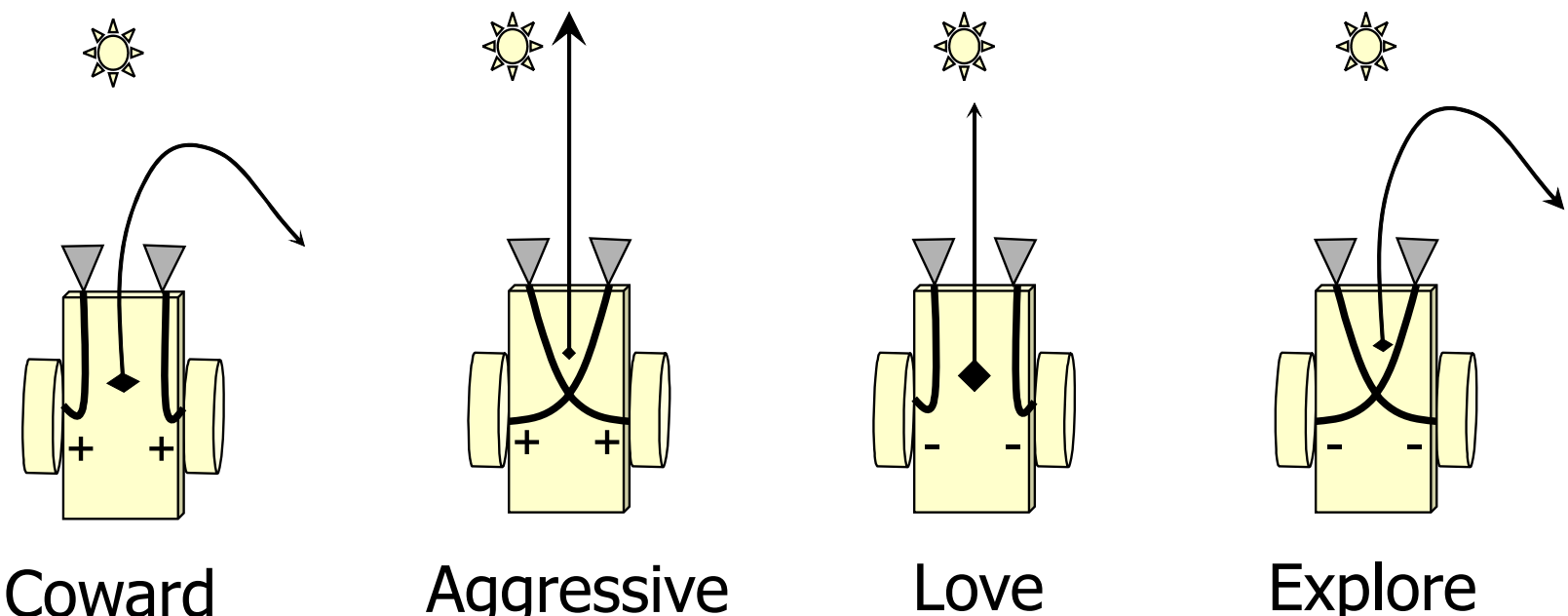
Rule 4 If you are holding an object, place it carefully on top of the object(s) in front of you.

Behavior-based system: assumptions

- Each agent has the relatively simple perceptual and physical talents to perform its task easily.
- The difficulties of the classical system are not just due to natural language.
- We were able to decompose the task into appropriate sub-tasks.

Braitenberg vehicles

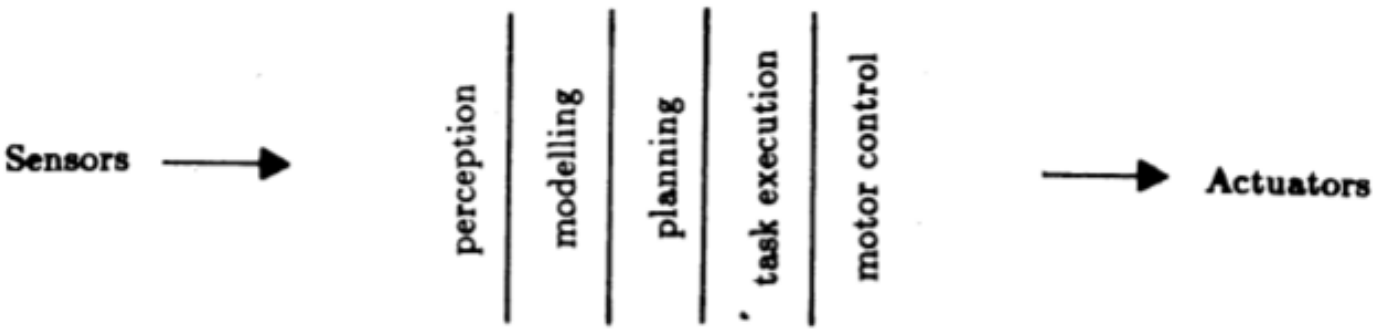
- Complex behavior can be achieved using very simple control mechanisms.
- Braitenberg vehicles (Valentino Braitenberg, early 1980s, originally thought experiment, implemented later): analog, differential drive mobile robots with two light sensors.



Rodney Brooks

- Born 1954, Australia.
- MIT, Computer Science and Artificial Lab (director 1997-2007).
- Professor of Robotics.
- Moved robotics towards reactive paradigm at the end of 1980s.
- Inventor of subsumption robot control architecture (which he later stopped developing).

Decomposition of robot control into functional modules



Traditional deliberative control architecture.

From R. Brooks: A Robust Layered Control System for a Mobile Robot, 1985.

R. Brooks - behavior languages, 3 theses, 2 ideas

Rodney Brooks, three theses:

1. Intelligent behavior can be generated without explicit representations of the kind that symbolic AI proposes.
2. Intelligent behavior can be generated without explicit abstract reasoning of the kind that symbolic AI proposes.
3. Intelligence is an emergent property of certain complex systems.

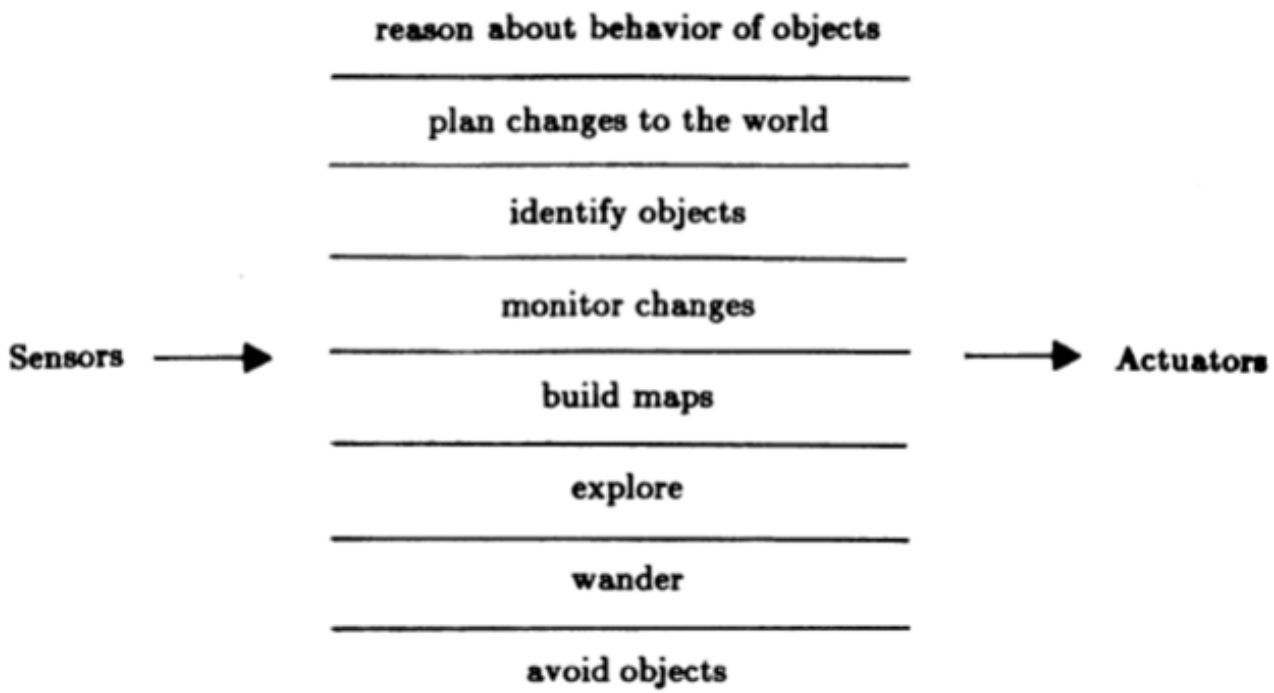
Rodney Brooks, two additional ideas:

1. Situatedness and embodiment: 'Real' intelligence is situated in the world, not in disembodied systems such as theorem provers or expert systems.
2. Intelligence and emergence: 'Intelligent' behavior arises as a result of an agent's interaction with its environment. Also, intelligence is 'in the eye of the beholder'; it is not an innate, isolated property.

R. Brooks – behavior languages (2)

- To illustrate his ideas, Brooks built some robots based on his subsumption architecture.
- A subsumption architecture is a hierarchy of task-accomplishing behaviors.
- Each behavior is a rather simple rule-like structure.
- Each behavior 'competes' with others to exercise control over the agent.
- Lower layers represent more primitive kinds of behavior (such as avoiding obstacles), and have precedence over layers further up the hierarchy.
- The resulting systems are, in terms of the amount of computation they do, extremely simple.
- Some of the robots do tasks that would be impressive if they were accomplished by symbolic AI systems. Recall the house fly example.

Robot control decomposition into task achieving behaviors



From R. Brooks: A Robust Layered Control System for a Mobile Robot, 1985

Reactive architecture, pros and cons

Advantages

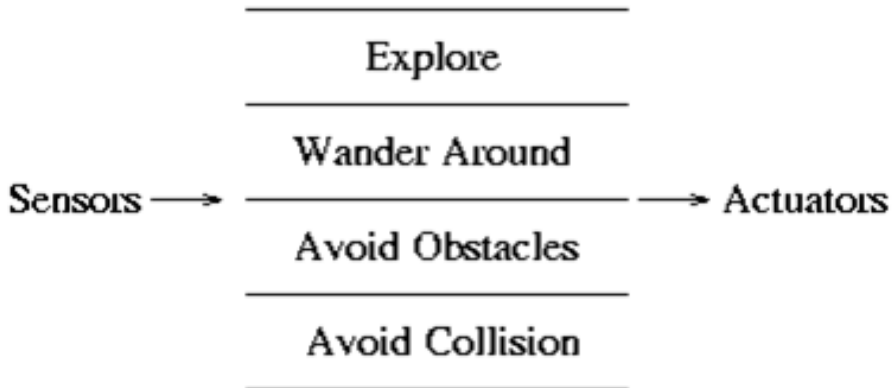
- Simplicity.
- Economy.
- Computational tractability.
- Robustness against failure.
- Elegance.

Disadvantages

- Agents without environment models need sufficient information from local environment.
- If decisions are based on local environment, it does not take into account non-local information (i.e., a 'short-term' view only).
- Difficult to make reactive agents that learn.
- Since behavior emerges from component interactions plus environment, engineering specific agents is difficult (no principled methodology exists).
- It is hard to engineer agents with large numbers of behaviors (dynamics of interactions become too complex to understand).

The subsumption architecture; principles of design

- Systems are built in the bottom up manner.
- Components are task-achieving actions/behaviors (avoid-obstacles, find-doors, . . .)
- All rules can be executed in parallel.
- Components are organized in layers, in a bottom up manner.
- Lowest layers handle most basic tasks.
- Newly added components and layers exploit the existing ones.

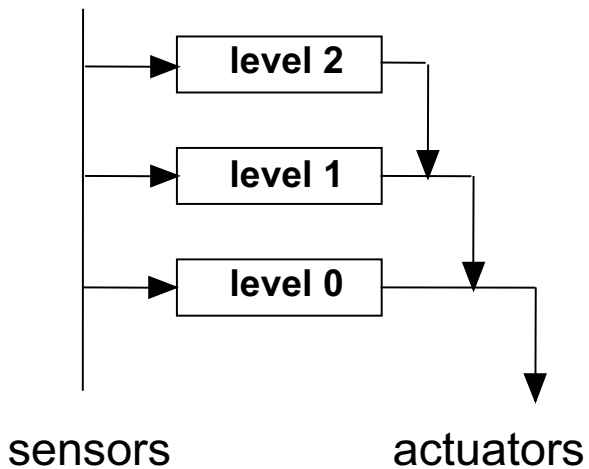


Designing in the subsumption architecture

- Qualitatively specify the overall behavior needed for the task.
- Decompose that into specific and independent behaviors (layers).
- Determine behavior granularity.
- Ground low-level behaviors in the robot's sensors and effectors.
- Incrementally build, test, and add.

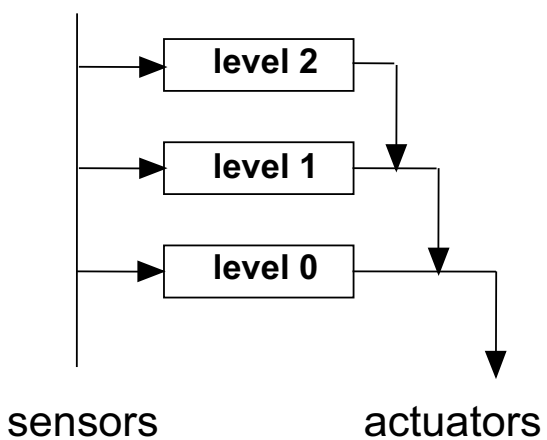
Subsumption layers

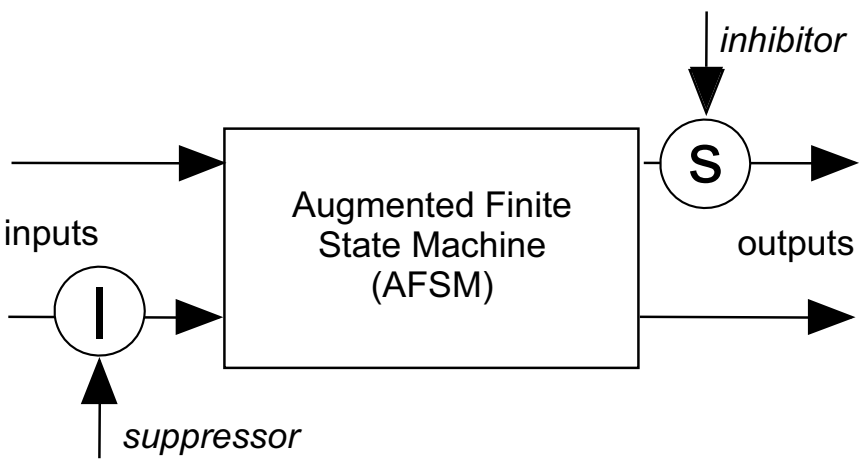
- First, design, implement and debug layer 0.
- Next, design layer 1
  - When layer 1 is designed, layer 0 is taken into consideration and utilized, its existence is subsumed (thus the name of the architecture).
  - As layer 1 is added, layer 0 continues to function.
- Continue designing layers, until the desired task is achieved.



Suppression and inhibition

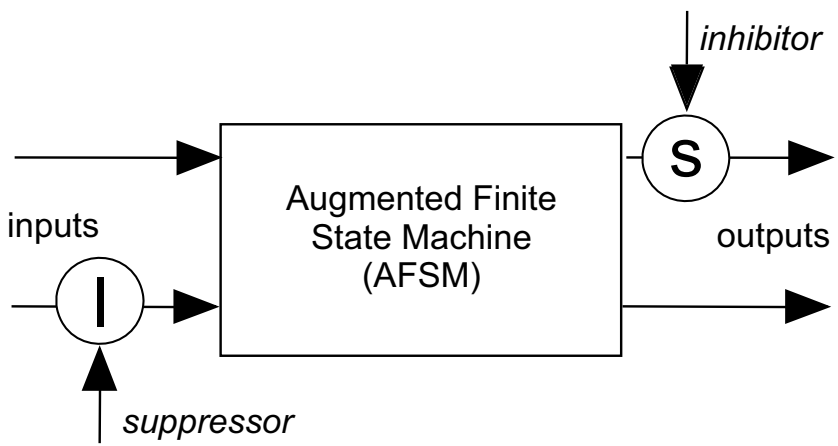
- Higher layers can disable the ones below. E.g. Avoid-obstacles can stop the robot from moving around.
- Layer 2 can either:
  - Inhibit the output of level 1 or
  - Suppress the input of level 1.
- The process is continued all the way to the top level.





Subsumption language and AFSMs

- The original Subsumption Architecture was implemented using the Subsumption Language
- It was based on finite state machines (FSMs), augmented with a very small amount of state (AFSMs).
- AFSMs were implemented in Lisp.

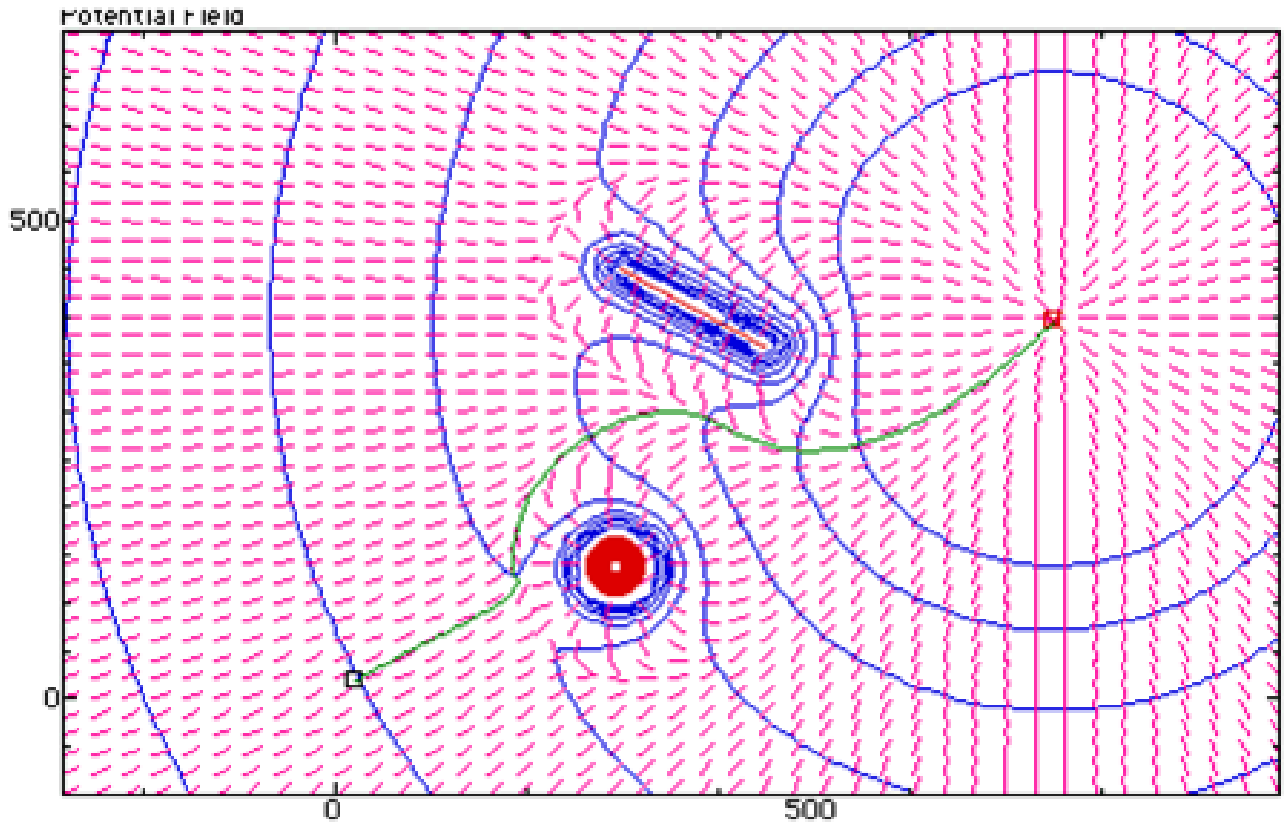


Subsumption language and AFSMs (2)

- Each behavior is represented as an augmented finite state machine (AFSMs).
- Stimulus (input) or response (output) can be inhibited or suppressed by other active be behaviors.
- 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 communication wires, which pass input and output messages between them; only the last message is kept.
- AFSMs run asynchronously.
- *Example: consider AFSM with input from the sonar, realizing behaviour collision detection, and providing the output halt.*

Reactive system using a potential field

Create a repulsion force around obstacles plus the attraction force to the goal.



Behavior-based architecture, summary

- An alternative to hybrid systems (to be discussed in the sequel), which has similar capabilities:
  - the ability to act reactively,
  - the ability to act deliberatively.
- There is no intermediate layer.
- An unified, consistent representation is used in the whole system $\Longrightarrow$ concurrent behaviors.
- Resolves issues of time-scale.

Hybrid architecture (1)

- Combines the two extremes:
  - reactive system at the bottom,
  - deliberative system on the top,
  - connected by some intermediate layer.
- Called also 3-layer systems.
- Layers operate concurrently.
- There are different representations and time-scales between the layers.
- Often, the reactive component is given some kind of precedence over the deliberative one.

Hybrid architecture (2)

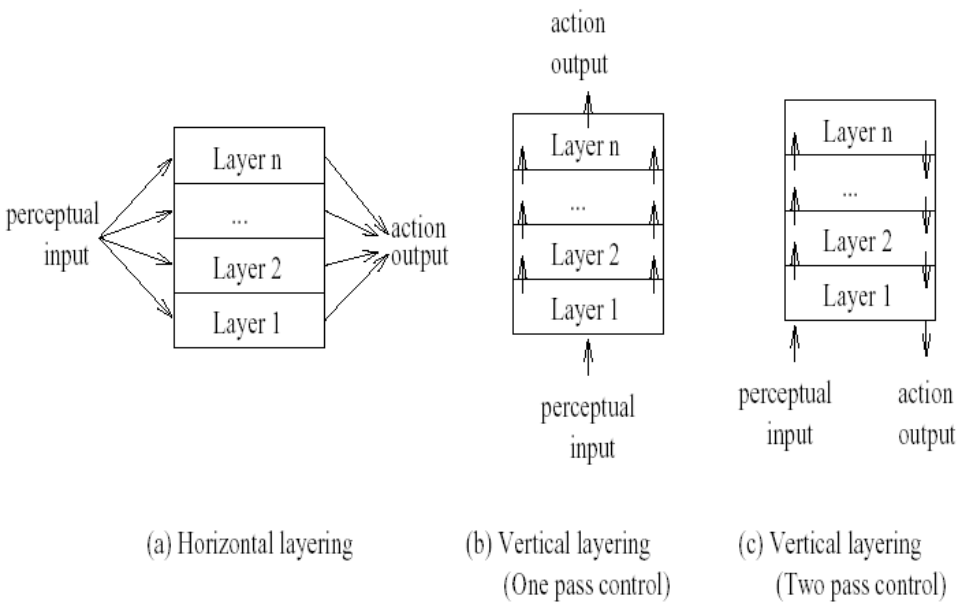
- A key issue is what kind of control framework use to manage the interactions between the various layers.
- Horizontal layering.

  Layers are each directly connected to the sensory input and action output. In effect, each layer itself acts like an agent, producing suggestions as to what action to perform.
- Vertical layering.

  Sensory input and action output are each dealt with by one layer at most.

Horizontal, vertical layering, comparison

Let assume n layers and m possible actions suggested by each layer.



Horizontal layering: m^n interactions, not tolerant to layer failure.

Vertical layering: $m^2(n-1)$ interactions. Introduces bottleneck in the central control system.