Cognitive Robot Control Architectures

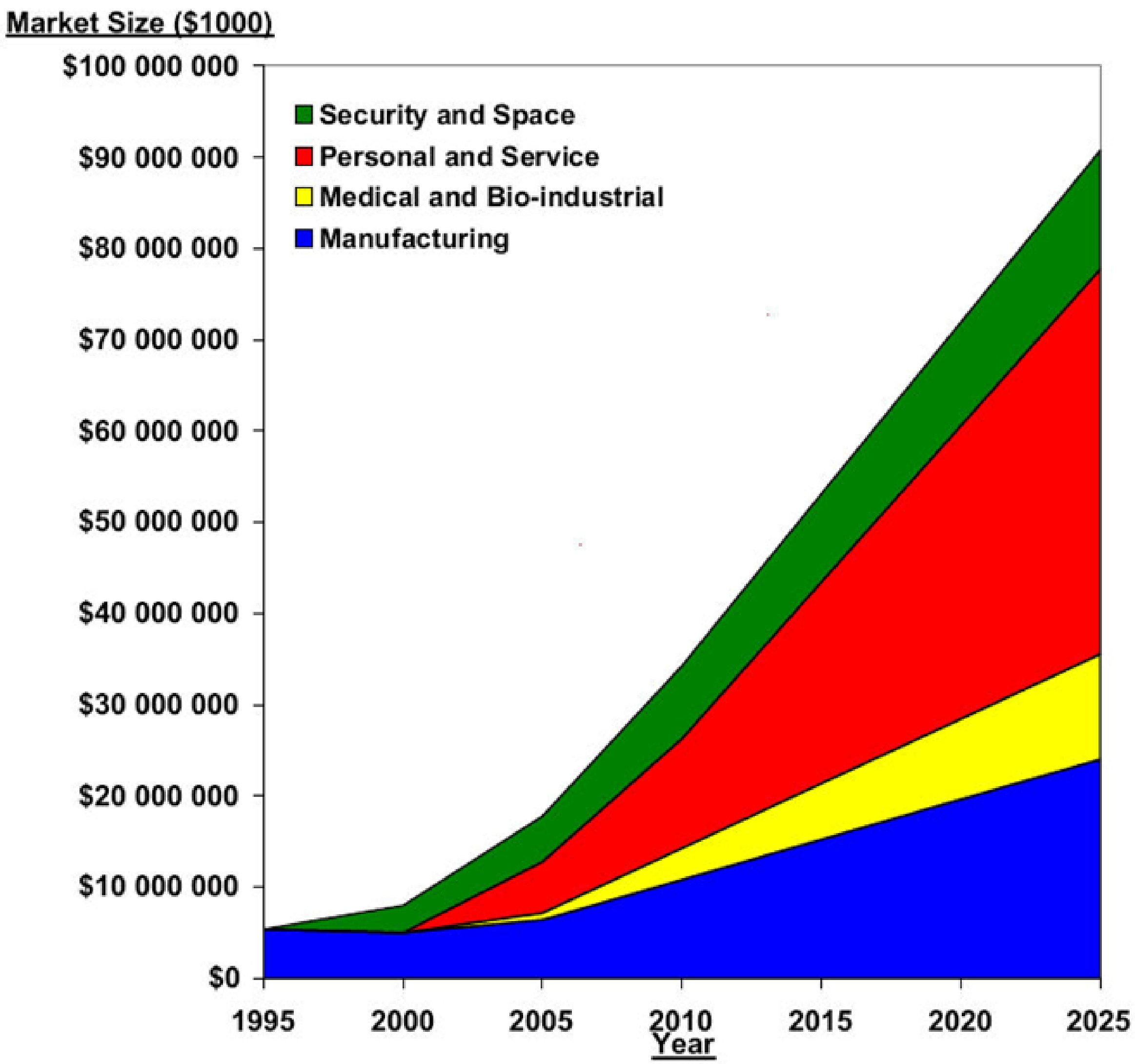
Václav Hlaváč
Czech Technical University in Prague
Faculty of Electrical Engineering
Department of Cybernetics
Czech Republic
http://cmp.felk.cvut.cz/~hlavac

Courtesy to several authors of presentations on the web.

Lecture outline

- Intelligent (cognitive) robotics
- Control architectures, general idea
- Cognitive robot architectures
  - Deliberative (sense → plan → act), since 1960'
  - Reactive
    - Pure reactive
    - Behavior-based
    - Subsumption architecture, R. Brooks 1986
  - Hybrid, since ~1995

Cogn. robotics = growing field



Source: Japanese Robotics Association

Outside of control architectures

- Human teleoperated robots.
- Passive robots which do not need explicit control at all.

da Vinci surgical robot

Lecture outline

- Intelligent (cognitive) robotics
- Control architectures, general idea
- Cognitive robot architectures
  - Deliberative (sense → plan → act), since 1960’
  - Reactive
    - Pure reactive
    - Behavior-based
    - Subsumption architecture, R. Brooks 1986
  - Hybrid, since ~1995

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).
- Intelligent 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 it! Why not?

Issues:

- What if more than one feedback control 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.
- A need for guiding principles.

The control architecture

- It provides guiding principles and constraints for organizing robot's control system.
- Analogy with the computer or software architecture – allows to construct the system from well understood building blocks.

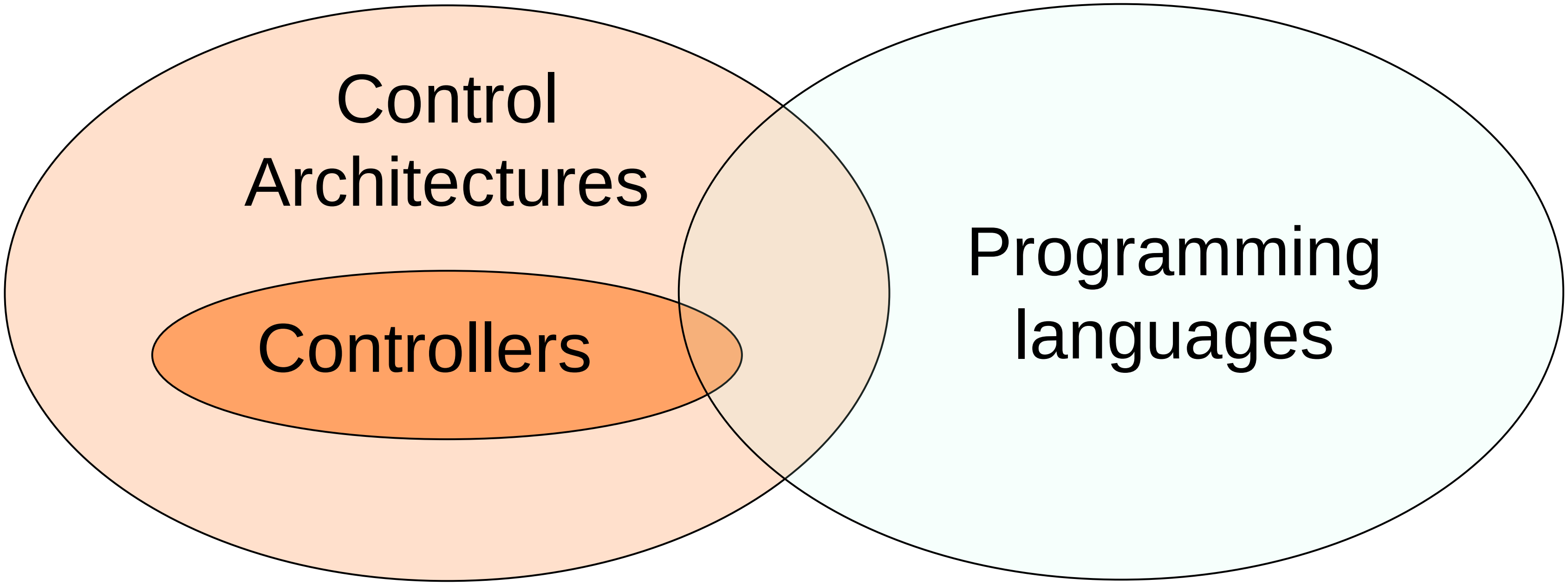
More in software than hardware. Why?

- Flexibility.

Not a single program. Why?

- Robustness to failure.
- Exchangeability of modules.

Relationship



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.
  - Generally, continuous-valued problems
  - Short time scale (under a second); high frequency loop; e.g. 1 kHz in haptics.
- Intermediate level control:
  - Navigating to a destination, or picking up an object.
  - Continuous or discrete valued problems.
  - Time scale, a few seconds.
- High level control:
  - Planning of a mission.
  - long time scale, minutes.

Three main control architectures

1. Deliberative control.
2. Reactive control *(subsumption architecture)*.
3. Hybrid control.
4. Behavior-based control.

Architectures differ in dealing with:

- Time.
- Modularity.
- Representation.

Control architectures comparison

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 but act. Combines long and short time scales.

Robot control – trade-offs

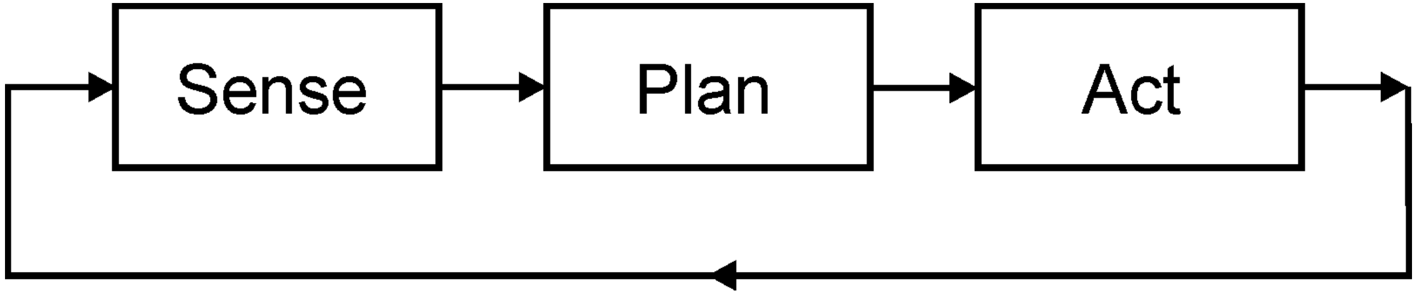
- Thinking is slow.
- Reaction must be fast.
- Thinking enables looking ahead (planning) to avoid bad solutions.
- Thinking too long can be dangerous (e.g., falling off a cliff, being run over).
- To think, the robot needs (a lot of) accurate information $\Rightarrow$ world models.

Lecture outline

- Intelligent (cognitive) robotics
- Control architectures, general idea
- Cognitive robot architectures
  - Deliberative (sense → plan → act), since 1960’
  - Reactive
    - Pure reactive
    - Behavior-based
    - Subsumption architecture, R. Brooks 1986
  - Hybrid, since ~1995

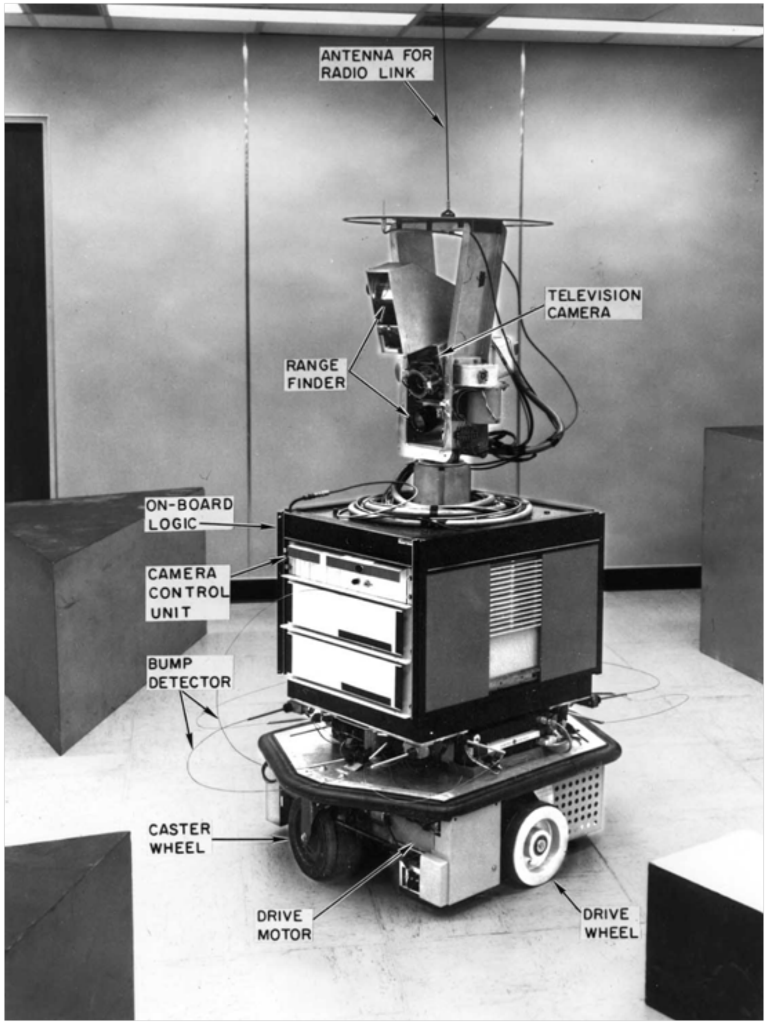
Deliberative control architecture

- University lab robotics was important playfield for starting artificial intelligence, ~ 1960.
- Inherently serial (sequential).
- Planning requires search, which are both slow.
- Requires a (precise) world model.

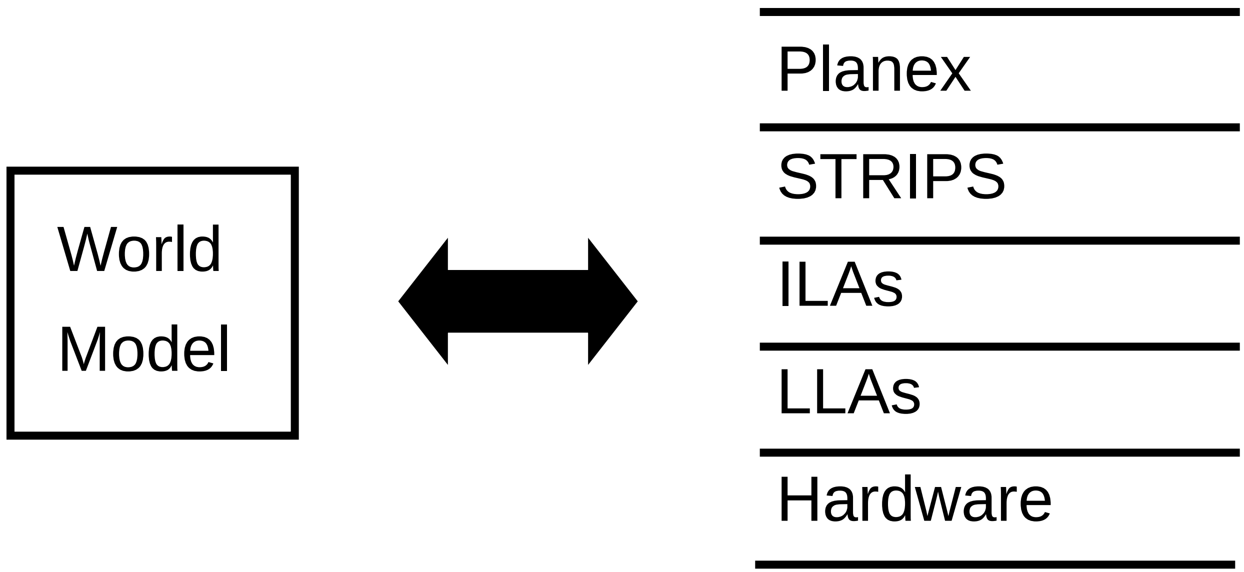


Deliberative control architecture 2

- 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 outline



- Central representation
- Logic based
- Error recovery at several levels
- Communication through a model

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.
- World model based on First Order Predicate Logic.

Deliberative arch. - Drawbacks

- 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.
- Very general → poor at lots of tasks.
- Passing representations between modules is slow.

Lecture outline

- Intelligent (cognitive) robotics
- Control architectures, general idea
- Cognitive robot architectures
  - Deliberative (sense → plan → act), since 1960’
  - Reactive
    - Pure reactive
    - Behavior-based
    - Subsumption architecture, R. Brooks 1986
  - Hybrid, since ~1995

Marvin Minsky: *Society of Mind*

From a book published in 1986:

EASY THINGS ARE HARD

“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, 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, 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.

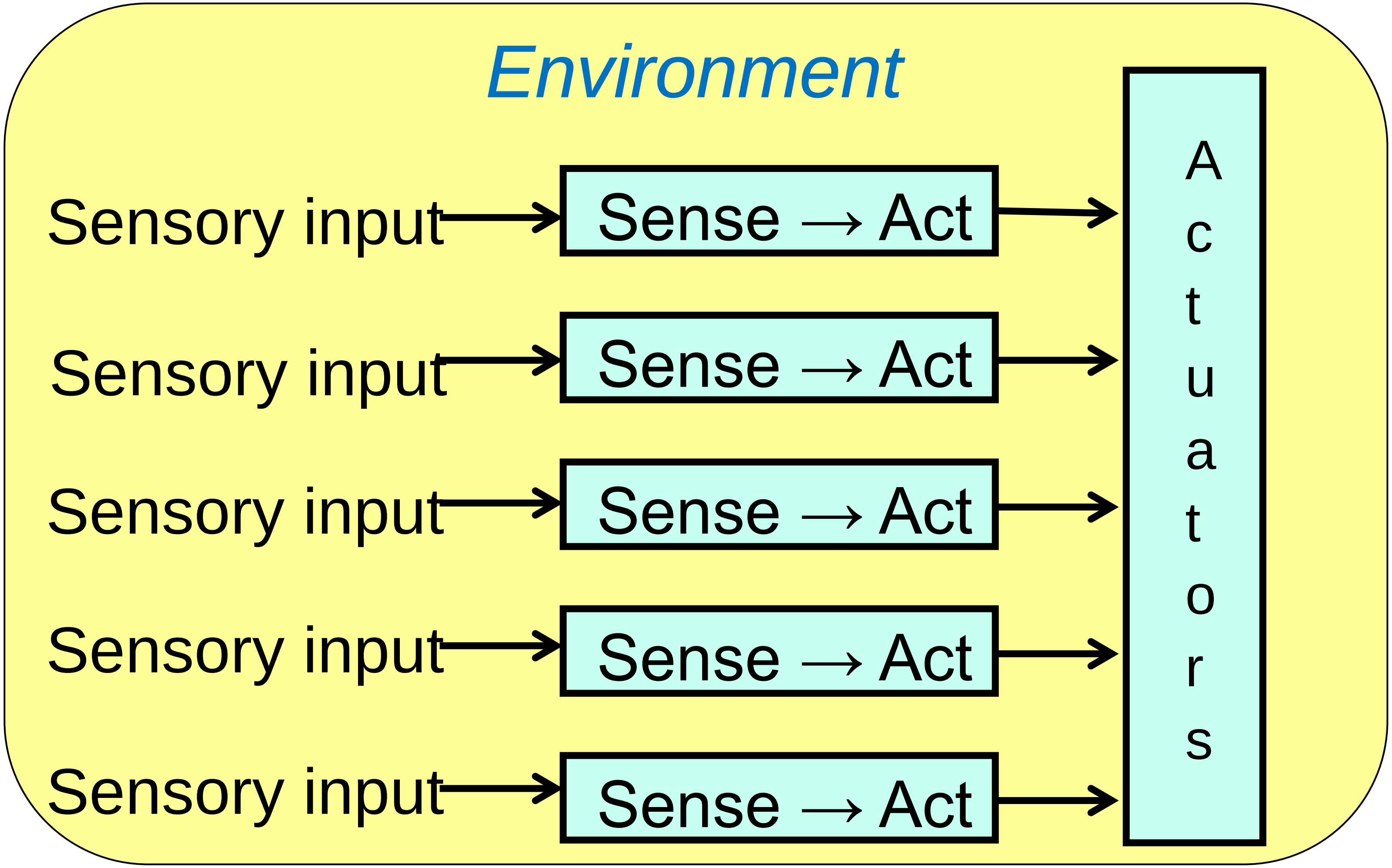
Reactive control (1)

- 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 → good at one or two tasks.
- Don’t pass representations between modules.

Reactive architecture diagram



Concurrent task-achieving modules

Example: The wall following (a)

- Left, right whisker – on/off.

Algorithm:

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

Limitations of reactive control

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

Example: The wall following (b)

Robot in a corner → oscillations.
What to do about oscillations?

1. Use a little randomness.
   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.

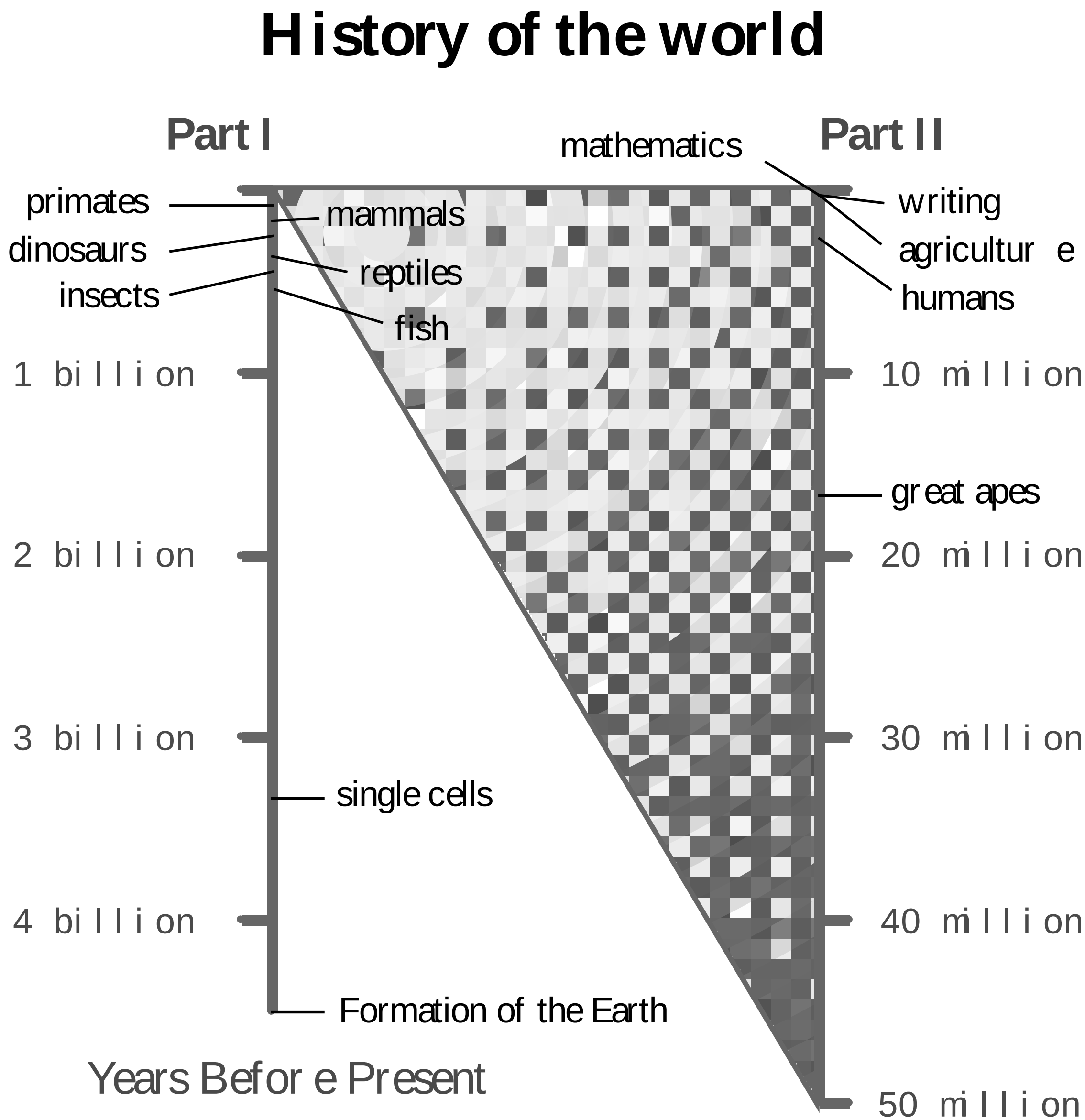
Behavior-based control

- Overall controller composed of two parts
  - Task achieving controllers
  - Arbitrating controller (also task specific)
- Controllers:
  - Are reactive (map perception to action)
  - Have no models, no planning
  - Are only capable of performing one task each

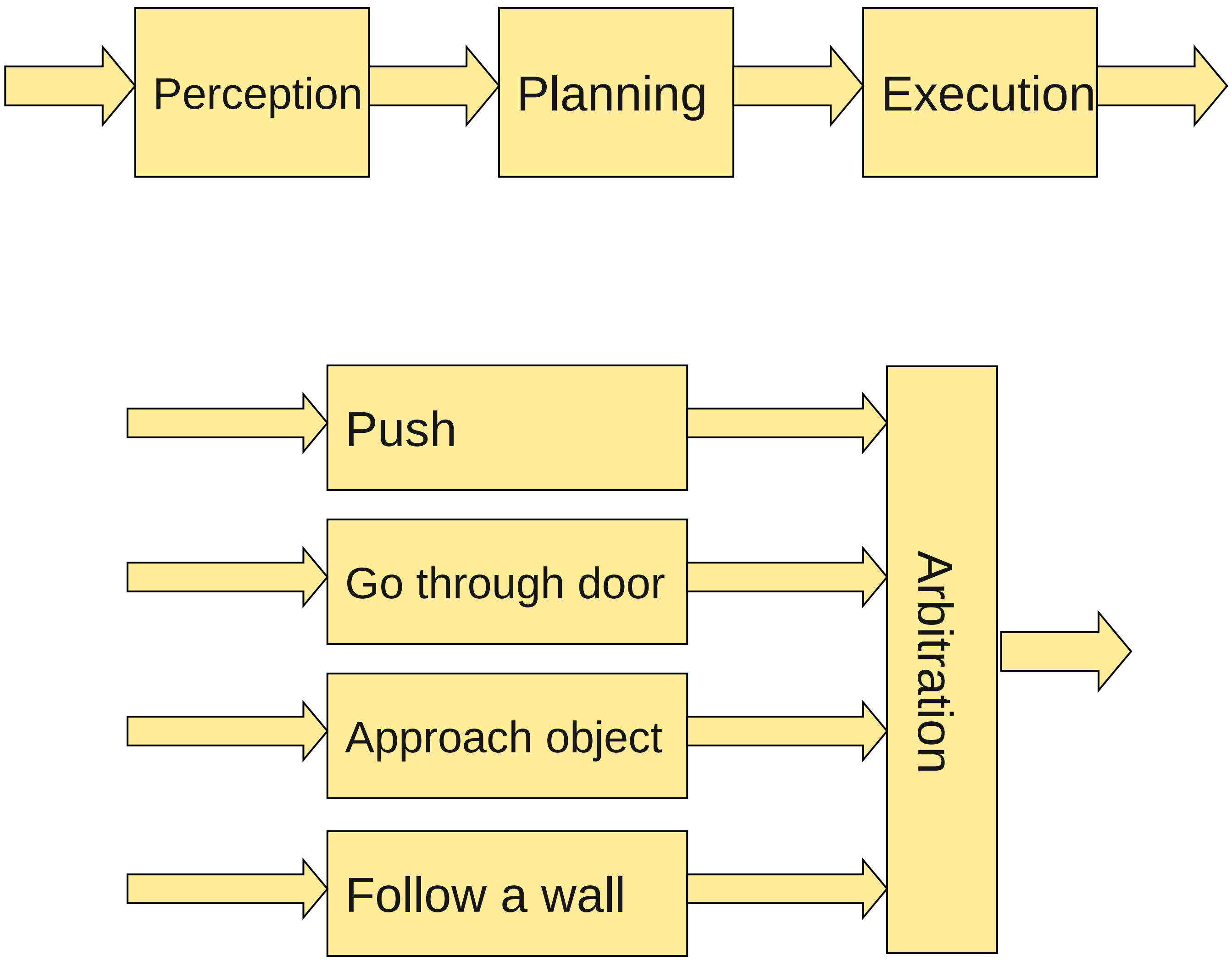
Biological Inspiration

- 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.

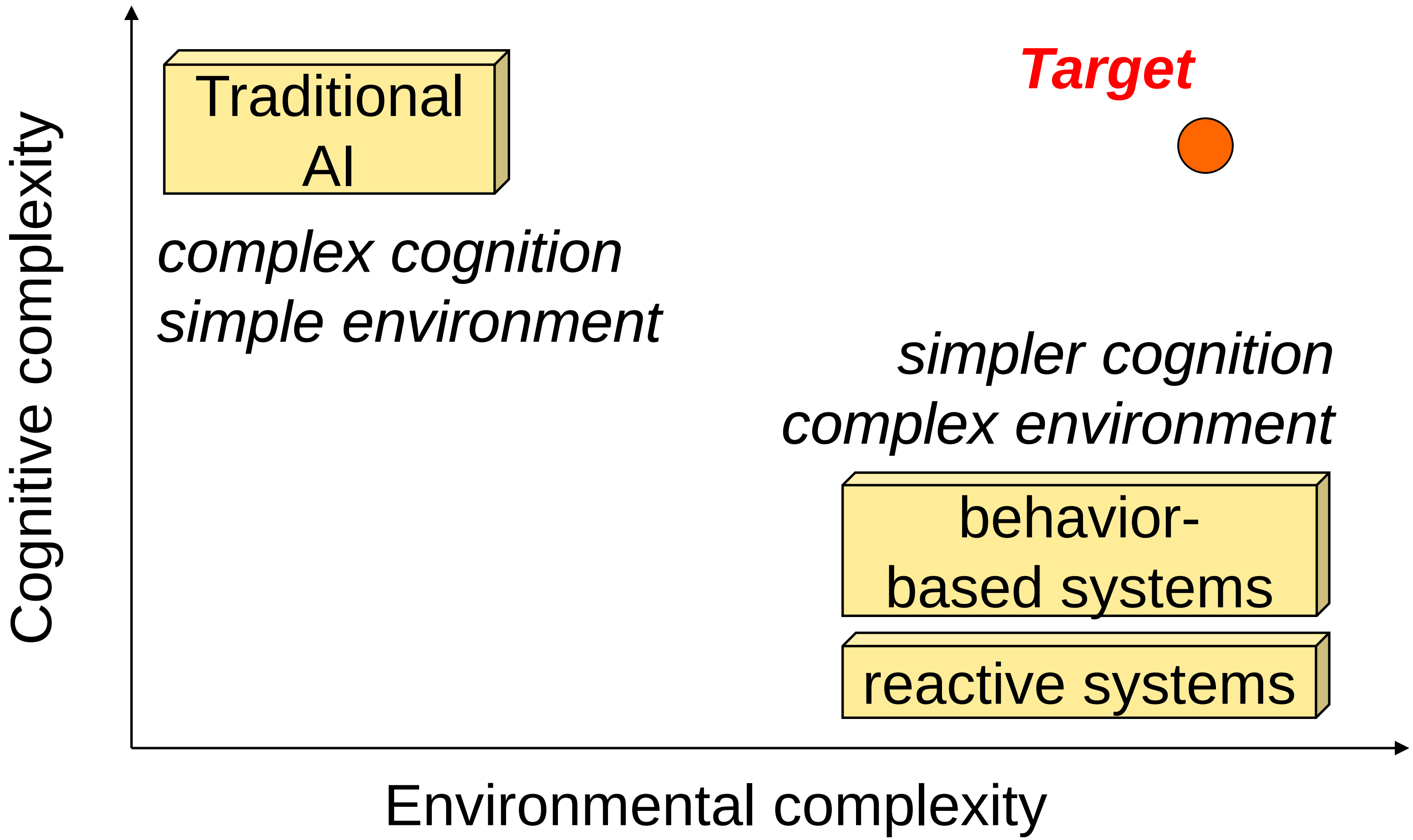
Where did evolution spent its time?



A comparison



Task / environment complexity



Eye, brain, hand: Rules

Eyes:

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

a) Yes/No questions, e.g. “Is there a box on the floor?”

b) How many? e.g. “How many boxes are in the room?”

c) Spatial relationship questions ,e.g. “Where is the nearest box to hand?” – “3 metres in front of hand”.

Eye, brain, hand: Rules

Brain:

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

a) Give instructions to Hand.

b) Ask questions (yes/no, how many, how far, where) of Eyes.

Eye, brain, hand: Rules

Hand:

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

Otherwise do what brain tells you.

Behaviour based system: Rules

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.

Behaviour based system: Rules

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.

Behaviour based system: Rules

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.

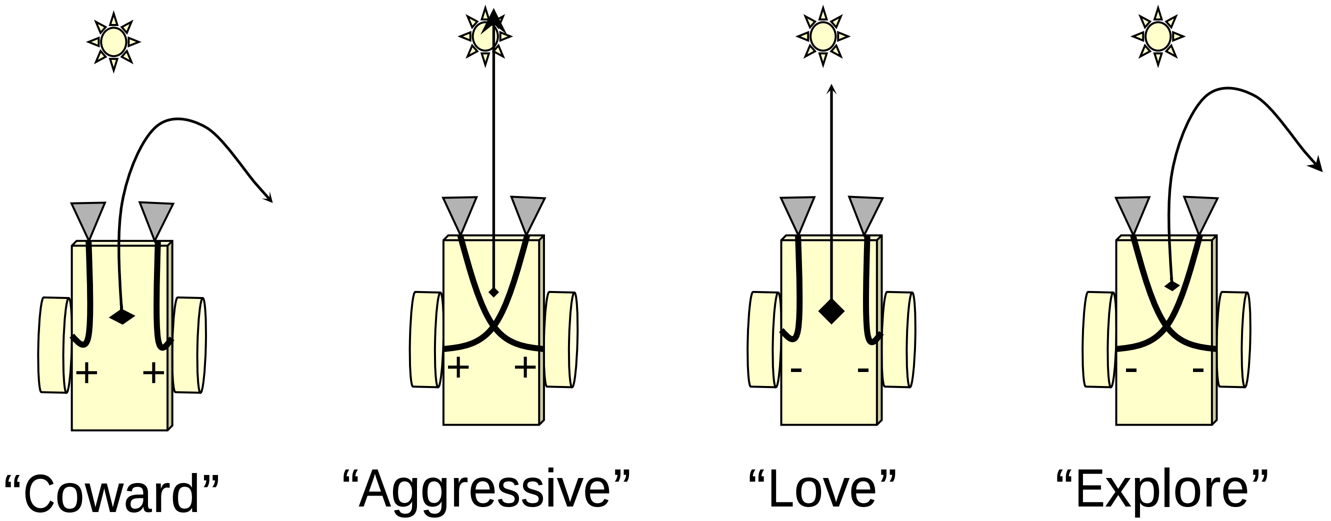
Rule 4: If you are holding an object, place it carefully on top of the object(s) in front of you.

Behaviour 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.

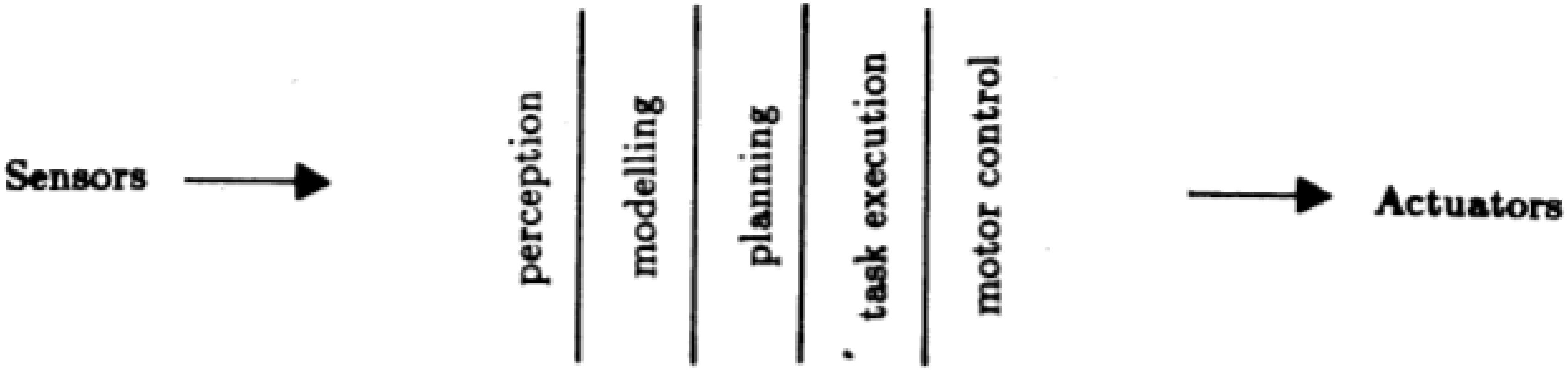


Braitenberg vehicles: differential drive mobile robots with two light sensors.

Rodney Brooks

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

A traditional decomposition of a robot control into functional modules



From Brooks, “A Robust Layered Control System for a Mobile Robot”, 1985.

Brooks – behavior languages

Rodney Brooks has put forward 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.

Brooks – behavior languages

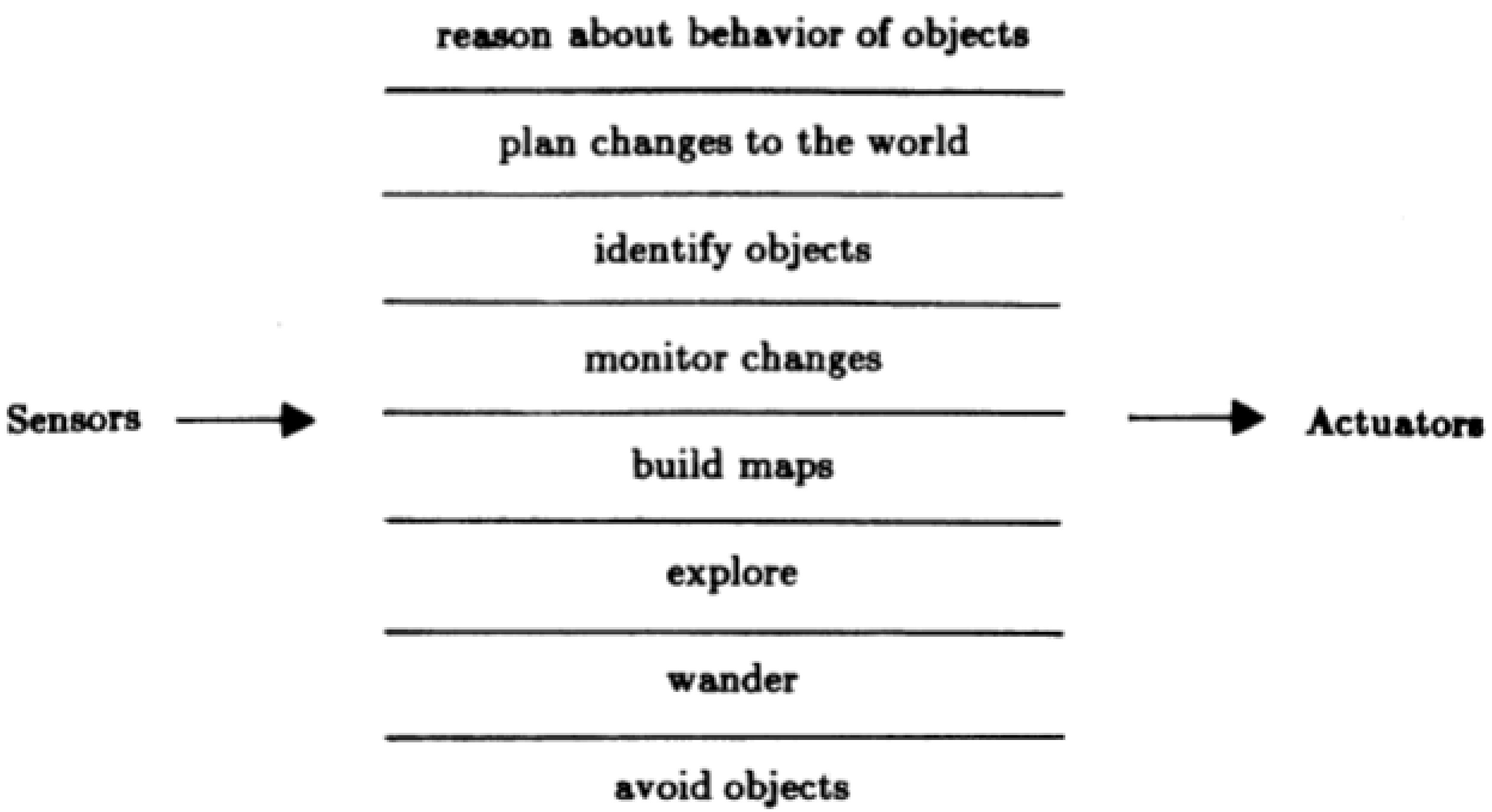
He identifies two key ideas that have informed his research:

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.

Brooks – behavior languages

- 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.

A decomposition of a robot control system based on task achieving behaviors



From Brooks, “A Robust Layered Control System for a Mobile Robot”, 1985

Advantages of reactive agents

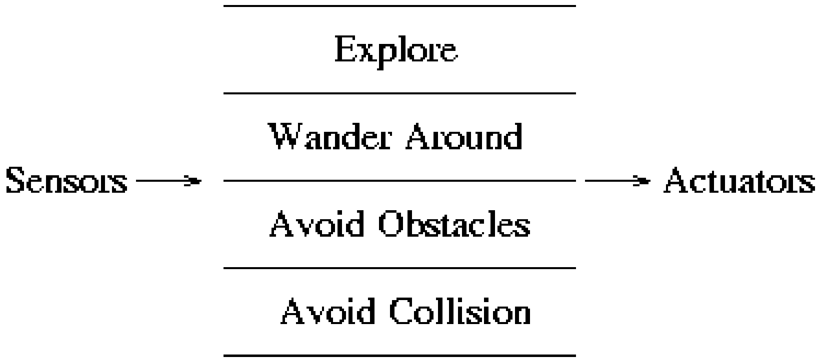
- Simplicity
- Economy
- Computational tractability
- Robustness against failure
- Elegance

Limitations of reactive agents

- Agents without environment models must have sufficient information available from local environment.
- If decisions are based on *local* environment, how does it take into account *non-local* information (i.e., it has a "short-term" view).
- Difficult to make reactive agents that learn.
- Since behavior emerges from component interactions plus environment, it is hard to see how to *engineer* specific agents (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.

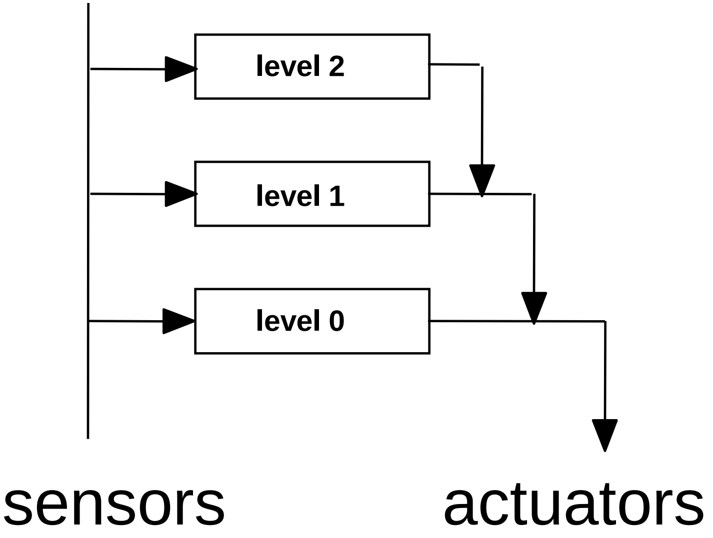


Designing in subsumption

- 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.

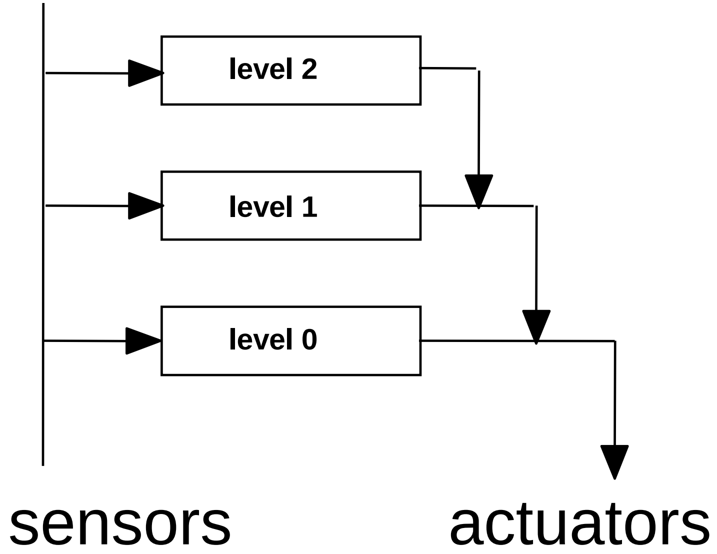
Subsumption layers

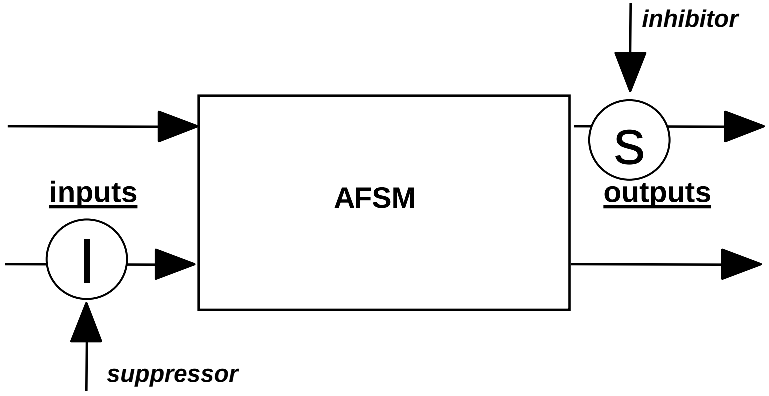
- First, we design, implement and debug layer 0.
- Next, we design layer 1
  - When layer 1 is designed, layer 0 is taken into consideration and utilized, its existence is subsumed (thus the name of the architecture) .
  - As layer 1 is added, layer 0 continues to function.
- Continue designing layers,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



Suppression and Inhibition

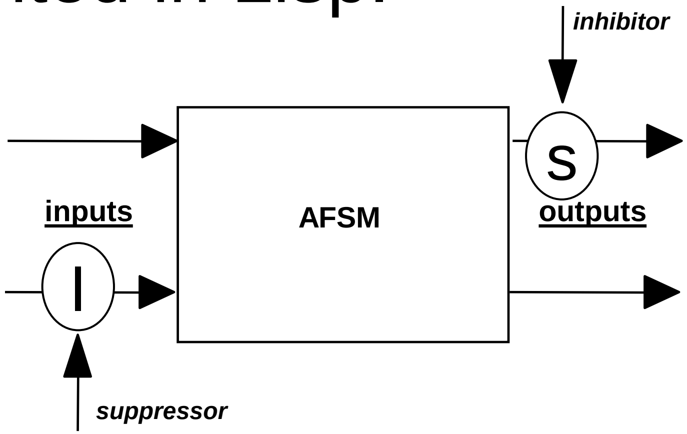
- Higher layers can disable the ones below
  - Avoid-obstacles can stop the robot from moving around
- Layer 2 can either:
  - Inhibit the output of level 1 or
  - Suppress the input of level 1
- The process is continued all the way to the top level





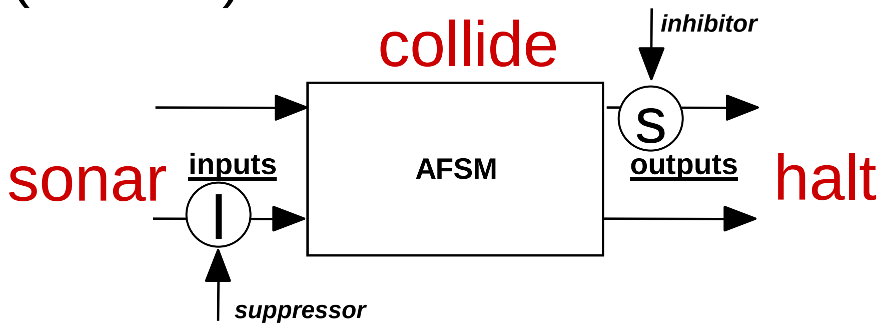
Subsumption Language and AFSMs

- 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

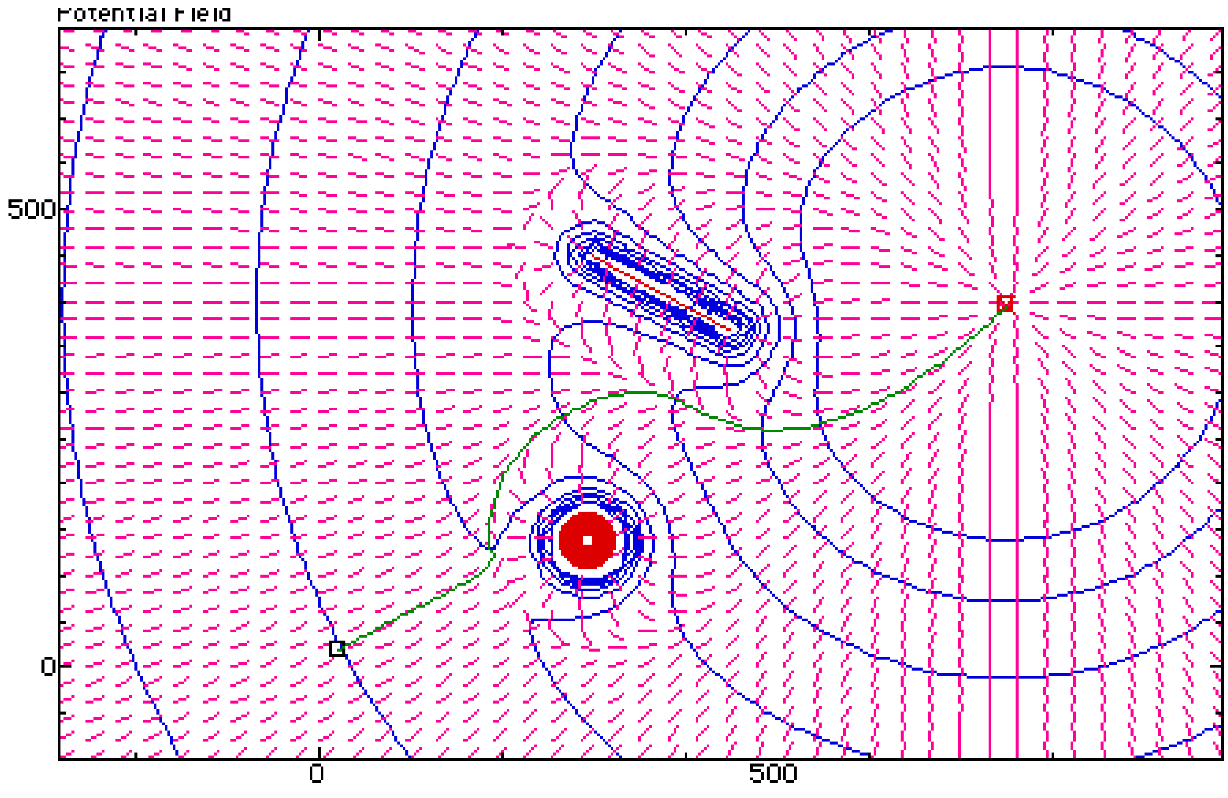
- Each behavior is represented as an augmented finite state machine (AFSMs)
- Stimulus (input) or response (output) can be inhibited or suppressed by other active be behaviors.



- 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.

Reactive system using Potential Fields

Create repulsion force around obstacles plus attraction force to the goal



Behavior-based architecture

- An alternative to hybrid systems, which has similar capabilities:
  - the ability to act reactively,
  - the ability to act deliberatively.
- There is no intermediate layer.
- A unified, consistent representation is used in the whole system $\Rightarrow$ concurrent behaviors.
- That resolves issues of time-scale.

Lecture outline

- Intelligent (cognitive) robotics
- Control architectures, general idea
- Cognitive robot architectures
  - Deliberative (sense → plan → act), since 1960’
  - Reactive
    - Pure reactive
    - Behavior-based
    - Subsumption architecture, R. Brooks 1986
  - Hybrid, since ~1995

Hybrid architecture (1)

- Combine the two extremes:
  - reactive system on the bottom,
  - deliberative system on the top,
  - connected by some *intermediate layer.*
- Called also 3-layer systems.
- Layers must operate concurrently
- Different representations and time-scales between the layers.

Hybrid Architectures

- Many researchers have argued that neither a completely deliberative nor completely reactive approach is suitable for building agents.
- They have suggested using *hybrid* systems, which attempt to marry classical and alternative approaches.
- An obvious approach is to build an agent out of two (or more) subsystems:
  - a *deliberative* one, containing a symbolic world model, which develops plans and makes decisions in the way proposed by symbolic AI.
  - a *reactive* one, which is capable of reacting to events without complex reasoning.

Hybrid Architectures

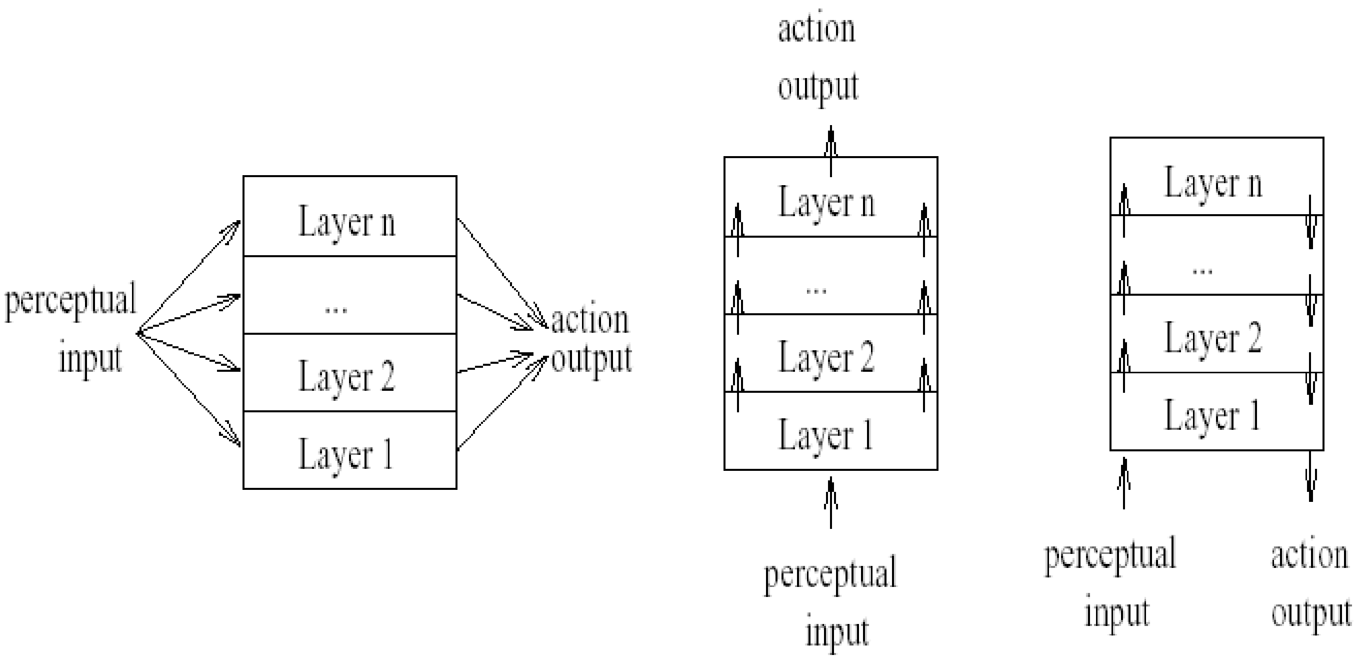
- Often, the reactive component is given some kind of precedence over the deliberative one.
- This kind of structuring leads naturally to the idea of a *layered* architecture, of which TOURINGMACHINES and INTERRAP are examples.
- In such an architecture, an agent's control subsystems are arranged into a hierarchy, with higher layers dealing with information at increasing levels of abstraction.

Hybrid Architectures

- A key problem in such architectures is what kind of control framework to embed the agent's subsystems in, 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 at most one layer each

Hybrid Architectures

m possible actions suggested by each layer, *n* layers



(a) Horizontal layering

(b) Vertical layering (One pass control)

(c) Vertical layering (Two pass control)

m^n interactions

$m^2(n-1)$ interactions

Not fault tolerant to layer failure

Introduces bottleneck in central control system