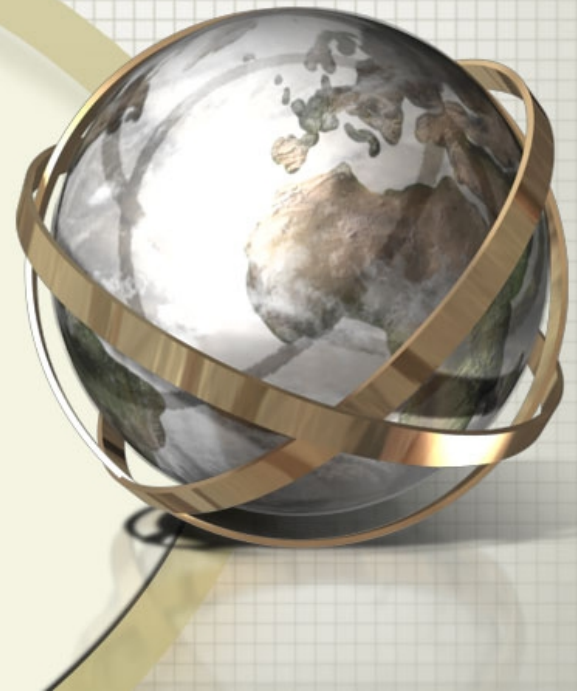


Subsumption Architectures

Support to Lecture 6



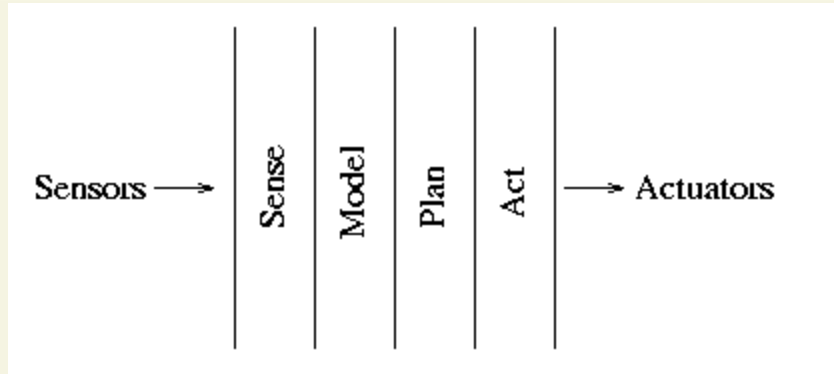
Overview

- Reactive control
- Complete control space
- *Action selection*
- The subsumption architecture
 - ♦ *Vertical vs. horizontal decomposition*

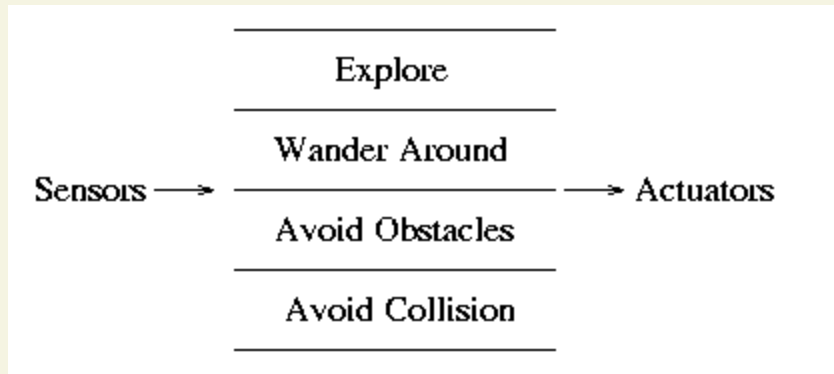


Vertical v. Horizontal Systems

*Traditional (SPA):
sense – plan – act*



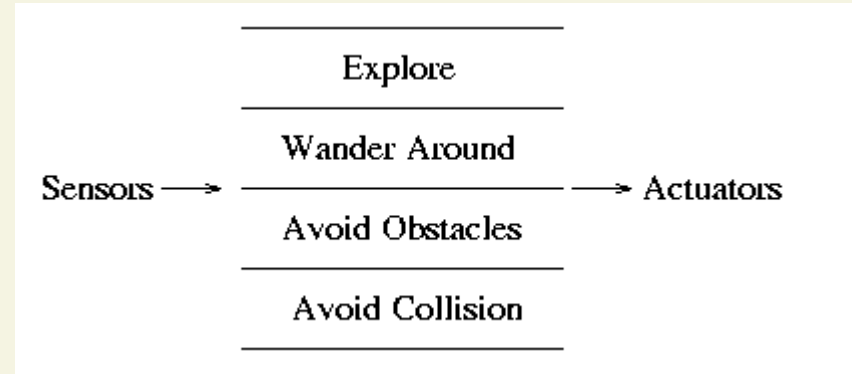
Subsumption:



The Subsumption Architecture

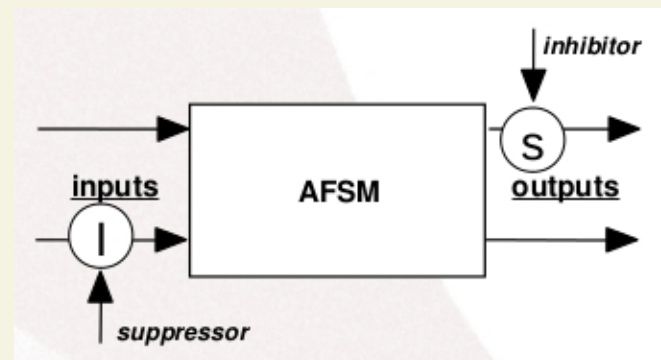
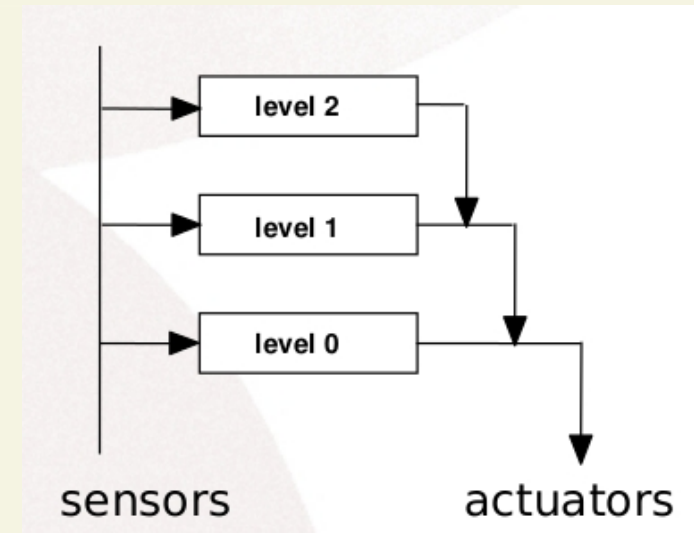
- *Principles of design*

- ♦ *systems are built from the bottom*
- ♦ *components are taskachieving*
- ♦ *actions/behaviors (avoidobstacles, find-doors, visitrooms)*
- ♦ *components are organized in layers, from the bottom*
- ♦ *lowest layers handle most basic tasks*
- ♦ *all rules can be executed in parallel, not in a sequence*
- ♦ *newly added components and layers exploit the existing ones*



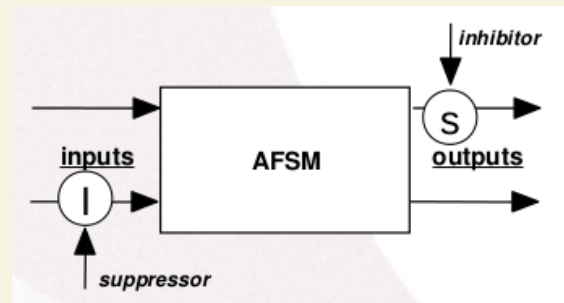
Subsumption Layers

- 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
 - ♦ Layer 0 continues to function
- Continue designing layers, until the desired task is achieved
- Higher levels can
 - ♦ Inhibit outputs of lower levels
 - ♦ Suppress inputs of lower levels



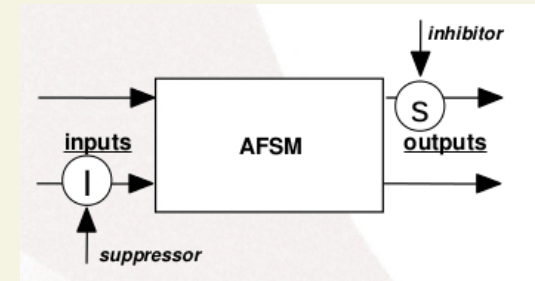
Subsumption Language and AFSMs

- 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



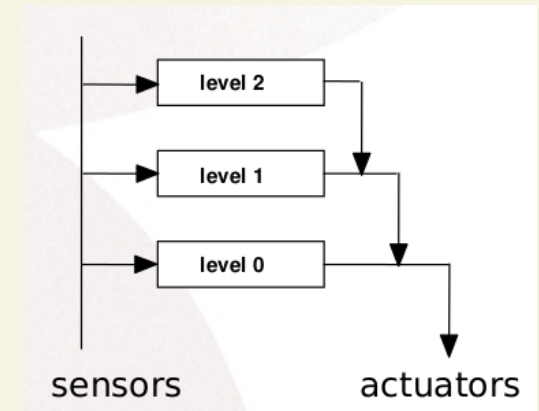
Subsumption Language and AFSMs

- Each **behavior** is represented as an *augmented finite state machine* (AFSMs)
- Stimulus (input) or response (output) can be *inhibited* or *suppressed* by other active 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 with communication wires*, which pass input and output messages between them; only the last message is kept
- AFSMs run *asynchronously*



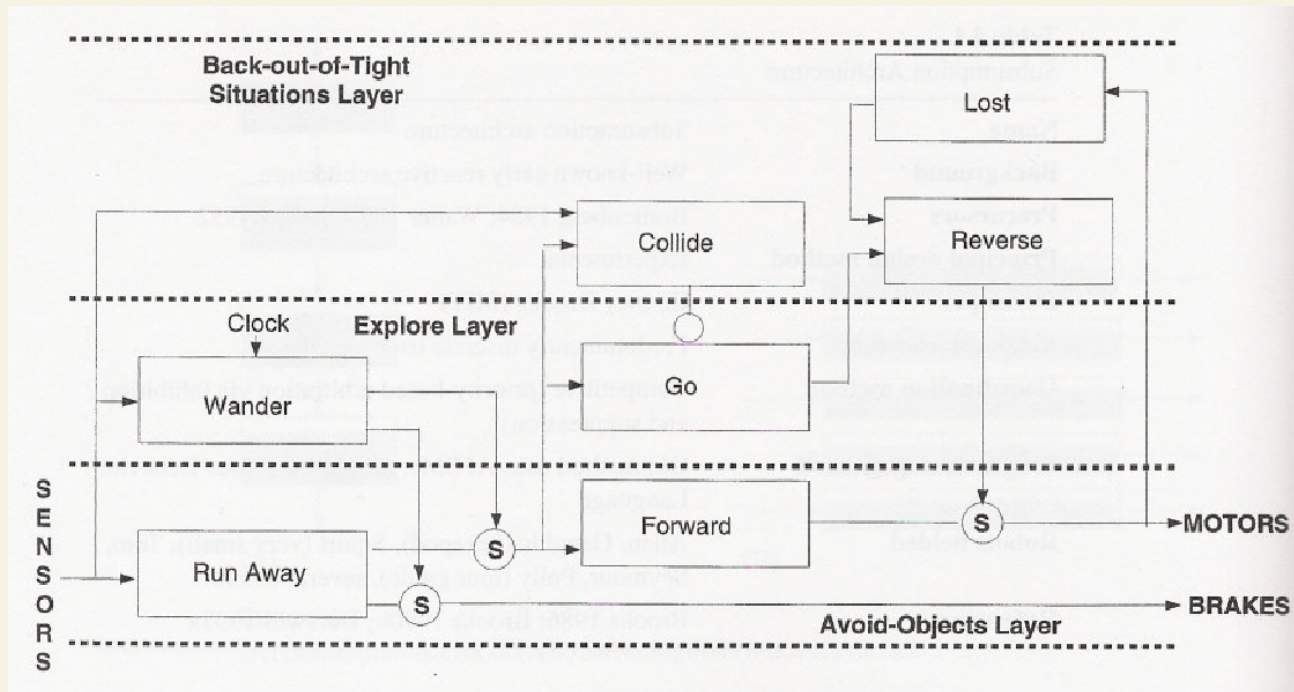
Networks of AFSMs

- Layers represent *task achieving behaviors*
 - ♦ Wandering, avoidance, goal seeking
- Layers work *concurrently and asynchronously*
- A Subsumption Architecture controller, using the AFSM-based programming language, is a *network of AFSMs divided into layers*
- Convenient for *incremental system design*



Wandering in Subsumption

- Brooks '87



Layering in AFSM Networks

- Layers modularize the reactive system
- Bad design:
 - ❖ putting a lot of behaviors within a single layer
 - ❖ putting a large number of connections between the layers, so that they are strongly coupled
- Strong coupling implies dependence between modules, which violates the modularity of the system
- If modules are interdependent, they are not as robust to failure
- In Subsumption, if higher layers fail, the lower ones remain unaffected



Module Independence

- Subsumption has one-way independence between layers
 - ✦ With upward independence, a higher layer can always use a lower one by using suppression and inhibition
- Two-way independence is not practical
 - ✦ No communication between layers is possible
- Do we always have to use these wires to communicate between parts of the system?



Sequencing in Subsumption

- How can you sequence activities in Subsumption?
- Coupling between layers need not be through the system itself (i.e., not through explicit communication wires)
- It could be through the world. How?



Communication Through the World

- Collecting soda cans: Herbert
- Herbert's capabilities
 - ◆ Move around without running into obstacles
 - ◆ Detect soda cans using a camera and a laser
 - ◆ An arm that could: extend, sense if there is a can in the gripper, close the gripper, tuck the arm in



- Look for soda cans, when seeing one approach it
- When close, extend the arm toward the soda can
- If the gripper sensors detect something, close the gripper
- If can is heavy, put it down, otherwise pick it up
- If gripper was closed, tuck the arm in and head home
- The robot did not keep internal state about what it had just done and what it should do next: it just sensed!



More on Herbert

- There is no internal wire between the layers that achieve can finding, grabbing, arm tucking, and going home
- However, the events are all executed in proper sequence. Why?
- Because the relevant parts of the control system interact and activate each other through sensing the world



World as the Best Model

- This is a key principle of reactive systems & Subsumption Architecture:
 - ◆ Use the world as its own best model!
- If the world can provide the information directly (through sensing), it is best to get it that way, than to store it internally in a representation (which may be large, slow, expensive, and outdated)



Subsumption System Design

- What makes a Subsumption Layer, what should go where?
- There is no strict recipe, but some solutions are better than others, and most are derived empirically
- How exactly layers are split up depends on the specifics of the robot, the environment, and the task



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



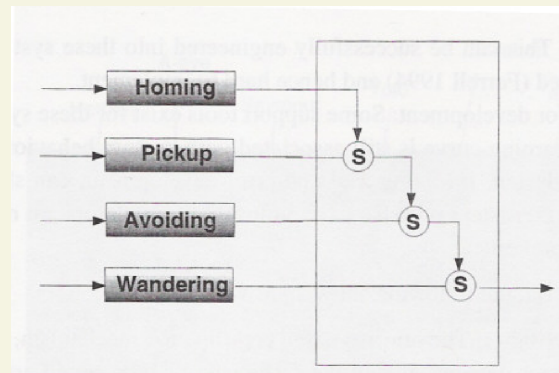
Genghis (MIT)

- Walk over rough terrain and follow a human (Brooks '89)
 - *Standup*
 - Control leg's swing position and lift
 - *Simple walk*
 - *Force balancing*
 - Force sensors provide information about the ground profile
 - *Leg lifting*: step over obstacles
 - *Obstacle avoidance* (whiskers)
 - *Pitch stabilization*
 - *Prowling*
 - *Steered prowling*

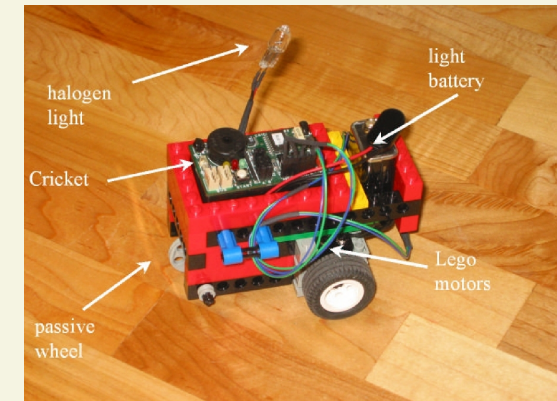
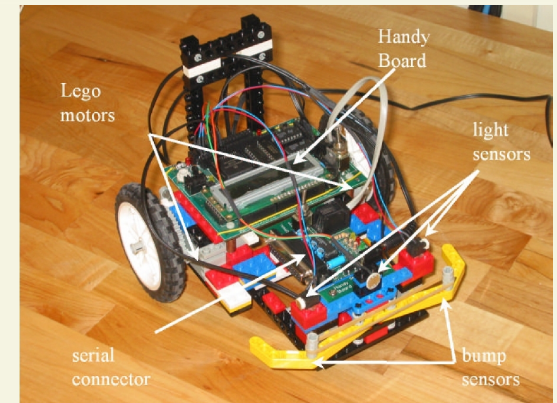
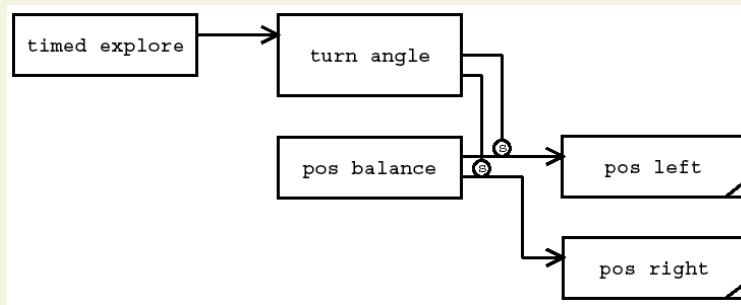
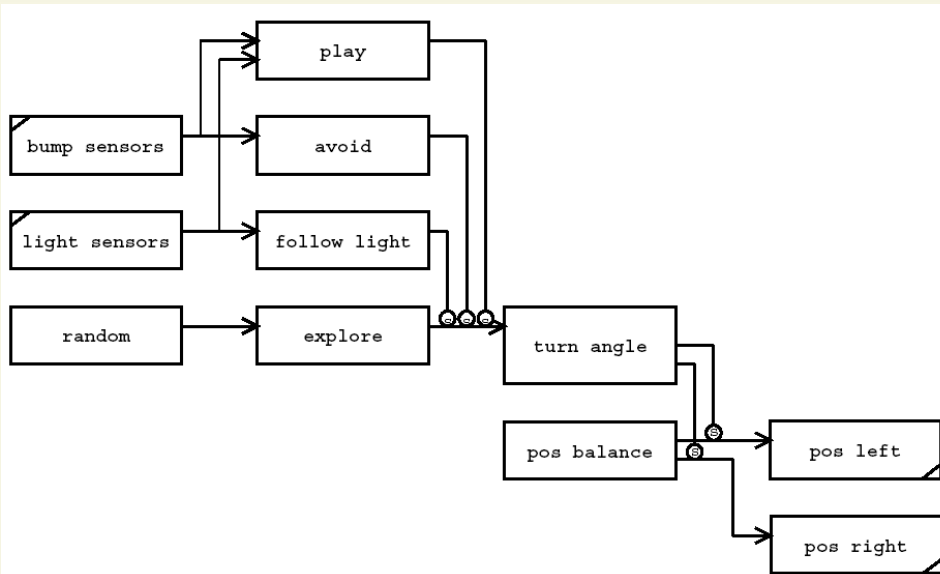


The Nerd Herd (MIT)

- *Foraging example* (Mataric '93)
 - ❖ R1 robots (IS robotics)
- Behaviors involved:
 - ❖ *Wandering*
 - ❖ *Avoiding*
 - ❖ *Pickup*
 - ❖ *Homing*



Tom and Jerry (MIT)



Pros and Cons

- Some critics consider the *lack of detail* about designing layers to be a *weakness of the approach*
- Others feel it is a strength, *allowing for innovation and creativity*
- *Subsumption has been used on a vast variety of effective implemented robotic systems*
- *It was the first architecture to demonstrate many working robots*



Benefits of Subsumption

- *Systems are designed incrementally*
 - ◆ Avoid design problems due to the complexity of the task
 - ◆ Helps the design and debugging process
- *Robustness*
 - ◆ If higher levels fail, the lower ones continue unaffected
- *Modularity*
 - ◆ Each “competency” is included into a separate layer, thus making the system manageable to design and maintain
 - ◆ Rules and layers can be reused on different robots and for different tasks



Behavior-Based Control

- **Reactive systems**

- ✦ too inflexible, use no representation, no adaptation or learning

- **Deliberative systems**

- ✦ Too slow and cumbersome

- **Hybrid systems**

- ✦ Complex interactions among the hybrid components

- *Behavior-based control involves the use of “behaviors” as modules for control*

