Control Arbitration

Oct 12, 2005

RSS II

Una-May O'Reilly

Agenda

I. Subsumption Architecture as an example of a behavior-based architecture. Focus in terms of how control is arbitrated

II. Arbiters and arbitration in general

III. Alternative (and more complex) Arbiters

Creature, or Behavior-Based, AI

creatures -- **live in messy worlds**
performance relative to the world
intelligence (emerges) on this substrate

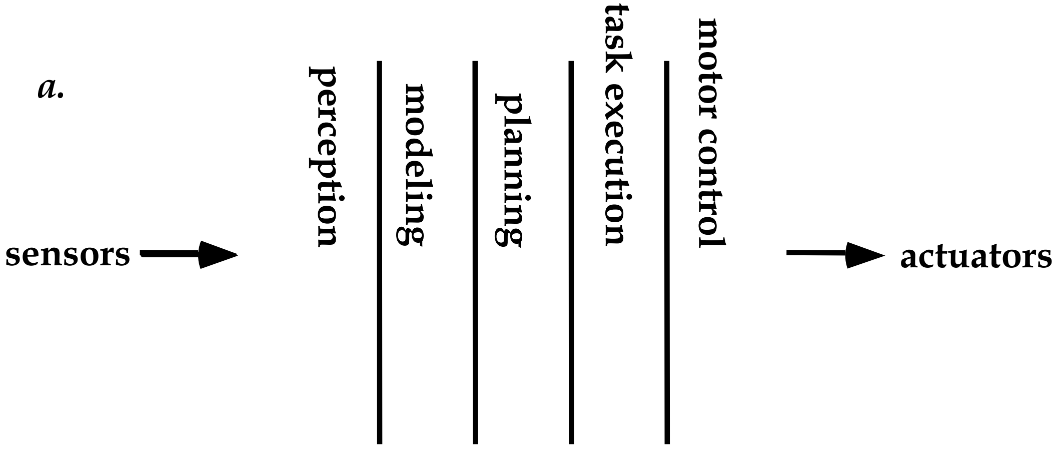
the **creature** → ***all*** **possible worlds**

Photo courtesy of Rodney Brooks, MIT CSAIL.

maintain **goals**

explore, survive

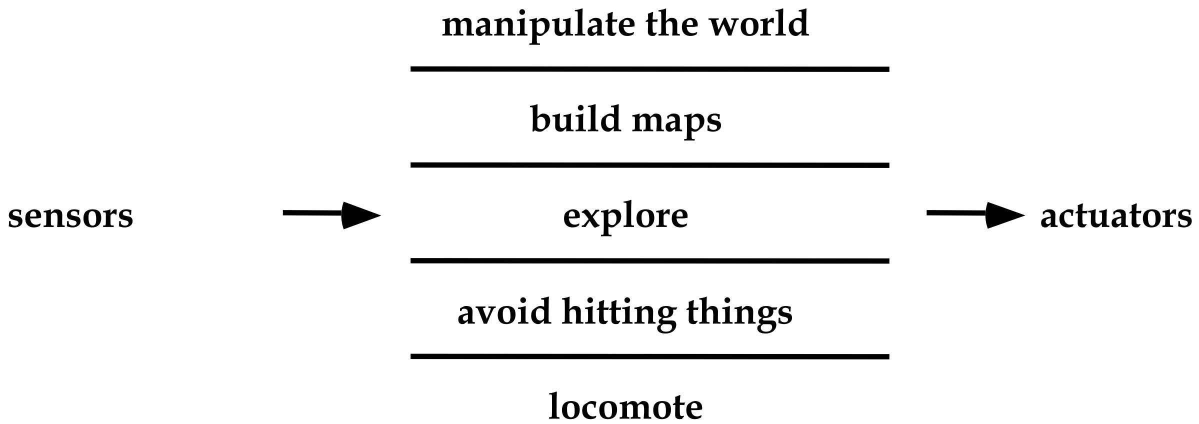
Traditional Problem Decomposition



Horizontal decomposition

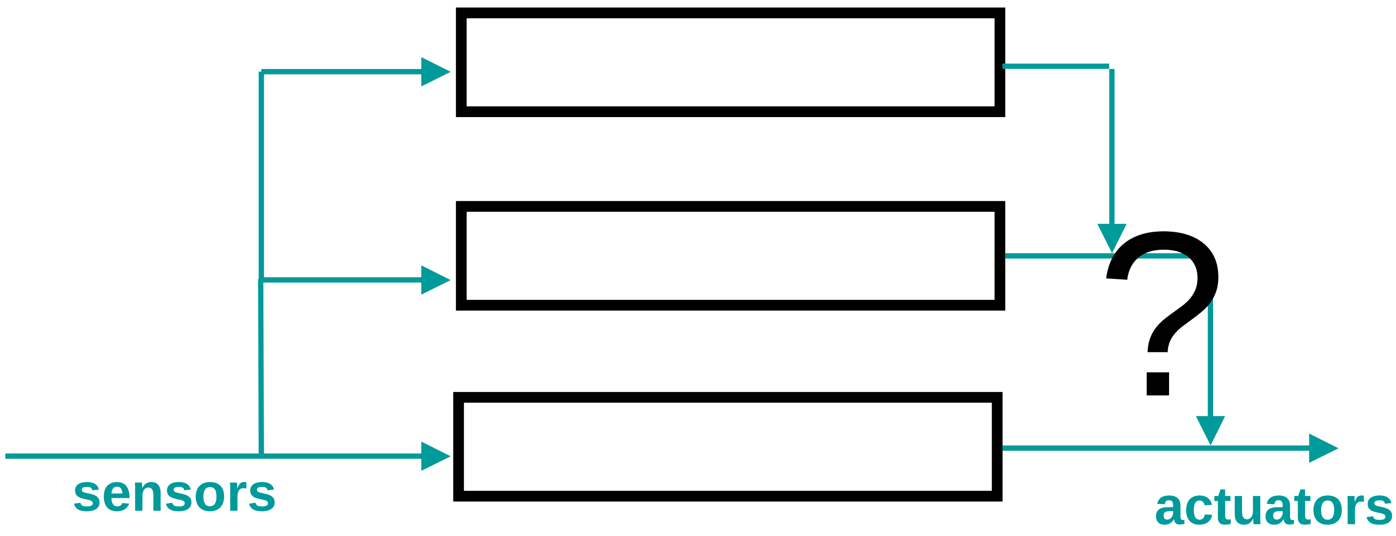
Behavior Based Decomposition

nouvelle



Vertical decomposition

How to Arbitrate



- each layer has some perception, ‘planning’, and action
- rather than sensor fusion, we have behavior fusion
- fusion happens at the action command level on the right
- there is a question of what sort of merge semantics there should be
- Some kind of arbitration is required

Suitable for Mobile Robots

- Handles multiple goals via different behaviors, with mediation, running concurrently
- Multiple sensors are not combined but complementary
- Robust: graceful degradation as upper layers are lost
- Additivity facilitates easy expansion for hardware resources

Eye Candy: Subsumption Robots

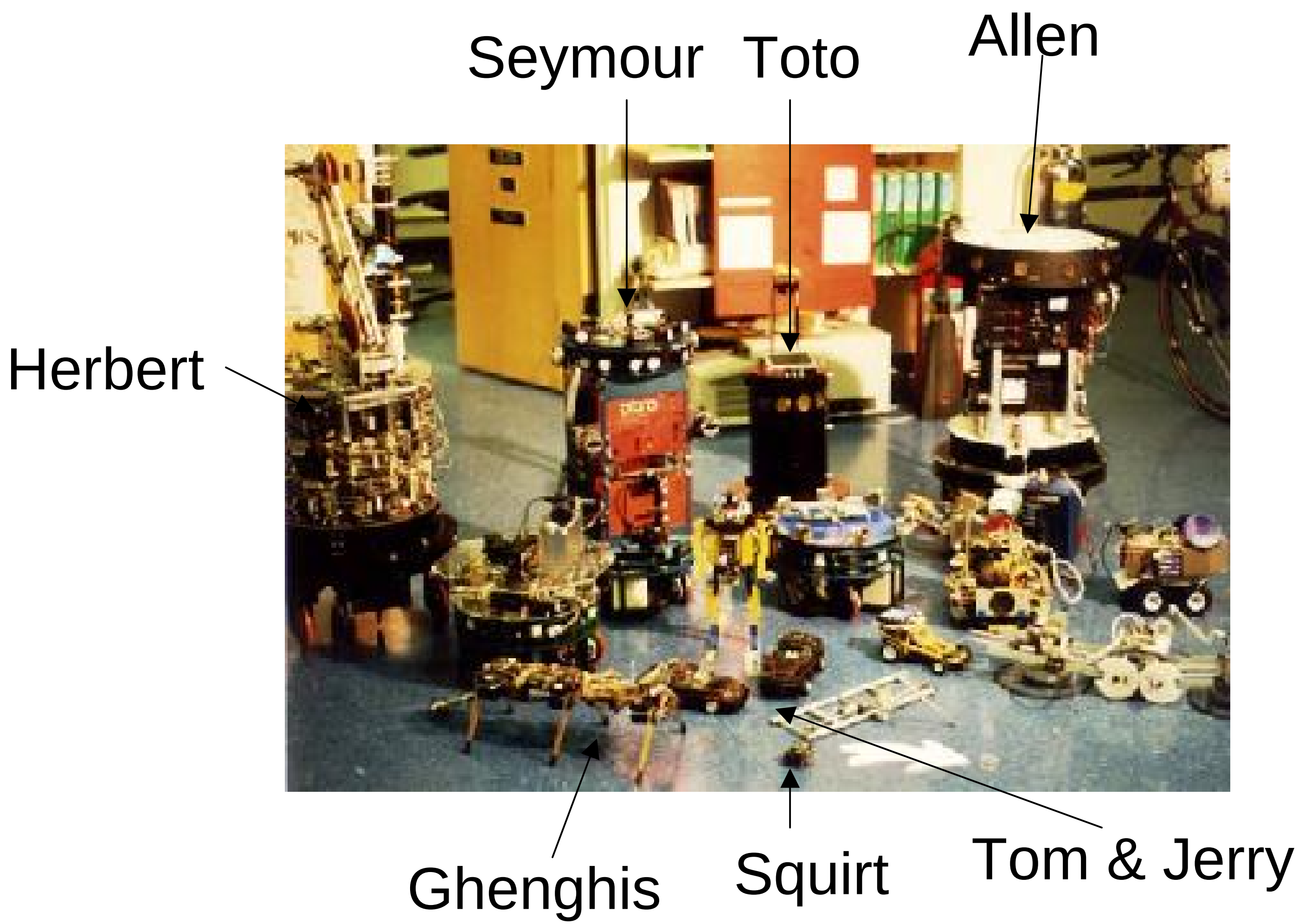


Photo courtesy of MIT MOBOT lab.

Subsumption Robots

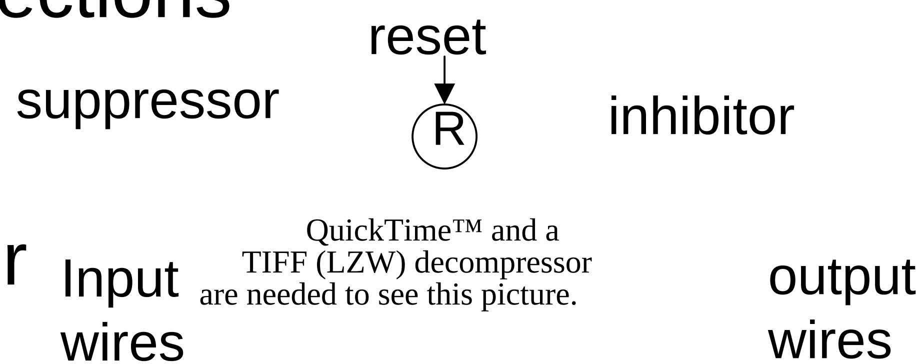
- Allen: oldest, sonar-based navigation
- Tom and Jerry: I/R proximity sensors on small toy car
- Genghis and Attila: 6-legged hexapods, autonomous walking
- Squirt: 2 oz robot responding to light
- Toto: map-construction robot, first to use Behaviour Language
- Seymour: visual, motion tracking robot
- Polly: robotic tour guide for the AI Lab

Subsumption Architecture

- Task achieving behaviors are represented in separate layers
- Individual layers work on individual goals concurrently and asynchronously
- No global memory, bus or clock
- Lowest level description of a behavior is an Augmented Finite State machine

AFSM to represent behavior

- Augmented
  - Registers, internal timer
- FSM: situation-action response:
  - Considers sensor filter, trigger, commands out
- Input and output connections
  - Suppressor
  - Inhibitor
- External reset timer for subsumption
- Later compiled via:
  - Behavior language

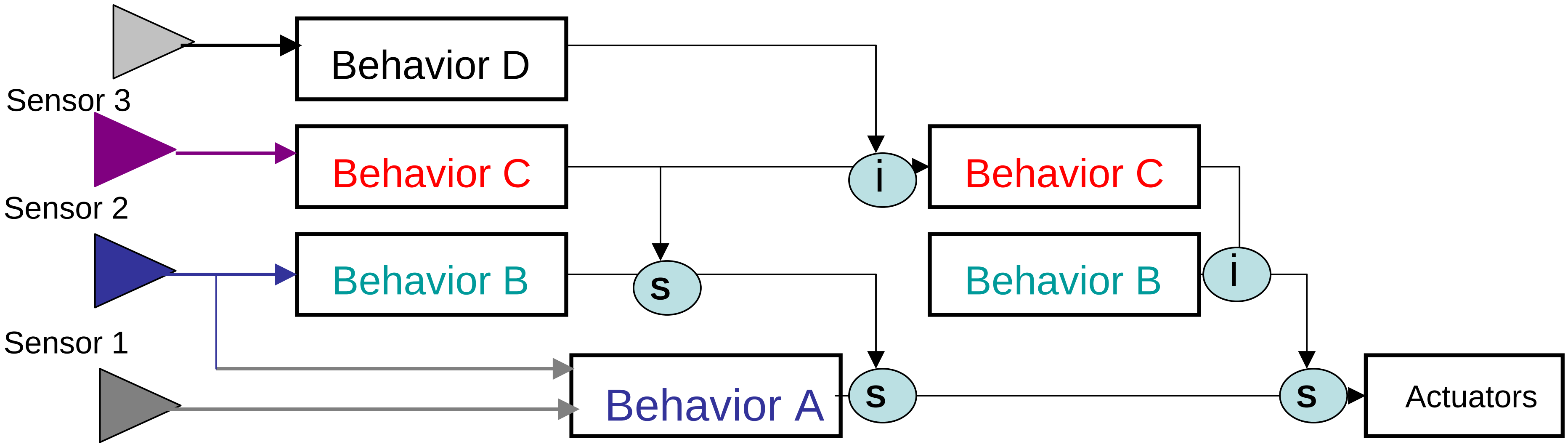


Connecting behaviors

- Concept of wire with sources and destinations
- Principle is: transfer of information between behaviors MUST be explicit in terms of
  - Who can change the info (SOURCES)
  - Who can access the info (DESTINATIONS)
- If connections are implemented as messages in Carmen publish/subscribe framework, MUST ensure abstraction violations of this sort are avoided.

  How?: design enforcement

Subsumption Architecture one layer



From p 94, Robot Programming, A Practical Guide to BB Robotics, Joseph L. Jones.

Suppressor node: eliminates lower level control signal and replaces it with one from higher level. Suppression only occurs when higher level is active.

Inhibitor node: eliminates lower level control signal without any substitution

Subsumption Architecture:

QuickTime™ and a
TIFF (LZW) decompressor
are needed to see this picture.

From “A Colony Architecture for an Artificial Creature”, Jonathon Connell, MIT AI TR-1151.

Subsumption Architecture

- A (purely reactive) behavior-based method
- Sound-bites
  - The world is its own best model
    - No central world model or global sensor representations
  - Intelligence is in the eye of the observer
  - All onboard computation is important
  - Systems should be built incrementally
  - No representation. No calibration, no complex computation, no high bandwidth computation
  - Is there state in an AFSM?
    - external timer “micro plan”..later removed
    - Registers (variables), timer, sequence steps are quite constrained by constraints of special purpose language

Using an External Timer on the AFSM

- From Connell's thesis:

QuickTime™ and a
TIFF (LZW) decompressor
are needed to see this picture.

From "A Colony Architecture for an Artificial Creature", Jonathon Connell, MIT AI TR-1151.

Using an Internal Timer Retriggerable monostable

- From Connell's thesis:

QuickTime™ and a
TIFF (LZW) decompressor
are needed to see this picture.

From "A Colony Architecture for an Artificial Creature", Jonathon Connell, MIT AI TR-1151.

- For responding to events rather than situations (time intervals)
- Triggering events sets mode to true and timer runs (memory latch)
- Timer expiration resets mode
- Reset upon use
- Outdated info is discarded like built-in watchdog timer that reboots at regular intervals

Reconsidering some of the dogma

- Mataric's Toto
  - Plans as behaviors
  - World model is distributed, not necessary consistent, at different (task-based) abstractions
- (Connell): State must exist for exploitation of history (as memory), may help choices

- Connell's Herbert:
- More dogmatic about (no) state and module independence: all S nodes with I's as applicability predicate inside module
- Less dogmatic about layers "soup" rather than "stratified heap"
- Less dogmatic about evolutionary progression and hierarchy of priority

Herbert- J Connell

QuickTime™ and a
TIFF (LZW) decompressor
are needed to see this picture.

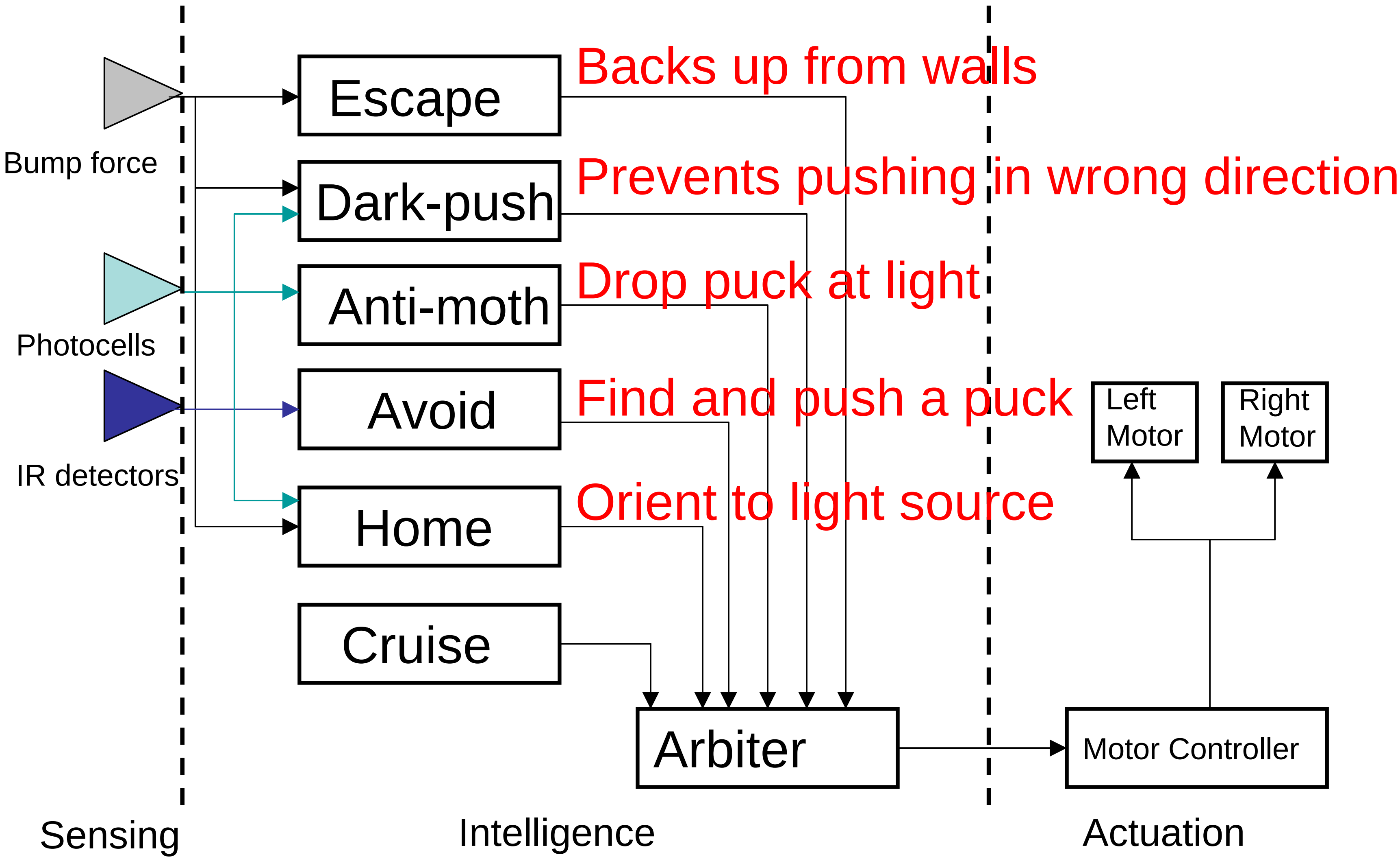
From “A Colony Architecture for an Artificial Creature”, Jonathon Connell, MIT AI TR-1151.

Subsumption Evaluated Practically

- Robust
- Modular
- Easy to tune each behavior
- But
  - Larger architectures are hard to decide priorities for
  - Robot may not take optimal path to goal

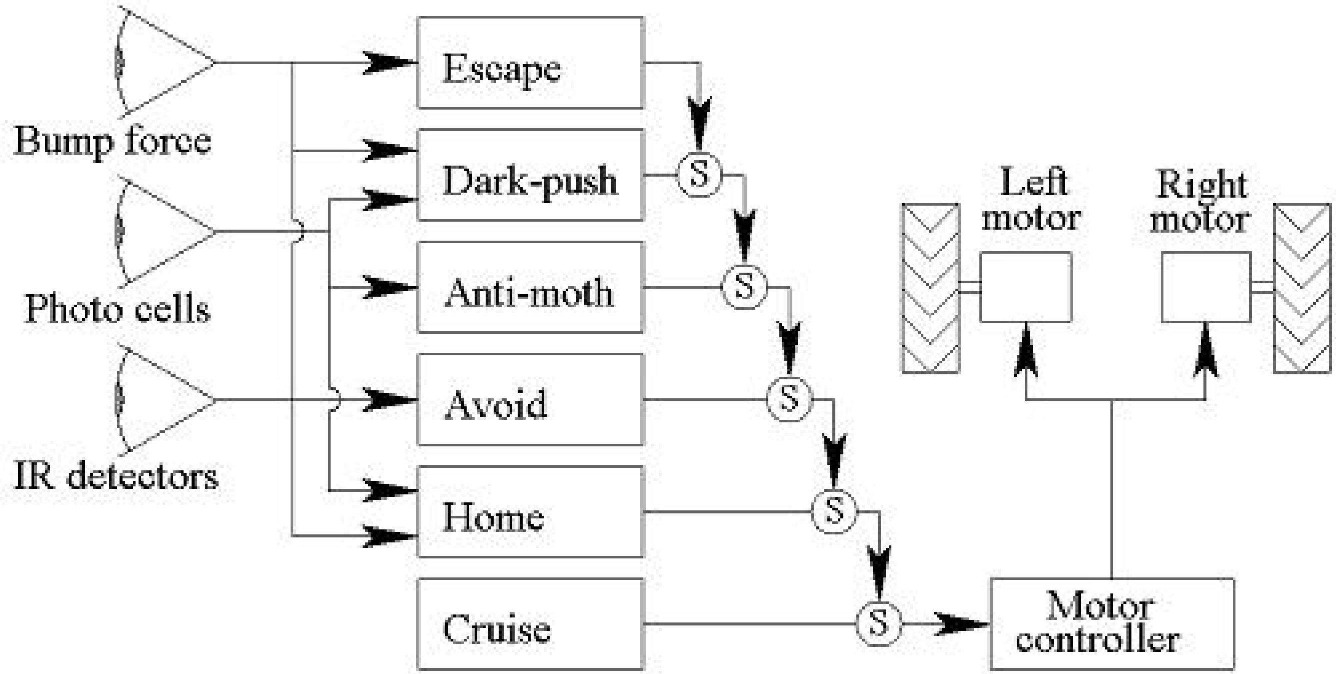
II. Arbitration in General

Collection Task Behavior Network



From Robot Programming, Joseph L. Jones, McGraw-Hill, 2004

Our Collection Task with Subsumption

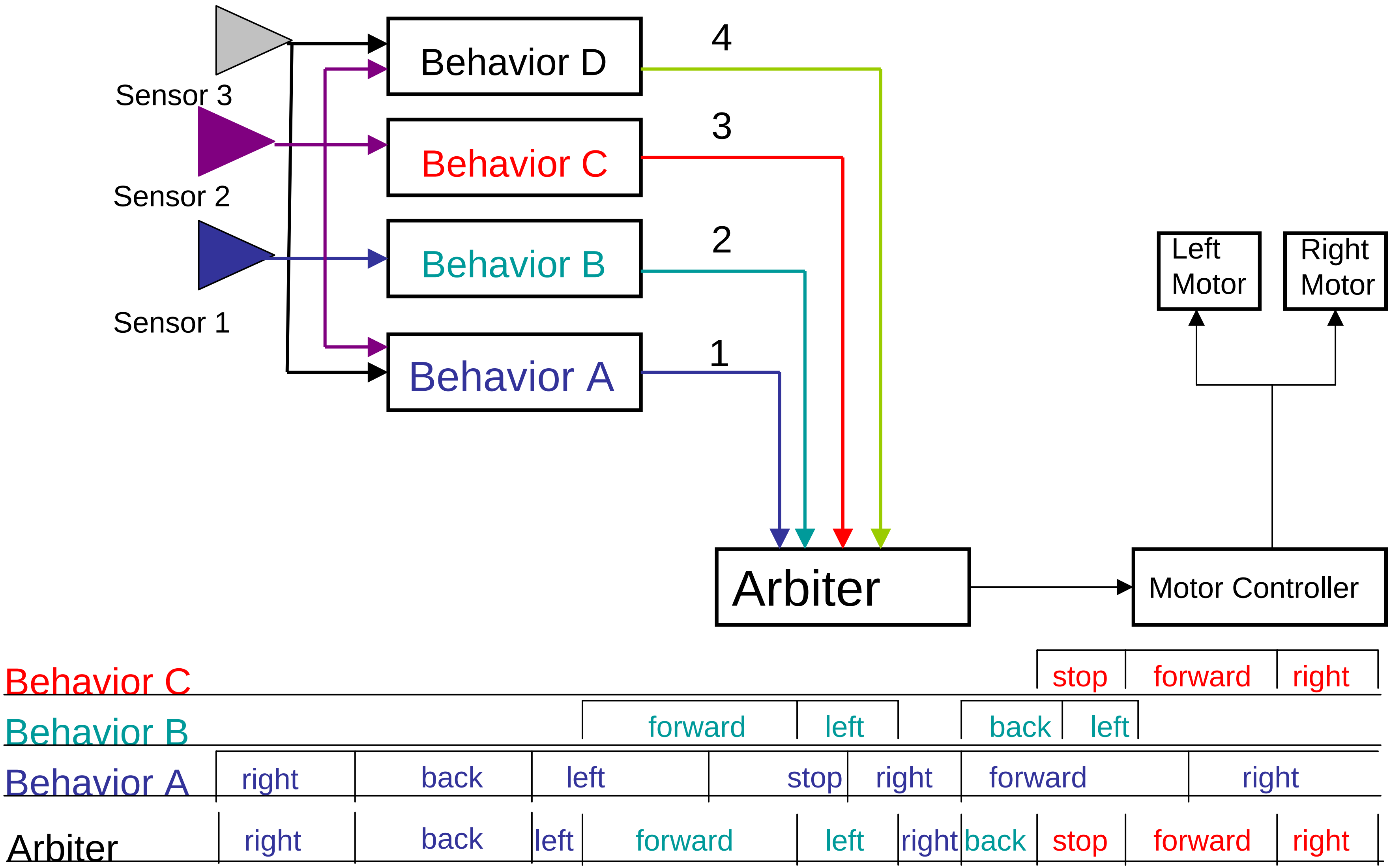


From Robot Programming, Joseph L. Jones, McGraw-Hill, 2004

On Arbitration in General

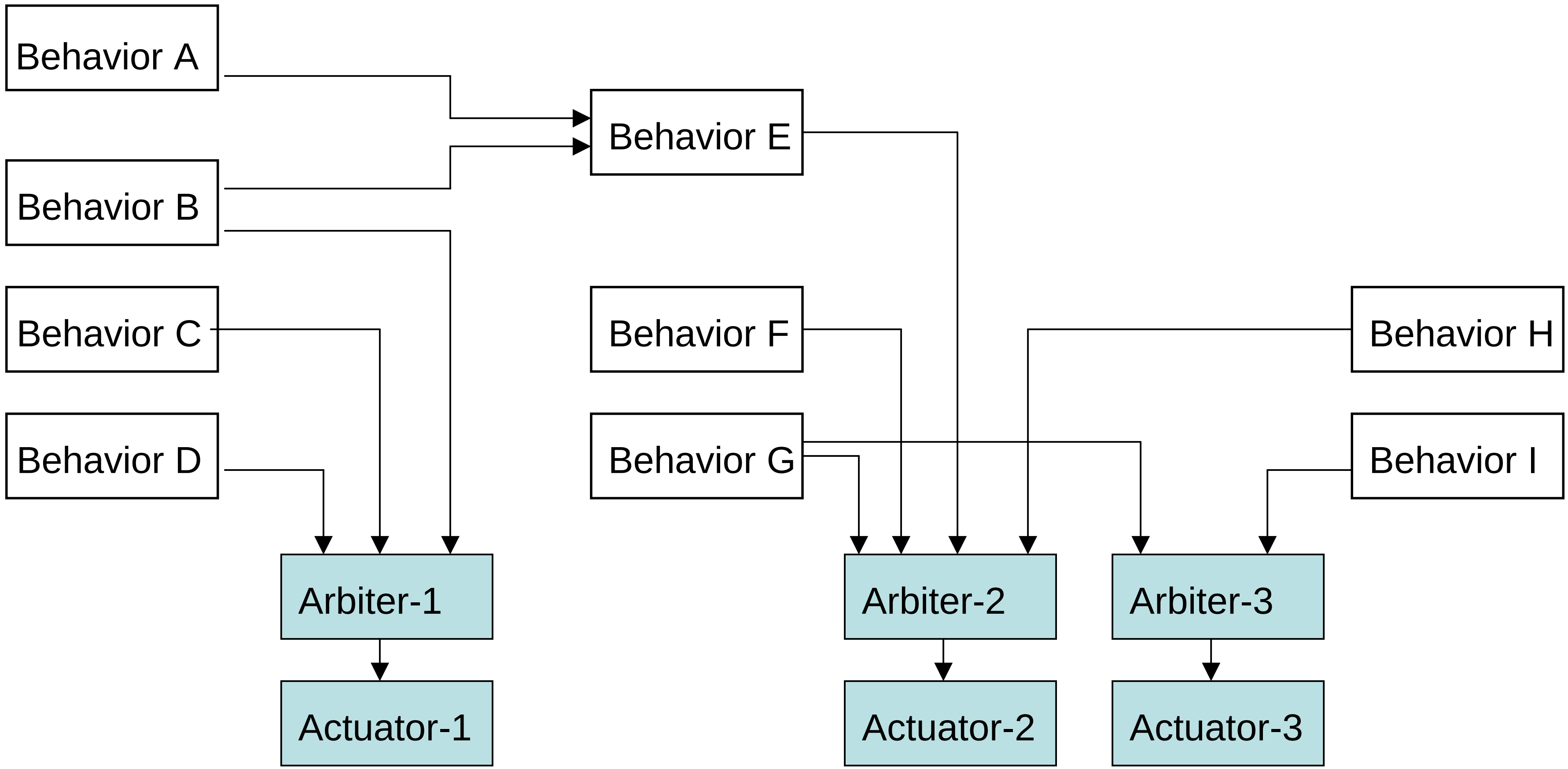
- When to arbitrate:
  - Eg. wander-behavior and recharge-behavior
- What to decide? Average, take turns, vote
    - Use urgency
    - Consider graceful degradation

Fixed Priority Arbitration



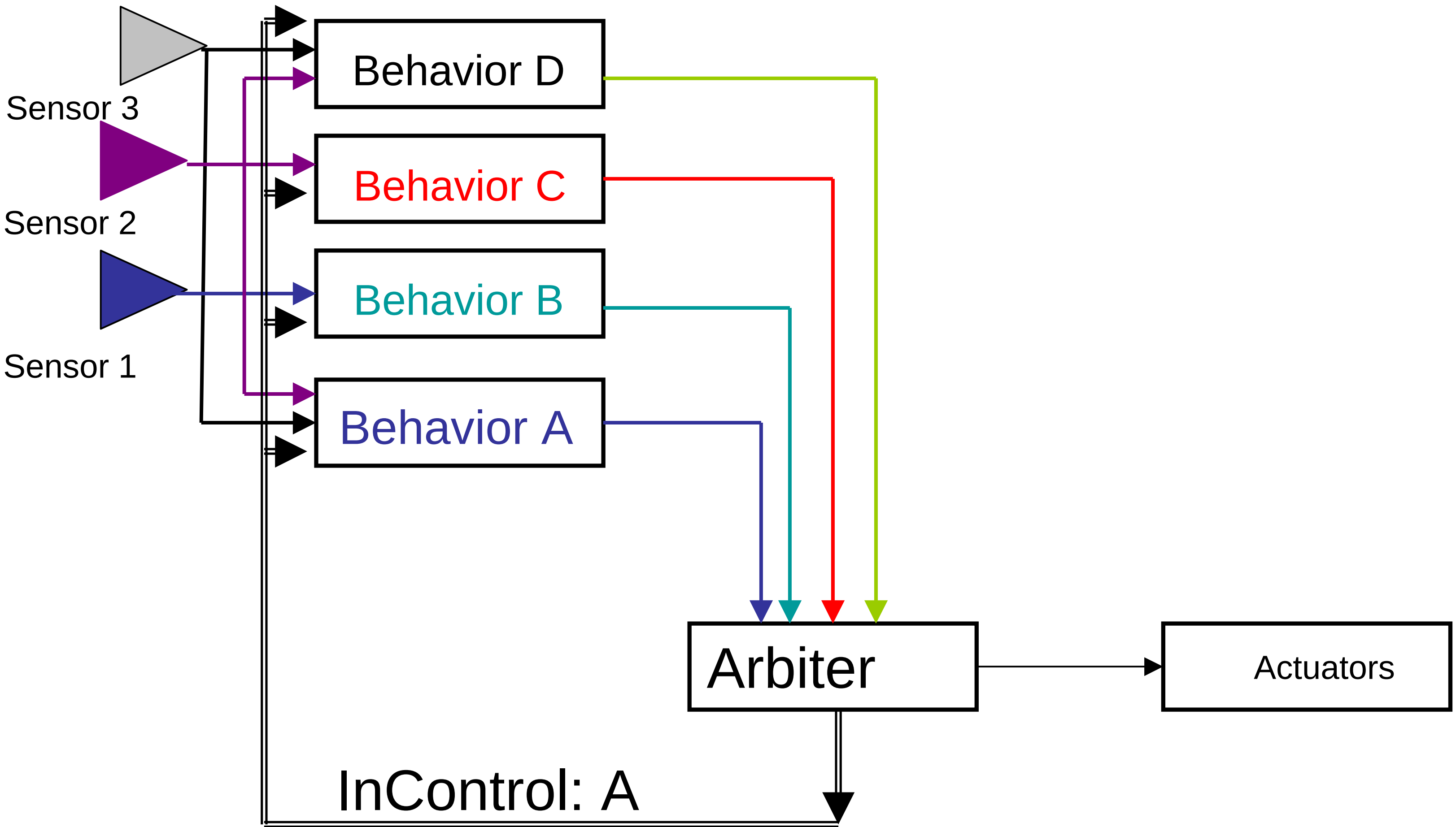
From Robot Programming, Joseph L. Jones, McGraw-Hill, 2004

Multiple Arbiters



From Robot Programming, Joseph L. Jones, McGraw-Hill, 2004

Who has control?



From Robot Programming, Joseph L. Jones, McGraw-Hill, 2004

Arbitration

- When is a variable priority scheme better?
  - Hard to say what happens from code or behavioral diagrams
  - Debugging is tricky
  - "With a well-reasoned decomposition of the problem, a fixed-priority scheme can almost always be engineered to accomplish a given task", J. Jones, p 93.
- Making a variable priority scheme work:
  - Id all dynamic conditions determining priority ordering
  - How to ensure 2 different behaviours NEVER have same priority
  - Lookout for conditions leading to cyclic priority reordering

From Robot Programming, Joseph L. Jones, McGraw-Hill, 2004

Behavior Collision

- How to handle behavior collision
- A) just send the control message
- B) ask for control and wait for it
- C) keep sending control message while behavior is triggered

- Subsumption uses c)
- Nodes have time constants
- After a higher priority message has been channeled thru a node (which never looks at its content!), it does NOT pass a message from a lower priority input until its timer expires
- Time constants are tuned up experimentally

From Robot Programming, Joseph L. Jones, McGraw-Hill, 2004

Behavior Collision

- Often used:
  - Each behavior sets a flag that the arbiter reads (ie on control line to command connection)
  - Arbiter uses command of highest priority which also has set flag
  - Flag eliminates a repetitive send
  - Eliminates complication of a new command to turn off old

From Robot Programming, Joseph L. Jones, McGraw-Hill, 2004

Spiral development in RSS

- Vs subsumption's incremental, experimental approach
  - Value is that the robot works "as expected" at every stage
  - Layers add more Supressors and Inhibiters
- Can a central arbiter have states where it handles only subset of messages from modules using it?

III. Alternative Arbitration Schemes

Action Selection

- Behaviors have continuous activation levels
- Still only one behavior ever active at a time
  - Aka “competitive” scheme
- “How to Do the Right Thing”, Pattie Maes, Connection Science, vol 1, pp 291-323.
- Network of competence modules
- Set of states expressing binary condition
- Each behavior has list of
  - [precondition states, post-true states, post-false states]
- System goals are states. Some are transitional others are protected

Action Selection -2

- 2 Steps:

1. Build a decision network with conflicter, successor and predecessor links
2. Energy spreading to determine active competence module

QuickTime™ and a
TIFF (LZW) decompressor
are needed to see this picture.

From Thesis: An Overview of Behavioural-Based Robotics with Simulated Impleme
On an Underwater Vehicle, Marc Carreras I Perez,U. of Girona, , July 2000

Action Selection
Building the Decision Network

QuickTime™ and a
TIFF (LZW) decompressor
are needed to see this picture.

From Thesis: An Overview of Behavioural-Based Robotics with Simulated Implementatio
On an Underwater Vehicle, Marc Carreras I Perez,U. of Girona, , July 2000

Energy Spread and Activation

- Activation by states, goals and protected goals
- Activation of successors, predecessor and inhibition of conflicters
- Each cycle energy is modulated until a global min/max is reached. Then choose which module to activate:
  - Passes threshold and is executable and has highest energy of those that do
- This is difficult to design but easy to execute once designed!

What about…

- Cooperative arbitration
  - Examples exist:
    - Motor Schemas by Ron Arkin
      - Eg. Behaviors generate potential fields to indicate direction robot should take

QuickTime™ and a
TIFF (LZW) decompressor
are needed to see this picture.

QuickTime™ and a
TIFF (LZW) decompressor
are needed to see this picture.

From Thesis: An Overview of Behavioural-Based Robotics with Simulated Implementation
On an Underwater Vehicle, Marc Carreras I Perez.U. of Girona, , July 2000

- Process description Language
  - Luc Steels, 1992. “The PDL Reference manual”, Memo 92-5, VUB AI Lab

Debugging Arbitration

- Develop and test each behavior in turn
- The difficulty will lie in understanding and managing the interactions between behaviors
- Example: thrashing
- Set up a debug tool: indicated which behavior is active, sensor values, state of arbiter
  - Could be tones or GUI

Primary Source Material

- Brooks, R. A. *"A Robust Layered Control System for a Mobile Robot"*, IEEE Journal of Robotics and Automation, Vol. 2, No. 1, March 1986, pp. 14-23; also MIT AI Memo 864, September 1985.
- Robot Programming: A Practical Guide to Behavior-based Robotics, Joseph L. Jones, McGraw-Hill, 2004.
- The Behavior Language: User's Guide, AI Memo 1227, April 1990.
- A Colony Architecture for an Artificial Creature, Jonathon Connell, AI-TR 1151, MIT, 1989.
- Motor Schema Based Navigation for a Mobile Robot: An Approach to Programming by Behavior, Ron Arkin, Proc of ICRA, 1987, pp 265-271.
- Behavior-based control: Main properties and Implications, Maja Mataric, *Proceedings, IEEE International Conference on Robotics and Automation, Workshop on Architectures for Intelligent Control Systems*, Nice, France, May 1992, 46-54.