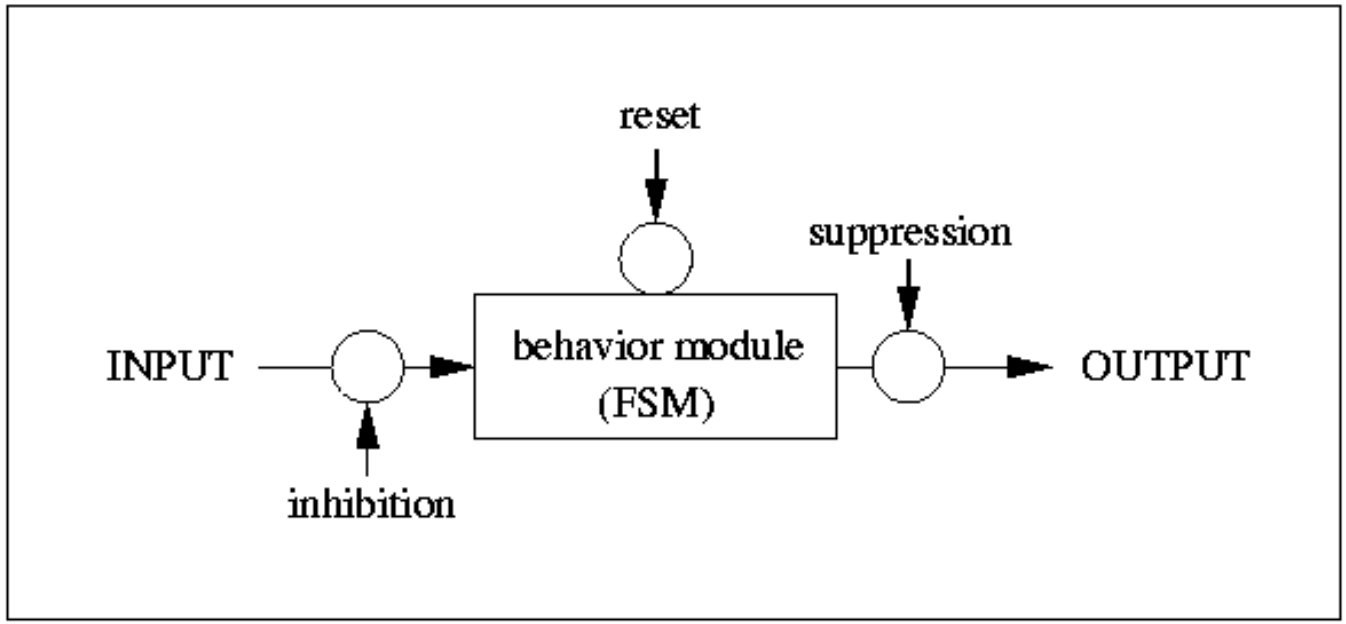
Examples for Behavior-Oriented Programming Languages

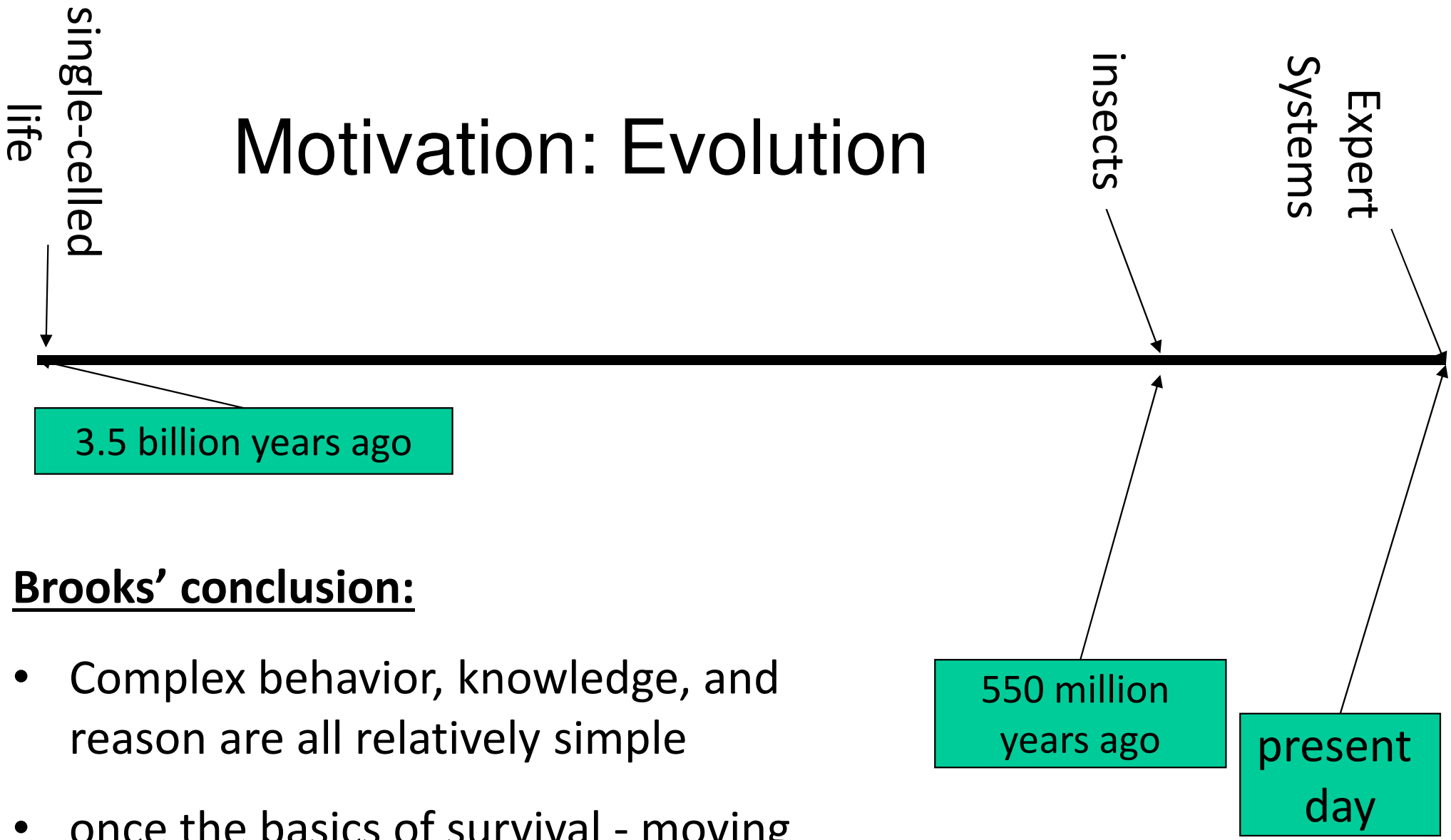
Subsumption Architecture

Rodney Brooks, MIT, mid-80s

- finite state machines (FSM)
- augmented (AFSM) with links to allow mutual influences
- layered control
  - bottom-up
  - stepwise refinement



Motivation: Evolution



Brooks' conclusion:

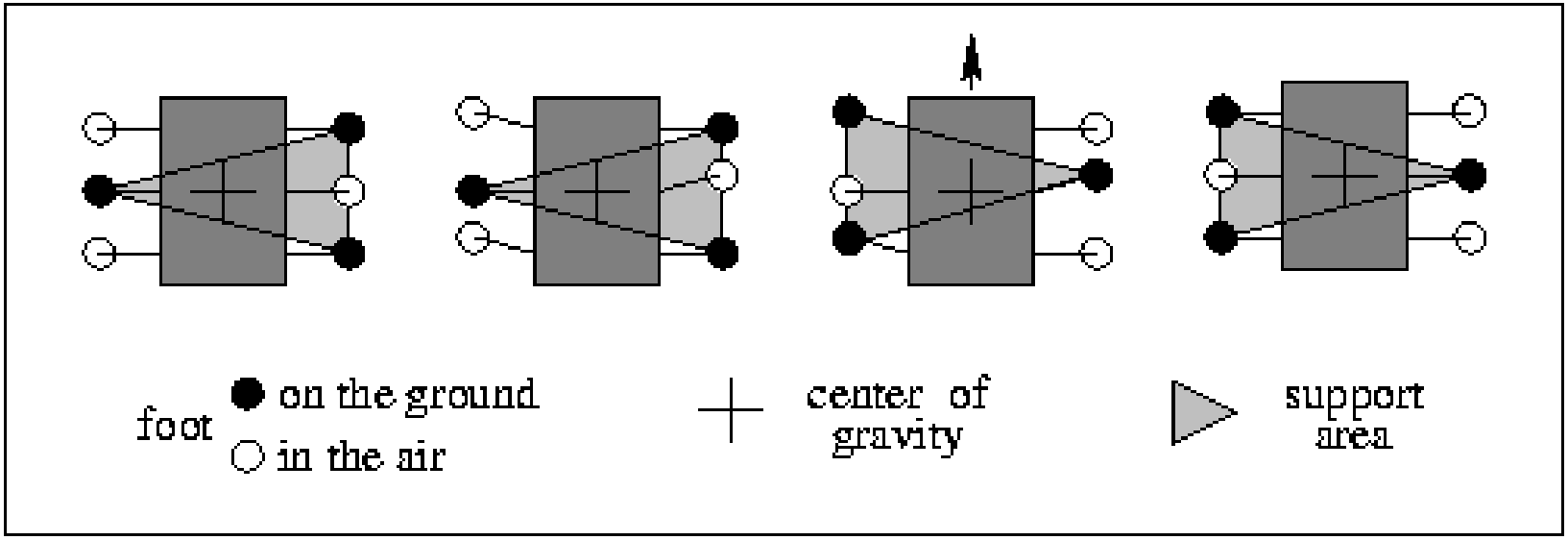
- Complex behavior, knowledge, and reason are all relatively simple
- once the basics of survival - moving around, sensing the environment, and maintaining life - are acquired.

6-Legged Walking

tripod-gait

- static
- 6-legged
- e.g., insects

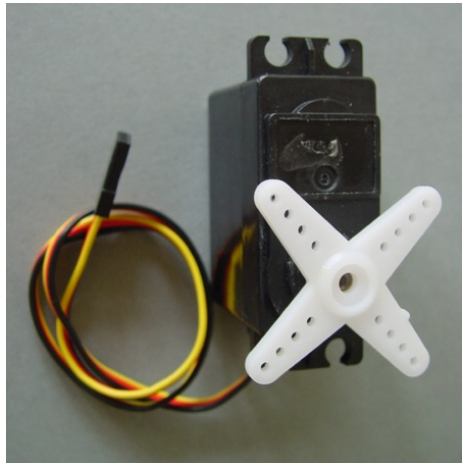
- simple control
- no need for modeling of physical dynamics
- but at least 12 DOF (6x lift/down & forward/back)

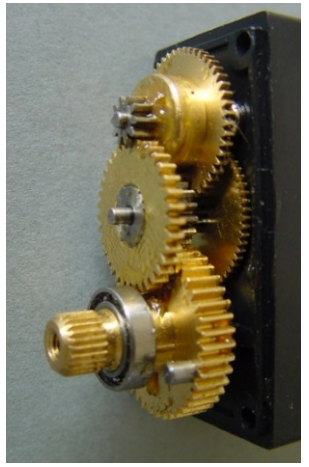


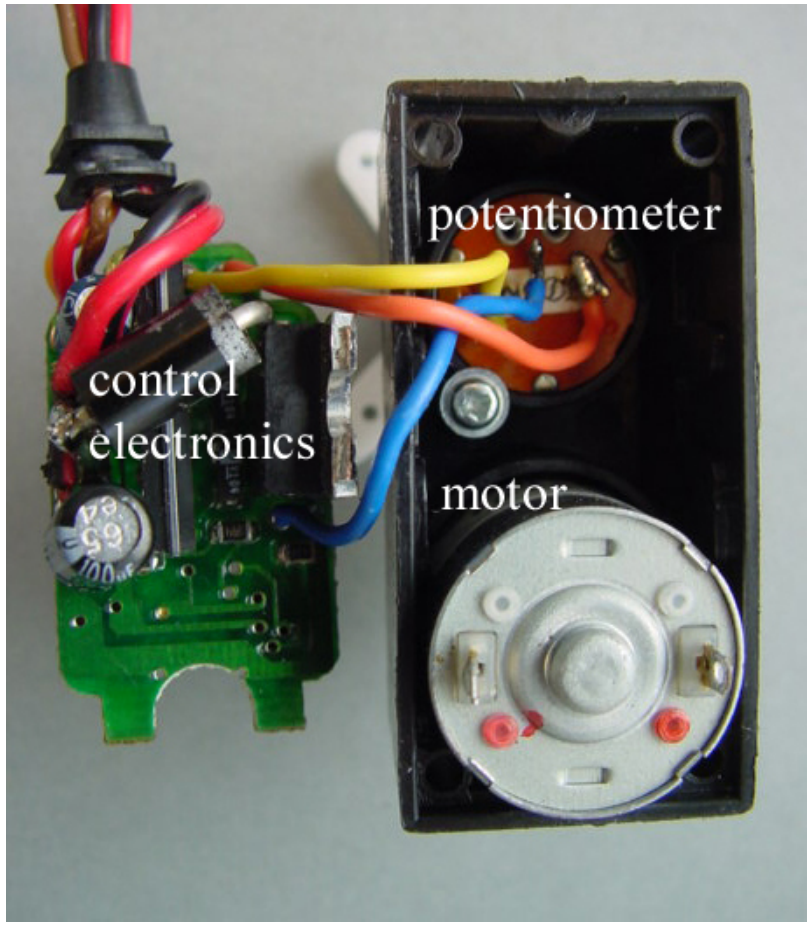
Servo

- DC-motor
- + gear-box
- + electronics
  - feedback loop
  - via potentiometer
  - actively holds servo axis at a given input angle

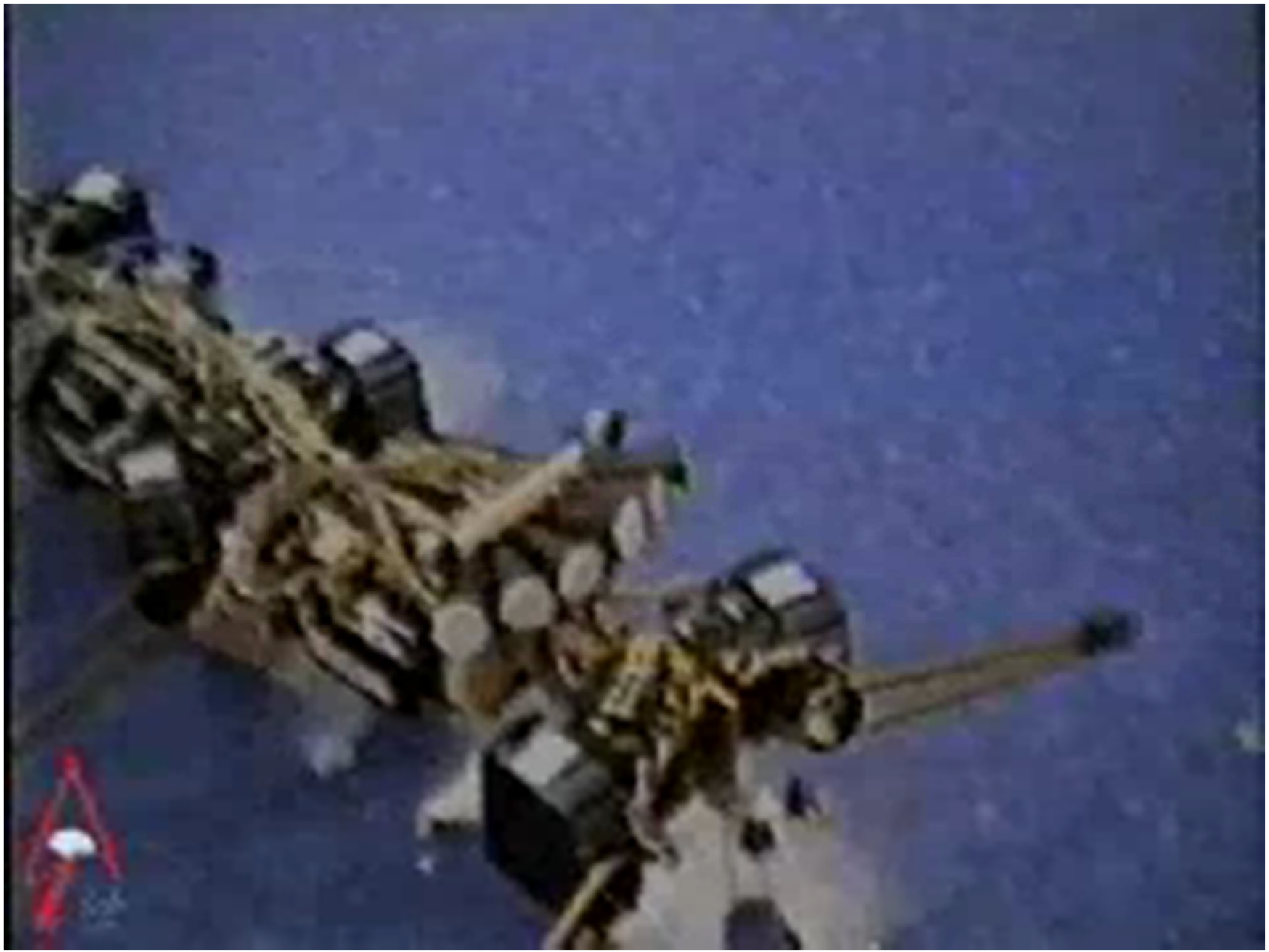
- 2 servos per leg (up/down, forw./back)
- or 3 (plus flexing like insects)



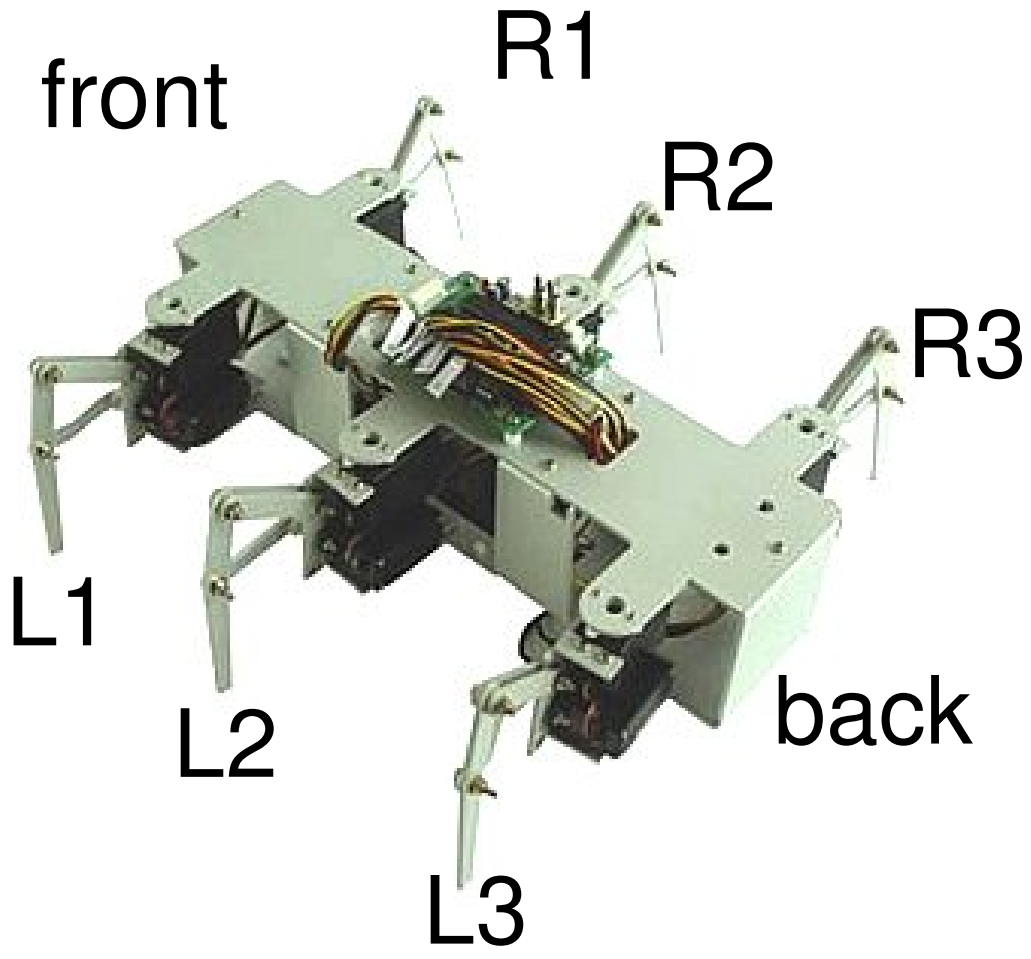




Tripod Gait



Tripod Gait



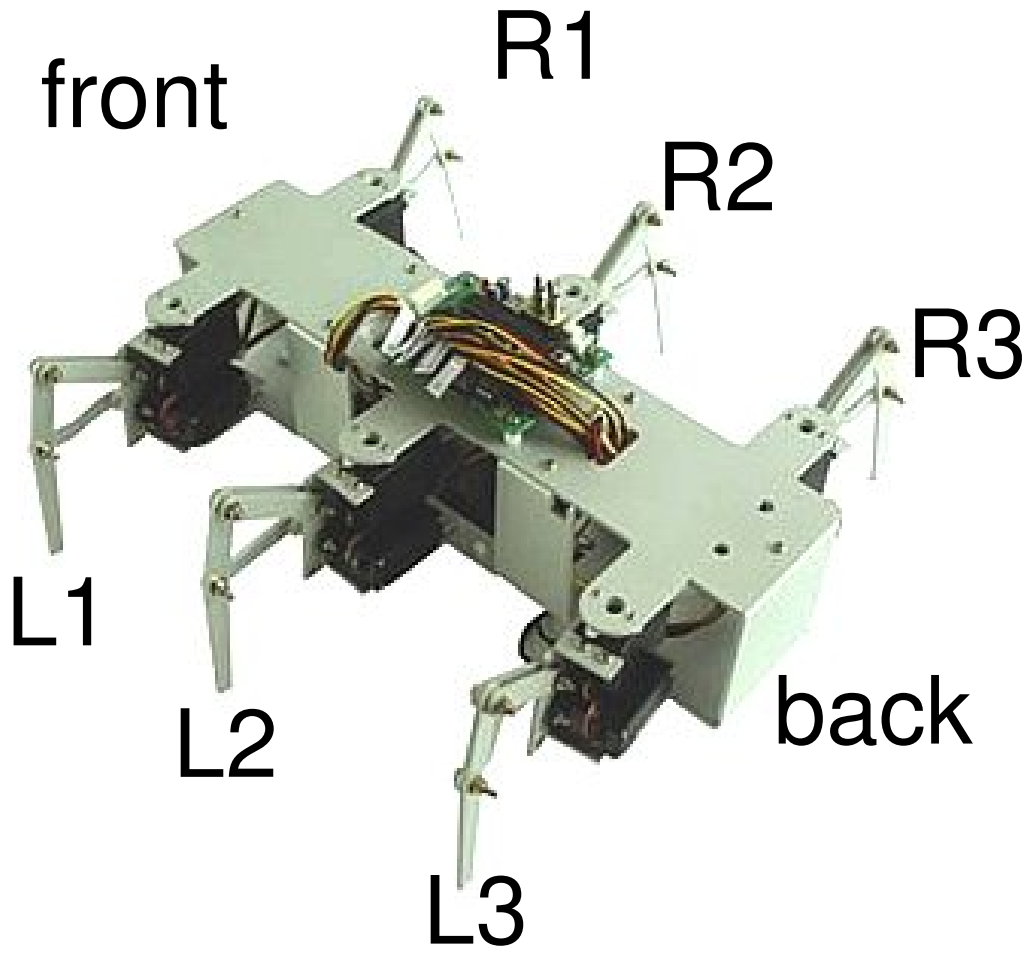
behaviors:

- up(leg X)
- down(leg X)
- forward(leg X)
- back(leg X)
- robot-foward()

...

can you sketch the overall controller?

Tripod Gait



behaviors:

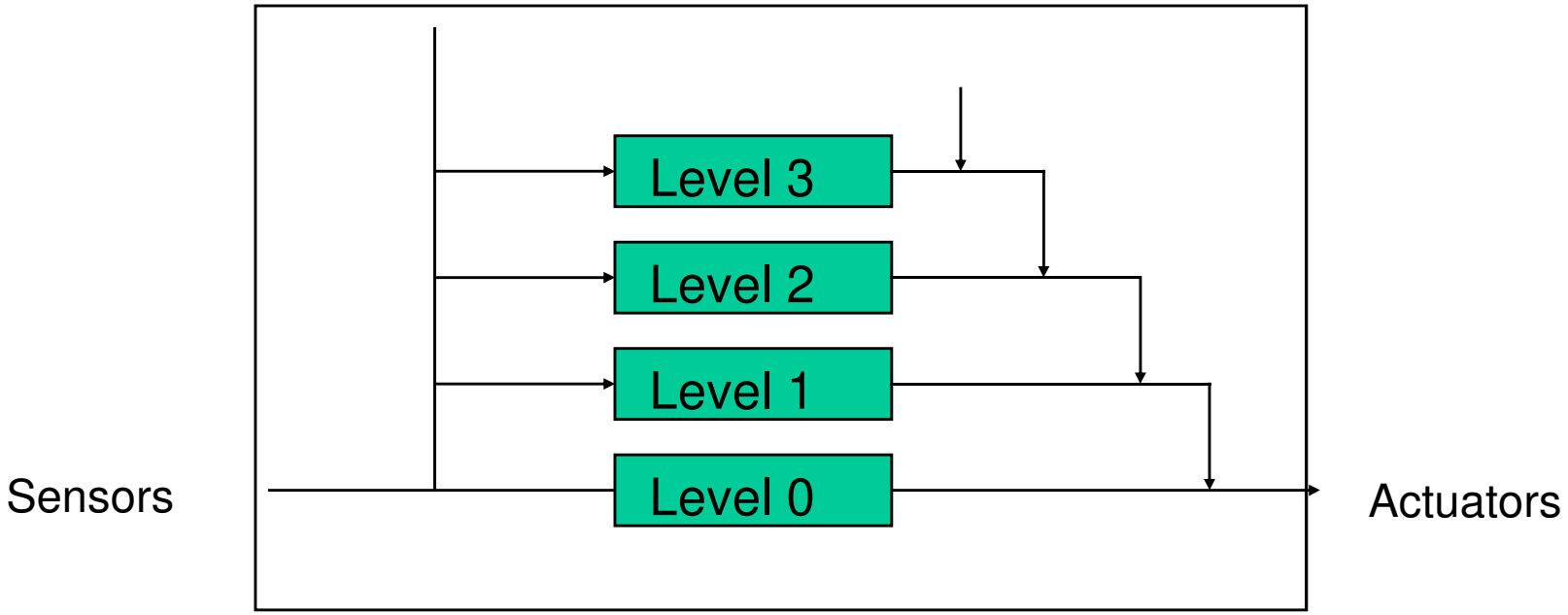
- up(leg X)
- down(leg X)
- forward(leg X)
- back(leg X)
- robot-foward()

...

what about turning the robot?

Subsumption Architecture

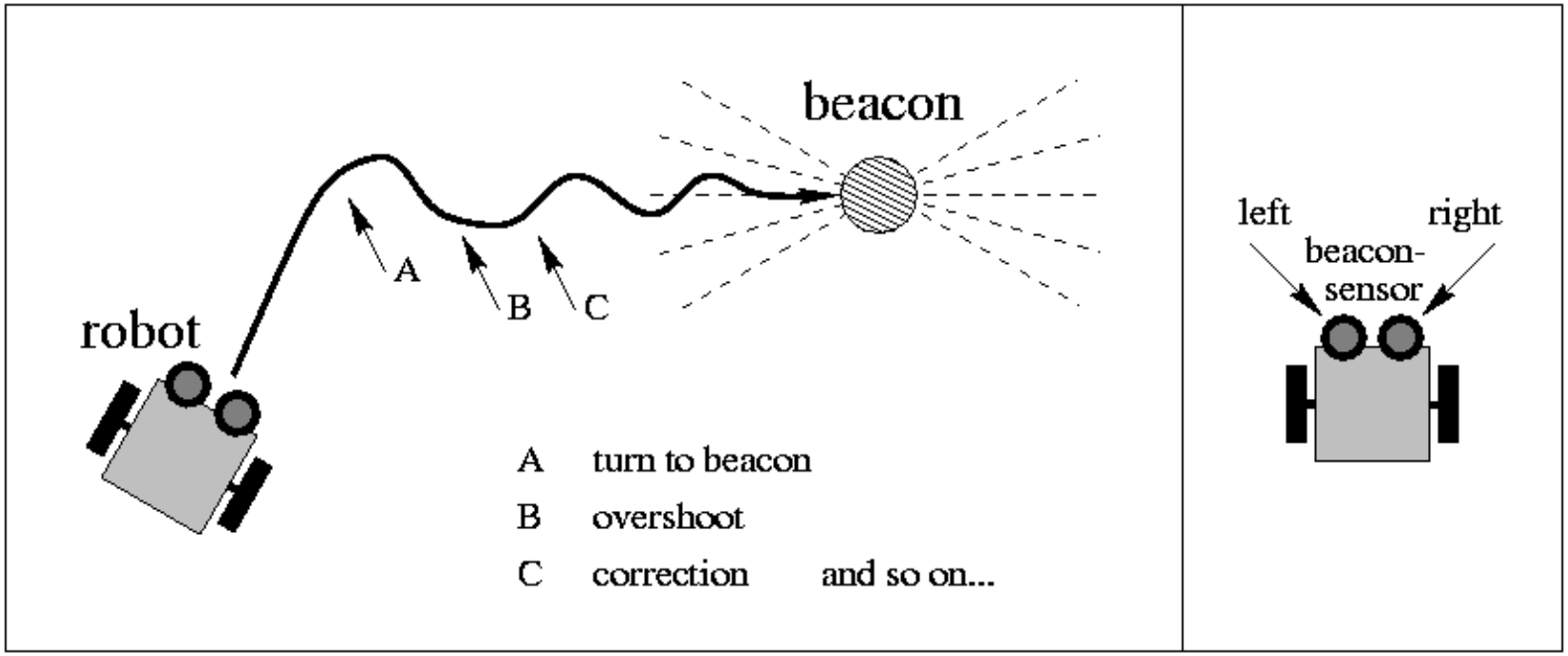
- FSM are actually not that elegant
- subsumption architecture programs originally implemented in LISP
- mainly relevant for historical reasons and the idea of layered behaviors controlling lower ones (though binary)

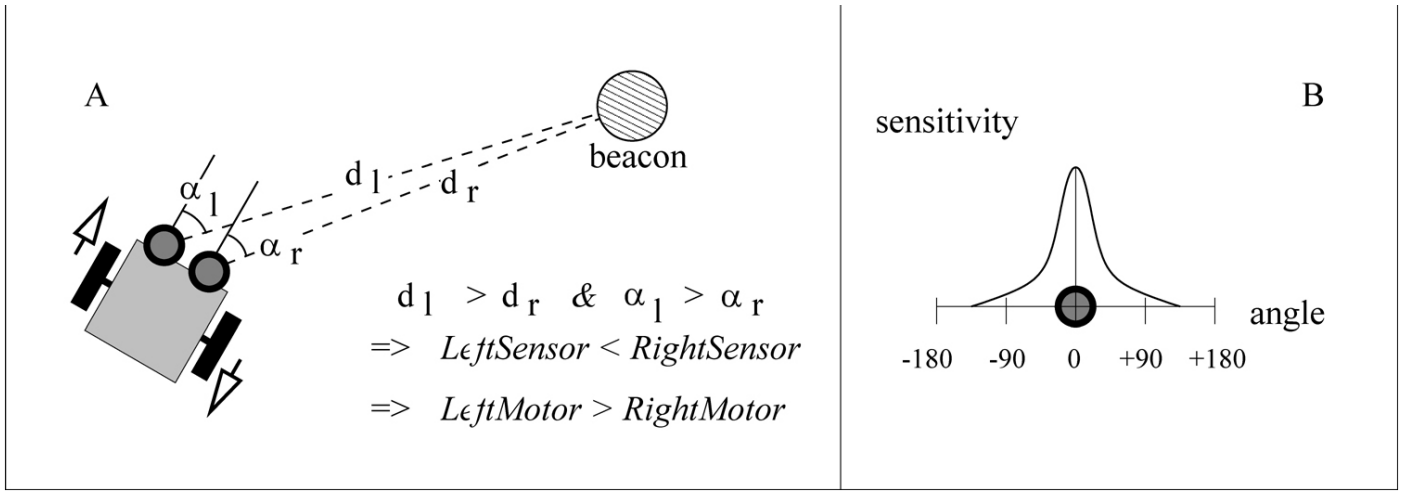


Process Description Language (PDL)

- runs in cycles
  - fast, fixed frequency
  - emulation of dynamical processes
- processes
  - all executed (pseudo-)parallel in one cycle
  - round robin scheduler (one process after the other)
- quantities
  - bounded variables
  - *value(q)* : returns value of q at previous cycle *t-1*
- additive update
  - *add_value(q, v)* : adds *v* to quantity *q*
  - all additive updates are in only in effect at end of current cycle *t*
  - there is no assignment operator

Example: Homing in on Beacons





... in PDL

- quantity
- process
- commands
  - value
  - add_value

```
1  Initialization
2      quantity   LeftSensor ∈ [0, 100]
3      quantity   RightSensor ∈ [0, 100]
4      quantity   LeftMotor ∈ [−100, +100]
5      quantity   RightMotor ∈ [−100, +100]
6      constant   DEFAULT_SPEED = +50
7      constant   MAX_CHANGE = 10
```

```
1  process(forward) {
2      add_value(LeftMotor,
3          −value(LeftMotor) + DEFAULT_SPEED)
4      add_value(RightMotor,
5          −value(RightMotor) + DEFAULT_SPEED)
6  }
```

```
1  process(taxis) {
2      b_direction = (value(LeftSensor) − value(RightSensor)) / SENSOR_MAX
3      b_intensity = (value(LeftSensor) + value(RightSensor)) / (2 · SENSORMAX)
4      add_value(LeftMotor,
5          −1 · b_direction · (1 − b_intensity) · MAX_CHANGE)
6      add_value(RightMotor,
7          +1 · b_direction · (1 − b_intensity) · MAX_CHANGE)
8  }
```

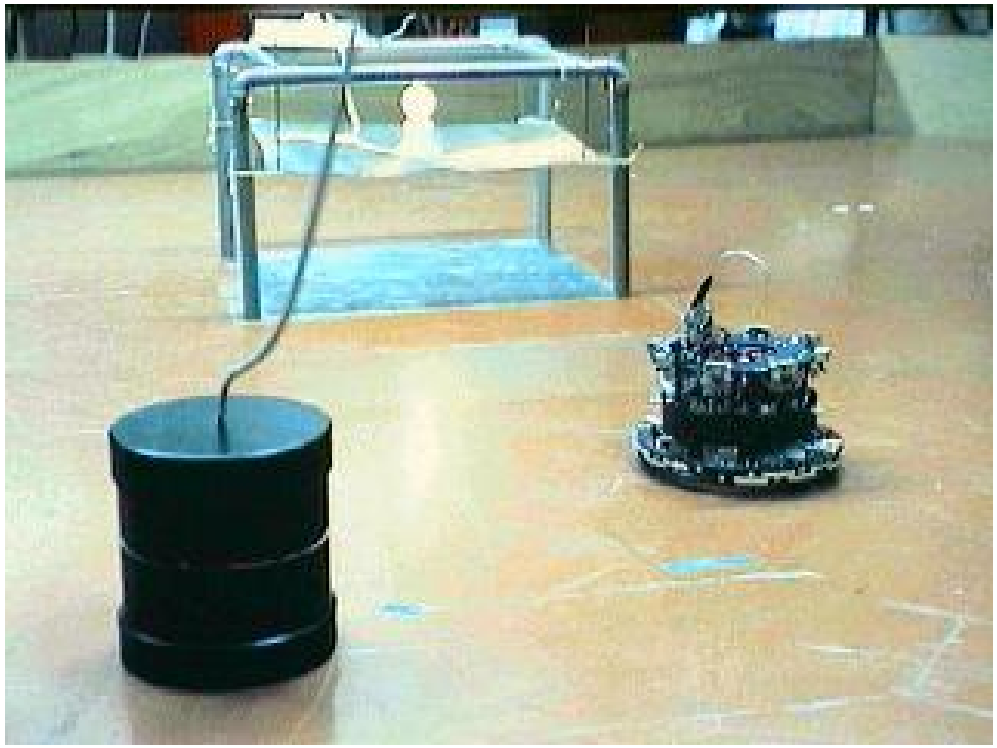
An example scenario

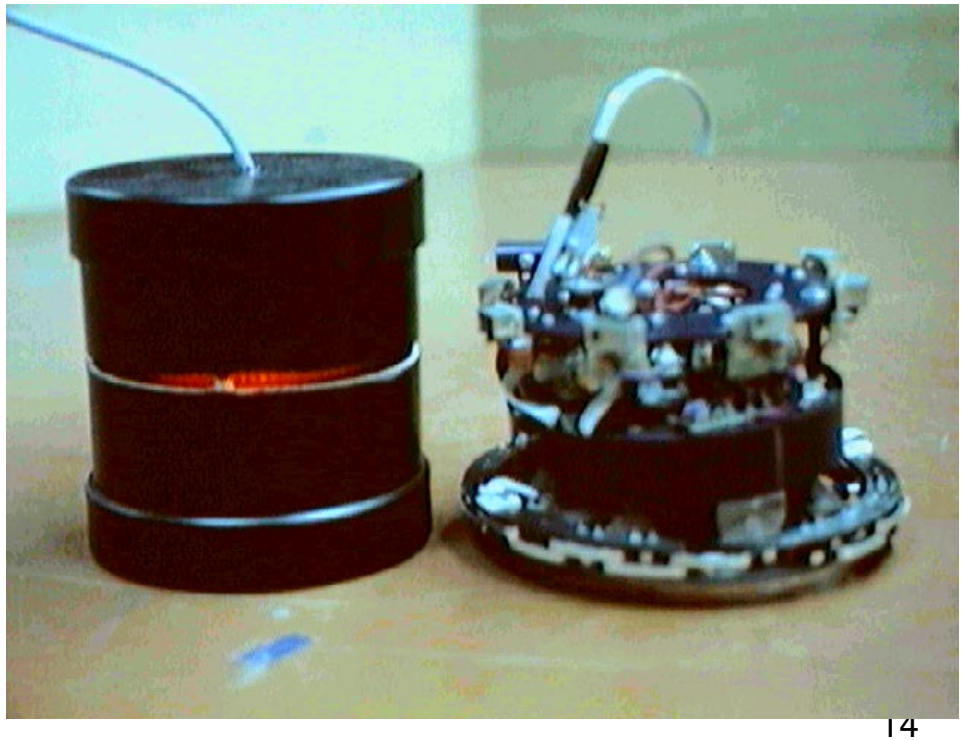
The Robot "Ecosystem"

basic set-up

[Steels, McFarland 1994]

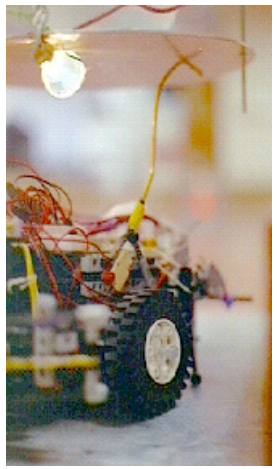
- "simple" mobile robots
- charging-station
- competitors
  - boxes housing lamps
  - working task

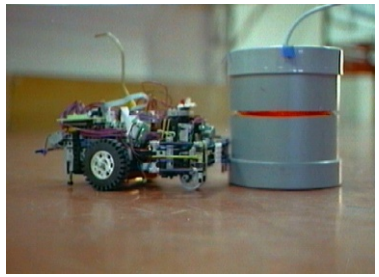


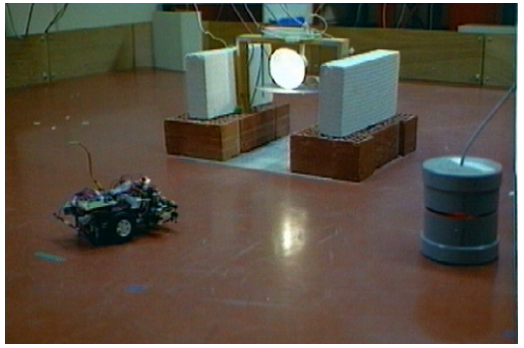


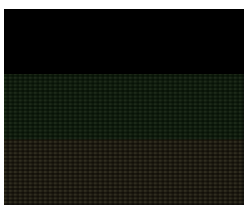
The behaviors in the Ecosystem

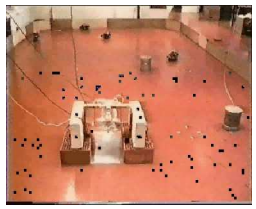
- touch-based obstacle avoidance
- active IR obstacle avoidance
  - active IR is a distance sensor
- photo-taxis to the charging-station
  - homing in on a beacon (white light)
- charging
  - stop when current is flowing
  - move when batteries are full
- attraction to competitors
  - homing on a beacon (red light)





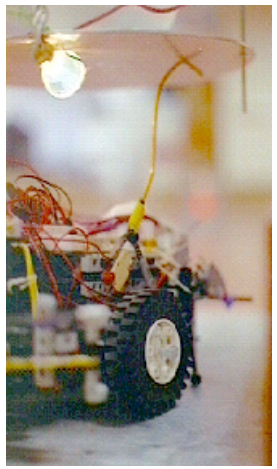


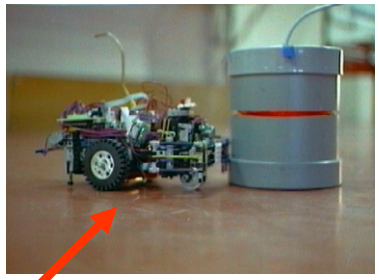


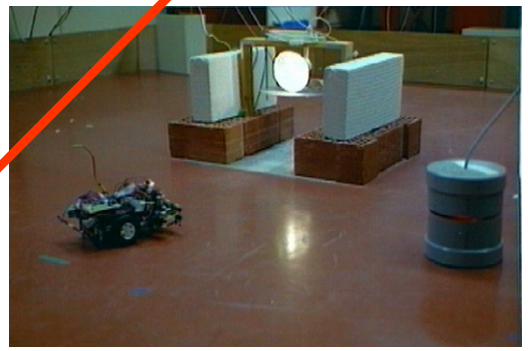


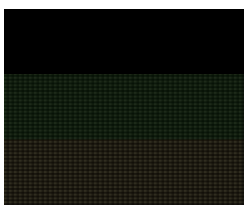
The behaviors in the Ecosystem

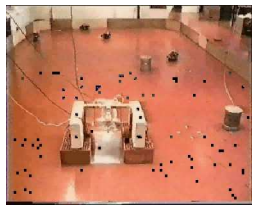
- touch-based obstacle avoidance
- active IR obstacle avoidance
  - active IR is a distance sensor
- photo-taxis to the charging-station
  - homing in on a beacon (white light)
- charging
  - stop when current is flowing
  - move when batteries are full
- attraction to competitors
  - homing on a beacon (red light)









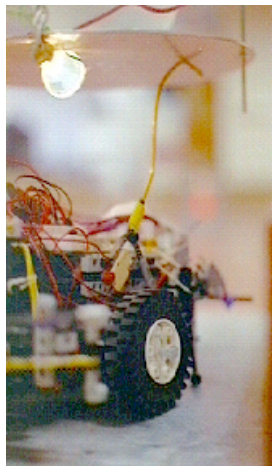


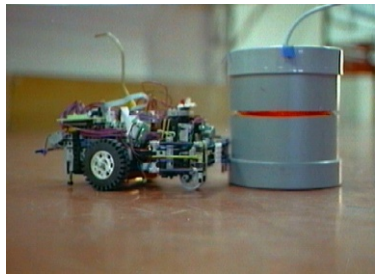
where is the "fight competitors"???

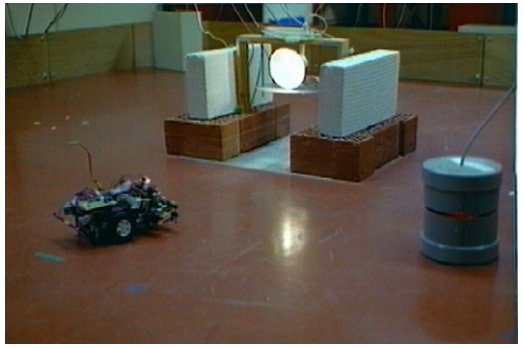
The behaviors in the Ecosystem

- touch-based obstacle avoidance
- active IR obstacle avoidance
  - active IR is a distance sensor
- photo-taxis to the charging-station
  - homing in on a beacon (white light)
- charging
  - stop when current is flowing
  - move when batteries are full
- attraction to competitors
  - homing on a beacon (red light)

emergence!!!

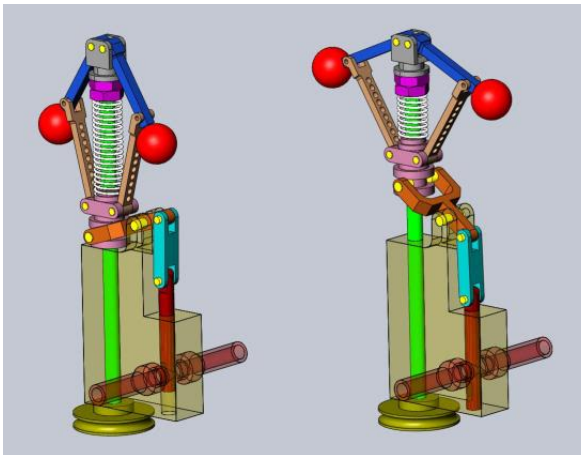


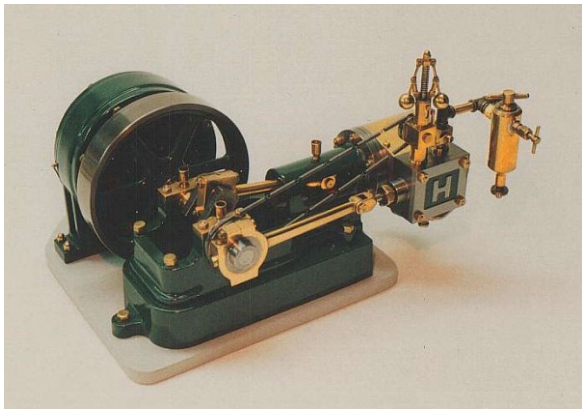




Behaviors = (Poor Man's) Control Theory?!?!

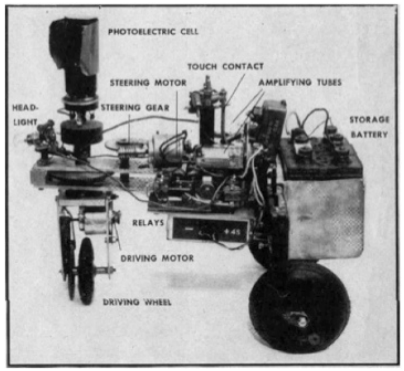
- control theory
  - starting in 19th century: study of steam engine and governor
  - using differential equations
- cybernetics
  - apply principles of control theory to model natural systems
  - 1940s, Norbert Wiener (MIT)

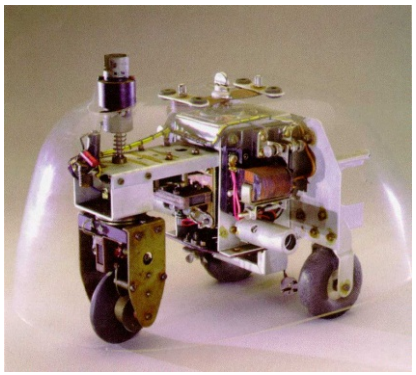


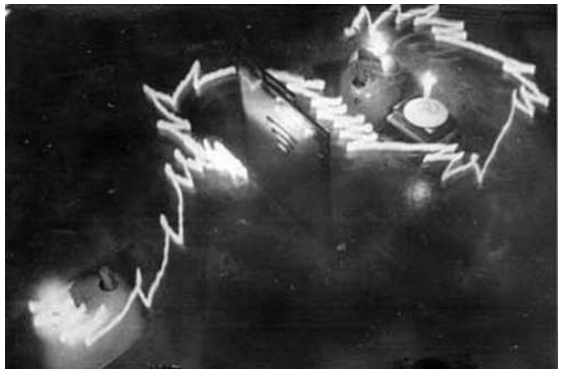


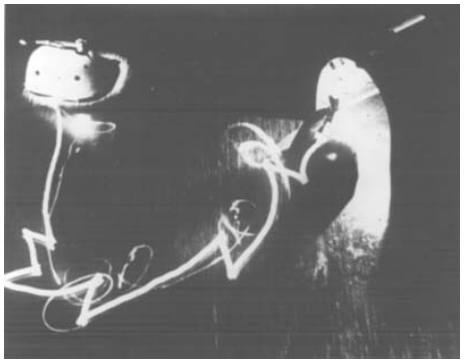
Behaviors = (Poor Man's) Control Theory?!?!

- Wiener: cybernetics is solid philosophy
  - philosophy: it deals with the working principles of animal and man
  - solid because it is based on differential equations
- William Grey Walter's tortoises (late 40s - 50s)
  - (W.) Grey Walter: neurologist from Bristol
  - analog circuits (radio vacuum tubes)
  - differential drive, light sensors, bumper
  - homing in on white light => charging station

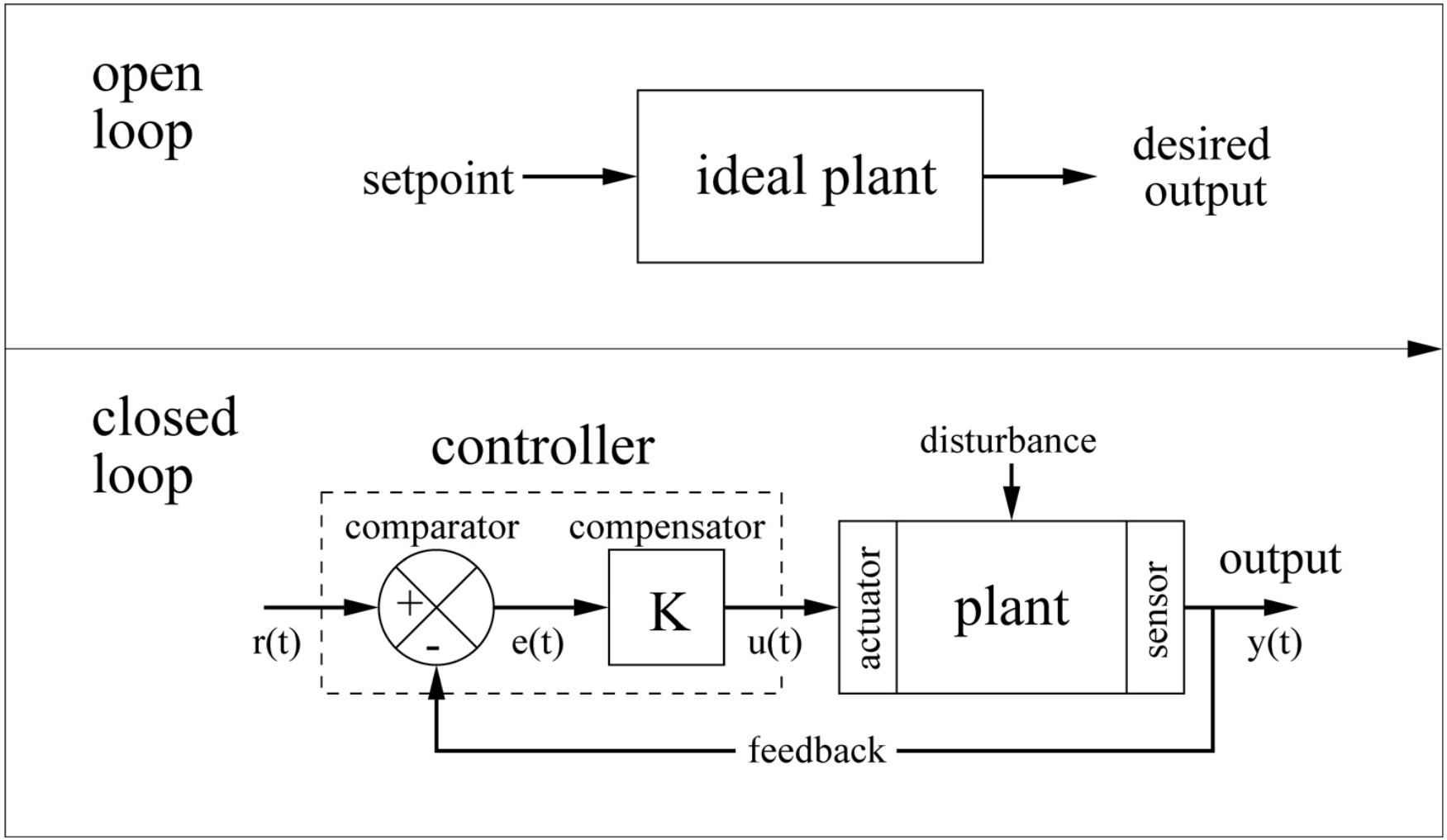






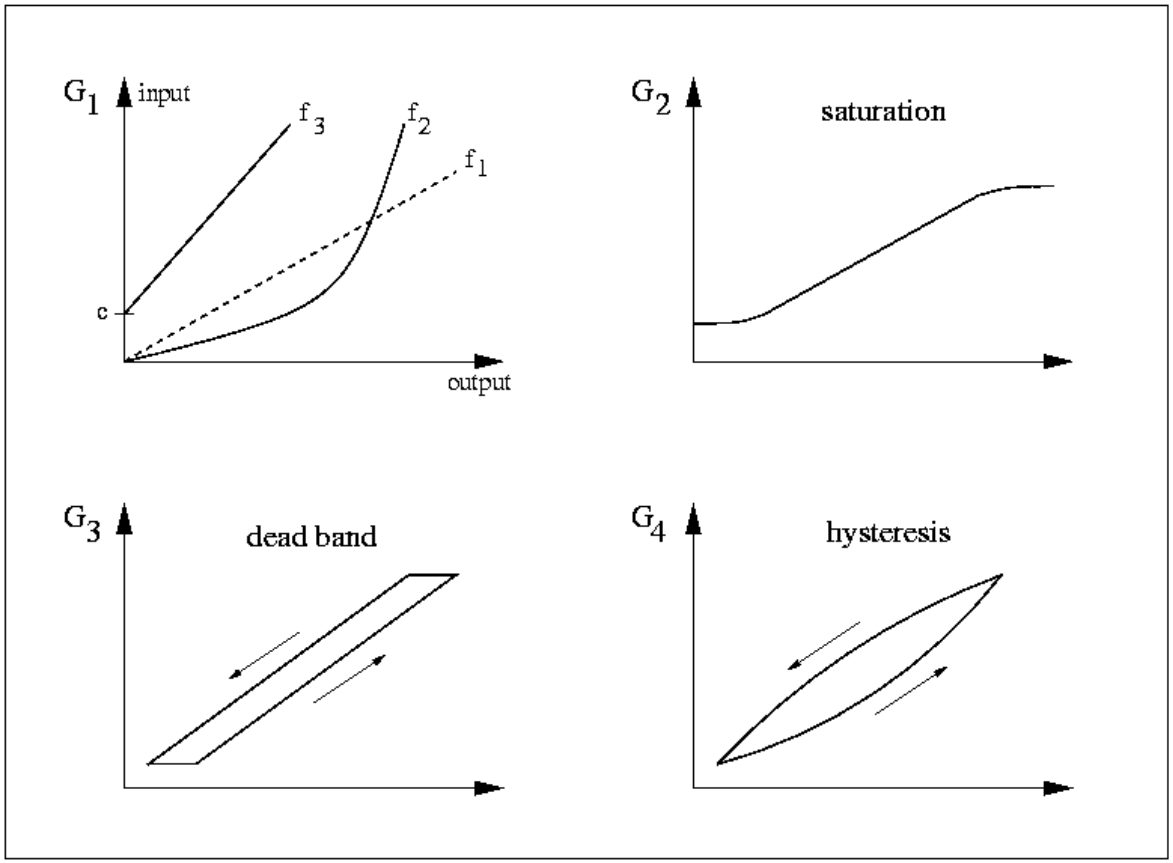


Control-Theory



Plant-types

- linear
  - follows the superposition principle
  - u1 + u2
    => (y1+y2)
- non-linear

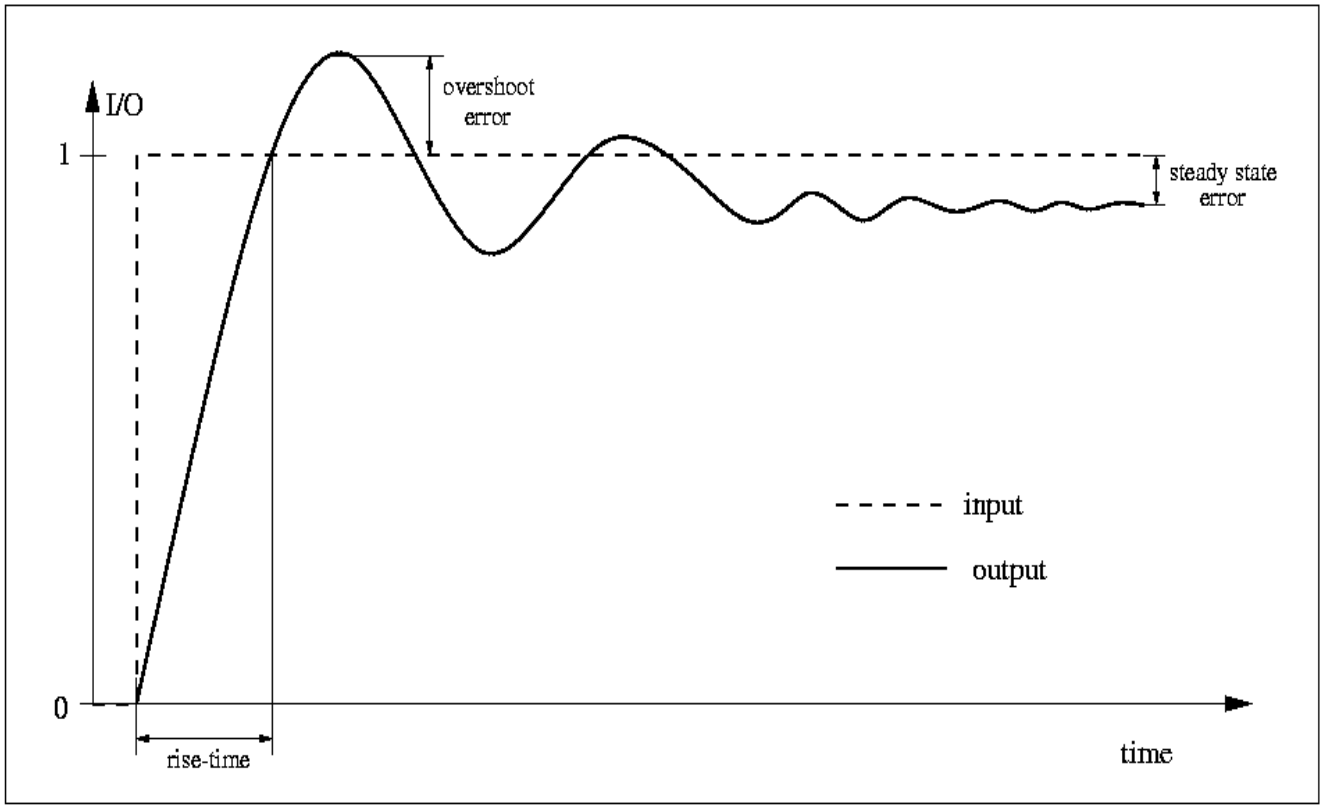


The behavior of a controller

step-response

error-types:

- overshoot
- rise-time
- maximum
- steady-state



PID-controller

$$u(t) = P \cdot e(t) + I \cdot \int e(t) + D \cdot \Delta e(t)$$

- **P** roportional factor increases
  - rise-time decreases
  - likelihood or amount of overshoot increases
- **I** ntegrative factor increases
  - steady state error decreases
  - likelihood or amount of overshoot increases
- **D** erivative
  - damping (negative sign)

PID Implementation and Tuning

- implementation
  - ***discrete time-steps***, typical: fixed frequency
  - ***recursive computation***
    - to avoid possibly unbounded integration
    - note that u(t-1) uses the integrated error up until t-1
    - therefore: express u(t) using u(t-1)
  - common combinations P, PI or PID
- tuning
  - [1] P first: minimize rise-time
  - [2] then I: eliminate steady-state error
  - [3] D last: damp the overshoot

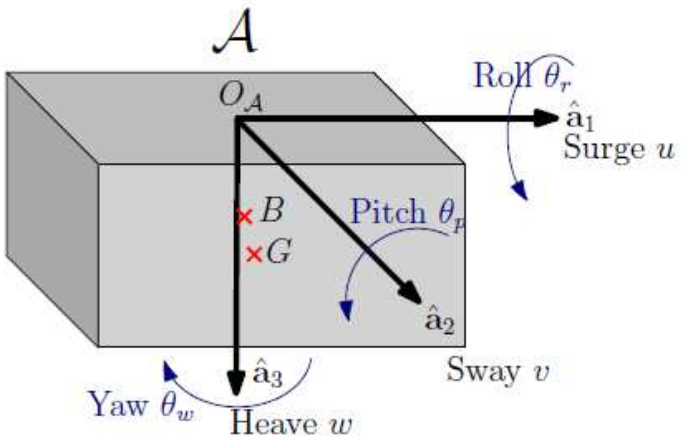
Control ***Theory?!?!***

PID parameter tuning by trial and error?

- in practice
  - often yes
  - due to lack of exact system models
- in theory
  - derivation of optimal controllers
  - which are at least very good starting points for subsequent trial and error tuning
  - (to compensate for incomplete model or/and imprecise parameters)

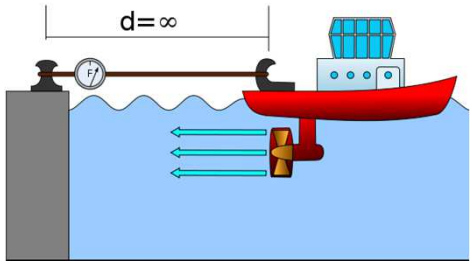
System Models

- systems may look simple
- but models can easily get quite complex
- example: underwater vehicle
  - open frame design
  - thrusters mounted orthogonally to each other
  - along surge, sway, heave axes



Underwater Vehicle

- thruster: voltage -> force
  - thrusters/propellers usually not symmetric, i.e., less backward then forward force
  - Bollard pull: idealized force measurement
  - Seabotix BTD150: roughly linear relation of electrical power to force



Direction	Voltage(DC)	Current Draw (A)	Thrust (kg)	Power (W)
Forward	12.0	1.91	0.75	22.92
	14.0	2.45	1.00	34.3
	16.0	2.78	1.20	44.48
	18.0	3.32	1.43	59.76
	20.0	3.78	1.65	75.6
	22.0	4.20	1.85	92.4
Reverse	12.0	1.92	0.68	23.04
	14.0	2.42	0.92	33.88
	16.0	2.89	1.15	46.24
	18.0	3.30	1.38	59.4
	20.0	3.75	1.60	75
	22.0	4.23	1.80	93.06

Underwater Vehicle

- force -> acceleration -> velocity
  - everything is simple, or???

Underwater Vehicle

- force -> acceleration -> velocity
  - everything is simple, or???
- taking inertia into account
  - typically assuming homogeneous mass distribution
  - center of mass not necessarily same as center of motion
- not only thruster force
  - gravity: simply constant
  - buoyancy: density of the water (fresh or salt) & vehicle volume
- drag
  - depends on vehicle geometry…

Back to: Behaviors = (Poor Man's) Control Theory?!?!

yes and no…

some additional crucial issues:

- need for good system models for control theory
  - see example of underwater robot
  - decoupling to keep things manageable, e.g., considering diving control loop independent from motions in 2D plane
  - hence very good for designing single behaviors
- how to combine control-loops/behaviors
  - matters of architecture