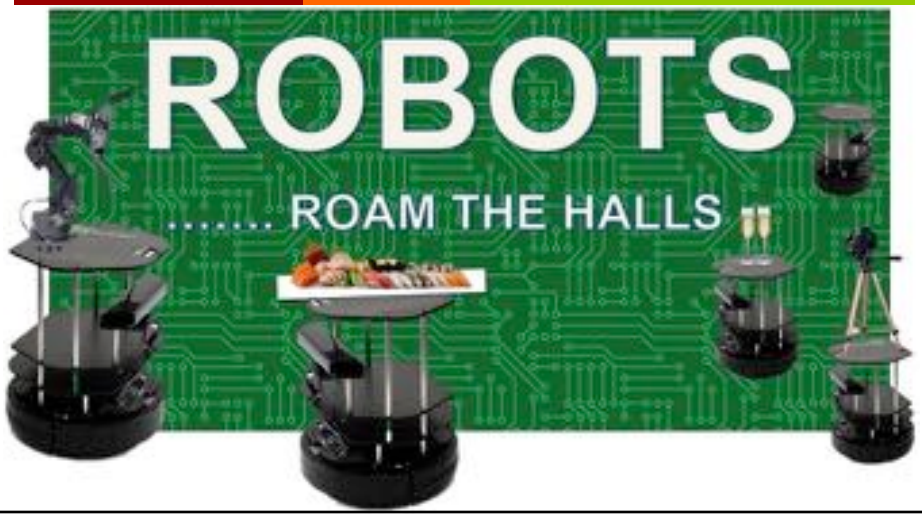
# CS 189: Autonomous Robot Systems

Spring 2019

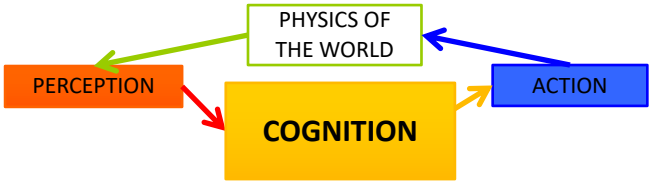



# Agenda

- Lecture: Autonomy III: Programming Complex Behaviors
  - Software Architectures for Robotics
  - Brief History of Robotics
- Demo Time:
  - Pset 3 (a) tests
  - Lab 3 (More Interactions: Text2Speech and ARTags)
- Upcoming:
  - Pset3 (b): Follower

- References: This lecture is based in part on
  "Introduction to AI Robotics", chapters 2-5, Robin Murphy, 2000 [trashcollector, architectures]
  "A Robotics Primer", chapter 2, Maja Mataric, 2007 [history]

# What Does it Mean to be Autonomous?




# Basics of Autonomy

ACTION

- Action (Actuators)
  - Locomotion: Wheels (Differential Drives, Kinematics)

PERCEPTION

- Perception (Sensors)
  - Proprioception and Exteroception (Bump, Depth)

COGNITION

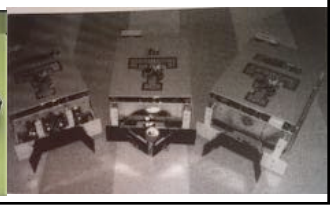
- Cognition (Control)
  - Reactive Behaviors + PID Control
  - **Software Architectures: Simple to Complex Tasks**

# Example: Trash Collector

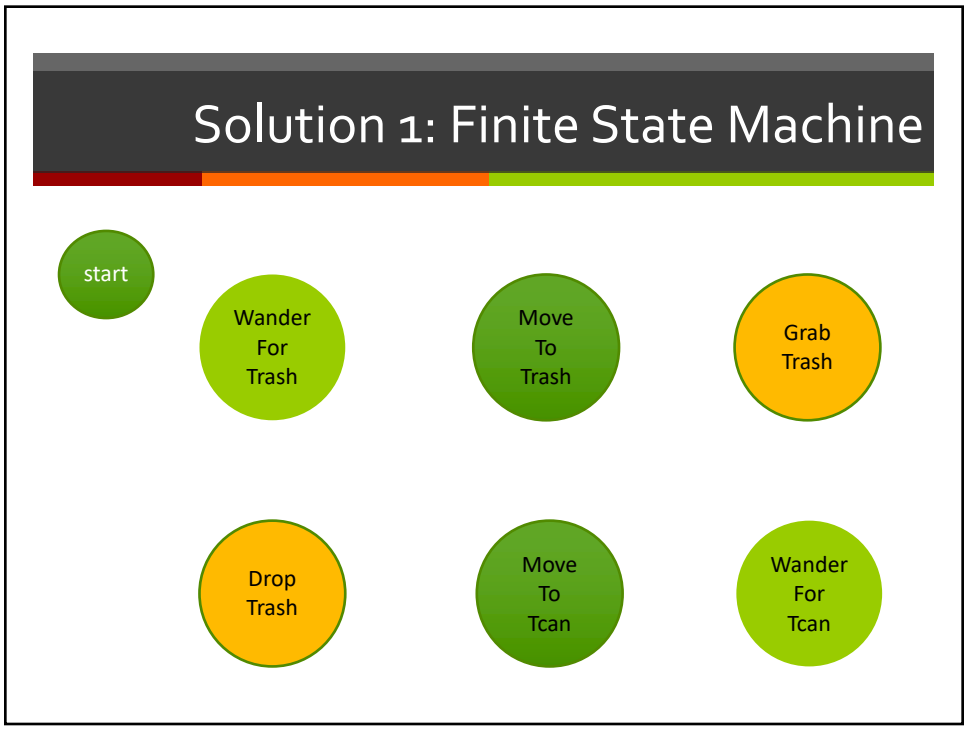
Murphy 2000

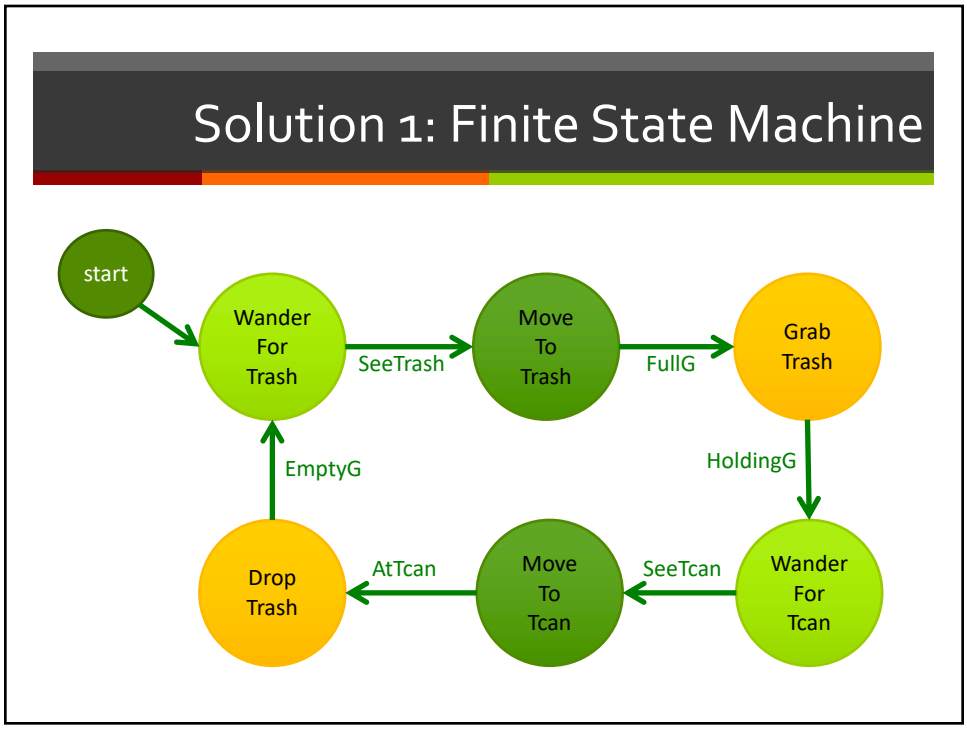
# Example: Trash Collector

- Lets solve a bigger problem
  - AAAI Competition: Pick Up the Trash
- Problem Specs
  - Coca Cola cans, collect them and deposit them in the trashcan (changes year-to-year)
- What are some primitive behaviors we might need?



LOLA (AAAI 1995 Winner)
GaTech (AAAI 1994 Winner)
Example based on
Murphy 2000, section 5.5


Solution 1: Finite State Machine
start
Wander For Trash
Move To Trash
Grab Trash
Drop Trash
Move To Tcan
Wander For Tcan


Solution 1: Finite State Machine
start
Wander For Trash
SeeTrash
Move To Trash
FullG
Grab Trash
HoldingG
Wander For Tcan
SeeTcan
Move To Tcan
AtTcan
Drop Trash
EmptyG

# Solution 1: Finite State Machine

At the start, initialize **self.state** = 'wander_for_trash'

**# Finite State Machine LOOP**

if self.state == 'wander_for_trash'
    if trash.visible == true, then self.state = 'move_to_trash'
    else do_search_movement()

elseif self.state == 'move-to-trash'
    if trash.withinreach == true then self.state = 'grab_trash'
    else do_homing_movement (trash.location)

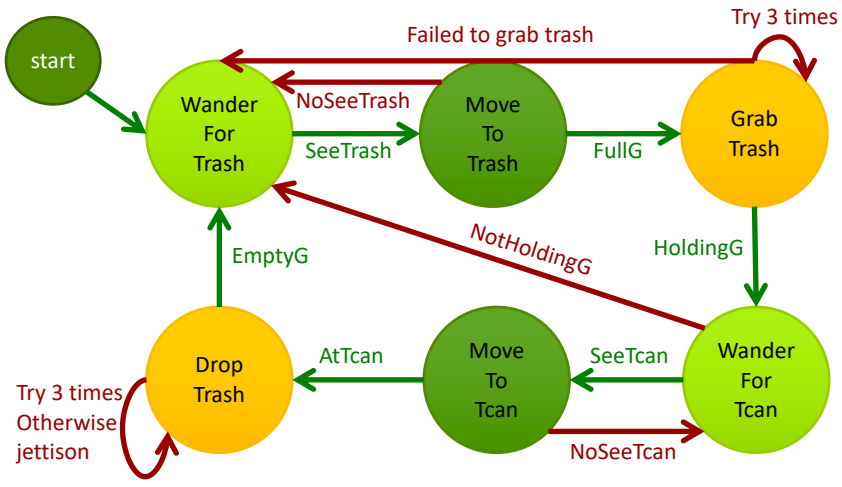
elseif self.state == 'grab_trash'
    if trash.grabbed == true, then self.state = 'wander_to_tcan'
    else do_grab()

elseif self.state == 'wander_to_tcan'
    if tcan.visible == true, then self.state= 'move_to_tcan'
    else do_search_movement()

..... And so on
Camera thread is always looking for trash and trashcan, and setting the visibility and location variables for both.

PATTERN
\When in state X
- Define transitions out of state
- Otherwise do X behavior

# Solution 1: Finite State Machine



# Solution 1: Finite State Machine

**# Finite State Machine LOOP**

if self.state == 'wander_for_trash'
    if trash.visible == true, then self.state = 'move_to_trash'
    else do_search_movement()

elseif self.state == 'move-to-trash'
    if trash.visible == false, then self.state = 'wander_for_trash'
    if trash.withinreach == true then self.state = 'grab_trash'
    else do_homing_movement (trash.location)

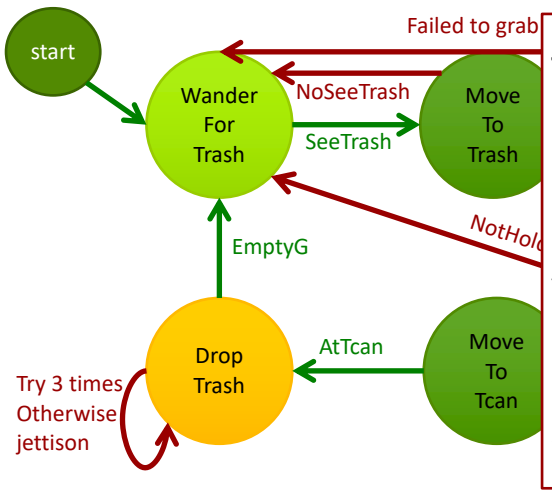
....

**Adding Collision Avoidance and Safety States**
If motion_obstructed
    do evasive maneuver instead of running finite state machine

if cliff_detected == true
    self.resume_state = self.state    #remember what you were doing
    self.state = 'STOPPED'
if self.state == STOPPED
    if cliff_detected == false, then self.state = self.resume_state
    else motors_off()

PATTERN
\When in state X
- Define transitions out of state (including error transitions)
- Otherwise do X behavior

# Solution 1: Finite State Machine




**Things to Note:**
1: Uses Internal State
2: Defines a Sequence

**Issues:**
1: Things may not happen in the expected order (explicitly anticipate problems)

2: Behaviors like *Avoid Obstacles* and *Reset* underlie everything

# Solution 2: Parallel Behaviors


# Solution 2: Parallel Behaviors

**# PARALLEL BEHAVIORS**

If about_to_collide == true
    suggest(avoid_behavior)

if trash.visible == false and tcan.visible == false
    suggest(wander_movement)

If trash.visible == true and trash.withinreach == false
  and trash.grabbed == false
    suggest(move_to_trash)

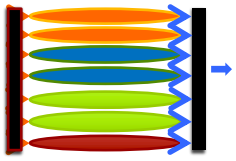
If tcan.visible == true and trash.grabbed == true
    suggest(move_to_tcan)

If trash.withinreach == true
    suggest(grab_trash)

If tcan.withinreach == true and trash.grabbed == true
    suggest(drop_trash)

**# ARBITOR**

Take all the suggested movements and pick based on some priority



PATTERN

**There's no explicit self.state**
(no memory of what you were doing in the last round…)

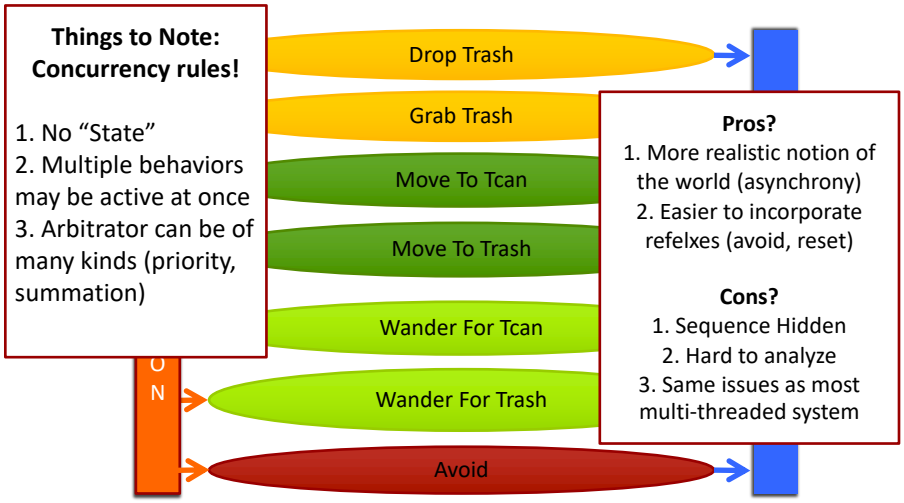
**All conditions are set by Sensors**
(camera, bump, cliff threads)

Many actions might be suggested
**Arbitor decides what to do**

# Solution 2: Parallel Behaviors

**Things to Note: Concurrency rules!**

1. No "State"
2. Multiple behaviors may be active at once
3. Arbitrator can be of many kinds (priority, summation)




**Pros?**
1. More realistic notion of the world (asynchrony)
2. Easier to incorporate refelxes (avoid, reset)

**Cons?**
1. Sequence Hidden
2. Hard to analyze
3. Same issues as most multi-threaded system

# Solution 3: Sense-Plan-Act

| Sense | Plan | Act |
|---|---|---|
| Generate "world model" | Generate plan | Execute each step of plan |

**Sense and Construct Model of the World**

- Camera: rotate in a circle to get 360 view
- Identify all visible cans, and trashcan, and esti
- `World Model: CANS ((x1,y1) (x2,y2) (x3 y3)`

**Plan a Sequence:** get each can and take them

- `Plan: Move-to-location(x1,y1), Grab Can, M location(x4,y4), DropCan, Move-to-location`

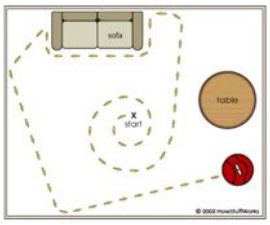
**Act:** each action corresponds to a "behavior" that

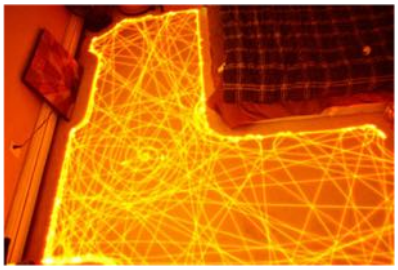
- *Example* `Move-to-location:` plan a path to a specific location

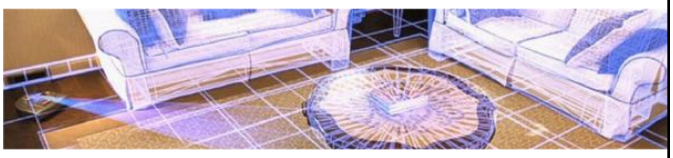
**Pros?**
**Efficiency** by being smart & looking ahead
**Generalize:** New Task

**Cons?**
Assumption of perfect (error-free) perception and actuation

# Roomba vs Neato





# Example: Trash Collector

- Three Different Ways to Assemble Behaviors
  - Finite-State-Machine (Hybrid Architecture)
  - Parallel Behaviors (Reactive/Behavior-based/Subsumption)
  - Sense-Plan-Act (Deliberative Architecture)

History of AI-robotics is intimately tied with the question of how to program complex behaviors

Brief But Gripping History of Robotics (based on Mataric 2007)

5 MINUTE BREAK

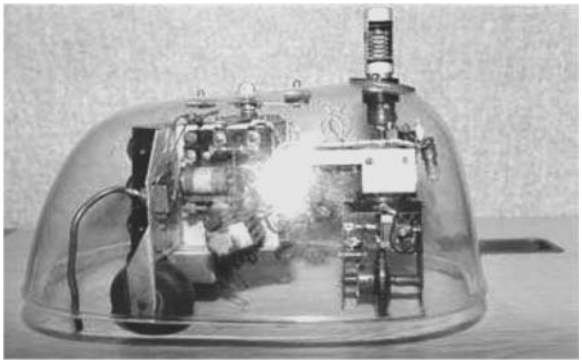
## Brief History

# Brief History

- The Word Robot
  - Czech playwright Karel Capek in his 1921 play "Rossum's Universal Robots"
    *Rabota = menial work; Robotnik = serf*
  - The word "Robotics" believed to be coined by Isaac Asimov

- Fields of Study Prior to Robotics and AI
  - **Control Theory** (Ancient!)
  - **Industrial Revolution** (Automation of human tasks)
  - **Cybernetics** (1940s, neuroscience and engineering)

# Grey Walter's Tortoises (1940s)



**Grey Walter**
(1910-1977; 1940s)
Neurophysiologist, Cyberneticist

Simple robots based on reflexes

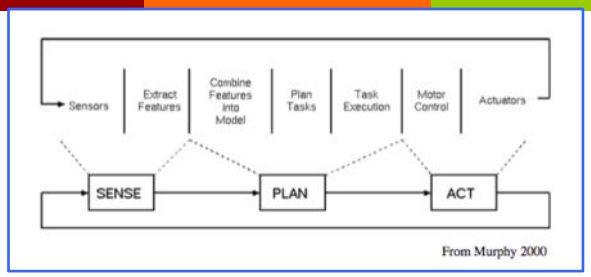
*Before AI existed*

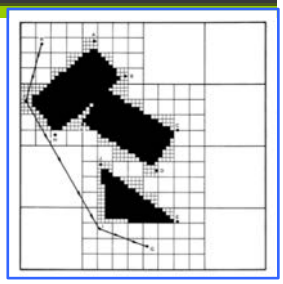
*Elmer (Electro Mechanical Robots)*
*Elsie (Light Sensitive)*
*Machina Docilis (Tameable)*

# Birth of AI (1956) and Shakey (1960s)

- **Birth of Artificial Intelligence 1956**
  - Dartmouth Conference: Marvin Minsky, John McCarthy, Alan Newell, Herbert Simon, etc)
  - *Theme: Intelligent Agents*
    - Internal Models of the world, Reason to solve problems, Symbolic representation of information
    - Intelligence == Playing chess, solving theorems
- **Shakey Robot (1960s-1970s)**
  - Stanford Research Institute (SRI)
  - Reason about its own actions
    (given a goal, construct *its own* plan)
  - Ambitious! included vision, natural language, A* search, STRIPs planning language
- **Stanford CART (1977, Hans Moravec)**
  - Birth of mobile robotics as we know it today

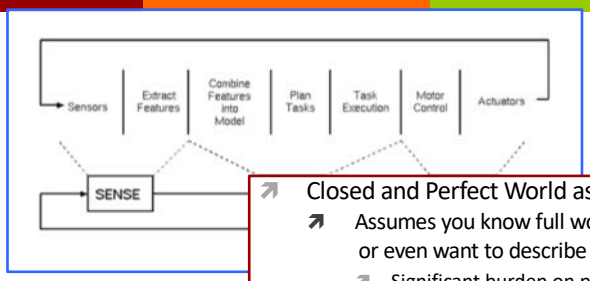
# Shakey

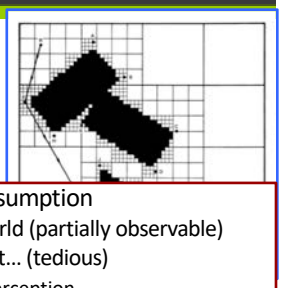


- World Model: Planar Grid Map with space, obstacles, and unknowns (Perception!!)
- Plans: generated using a STRIPs Planner ("theorem-proving")
  - Goal: Move three objects to a common location
    **Exists (p,s), s.t. pos(OB1,s) and pos(OB2, s) and pos(OB3,s)**
    (aka There exists a situation s and place p, such that OBI, OB2, and OB3 are all at place p in situation)
  - The task for the planner is to "prove" that this conjecture follows from axioms that describe present position of objects & effects of actions.

# Shakey



- World Model: Planar G
- Plans: generated using
  - Goal: Move three ob
    Exists (p,s), s.t. pos(OB1
    There exists a situation s
  - The task for the plar
    describe present po
- Closed and Perfect World assumption
  - Assumes you know full world (partially observable)
    or even want to describe it… (tedious)
    - Significant burden on perception
  - But also Planner relies on a model of how the world works [(put A on B) => AonB]
    - Deriving a reasonable model of world v. hard!
- Too much planning only to fail on the first step…
  (replanning all the time without getting anything done)

# Even a simple world is complex

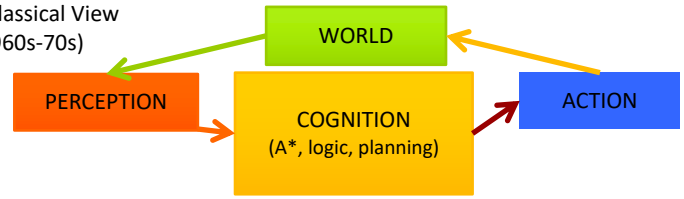
The initial environment of the Automaton was real, but contrived. It has been sufficiently simple to allow current visual capabilities to be useful to the Automaton, and sufficiently complex to indicate the weaknesses of current methods and to suggest areas of further research. Perhaps the most important result of our vision-research effort on the Automaton project is an appreciation of the potential complexity of the problem of vision when the real world is the subject matter, and a strong notion that the first step we have taken towards a general capability is very small indeed.

Nilsson, 1975, SRI Technical report

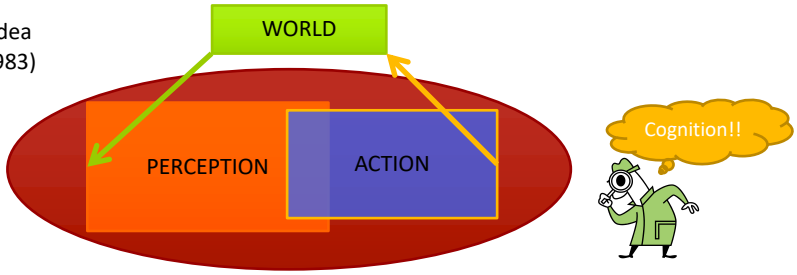
**Moravec's Paradox**
it is comparatively easy to make computers exhibit adult level performance on intelligence tests or playing checkers ---- and difficult or impossible to give them the skills of a one-year-old when it comes to perception and mobility

# An Alternative View (1980s)

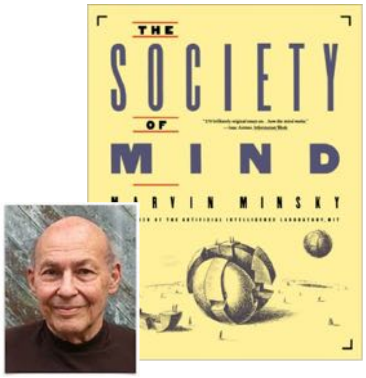
The Classical View
(1960s-70s)




A Radical Idea
(Brooks, 1983)

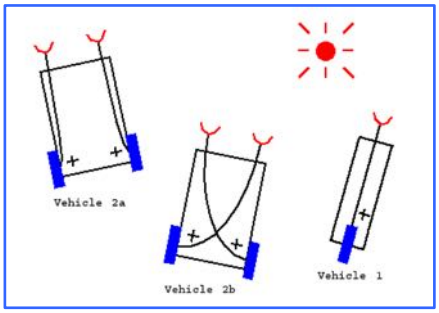



# Related Influences in 1980s





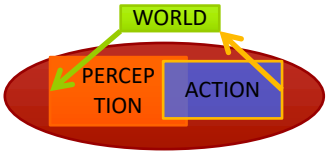

**Society of Mind (Minsky, 1986)**
Idea that human intelligence is not a centrally controlled system, but rather a *confederate of competing interests*

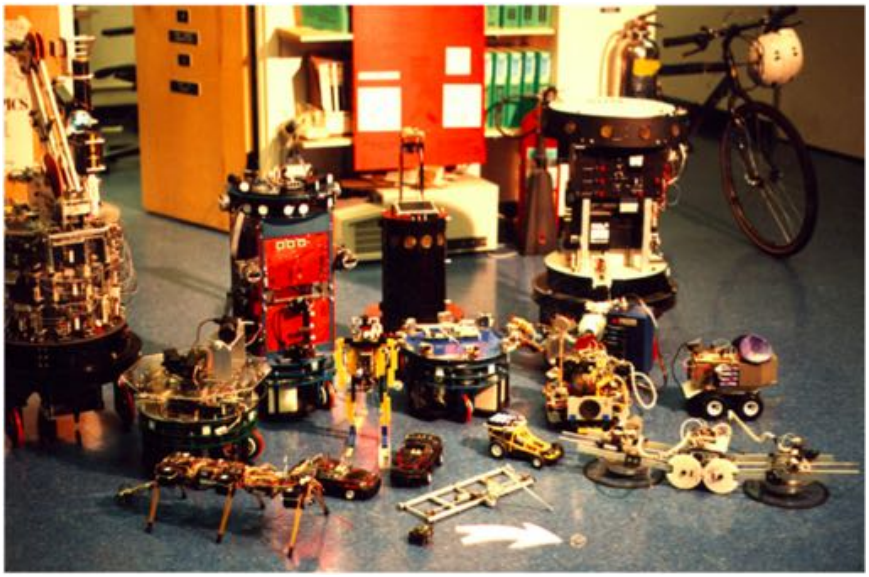



**Braitenberg Vehicles:**
Experiments in Synthetic Psychology, MIT Press [1984]

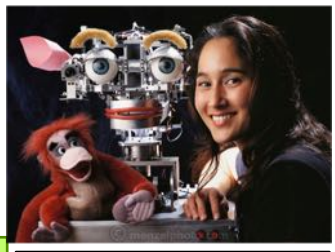
# Behavior-based Robotics (1980s-90s)

- Rodney Brooks
  - *Started Behavior-based Robotics movement*
  - PhD at Stanford, 1981; Joined MIT 1984
  - MIT CSAIL director, 1997-2008
  - IJCAI Computers and Thought award (1991)
  - Co-Founder of iRobot (1991), Rethink Robotics (2009)
- Wrote Papers with Provocative titles!
  - *Elephants don't play chess*
  - *Intelligence without representation*
  - *Intelligence without without reason*
  - *Planning is a just a way to avoid figuring out what to do next…*
- Robots
  - Incredible number of robots to prove his point
  - And produced a generation of famous students



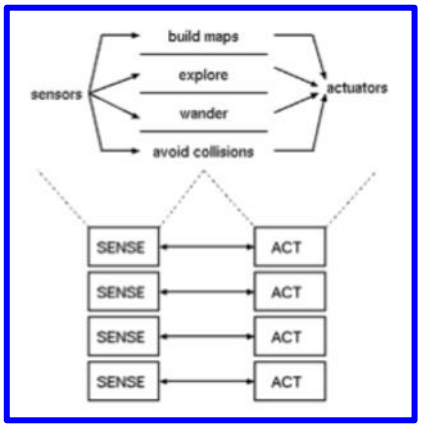


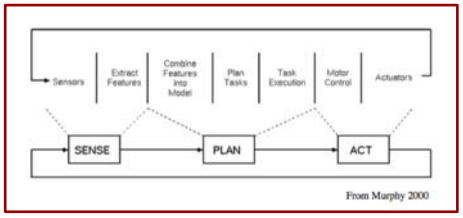






**Generation of Robotics Faculty and Entrepreneurs (to name a few)**
Maja Mataric (USC)
Yoky Matsuok (Uwash, NEST, MacArthur Genuis)
Brian Scasellati (Yale)
Cynthia Brazael (MIT, Founder Jibo)
Aaron Edsinger (Founder Meka Robotics)
James McLurkin (Rice, Google)
Colin Angle (iRobot)
Helen Greiner (Founder iRobot, CyPhyWorks)
Holly Yanco (Umass)
[clockwise]

# Basic Idea: Reactive Architectures

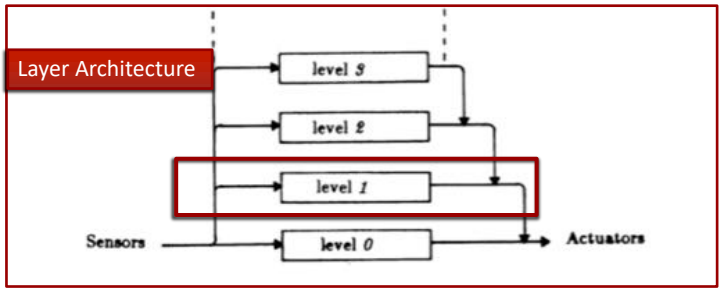



OLD Model

**NEW Model**
Concurrent "Behaviors"
Every behavior can access raw inputs, and affect raw outputs…)

# "Subsumption" Architecture

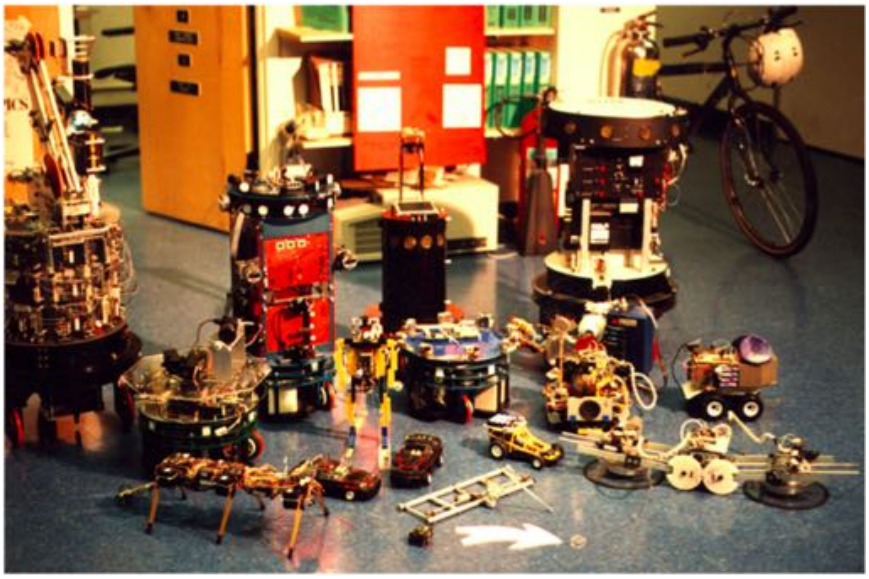



- **Rod Brooks defined the "Subsumption" Architecture**
  - Layers of Competency; Inhibit each other
- **Behavior-based Control** (Ron Arkin, Maja Mataric, etc)
  - E.g. Adding an "arbitrator" box between behaviors and actuators

# Genghis and other "Veterans"

- Six-legged Robot
  - Many degrees of freedom, many sensors (hard!)
    - Standup layer
    - Simple Walk layer
    - Balance/Pitch Layer
    - Whiskers Behavior
    - Steer, Prowl (follow)
    - Every layer " subsumes" other!
    - "Decentralized intelligence" (seemingly purposeful behavior)

Decentralized Control for a Six-legged Robot (1989)



# Behavior-Based Robotics (1980s-90s)

- Key Ideas
  - Behaviors: Concurrent and Layered
  - Sensor->Action relationship in all behaviors
  - Avoid memory and internal state, Avoid symbols
- Key Limitations
  - No systematic design methodology, Hard to analyze
  - Pre-planning is helpful in certain cases ("Hybrid" architectures)
- But Many Lasting Ideas: "*The world is its own best model*"
  - Avoid world->symbolic transformation
  - Assume highly Dynamic world (less reliance on knowing and memory)
  - Perception is direct, distributed, and *ego-centric*

# Modern Robotics (2000-now)

- **Competitions moved the field forward**
  - Darpa Urban Challenge (self-driving cars)
  - Robot Soccer (goal: beat human champions in 2050)
  - AAAI Competitions ("Hor D' Oeuvres Anyone?")
- **Many new architectural ideas…**
  - *Probabilistic Reasoning for Navigation*
  - *Coordination Architectures (Role-based, Play-based, etc)*
  - *Human-Robot Interaction Architectures*

# Probabilistic Robotics


- Prof. Sebastian Thrun, Stanford
  - E.g. FastSLAM = Simultaneous Localization and mapping

# Multi-Robot Systems




***Grand Challenge for robotics and AI:*** *By the year 2050, team of fully autonomous humanoid robots that can win against the human world soccer champion team.*
(Prof Manuela Veloso, CMU, co-founder)

# Personal Robots

GRACE, 2002

KISMET(MIT) to Jibo

Diligent Robotics

# Modern Robotics (2000-now)

- Competitions moved the field forward
  - AAAI Scavenger hunts and "Hor D' Oeuvres Anyone?"
  - Robot Soccer (beat human champions in 2050)
  - Darpa Urban Challenge (self-driving cars)
- Many new architectural ideas…
  - Probabilistic Reasoning for Navigation
  - Coordination Architectures (Role-based, Play-based, etc)
  - Human-Robot Interaction Architectures
- **History of Robotics is far from "written"**
  - **Most debates on autonomy remain open**