CSc 84200
artificial intelligence and robotics seminar
fall 2006
lecture # 1
introduction

topics:

(0) introduction to the course

(1) autonomous agents and autonomous robotics

(2) behaviour-based robotics

instructors:

- Prof Elizabeth Sklar, *sklar@sci.brooklyn.cuny.edu*
- Prof Simon Parsons, *parsons@sci.brooklyn.cuny.edu*

course web page:

- `http://www.sci.brooklyn.cuny.edu/~sklar/air`

introduction to the course

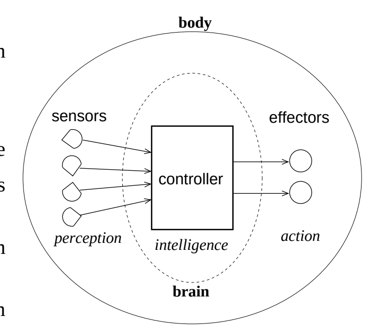
- seminar in 14 sessions:
  - 1 lecture (today); 12 paper presentations and discussions; 1 project day
- assessment:
  - 1 paper presentation (20%); 1 discussion leader (20%); in-class participation (20%); final project (20% written, 20% oral)
- topics:
  - classic robotics
    (vision; localization, mapping, SLAM; path planning)
  - emerging topics
    (coordination; humanoids; human-robot interaction; learning; evolutionary robotics)
  - applications
    (robocup soccer; robot rescue; darpa grand challenge; educational robotics)

(1) autonomous agents and autonomous robotics

- we will be discussing *autonomous mobile robots*
- what is a robot?
  - "a programmable, multifunction manipulator designed to move material, parts, tools or specific devices through variable programmed motions for the performance of various tasks." [Robot Institute of America]
  - "an active, artificial *agent* whose environment is the physical world" [Russell&Norvig, p773]
- what is an agent?
  - "anything that can be viewed as perceiving its environment through sensors and acting upon that environment through effectors." [Russell&Norvig, p32]
- what is autonomy?
  - no remote control!!
  - an agent makes decisions on its own, guided by feedback from its sensors; but you write the program that tells the agent how to make its decisions environment.

(1) our definition of a *robot*

- *robot = autonomous embodied agent*
- has a *body* and a *brain*
- exists in the physical world (rather than the virtual or simulated world)
- is a mechanical device
- contains *sensors* to perceive its own state
- contains *sensors* to perceive its surrounding environment
- possesses *effectors* which perform actions
- has a *controller* which takes input from the sensors, makes *intelligent* decisions about actions to take, and effects those actions by sending commands to motors



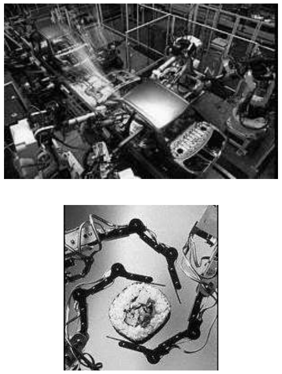
our canonical agent

(1) a bit of robot history

- the word *robot* came from the Czech word *robota*, which means *slave*
- used first by playwrite Karel Capek, “Rossum’s Universal Robots” (1923)
- human-like automated devices date as far back as ancient Greece
- modern view of a robot stems from science fiction literature
- foremost author: Isaac Asimov, “I, Robot” (1950)
- the *Three Laws of Robotics*
  1. A robot may not injure a human being, or, through inaction, allow a human being to come to harm.
  2. A robot must obey the orders given it by human beings except where such orders would conflict with the First Law.
  3. A robot must protect its own existence as long as such protection does not conflict with the First or Second Law.
- Hollywood broke these rules: e.g., “The Terminator” (1984)

(1) effectors

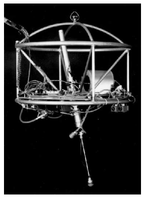
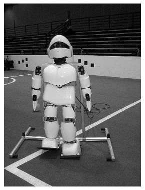
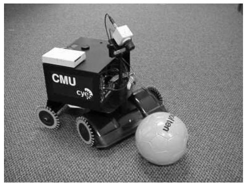
- comprises all the mechanisms through which a robot can *effect* changes on itself or its environment
- *actuator* = the actual mechanism that enables the effector to execute an action; converts software commands into physical motion
- types:
  - arm
  - leg
  - wheel
  - gripper
- categories:
  - *manipulator*
  - *mobile*



some manipulator robots

(1) mobile robots

- classified by manner of locomotion:
  - *wheeled*
  - *legged*
- stability is important
  - *static stability*
  - *dynamic stability*

(1) *degrees of freedom*

- number of directions in which robot motion can be controlled
- free body in space has 6 degrees of freedom:
  - three for position (x, y, z)
  - three for orientation $(roll, pitch, yaw)$
    - yaw refers to the direction in which the body is facing i.e., its orientation within the xy plane
    - $roll$ refers to whether the body is upside-down or not i.e., its orientation within the yz plane
    - $pitch$ refers to whether the body is tilted i.e., its orientation within the xz plane
- if there is an actuator for every degree of freedom, then all degrees of freedom are controllable $\Rightarrow$ *holonomic*
- most robots are *non-holonomic*

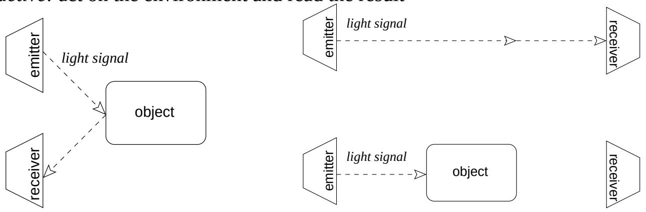
(1) sensors

- ⇒ perception
  - *proprioceptive:* know where your joints/sensors are
  - *odometry:* know where you are
- function: to convert a physical property into an electronic signal which can be interpreted by the robot in a useful way

property being sensed	type of sensor
contact	bump, switch
distance	ultrasound, radar, infra red (IR)
light level	photo cell, camera
sound level	microphone
smell	chemical
temperature	thermal
inclination	gyroscope
rotation	encoder
pressure	pressure gauge
altitude	altimeter

(1) more on sensors

- operation
  - *passive*: read a property of the environment
  - *active*: act on the environment and read the result



reflectance *break-beam*

- noise
  - *internal*: from inside the robot
  - *external*: from the robot's environment
  - *calibration*: can help eliminate/reduce noise

(1) environment

- *accessible* vs *inaccessible*
  - robot has access to all necessary information required to make an informed decision about to do next
- *deterministic* vs *nondeterministic*
  - any action that a robot undertakes has only one possible outcome.
- *episodic* vs *non-episodic*
  - the world proceeds as a series of repeated episodes.
- *static* vs *dynamic*
  - the world changes by itself, not only due to actions effected by the robot
- *discrete* vs *continuous*
  - sensor readings and actions have a discrete set of values.

(1) state

- knowledge about oneself and one's environment
  - *kinematics* = study of correspondance between actuator mechanisms and resulting motion
    - ∗ motion:
      - · rotary
      - · linear
  - combines sensing and acting
  - *did i go as far as i think i went?*
- but one's environment is full of information
- for an agent, what is relevant?

(1) control

- autonomy
- problem solving
- modeling
  - knowledge
  - representation
- control architectures
- deliberative control
- reactive control
- hybrid control

(1) autonomy

- to be truly autonomous, it is not enough for a system simply to establish direct numerical relations between sensor inputs and effector outputs
- a system must be able to accomplish *goals*
- a system must be able to *solve problems*
- ⇒ need to represent problem space
  - which contains goals
  - and intermediate states
- there is always a trade-off between *generality* and *efficiency*
  - more specialized ⇒ more efficient
  - more generalized ⇒ less efficient

(1) problem solving: example

- GPS = General Problem Solver [Newell and Simon 1963]
- Means-Ends analysis

operator	*preconditions*	*results*
$PUSH(obj, loc)$	$at(robot, obj) \wedge large(obj) \wedge$ $clear(obj) \wedge armempty()$	$at(obj, loc) \wedge$ $at(robot, loc)$
$CARRY(obj, loc)$	$at(robot, obj) \wedge small(obj)$	$at(obj, loc) \wedge$ $at(robot, loc)$
$WALK(loc)$	$none$	$at(robot, loc)$
$PICKUP(obj)$	$at(robot, obj)$	$holding(obj)$
$PUTDOWN(obj)$	$holding(obj)$	$\neg holding(obj)$
$PLACE(obj1, obj2)$	$at(robot, obj2) \wedge holding(obj1)$	$on(obj1, obj2)$

(1) modeling the robot's environment

- modeling
  - the way in which *domain knowledge* is embedded into a control system
  - information about the environment stored internally: *internal representation*
  - e.g., maze: robot stores a *map* of the maze "in its head"
- knowledge
  - information in a context
  - organized so it can be readily applied
  - understanding, awareness or familiarity acquired through learning or experience
  - physical structures which have correlations with aspects of the environment and thus have a predictive power for the system

(1) memory

- divided into 2 categories according to duration
- *short term memory (STM)*
  - transitory
  - used as a buffer to store only recent sensory data
  - data used by only one behaviour
  - examples:
    - *avoid-past*: avoid recently visited places to encourage exploration of novel areas
    - *wall-memory*: store past sensor readings to increase correctness of wall detection
- *long term memory (LTM)*
  - persistent
  - *metric maps*: use absolute measurements and coordinate systems
  - *qualitative maps*: use landmarks and their relationships
  - examples:
    - *Markov models*: graph representation which can be augmented with probabilities for each action associated with each sensed state

(1) knowledge representation

- must have a relationship to the environment (temporal, spatial)
- must enable predictive power (look-ahead), but if inaccurate, it can deceive the system
- *explicit*: symbolic, discrete, manipulable
- *implicit*: embedded within the system
- *symbolic*: connecting the meaning (semantics) of an arbitrary symbol to the real world
- difficult because:
  - sensors provide signals, not symbols
  - symbols are often defined with other symbols (circular, recursive)
  - requires interaction with the world, which is noisy
- other factors
  - speed of sensors
  - response time of effectors

(1) components of knowledge representation

- *state*
  - totally vs partially vs un- observable
  - discrete vs continuous
  - static vs dynamic
- *spatial*: navigable surroundings and their structure; metric or topological maps
- *objects*: categories and/or instances of detectable things in the world
- *actions*: outcomes of specific actions on the self and the environment
- *self/ego*: stored proprioception (sensing internal state), self-limitations, capabilities
  - *perceptive*: how to sense
  - *behaviour*: how to act
- *intentional*: goals, intended actions, plans
- *symbolic*: abstract encoding of state/information

(1) types of representations

- maps
  - *euclidean map*
    - represents each point in space according to its metric distance to all other points in the space
  - *topological map*
    - represents locations and their connections, i.e., how/if they can be reached from one another; but does not contain exact metrics
  - *cognitive map*
    - represents behaviours; can store both previous experience and use for action
    - used by animals that forage and home (animal navigation)
    - may be simple collections of vectors
- graphs
  - nodes and links
- Markov models
  - associates probabilities with states and actions

(1) control architecture

- a control architecture provides a set of principles for organizing a control system
- provides structure
- provides constraints
- refers to software control level, not hardware!
- implemented in a programming language
- don't confuse "programming language" with "robot architecture"
- architecture guides how programs are structured

(1) classes of robot control architectures

- *deliberative*
  - look-ahead; think, plan, then act
- *reactive*
  - don't think, don't look ahead, just react!
- *hybrid*
  - think but still act quickly
- *behaviour-based*
  - distribute thinking over acting

(1) deliberative control

- classical control architecture (first to be tried)
- first used in AI to reason about actions in non-physical domains (like chess)
- natural to use this in robotics at first
- example: Shakey (1960's, SRI)
  - state-of-the-art machine vision used to process visual information
  - used classical planner (STRIPS)
- planner-based architecture
  1. sensing (S)
  2. planning (P)
  3. acting (A)
- requirements
  - lots of time to think
  - lots of memory
  - (but the environment changes while the controller thinks)

(1) reactive control

- operate on a short time scale
- does not look ahead
- based on a tight loop connecting the robot's sensors with its effectors
- purely reactive controllers do not use any internal representation; they merely react to the current sensory information
- collection of rules that map situations to actions
  - simplest form: divide the perceptual world into a set of mutually exclusive situations recognize which situation we are in and react to it
  - (but this is hard to do!)
- example: subsumption architecture (Brooks, 1986)
  - hierarchical, layered model

(1) hybrid control

- use the best of both worlds (deliberative and reactive)
- combine open-loop and closed-loop execution
- combine different time scales and representations
- typically consists of three layers:
  1. reactive layer
  2. planner (deliberative layer)
  3. integration layer to combine them
  4. (but this is hard to do!)

(2) behaviour-based robotics

- control
- behaviour-based systems
- expressing behaviours
- behavioural encoding
- representations
- behaviour coordination
- emergent behaviour

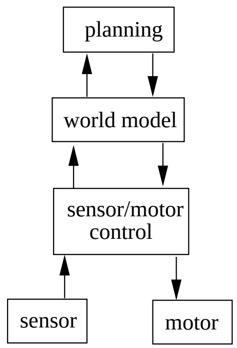
(2) control: models

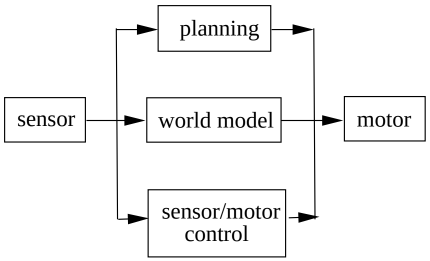
- we like to make a distinction between:
  - classic "model-based" AI
    - symbolic representations
  - neo "behaviour-based" AI
    - numeric representations
- classic models are "good old-fashioned AI", in the tradition of McCarthy.
- behaviour-based models are "nouvelle AI" in the tradition of Brooks.
- (There are also hybrid models that combine aspects of both model-based and behaviour-based, but we have no time to cover them in detail here.)

(2) control: classic models

- deliberative... sense, plan, act
  - functional decomposition
  - systems consist of sequential modules achieving independent functions
    - sense world
    - generate plan
    - translate plan into actions
- reactive architectures
  - task-oriented decomposition
  - systems consist of concurrently executed modules achieving specific tasks
    - avoid obstacle
    - follow wall

(2) control: two orthogonal flows





(2) control: behaviour based systems

- behaviours are the underlying module of the system
- behavioural decomposition
  - rather than a functional or a task-oriented decomposition
- systems consist of sequential modules achieving independent functions
- natural fit to robotic behaviour
  - generate a motor response from a given perceptual stimulus
  - basis in biological studies
  - biology is an inspiration for design
- abstract representation is avoided

(2) behaviour based systems: behaviour vs action

behaviour is:

- based on dynamic processes
  - operating in parallel
  - lack of central control
  - fast couplings between sensors and motors
- exploiting emergence
  - side-effects from combined processes
  - using properties of the environment
- reactive

(2) behaviour-based systems: behaviour vs action

action is:

- discrete in time
  - well defined start and end points
  - allows pre- and post-conditions
- avoidance of side-effects
  - only one (or a few) actions at a time
  - conflicts are undesired and avoided
- deliberative

actions are building blocks for behaviours.

(2) behaviour-based systems: properties

- achieve specific tasks/goals
  - avoid others, find friend, go home
- typically execute concurrently
- can store state and be used to construct world models/representations
- can directly connect sensors to effectors
- can take inputs from other behaviours and send outputs to other behaviours
  - connection in networks
- typically higher-level than actions (go home, not turn left 45 degrees)
- typically closed loop, but extended in time
- when assembled into distributed representations, behaviours can be used to look ahead but at a time-scale comparable with the rest of the system

(2) behaviour-based systems: key properties

- ability to act in real time
- ability to use representations to generate efficient (not only reactive) behaviour
- ability to use a uniform structure and representation throughout the system (so no intermediate layer)

(2) behaviour-based systems: challenges

- how can representation be effectively distributed over the behaviour structure?
  - time scale must be similar to that of real-time components of the system
  - representation must use same underlying behaviour structure for all components of the system
- some components may be reactive
- not every component is involved with representational computation
- some systems use a simple representation
- as long as the basis is in behaviours and not rules, the system is a BBS

(2) behaviour-based systems: what are behaviours?

- behaviour: anything observable that the system/robot does
  - how do we distinguish internal behaviours (components of a BBS) and externally observable behaviours?
  - should we distinguish?
- reactive robots display desired external behaviours
  - avoiding
  - collecting cans
  - walking
- but controller consists of a collection of rules, possibly in layers
- BBS actually consist and are programmed in the behaviours, which are higher granularity, extended in time, capable of representation

(2) behaviour-based systems: expressing behaviours

- behaviours can be expressed with various representations
- when a control system is being designed, the task is broken down into desired external behaviours
- those can be expressed with
  - functional notation
  - stimulus response (SR) diagrams
  - finite state machines/automata (FSA)
  - schema

(2) expressing behaviours: functional notation

- mathematical model:
  - represented as triples (S, R, β)
    S = stimulus
    R = range of response
    β = behavioural mapping between S and R
- easily convert to functional languages like LISP

```
coordinate-behaviours [
   move-to-classroom ( detect-classroom-location ),
   avoid-objects ( detect-objects ),
   dodge-students ( detect-students ),
   stay-to-right-on-path ( detect-path ),
   defer-to-elders ( detect-elders )
] = motor-response
```

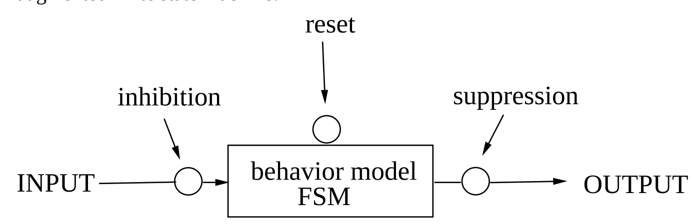

(2) expressing behaviours: FSA diagrams

- states of the diagram can also be called behaviours, diagrams show sequences of behaviour transitions
- situated automata.
  - formalism for specifying FSAs that are situated [Kaelbling & Rosenschein, 1991]
  - task described in high-level logic expressions, as a set of goals and a set of operators that achieve (ach) and maintain (maint) the goals
  - once defined, tasks can be compiled into circuits (using special purpose languages), which are reactive

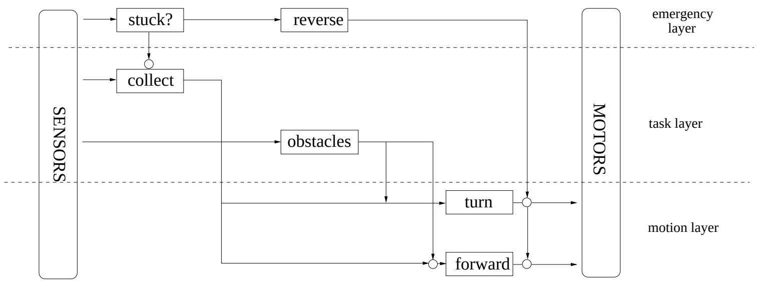
```
(defgoalr (ach in-classroom)
(if (not start-up)
    (maint (and (maint move-to-classroom)
                (maint avoid-objects)
                (maint dodge-students)
                (maint stay-to-right-on-path)
                (maint defer-to-elders)))))
```


(2) expressing behaviours: subsumption architecture

- Rodney Brooks, 1986, MIT AI lab
- reactive elements
- behaviour-based elements
- layered approach based on levels of competence
- augmented finite state machine:



(2) expressing behaviours: subsumption architecture



(2) expressing behaviours: formally

- behavioural response in physical space has a strength and an orientation
- expressed as (S, R, β)
- S = stimulus, necessary but not sufficient condition to evoke a response (R); internal state can also be used
- β = behavioural mapping categories
  - null
  - discrete
  - continuous

(2) expressing behaviours: behavioural mapping

- discrete encoding
  - expressed as a finite set of situation-response pairs/mappings
  - mappings often include rule-based form IF-THEN
  - examples:
    - Gapps [Kaelbling & Rosenschein]
    - subsumption language [Brooks]
- continuous encoding
  - instead of discretizing the input and output, a continuous mathematical function describes the input-output mapping
  - can be simple, time-varying, harmonic
  - examples:
    - potential field
    - schema
  - problems with local minima, maxima, oscillatory behaviour

(2) expressing behaviours: motor schemas

- type of behaviour encoding
- based on schema theory
- provide large grain modularity
- distributed, concurrent schemas used
- based on neuroscience and cognitive science
- represented as vector fields
- decomposed into *assemblages* by fusion, not competition
- assemblages
  - recursively defined aggregations of behaviours or other assemblages
  - important abstractions for constructing behaviour-based robots

(2) expressing behaviours: schemas

- representation
  - responses represented in uniform vector format
  - combination through cooperative coordination via vector summation
  - no predefined schema hierarchy
  - arbitration not used — gain values control behavioural strengths
- designing with schemas
  - characterize motor behaviours needed
  - decompose to most primitive level, use biological guidelines where appropriate
  - develop formulas to express reactions
  - conduct simple simulations
  - determine perceptual needs to satisfy motor schema inputs
  - design specific perpetual algorithms
  - integrate/test/evaluate/iterate

(2) behavioural encoding: strengths and weaknesses

- strengths
  - support for parallelism
  - run-time flexibility
  - timeliness for development
  - support for modularity
- weaknesses
  - niche targetability
  - hardware retargetability
  - combination pitfalls (local minima, oscillations)

(2) behaviour coordination

- BBS consist of collection of behaviours
- execution must be coordinated in a consistent fashion
- coordination can be
  - competitive
  - cooperative
  - combination of the two
- deciding what to do next.
  - action-selection problem
  - behaviour-arbitration problem

(2) behaviour coordination

- competitive coordination.
  - perform arbitration (selecting one behaviour among a set of candidates)
    - priority-based: subsumption
    - state-based: discrete event systems
    - function-based: spreading of activation action selection
- cooperative coordination.
  - perform command fusion (combine outputs of multiple behaviours)
  - voting
  - fuzzy (formalized voting)
  - superposition (linear combinations)
    - potential fields
    - motor schemas
    - dynamical systems

(2) emergent behaviour: what is it?

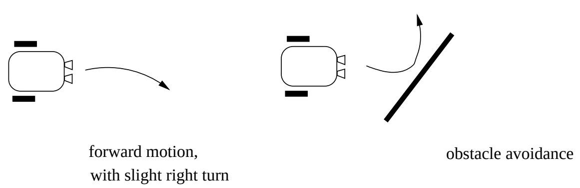
- important but not well-understood phenomenon
- often found in behaviour-based robotics
- robot behaviours "emerge" from
  - interactions of rules
  - interactions of behaviours
  - interactions of either with environment
- coded behaviour
  - in the programming scheme
- observed behaviour
  - in the eyes of the observer
  - emergence
- there is no one-to-one mapping between the two!

(2) emergent behaviour: how does it arise?

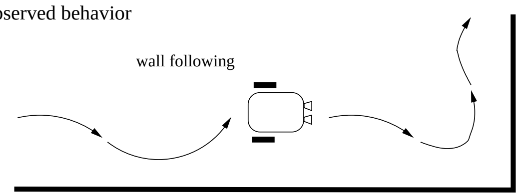
- is it magic?
  - sum is greater than the parts
  - emergent behaviour is more than the controller that produces it
- interaction and emergence.
  - interactions between rules, behaviours and environment
  - source of expressive power for a designer
  - systems can be designed to take advantage of emergent behaviour
- emergent flocking.
  - program multiple robots:
    * dont run into any other robot
    * dont get too far from other robots
    * keep moving if you can
  - when run in parallel on many robots, the result is flocking

(2) emergent behaviour: wall following

coded behavior



observed behavior



(2) emergent behaviour: wall following

- can also be implemented with these rules:
  - if too far, move closer
  - if too close, move away
  - otherwise, keep on
- over time, in an environment with walls, this will result in wall-following
- is this emergent behaviour?
- it is argued yes because
  - robot itself is not aware of a wall, it only reacts to distance readings
  - concepts of "wall" and "following" are not stored in the robot's controller
  - the system is just a collection of rules

(2) emergent behaviour: conditions on emergence

- notion of emergence depends on two aspects:
  - existence of an external observer, to observe and describe the behaviour of the system
  - access to the internals of the controller itself, to verify that the behaviour is not explicitly specified anywhere in the system
- unexpected vs emergent.
  - some researchers say the above is not enough for behaviour to be emergent, because above is programmed into the system and the "emergence" is a matter of semantics
  - so emergence must imply something unexpected, something "surreptitiously discovered" by observing the system.
  - "unexpected" is highly subjective, because it depends on what the observer was expecting
  - naïve observers are often surprised!
  - informed observers are rarely surprised
- once a behaviour is observed, it is no longer unexpected
- is new behaviour then "predictable"?