Review

- Expression of behaviors
  - Stimulus Response
  - Finite State Acceptor
  - Situated Automata
- Behavioral encoding
  - Discrete: rule-based systems
  - Continuous: potential fields, motor schemas
- Behavior coordination

Emergent Behavior

- The resulting robot behavior may sometimes be surprising or unexpected $\Rightarrow$ emergent behavior
- Emergence arises from
  - A robot's interaction with the environment
  - The interaction of behaviors

Emergence

- A “holistic” property, where the behavior of the robot is greater than the sum of its parts
- A property of a collection of interacting components
- Often occurs in reactive and behavior-based systems (BBS)
- Typically exploited in reactive and BBS design

Emergent Behavior

- **Emergent behavior** is structured behavior that is apparent at one level of the system (the observer's point of view) and not apparent at another (the controller's point of view)
- The robot generates interesting and useful behavior without explicitly being programmed to do so!!
- E.g.: Wall following can emerge from the interaction of the avoidance rules and the structure of the environment

Components of Emergence

- The notion of emergence depends on two components
  - The existence of an external observer, to observe the emergent behavior and describe it
  - Access to the internals of the controller, to verify that the behavior is not explicitly specified in the system
- The combination of the two is, by many researchers, the definition of emergent behavior

Unexpected & Emergent Behavior

- Some argue that the description above is not emergent behavior and that it is only a particular style of robot programming
  - Use of the environment and side-effects leads to the novel behavior
- Their view is that emergent behavior must be truly unexpected, and must come to a surprise to the external observer

Expectation and Emergence

- The problem with unexpected surprise as property of behavior is that:
  - it entirely depends on the expectations of the observer which are completely subjective
  - it depends on the observer's knowledge of the system (informed vs. naïve observer)
  - once observed, the behavior is no longer unexpected

Emergent Behavior and Execution

- Emergent behavior cannot always be designed in advance and is indeed unexpected
- This happens as the system runs, and only at run-time can emergent behavior manifest itself
- The **exact behavior of the system cannot be predicted**
  - Would have to consider all possible sequences and combinations of actions in all possible environments
  - The real world is filled with uncertainty and dynamic properties
- The behaviour controller is **simple** (does not account of all the possible circunstances / effects on the World)
- If we could sense / address the world perfectly, accurate actions and predictions could be made and emergence would not exist!

Desirable/Undesirable Emergent Behavior

- New, unexpected behaviors will always occur in any complex systems interacting with the real world
- Not all behaviors (patterns, or structures) that emerge from the system's dynamics are desirable!
- Example: a robot with simple obstacle avoidance rules can oscillate and get stuck in a corner
- This is also emergent behavior, but regarded as a bug rather than a feature

Sequential and Parallel Execution

- Emergent behavior can arise from interactions of the robot and the environment over time and/or over space
- Time-extended execution of behaviors and interaction with the environment (wall following)
- Parallel execution of multiple behaviors (flocking)
- Given the necessary structure in the environment and enough space and time, numerous emergent behaviors can arise

Architectures and Emergence

- Different architectures have different methods for dealing with emergent behaviors: modularity directly affects emergence
- Reactive systems and behavior-based systems exploit emergent behavior by design
  - Use parallel rules and behaviors which interact with each other and the environment
- Deliberative systems and hybrid systems aim to minimize emergence
  - Sequential, no interactions between components, attempt to produce a uniform output of the system

Hybrid Control

- Idea: get the best of both worlds
- Combine the speed of reactive control and the brains of deliberative control
- Fundamentally different controllers must be made to work together
  - Time scales: short (reactive), long (deliberative)
  - Representations: none (reactive), elaborate world models (deliberative)
- This combination is what makes these systems hybrid

Biological Evidence

- Psychological experiments indicate the existence of two modes of behavior: willed and automatic
- Norman and Shallice (1986) have designed a system consisting of two such modules:
  - Automatic behavior: action execution without awareness or attention, multiple independent parallel activity threads
  - Willed behavior: an interface between deliberate conscious control and the automatic system
- Willed behavior:
  - Planning or decision making, troubleshooting, novel or poorly learned actions, dangerous/difficult actions, overcoming habit or temptation

Hybrid System Components

- Typically, a hybrid system is organized in three layers:
  - A reactive layer
  - A planner
  - A layer that puts the two together
- They are also called three-layer architectures or three-layer systems

The Middle Layer

The middle layer has a difficult job:

- compensate for the limitations of both the planner and the reactive system
- reconcile their different time-scales
- deal with their different representations
- reconcile any contradictory commands between the two
- The main challenge of hybrid systems is to achieve the right compromise between the two layers

An Example

- A robot that has to deliver medication to a patient in a hospital
- Requirements:
  - Reactive: avoid unexpected obstacles, people, objects
  - Deliberative: use a map and plan short paths to destination
- What happens if:
  - The robot needs to deliver medication to a patient, but does not have a plan to his room?
  - The shortest path to its destination becomes blocked?
  - The patient was moved to another room?
  - The robot always goes to the same room?

Bottom-up Communication

Dynamic Re-Planning

- If the reactive layer cannot do its job

  ⇒ It can inform the deliberative layer

- The information about the world is updated
- The deliberative layer will generate a new plan
- The deliberative layer cannot continuously generate new plans and update world information

  ⇒ the input from the reactive layer is a good indication of when to perform such an update

Top-Down Communication

- The deliberative layer provides information to the reactive layer
  - Path to the goal
  - Directions to follow, turns to take
- The deliberative layer may interrupt the reactive layer if better plans have been discovered
- Partial plans can also be used when there is no time to wait for the complete solution
  - Go roughly in the correct direction, plan for the details when getting close to destination

Reusing Plans

- Frequently planned decisions could be reused to avoid re-planning
- These can be stored in an intermediate layer and can be looked up when needed
- Useful when fast reaction is needed
- These mini-plans can be stored as contingency tables
  - intermediate-level actions
  - macro operators: plans compiled into more general operators for future use

Universal Plans

- Assume that we could pre-plan in advance for all possible situations that might come up
- Thus, we could generate and store all possible plans ahead of time
- For each situation a robot will have a pre-existing optimal plan, and will react optimally
- It has a **universal plan**:
  - A set of all possible plans for all initial states and all goals within the robot's state space
- The system is a reactive controller!!

Applicability of Universal Plans

- Examples have been developed as situated automata
- Universal plans are not useful for the majority of real-world domains because:
  - The state space is too large for most realistic problems
  - The world must not change
  - The goals must not change
- Disadvantages of pre-compiled systems
  - Are not flexible in the presence of changing environments, tasks or goals
  - It is prohibitively large to enumerate the state space of a real robot, and thus pre-compiling generally does not scale up to complex systems

Reaction - Deliberation Coordination

- **Selection:**

  Planning is viewed as configuration

- **Advising:**

  Planning is viewed as advice giving

- **Adaptation:**

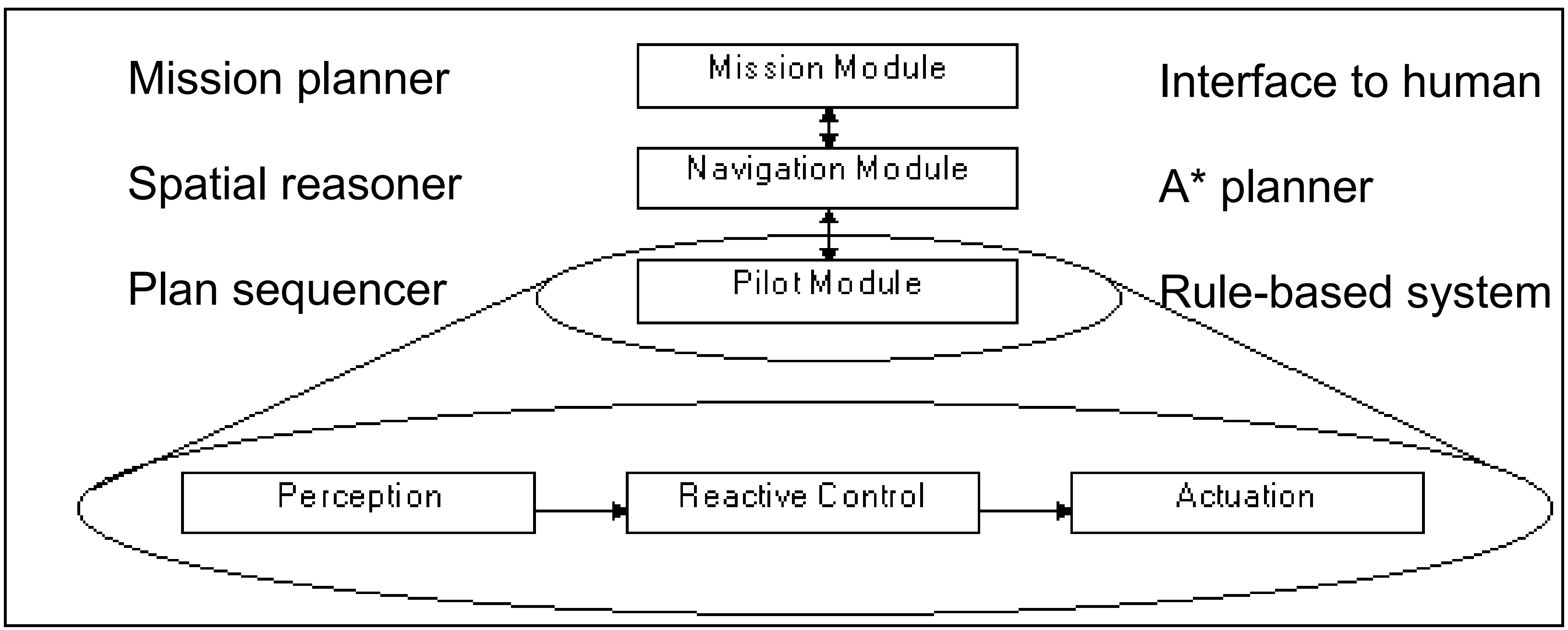
  Planning is viewed as adaptation

- **Postponing:**

  Planning is viewed as a least commitment process

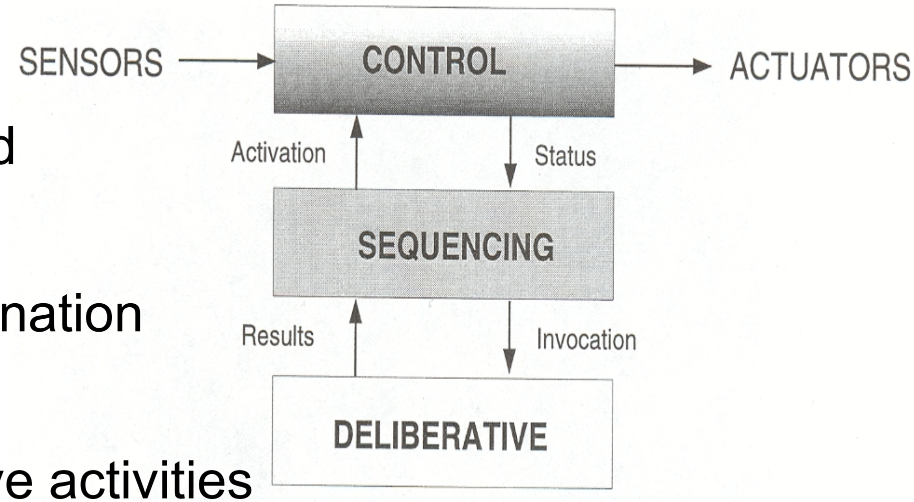
Selection Example: AuRA

- Autonomous Robot Architecture (R. Arkin, ’86)
  - A deliberative hierarchical planner and a reactive controller based on schema theory



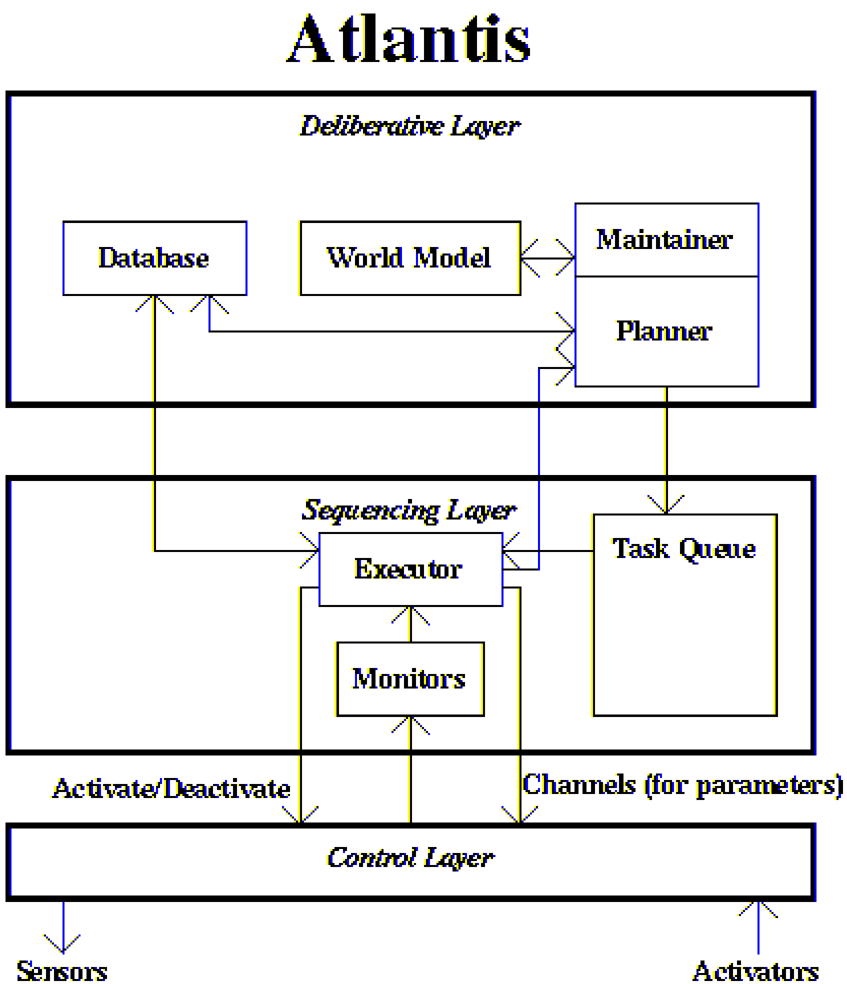
Advising Example: Atlantis

- E. Gat, Jet Propulsion Laboratory (1991)
- Three layers:
  - Deliberator: planning and world modeling
  - Sequencer: initiation and termination of low-level activities
  - Controller: collection of primitive activities

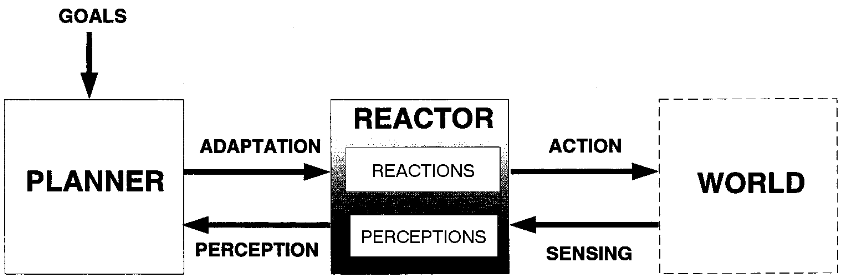


- Asynchronous, heterogeneous architecture
- Controller implemented in ALFA (A Language for Action)
- Introduces the notion of cognizant failure
- Planning results view as advice, not decree
- Tested on NASA rovers

Atlantis Schematic

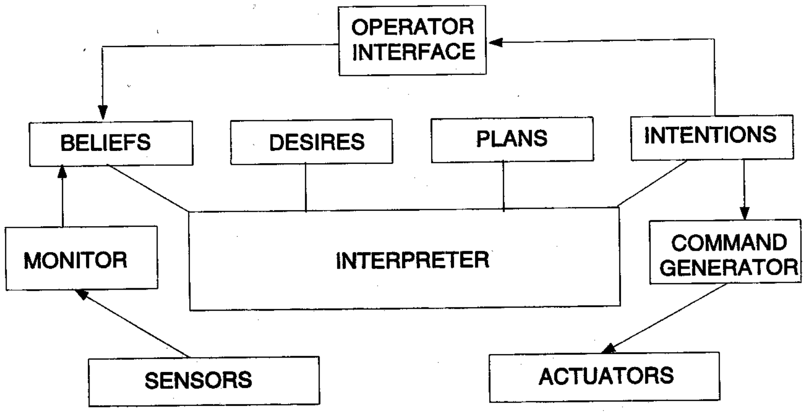


Adaptation Example: Planner-Reactor



- D. Lyons (1992)
- The planner continuously modifies the reactive control system
- Planning is a form of reactor adaptation
  - Monitor execution, adapts control system based on environment changes and changes of the robot's goals
- Adaptation is on-line rather than off-line deliberation
- Planning is used to remove performance errors when they occur and improve plan quality
- Tested in assembly and grasp planning

Postponing Example: PRS



- Procedural Reasoning System, Georgeff and A. Lansky (1987)
- Reactivity refers to postponement of planning until it is necessary
- Information necessary to make a decision is assumed to become available later in the process
- Plans are determined in reaction to current situation
- Previous plans can be interrupted and abandoned at any time
- Tested on SRI Flakey

Postponing Example: SSS

- Servo Subsumption Symbolic, J. Connell (1992)
- 3 layers: servo, subsumption, symbolic
- World models are viewed as a convenience, not a necessity
- The symbolic layer selectively turns behaviors on/off and handles strategic decisions (where-to-go-next)
- The subsumption layer handles tactical decisions (where-to-go-now)
- The servo layer deals with making the robot go (continuous time)
- Tested on TJ

Other Examples

- Multi-valued logic
  - Saffiotti, Konolige, Ruspini (SRI)
  - Variable planner-controller interface, strongly dependent on the context
- SOMASS hybrid assembly system
  - C. Malcolm and T. Smithers (Edinburgh U.)
  - Cognitive/subcognitive components
  - Cognitive component designed to be as ignorant as possible
  - Planning as configuration

Other Examples

- Agent architecture
  - B. Hayes-Roth (Stanford)
  - 2 levels: physical and cognitive
  - Claim: reactive and deliberative behaviors can exist at each level $\Rightarrow$ blurry functional boundary
  - Difference consists in: time-scale, symbolic/metric representation, level of abstraction
- Theo-Agent
  - T. Mitchell (CMU, 1990)
  - Reacts when it can plans when it must
  - Emphasis on learning: how to become more reactive?

More Examples

- Generic Robot Architecture
  - Noreils and Chatila (1995, France)
  - 3 levels: planning, control system, functional
  - Formal method for designing and interfacing modules (task description language)
- Dynamical Systems Approach
  - Schoner and Dose (1992)
  - Influenced by biological systems
  - Planning is selecting and parameterizing behavioral fields
  - Behaviors use vector summation

More Examples

- Supervenience architecture
  - L. Spector (1992, U. of Maryland)
  - Integration based on “distance from the world”
  - Multiple levels of abstraction: perceptual, spatial, temporal, causal
- Teleo-reactive agent architecture
  - Benson and N. Nilsson (1995, Stanford)
  - Plans are built as sets of teleoreactive (TR) operators
  - Arbitrator selects operator for execution
  - Unifying representation for reasoning and reaction

More Examples

- Reactive Deliberation
  - M. Sahota (1993, U. of British Columbia)
  - Reactive executor: consists of action schemas
  - Deliberator: enables one schema at a time and provides parameter values $\Rightarrow$ action selection
  - Robosoccer
- Integrated path planning and dynamic steering control
  - Krogh and C. Thorpe (1986, CMU)
  - Relaxation over grid-based model with potential fields controller
  - Planner generated waypoints for controller
- Many others (including several for UUVs)

BBS vs. Hybrid Control

- Both BBS and Hybrid control have the same expressive and computational capabilities
  - Both can *store representations* and *look ahead*
- BBS and Hybrid Control have different niches in the set of application domains
  - BBS: multi-robot domains, hybrid systems: single-robot domain
- Hybrid systems:
  - Environments and tasks where internal models and planning can be employed, and real-time demands are few
- Behavior-based systems:
  - Environments with significant dynamic changes, where looking ahead would be required

Learning & Adaptive Behavior

- Learning produces changes within an agent that over time enable it to perform more effectively within its environment

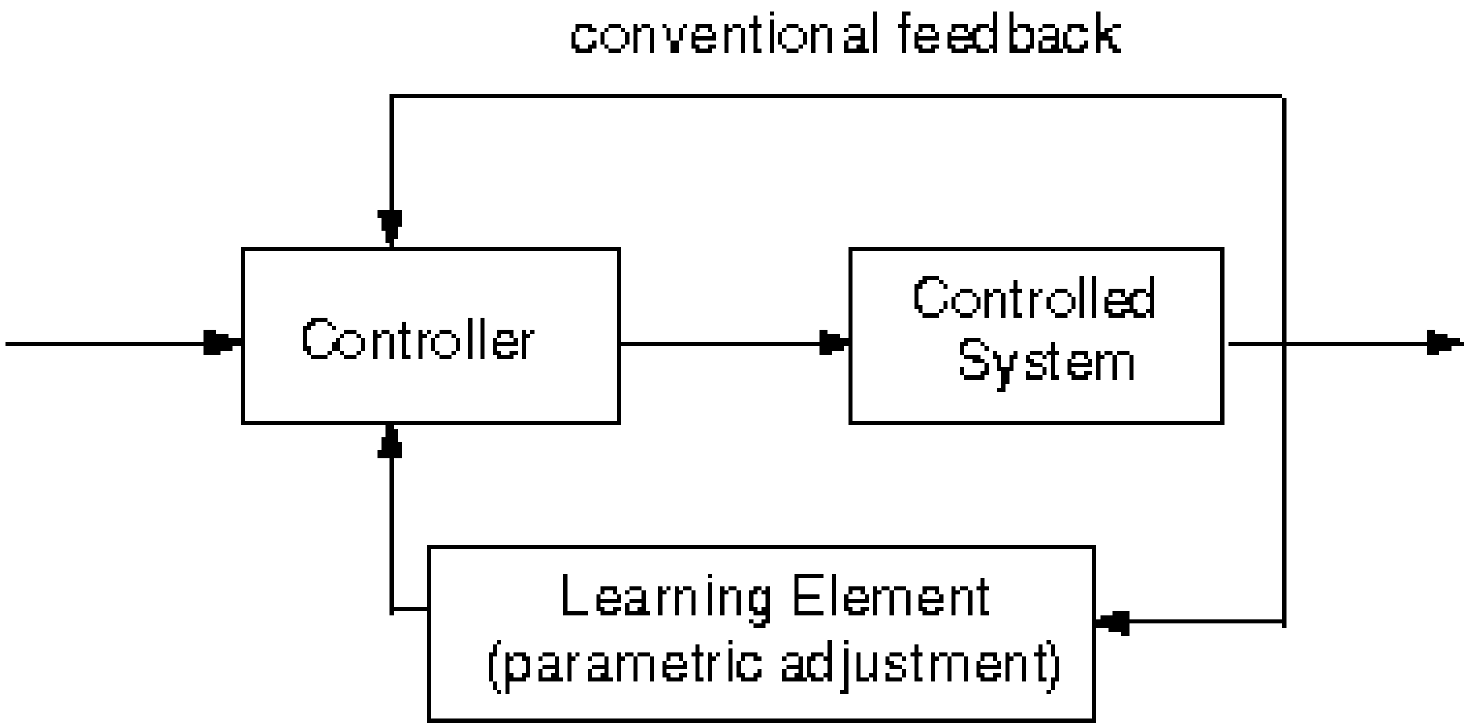
- Adaptation refers to an agent's learning by making adjustments in order to be more attuned to its environment
  - Phenotypic (within an individual agent) or genotypic (evolutionary)
  - Acclimatization (slow) or homeostasis (rapid)

Types of Adaptation

- Behavioral adaptation
  - Behaviors are adjusted relative to each other
- Evolutionary adaptation
  - Descendants change over long time scales based on ancestor's performance
- Sensory adaptation
  - Perceptual system becomes more attuned to the environment
- Learning as adaptation
  - Anything else that results in a more ecologically fit agent

Adaptive Control

- Astrom 1995
  - Feedback is used to adjust controller's internal parameters



Learning

Learning can improve performance in additional ways:

- Introduce new knowledge (facts, behaviors, rules)
- Generalize concepts
- Specialize concepts for specific situations
- Reorganize information
- Create or discover new concepts
- Create explanations
- Reuse past experiences

At What Level Can Learning Occur?

- Within a behavior
  - Suitable stimulus for a particular response
  - Suitable response for a given stimulus
  - Suitable behavioral mapping between stimulus and responses
  - Magnitude of response
  - Whole new behaviors
- Within a behavior assemblage
  - Component behavior set
  - Relative strengths
  - Suitable coordination function

Challenges of Learning Systems

- Credit assignment
  – How is credit/blame assigned to the components for the success or failure of the task?
- Saliency problem
  – What features are relevant to the learning task?
- New term problem
  – When to create a new concept/representation?
- Indexing problem
  – How can memory be efficiently organized?
- Utility problem
  – When/what to forget?

Classification of Learning Methods

Tan 1991

- Numeric vs. symbolic
  - Numeric: manipulate numeric quantities (neural networks)
  - Symbolic: manipulate symbolic representations
- Inductive vs. deductive
  - Inductive: generalize from examples
  - Deductive: produce a result from initial knowledge
- Continuous vs. batch
  - Continuous: during the robot's performance in the world
  - Batch: from a large body of accumulated experience

Learning Methods

- Reinforcement learning
- Neural network (connectionist) learning
- Evolutionary learning
- Learning from experience
  - Memory-based
  - Case-based
- Learning from demonstration
- Inductive learning
- Explanation-based learning
- Multistrategy learning

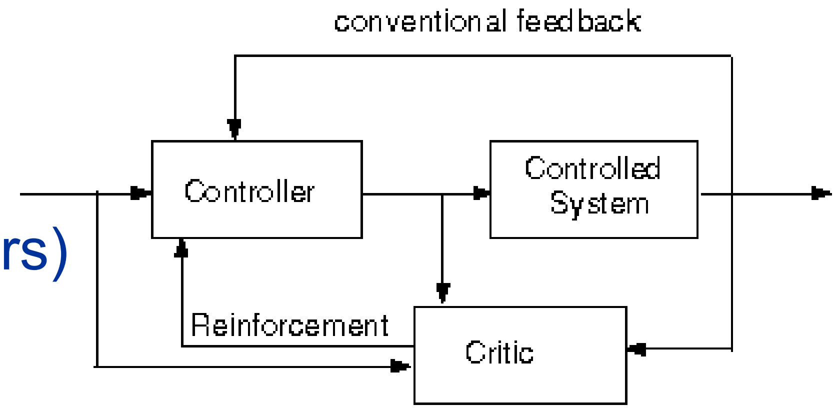
Reinforcement Learning (RL)

- Motivated by psychology (the Law of Effect, Thorndike 1991):

  Applying a reward immediately after the occurrence of a response increases its probability of reoccurring, while providing punishment after the response will decrease the probability

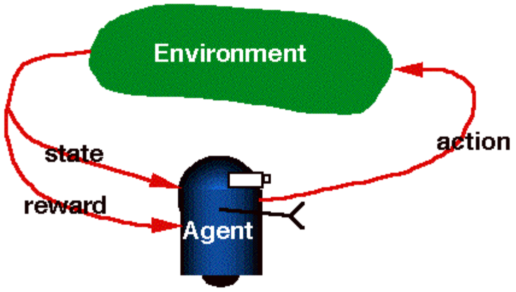
- One of the most widely used methods for adaptation in robotics

Reinforcement Learning



- Combinations of stimuli (i.e., sensory readings and/or state) and responses (i.e., actions/behaviors) are given positive/negative reward in order to increase/decrease their probability of future use
- Desirable outcomes are strengthened and undesirable outcomes are weakened
- *Critic*: evaluates the system's response and applies reinforcement
  - external: the user provides the reinforcement
  - internal: the system itself provides the reinforcement (reward function)

Decision Policy



- The robot can observe the *state* of the environment
- The robot has a set of *actions* it can perform
  - Policy: state/action mapping that determines which actions to take
- Reinforcement is applied based on the results of the actions taken
  - Utility: the function that gives a utility value to each state
- Goal: learn an optimal policy that chooses the best action for every set of possible inputs

Unsupervised Learning

- RL is an unsupervised learning method:
  - No target goal state
- Feedback only provides information on the quality of the system's response
  - Simple: binary fail/pass
  - Complex: numerical evaluation
- Through RL a robot learns on its own, using its own experiences and the feedback received
- The robot is never told what to do

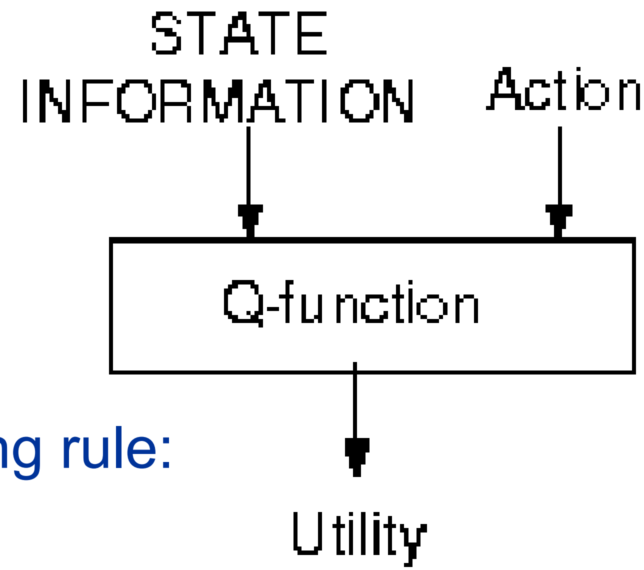
Challenges of RL

- Credit assignment problem:
  - When something good or bad happens, what exact state/condition-action/behavior should be rewarded or punished?
- Learning from delayed rewards:
  - It may take a long sequence of actions that receive insignificant reinforcement to finally arrive at a state with high reinforcement
  - How can the robot learn from reward received at some time in the future?

Challenges of RL

- Exploration vs. exploitation:
  - Explore unknown states/actions or exploit states/actions already known to yield high rewards
- Partially observable states
  - In practice, sensors provide only partial information about the state
  - Choose actions that improve observability of environment
- Life-long learning
  - In many situations it may be required that robots learn several tasks within the same environment

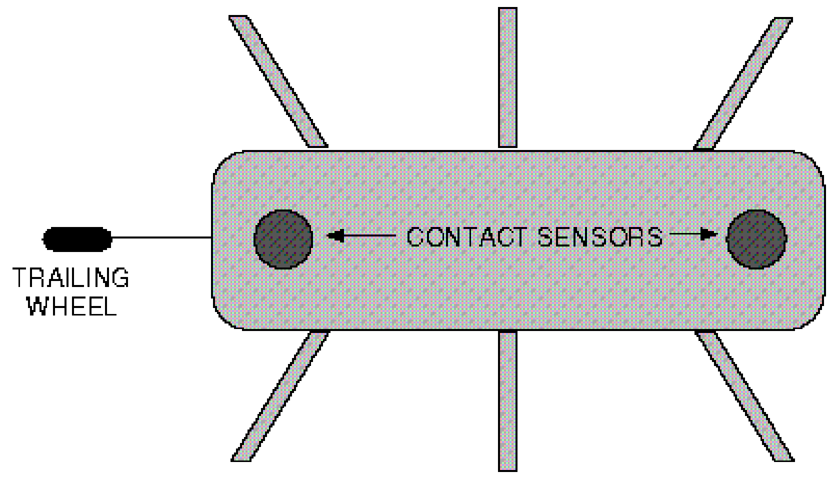
Q-Learning



- Watkins 1980's
- A single utility Q-function is learned to evaluate both actions and states
- Q values are stored in a table
- Updated at each step, using the following rule:
  $Q(x,a) \leftarrow Q(x,a) + \beta\,(r + \lambda E(y) - Q(x,a))$
- x: state; a: action; β: learning rate; r: reward; λ: discount factor (0,1);
- E(y) is the utility of the state y: $E(y) = \max(Q(y,a))\ \forall$ actions a
- Guaranteed to converge to optimal solution, given infinite trials

Learning to Walk

- Maes, Brooks (1990)
- Genghis: hexapod robot
- Learned stable tripod stance and tripod gait
- Rule-based subsumption controller



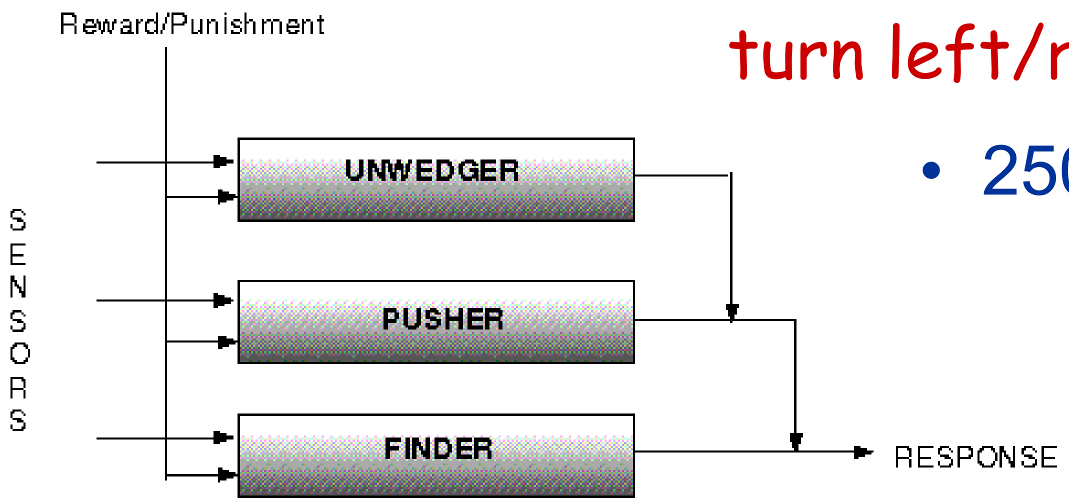
- Two sensor modalities for feedback:
  – Two touch sensors to detect hitting the floor: - feedback
  – Trailing wheel to measure progress: + feedback

Learning to Walk

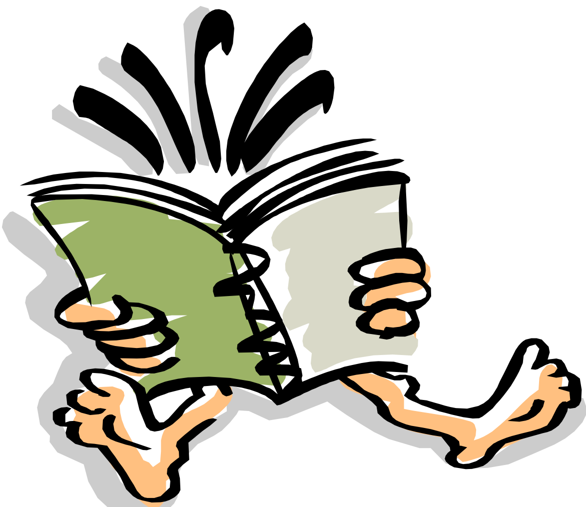
- Nate Kohl & Peter Stone (2004)

Learning to Push

- Mahadevan & Connell 1991
- Obelix: 8 ultrasonic sensors, 1 IR, motor current
- Learned how to push a box (Q-learning)
- Motor outputs grouped into 5 choices: move forward, turn left or right (22 degrees), sharp turn left/right (45 degrees)
  - 250,000 states



Readings



- M. Matarić: Chapters 17, 18

The Robotics Primer,

From Intelligent Robotics and Autonomous Agents series

MIT Press