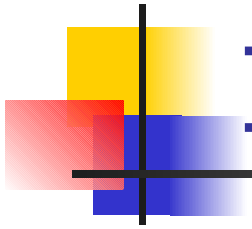
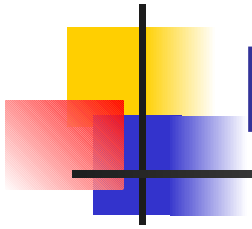


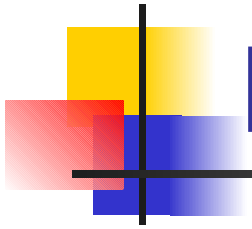
Intelligent Agents

- Agent: anything that can be viewed as…
  - perceiving its environment through sensors
  - acting upon its environment through actuators
- Examples:
  - Human
  - Web search agent
  - Chess player
- What are sensors and actuators for each of these?



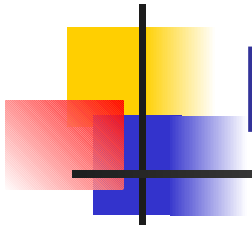
Rational Agents

- Conceptually: one that does the right thing
- Criteria: Performance measure
- Performance measures for
  - Web search engine?
  - Tic-tac-toe player? Chess player?
- When performance is measured plays a role
  - short vs. long term



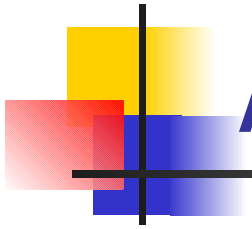
Rational Agents

- Omniscient agent
  - Knows actual outcome of its actions
  - What info would chess player need to be omniscient?
- Omniscience is (generally) impossible
  - Rational agent should do right thing based on knowledge it has



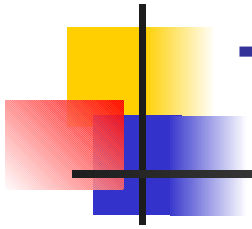
Rational Agents

- What is rational depends on four things:
  - Performance measure
  - Percept sequence: everything agent has seen so far
  - Knowledge agent has about environment
  - Actions agent is capable of performing
- Rational Agent definition:
  - Does whatever action is expected to maximize its performance measure, based on percept sequence and built-in knowledge



Autonomy

- "Independence"
- A system is autonomous if its behavior is determined by its percepts
  - An alarm that goes off at a prespecified time is not autonomous
  - An alarm that goes off when smoke is sensed is autonomous
- A system without autonomy lacks flexibility



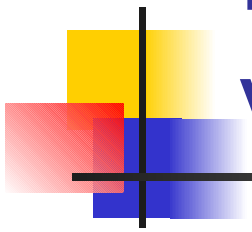
The Task Environment

- An agent's rationality depends on
  - Performance Measure
  - Environment
  - Actuators
  - Sensors

What are each of these for:

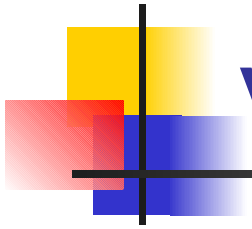
- Chess Player?
- Web Search Tool?
- Matchmaker?
- Musical performer?

Environments: Fully Observable vs. Partially Observable



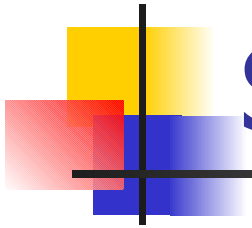
- Fully observable: agent's sensors detect all aspects of environment relevant to deciding action
- Examples?
- Which is more desirable?

Environments: Determinstic vs. Stochastic



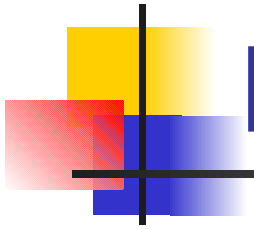
- Deterministic: next state of environment is completely determined by current state and agent actions
- Stochastic: uncertainty as to next state
- If environment is partially observable but deterministic, may appear stochastic
- If environment is determinstic except for actions of other agents, called strategic
- Agent's point of view is the important one
- Examples?
- Which is more desirable?

Environments: Episodic vs. Sequential



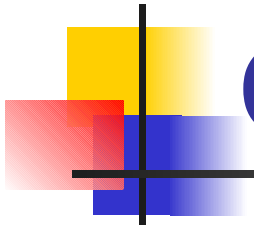
- Episodic: Experience is divided into "episodes" of agent perceiving then acting. Action taken in one episode does not affect next one at all.
- Sequential typically means need to do lookahead
- Examples?
- Which is more desirable?

Environments: Static vs. Dynamic



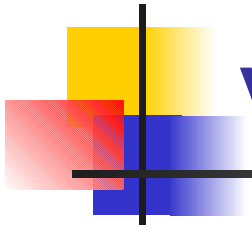
- Dynamic: Environment can change while agent is thinking
- Static: Environment does not change while agent thinks
- Semidynamic: Environment does not change with time, but performance score does
- Examples?
- Which is more desirable?

Environments: Discrete vs. Continuous



- Discrete: Percepts and actions are distinct, clearly defined, and often limited in number
- Examples?
- Which is more desirable?

Environments: Single agent vs. multiagent



- What is distinction between environment and another agent?
  - for something to be another agent, maximize a performance measure depending on your behavior
- Examples?

Structure of Intelligent Agents

- What does an agent program look like?
- Some extra Lisp: Persistence of state (static variables)
- Allows a function to keep track of a variable over repeated calls.
  - Put functions inside a let block
  - ```
    (let ((sum 0))
      (defun myfun (x)
        (setf sum (+ sum x)))
      (defun report ()
         sum)
    )
    ```
    ```

Generic Lisp Code for an Agent

```
(let ((memory nil))
  (defun skeleton-agent (percept)
    (setf memory
      (update-memory memory percept))
    (setf action
      (choose-best-action memory))
    (setf memory
      (update-memory memory action))
    action  ; return action
  ))
```

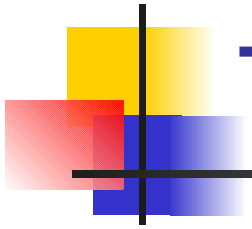


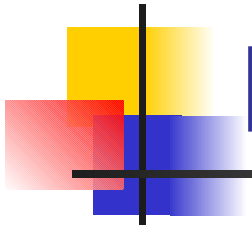
Table Lookup Agent

- In theory, can build a table mapping percept sequence to action
- Inputs: percept
- Outputs: action
- Static Variable: percepts, table

Lookup Table Agent

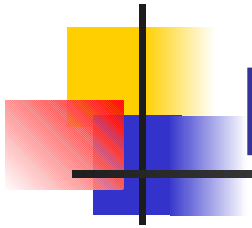
```
(let ((percepts nil) (table ????)
  (defun table-lookup-agent (percept)
    (setf percepts
      (append (list percept) percepts))
    (lookup percepts table))
))
```

Specific Agent Example: Pathfinder (Mars Explorer)

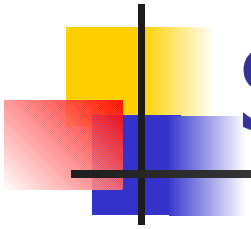


- Performance Measure:
- Environment:
- Actuators:
- Sensors:
- Would table-driven work?

Four kinds of better agent programs

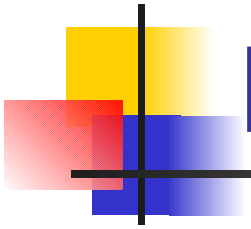


- Simple reflex agents
- Model-based reflex agents
- Goal-based agents
- Utility-based agents



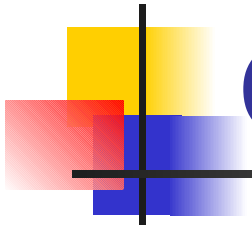
Simple reflex agents

- Specific response to percepts, i.e. condition-action rule
- if new-boulder-in-sight then
  move-towards-new-boulder
- Advantages:
- Disadvantages:



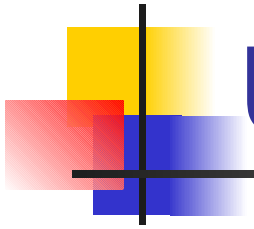
Model-based reflex agents

- Maintain an internal state which is adjusted by each percept
  - Internal state: looking for a new boulder, or rolling towards one
  - Affects how Pathfinder will react when seeing a new boulder
- Can be used to handle partial observability by use of a model about the world
- Rule for action depends on both state and percept
- Different from reflex, which only depends on percept



Goal-Based Agents

- Agent continues to receive percepts and maintain state
- Agent also has a goal
  - Makes decisions based on achieving goal
- Example
  - Pathfinder goal: reach a boulder
  - If pathfinder trips or gets stuck, can make decisions to reach goal



Utility-Based Agents

- Goals are not enough – need to know value of goal
  - Is this a minor accomplishment, or a major one?
  - Affects decision making – will take greater risks for more major goals
- Utility: numerical measurement of importance of a goal
- A utility-based agent will attempt to make the appropriate tradeoff