COMS W4701: Artificial Intelligence

Lecture 2: Intelligent Agents

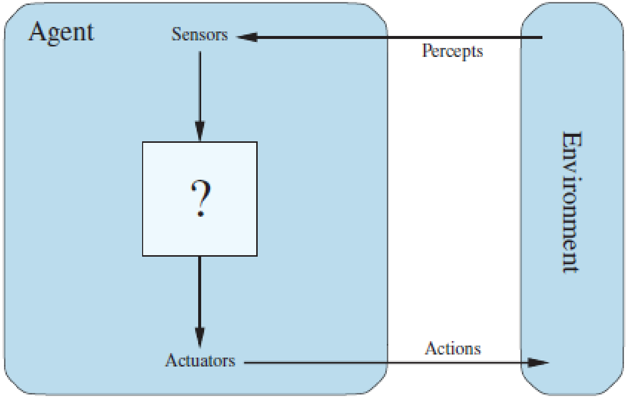
Tony Dear, Ph.D.
Department of Computer Science
School of Engineering and Applied Sciences

Today

- Rational agents
- Task environments
- Agent programs

Agents and Environments

- An **agent** perceives its environment through *sensors* and acts upon its environment through *actuators*
- Agents may be *controlled* or *autonomous*



- **Agent functions** map percepts to actions
- Other dependencies: Prior knowledge, past experience, goals, preferences, capabilities

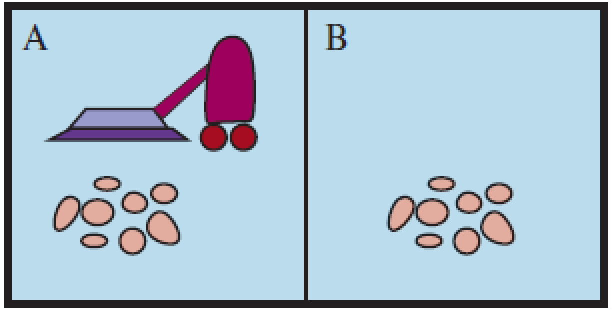
- Environment may be arbitrarily general, but in practice limited to aspects that directly interact with the agent

Examples of Agents

- Humans are agents!
  - Sensors: Vision, hearing, touch, smell, taste, proprioception
  - Actuators: Muscles, reflexes, changing brain state

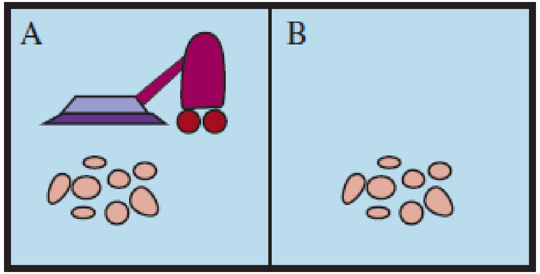
- AI is more interested in “artificial” agents in complex environments that require nontrivial decision making

- Example: Vacuum cleaner (think Roomba)
- Environment: Square A and square B



Example: Vacuum Cleaner World

- Percepts: Current square; is the square dirty?
- Actions: Move left, move right, clean, do nothing



- Agent function:

[A, IsClean]	Move right
[A, IsDirty]	Clean
[B, IsClean]	Move left
[B, IsDirty]	Clean
[[A, IsDirty], [A, IsClean]]	Move right
[[B, IsDirty], [A, IsClean]]	Move right
[[B, IsClean], [A, IsClean]]	Do nothing
……	……

Rational Agents

- Environment state sequence can be evaluated by a **performance measure**
- Performance measures usually based on desired *outcomes*, not *behaviors*

- A **rational agent** selects an action to *maximize* its performance measure given percept sequence and in-built knowledge.
  - What performance measure makes our vacuum cleaner rational or irrational?

- Rational agents maximize *expected* performance, are not omniscient
- Rationality may involve info gathering, exploration, learning

Task Environments

- A rational agent is a *solution* to a **task environment** problem
- **PEAS**: Performance measure, environment, actuators, sensors

- Vacuum cleaner task environment
  - P: Cleanliness, power usage, time taken
  - E: The small grid world
  - A: Wheels to move, filter to clean
  - S: “GPS”, cleanliness sensor

Task Environment Properties

- Fully observable vs partially observable vs unobservable
  - Can agent sense all *relevant* information? Is internal state (memory) required?
- Single-agent vs multi-agent
  - Does maximization of performance measure depend on other agents' behaviors?
- Deterministic vs stochastic
  - Can we completely predict the next state of the environment?
- Episodic vs sequential
  - Do current decisions depend on past ones? Do current decisions affect future ones?
- Static vs dynamic
  - Does the environment change while the agent is thinking?
- Discrete vs continuous
  - Is number of states, actions, percepts, time, etc. finite?

Examples of Environments

Environment	Partially / Fully Observable	Single- / Multi-Agent	Deterministic / Stochastic	Sequential / Episodic	Dynamic / Static	Continuous / Discrete
Vacuum cleaner world	Depends	Single	Deterministic	Sequential	Static	Discrete
Chess	Fully	Multi (adversarial)	Deterministic	Sequential	Static	Discrete
Self-driving car	Partially	Multi (cooperative)	Stochastic	Sequential	Dynamic	Continuous
Image classification	Fully	Single	Deterministic	Episodic	Static	Depends

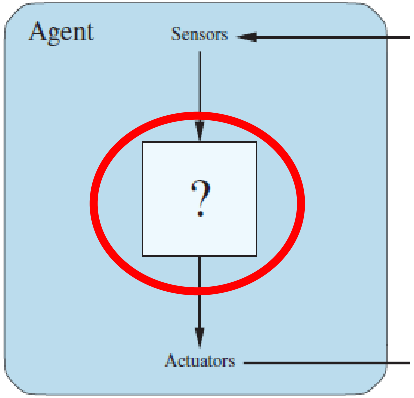
Agent Design

- Understanding the task environment tells us how to design our agents
- The more difficult the task environment, the more complex the agent

- Partially observable env -> agent requires memory / state
- Multi-agent env -> agent requires tracking of other agents
- Stochastic env -> agent must consider multiple scenarios or contingencies
- Sequential env -> agent must consider past and future states
- Dynamic env -> agent must maintain model of the world

Agent Programs

- An **agent program** is an *implementation* of an agent function
- Specifies *how* something is computed rather than *what* needs to be computed



- A given agent program is not a universal solution!
- Program usefulness depends on hardware, tractability

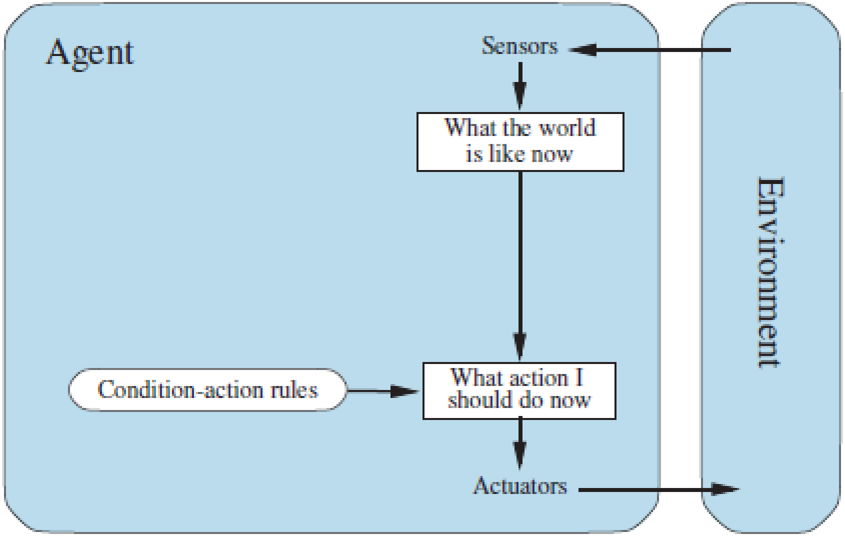
- Agent programs also depend on the desired *solutions*
- E.g., *optimal* wrt some utility, approximately optimal, *satisficing* ("good enough"), *probable*

Simple Reflex Agents

- Simple reflex agent: Use current percept only
- Percepts may be mapped to internal *states*
- Match state to condition-action (if-then) rules

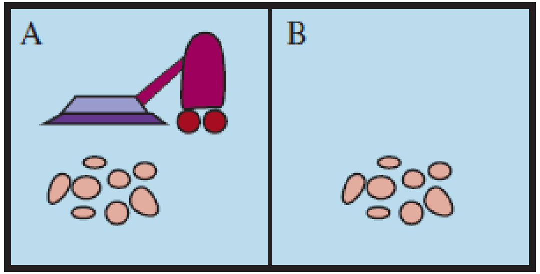
function SIMPLE-REFLEX-AGENT(*percept*) **returns** an action
  persistent: *rules*, a set of condition–action rules

  state ← INTERPRET-INPUT(*percept*)
  rule ← RULE-MATCH(*state*, *rules*)
  action ← *rule*.ACTION
  return *action*



Example: Vacuum Cleaner Robot

- Example: Vacuum cleaner robot as a reflex agent
- Use only the *current* (no past) percept
- What is its resultant behavior?



function Reflex-Vacuum-Agent([location, status]) **returns** action

  if status **is** dirty **then return** *clean*

  if location **is** A **then return** *move right*

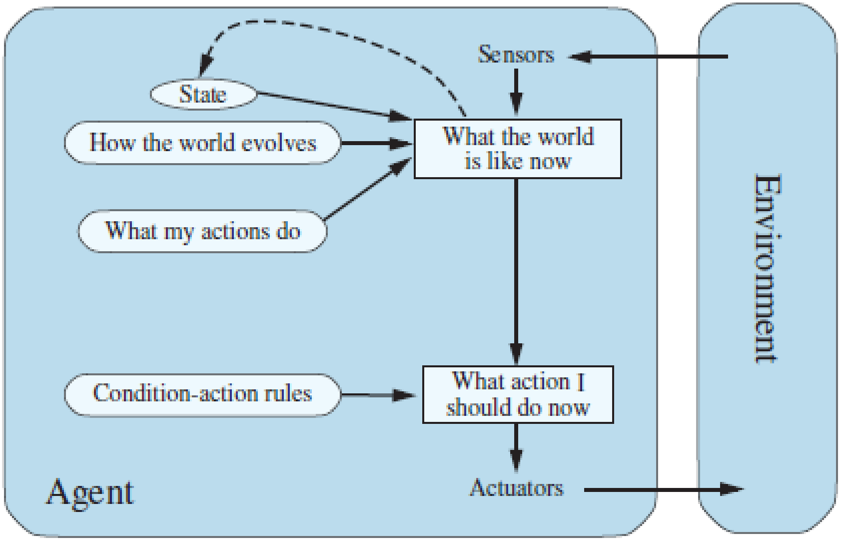
  if location **is** B **then return** *move left*

  return *do nothing*

Model-Based Reflex Agents

- What about partially observable environments?

- Maintain **internal state** and **transition model**
- May also have *sensor model* mapping percepts
- Use all information to *update* the state




```
function MODEL-BASED-REFLEX-AGENT(percept) returns an action
  persistent: state, the agent's current conception of the world state
              model, a description of how the next state depends on current state and action
              rules, a set of condition–action rules
              action, the most recent action, initially none

  state ← UPDATE-STATE(state, action, percept, model)
  rule ← RULE-MATCH(state, rules)
  action ← rule.ACTION
  return action
```

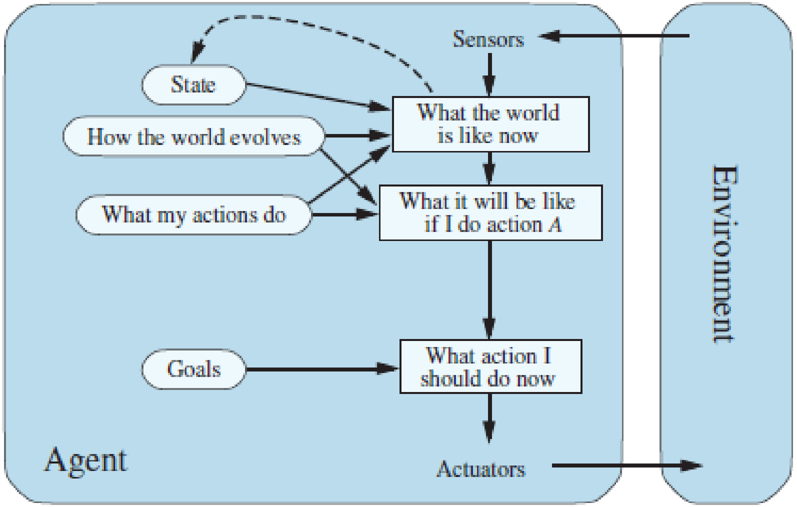
Goal- and Utility-Based Agents

- Reflex agents are very rigid and predictable

- **Goal-based** agents try to achieve particular states
- Problems often solved using search and planning

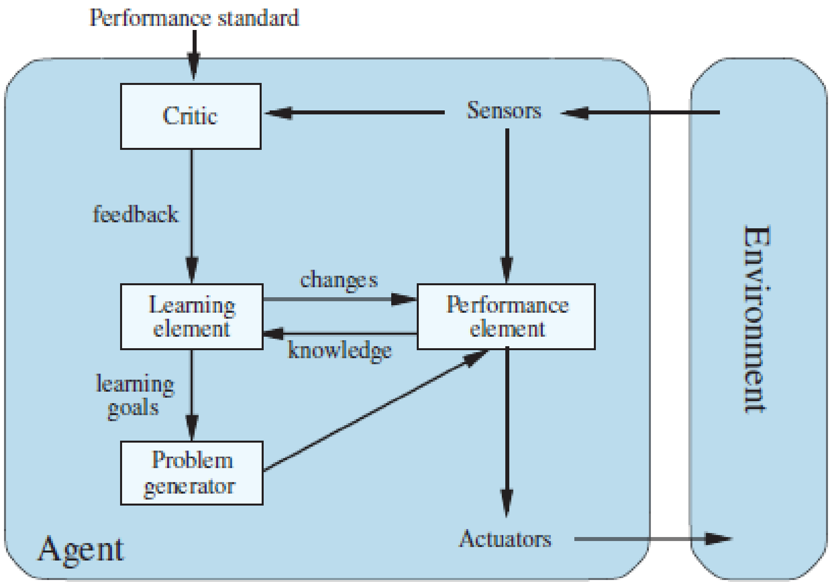
- **Utility-based** agents can compare different states
- Utility functions map state to "desirability"
- Internalize the overall performance measure

- Utilities specify tradeoffs for competing goals
- Also useful in face of uncertainty



Learning Agents

- **Learning agents** can be used to *create* or *improve* upon initial models in unknown envs
- A *learning element* retrieves knowledge from and then improves the *performance element*
- A *critic* evaluates the learning element according to a *performance standard* (measure)
- A *problem generator* suggests actions that can help gather new information and experiences



Summary

- Agents interact with their environments through sensors and actuators
- Rational agents maximize expected performance measure

- PEAS descriptors define task environments and influence agent design
- Environment properties vary from easy to extremely challenging

- Agent programs may use current percept only, use a model, and/or try to achieve certain goals quantified by utilities
- Agent programs may also be created or improved via learning