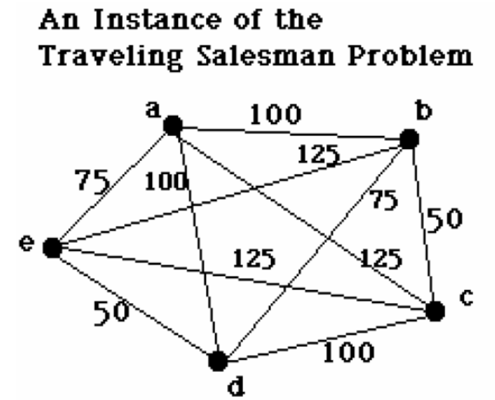
Project 2: Travelling Salesperson Problem – **Search (Ch. 3)**

- Learning objectives.
  - Search Techniques for graphs
  - BFS and DFS algorithms

Problem



- For this lab we are looking at a special case of TSP in which not all cities are connected and the salesperson only needs to find the best path to a target city not visit all cities.
  - For the given dataset (11PointDFSBFS.tsp), starting at the first city (city 1) find the shortest path to the goal city (city 11).
  - Implement Breadth First Search (BFS) and Depth First Search (DFS) algorithms
  - Visit cities in numerical order if you need to break a tie. You can hardcode connected edges into your algorithm for this problem

Data for Project 2

pt	1	2	3	4	5	6	7	8	9	10	11
1		x	x	x							
2			x								
3				x	x						
4					x	x	x				
5							x	x			
6								x			
7									x	x	
8									x	x	x
9											x
10											x

Deliverables

- Project report (3-4 pages) describing results of your experiments and your implementation. Which algorithm was faster in finding an acceptable solution? How long did it take?

- Well-commented source code for your project. You can use any language you like, but I reserve the right to ask you to demo performance of your algorithm on a new dataset.

- You don’t have to include a GUI with visual representation of the solutions for this project, but it might be useful for your future TSP related projects in this course.

Practice Exercises from Chapter 2

- Work on the following problems in your book:
- You can work in teams of 3 or 4
- 2.3, 2.5, 2.6(a-d)
- Submit your solutions on Blackboard

Exercise 2.3

2.3 For each of the following assertions, say whether it is true or false and support your answer with examples or counterexamples where appropriate.

a. An agent that senses only partial information about the state cannot be perfectly rational.

b. There exist task environments in which no pure reflex agent can behave rationally.

c. There exists a task environment in which every agent is rational.

d. The input to an agent program is the same as the input to the agent function.

e. Every agent function is implementable by some program/machine combination.

f. Suppose an agent selects its action uniformly at random from the set of possible actions. There exists a deterministic task environment in which this agent is rational.

g. It is possible for a given agent to be perfectly rational in two distinct task environments.

h. Every agent is rational in an unobservable environment.

i. A perfectly rational poker-playing agent never loses.

Exercise 2.5

2.5 Define in your own words the following terms: agent, agent function, agent program, rationality, autonomy, reflex agent, model-based agent, goal-based agent, utility-based agent, learning agent.

Exercise 2.6 (a-d)

2.6 This exercise explores the differences between agent functions and agent programs.

a. Can there be more than one agent program that implements a given agent function? Give an example, or show why one is not possible.

b. Are there agent functions that cannot be implemented by any agent program?

c. Given a fixed machine architecture, does each agent program implement exactly one agent function?

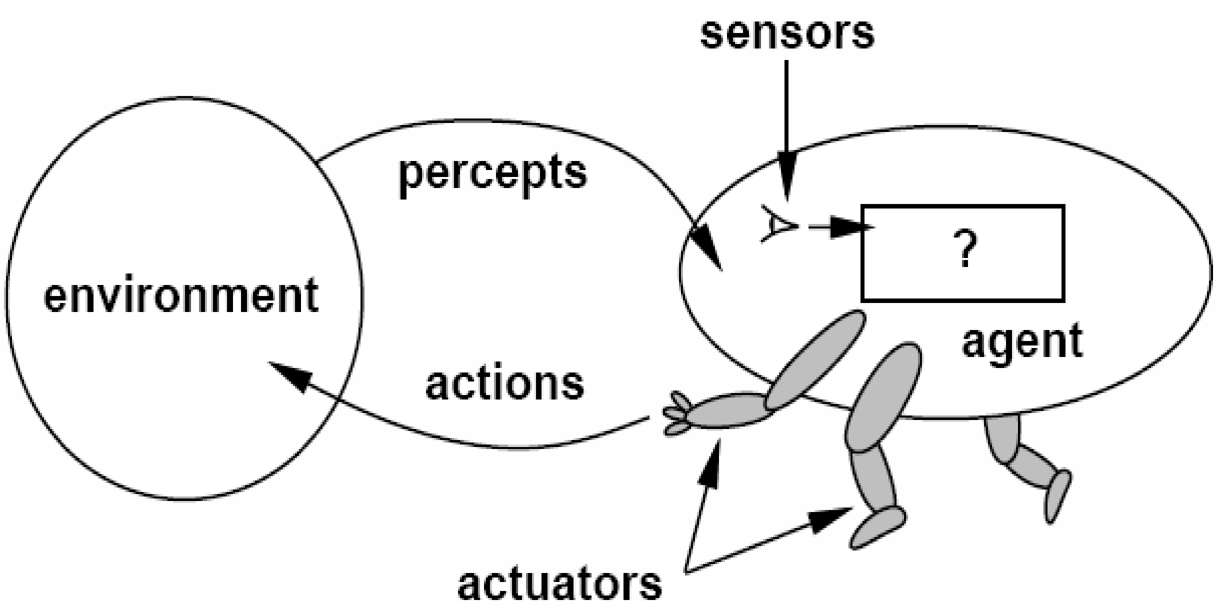
d. Given an architecture with n bits of storage, how many different possible agent programs are there?

CECS545-Artificial Intelligence

Agents 2

Dr. Roman Yampolskiy

Agents and environments



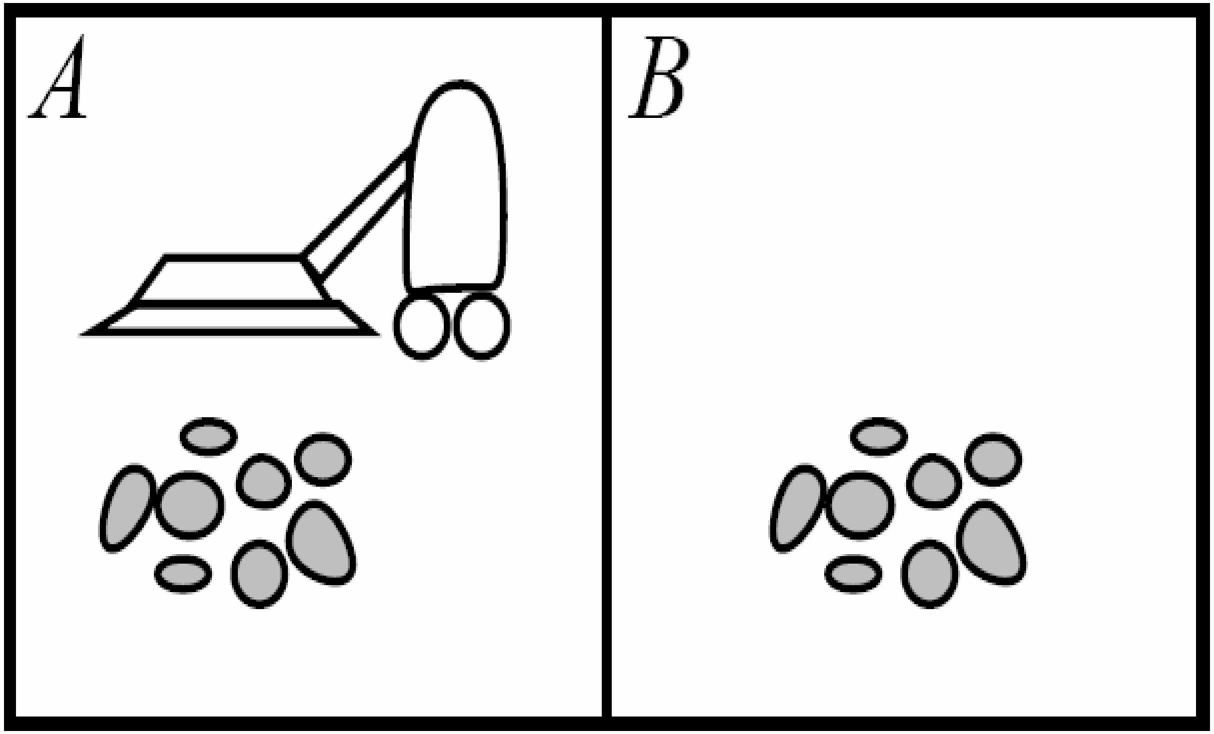
Agents include humans, robots, softbots, thermostats, etc.

The agent function maps from percept histories to actions:

$$f : \mathcal{P}^* \to \mathcal{A}$$

The agent program runs on the physical architecture to produce f

Vacuum-cleaner world



Percepts: location and contents, e.g., $[A, Dirty]$

Actions: $Left$, $Right$, $Suck$, $NoOp$

A vacuum-cleaner agent

Percept sequence	Action
$[A, Clean]$	$Right$
$[A, Dirty]$	$Suck$
$[B, Clean]$	$Left$
$[B, Dirty]$	$Suck$
$[A, Clean]$, $[A, Clean]$	$Right$
$[A, Clean]$, $[A, Dirty]$	$Suck$
$\vdots$	$\vdots$

function REFLEX-VACUUM-AGENT([*location*,*status*]) **returns** an action

  if *status* = *Dirty* **then return** *Suck*
  else if *location* = *A* **then return** *Right*
  else if *location* = *B* **then return** *Left*

Rationality

Fixed performance measure evaluates the environment sequence

- one point per square cleaned up in time T?
- one point per clean square per time step, minus one per move?
- penalize for $> k$ dirty squares?

A rational agent chooses whichever action maximizes the expected value of the performance measure given the percept sequence to date

Rational $\neq$ omniscient

- percepts may not supply all relevant information

Rational $\neq$ clairvoyant

- action outcomes may not be as expected

Hence, rational $\neq$ successful

Rational $\Rightarrow$ exploration, learning, autonomy

PEAS

To design a rational agent, we must specify the task environment

Consider, e.g., the task of designing an automated taxi:

Performance measure?? safety, destination, profits, legality, comfort, ...

Environment?? US streets/freeways, traffic, pedestrians, weather, ...

Actuators?? steering, accelerator, brake, horn, speaker/display, ...

Sensors?? video, accelerometers, gauges, engine sensors, keyboard, GPS, ...

Internet shopping agent

Performance measure?? price, quality, appropriateness, efficiency

Environment?? current and future WWW sites, vendors, shippers

Actuators?? display to user, follow URL, fill in form

Sensors?? HTML pages (text, graphics, scripts)

Environment types

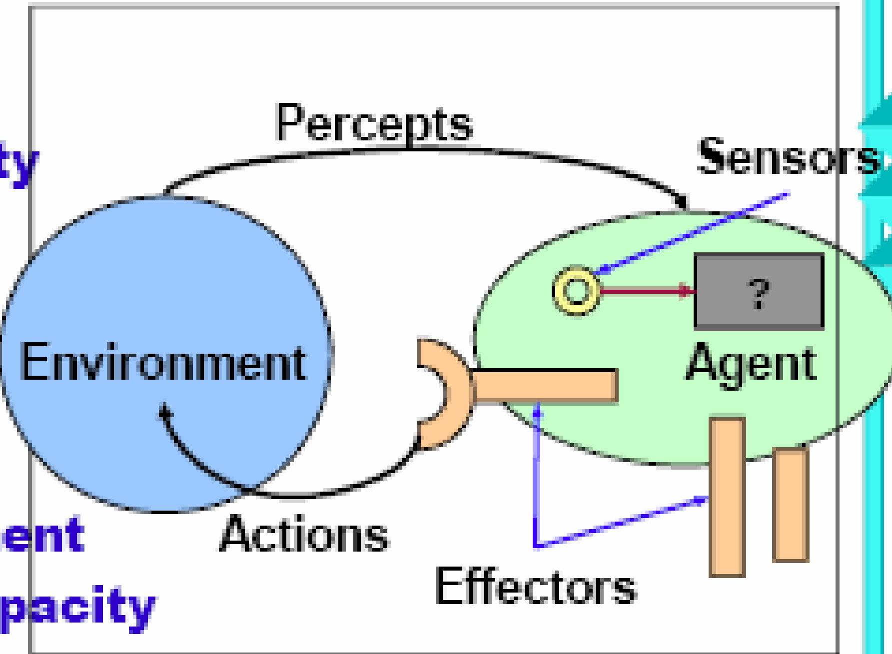
	Solitaire	Backgammon	Internet shopping	Taxi
Observable??	Yes	Yes	No	No
Deterministic??	Yes	No	Partly	No
Episodic??	No	No	No	No
Static??	Yes	Semi	Semi	No
Discrete??	Yes	Yes	Yes	No
Single-agent??	Yes	No	Yes (except auctions)	No

The environment type largely determines the agent design

The real world is (of course) partially observable, stochastic, sequential, dynamic, continuous, multi-agent

Intelligent Agents:Overview

- **Agent: Definition**
  - Any entity that perceives its environment through sensors and acts upon that environment through effectors
  - Examples (class discussion): human, robotic, software agents
- **Perception**
  - Signal from environment
  - May exceed sensory capacity
- **Sensors**
  - Acquires percepts
  - Possible limitations
- **Action**
  - Attempts to affect environment
  - Usually exceeds effector capacity
- **Effectors**
  - Transmits actions
  - Possible limitations



How Agents Should Act

- Rational Agent: Definition
  - Informal: "does the right thing, given what it believes from what it perceives"
  - What is "the right thing"?
    - First approximation: *action that maximizes success of agent*
    - Limitations to this definition?
  - Issues to be addressed now
    - How to evaluate *success*
    - When to evaluate *success*
  - Issues to be addressed later in this course
    - Uncertainty (in environment, in actions)
    - How to express beliefs, knowledge

How Agents Should Act

- **Why Study Rationality?**
  - Recall: aspects of intelligent behavior (last lecture)
    - Engineering objectives: optimization, problem solving, decision support
    - Scientific objectives: modeling correct inference, learning, planning
  - Rational cognition: formulating *plausible* beliefs, conclusions
  - Rational action: "doing the right thing" given beliefs

Rational Agents

- "Doing the Right Thing"
  - Committing actions
    - Limited to set of effectors
    - In context of what agent knows
  - Specification (cf. software specification)
    - Preconditions, postconditions of operators
    - Caveat: not always perfectly known (for given environment)
    - Agent may also have limited knowledge of specification

Rational Agents

- **Agent Capabilities: Requirements**
  - <u>Choice</u>: select actions (and carry them out)
  - <u>Knowledge</u>: represent knowledge about environment
  - <u>Perception</u>: capability to sense environment
  - <u>Criterion</u>: *performance measure to define degree of success*
- **Possible Additional Capabilities**
  - <u>Memory</u> (internal model of <u>state</u> of the world)
  - Knowledge about effectors, reasoning process (<u>reflexive</u> reasoning)

Measuring Performance

- <u>**Performance Measure**</u>**: *How* to Determine Degree of Sucesss**
  - **Definition: *criteria that determine how successful agent is***
  - **Clearly, varies over**
    - **Agents**
    - **Environments**
  - **Possible measures?**
    - **Subjective (agent may not have capability to give accurate answer!)**
    - <u>**Objective**</u>**: *outside observation***

Measuring Performance

- Example: web crawling agent
  - Number of hits
  - Number of *relevant* hits
  - *Ratio* of relevant hits to pages explored, resources expended
  - Caveat: "you get what you ask for" (issues: redundancy, etc.)

When to Evaluate Success

- Depends on objectives (short-term efficiency, consistency, etc.)
- Is task episodic? Are there milestones? Reinforcements? (e.g., games)

Knowledge in Agents

- Rationality versus Omniscience
  - Nota Bene (*NB*): not the same
  - Distinction
    - Omniscience: knowing *actual* outcome of all actions
    - Rationality: knowing *plausible* outcome of all actions
    - Example: is crossing the street to greet a friend too risky?
  - Key question in AI
    - *What is a plausible outcome?*
    - Especially important in knowledge-based expert systems
    - Of practical important in planning, machine learning

Knowledge in Agents

- **Related questions**
  - ***How can an agent make rational decisions given beliefs about outcomes of actions?***
  - ***Specifically, what does it mean (algorithmically) to "choose the best"?***

Limitations of Rationality

- **Based only on what agent *can* perceive and do**
- **Based on what is "likely" to be right, not what "turns out" to be right**

What Is Rational?

- **Criteria**
  - **Determines what is rational *at any given time***
  - **Varies with agent, environment, *situation***
- **Performance Measure**
  - **Specified by outside observer or evaluator**
  - **Applied (consistently) to (one or more) IAs in given environment**
- **Percept Sequence**
  - **Definition: *entire history* of percepts gathered by agent**
  - **NB: may or may not be retained fully by agent (issue: state and memory)**

What Is Rational?

- **Agent Knowledge**
  - **Of environment – “required”**
  - **Of self (reflexive reasoning)**
- **Feasible Action**
  - **What can be performed**
  - **What agent believes it can attempt?**

Ideal Rationality

- **Ideal Rational Agent**
  - **Given: any possible <u>percept sequence</u>**
  - **Do: <u>ideal rational behavior</u>**
    - **Whatever action is <u>expected</u> to maximize performance measure**
    - **NB: expectation – informal sense (for now); mathematical foundation soon**
  - **Basis for action**
    - **<u>Evidence</u> provided by percept sequence**
    - **<u>Built-in knowledge</u> possessed by the agent**

Ideal Rationality

- **Ideal Mapping from Percepts to Actions**
  - **Mapping *p*: *percept sequence* $\rightarrow$ *action***
    - **<u>Representing</u> *p* as list of pairs: infinite (unless explicitly bounded)**
    - **<u>Using</u> *p*: specifies <u>ideal mapping</u> from percepts to actions (i.e., <u>ideal agent</u>)**
    - **<u>Finding</u> explicit *p*: in principle, could use trial and error**
    - ***Other (implicit) representations may be easier to acquire!***

Autonomy

- **Built-In Knowledge**
  - ***What if agent ignores percepts?***
  - **Possibility**
    - **All actions based on agent's own knowledge**
    - **Agent said to *lack* autonomy**
  - **Examples**
    - **"Preprogrammed" or "hardwired" industrial robots**
    - **Clocks**
    - **Other sensorless automata**
    - **NB: to be distinguished from closed versus open loop systems**

Autonomy[2]

- Justificiation for Autonomous Agents
  - Sound engineering practice: “Intelligence demands robustness, adaptivity”
  - This course: mathematical and CS basis of autonomy in IAs

Structure of Intelligent Agents

- **Agent Behavior**
  - **Given: sequence of percepts**
  - **Return: IA's actions**
    - **Simulator: description of results of actions**
    - **Real-world system: committed action**
- **Agent Programs**
  - **Functions that implement *program***
  - **Assumed to run in computing environment (architecture)**
    - **Hardware architecture: computer organization**
    - **Software architecture: programming languages, operating systems**
  - ***Agent = architecture + program***

Looking Ahead: Search

- Solving Problems by Searching
  - Problem solving agents: design, specification, implementation
  - Specification components
    - Problems – formulating *well-defined* ones
    - Solutions – requirements, constraints
  - Measuring performance
- Formulating Problems as (State Space) Search
- Data Structures Used in Search

Problem-Solving Agents [1]: Preliminary Design

- **Justification**
  - **Rational IAs: act to *reach* environment that maximizes performance measure**
  - **Need to formalize, <u>operationalize</u> this definition**
- **Practical Issues**
  - **Hard to find appropriate *sequence of states***
  - **Difficult to translate into IA design**
- **Goals**
  - **Chapter 2, R&N: simplifies task of translating agent specification to formal design**
  - **First step in problem solving: formulation of goal(<u>s</u>) – "accept no substitutes"**

Problem-Solving Agents [2]: Preliminary Design

- **Problem Formulation**
  - <u>Given</u>: initial state, desired goal, specification of actions
  - <u>Find</u>: *achievable* sequence of states (actions) mapping from initial to goal state
- **Search**
  - <u>Actions</u>: cause transitions between world states (e.g., applying effectors)
  - Typically specified in terms of finding sequence of states (operators)

Problem-Solving Agents [1]:Specification

- <u>Input</u>: **Informal Objectives; Initial, Intermediate, Goal States; Actions**
- <u>**Output**</u>
  - **Path from initial to goal state**
  - **Leads to design requirements for <u>state space search</u> problem**
- **Logical Requirements**
  - **States: representation of state of world (example: starting city, graph representation of Romanian map)**
  - **Operators: descriptors of possible actions (example: moving to adjacent city)**
  - **Goal test: state $\rightarrow$ boolean (example: at destination city?)**
  - **Path cost: *based on search, action costs* (example: number of edges traversed)**

Problem-Solving Agents [2]: Specification

- **Operational Requirements**
  - **Search algorithm to find path**
  - **Objective criterion: minimum cost**
- **Environment**
  - **Agent can search in environment according to specifications**
  - **Sometimes has full state and action descriptors; *sometimes not*!**

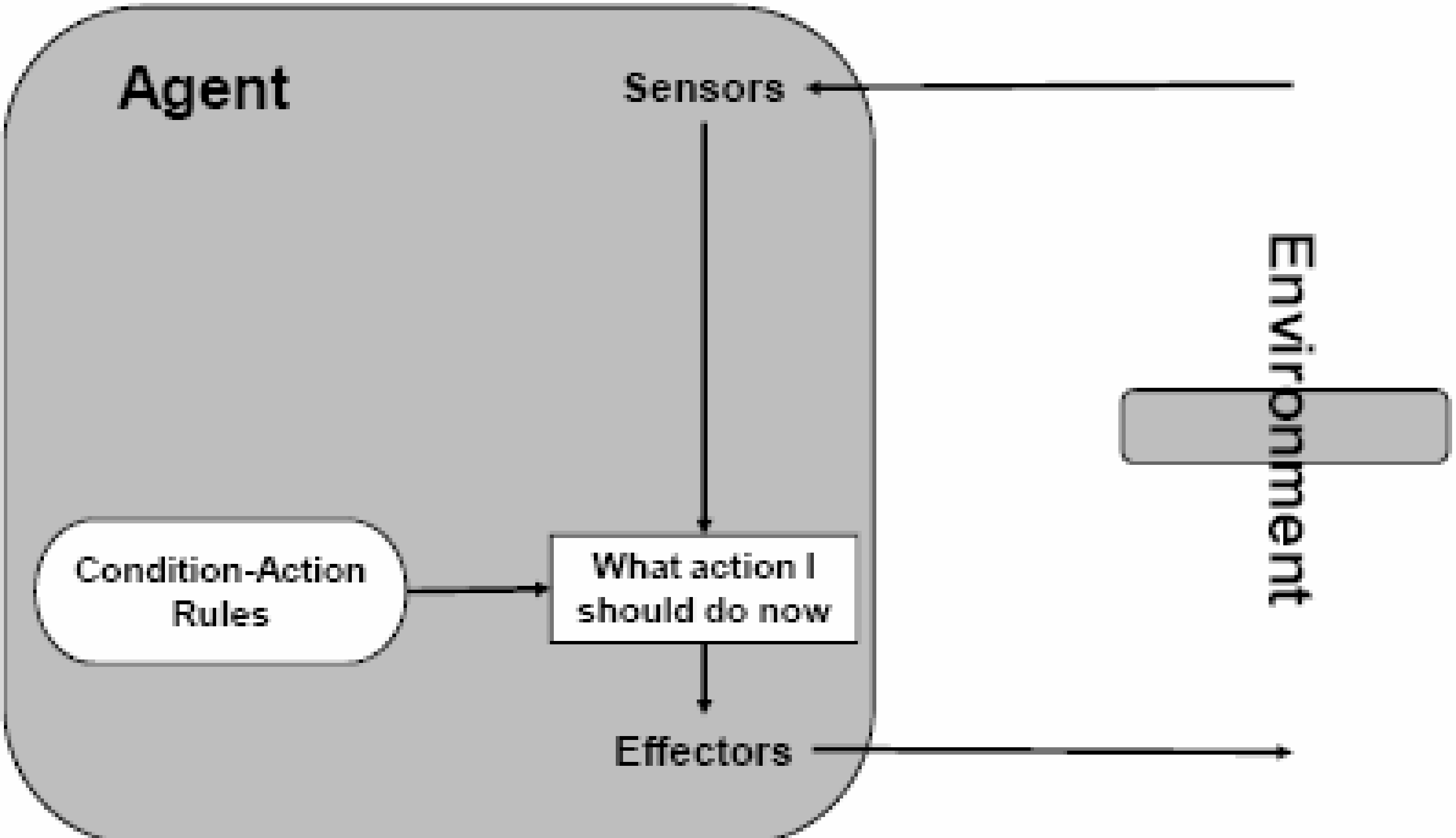
Agent types

Four basic types in order of increasing generality:

- simple reflex agents
- reflex agents with state
- goal-based agents
- utility-based agents

All these can be turned into learning agents

Agent Framework: Simple Reflex Agents [1]



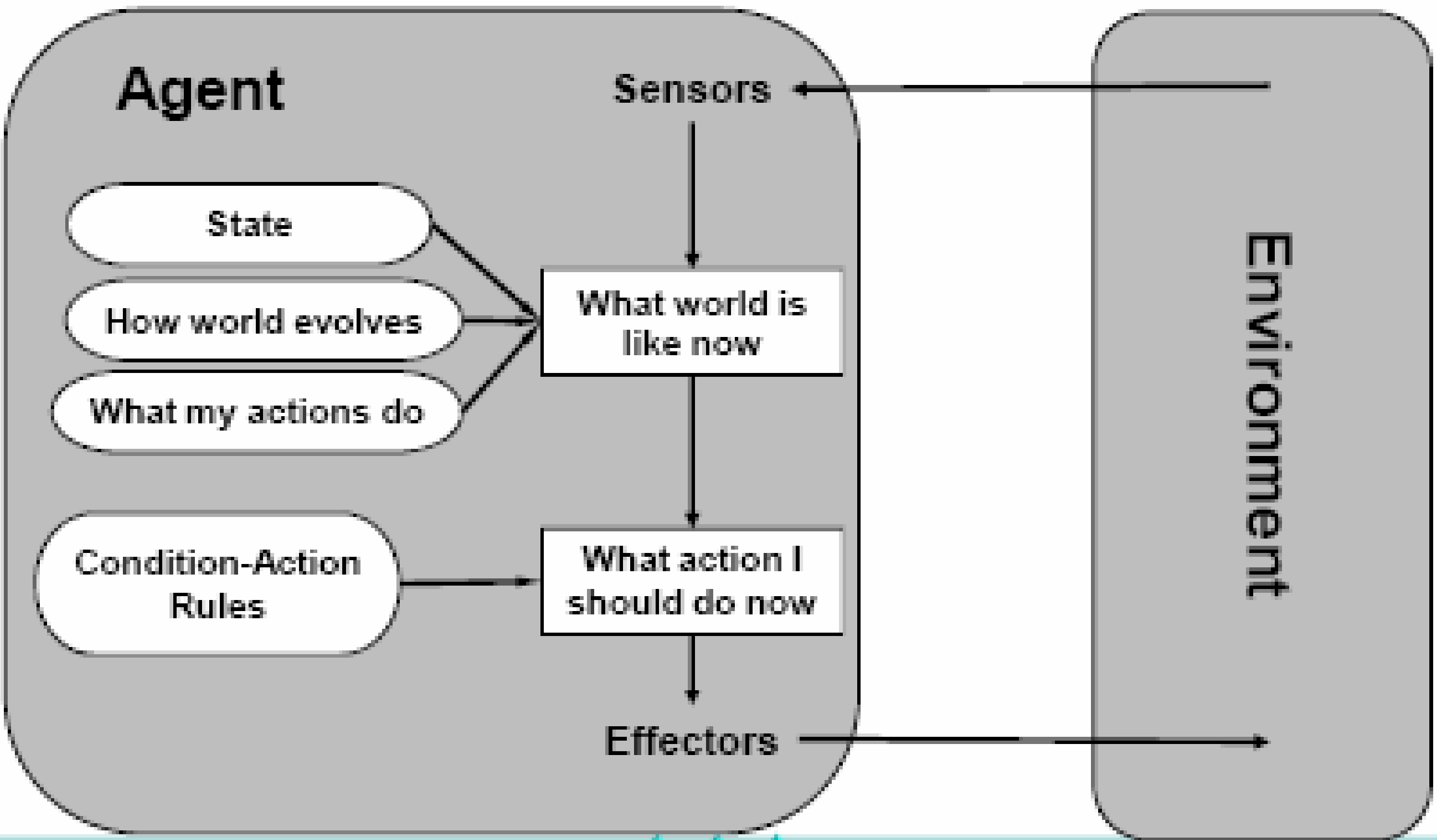
Agent Framework: Simple Reflex Agents [2]

- **Implementation and Properties**
  - **Instantiation of generic skeleton agent**
  - **function *SimpleReflexAgent* (*percept*) returns action**
    - **static: *rules*, set of condition-action rules**
    - ***state* ← *Interpret-Input* (*percept*)**
    - ***rule* ← *Rule-Match* (*state, rules*)**
    - ***action* ← *Rule-Action* {*rule*}**
    - **return *action***

Agent Framework: Simple Reflex Agents [3]

- **Advantages**
  - **Selection of best action based only on current state of world and rules**
  - **Simple, very efficient**
  - ***Sometimes* robust**
- **Limitations and Disadvantages**
  - **No memory (doesn't keep track of world)**
  - **Limits range of applicability**

Agent Frameworks: (Reflex) Agents with State [1]



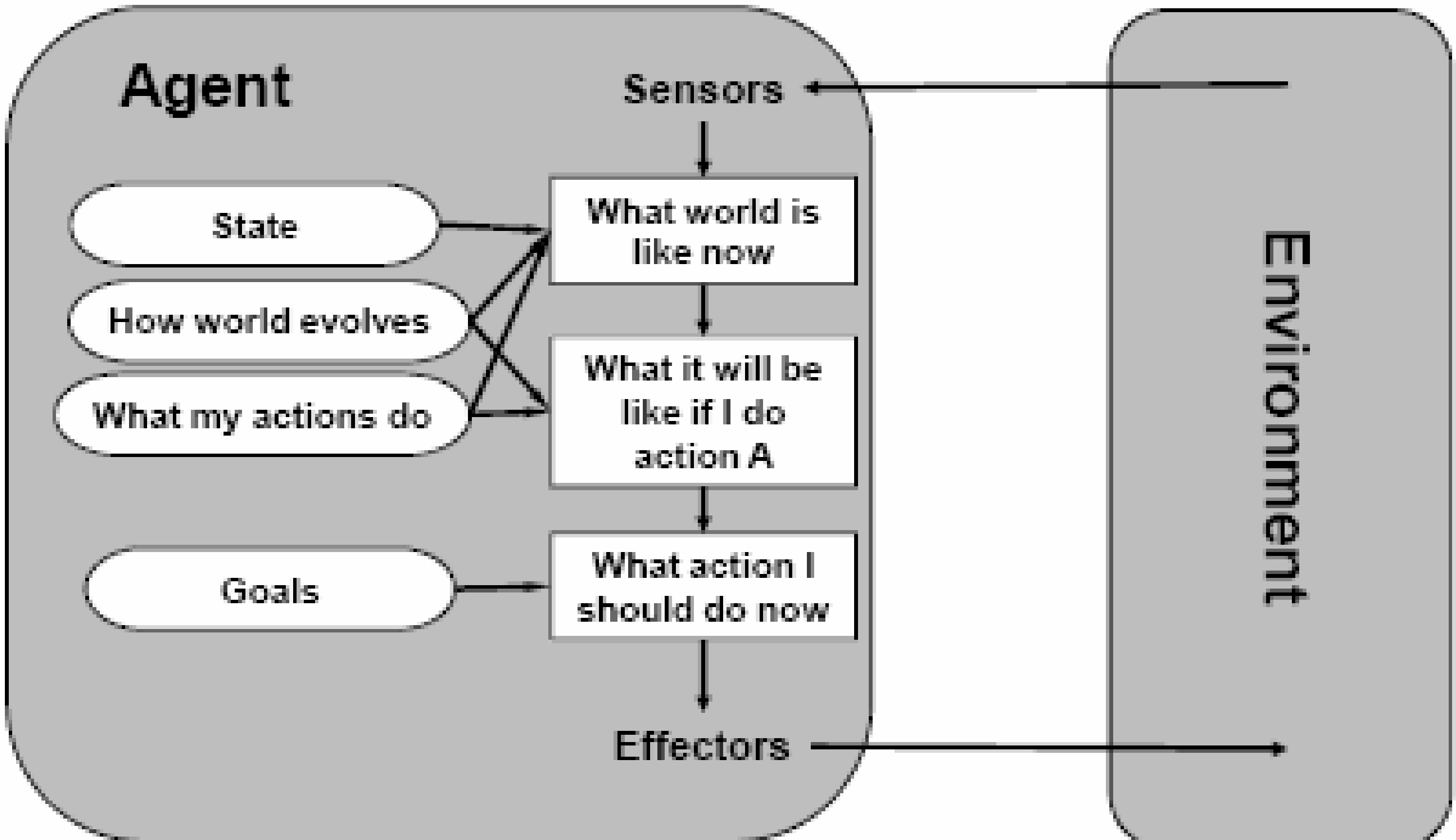
Agent Frameworks: (Reflex) Agents with State [2]

- **Implementation and Properties**
  - <u>Instantiation</u> of generic skeleton agent
  - <u>function</u> *ReflexAgentWithState* (*percept*) <u>returns</u> action
    - <u>static</u>: *state*, description of current world state; *rules*, set of condition-action rules
    - *state* ← *Update-State* (*state*, *percept*)
    - *rule* ← *Rule-Match* (*state, rules*)
    - *action* ← *Rule-Action* {*rule*}
    - <u>return</u> *action*

Agent Frameworks: (Reflex) Agents with State [3]

- **Advantages**
  - Selection of best action based only on current state of world and rules
  - Able to reason over past states of world
  - Still efficient, *somewhat* more robust
- **Limitations and Disadvantages**
  - No way to express goals and preferences relative to goals
  - Still limited range of applicability

Agent Frameworks: Goal-Based Agents [1]



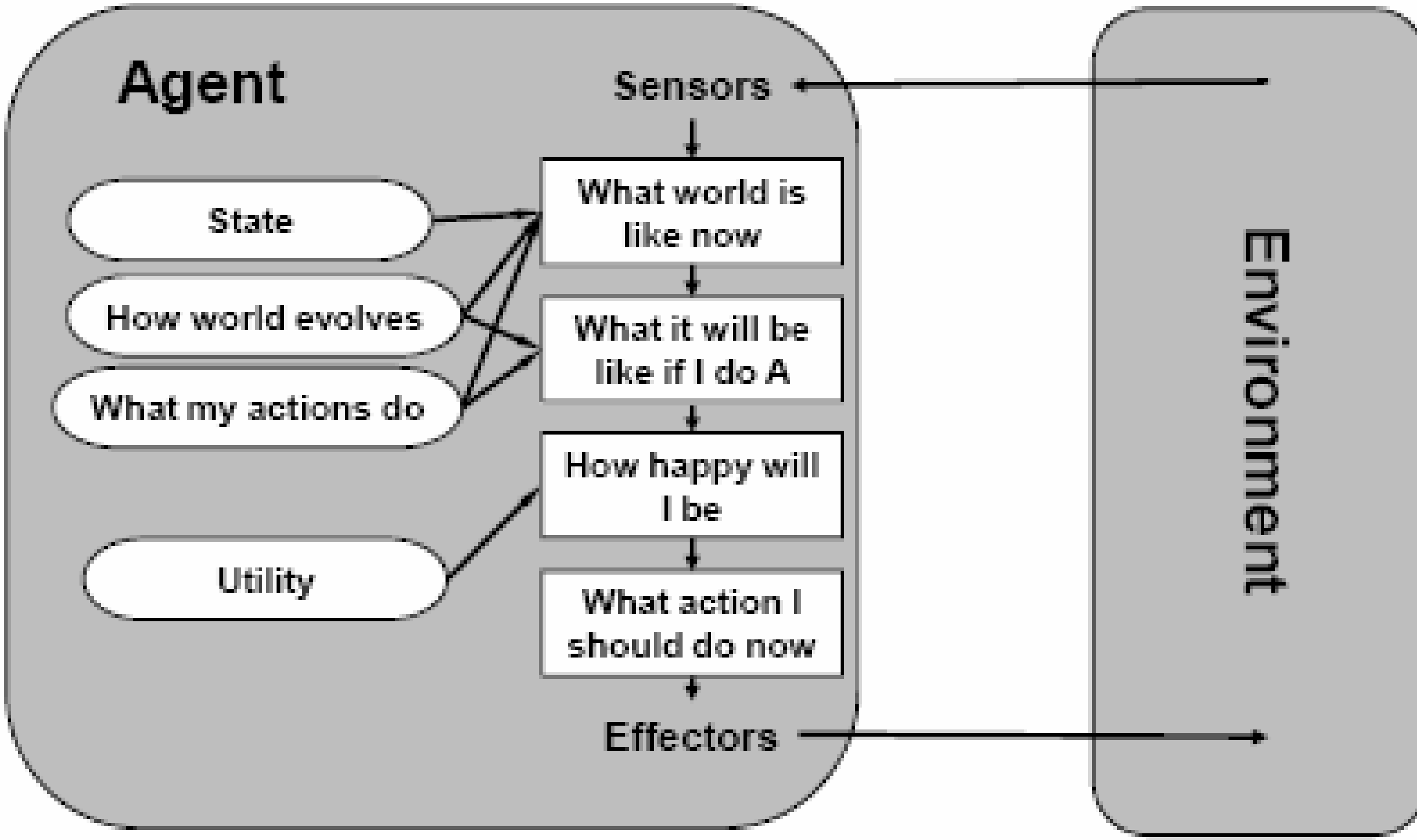
Agent Frameworks: Goal-Based Agents [2]

- Implementation and Properties
- Instantiation of generic skeleton agent
- Functional Description
  - Classical Planning (Ch. 10)
  - Requires more formal specification

Agent Frameworks: Goal-Based Agents [3]

- **Advantages**
  - Able to reason over goal, intermediate, and initial states
  - Basis: automated reasoning
    - One implementation: theorem proving (first-order logic)
    - Powerful representation language and inference mechanism
- **Limitations and Disadvantages**
  - Efficiency limitations: can't feasible solve many general problems
  - No way to express <u>preferences</u>

Agent Frameworks: Utility-Based Agents [1]



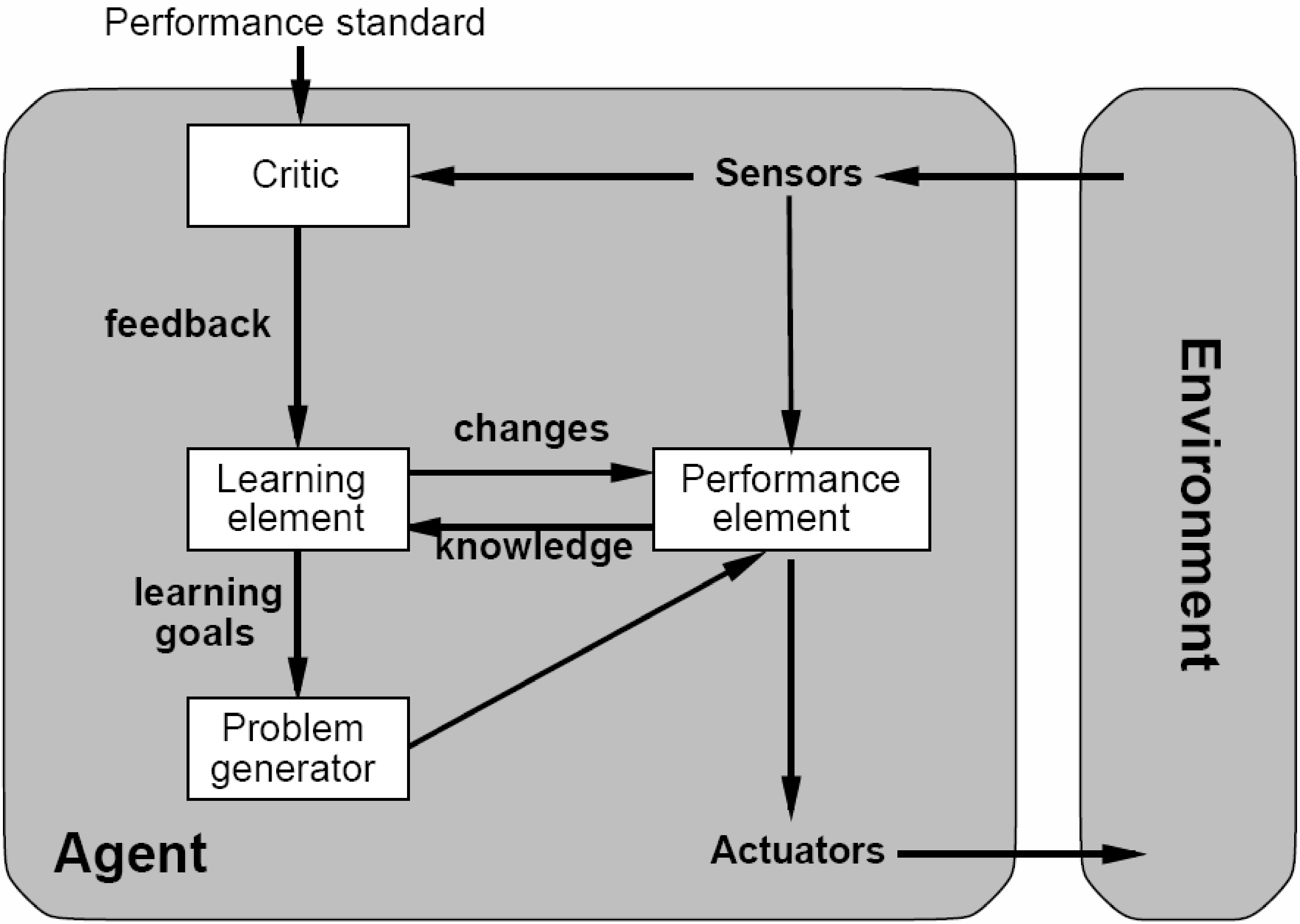
Agent Frameworks: Utility-Based Agents [2]

- **Implementation and Properties**
  - **Instantiation of generic skeleton agent**
  - **function *SimpleReflexAgent* (*percept*) returns action**
    - **static: *rules*, set of condition-action rules**
    - ***state* ← *Interpret-Input* (*percept*)**
    - ***rule* ← *Rule-Match* (*state, rules*)**
    - ***action* ← *Rule-Action* (*rule*)**
    - **return *action***

Agent Frameworks: Utility-Based Agents [3]

- **Advantages**
  - **Selection of best action based only on current state of world and rules**
  - **Simple, very efficient**
  - ***Sometimes* robust**
- **Limitations and Disadvantages**
  - **No memory (doesn't keep track of world)**
  - **Limits range of applicability**

Learning agents



Summary

Agents interact with environments through actuators and sensors

The agent function describes what the agent does in all circumstances

The performance measure evaluates the environment sequence

A perfectly rational agent maximizes expected performance

Agent programs implement (some) agent functions

PEAS descriptions define task environments

Environments are categorized along several dimensions:
observable? deterministic? episodic? static? discrete? single-agent?

Several basic agent architectures exist:
reflex, reflex with state, goal-based, utility-based

The End!