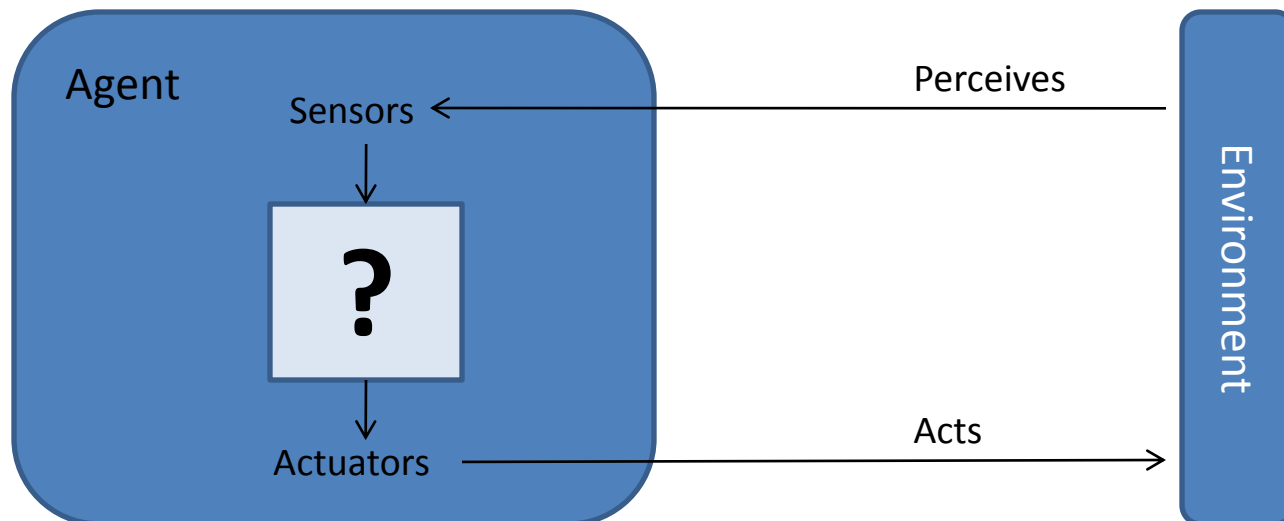


Homework 1

AGENT

- Something that acts
- Something that does something for someone else
- Something that acts to achieve the best outcome
- Something that perceives its environment through sensors and acts upon that environment through actuators
- Agent = architecture + program



AGENT

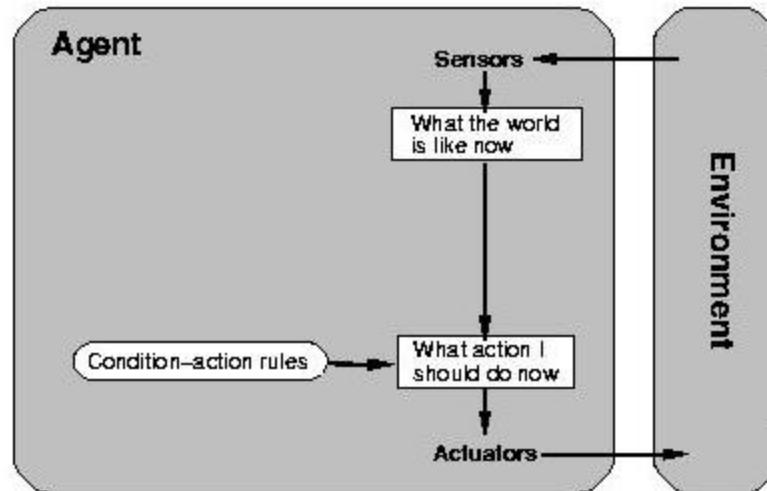
- Agent function: a function that specifies the agent's action in response to every possible percept sequence
- Agent function maps perceptions into actions
- Agent program, combined with a machine architecture, implements an agent function

- Rationality – leading to the optimal
- Rational behavior – doing what is expected to maximize one's utility function
- Rational agent – an agent that acts to achieve the best outcome

- Autonomy – independency, ability that one can act on its own
- Autonomous agent – agent that has ability to learn and adapt

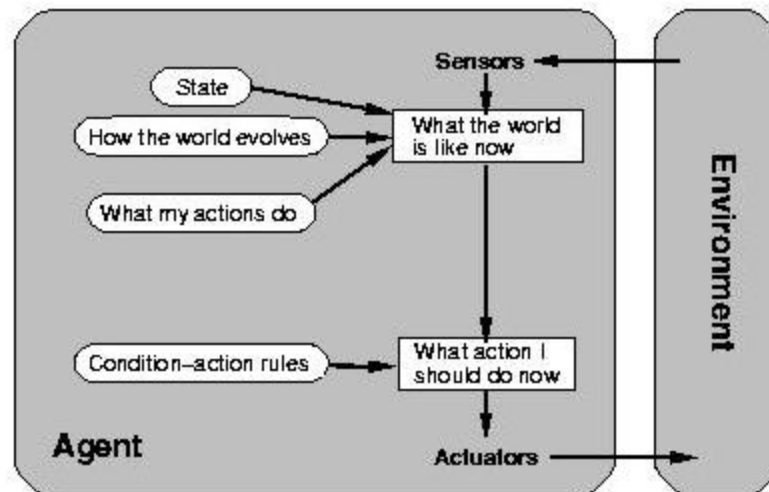
AGENT

- Reflex agent:
 - Simplest kind of agent
 - Actions are selected based on the current percept (and independent of the percept history)
 - Condition-action (if-then) rule



AGENT

- Model-based agent:
 - Actions are derived directly from an internal model of the current world state that is updated over time.
 - Models the state of the world by modeling how the world works
 - Keeps track of the part of the world it can't see at the moment
 - Actions are based on the knowledge about current state (from sensors) and previous states



AGENT

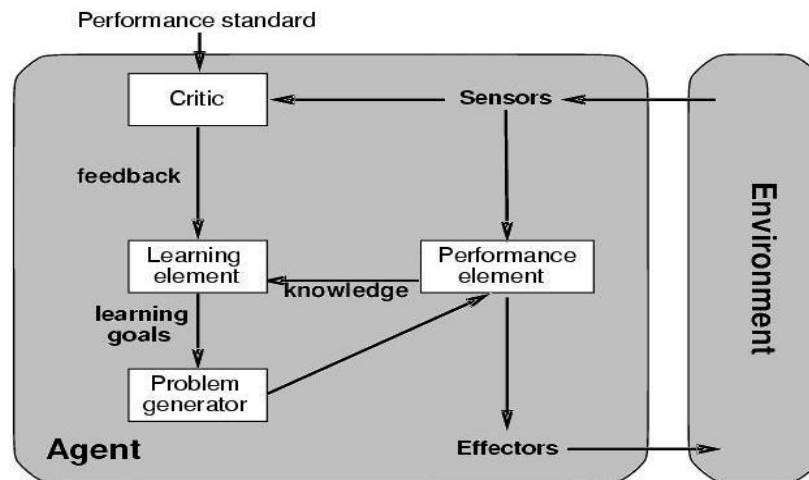
- Goal-based agent:
 - Selects actions that it believes will achieve explicitly represented goals
 - Goal provide a reason to prefer one action over the other
 - Involves decision making (not only condition-action rules)

AGENT

- Utility-based agent:
 - Selects actions that it believes will maximize the expected utility of the outcome state.
 - Some paths to the goal are better (more efficient) than others) - which combination of goals is preferred

AGENT

- Learning agent:
 - Behavior improves over time based on its experience
 - Monitoring agent's performances over time
 - Learn based on preferences of actions (feedback)
 - Problem generation – suggesting actions that will lead to new and informative experiences



Performance measure vs. utility function

- A performance measure (typically imposed by the designer) is used to evaluate the behavior of the agent in environment. It tells does agent do what it's supposed to do in the environment
- A utility function is used by an agent itself to evaluate how desirable states are. Some paths to the goal are better (more efficient) than others –which path is the best
- Does agent do what it's supposed to do vs. does agent do it in optimal way
- The utility function may not be the same as the performance measure
- An agent may have no explicit utility function at all, whereas there is always a performance measure

PEAS

Performance – which qualities it should have?

Environment – where it should act?

Actuators – how will it perform actions?

Sensors – how will it perceive environment?

Mathematician's theorem-proving assistant

P: good math knowledge, can prove theorems accurately and in minimal steps/time

E: Internet, library

A: display

S: keyboard

Autonomous Mars rover

P: Terrain explored and reported, samples gathered and analyzed

E: Launch vehicle, lander, Mars

A: Wheels/legs, sample collection device, analysis devices, radio transmitter

S: Camera, touch sensors, accelerometers, orientation sensors, wheel/joint encoders, radio receiver

Internet book-shopping agent

P: Obtain requested/interesting books, minimize expenditure

E: Internet

A: Follow link, enter/submit data in fields, display to user

S: Web pages, user requests

Robot soccer player

P: Winning game, goals for/against

E: Field, ball, own team, other team, own body

A: Devices (e.g., legs) for locomotion and kicking

S: Camera, touch sensors, accelerometers,
orientation sensors, wheel/joint encoders

- **Fully/partially observable** – complete or partial state of the environment?
- **Deterministic/stochastic** – can next state be completely determined by the current state and the action?
- **Episodic/sequential** – can actions be divided into episodes, and are episodes independent?
- **Static/dynamic** – does environment change?
- **Discrete/continuous** – is there a limited number of distinct, well defined percepts and actions?
- **Single/multi-agent** – does agent operate alone or in collaboration with other agents?

Task Environment	Observable	Deterministic	Episodic	Static	Discrete	Agents
Robot soccer	Partially	Stochastic	Sequential	Dynamic	Continuous	Multi
Internet book-shopping	Partially	Deterministic*	Sequential	Static*	Discrete	Single
Autonomous Mars rover	Partially	Stochastic	Sequential	Dynamic	Continuous	Single
Mathematician's assistant	Fully	Deterministic	Sequential	Semi	Discrete	Multi

Problem 3.1 Searching

- A **state** is a situation that an agent can find itself in. We distinguish two types of states:
 - world states (the actual concrete situations in the real world)
 - representational states (the abstract descriptions of the real world that are used by the agent in deliberating about what to do).
- A **state space** is a graph whose nodes are the set of all states, and whose links are actions that transform one state into another.
- A **search tree** is a tree (a graph with no undirected loops) in which the root node is the start state and the set of children for each node consists of the states reachable by taking any action.

Problem 3.1 cont.

- A **search node** is a node in the search tree.
- A **goal** is a state that the agent is trying to reach.
- An **action** is something that the agent can choose to do.
- A **successor function** described the agent's options: given a state, it returns a set of (action, state) pairs, where each state is the state reachable by taking the action.
- The **branching factor** in a search tree is the maximum number of actions available to the agent at any node.

Problem 3.3

LEGAL-ACTIONS(s) – set of actions that are legal in state s

RESULT(a, s) – state that results from performing legal action a in state s

SUCCESSOR-FN(s) – set of <action, successor> ordered pairs

```
##### successor_fn defined in terms of result and legal_actions
```

```
def successor_fn(s):
```

```
    return [(a, result(a, s)) for a in legal_actions(s)]
```

```
##### legal_actions and result defined in terms of successor_fn
```

```
def legal_actions(s):
```

```
    return [a for (a, s) in successor_fn(s)]
```

```
def result(a, s):
```

```
    for (a1, s1) in successor_fn(s):
```

```
        if a == a1:
```

```
            return s1
```


N Queens problem

Therefore, in the worst case, there are n choices for where to place 1st queen, $(n-3)$ choices where to place 2nd queen, $(n-6)$ places where to place 3rd queen, etc.

If we mark with QS total size of space where we can place n queens, we have:

$$QS = n(n-3)(n-6)(n-9)\dots(4)(1)$$

Note: we're considering the worst case

N Queens problem

$(n) \geq (n)$
 $(n) \geq (n-1)$
 $(n) \geq (n-2)$
 $(n-3) \geq (n-3)$
 $(n-3) \geq (n-4)$
 $(n-3) \geq (n-5)$
.
.
.
 $4 \geq 4$
 $4 \geq 3$
 $4 \geq 2$
 $1 \geq 1$

Multiply
expressions on the
left and the right
sides

Note:
 $n \geq 1$

$$(n)(n)(n)(n-3)(n-3)(n-3)\dots(4)(4)(4)(1) \geq (n)(n-1)(n-2)(n-3)(n-4)(n-5)\dots(4)(3)(2)(1)$$

$$(QS)(QS)(QS) \geq n! \quad \longrightarrow \quad QS \geq \text{cuberoot}(n!)$$

N Queens problem

- Assume a computer tests 1000 states/sec

N	Time
10	0.15s
15	11s
20	22m
25	3d
28	78d
30	2y

