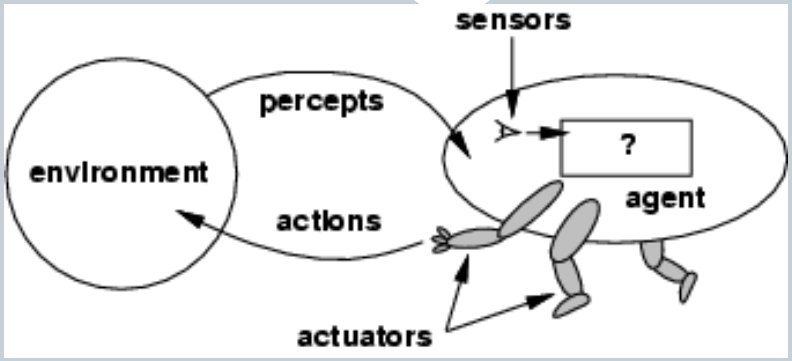
Agents

- An agent is anything that can be viewed as perceiving its environment through sensors and acting upon that environment through actuators

- Human agent:
  - eyes, ears, and other organs for sensors;
  - hands, legs, mouth, and other body parts for actuators

- Robotic agent:
  - cameras and infrared range finders for sensors
  - various motors for actuators

Agents and environments

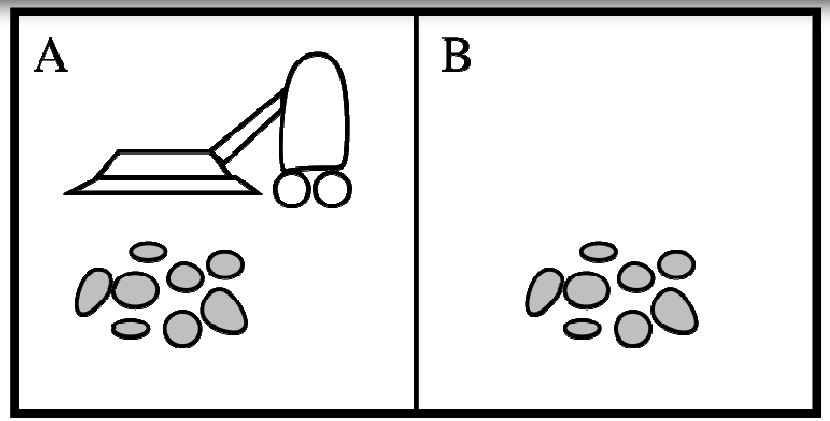


- The agent function maps from percept histories to actions:

$$[f: \mathcal{P}^* \rightarrow \mathcal{A}]$$

- The agent program runs on the physical architecture to produce f
- agent = architecture + program

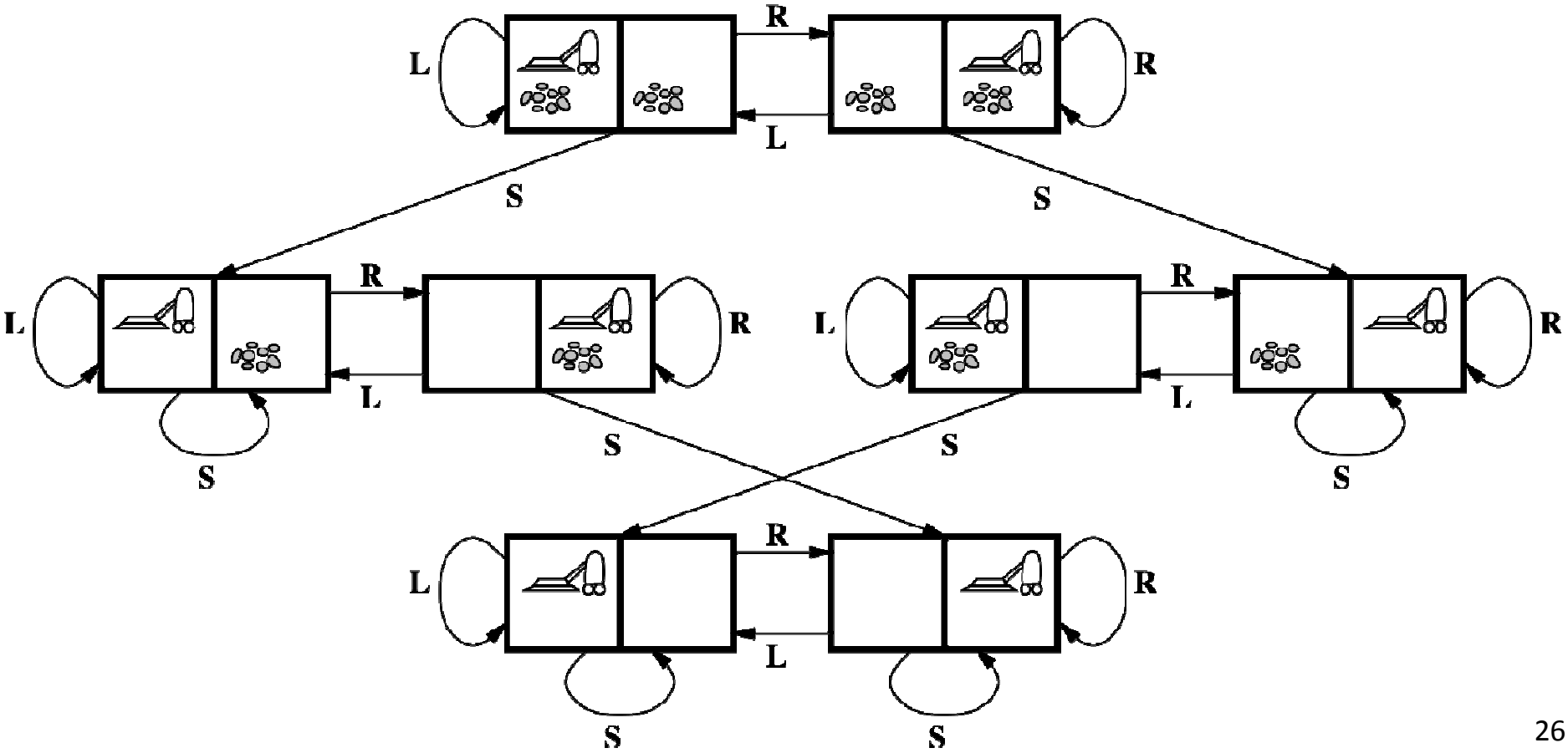
Example: Vacuum cleaning robot



- Percepts P = {[A,Clean], [A,Dirty], [B,Clean], [B,Dirty]}
- Actions A = {Left, Right, Suck, NoOp}
- Agent function: $f : P^* \rightarrow A$
- Example: $f([A, dirty]) = Suck$
  $f([A, clean]) = Right$
  $f([A, clean], [B, dirty]) = Suck$

Modeling the environment

- Set of states S (not necessarily finite)
- State transitions depend on current state and actions (can be stochastic or nondeterministic)



Rational agents

- Rationality
  - Performance measuring success
  - Agents prior knowledge of environment
  - Actions that agent can perform
  - Agent's percept sequence to date

- Rational Agent: For each possible percept sequence, a rational agent should select an action that is expected to maximize its performance measure, given the evidence provided by the percept sequence and whatever built-in knowledge the agent has.

Rational agents

- An agent should strive to "do the right thing", based on what it can perceive and the actions it can perform.
- The right action is the one that will cause the agent to be most successful
- Performance measure: An objective criterion for success of an agent's behavior
- E.g., performance measure of a vacuum-cleaner agent could be amount of dirt cleaned up, amount of time taken, amount of electricity consumed, amount of noise generated, etc.
- As a general rule, it is better to design performance measures according to what one actually wants in the environment. Rather than according to how one thinks the agent should behave (amount of dirt cleaned vs a clean floor)
- A more suitable measure would reward the agent for having a clean floor

Rationality

- Rational is different to omniscient
  - Percepts may not supply all relevant information
- Rational is different to being perfect
  - Rationality maximizes expected outcome while perfection maximizes actual outcome.

Autonomy in Agents

The **autonomy** of an agent is the extent to which its behaviour is determined by its own experience

- Extremes
  - No autonomy – ignores environment/data
  - Complete autonomy – must act randomly/no program
- Example: baby learning to crawl
- Ideal: design agents to have some autonomy
  - Possibly good to become more autonomous in time

Specifying the task environment (PEAS)

- PEAS:
  - **P**erformance measure,
  - **E**nvironment,
  - **A**ctuators,
  - **S**ensors
- In designing an agent, the first step must always be to specify the task environment (PEAS) as fully as possible

Other PEAS Examples

Agent Type	Performance Measure	Environment	Actuators	Sensors
Medical diagnosis system	healthy patient, costs, lawsuits	patient, hospital, stuff	display questions, tests, diagnoses, treatments, referrals	keyboard entry of symptoms, findings, patient's answers
Satellite image analysis system	correct image categorization	downlink from orbiting satellite	display categorization of scene	color pixel arrays
Part-picking robot	percentage of parts in correct bins	conveyor belt with parts, bins	jointed arm and hand	camera, joint angle sensors
Refinery controller	purity, yield, safety	refinery, operators	valves pumps, heaters displays	temperature, pressure, chemical sensors
Interactive English tutor	student's score on test	set of students, testing agency	display exercises, suggestions, corrections	keyboard entry

Environment types

- Fully observable vs. partially observable
- Deterministic vs. stochastic
- Episodic vs. sequential
- Static vs. dynamic
- Discrete vs. continuous
- Single agent vs. multiagent

Environment types

Fully observable vs. partially observable:

- An environment is fully observable if an agent's sensors give it access to the complete state of the environment at each point in time.
- Fully observable environments are convenient, because the agent need not maintain any internal state to keep track of the world
- An environment might be partially observable because of noisy and inaccurate sensors or because parts of the state are simply missing from the sensor data
- Examples: vacuum cleaner with local dirt sensor, taxi driver

Environment types

Deterministic vs. stochastic:

- The environment is deterministic if the next state of the environment is completely determined by the current state and the action executed by the agent.
- In principle, an agent need not worry about uncertainty in a fully observable, deterministic environment
- If the environment is partially observable then it could appear to be stochastic
- Examples: Vacuum world is deterministic while taxi driver is not
- If the environment is deterministic except for the actions of other agents, then the environment is strategic

Environment types

Episodic vs. sequential:

- In episodic environments, the agent's experience is divided into atomic "episodes" (each episode consists of the agent perceiving and then performing a single action), and the choice of action in each episode depends only on the episode itself.
- Examples: classification tasks

- In sequential environments, the current decision could affect all future decisions
- Examples: chess and taxi driver

Environment types

Static vs. dynamic:

- The environment is unchanged while an agent is deliberating.
- Static environments are easy to deal with because the agent need not keep looking at the world while it is deciding on the action or need it worry about the passage of time
- Dynamic environments continuously ask the agent what it wants to do
- The environment is semi-dynamic if the environment itself does not change with the passage of time but the agent's performance score does
- Examples: taxi driving is dynamic, chess when played with a clock is semi-dynamic, crossword puzzles are static

Environment types

Discrete vs. continuous:

- A limited number of distinct, clearly defined states, percepts and actions.
- Examples: Chess has finite number of discrete states, and has discrete set of percepts and actions. Taxi driving has continuous states, and actions

Environment types

Single agent vs. multiagent:

- An agent operating by itself in an environment is single agent
- Examples: Crossword is a single agent while chess is two-agents
- Question: Does an agent A have to treat an object B as an agent or can it be treated as a stochastically behaving object
- Whether B's behaviour is best described by as maximizing a performance measure whose value depends on agent's A behaviour
- Examples: chess is a competitive multiagent environment while taxi driving is a partially cooperative multiagent environment

Environment types

Task Environment	Observable	Deterministic	Episodic	Static	Discrete	Agents
Crossword puzzle	Fully	Deterministic	Sequential	Static	Discrete	Single
Chess with a clock	Fully	Strategic	Sequential	Semi	Discrete	Multi
Poker	Partially	Stochastic	Sequential	Static	Discrete	Multi
Backgammon	Fully	Stochastic	Sequential	Static	Discrete	Multi
Taxi driving	Partially	Stochastic	Sequential	Dynamic	Continuous	Multi
Medical Diagnosis	Partially	Stochastic	Sequential	Dynamic	Continuous	Single
Image Analysis	Fully	Deterministic	Episodic	Semi	Continuous	Single
Part-picking robot	Partially	Stochastic	Episodic	Dynamic	Continuous	Single
Refinery controller	Partially	Stochastic	Sequential	Dynamic	Continuous	Single
Interactive English Tutor	Partially	Stochastic	Sequential	Dynamic	Discrete	Multi

- The environment type largely determines the agent design
- The real world is (of course) partially observable, stochastic, sequential, dynamic, continuous, multi-agent

Simple and Difficult Environments

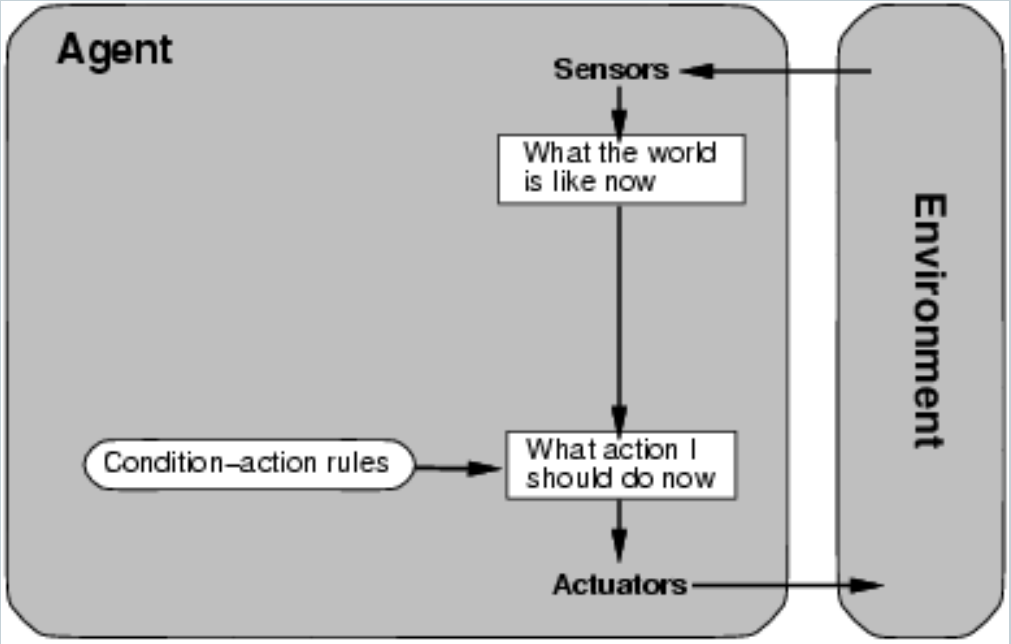
Property	Simple	Difficult
Knowledge	Known	Unknown
Observability	Accessible	Inaccessible
Dynamics of Changes	Static	Dynamic
Detail of Models	Discrete	Continuous
Short-term Action Effects	Deterministic	Stochastic
Long-term Action Effects	Episodic	Sequential
Number of Agents	Single	Multiple

- ***The key in designing successful AI applications is to understand how we can make environments simpler for the agent!***

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

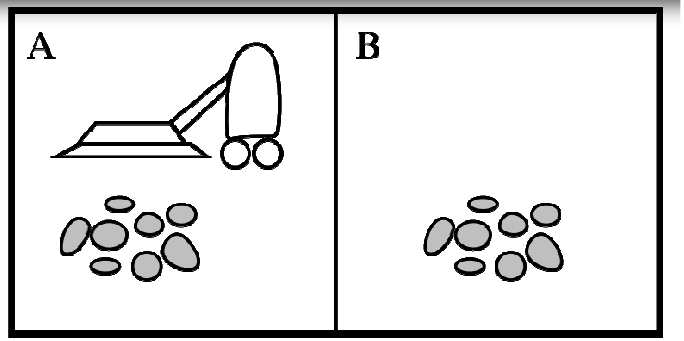
Simple reflex agents




```
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
```

Example

Percept	Action
[A,dirty]	Suck
[B,dirty]	Suck
[A,clean]	Right
[B,clean]	Left

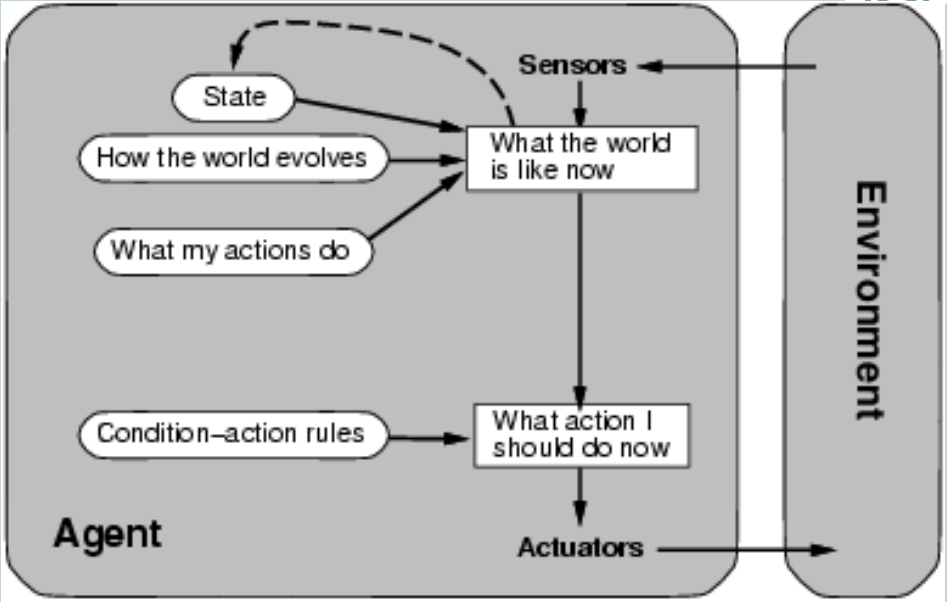


- Will never stop (noop), since we can't remember state
- This is a fundamental problem of simple reflex agents in partially observable environments!

Simple reflex agents

- Simple but very limited intelligence
- Infinite loops
  - Suppose vacuum cleaner does not keep track of location. What do you do on clean left of A or right on B is infinite loop
  - Randomize action
- Chess – openings, endings
  - Lookup table (not a good idea in general)
    - 35^{100} entries required for the entire game

Model-based reflex agents



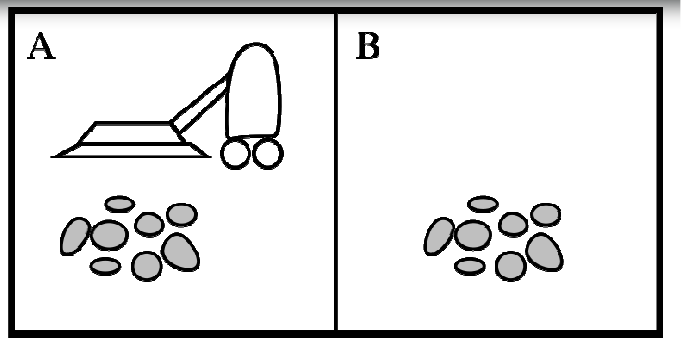
- Know how world evolves
  - Overtaking car gets closer from behind
- How agents actions affect the world
  - Wheel turned clockwise takes you right
- Model base agents update their state

```
function REFLEX-AGENT-WITH-STATE(percept) returns action
  static: state, a description of the current world state
          rules, a set of condition-action rules

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

Example

State vars: cleanA = cleanB = false



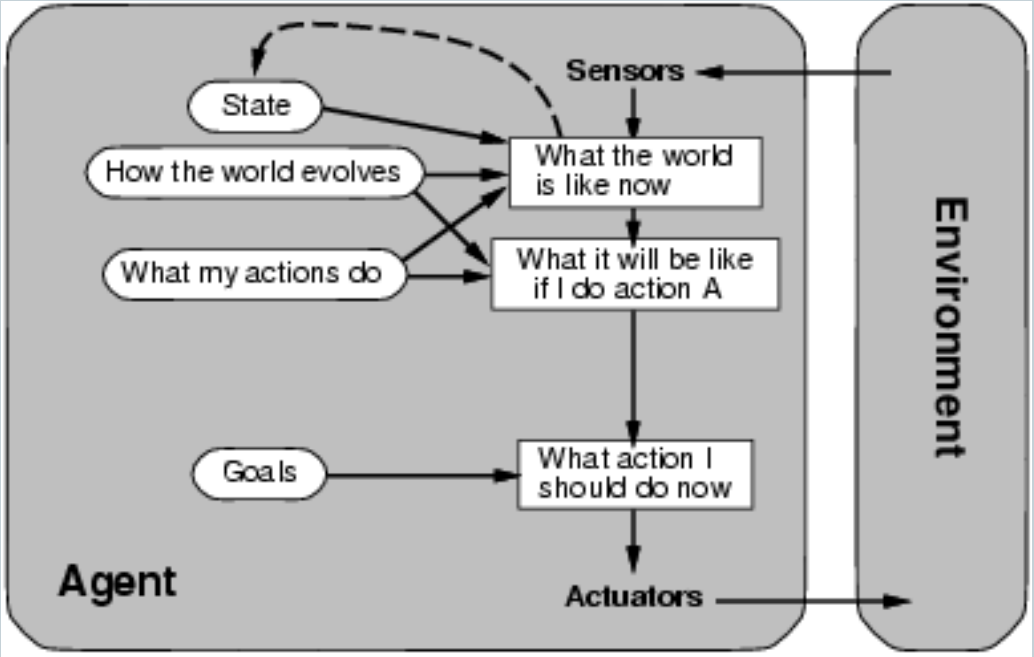
Percept	cleanA	cleanB	Action	State change
[X,dirty]	?	?	Suck	cleanX = true
[A,clean]	?	true	NoOp	
[A,clean]	?	false	Right	
[B,clean]	true	?	NoOp	
[B,clean]	false	?	Left	

? means “don’t care”

Goal-based agents

- knowing state and environment? Enough?
  - Taxi can go left, right, straight
- Have a goal
  - A destination to get to
- Uses knowledge about a goal to guide its actions
  - E.g., Search, planning

Goal-based agents



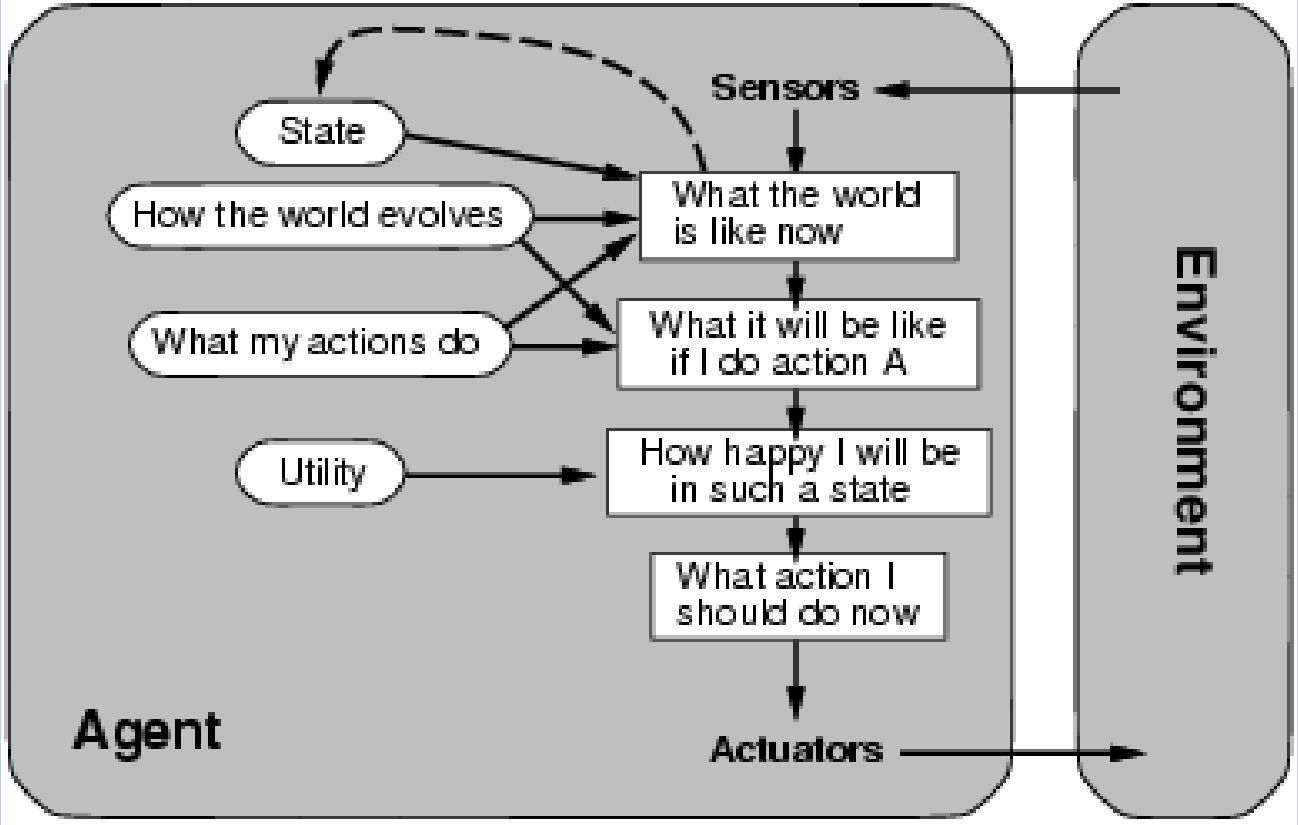
- Reflex agent breaks when it sees brake lights. Goal based agent reasons
  - Brake light -> car in front is stopping -> I should stop -> I should use brake

Utility-based agents

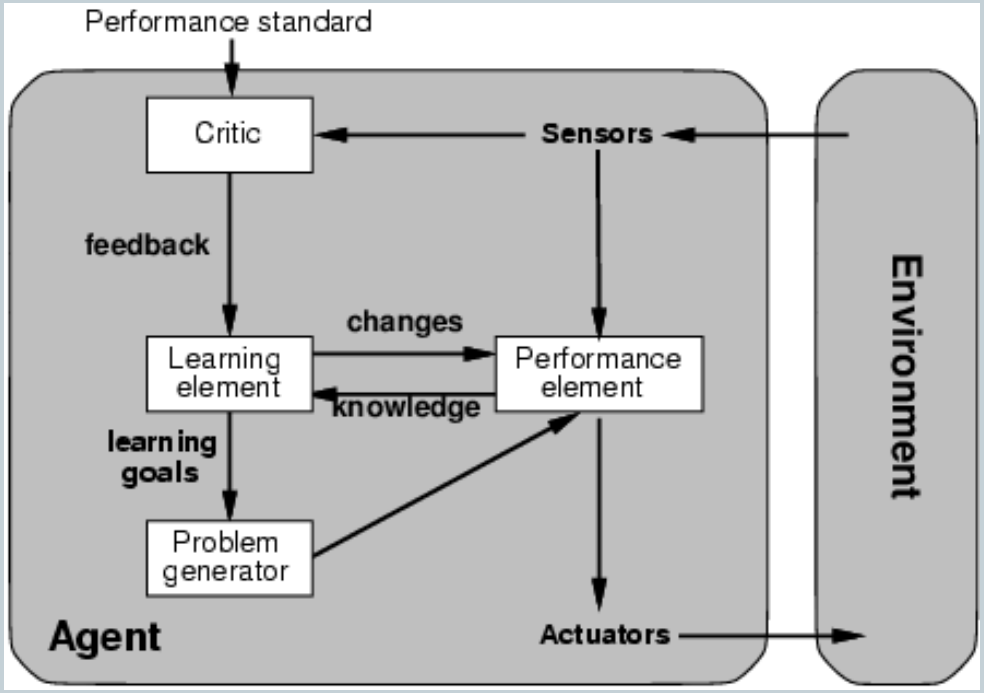
- Goals are not always enough
  - Many action sequences get taxi to destination
  - Consider other things. How fast, how safe.....

- A utility function maps a state onto a real number which describes the associated degree of happiness.

Utility-based agents

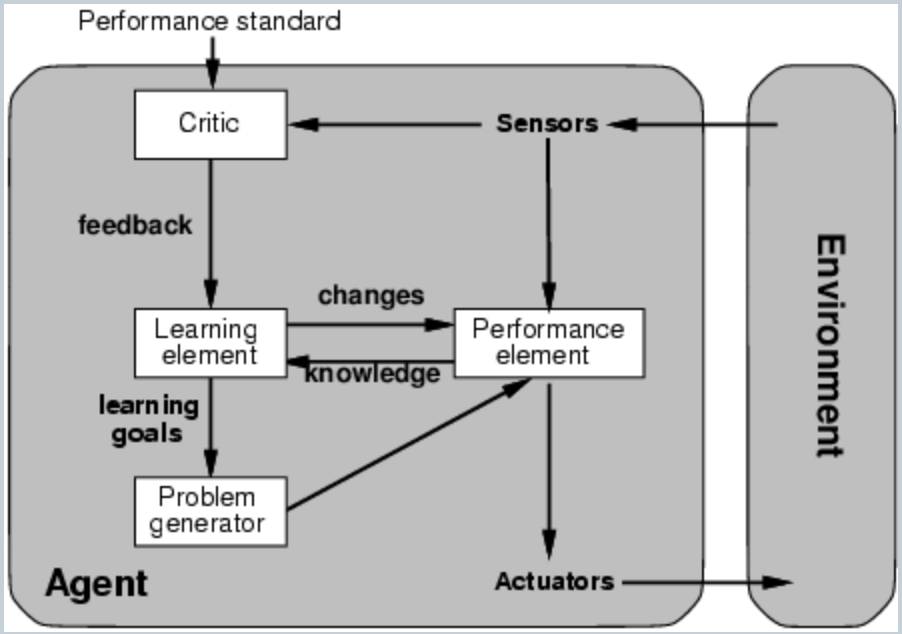


Learning agents



- Performance element is what was previously the whole agent
  - Input sensor
  - Output action
- Learning element
  - Modifies performance element

Learning agents



- Critic: how the agent is doing
  - Input: checkmate?
  - Fixed
- Problem generator
  - Tries to solve the problem differently instead of optimizing

Learning agents(Taxi driver)

- Performance element
  - How it currently drives
- Taxi driver Makes quick left turn across 3 lanes
  - Critics observe shocking language by passenger and other drivers and informs bad action
  - Learning element tries to modify performance elements for future
  - Problem generator suggests experiment out something called Brakes on different Road conditions
- Critics is not always easy
  - shocking language
  - Less tip
  - Less passengers