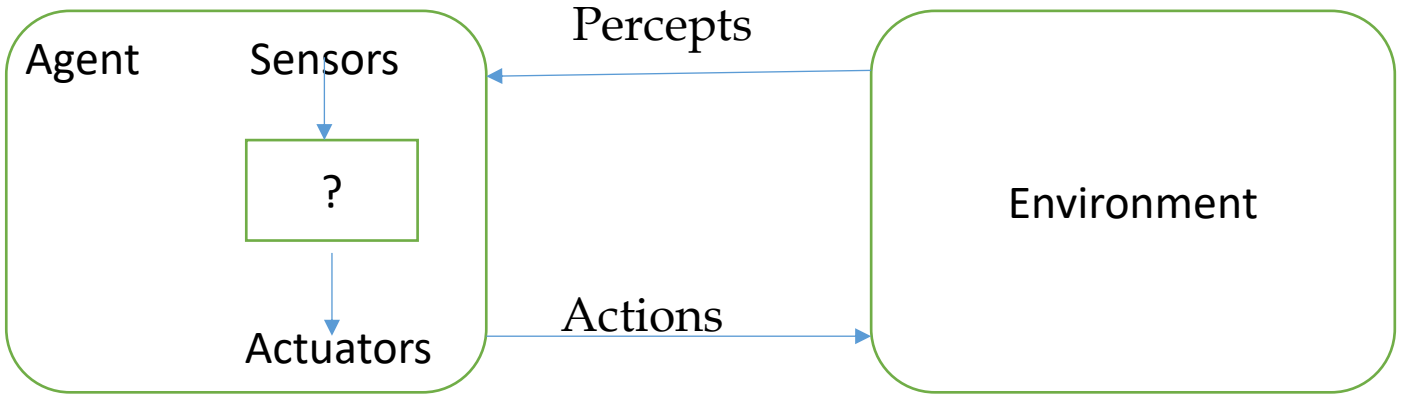
Agents

- Rational Agents – central to approach to AI
- Design Principles to build successful agents - systems that can be called reasonably intelligent
- Agents ← → Environment coupling
- Some agents behave better --- Rational Agents
- How well an Agent can behave? - depends on the nature of the environment – categorization of environments
- Properties of environment influence design of suitable agents for that environment
- Agent: An Agent is anything that can be viewed as perceiving it's environment through sensors and acting upon that environment through Actuators



Agents

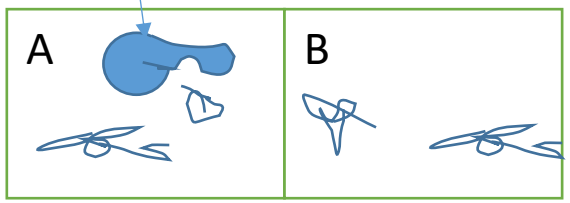
- Percepts:
  - Agent's perceptual inputs at any given instant
- Percept Sequence:
  - complete history of everything the agent has ever perceived
- Action:
  - Agents choice of action at any given instant can depend on the entire percept sequence observed to date, but not on anything it hasn't perceived
- Agent Function:
  - Specifying the agents choice of action for every possible percept sequence specifies agent - Agents behaviour - description that maps any given percept sequence to an action by the agent

Percept Sequence	Action
All possible sequences	Actions in response

- Could be very large - bound on percept sequences' length?

Agents

- Table of actions is external characterization of Agent
- Agent Program:
  - Internal implementation that is running within some physical system
- Vacuum Cleaner world with two locations



- Agents Function/ Agents Behaviour

- 2 locations squares A, B
- Vacuum agent perceives 1. which square it is in and 2. whether there is dirt in it
- It can choose to 1. Move Right, 2. Move Left, 3. Suck Up Dirt, 4. Do nothing

Percept Sequence	Action
[A, Clean]	Move Right
[A, Dirty]	Suck up Dirt
[B, Clean]	Move Left
[B, Dirty]	Suck up Dirt
[A, Clean] [A, Clean]	Move Right
.	
.	
[A, Clean] [A, Clean] [A, Clean]	Move Right
[A, Clean] [A, Clean] [A, Dirty]	Suck up Dirt

Agents – Agent Function

- Agent program for two state vacuum environment

```
Function Reflex – Vacuum-Agent ( [ location, Status])  returns an action

If    status = dirty then return suck up dirt
else   if   location = A then return turn Right
else   if   location = B then return  turn Left
```

- Agent Behaviour

Percept Sequence	Action
[A, Clean]	Move Right
[A, Dirty]	Suck up Dirt
[B, Clean]	Move Left
[B, Dirty]	Suck up Dirt
[A, Clean] [A, Clean]	Move Right
.	
.	
[A, Clean] [A, Clean] [A, Clean]	Move Right
[A, Clean] [A, Clean] [A, Dirty]	Suck up Dirt

Agents

- Percept sequence, Actions can be filled in various ways. What is the right way to fill the table?
- What makes an agent good or Bad?
- Rationality: good behaviour
- A Rational Agent is the one that does the right thing. What does it mean to do right thing?
- Consequences of the agents behaviour –
  - when an agent is in an environment, it generates a sequence of actions according to the percepts it receives
  - This sequence of actions causes the environment to go through a sequence of states
  - If the sequence of states is desirable, the agent has performed well.
- Performance Measure: notion of desirability
  - evaluates any given sequence of environment states
  - Needs to be suitable for the environment
- Vacuum Cleaner Environment - Performance measure?
  - Amount of dirt cleaned up in a single 8-hour shift?
  - Reward the agent for having a clean floor? One point for each clean square at each time step
  - (with a penalty for electricity consumed and noise generated)?

Agents

- Design performance measures according to what one actually wants in the environment, rather than according to how one thinks the agent should behave
- Issues: clean floor - average cleanliness over time
  - One cleans slowly all the time
  - One cleans fast and takes break
  - Which is preferable?
- Rationality: what is rational at any given instant depends on
  - The performance measure that defines the criterion of success
  - The agents prior knowledge of the environment
  - The actions that agent can perform
  - The agents 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
- Vacuum cleaner agent: cleans a square if it is dirty and moves to other square if not
- Performance measure: award one point for each clean square at each time step over a life time of 1000 time steps

Agents

- Vacuum cleaner agent: cleans a square if it is dirty and moves to other square if not – Rational?
- Performance measure:
  - award one point for each clean square at each time step over a life time of 1000 time steps
- Knowledge of the environment
  - Geography of the environment is known a priori
  - Dirt distribution and initial location of agent are not known
  - Move left, Move right actions move agent left and right except when this would take the agent outside the environment, in which case the agent remains where it is.
- Actions
  - The only actions are move left, move right, suck up dirt
- Percept sequence
  - The agent correctly perceives its location and whether that location contains dirt

Agents

- Different Circumstances
- Once all dirt is cleaned up, the agent will oscillate needlessly back and forth
- Including in performance measure a penalty of one point for each movement left or right – agent will perform poorly
- Better Agent for this case
  - would " do nothing" once it is sure that all squares are clean.
  - If clean squares become dirty again, the agent should occasionally check and re-clean them if needed.
  - If the geography of the environment is unknown, the agent will need to explore it rather than stick to squares A, B

Agents

- Omniscience
  - An Omniscience agent knows the actual outcome of the actions and can act accordingly
  - Not practical
  - Rationality is not same as perfection
  - Rationality maximizes expected performance
  - Perfection maximizes actual performance
- Information Gathering
  - **Rational Choice** depends only on the percept sequence to date.
  - Ensure not to allow agent to engage in under-intelligent activities
  - A rational agent should do actions to modify future percepts - Information gathering – important part of rationality

  Vacuum cleaning agent
  - Information gathering by exploration that must be undertaken by the agent in an initially unknown environment

  A Rational agent requires to gather information, learn as much as possible from what it perceives. The agent's initial configuration could reflect some prior knowledge of the environment, but as the agent gains experience, this may be modified/augmented.
- Autonomy
  - To the extent that an agent relies on the prior knowledge of its designer, rather than on its own percepts, the agent lacks Autonomy

Agents

- A rational Agent should be Autonomous
  - It should learn what it can to compensate for partial or incorrect prior knowledge

  A vacuum cleaning agent that learns to foresee where and when the additional dirt will happen will do better than one that does not
  - It is reasonable to give AI agent with some initial knowledge as well as ability to learn.
  - After sufficient experience of its environment, the behaviour of the agent can become effectively independent of its prior knowledge.
  - Incorporation of learning allows one to design a single rational agent that will succeed in a vast variety of environments.

- PAGE: Percepts, Acts, Goals, Environment - Setting for Intelligent Agent Design

Environments

- Task Environments
  - are problem spaces to which agents are solutions
  - Directly effects the appropriate design of the agents
  - How to specify task environment?
- Specification of Task environment: PEAS
  - Performance Measure
  - Environment
  - Actuators
  - Sensors

Agent	Performance Measure	Environment	Actuators	Sensors
Autonomous Taxi	Safe, fast, legal, comfortable trip	Roads, other traffic, Pedestrians, Customers	Steering, Accelerator, Brake, Signal, Horn, Display	Cameras, Sonar, Speedometer, GPS, Accelerometer, Engine sensors, Keyboard
Part Picking Robot	Percentage of parts in correct bins	Conveyor belt with parts, bins	Jointed arm and hand	Camera, Joint angle sensor

Environments

- Properties of Environment
  - Fully Observable Vs. Partially Observable
    - A task environment is effectively fully observable if the sensors detect all aspects that are relevant to choice of actions. Relevance depends on choice of performance measure
    - In a fully observable environment, agent need not maintain an internal state to keep track of the world
  - Partially Observable
    - Noisy, inaccurate sensors, parts of environment state missing from sensor data
    - Ex: vacuum agent with only a local sensor can not tell whether there is dirt in other squares
    - Ex: autonomous taxi can not see what other drivers are thinking
  - Unobservable
    - If the agent has no sensors

Environments

- Single agent Vs. Multiple Agent
  - Single agent
    - Agent solving crossword puzzle
  - Multiple agent
    - Agent playing chess - two agent
- Which entities must be viewed as agents? Or objects in environment?
  - Key distinction: whether B's behaviour is best described as maximizing a performance measure whose value is depends on A's behaviour
- Competitive multi agent Environment
  - Ex: Chess – opponent B is trying to maximize its performance measure, which , by the rules of chess, minimizes agent A's performance.
  - (randomized behaviour is rational sometimes as it avoids pitfalls of predictability. How?)
- Cooperative multi environment
  - Ex: Taxi driving environment – avoiding collisions maximizes the performance measure of all agents
  - (this environment is also partially competitive. Why?)

Environments

- Deterministic vs. Stochastic
  - Deterministic
    - If the next state of the environment is completely determined by the current state and the action executed by the agent, the environment is deterministic, else Stochastic
    - If an environment is fully observable, deterministic, agent need not worry about uncertainty.
    - Ex: vacuum world
  - Non Deterministic
    - An environment is uncertain if it is not fully observable, or non- deterministic
    - Actions are characterised by their possible outcomes, no probabilities are attached to the outcomes
    - Actions associated with performance measure require agent to succeed for all possible outcomes of its actions
  - Stochastic
    - Uncertainty about outcomes is characterised by their probabilities
    - Ex: autonomous Taxi driving

Environments

- Episodic vs. Sequential
  - Episodic
    - Agents experience is divided into episodes.
    - In each episode, the agent receives a percept, and then performs a single action
    - The next episode does not depend on the actions taken in previous episodes
    - Ex: classification tasks – an agent that has to spot defective parts on an assembly line bases each decision on the current part, regardless of previous decisions. The current decision does not effect whether the next part is defective
  - Sequential
    - Current decision could effect all future decisions
    - Ex: chess, taxi driving – short term actions can have long-term consequences
  - Episodic environments are much simpler to design than sequential environments. Why?

Environments

- Static vs. Dynamic
  - If the environment state can change while an agent is deliberating an action, then the environment is Dynamic, else Static.
  - Ex: Autonomous taxi driving is dynamic. Why?
  - Crossword puzzle is static
- Semi-dynamic
  - If the environment itself does not change with the passage of time, but the agent's performance score does change, then the environment is semi- dynamic
  - Ex: chess when played with a clock. Why?
- Static environments are easy to design for as agent need not keep looking at the world, while it is deciding on an action, nor it need to worry about the passage of time

Environments

- Discrete vs. Continuous
  - Applies to the state of the environment, to the percepts and agents actions
  - To the way time is handled
  - Ex: Chess – has a finite number of distinct states, has a discrete set of percepts and actions
  - Ex: Autonomous Taxi driving – is a continuous state and continuous time problem, taxi driving actions are continuous

- Known vs. Unknown
  - In a known environment, the outcomes or outcome probabilities for all actions are given.
  - Known environment can be partially observable
  - Ex: Solitaire card games.
  - If the environment is unknown, the agent will have to learn how it works in order to make good decisions.
  - Unknown environment can be fully observable

- Hardest case: partially observable, multi-agent, stochastic, sequential, dynamic, continuous and unknown!
- Ex: Taxi driving is hard in all these senses, except that for the most part, driver's environment is known

Environments

•

Environment/ specification	Observable	Agents	Deterministic /Stochastic	Episodic/ sequential	Static/ Dynamic	Discrete/ Continuous
Part picking Robot	Partially	single	Stochastic	Episodic	dynamic	Continuous
Autonomous Taxi Driving	Partially	multi	stochastic	sequential	dynamic	continuous
Crossword Puzzle	Fully	single	deterministic	sequential	static	discrete
Chess with a Clock	Fully	multi	deterministic	sequential	Semi-dynamic	discrete

Structure of Agents

- Agent Function – Agent Behaviour + Agent Program
- Agent Programs
- Trivial agent program

  Current percept/input from sensors → Agent program → returns an action to actuators

  remembrance of percepts, action - complete agent function

  A trivial agent program , that keeps track of the percept sequence and then uses it to index into a table of actions to decide what to do.

  Agent function is represented by the table and is implemented by the agent program
- To build a Trivial Table driven rational agent designer must construct a table that contains the appropriate action for every possible percept sequence

Table Driven Agent

function Table-Driven-Agent (percept) returns an action

persistent: **percepts,** a sequence, initially empty

table, a table of actions indexed by percept sequences, initially fully specified

append percept to the end of percepts

action ← Lookup (percepts, table)

return action

Table-driven-agent program is invoked for each new percept and returns action each time.

It retains the complete percept sequence in memory

P: possible set of percepts,

T: Life time of the agent

Lookup table will contain $\Sigma^{T}_{t=1} |P|^{t}$ entries.

Look up table for chess ~ 10^{150} entries

Table sizes are large.

Table Driven Agent

- Tabel-Driven-Agent implements desired agent Function
- Issues:
  - Storage space will be a problem for large Tables
  - Time required for designer to create the table huge
  - No agent could ever learn all the right table entries from its experience
  - No guidance as to how to fill the table even for simple environments
- Challenge:
  - How to write programs that to the extent possible, produce rational behaviour from a small program, rather than from a vast Table?
- (square root tables used by engineers are replaced by a program for Newton's method on electronic calculators.)
- Can AI do this for general intelligent behaviour?

Kinds of Agents

- Basic Kinds of Agents
  - Simple reflex Agents
  - Model-based Agents
  - Goal-based Agents
  - Utility–based Agents

Simple Reflex Agents

```
Function Reflex-Vacuum-Agent ([location, status])  return  action
        if status = dirty  then  return  Suck-dirt
        else if  location = A  then  return  Right
        else if  location = B   then  return  Left
```

Agent program for a simple reflex agent in the two state vacuum environment

Simple reflex agents select actions on the basis of the current percept, ignoring the rest of the percept history

Ex: vacuum agent's decision is based only on the current location and on whether that location contains dirt

Reduction from ignoring percept sequence, number of possible sequences cut down from 4^T → 4

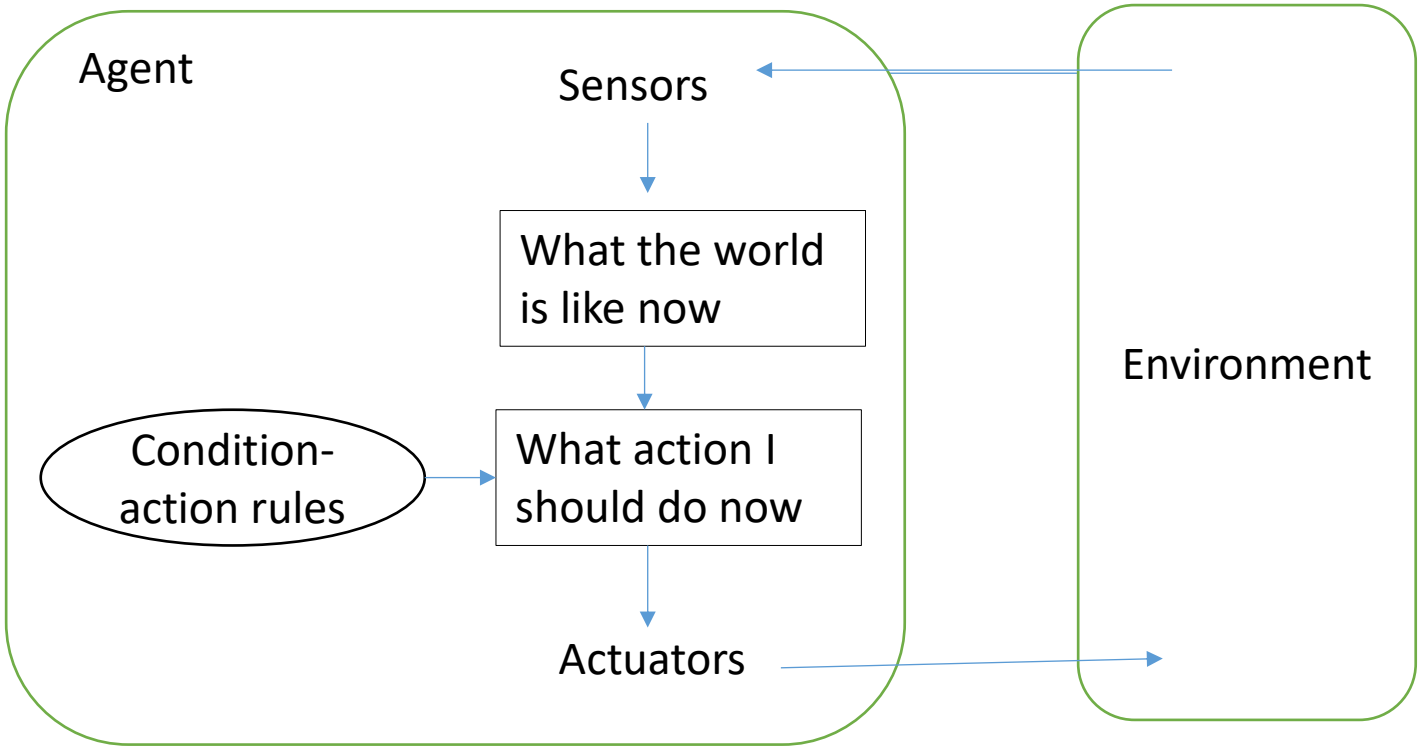
Simple reflex behaviours occur even in complex environments

Ex: if the brake lights of car in front come on, notice them and initiate braking

Condition – Action Rule?

Simple Reflex Agents

- Condition- Action Rule:
  - If car-in-front-is-braking then initiate-braking
- Build a general purpose interpreter for condition-action rule sets for specific task environment
- Condition-action rules allow the agent to make the connection from percept to action



A simple reflex agent.

- acts according to a rule whose condition matches the current state as defined by the percept

Simple
But limited intelligence

Simple Reflex Agents

Function Simple-Reflex-Agent (percept) return action

persistent: **rules**, a set of condition-action rules

State ← Interpret-Input (percept)

rule ← Rule-Match (state, rules)

action ← rule.action

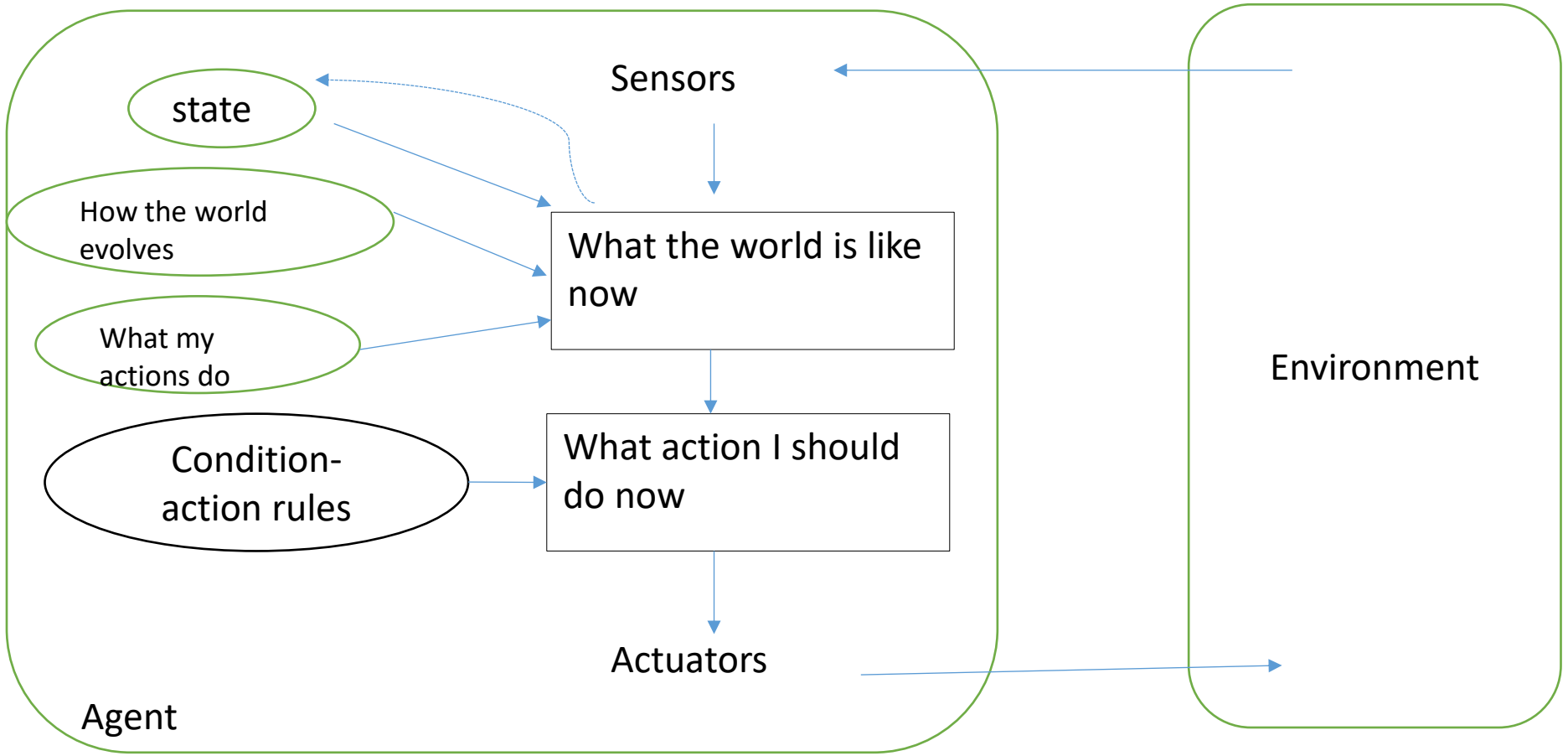
return action

- Simple reflex agent works only if the correct decision can be made on the basis of the current percept - that is only if the environment is fully observable
- Ex: autonomous taxi – single image of braking may not be sufficient, vacuum cleaner without location sensor can not move left or right
- In partially observable environments, randomized actions can avoid infinite loops - vacuum cleaner without location sensor can randomly move left or right, instead of failing in moving left or right when [clean] is seen.
- A randomized simple reflex agent might outperform a deterministic simple reflex agent

Model based Agents

- Agent should maintain some sort of **internal-state** that depends on the percept history reflecting at least some of the unobserved aspects of the current state.
  - Ex: for braking, previous image frames to detect brake lights going on-off simultaneously
- Updating internal state information as time goes by, requires two kinds of knowledge to be encoded in the agent Program - Model of the world
  - How the world evolves independent of the agent?
  - How agent actions affect the world?
- Model based agent: An agent that uses a model of the world
  - Current percept is combined with old internal state to generate the updated description of the current state based on the agent's model of 'how the world works'
- Uncertainty about the current state may be unavoidable, but the agent still has to take a decision

Model based Agents



Model based Agents

Function Model-Reflex--Agent (percept) return 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

Agent program for a model-based reflex agent –

keeps track of current state of the world, using an internal model.

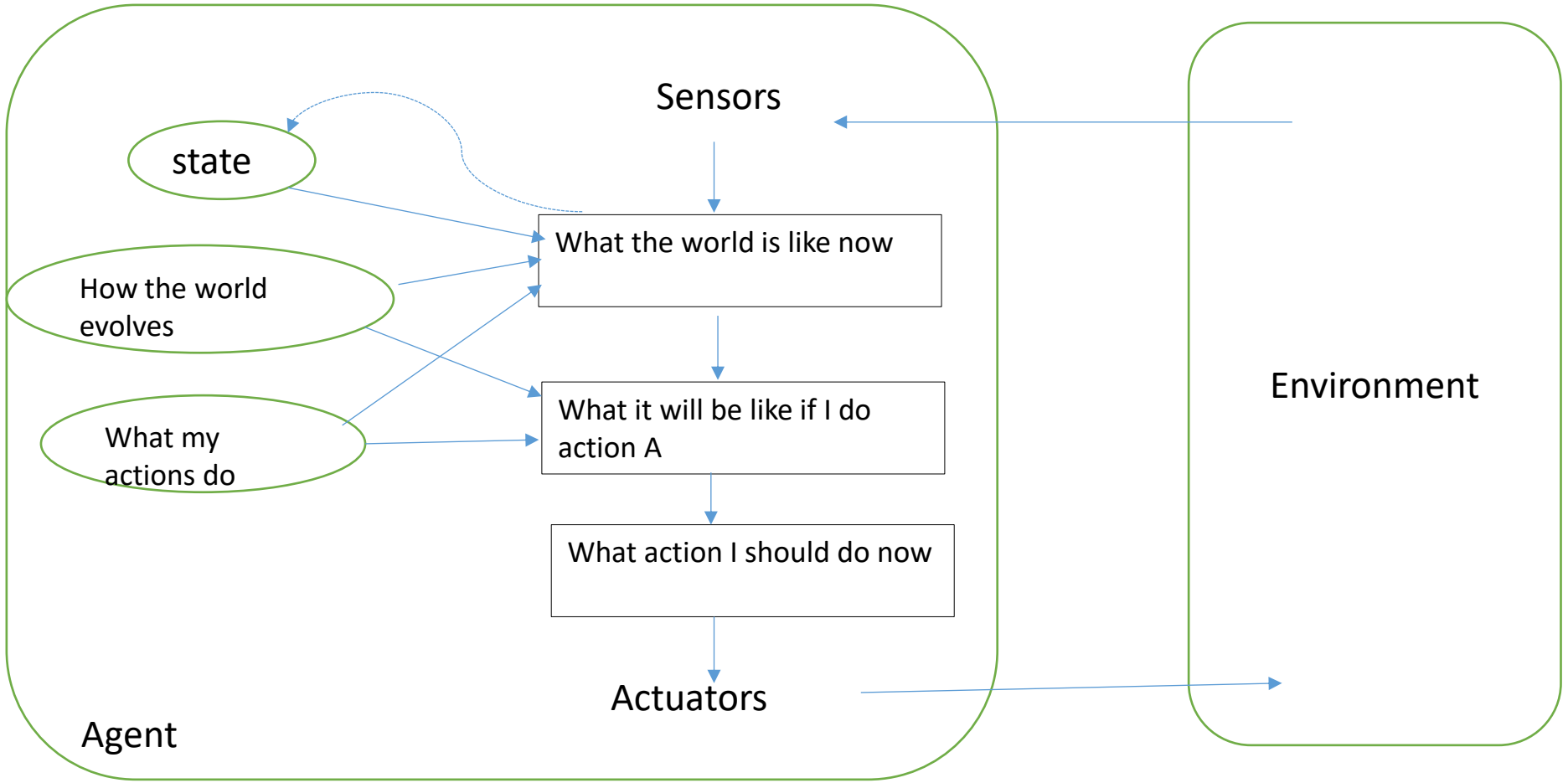
chooses the action the same way as the reflex agent

'What the world is like, now' represents agent's best guess

Goal based Agents

- Knowing about current state of the environment – not always enough to decide what to do
- The agent needs some sort of goal information that describes situations that are desirable
  - Choose actions that achieve the goal
  - Goal may be achievable through single action or a sequence of actions
- Finding actions to achieve the agent's goals can be through 'search and planning'
- A model based, goal based agent
  - Keeps track of the world state
  - A set of goals it is trying to achieve
  - Chooses an action that will lead to the achievement of its goals
- goal based decision making considers
  - 'the future – 'what will happen if I do this action'
    - ' will that make me happy/ reach goal'
- Different from decision making using 'condition-action' rules. In reflex agent designs this information is not explicitly represented. Built in rules map directly from percepts-to-actions

Goal based Agents



Goal based Agents

- Ex.
- Reflex agent brakes when it sees brake lights
- A goal based agent, in principle could reason that
  - If the car in front has brake lights on, it will slow down,
  - given the way the world usually evolves, the only action that will achieve the goal of not hitting other cars is to brake!
- Goal based looks less efficient? -

  It is more flexible – the knowledge that supports its decisions is represented explicitly and can be modified

  -If it starts to rain,
  - the agent can update its knowledge of how effectively its brakes will operate
  - and the knowledge automatically causes all the relevant behaviours to be altered to suit the new conditions

A reflex agent needs that the designer would have to re-write many condition –action rules to suit the new conditions

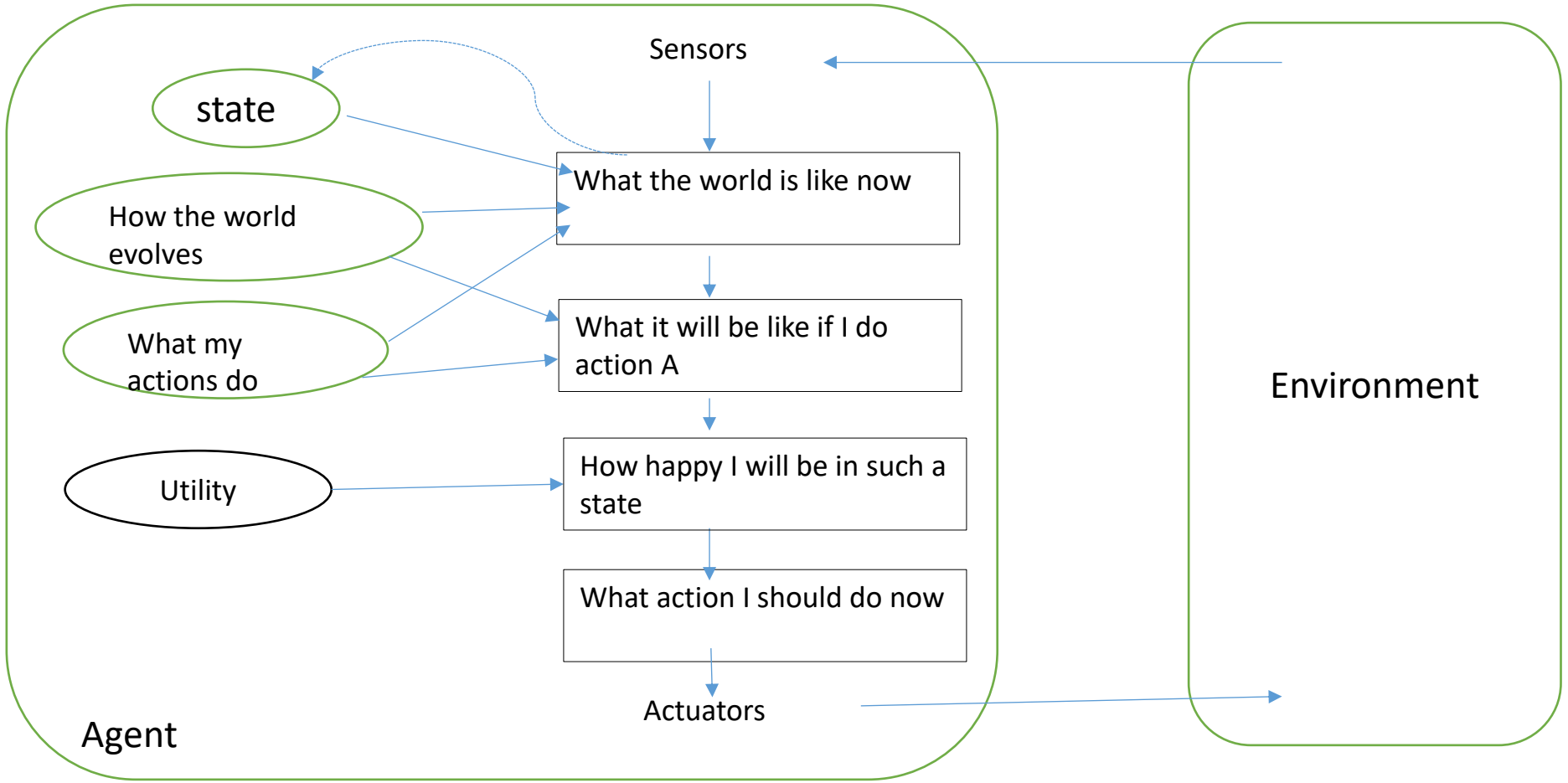
- In order to go to a different destination, goal based agent's behaviour can easily be changed, by simply specifying that destination as the goal.

The reflex agent's rules for when to turn left and when to go straight will work only for one destination. The need to be replaced to go somewhere new!

Utility- based Agents

- Goals are not enough for better behaviour in most environments
- Ex: many action sequences (paths) will get the taxi to destination, there by achieving the goal
  - Some are quicker, safer, more reliable or cheaper than others
- Performance measure is not just binary – reached destination or not
- Performance measure needs to allow comparison of different world states according to exactly
  - ‘ how happy they would make the agent - **utility** value maximization
- Performance measure can assign a score to any given sequence of environment states
  - To distinguish between more and less desirable ways of getting to the destination
- Utility Function:
  - an Agent’s utility function is essentially an internalization of the performance measure. If the internal utility function and the external performance measure are in agreement, then the agent that chooses actions to maximize its utility will rational according to the external performance measure
- (this is not the only way to be rational! A rational agent program for vacuum world has no idea what its utility function is)
- Both goal based and utility based agents have the advantage of flexibility and learning compared to reflex agents

Utility- based Agents



Utility- based Agents

- Goals are inadequate when
  - there are conflicting goals, only some of which can be achieved (ex: speed and safety)
  - There are several goals that the agent can aim for, none of which can be achieved with certainty
- Utility based agent can make rational decisions in both the above cases
  - Utility function specifies appropriate trade off for conflicting goals
  - Utility function provides a way in which the likelihood ness of success can be weighed against importance of the goals

"Partial Observability, Stochasticity are ubiquitous in the real world, and so, therefore, is decision making under uncertainty"

A rational utility based agent chooses the action that maximizes the expected utility of the action outcomes – the utility agent expects to derive, on average, given the probabilities and utilities of each outcome.

Utility based model agent has to model and keep track of its environment – requires perception, representation, reasoning, learning

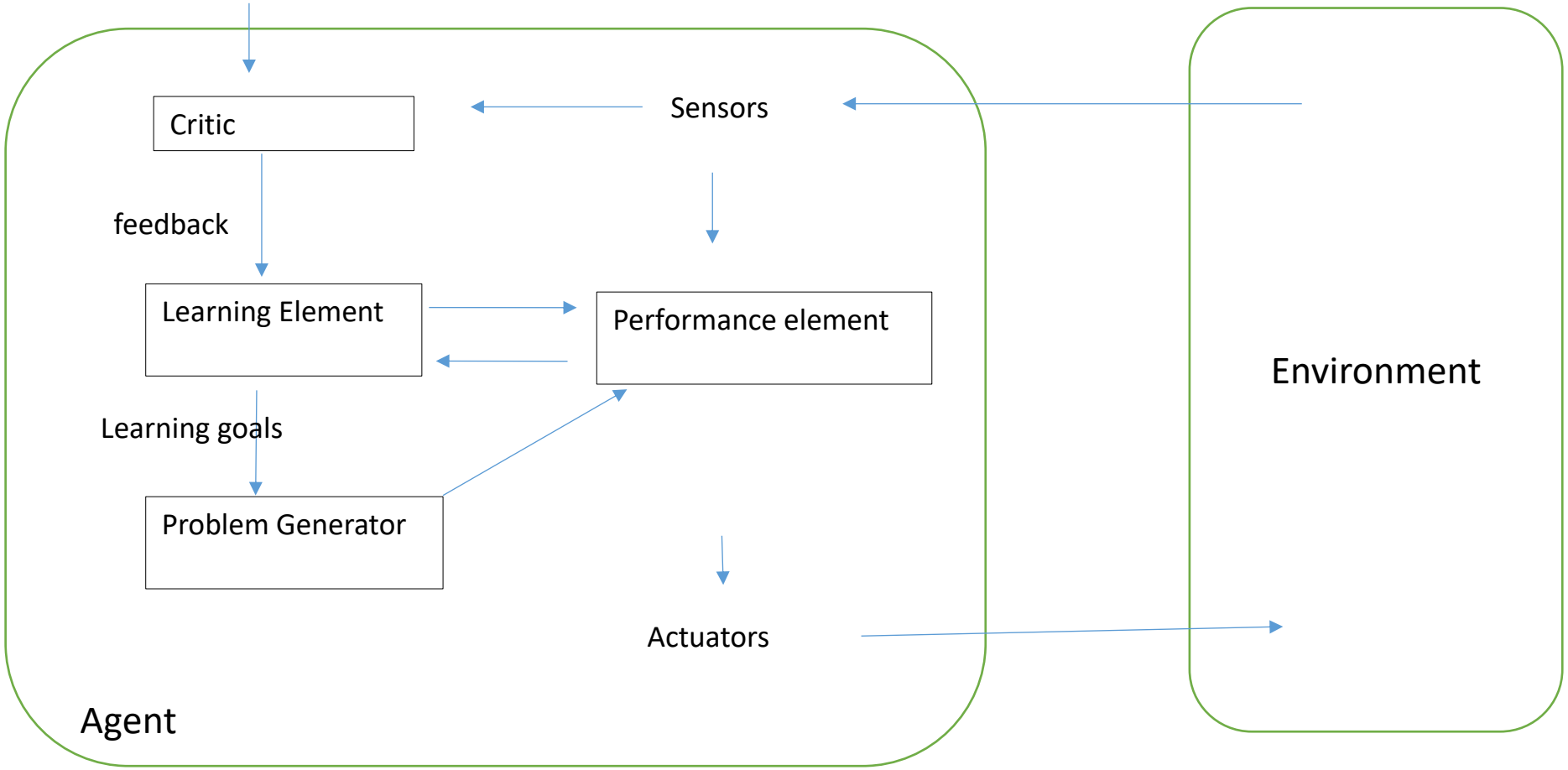
Choosing utility maximizing course of action is also a difficult task – requires ingenious algorithms

Learning Agents

- Build learning machines and teach them
- Allows agents to operate in initially unknown environments and to become more competent than its initial knowledge alone might allow.

Learning Agents

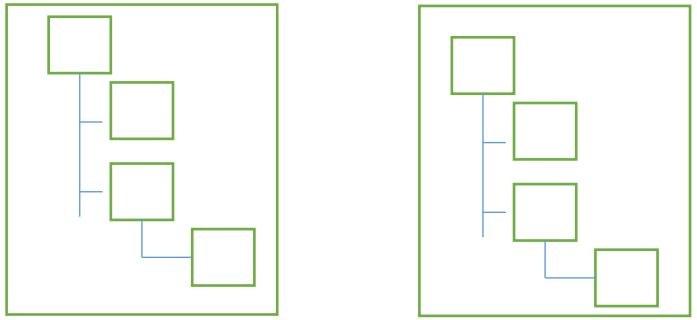
- Performance Standard



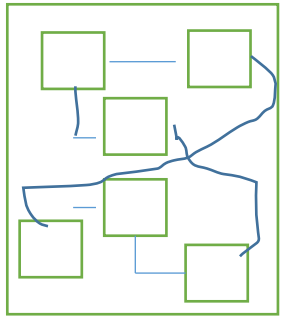
Agents - states

- Atomic

- Factored



- Structured



examples

Agent Type	Performance measure	Environment	Actuators	Sensors
Robot Soccer Player				
Internet book shopping agent				
Autonomous Rover				
Mathematician’s theorem-proving agent				

examples

Agent Type	Performance measure	Environment	Actuators	Sensors
Robot Soccer Player	Winning game, goals for/against	Field, ball, own team, other team, own body	Devices (e.g., legs) for locomotion and kicking	Camera, touch sensors, accelerometers, orientation sensors, wheel, joint encoders
Internet book shopping agent	Obtain requested/interesting books, minimize expenditure	Internet	Follow link, enter/submit data in fields, display to user	Web pages, user requests
Autonomous Rover	Terrain explored and reported, samples gathered and analysed	Launch vehicle, lander, Mars	Wheels/legs, sample collection device, analysis device, radio transmitter	Camera, touch sensors, accelerometers, orientation sensors,

Examples

Task environment	Observable	Deterministic	episodic	static	discrete	agents
Robot- soccer						
Internet book-shopping						
Autonomous Mars Rover						
Mathematician's assistant						

Examples

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-dynamic	Discrete	Multi

examples

- Implement an environment for *a n x m* rectangular room, where each square has a 5% chance of containing dirt, and *n* and *m* are chosen at random from the range 8 to 15, inclusive.
- Calculate the size of the table for a table-lookup agent. Explain your calculation. You need not fill in the entries for the table.