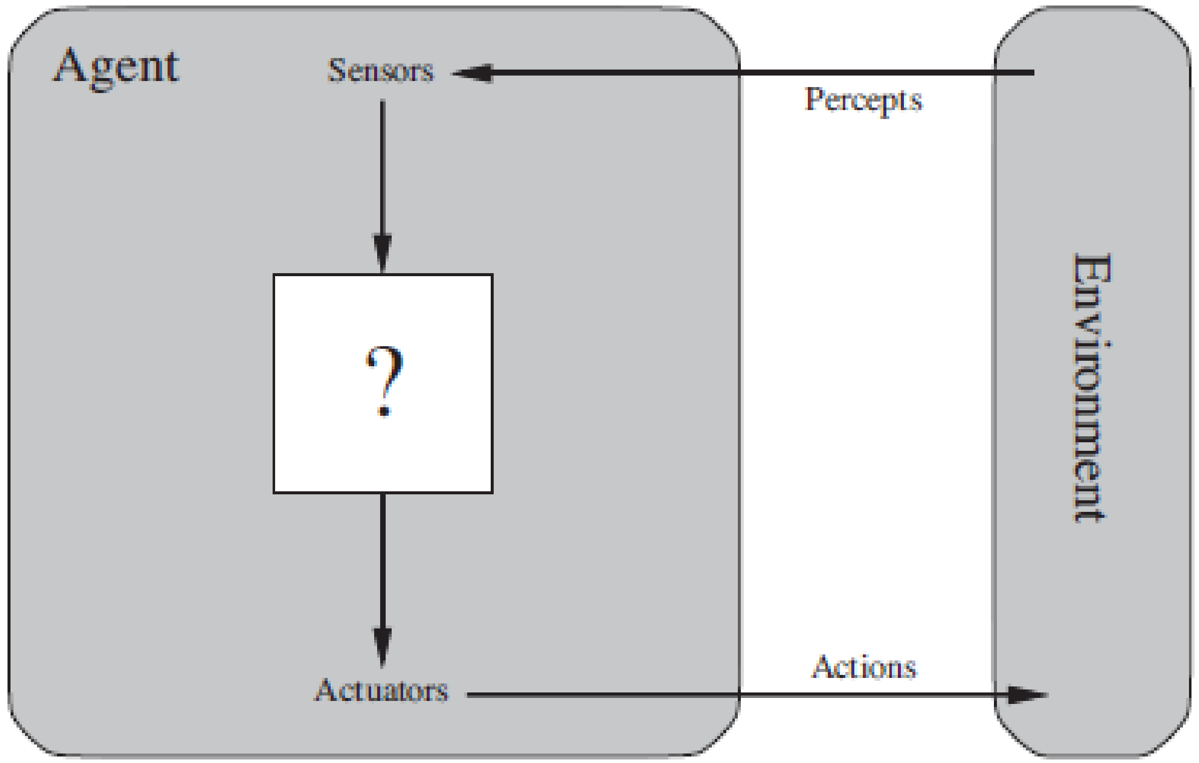
COMP338: **ARTIFICIAL INTELLIGENCE**

Agents

Dr. Radi Jarrar
Department of Computer Science

Agents

- An **Agent** is anything that perceives its **environment** through **sensors** and acts upon that environment through **actuators**



Agents

- The focus is to study ***rational agents***
- The rational behavior refers to doing the right thing; which maximises the expected performance measure given the available information fed to the agent

Thinking Humanly	Thinking Rationally
Acting Humanly	**Acting Rationally**

Agents

- Agents include humans, robots, software, ..., etc
- Human Agents:
  - Sensors: eyes, ears, skin, ears, ...
  - Actuators: hands, legs, vocal tracts,...

Agents

- Agents include humans, robots, software, …, etc
- Human Agents:
  - Sensors: eyes, ears, skin, ears, …
  - Actuators: hands, legs, vocal tracts,…
- Machine Agents:
  - Sensors: infrared, light-sensors, camera, …
  - Actuators: wheels, lights, speakers, …

Agents

- Agents include humans, robots, software, ..., etc
- Human Agents:
  - Sensors: eyes, ears, skin, ears, ...
  - Actuators: hands, legs, vocal tracts,…
- Machine Agents:
  - Sensors: infrared, light-sensors, camera, ...
  - Actuators: wheels, lights, speakers, ...
- Software Agents:
  - Sensors: keystrokes, file contents, ...
  - Actuators: display on the scree, write to files, ...

Agents

- Agents everywhere
- Thermostat
- Cell phone
- Vacuum cleaner
- Robots
- Self-driving car
- Human
- etc.

Agents and environments

- Percept: *the agent's perceptual inputs at any given instant*
- An agent's **percept sequence** is the complete history of everything the agent has ever perceived
- The agent's behaviour is described by the agent function
- The agent function maps from percept histories (percept sequence) to actions

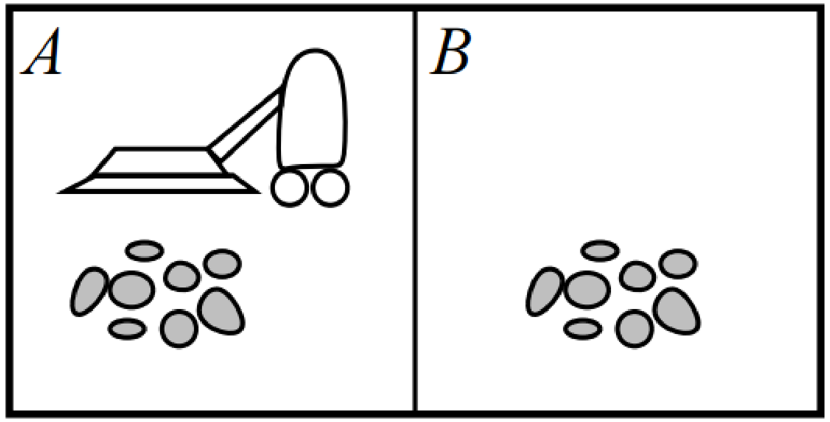
$$f : P \rightarrow A$$

Agents and environments

- The agent function for an artificial intelligent is implemented using **agent program**
- Agent program runs on physical **architecture** (platform) to produce *f*
- Agent = architecture + program

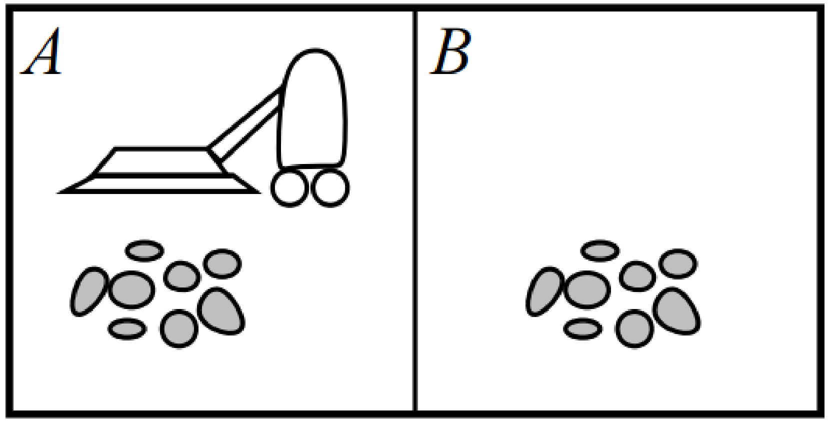
- *An agent is meant to be a tool for analyzing systems, not an absolute characterization that divides the world into agents and non-agents*

Example – The Vacuum-cleaner world



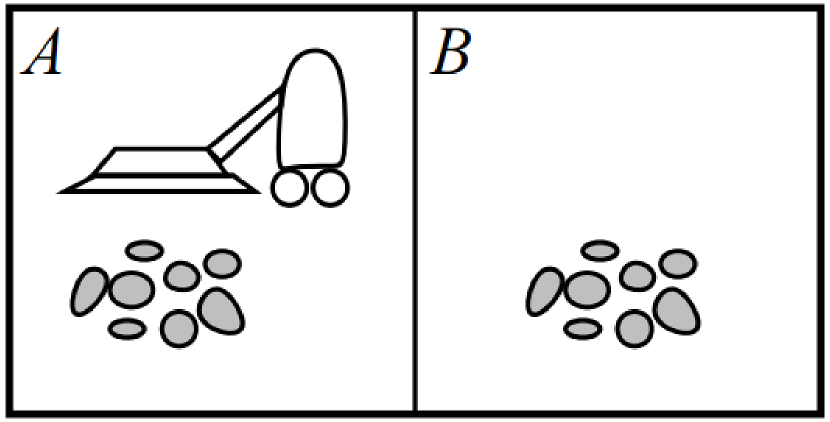
- There are two locations in this world: Square A & Square B

Example – The Vacuum-cleaner world



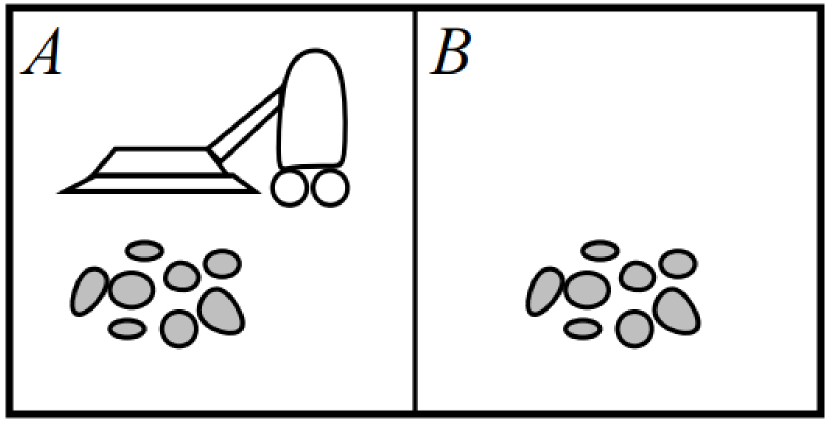
- There are two locations in this world: Square A & Square B
- The vacuum agent perceives which square it is in and whether there is dirt in the square

Example – The Vacuum-cleaner world



- There are two locations in this world: Square A & Square B
- The vacuum agent perceives which square it is in and whether there is dirt in the square
- Percepts: Location and Content (i.e., clean or dirty)

Example – The Vacuum-cleaner world



- There are two locations in this world: Square A & Square B
- The vacuum agent perceives which square it is in and whether there is dirt in the square
- Percepts: Location and Content (i.e., clean or dirty)
- Actions: Left, Right, Suck, NoOp
- Agent function: mapping from percepts to actions.

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

Example – The Vacuum-cleaner world

- The pseudo-code of this agent is:

```
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
```

- How do we know if this is a good agent function?
- In other words, what makes an agent good or bad, intelligent or stupid?
- What is the best function to implement this agent?
- Is there one? Who decides this?

The Concept of Rationality

- A **rational agent** is one that does the right thing
- *What does doing the right thing mean?*
- A Rational Agent selects the action that is expected to maximize its performance
  - based on a performance measure
  - depends on the percept sequence, background knowledge, and feasible actions
- Performance measure: An objective criterion for success of an agent's behaviour

The Concept of Rationality

- What is rational at any given time depends on four things
  - The performance measure that defines the criterion of success
  - The agent’s prior knowledge of the environment
  - The actions that the agent can perform
  - The agent’s percept sequence to date

The Concept of Rationality

- Fixed performance measure evaluates the environment sequence in the Vacuum Cleaner agent could be?

The Concept of Rationality

- Fixed performance measure evaluates the nvironment sequence in the Vacuum Cleaner agent could be:
  - Amount of dirt cleaned up? (+1 for each clean square at each time step)
  - Amount of time taken?
  - Amount of electricity consumed? (penalty?)
  - Amount of noise generated?
  - Number of moves to clean all squares?

Designing Performance Measure

- 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

Omniscience, learning, and autonomy

- A rational agent is not an omniscient
- It doesn’t know the actual outcome of its action
- Percepts may not supply all relevant information about the environment
- An omniscient agent knows the ***actual*** outcome of its actions and can act accordingly
- Omniscience is impossible in reality!
- Rational choice depends only on the percept sequence to date – doesn’t require omniscience

Omniscience, learning, and autonomy

- However, an agent should not engage in under-intelligent activities
  - E.g., agent crossing the street without looking both ways!
    1. It is not rational to cross the street given this uninformative percept sequence: risk of accident is high
    2. A rational agent should choose the “looking” action before crossing the road because it will help maximising the expected performance

Omniscience, learning, and autonomy

- Agents perform actions in order to modify future percepts so as to obtain useful information (**information gathering, exploration**)
- Information gathering is an important part of rationality
- Example of exploration: the vacuum-cleaning agent is put in an unknown environment

Omniscience, **learning**, and autonomy

- Agent should Learn from the gathered information
- Initial configurations reflect some prior knowledge of the environment
- Agent should adapt to changes in the environment
- No change – agent acts correctly

Omniscience, learning, and **autonomy**

- A rational agent should be **autonomous**
- It should learn what it can to compensate for partial or incorrect prior knowledge
- For example, a vacuum-cleaning agent that learns to foresee where and when additional dirt will appear will do better than one that does not
- After sufficient experience of its environment, the behaviour of a rational agent becomes independent from its prior knowledge

Rationality - revised

- An agent is autonomous if its behaviour is determined by its own percepts & experience (with ability to learn and adapt) without depending solely on build-in knowledge
- Rational ⇒ exploration, learning, autonomy
- *"The intuitive mind is a sacred gift and the rational mind is a faithful servant. We have created a society that honors the servant and has forgotten the gift.“ – Albert Einstein*

Task Environment

- Before designing a rational agent, we must specify the task environment
- Task environments are the **problems** to which rational agents are the **solutions**
- Task Environment (PEAS):
  - Performance Measure
  - Environment
  - Actuators
  - Sensors

Task Environment

- Example: Agent = Automated Taxi Driver
- Performance Measure

Task Environment

- Example: Agent = Automated Taxi Driver
- Performance Measure
  - Safe, fast, comfortable, legal, maximise profits, …
- Environment

Task Environment

- Example: Agent = Automated Taxi Driver
- Performance Measure
  - Safe, fast, comfortable, legal, maximise profits, …
- Environment
  - Streets, traffic, pedestrians, weather, …
- Actuators

Task Environment

- Example: Agent = Automated Taxi Driver
- Performance Measure
  - Safe, fast, comfortable, legal, maximise profits, …
- Environment
  - Streets, traffic, pedestrians, weather, …
- Actuators
  - Steering wheel, accelerators, brake, horn, signals,…
- Sensors

Task Environment

- Example: Agent = Automated Taxi Driver
- Performance Measure
  - Safe, fast, comfortable, legal, maximise profits, ...
- Environment
  - Streets, traffic, pedestrians, weather, ...
- Actuators
  - Steering wheel, accelerators, brake, horn, signals,...
- Sensors
  - Cameras, sonar, speedometer, GPS, gauges, engine sensors,...

Task Environment

- Example: Agent = Medical diagnosis system
- Performance Measure

Task Environment

- Example: Agent = Medical diagnosis system
- Performance Measure
  - Healthy patients, minimise costs, secure, private,…
- Environment

Task Environment

- Example: Agent = Medical diagnosis system
- Performance Measure
  - Healthy patients, minimise costs, secure, private,…
- Environment
  - Patients, hospitals, physicians, …
- Actuators

Task Environment

- Example: Agent = Medical diagnosis system
- Performance Measure
  - Healthy patients, minimise costs, secure, private,…
- Environment
  - Patients, hospitals, physicians, …
- Actuators
  - Screen that shows the questions, diagnosis, tests, treatments, referrals, …
- Sensors

Task Environment

- Example: Agent = Medical diagnosis system
- Performance Measure
  - Healthy patients, minimise costs, secure, private,…
- Environment
  - Patients, hospitals, physicians, …
- Actuators
  - Screen that shows the questions, diagnosis, tests, treatments, referrals, …
- Sensors
  - Keyboard to enter the patient’s answers, symptoms, findings, …

Task Environment

- Example: Agent = Internet Shopping Agent
- Performance Measure

Task Environment

- Example: Agent = Internet Shopping Agent
- Performance Measure
  - Price, quality, credibility, …
- Environment

Task Environment

- Example: Agent = Internet Shopping Agent
- Performance Measure
  - Price, quality, credibility, …
- Environment
  - Web-sites, vendors, shippers, …
- Actuators

Task Environment

- Example: Agent = Internet Shopping Agent
- Performance Measure
  - Price, quality, credibility, …
- Environment
  - Web-sites, vendors, shippers, …
- Actuators
  - Display, forms to fill in the order,…
- Sensors

Task Environment

- Example: Agent = Internet Shopping Agent
- Performance Measure
  - Price, quality, credibility, …
- Environment
  - Web-sites, vendors, shippers, …
- Actuators
  - Display, forms to fill in the order,…
- Sensors
  - HTML pages (text, graphics, scripts, …)

Task Environment

- Example: Agent = Question Answering System
- Performance Measure

Task Environment

- Example: Agent = Question Answering System
- Performance Measure
  - User satisfaction, known questions…
- Environment

Task Environment

- Example: Agent = Question Answering System
- Performance Measure
  - User satisfaction, known questions…
- Environment
  - Wikipedia, Wolfarm alpha, encyclopedia…
- Actuators

Task Environment

- Example: Agent = Question Answering System
- Performance Measure
  - User satisfaction, known questions…
- Environment
  - Wikipedia, Wolfarm Alpha, encyclopaedia…
- Actuators
  - Spoken/written language
- Sensors

Task Environment

- Example: Agent = Question Answering System
- Performance Measure
  - User satisfaction, known questions…
- Environment
  - Wikipedia, Wolfarm Alpha, encyclopaedia…
- Actuators
  - Spoken/written language
- Sensors
  - Spoken/written input

Environment types

- Fully observable (vs. Partially observable)
  - An agent’s sensors give it access to the complete state of the environment at each point in time
  - A task environment is fully observable if the sensors detect all aspects that are relevant to the choice of action (relevance, depends on the performance measure)

Environment types

- Fully observable (vs. Partially observable)
  - 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

Environment types

- Single agent (vs. multi-agent)
  - An agent operating by itself in an environment
  - E.g., An agent to solve crossword puzzle vs. agent that plays chess
  - Does an agent A (taxi driver) have to treat another object B (a vehicle) as another agent or as another object?

Environment types

- Single agent (vs. multi-agent)
  - An agent operating by itself in an environment
  - E.g., An agent to solve crossword puzzle vs. agent that plays chess
  - Does an agent A (taxi driver) have to treat another object B (a vehicle) as another agent or as another object?
  - The key distinction is whether B's behaviour is best described as maximizing a performance measure whose value depends on agent A's behaviour

Environment types

- For example, in chess, the opponent entity B is trying to maximize its performance measure, which, by the rules of chess, minimizes agent A's performance measure
- Thus, chess is a **competitive** multiagent environment
- In the taxi-driving environment, on the other hand, avoiding collisions maximizes the performance measure of all agents, so it is a partially **cooperative** multiagent environment
-

Environment types

- Deterministic (vs. stochastic): The next state of the environment is completely determined by the current state and the action executed by the agent
- Otherwise, it is stochastic
- If the environment is deterministic except for the actions of other agents, then the environment is *strategic*
- If the environment is partially observable, then it could *appear* to be stochastic

Environment types

- Most real situations are complex that it is impossible to keep track of all the unobserved aspects. So for practical purposes, they must be treated as stochastic
- Taxi driver? Vacuum world?
- An environment is **uncertain** if it is not fully observable or not deterministic
- Stochastic implies that uncertainty about outcomes is quantified in terms of probabilities
- A **nondeterministic** environment is one in which actions are characterized by their *possible* outcomes

Environment types

- Episodic (vs. sequential): An agent's action is divided into atomic episodes
- In each episode the agent receives a percept and performs an action
- Decisions of next episodes do not depend on previous decisions/actions

Environment types

- Many classification tasks are considered episodic
- In sequential environments the current decision could affect all future decisions
- Chess and taxi driving are sequential (short-term actions can have long-term consequences)

Environment types

- Static (vs. dynamic): The environment is unchanged while an agent is deliberating
- Dynamic environments are continuously asking the agent what it wants to do
- The environment is **semidynamic** if the environment itself does not change with the passage of time but the agent's performance score does
- Taxi driving? Chess? Crossword?

Environment types

- Discrete (vs. continuous): The discrete/continuous distinction applies to the *state* of the environment, to the way *time* is handled, and to the *percepts* and *actions* of the agent
- Chess has a discrete set of percepts and actions. Its eenvironment has a finite number of distinct states
- Taxi driving is a continuous-state and continuous-time problem
- Taxi-driving actions are also continuous (steering angles

Environment types

Task Env.	Observable	Agents	Determ.	Episodic	Static	Discrete
Crossword Puzzle						
Chess						
Poker						
Backgammon						
Taxi driving						
Medical Diagnosis						
Image analysis						
Interactive English Tutor						

Environment types

Task Env.	Observable	Agents	Determ.	Episodic	Static	Discrete
Crossword Puzzle	Fully	Single	Determ.	Sequential	Static	Discrete
Chess	Fully	Multi	Determ.	Sequential	Semi	Discrete
Poker	Partially	Multi	Stoch.	Sequential	Static	Discrete
Backgammon	Fully	Multi	Stoch.	Sequential	Static	Discrete
Taxi driving	Partially	Multi	Stoch.	Sequential	Dynam.	Cont.
Medical Diagnosis	Partially	Single	Stoch.	Sequantial	Dynam.	Cont.
Image analysis	Partially	Single	Stoch.	Sequential	Dynam.	Cont.
Interactive English Tutor	Partially	Multi	Stoch.	Seuqntial	Dynam.	Discrete

Types of Agents

- Six basic types in order of increasing generality:

1. Table Driven Agents
2. Simple Reflex Agents
3. Model-Based Reflex Agents
4. Goal-Based Agents
5. Utility-Based Agents
6. Learning Agents

Table Driven Agents

- Table-driven-agent program is invoked for each new percept and returns an action each time
- It retains the complete percept sequence in memory
- When the agent program is designed, there is a table that contains the appropriate action for each possible percept sequence

Table Driven Agents

```
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
```

- This program is invoked for each new percept and returns an action each time

Table Driven Agents

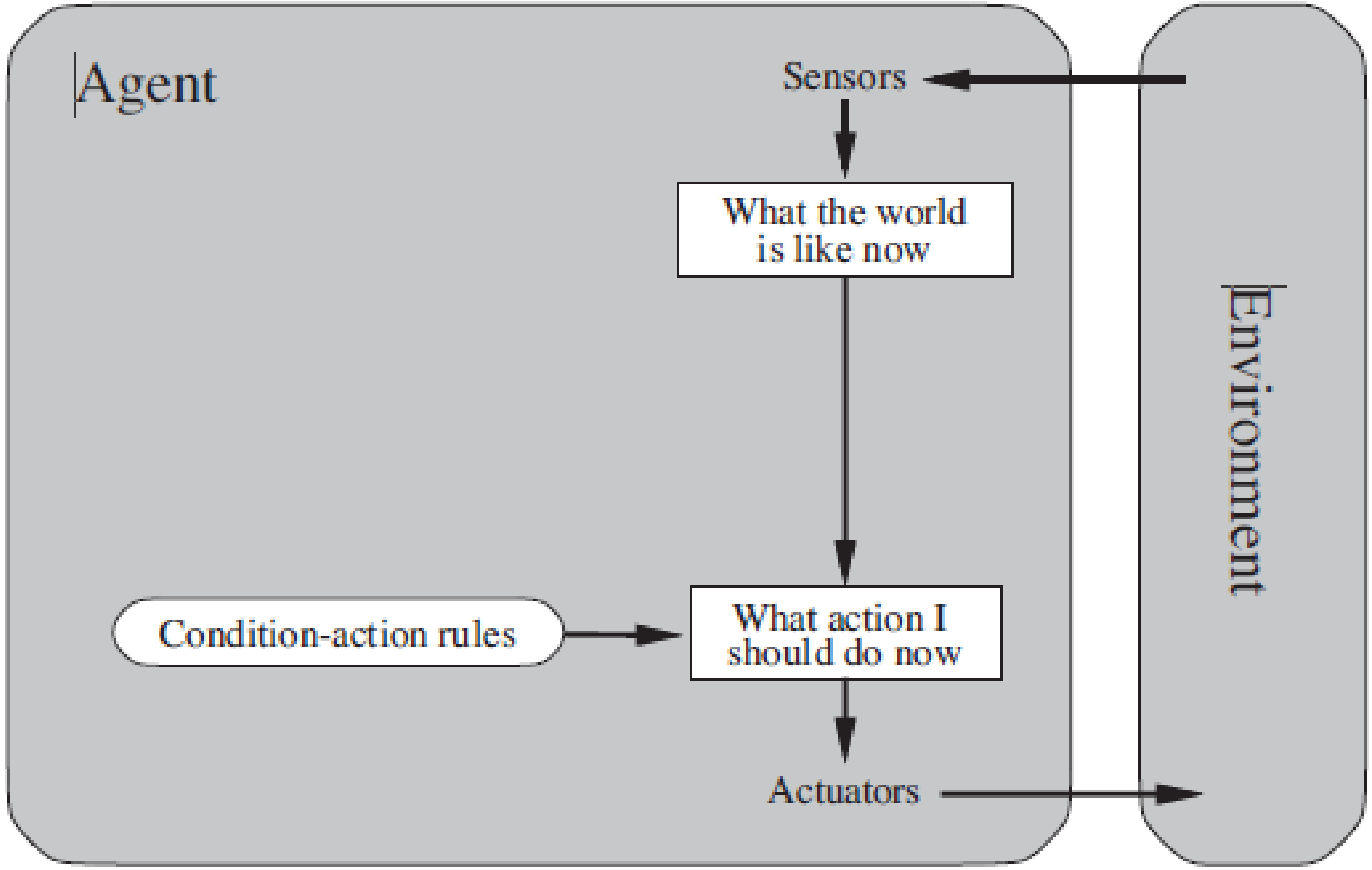
- Failed!
  - The size of the table (very large)
  - No agent could ever learn all the right table entries from its experience
  - No autonomy – all actions are predetermined
  - Even if the environment is simple enough to yield a feasible table size, the designer still has no guidance about how to fill in the table entries
  - IMPRACTICAL

Simple Reflex Agents

- Associations between the input/output are recorded
- Selects action based on current percept (ignoring the rest of percept history)
- Condition-Action rule (if-then rule): a method of specification between the input (percept) and the output (actuator) *(e.g., if percept then action)*
- Works only if the correct decision can be made on the basis of only the current percept—that is, only if the environment is fully observable

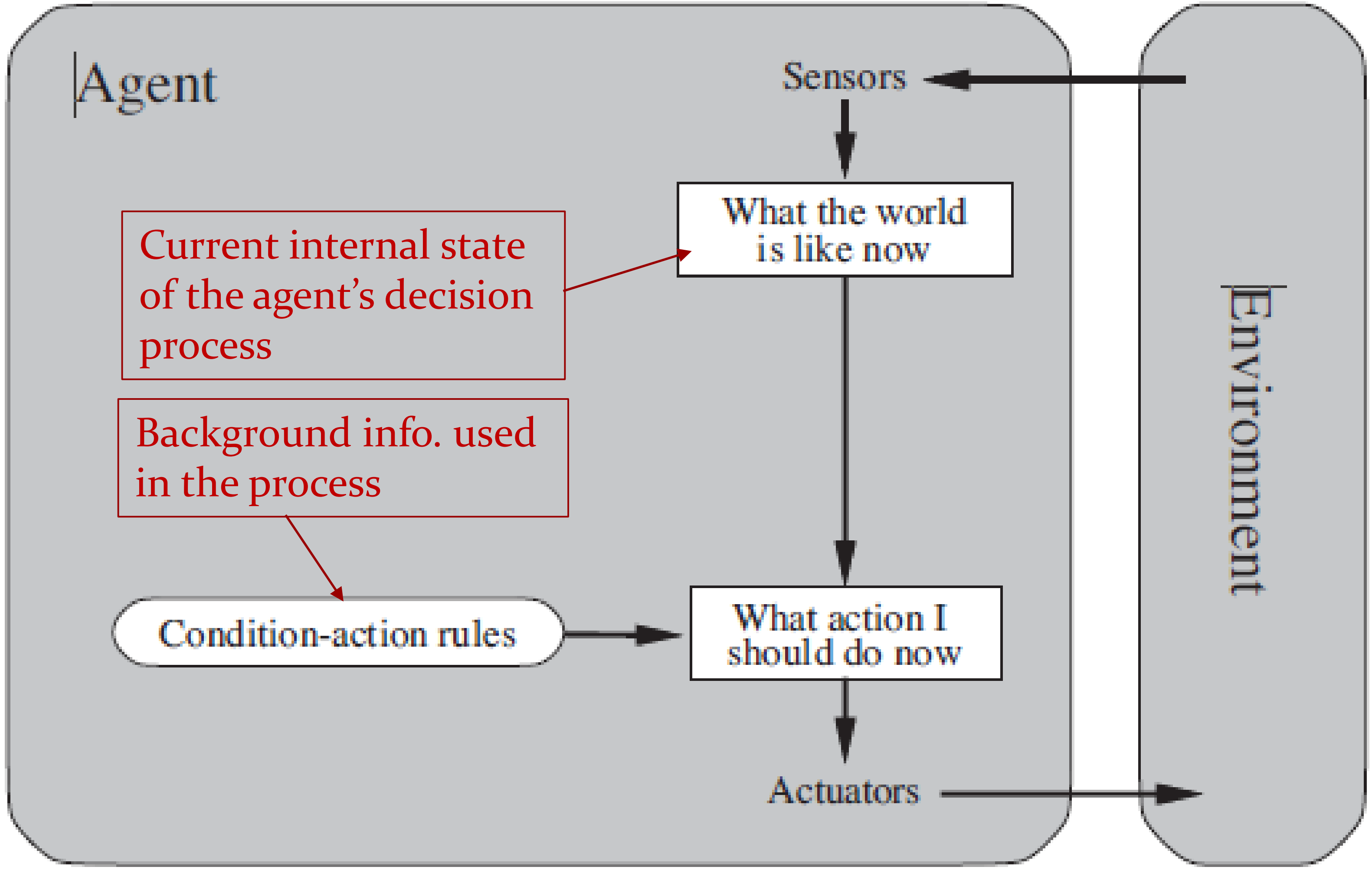
Simple Reflex Agents

- Associa
- Selects percept history
- Conditi n betwee *f percept then ac*
- Easy to



Simple Reflex Agents

- Associa
- Selects percept history
- Conditi n betwee *f percept then ac*
- Easy to


Agent
Sensors
What the world is like now
Current internal state of the agent's decision process
Background info. used in the process
Condition-action rules
What action I should do now
Actuators
Environment

Simple Reflex Agents

```
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
```

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

Simple Reflex Agents

- Easy to implement and simple but limited in intelligence
- Fast but too simple
- No memory—Fails if the environment is partially observable
- *What if the vacuum agent has only the dirt sensor?*

Simple Reflex Agents

- Easy to implement and simple but limited in intelligence
- Fast but too simple
- No memory—Fails if the environment is partially observable
- *What if the vacuum agent has only the dirt sensor?*
  - *Agents goes into infinite loop!*
  - *Avoidable through Randomisation of actions*
  - For example, if the vacuum agent perceives [Clean], it might flip a coin to choose between Left and Right

Model-Based Reflex Agents

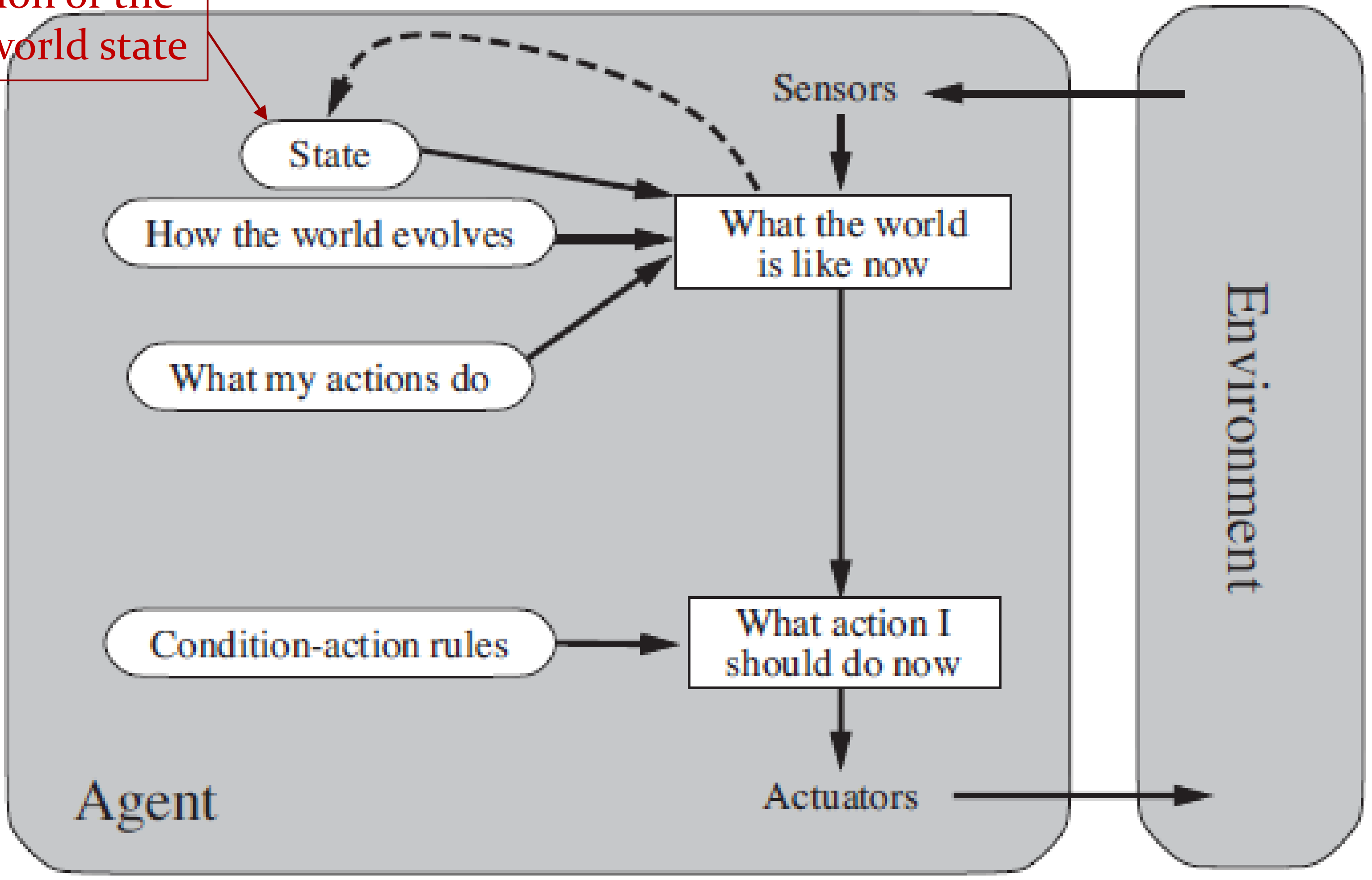
- Model-Based Reflex Agents deal with Partially Observable environments by keeping track of the part of the world it can see now
- A model-based reflex agent keeps track of the current state of the world using an internal model
- It then chooses an action in the same way as the reflex agent

Model-Based Reflex Agents

- The internal states of the model reflect the knowledge of this agent about the world
- It models the state of the world by
  - Modelling how the world changes
  - How it’s actions change the world

Model-Based Reflex Agents

Description of the current world state



Model-Based Reflex Agents

```
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
```

- A model-based reflex agent keeps track of the current state of the world, using an internal model. It then chooses an action in the same way as the reflex agent.

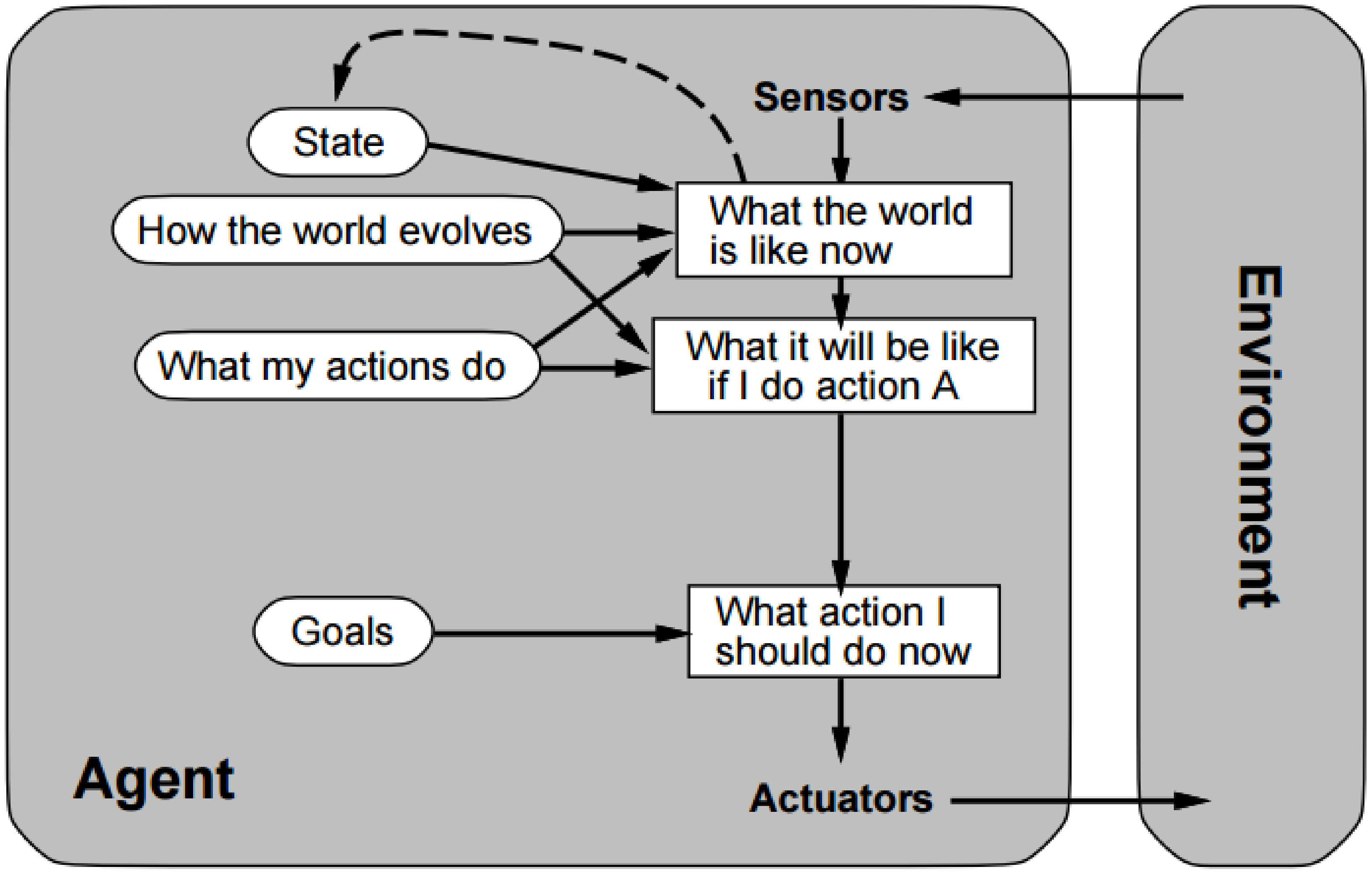
Model-Based Reflex Agents

- It’s is unclear what to do without a clear **goal**

Goal-Based Agents

- Goal-Based Agents aim to reach a desirable state (Goal)
- The goal might be provided from the designer, users, environment,…
- Actions of the agent are considered with respect to the goal
- Goals provide reason to prefer one action over the other
- We need to predict the future: we need to plan & search

Goal-Based Agents



Sensors
State
How the world evolves
What my actions do
What the world is like now
What it will be like if I do action A
Goals
What action I should do now
Environment
Agent
Actuators

Utility-Based Agents

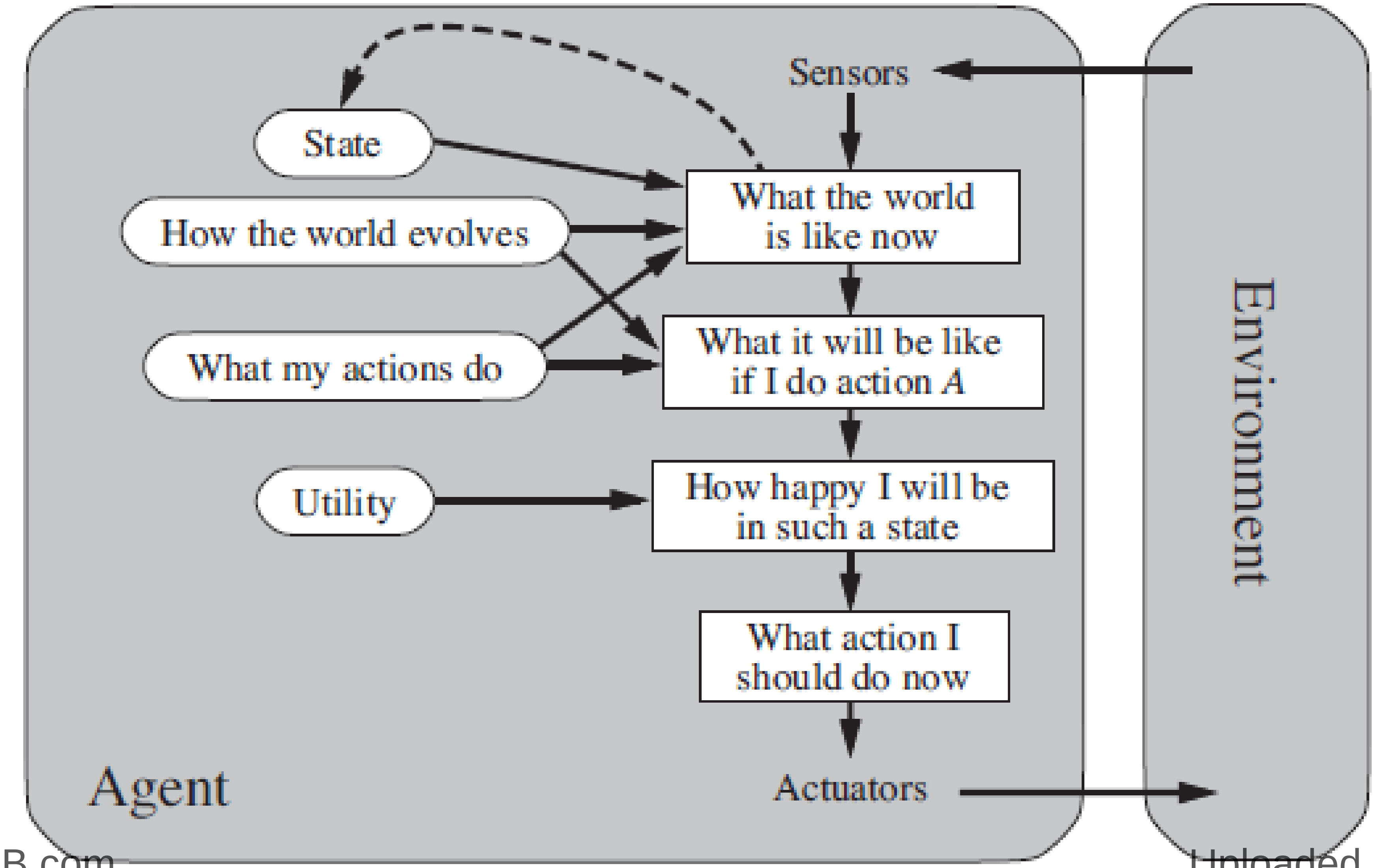
- Goals alone are not enough to generate high-quality behaviour in most environments
- For example, many action sequences will get the taxi to its destination (thereby achieving the goal) but some are quicker, safer, more reliable, or cheaper than others
- Goals just provide a distinction between “happy” and “unhappy” states

Utility-Based Agents

- Utility functions map states onto a real number that may be interpreted as the degree of happiness
- It allows rational actions for more complex tasks
  - Resolutions of conflicts between goals (tradeoffs)
  - Multiple goals
- A utility function is necessary for rational behaviour

Utility-Based Agents

- Utility y be interp
- It allov
  - Resolu
  - Multi
- A utilit


Sensors
State
How the world evolves
What my actions do
Utility
What the world is like now
What it will be like if I do action A
How happy I will be in such a state
What action I should do now
Agent
Actuators
Environment

Utility-Based Agents

- 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 an agent that chooses actions to maximize its utility will be rational according to the external performance measure
- A rational utility-based agent chooses the action that maximizes the **expected utility** of the action outcomes

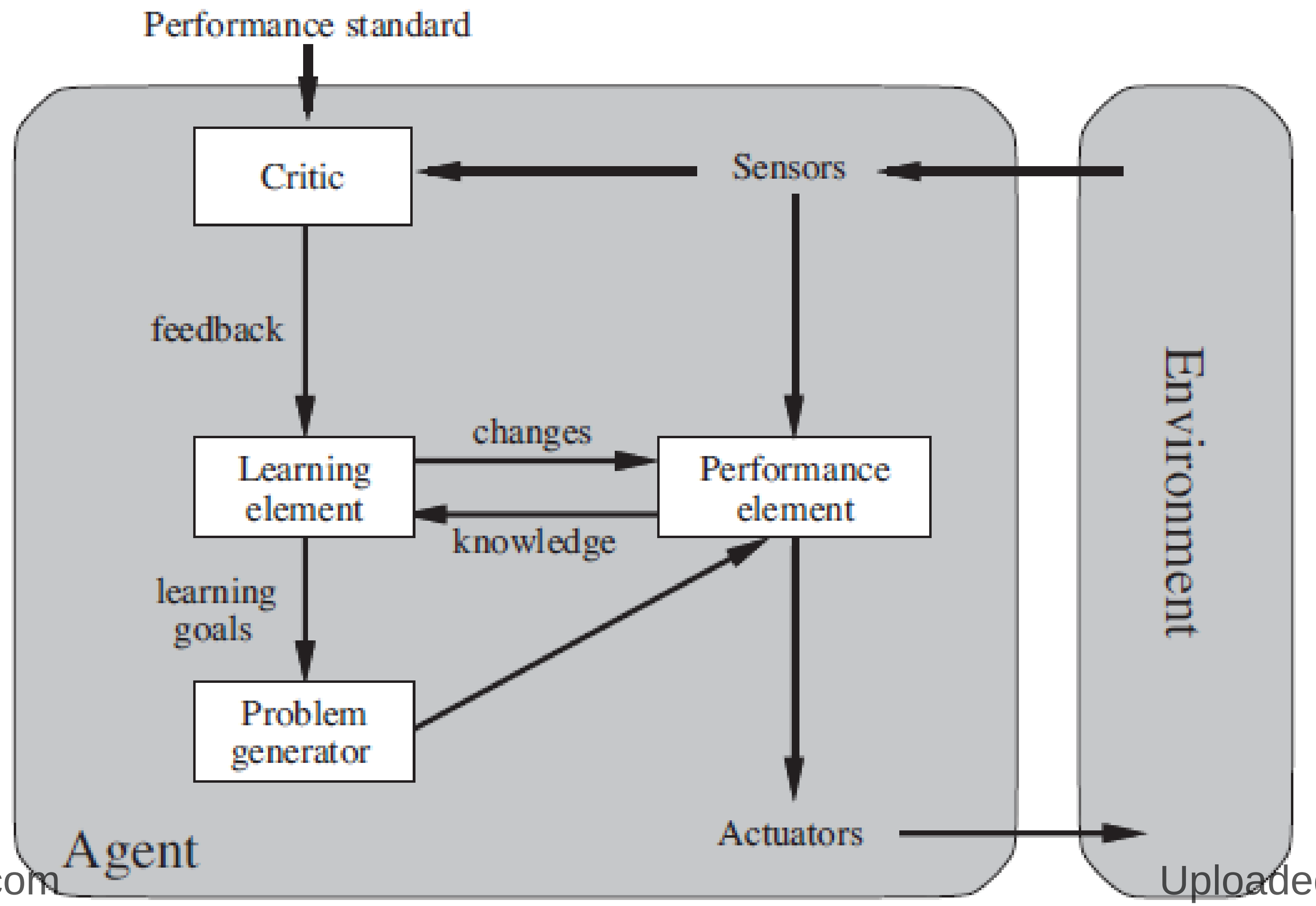
Learning Agents

- How does an agent improve over time? By monitoring it's performance and suggesting better modelling, new action rules,
- A Learning agent is divided into four components
  - Learning element: Changes action rules to make improvements
  - Performance element: Responsible to make external actions. Actions are made based on percepts, internal states, and background knowledge (i.e., it can be one of the aforementioned agents)

Learning Agents

- Critic:
  - Evaluates the current state of the world
  - The critic tells the learning element how well the agent is doing with respect to a fixed performance standard
- Problem Generator:
  - Suggests actions that will lead to new and informative experiences
  - Promotes the agent to explore what may improve its performance

Learning Agents


Performance standard
Critic
Sensors
feedback
changes
Learning element
Performance element
knowledge
learning goals
Problem generator
Actuators
Agent
Environment