Agents, States, and Environments

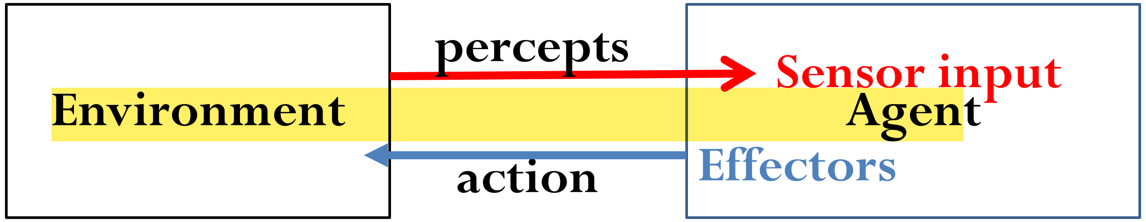
Dr. Animesh Chaturvedi
Assistant Professor: IIIT Dharwad
Young Researcher Alumni: Heidelberg Laureate Forum
Postdoc: King's College London & The Alan Turing Institute
PhD: IIT Indore MTech: IIITDM Jabalpur

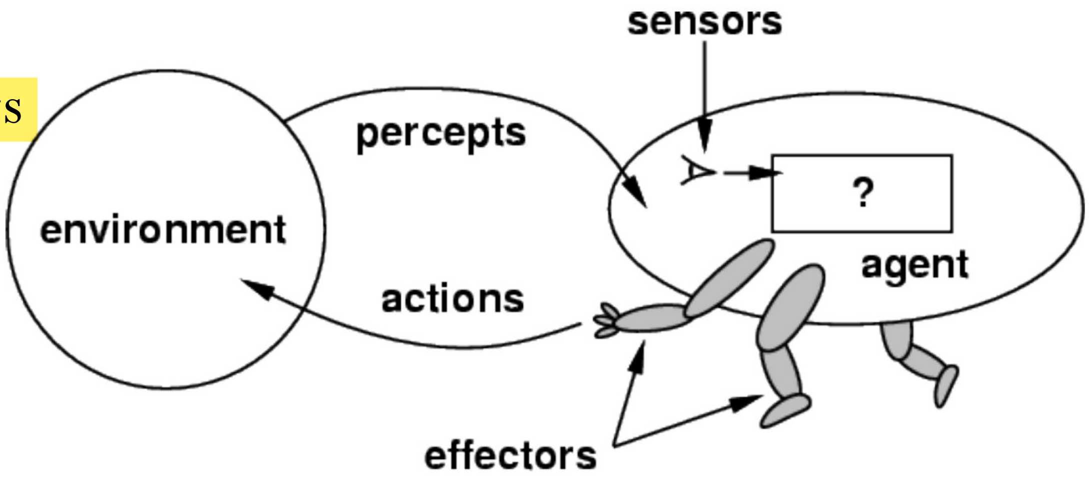
Agents, States, and Environments

- Agents and Environments
  - Rational agent
  - Intelligent agent
  - Autonomous Agents
  - Omniscience agents
  - PEAS (Performance measure, Environment, Actuators, Sensors)
- Environment types
- Agent types

Agents

- An agent is anything that can be viewed as perceiving its environment through sensors and acting upon that environment through actuators
  - Agent action is decided upon any input perceived by any agent.
  - Agent program is the action taken against that percept sequence
- 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

- The agent view is really quite generic.
- In a sense, all areas of engineering can be seen as designing artifacts that interact with the world.
- AI operates on that end of the spectrum, where the artifacts use significant computational resources and the task and environment requires non-trivial decision making.
- The definition of "agents" does technically also include, e.g., calculators or cars, artifacts with very limited to no intelligence.
- Agents can perform actions in order to modify future percepts so as to obtain useful information: information gathering and exploration.

Agent functions and programs

- An agent is completely specified by the agent function mapping percept sequences to actions
- The agent function maps from percept histories to actions:

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

- One agent function (or a small equivalence class) is rational
- The agent program runs on the physical architecture to produce f

agent = architecture + program

- Aim: find a way to implement the rational agent function concisely
- Design an agent program assuming an architecture that will make the percepts from the sensors available to the program.

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
- For each possible percept sequence, a rational agent should select an action that maximizes its performance measure (in expectation) given the evidence provided by the percept sequence and whatever built- in knowledge the agent has.
- "Expectation": Captures actions with stochastic / uncertain effects or actions performed in stochastic environments. We can then look at the expected value of an action.
- Rational is different from being perfect
  - Rationality maximizes expected outcome while perfection maximizes actual outcome.
  - In high-risk settings, we may also want to minimize the worst-case behavior.
  - We can behave rationally even when faced with incomplete information.

Rational agents VS Omniscience agents

- Omniscience means all-knowing with infinite knowledge.
- Rationality
  - Performance measuring success
  - Agents prior knowledge of environment
  - Actions that agent can perform
  - Agent’s percept sequence to date
- Rationality is different from omniscience (“all knowing”).
  - Percepts may not supply all relevant information
  - E.g., in card games, you don’t know the cards of others.

Intelligent agents

- Sufficiently complex rational agents can be viewed as "intelligent agents."
- Self-driving cars come closer to what we view as intelligent agents.
- Intelligent agent-view provides a framework to integrate the many subareas of AI.
- This is somewhat high-level and abstract.
- Much of the technical framework of how intelligent agents are actually built.

Autonomous Agents

- An agent is autonomous if its behavior is determined by its own experience (with ability to learn and adapt) rather than knowledge of the designer.
- 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 become more autonomous with experience

PEAS

- Task performance can be measured by following parameters
- PEAS (Performance, Environment, Actuators, Sensors) description

Agent Type	Performance Measure	Environment	Actuators	Sensors
Taxi driver	Safe, fast, legal, comfortable trip, maximize profits	Roads, other traffic, pedestrians, customers	Steering, accelerator, brake, signal, horn, display	Cameras, sonar, speedometer, GPS, odometer, accelerometer, engine sensors, keyboard

Figure 2.4 PEAS description of the task environment for an automated taxi.

Agent Types and PEAS

Must first specify the setting for intelligent agent design

Agent Type	Performance Measure	Environment	Actuators	Sensors
Medical diagnosis system	Healthy patient, reduced costs	Patient, hospital, staff	Display of 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 of scene categorization	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 of exercises, suggestions, corrections	Keyboard entry

Performance measures

- **Performance measure:** An objective criterion for success of an agent's behavior.
- **Performance measures of a vacuum-cleaner agent:** amount of dirt cleaned up, amount of time taken, amount of electricity consumed, level of noise generated, etc.
- **Performance measures self-driving cars:** time to reach destination (minimize), safety, predictability of behavior for other agents, reliability, etc.
- **Performance measure of game-playing agent:** win/loss percentage (maximize), robustness, unpredictability (to "confuse" opponent), etc.

Good behaviour

- Agent learn from perceive sequences and act upon it
- Rational Agent selects action which maximizes the performance
- Intelligent agent has sufficiently complex and multiple rationality working together
- Autonomous agent should compensate for half/inaccurate knowledge
- Omniscient agent knows the outcome of input and its actions. It is ideal.
- Any agent should satisfy PEAS (Performance, Environment, Actuators, Sensors)

Environment types

How to make the right decisions?

Decision theory

Nature of Environments

- Specifying the task environment task environment specification includes the performance measure, the external environment, the actuators, and the sensors

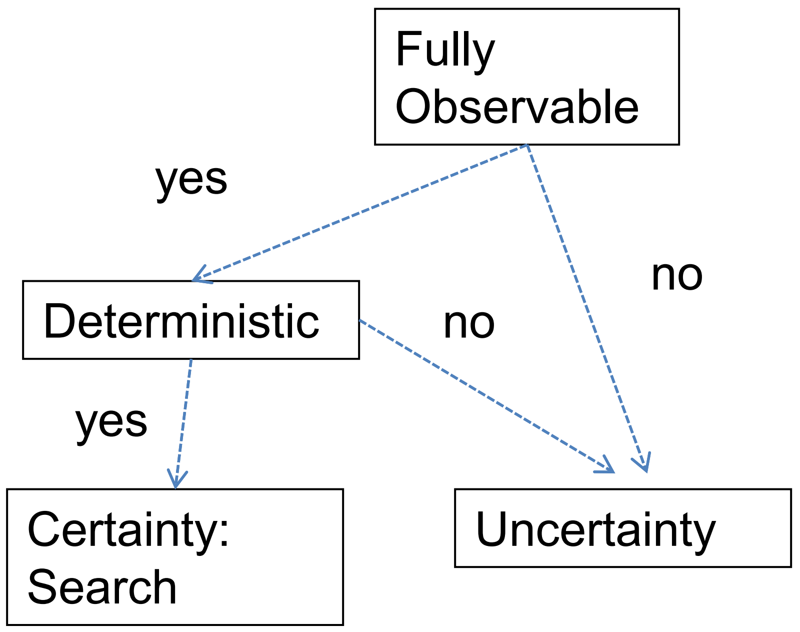
Task Environment	Observable	Agents	Deterministic	Episodic	Static	Discrete
Crossword puzzle	Fully	Single	Deterministic	Sequential	Static	Discrete
Chess with a clock	Fully	Multi	Deterministic	Sequential	Semi	Discrete
Poker	Partially	Multi	Stochastic	Sequential	Static	Discrete
Backgammon	Fully	Multi	Stochastic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

Figure 2.6 Examples of task environments and their characteristics.

Environment types

Choice under (Un)certainty

1. Fully observable (vs. partially observable)
2. Deterministic (vs. stochastic)
3. Episodic (vs. sequential)
4. Static (vs. dynamic)
5. Discrete (vs. continuous)
6. Single agent (vs. multiagent)



1. Fully observable (vs. partially observable)

- An agent's sensors give it access to the complete state at each point in time.
- Is everything an agent requires to choose its actions available to it via its sensors? Perfect or Full information.
  - to choose an action, the environment is fully accessible (or observable)
- If not, parts of the environment are inaccessible
  - Agents must make informed guesses about the world.
- In decision theory: perfect information vs. imperfect information.

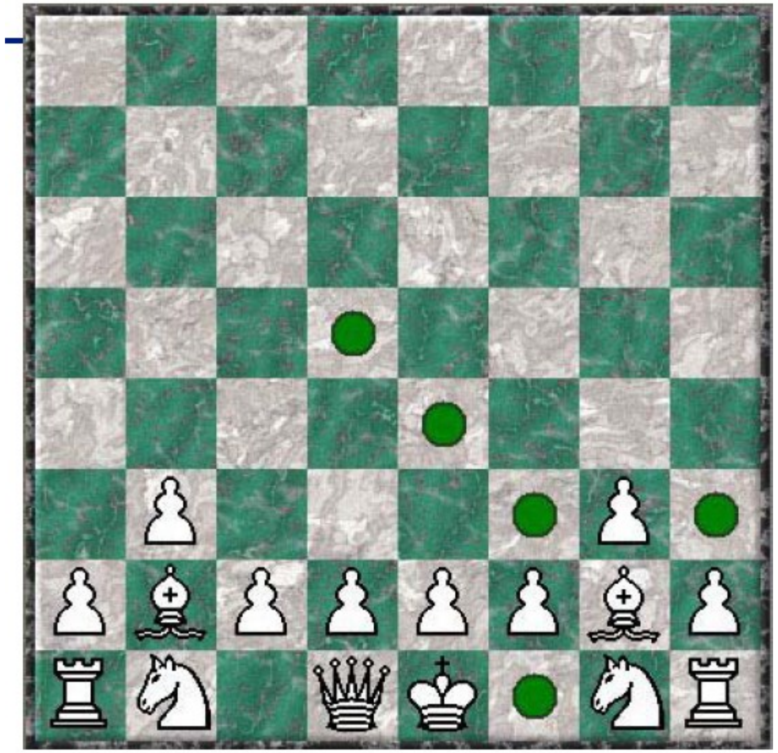
Cross Word	**Poker**	**Backgammon**	**Taxi driver**	**Part picking robot**	**Image analysis**
Fully	Partially	Partially	Partially	Fully	Fully

Kriegspiel Chess

1. Fully observable / Partially observable

(e.g. chess – what about Kriegspiel?)

- *Kriegspiel* --- you can't see your opponent!
- Making things a bit more challenging…
- Incomplete /uncertain information inherent in the game.
- Balance exploitation (best move given current knowledge) and exploration (moves to explore where the opponent's pieces might be).
- Use probabilistic reasoning techniques.



2. Deterministic (vs. stochastic)

- An environment is **deterministic** if the next state of the environment is completely determined by the current state of the environment and the action of the agent;
- In a **stochastic** environment, there are multiple, unpredictable outcomes. (If the environment is deterministic except for the actions of other agents, then the environment is **strategic**).
- Uncertainty can also arise because of computational limitations.
- In a fully observable, deterministic environment, the agent need not deal with uncertainty.
  - e.g., we may be playing an omniscient ("all knowing") opponent but we may not be able to compute his/her moves.
- Does the change in world state
  - Depend only on current state and agent's action?
- Non-deterministic environments
  - Have aspects beyond the control of the agent
  - Utility functions have to guess at changes in world

Cross Word	Poker	Backgammon	Taxi driver	Part picking robot	Image analysis
Deterministic	Stochastic	Stochastic	Stochastic	Stochastic	Deterministic

3. Episodic (vs. sequential)

- The agent's experience is divided into atomic "episodes" (each episode consists of the agent perceiving and then performing a single action).
- Subsequent episodes do not depend on what actions occurred in previous episodes. Choice of action in each episode depends only on the episode itself.
  - (E.g., classifying images.)
- In a sequential environment, the agent engages in a series of connected episodes. Current decisions can affect future decisions.
  - (E.g., chess and driving)
- Is the choice of current action
  - Dependent on previous actions?
  - If not, then the environment is episodic
- In non-episodic environments:
  - Agent has to plan ahead:
    - Current choice will affect future actions

Cross Word	**Poker**	**Backgammon**	**Taxi driver**	**Part picking robot**	**Image analysis**
Sequential	Sequential	Sequential	Sequential	Episodic	Episodic

4. Static (vs. dynamic):

- A static environment does not change
  - while the agent is thinking over what to do
- Dynamic environments do change
  - So agent should /could consult the world when choosing actions
  - Alternatively: anticipate the change during deliberation OR make decision very fast
- Semi-Dynamic: If the environment itself does not change with the passage of time but the agent's performance score does.
- Another example: off-line route planning vs. on-board navigation system

Cross Word	**Poker**	**Backgammon**	**Taxi driver**	**Part picking robot**	**Image analysis**
Static	Static	Static	Dynamic	Dynamic	Semi

5. Discrete (vs. continuous)

- A limited number of distinct, clearly defined percepts and actions vs. a range of values (continuous)
- If the number of distinct percepts and actions is limited, the environment is discrete, otherwise it is continuous.

Cross Word	**Poker**	**Backgammon**	**Taxi driver**	**Part picking robot**	**Image analysis**
Discrete	Discrete	Discrete	Conti	Conti	Conti

6. Single agent (vs. multiagent):

- An agent operating by itself in an environment or there are many agents working together
- If the environment contains other intelligent agents, the agent needs to be concerned about strategic, game-theoretic aspects of the environment (for either cooperative *or* competitive agents).
- Most engineering environments don't have multi-agent properties, whereas most social and economic systems get their complexity from the interactions of (more or less) rational agents.

Cross Word	**Poker**	**Backgammon**	**Taxi driver**	**Part picking robot**	**Image analysis**
Single	Multi	Multi	Multi	Single	Single

Summary

	Observable	Deterministic	Episodic	Static	Discrete	Agents
Cross Word	Fully	Deterministic	Sequential	Static	Discrete	Single
Poker	Fully	Stochastic	Sequential	Static	Discrete	Multi
Backgammon	Partially	Stochastic	Sequential	Static	Discrete	Multi
Taxi driver	Partially	Stochastic	Sequential	Dynamic	Conti	Multi
Part picking robot	Partially	Stochastic	Episodic	Dynamic	Conti	Single
Image analysis	Fully	Deterministic	Episodic	Semi	Conti	Single

Agent types

Agent types

- Six basic types in order of increasing generality:

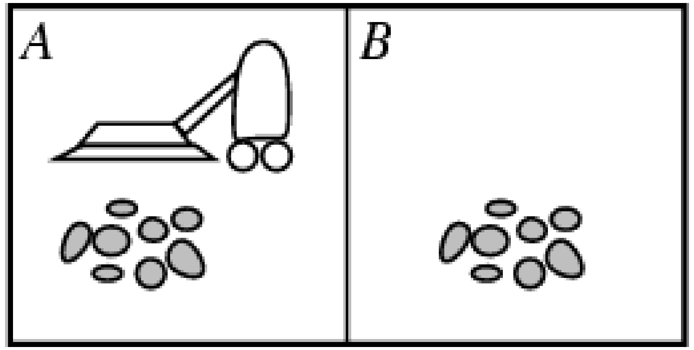
1. Table-lookup agent
2. Simple reflex agents
3. Reflex agents with state/model
4. Goal-based agents
5. Utility-based agents
6. Learning agents (combination of all above)

1. Table-lookup agent

- Uses a percept sequence / action table in memory to find the next action. Implemented as a (large) lookup table.
- Drawbacks:
  - Huge table (often simply too large)
  - Takes a long time to build the table
  - No autonomy
  - Need a long time to learn the table entries
- Action sequence of length K, gives 4^K different possible sequences.
- At least many entries are needed in the table. So, even in this very toy world, with K = 20, you need a table with over 4^20 > 10^12 entries.

Vacuum-cleaner world

- Toy example
- Percepts: robot senses its location and "cleanliness."
- So, location and contents, e.g., [A, Dirty], [B, Clean].
- Actions: *Left*, *Right*, *Suck*, *NoOp*
- *Agent's function* → *look-up table*
  - *For many agents this is a very large table*



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

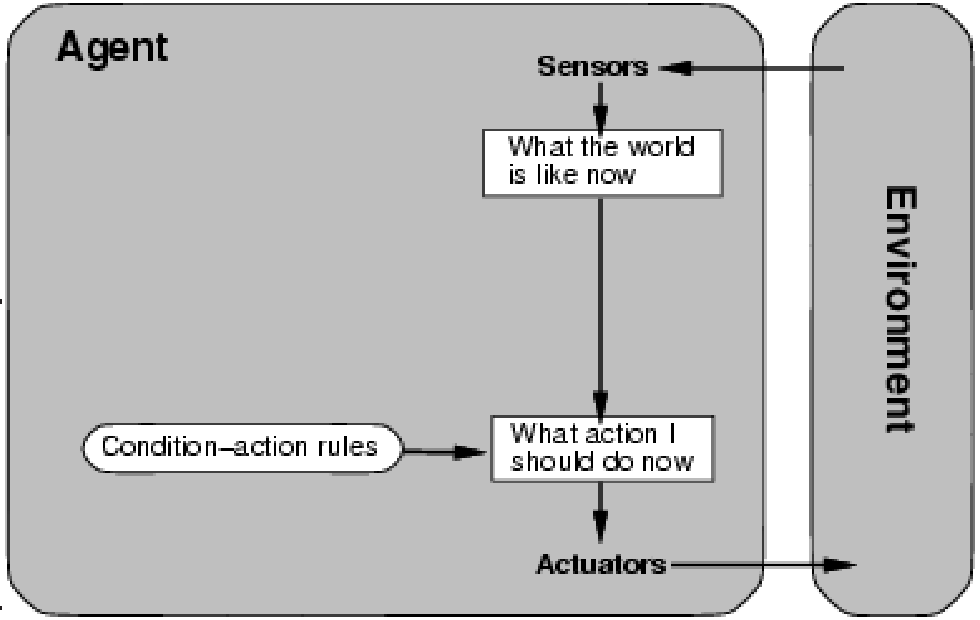
1. Table-lookup agent

- In more real-world scenarios, one would have many more different percepts (e.g. many more locations), e.g., >=100.
- There will therefore be 100^K different possible sequences of length K. For K = 20, this would require a table with over 100^20 = 10^40 entries. Infeasible to even store.
- Table lookup formulation is mainly of theoretical interest.
- For practical agent systems, we need to find much more compact representations.
- For example, logic-based representations, Bayesian net representations, or neural net style representations, or use a different agent architecture, e.g., "ignore the past" --- Reflex agents.
- Chess – openings, endings
  - Lookup table (not a good idea in general)
    - 35^{100} entries required for the entire game

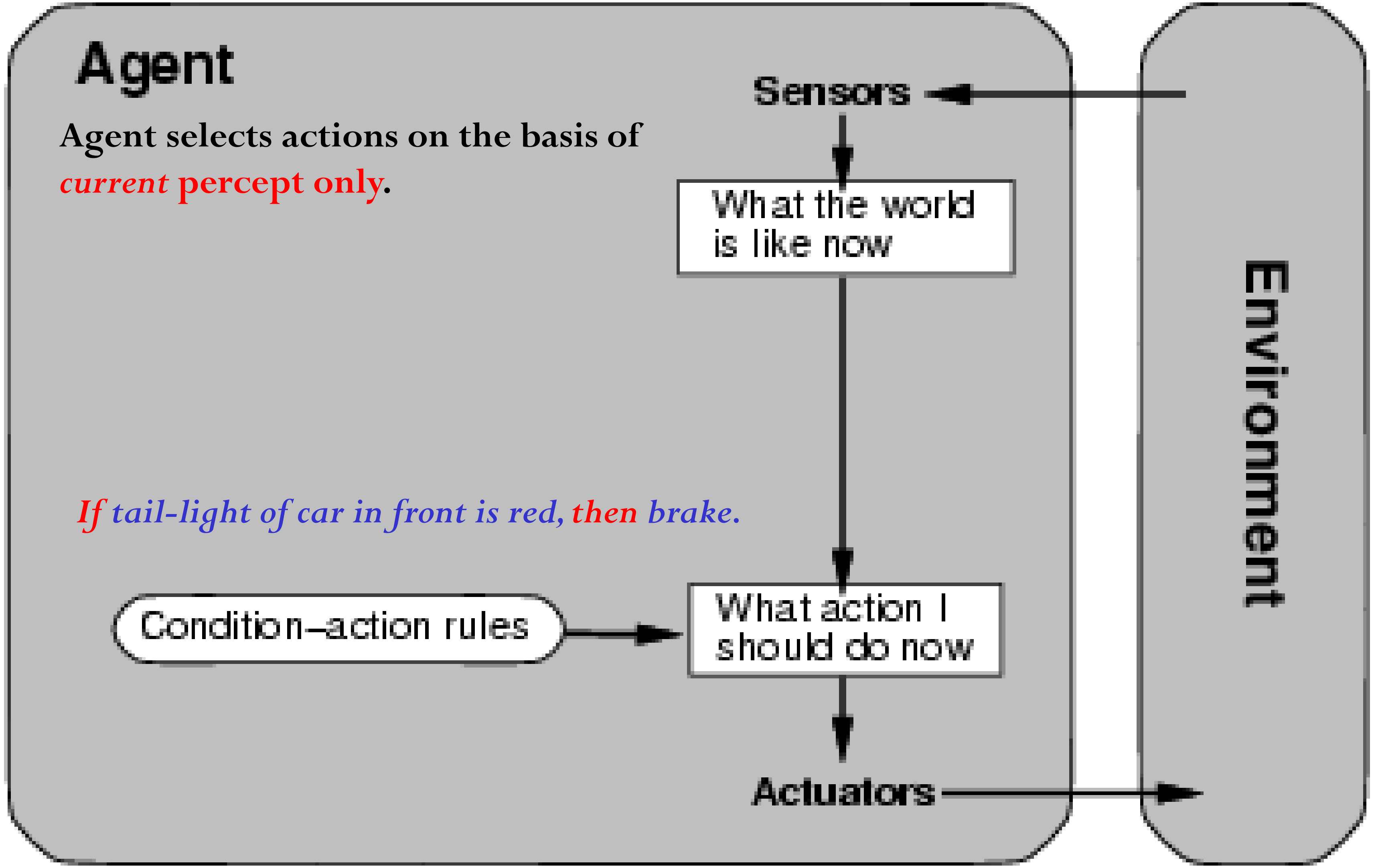
2. Simple reflex agents

- Agents **do not have memory** of past world states or percepts. Actions depend solely on **current perception**. Action becomes a "reflex." Uses **condition-action rules**.
- Simple but very limited intelligence. **Action does not depend on percept history, only on current percept.** Therefore, no memory requirements.
- Infinite loops
  - Suppose the vacuum cleaner does not observe location. What do you do given location = clean? Left of A or right on B -> infinite loop.
  - Possible Solution: Randomize action.

```
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
```



2. Simple reflex agents



2. Simple reflex agents

- Closely related to "behaviorism" (psychology; quite effective in explaining lower-level animal behaviors, such as the behavior of ants and mice).
- The Robot largely behaves like this. Behaviors are robust and can be quite effective and surprisingly complex.
- But, how does complex behavior arise from simple reflex behavior?
- E.g. Ant colonies and bee hives are quite complex.
- Simple rules in a diverse environment can give rise to surprising complexity.

```
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
```

3. Reflex agents with state/model

- Recall the agent function that maps from percept histories to actions:

$$[f: P^* \rightarrow A]$$

- An agent program can implement an agent function by maintaining an **internal state**.
- The internal state can contain information about the state of the external environment.
- The state depends on the history of percepts and on the history of actions taken:

$$[f: P^*, A^* \rightarrow S \rightarrow A] \text{ where } S \text{ is the set of states.}$$

- If each internal state includes all information relevant to information making, the state space is **Markovian**.

3. Reflex agents with state/model

- States: Beyond Reflexes
- States and Memory: Game Theory
- If each state includes the information about the percepts and actions that led to it, the state space has **perfect recall**.
- **Perfect Information** = Perfect Recall + Full Observability + Deterministic Actions.

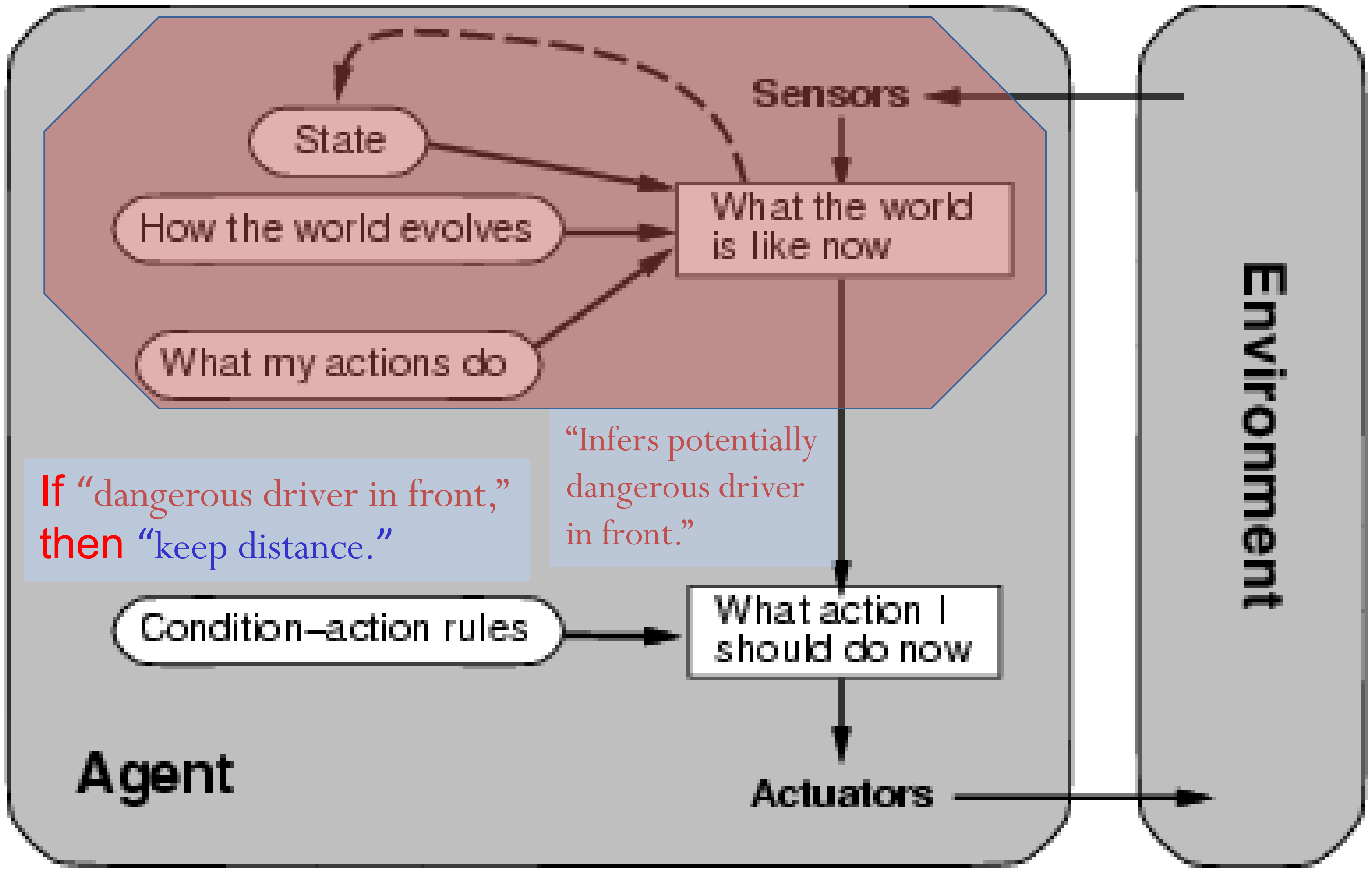
```
function MODEL-BASED-REFLEX-AGENT(percept) returns an action
   persistent: state, the agent's current conception of the world state
               model, a description of how the next state depends on current state and action
               rules, a set of condition–action rules
               action, the most recent action, initially none

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

3. Reflex agents with state/model

- 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
- Key difference (w.r.t. simple reflex agents):
  - Agents have **internal state**, which is used to keep track of past states of the world.
  - Agents have the ability **to represent change in the World**.
- Example: behavior based robots.

3. Reflex agents with state/model

Module: Logical Agents Representation and Reasoning

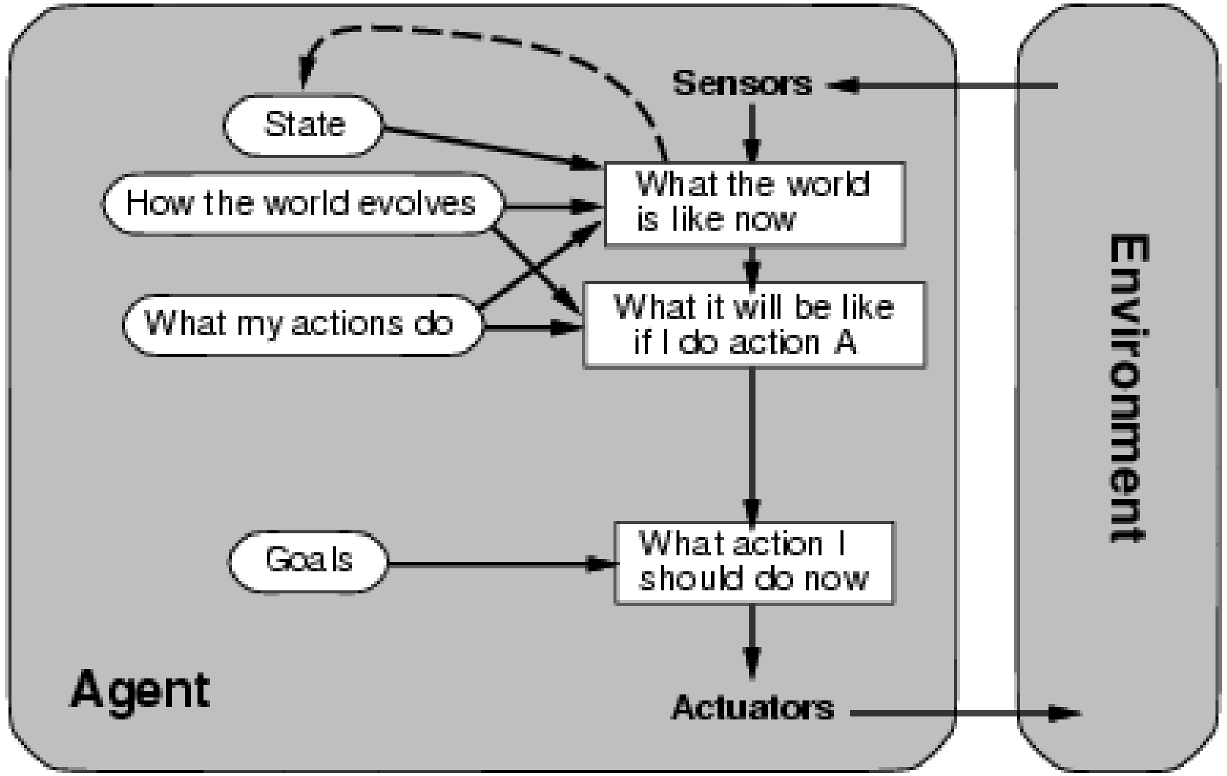


3. Reflex agents with state/model

- Build complex and intelligent robots by decomposing behaviors into a hierarchy of skills, each defining a percept-action cycle for one very specific task.
- Each behavior is modeled with a few states corresponding to a complex function or module.
- Increasingly complex behaviors arise from the combination of simple behaviors.
- A more complex behavior that sits on top of simple behaviors.
- The more complex behaviors subsume the less complex ones to accomplish their goal.
- Examples: The most basic simple behaviors are on the level of reflexes:
  - avoid an object; go toward food if hungry; move randomly.
  - collision avoidance, wandering, exploring, recognizing doorways, etc.

4. Goal-based agents

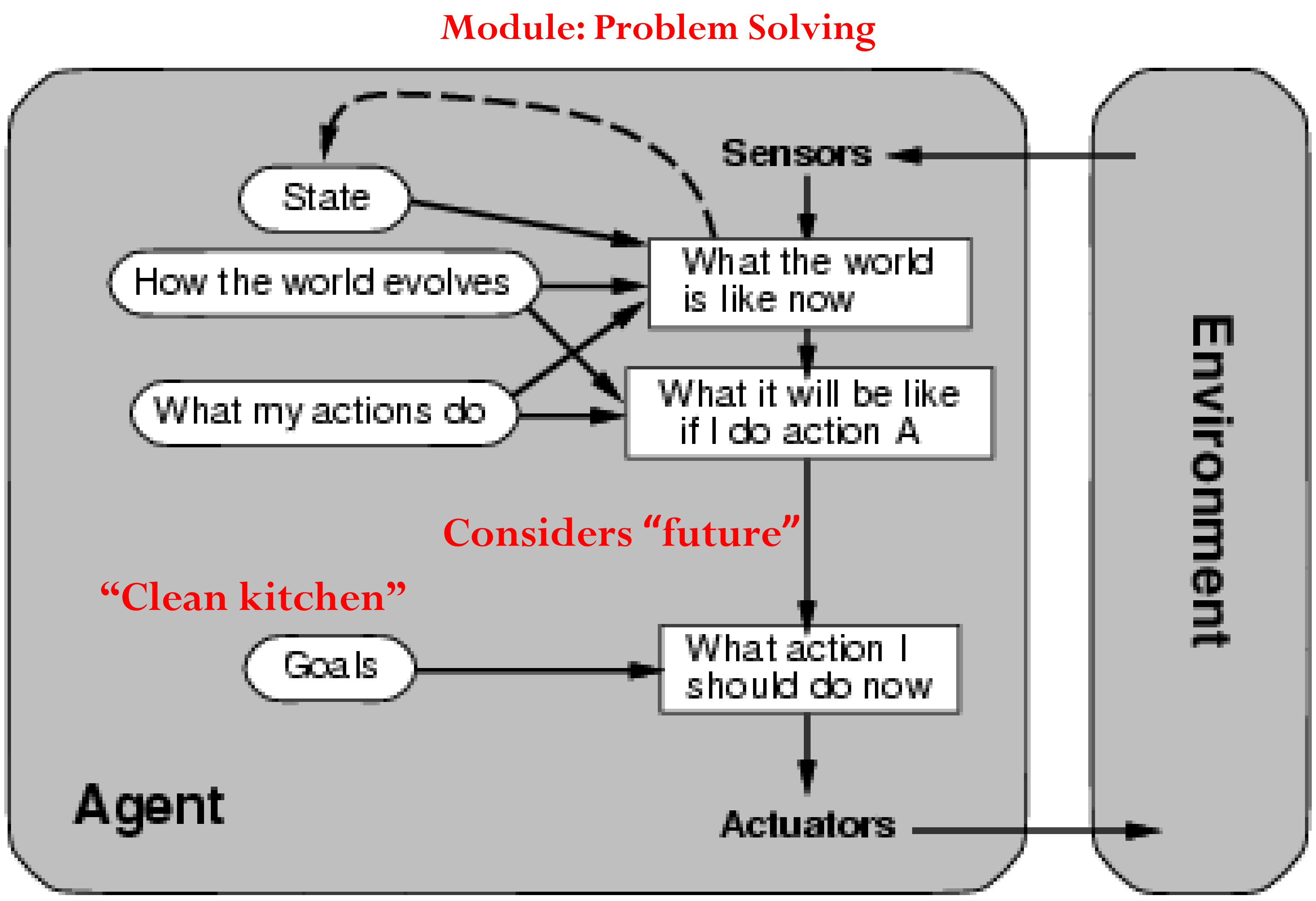
- Reflex agent breaks when it sees brake lights. Goal based agent reasons
  - Brake light -> car in front is stopping -> I should stop -> I should use brake
- knowing state and environment? Enough?
  - Taxi can go left, right, straight
- Have a goal
  - A destination to get to
- Uses knowledge about a goal
  - to guide its actions
  - E.g., Search, planning



4. Goal-based agents

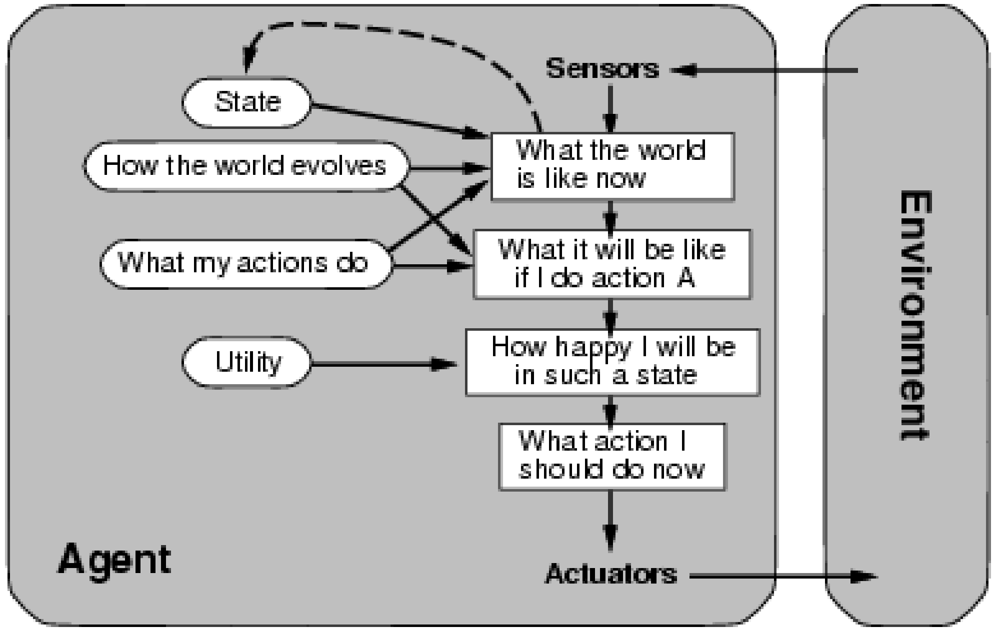
- Key difference w.r.t. Model-Based Agents:
  - In addition to state information, have goal information that describes desirable situations to be achieved.
- Agents of this kind take future events into consideration.
- What *sequence* of actions can I take to achieve certain goals?
- Choose actions so as to (eventually) achieve a (given or computed) goal.
- Agent keeps track of the world state as well as the set of goals it's trying to achieve: chooses actions that will (eventually) lead to the goal(s).
- More flexible than reflex agents → may involve Search and Planning

4. Goal-based agents



5. 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", "goodness", "success".
- Where does the utility measure come from?
  - Economics: money.
  - Biology: number of offspring.
  - Your life?

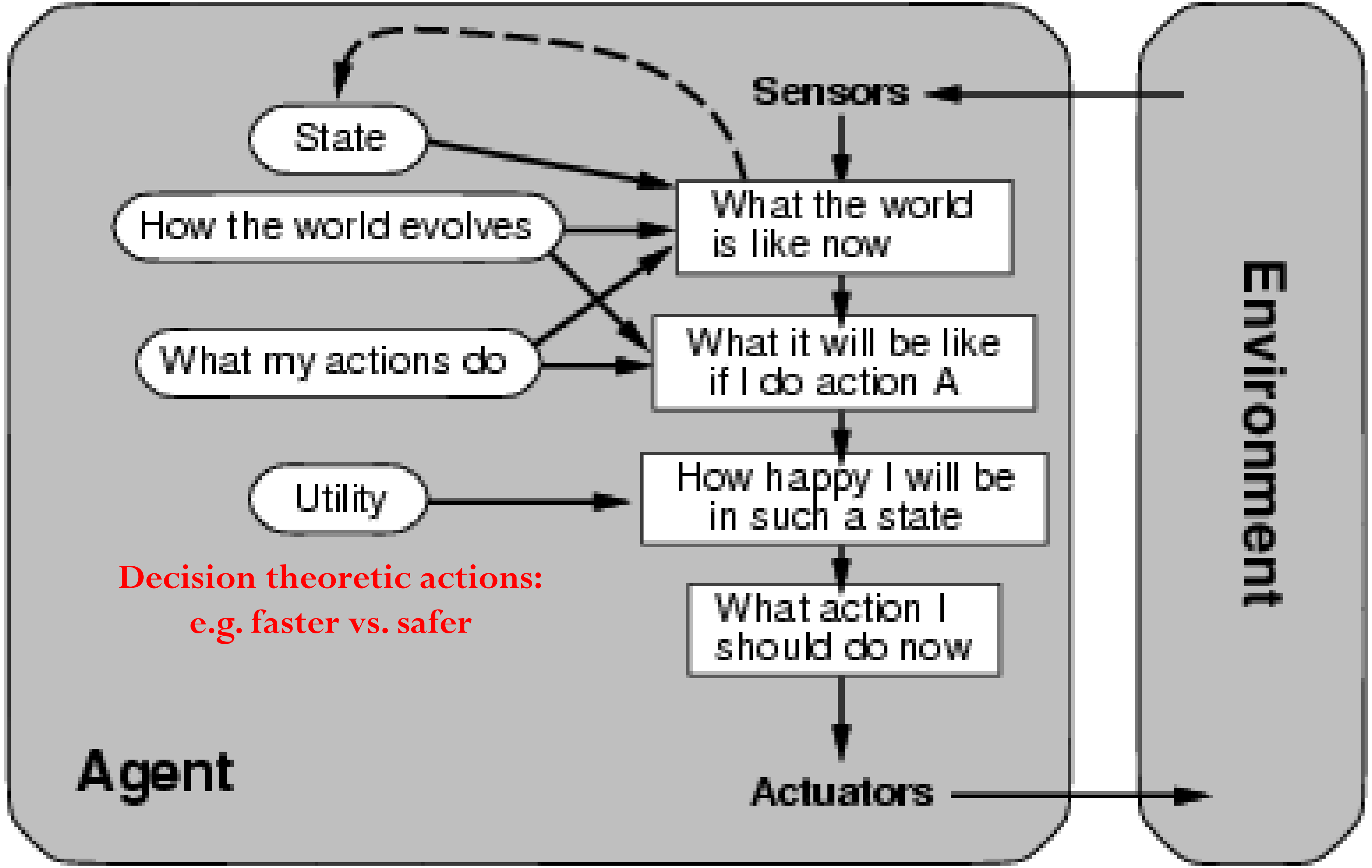


5. Utility-based agents

- When there are multiple possible alternatives, how to decide which one is best?
- Goals are qualitative: A goal specifies a crude distinction between a happy and unhappy state, but often needs a more general performance measure that describes "degree of happiness."
- Utility function U: State $\rightarrow$ R indicating a measure of success or happiness when at a given state.
- Important for making tradeoffs: Allows decisions comparing choice between conflicting goals, and choice between likelihood of success and importance of goal (if achievement is uncertain).
- Use decision theoretic models: e.g., faster vs. safer.

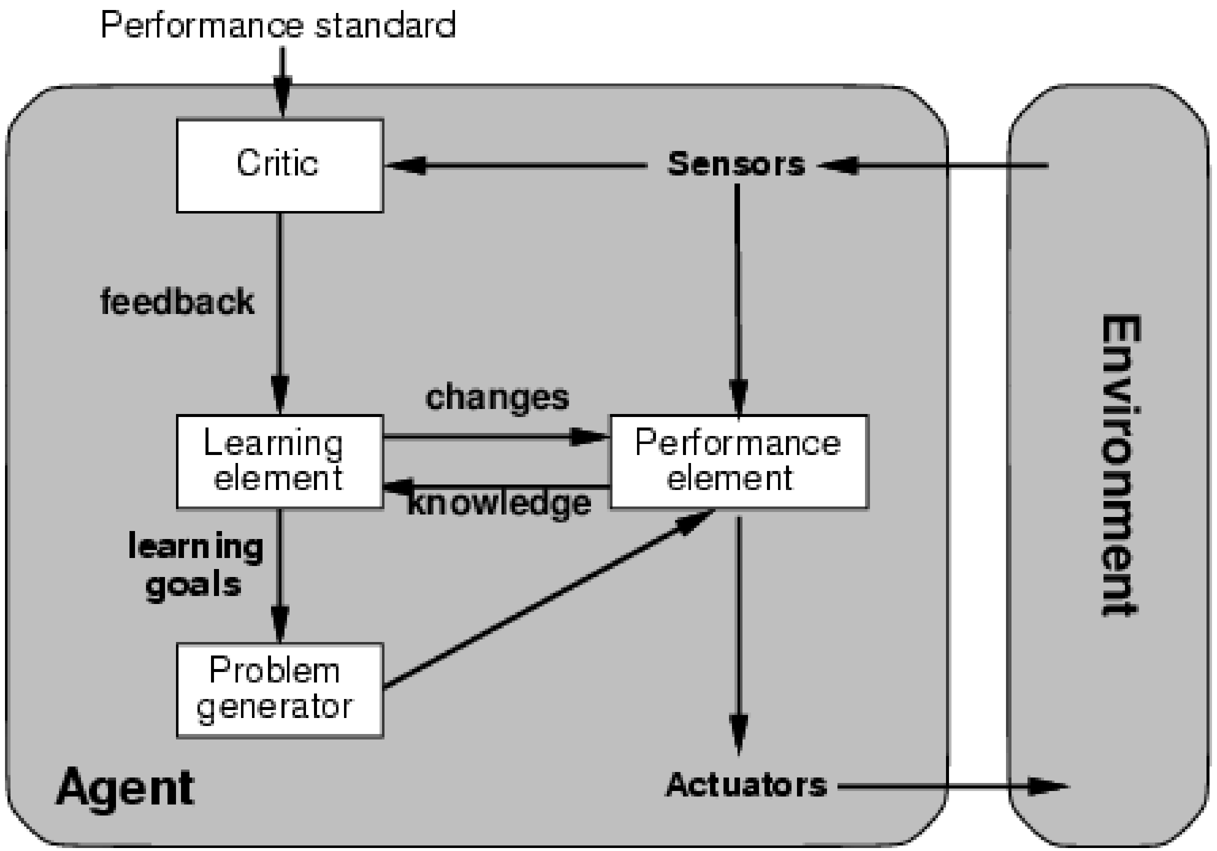
5. Utility-based agents

Module: Decision Making



6. Learning agents

- Performance element of agent
  - Input sensor
  - Output action
- Learning element
- Modifies performance elements.
  - Critic: how the agent is doing
  - Input: checkmate?
  - Fixed
- Problem generator
  - Tries to solve the problem differently instead of optimizing.
  - Suggests **exploring** new actions -> new problems.



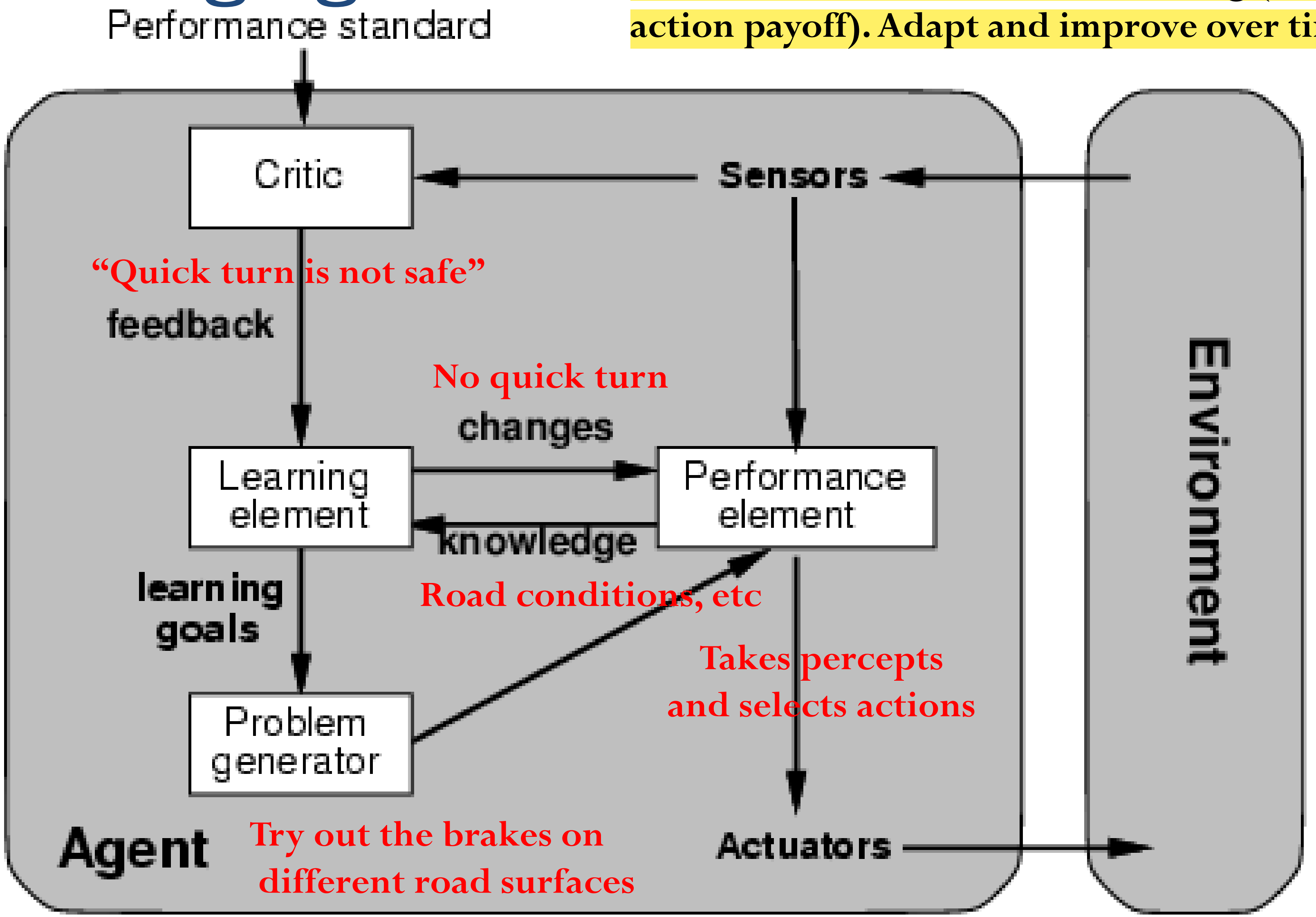
6. Learning agents (Taxi driver)

- Performance element
  - How it currently drives
- Taxi driver Makes quick left turn across 3 lanes
  - Critics observes 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
- Exploration vs. Exploitation
  - Learning experience can be costly in the short run

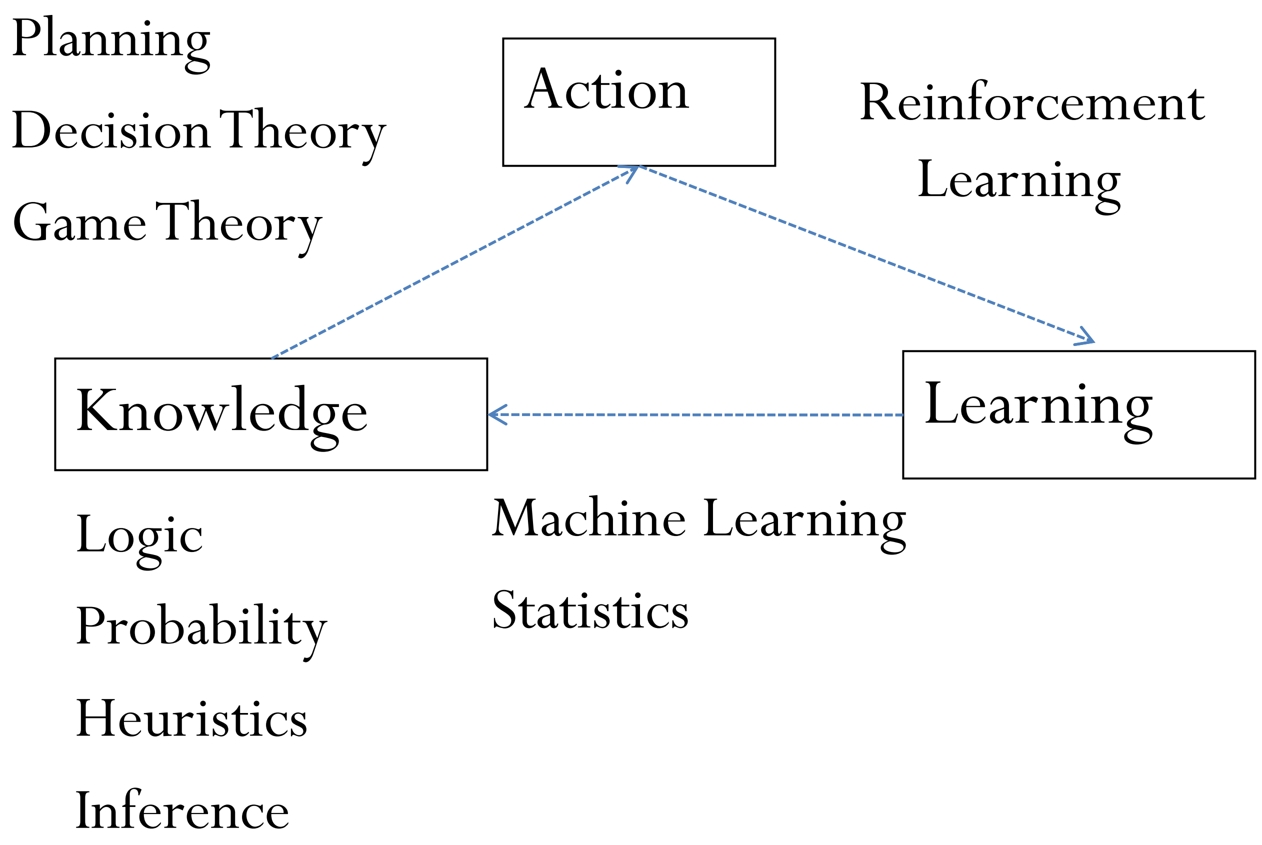
6. Learning agents

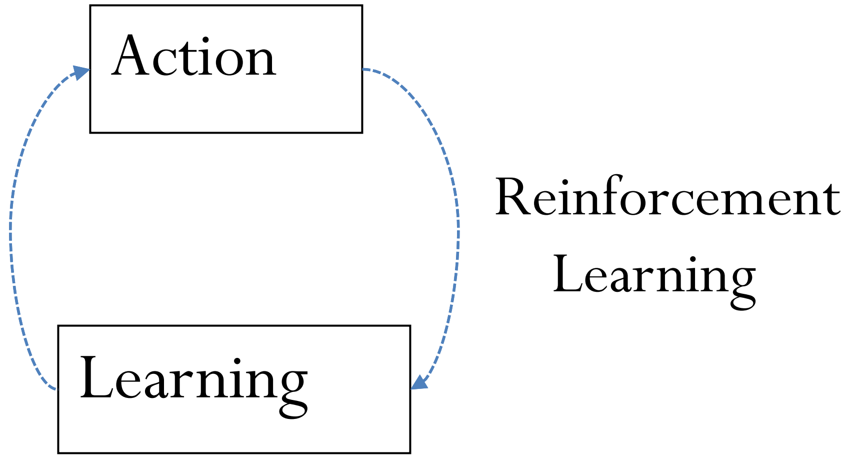
More complicated when agent needs to learn utility information: Reinforcement learning (based on action payoff). Adapt and improve over time

Module: Learning



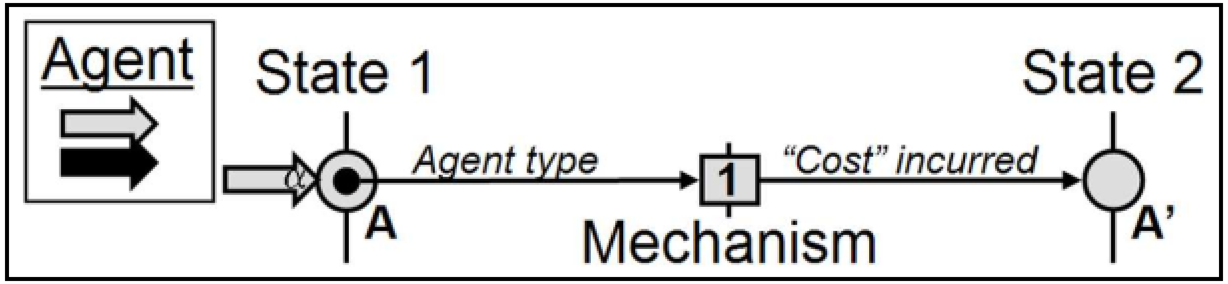
The Big Picture: AI for Model-Based Agents





Studied in AI, Cybernetics, Control Theory, Biology, Psychology.

Changeability



- System "changeability" (Ross et al.) define it as system ilities
- **Change Agent:** Instigator, or force, which employs a given change mechanism in order to achieve a desired change effect
- **Change Effect:** The difference in system states (performance or value) before and after a change has taken place
- **Change Objective:** The specific approach / plan / goal / strategies employed to achieve a desired change effect
- **Change Enablers:** (e.g. design elements) enable desired objective
- **Change Considerations:** Design considerations (e.g. conditions, resources, constraints, etc.) applied to design / operational approaches

Ross, A.M., Rhodes, D.H., and Hastings, D.E., "Defining Changeability: Reconciling Flexibility, Adaptability, Scalability, Modifiability, and Robustness for Maintaining Lifecycle Value," *Systems Engineering*, Vol. 11, No. 3, pp. 246-262, Fall 2008.

Summary: Agents

- An agent perceives and acts in an environment, has an architecture, and is implemented by an agent program.
- A rational agent always chooses the action which maximizes its expected performance, given its percept sequence so far.
- An autonomous agent uses its own experience rather than built-in knowledge of the environment by the designer.
- An agent program maps from percept to action and updates its internal state.
- Representing knowledge is important for successful agent design.
- The most challenging environments are partially observable, stochastic, sequential, dynamic, and continuous, and contain multiple intelligent agents.

Summary: Environments

- Agents can be described by their PEAS.
- Environments can be described by several key properties.
- A agent maximizes the performance measure for their PEAS.
- The performance measure depends on the **agent function**.
- The **agent program implements** the agent function.
- Common and useful combinations of environment/agent architecture.

Summary: Agent types

- Table-driven agents
  - use a percept sequence/action table in memory to find the next action. They are implemented by a (large) lookup table.
- Simple reflex agents
  - are based on condition-action rules, implemented with an appropriate production system.
  - They are stateless devices which do not have memory of past world states.
  - respond immediately to percepts.
- Agents with memory - Model-based reflex agents
  - have an internal state, which is used to keep track of past states of the world.

Summary: Agent types

- Agents with goals – Goal-based agents
  - are agents that, in addition to state information, have goal information that describes desirable situations.
  - act in order to achieve their goal(s), possible sequence of steps.
- Utility-based agents
  - base their decisions on classic axiomatic utility theory in order to act rationally.
  - maximize their own utility function.
- Learning agents
  - they have the ability to improve performance through learning.

References

- Stuart Russel, and Peter Norvig. "Artificial intelligence: A modern approach. third edit." Upper Saddle River, New Jersey 7458 (2015).
- Bart Selman, "CS 4700: Foundations of Artificial Intelligence" https://www.cs.cornell.edu/courses/cs4700/2013fa/

ขอบคุณ

Thai

Grazie

Italian

תודה רבה

Hebrew

Gracias

Spanish

Спасибо

Russian

English

Thank You

Obrigado

Portuguese

شكراً

Arabic

Merci

French

多謝

Traditional Chinese

https://sites.google.com/site/animeshchaturvedi07

Danke

German

धन्यवाद

Hindi

多谢

Simplified Chinese

நன்றி

Tamil

Tamil

ありがとうございました

Japanese

감사합니다

Korean