INTELLIGENT AGENT

2/7/2018

Module 1

Introduction

- Computers are not very good at knowing what to do: every action a computer performs must be explicitly anticipated, planned for, and coded by a programmer.
- If a computer program ever encounters a situation that its designer did not anticipate, then the results not usually pretty — a system crash at best, multiple loss of life at worst.
- This mundane fact is at the heart of our relationship with computers.
- For the most part, we are happy to accept computers as obedient, literal, unimaginative servants.

Introduction

- We require systems that can decide for themselves what they need to do in order to satisfy their design objectives. Such computer systems are known as agents.
- Agents that must operate robustly in rapidly changing, unpredictable, or open environments, where there is a significant possibility that actions can fail are known as intelligent agents, or sometimes autonomous agents.

Example

- When a space probe makes its long flight from Earth to the outer planets, a ground crew is usually required to continually track its progress, and decide how to deal with unexpected eventualities.

- This is costly and, if decisions are required quickly, it is simply not practicable. For these reasons, organizations like NASA are seriously investigating the possibility of making probes more autonomous — giving them richer decision making capability and responsibilities.

Motivation

1. Agents are used to provide a consistent viewpoint on various topics in the field AI.

2. Agents require essential skills to perform tasks that require intelligence.

3. Intelligent agents use methods and techniques from the field of AI.

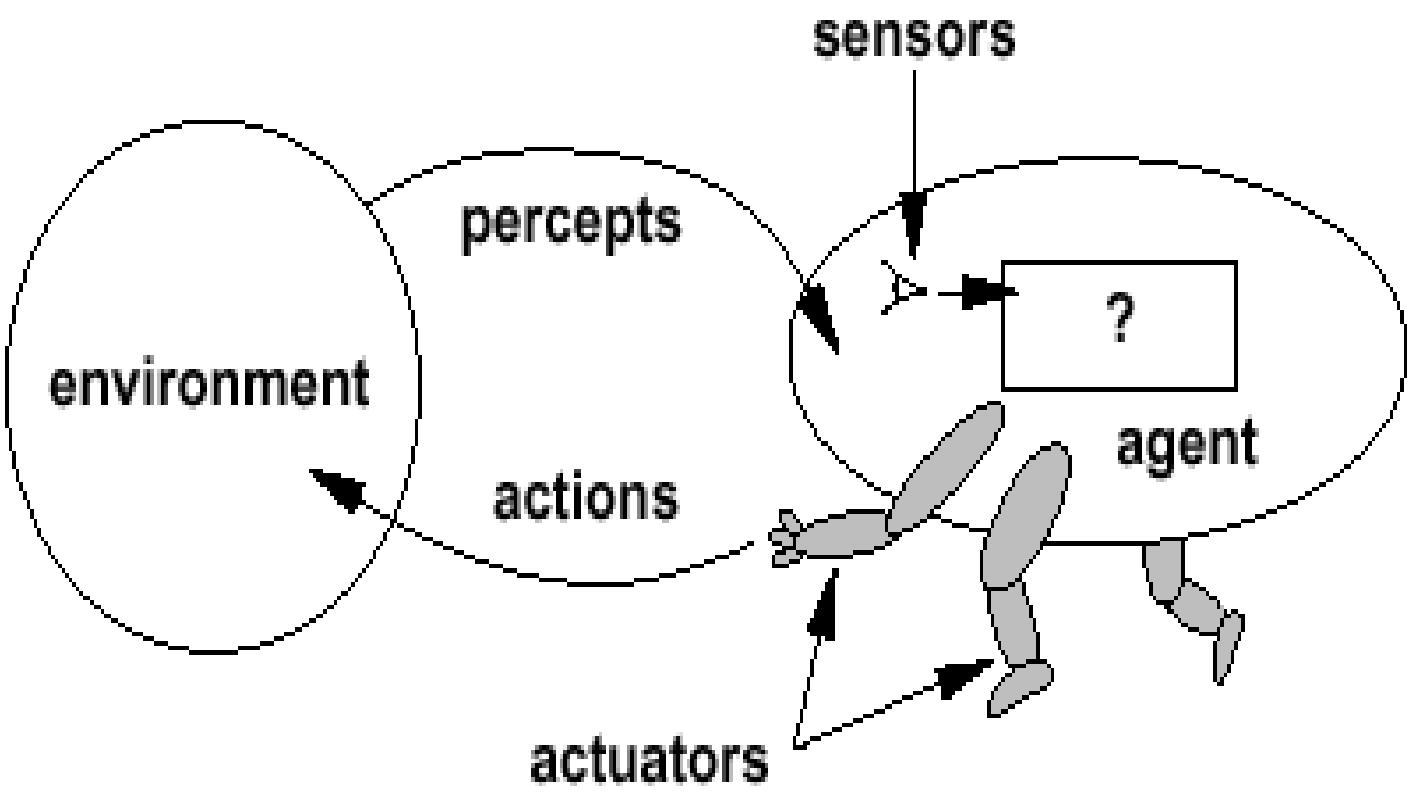
What is a Agent?

- An **agent** is anything that **perceiving** its environment through **sensors** and **acting** upon that environment through **actuators.**

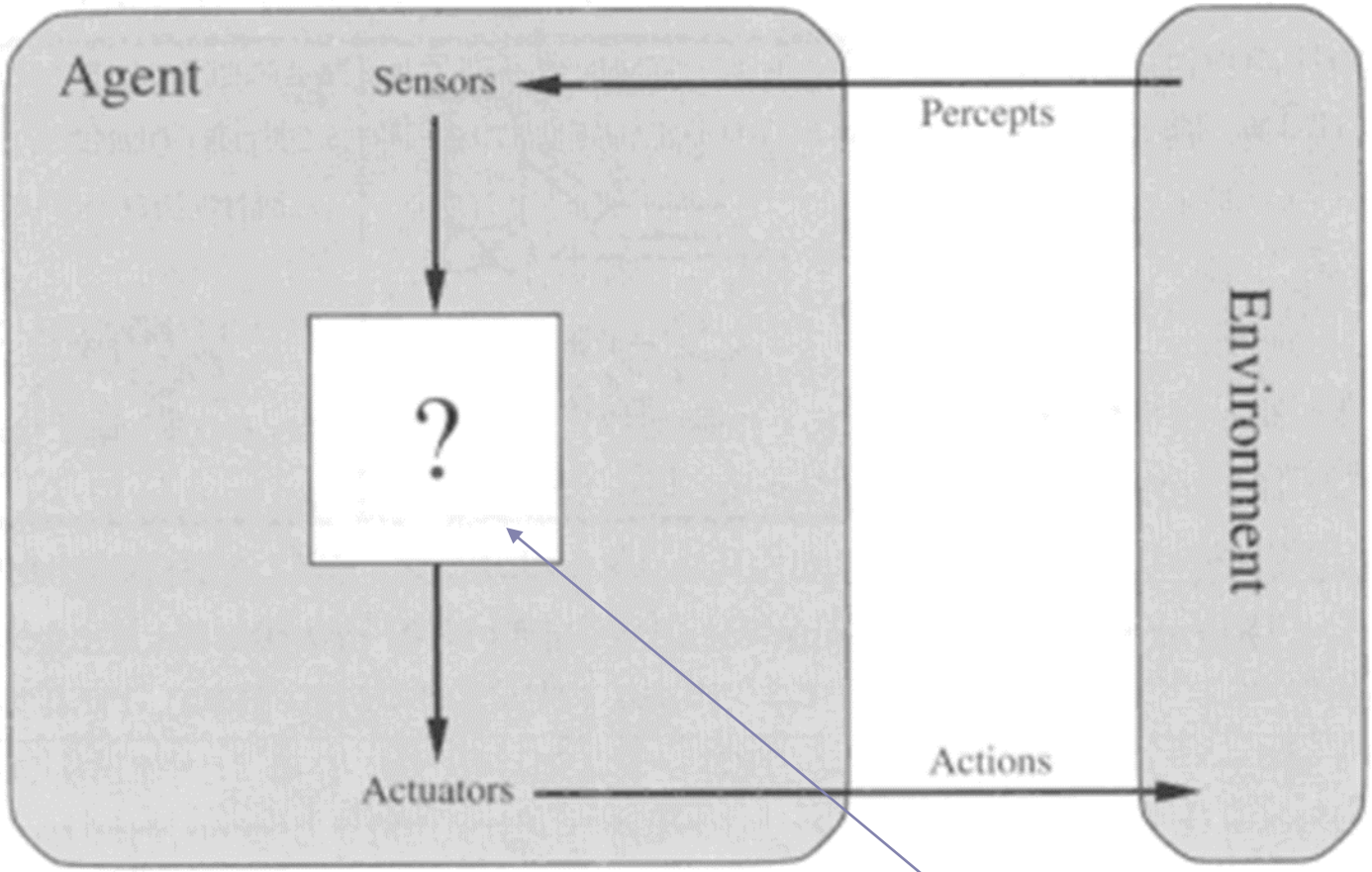
- The right action is the one that will cause the agent to be most successful.

- *An agent is autonomous if its behavior is determined by its own experience (with ability to learn and adapt).*

Intelligent Agents



Intelligent Agents



Simple Terms

- Percept
  - Agent's perceptual inputs at any given instant
- Percept sequence
  - Complete history of everything that the agent has ever perceived.
- Agent's behavior is *mathematically* described by **Agent function**
  - A function mapping any given percept sequence to an action
- *Practically* it is described by An **agent program**
  - The real implementation

Examples of Agents

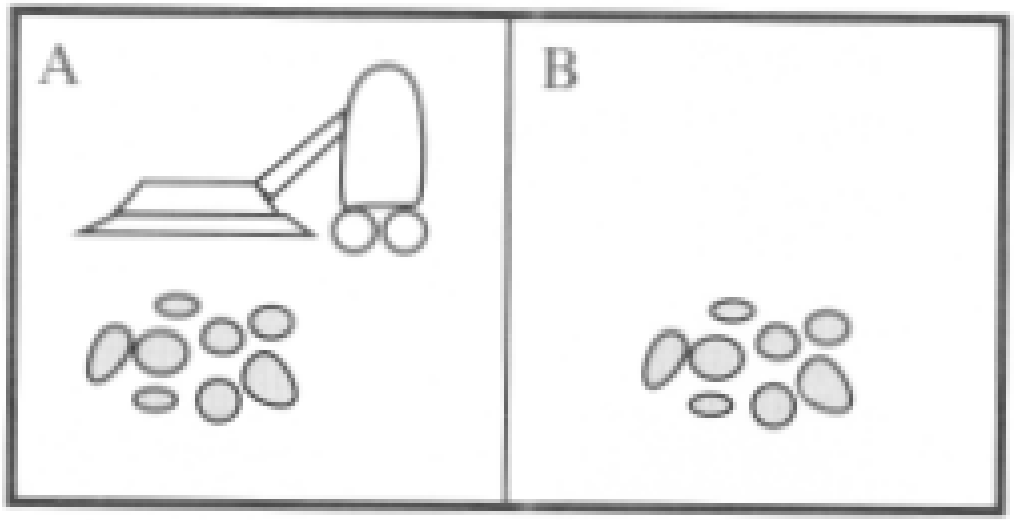
- Human agent
  - Eyes, ears, skin, taste buds, etc. For sensors
  - Hands, fingers, legs, mouth, etc. For actuators
    - Powered by muscles
- Robot
  - Camera, infrared, bumper, etc. For sensors
  - Grippers, wheels, lights, speakers, etc. For actuators
    - Often powered by motors
- Software agent
  - Functions as sensors
    - Information provided as input to functions in the form of encoded bit strings or symbols
  - Functions as actuators
    - Results deliver the output

Agents and Environments

- An agent perceives its environment through sensors.
- The complete set of inputs at a given time is called a percept.
- The current percept, or a sequence of percepts may influence the actions of an agent.
- It can change the environment through actuators.
- An operation involving an actuator is called an action
- Actions can be grouped into action sequences.

Vacuum-cleaner world

- Perception: Clean or Dirty? where it is in?
- Actions: Move left, Move right, suck, do nothing



Vacuum-cleaner world

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
⋮	⋮
[A, Clean], [A, Clean], [A, Clean]	Right
[A, Clean], [A, Clean], [A, Dirty]	Suck
⋮	⋮

Figure 2.3 Partial tabulation of a simple agent function for the vacuum-cleaner world shown in Figure 2.2.

Program implements the agent function

Function Reflex-Vacuum-Agent([*location*, *status*]) return an action

If *status* = *Dirty* **then return** *Suck*

else if *location* = *A* **then return** *Right*

else if *location* = *B* **then return** *left*

Concept of Rationality

- Rational agent
  - One that does the right thing
  - = every entry in the table for the agent function is correct (rational).
- What is correct?
  - The actions that cause the agent to be most successful
  - So we need ways to measure success.

Performance Measure

- Performance measure
  - An objective function that determines
    - How the agent does successfully
    - E.g., 90% or 30% ?
- An agent, based on its percepts
  - → action sequence :

if desirable, it is said to be performing well.

- *No universal performance measure for all agents*

Performance Measure

- A general rule:
  - Design performance measures according to
    - What one actually wants in the environment
    - Rather than how one thinks the agent should behave
- E.g., in vacuum-cleaner world
  - We want the floor clean, no matter how the agent behave.
  - We don't restrict how the agent behaves.

Rationality

- What is rational at any given time depends on four things:
  1. The performance measure defining the criterion of success.
  2. The agent’s prior knowledge of the environment.
  3. The actions that the agent can perform.
  4. The agent's percept sequence up to now.

Rational agent

- For each possible percept sequence,
  - an rational agent should select
- An **ideal rational agent:** ***for each possible percept sequence, an*** *ideal rational agent should do whatever action is expected to maximize its performance measure, on the basis of the evidence provided by the percept sequence and whatever built-in knowledge the agent has.*
- E.g., an exam
  - Maximize marks, based on the questions on the paper & your knowledge

Example of a rational agent

- Performance measure
  - Awards one point for each clean square
    - at each time step, over 10000 time steps
- Prior knowledge about the environment
  - The geography of the environment
  - Only two squares
  - The *effect* of the actions

Example of a rational agent

- Actions that can perform
  - Left, Right, Suck and NoOp
- Percept sequences
  - Where is the agent?
  - Whether the location contains dirt?
- Under this circumstance, the agent is rational.

Omniscience

- An omniscient agent
  - Need to be careful to distinguish between rationality and **omniscience.**
  - **An omniscient** agent knows the *actual outcome of its actions, and can act accordingly;*
  - *But omniscience is* impossible in reality
- An example
  - crossing a street but died of the fallen cargo door from 33,000ft

Omniscience

- Based on the circumstance, it is rational.
- As rationality maximizes
  - *Expected* performance
- Perfection maximizes
  - *Actual* performance
- Hence rational agents are not *omniscient*.

Learning

- Does a rational agent depend on only current percept?
  - No, the past percept sequence should also be used
  - This is called **learning**
  - After experiencing an episode, the agent
    - should adjust its behaviors to perform better for the same job next time.

Autonomy

- An agent's behavior can be based on both its own experience and the built-in knowledge used in constructing the agent for the particular environment in which it operates.
- A system is autonomous to the extent that its behavior is determined by its own experience.
- It would be reasonable to provide an artificial intelligent agent with some initial knowledge as well as an ability to learn.

Autonomy

- If an agent just relies on the prior knowledge/Built-in-knowledge of its designer rather than its own percepts then the agent lacks ***autonomy.***

A rational agent should be autonomous- it should learn what it can to compensate for partial or incorrect prior knowledge.

- E.g., a clock
  - No input (percepts)
  - Run only but its own algorithm (prior knowledge)
  - No learning, no experience, etc.

Software Agents

- Sometimes, the environment may not be the real world
  - E.g., flight simulator, video games, Internet
  - They are all artificial but very complex environments
  - Those agents working in these environments are called
    - Software agent (softbots)
    - Because all parts of the agent are software

Task Environments

- Task environments are the problems
  - While the rational agents are the solutions
- Specifying the task environment
  - **PEAS** description as fully as possible
    - **Performance**
    - **Environment**
    - **Actuators**
    - **Sensors**

  In designing an agent, the first step must always be to specify the task environment as fully as possible.
- Use automated taxi driver as an example

Task Environments

- Performance measure
  - How can we judge the automated driver?
  - Which factors are considered?
    - Getting to the correct destination
    - Minimizing fuel consumption
    - Minimizing the trip time and/or cost
    - Minimizing the violations of traffic laws
    - Maximizing the safety and comfort, etc.

Task environments

- Environment
  - A taxi must deal with a variety of roads
  - Traffic lights, other vehicles, pedestrians, stray animals, road works, police cars, etc.
  - Interact with the customer

Task Environments

- Actuators (for outputs)
  - Control over the accelerator, steering, gear shifting and braking.
  - A display to communicate with the customers.
- Sensors (for inputs)
  - Detect other vehicles, road situations.
  - GPS (Global Positioning System) to know where the taxi is.
  - Many more devices are necessary

Task environments

- A sketch of automated taxi driver

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.

Properties Of Task Environments

- Fully observable vs. Partially observable
  - If an agent's sensors give it access to the complete state of the environment at each point in time then the environment is effectively and fully observable
    - if the sensors detect all aspects
    - That are relevant to the choice of action

Properties Of Task Environments

- Partially observable

✓ 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.

Example:

- A local dirt sensor of the cleaner cannot tell
- Whether other squares are clean or not

Properties of task environments

- Deterministic vs. stochastic
  - Next state of the environment completely determined by the current state and the actions executed by the agent, then the environment is deterministic, otherwise, it is stochastic.
  - Strategic environment: deterministic except for actions of other agents

-Cleaner and taxi driver are:

  - Stochastic because of some unobservable aspects → noise or unknown

Properties of task environments

- Episodic vs. sequential
  - An episode = agent's single pair of perception & action
  - The quality of the agent's action does not depend on other episodes
    - Every episode is independent of each other
  - Episodic environment is simpler
    - The agent does not need to think ahead
- Sequential
  - Current action may affect all future decisions

-Ex. Taxi driving and chess.

Properties of task environments

- Static vs. dynamic
  - A dynamic environment is always changing over time
    - E.g., the number of people in the street
  - While static environment
    - E.g., the destination
- Semidynamic
  - Environment is not changed over time
  - But the agent's performance score does

Properties of task environments

- Discrete vs. continuous
  - If there are a limited number of distinct states, clearly defined percepts and actions, the environment is discrete
  - E.g., Chess game
  - Continuous: Taxi driving

Properties of task environments

- Single agent VS. multiagent
  - Playing a crossword puzzle – single agent
  - Chess playing – two agents
  - Competitive multiagent environment
    - Chess playing
  - Cooperative multiagent environment
    - Automated taxi driver
    - Avoiding collision

Properties of task environments

- Known vs. unknown

- This distinction refers not to the environment itself but to the agent's (or designer's) state of knowledge about the environment.
- -In known environment, the outcomes for all actions are given. (example: solitaire card games).
- - If the environment is unknown, the agent will have to learn how it works in order to make good decisions.(example: new video game).

Examples Of Task Environments

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	Strategic	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

Figure 2.6 Examples of task environments and their characteristics.

Structure of Intelligent Agents

- Agent = Architecture + Program
  - Architecture = some sort of computing device (sensors + actuators)
  - (Agent) Program = some function that implements the agent mapping = “?”
  - Agent Program = Job of AI

Agent programs

- Input for Agent Program
  - Only the current percept
- Input for Agent Function
  - The entire percept sequence
  - The agent must remember all of them
- Implement the agent program as
  - A look up table (agent function)

Agent programs

- Skeleton design of an agent program

```
function TABLE-DRIVEN-AGENT(percept) returns action
  static: percepts, a sequence, initially empty
          table, a table, indexed by percept sequences, initially fully specified

  append percept to the end of percepts
  action ← LOOKUP(percepts, table)
  return action
```

Agent Programs

- P = The set of possible percepts
- T= Lifetime of the agent
  - The total number of percepts it receives
- Size of the look up table
- Consider playing chess $\sum_{t=1}^{T} |P|^t$
  - P =10, T=150
  - Will require a table of at least 10^{150} entries

Agent Programs

- Despite of huge size, look up table does what we want.
- The key challenge of AI
  - Find out how to write programs that, to the extent possible, produce rational behavior
    - From a small amount of code
    - Rather than a large amount of table entries
  - E.g., a five-line program of Newton's Method
  - V.s. huge tables of square roots, sine, cosine, …

Types of agent programs

- Four types
  1. Simple reflex agents
  2. Model-based reflex agents
  3. Goal-based agents
  4. Utility-based agents

Simple reflex agents

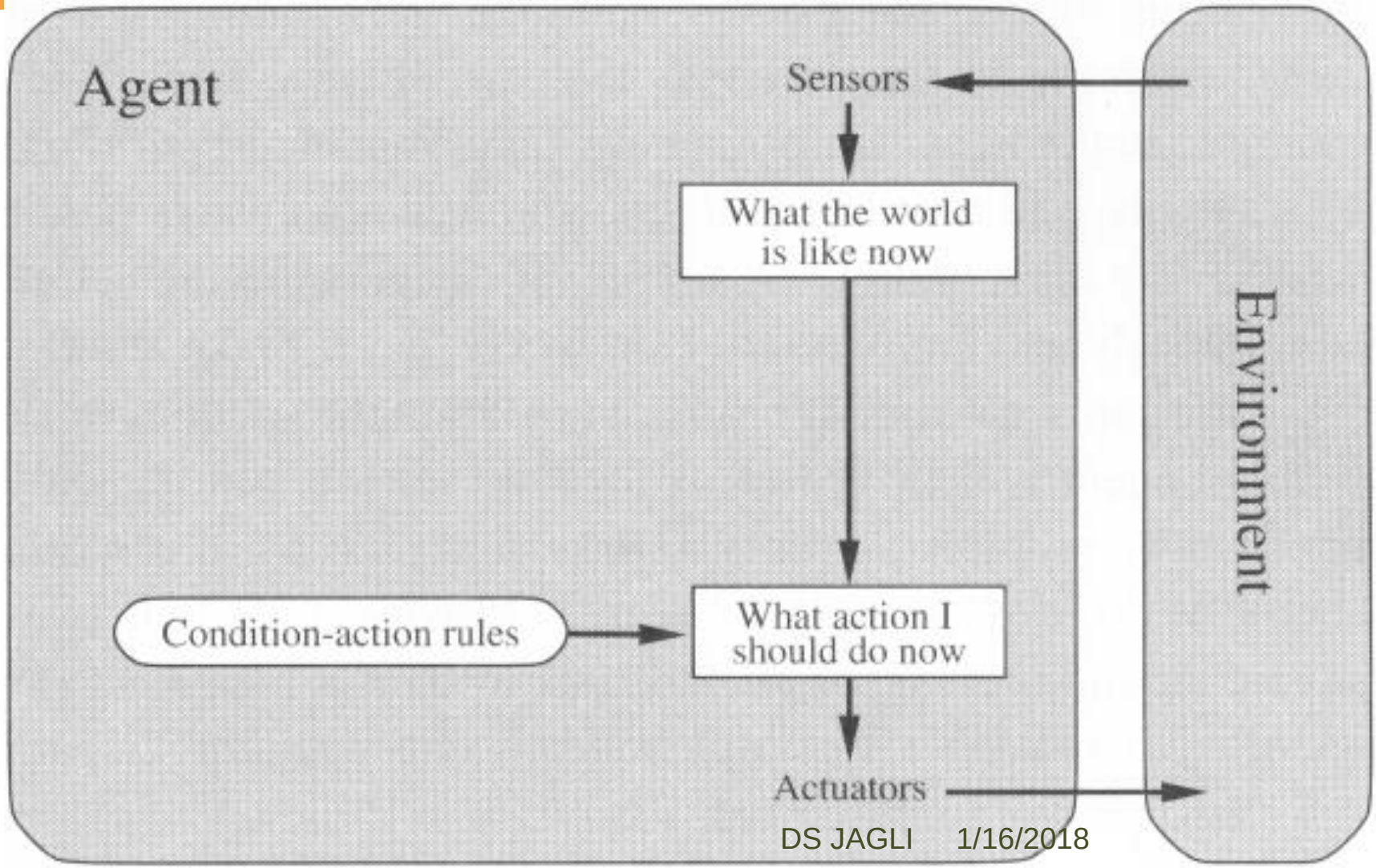
- It uses just ***condition-action rules***
  - The rules are like the form “if … then …”
  - Efficient but have narrow range of applicability
  - Because knowledge sometimes cannot be stated explicitly
  - Work only
    - If the environment is fully observable

Simple reflex agents


```
function SIMPLE-REFLEX-AGENT(percept) returns action
  static: rules, a set of condition-action rules

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

Simple reflex agents (2)



A Simple Reflex Agent in Nature

percepts
(size, motion)

RULES:

(1) If small moving object,
then activate SNAP

(2) If large moving object,
then activate AVOID and inhibit SNAP

ELSE (not moving) then NOOP

needed for completeness

Action: SNAP or AVOID or NOOP

Model-based Reflex Agents

- For the world that is partially observable
  - The agent has to keep track of an internal state
    - That depends on the percept history
    - Reflecting some of the unobserved aspects
    - E.G., Driving a car and changing lane
- Requiring two types of knowledge
  - How the world evolves independently of the agent
  - How the agent's actions affect the world

Example Table Agent With Internal State

IF	THEN
Saw an object ahead, and turned right, and it's now clear ahead	Go straight
Saw an object Ahead, turned right, and object ahead again	Halt
See no objects ahead	Go straight
See an object ahead	Turn randomly

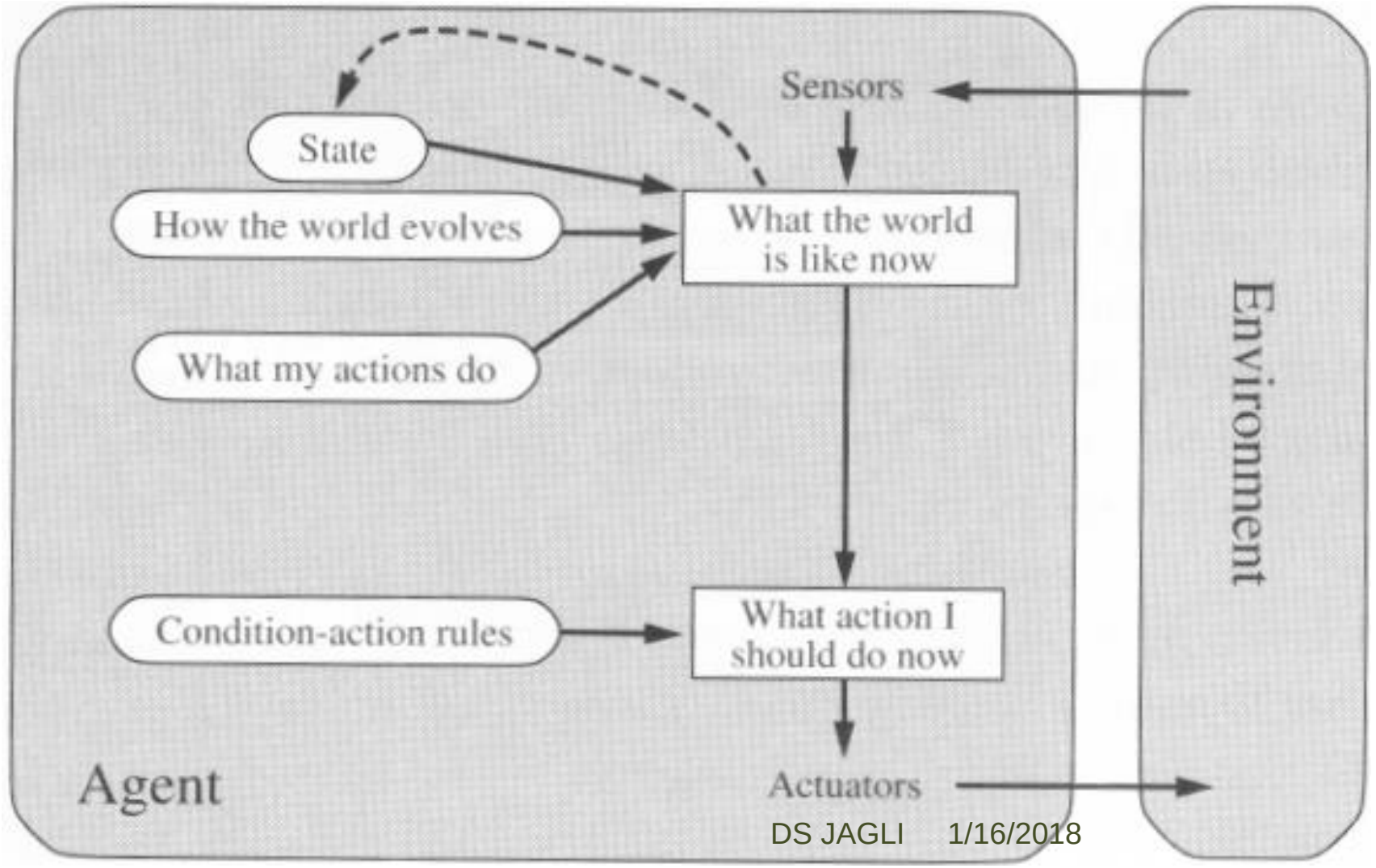
Model-based Reflex Agents


```
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
```

The agent is with memory

Model-based Reflex Agents



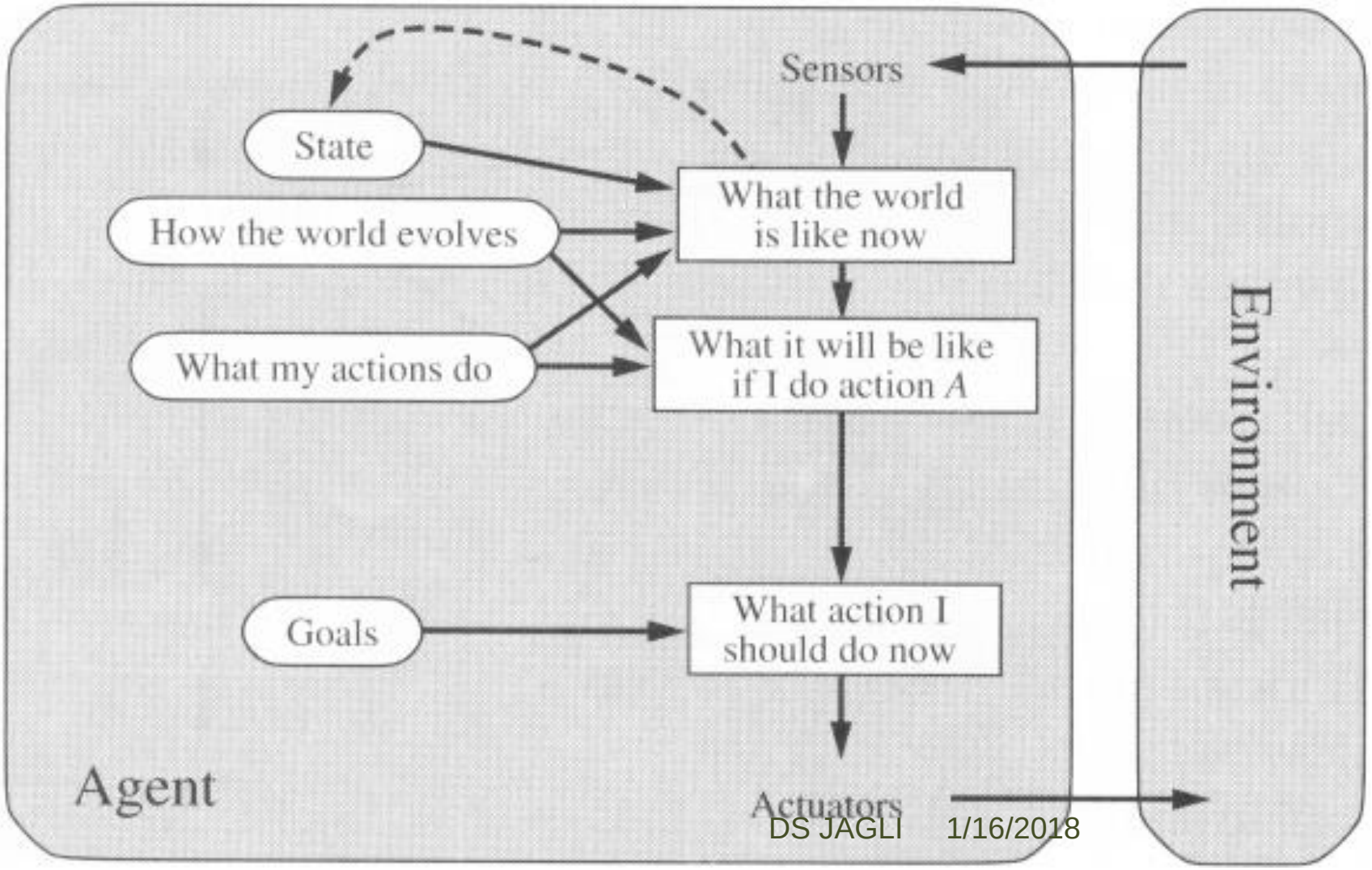
Goal-based agents

- Current state of the environment is always not enough
- The goal is another issue to achieve
  - Judgment of rationality / correctness
- Actions chosen → goals, based on
  - the current state
  - the current percept

Goal-based agents

- Conclusion
  - Goal-based agents are less efficient
  - but more flexible
    - Agent ← Different goals ← different tasks
  - Search and planning
    - two other sub-fields in AI
    - to find out the action sequences to achieve its goal

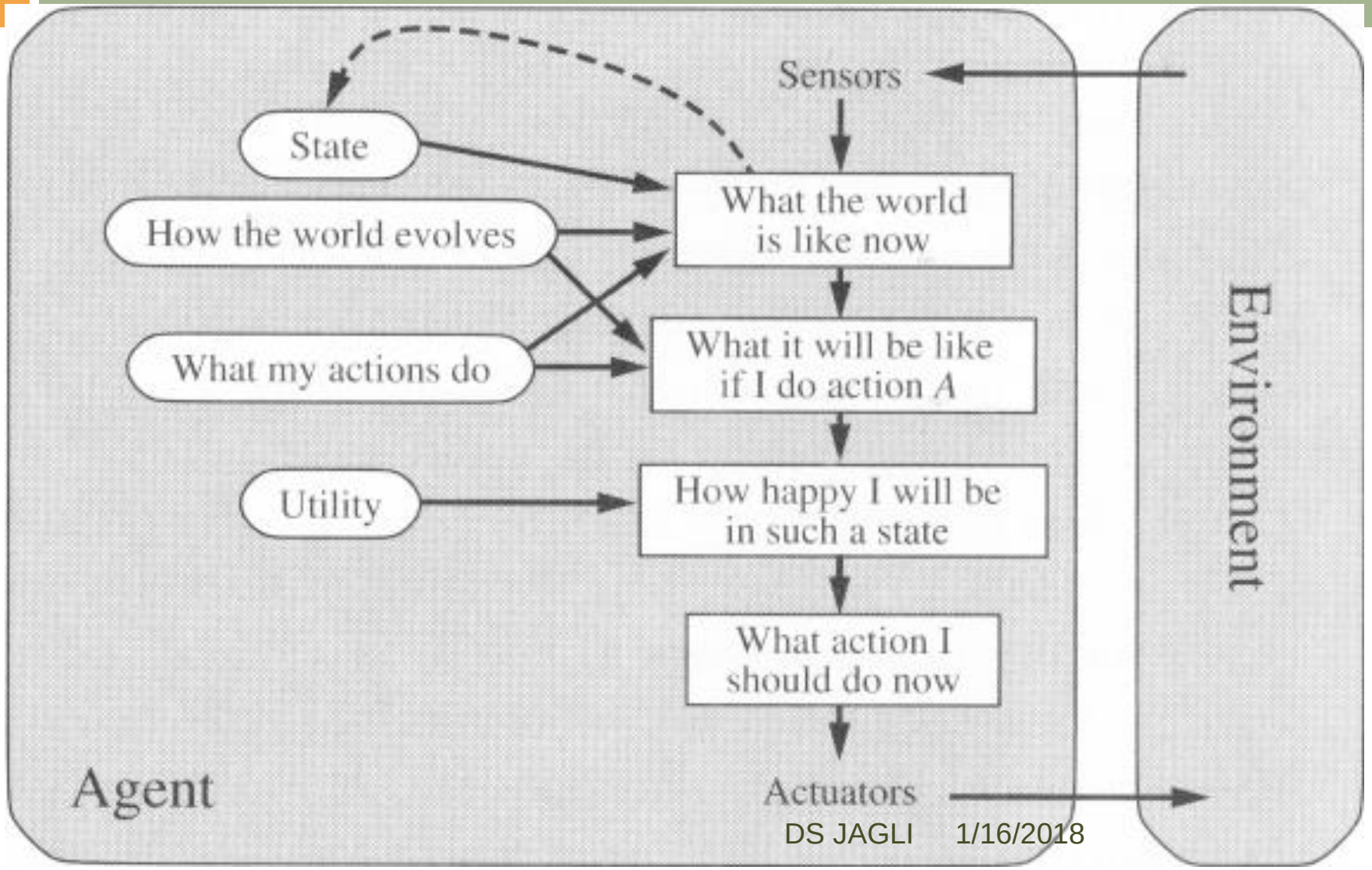
Goal-based agents



Utility-based agents

- Goals alone are not enough
  - To generate **high-quality** behavior
  - E.g. meals in Canteen, good or not ?
- Many action sequences → the goals
  - some are better and some worse
  - If goal means success,
  - then **utility** means the degree of success (how successful it is)

Utility-based agents



Utility-based agents

- It is said state A has higher utility
  - If state A is more preferred than others
- Utility is therefore a function
  - That maps a state onto a real number
  - The degree of success

Utility-based agents

- Utility has several advantages:
  - When there are conflicting goals,
    - Only some of the goals but not all can be achieved
    - utility describes the appropriate trade-off
  - When there are several goals
    - None of them are achieved **certainly**
    - utility provides a way for the decision-making

Learning Agents

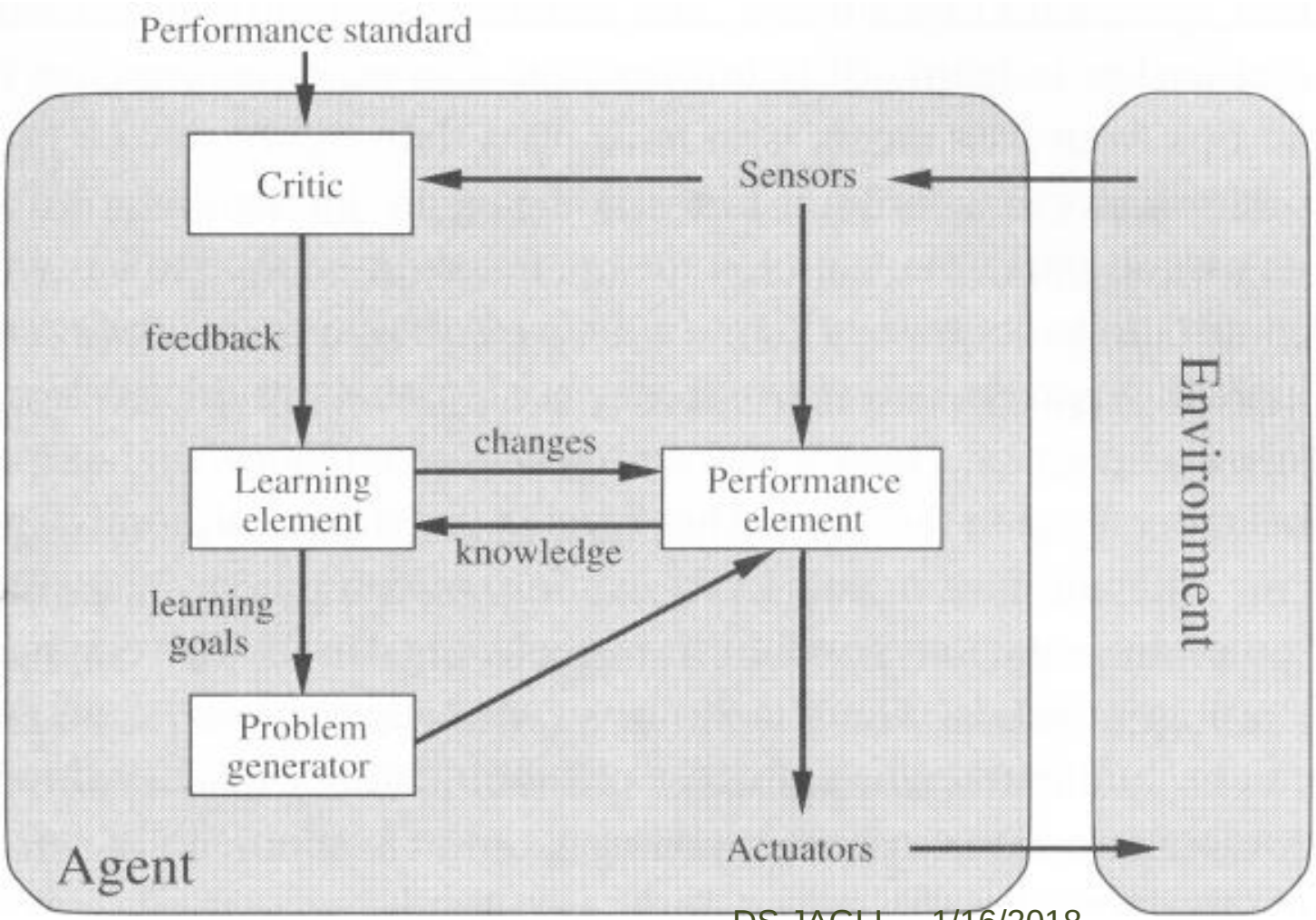
- After an agent is programmed, can it work immediately?
  - No, it still need teaching
- In AI,
  - Once an agent is done
  - We teach it by giving it a set of examples
  - Test it by using another set of examples
- We then say the agent learns
  - A learning agent

Learning Agents

- Four conceptual components
  - Learning element
    - Making improvement
  - Performance element
    - Selecting external actions
  - Critic
    - Tells the Learning element how well the agent is doing with respect to fixed performance standard.

    (Feedback from user or examples, good or not?)
  - Problem generator
    - Suggest actions that will lead to new and informative experiences.

Learning Agents



Summary

- agents perceive and act in an environment
- ideal agents maximize their performance measure
  - autonomous agents act independently
- basic agent types
  1. Simple reflex
  2. Reflex with state
  3. Goal-based
  4. Utility-based
  5. Learning
- some environments may make life harder for agents
  - inaccessible, non-deterministic, non-episodic, dynamic, continuous