Lecture 2: Intelligent Agents

Artificial Intelligence

CS-6364

Intelligent agents

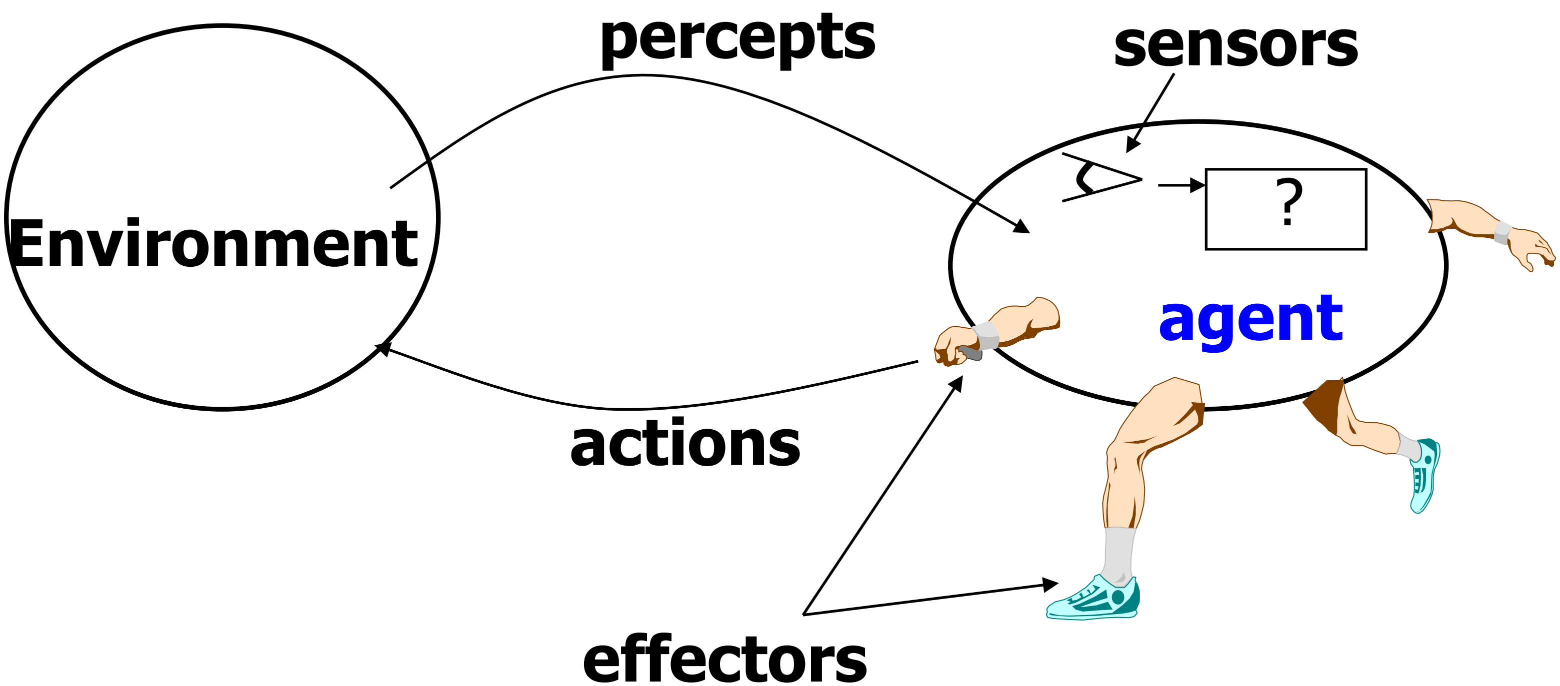
Intelligent agents:

In this lecture we look in detail at what an agent is, at the interaction between agent design and the environment in which the agent must operate, and at questions of empirical validation

What is <u>the structure</u> of an agent?

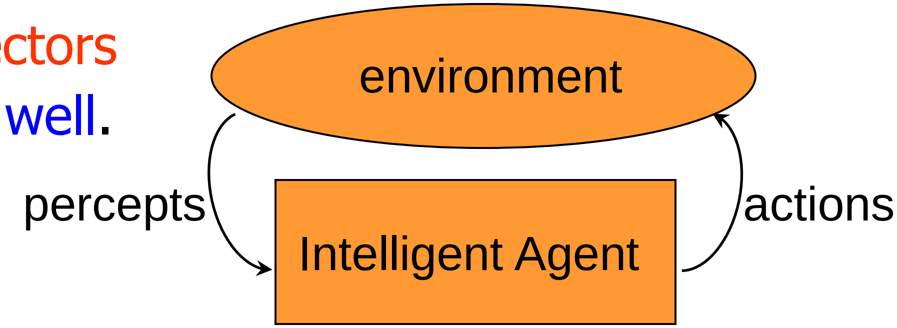
How does the environment affect how an agent acts?

What is an agent?



Remember: Intelligent agents

- <u>The Agent is in an environment</u>
- perceives environment through sensors
- acts on environment through effectors
- goal: design agents that perform well.



Intelligence is based on: (1) perception and (2) capability of actions.

Agents must perform actions to obtain useful information (acquire more percepts).

An agent may be regarded as a mapping of percepts into actions.

– A **<u>performance measure</u>** is needed to determine how successful an agent is.

An <u>ideal rational agent</u> provides an action that maximizes its performance measure, based on the percepts received and its built-in knowledge.

Autonomous agents

We want to build intelligent agents capable of reacting to the environment on their own. That is to say that we want agents to be able to <u>learn from their experiences</u> and face new situations.

This is different than programming extensively, hoping that the programmer foresaw all possible situations.

- An agent is autonomous when its behavior is determined by its own experience. (autonomous from other agents or humans)

Example: a clock adjusting to time zones, a self-driving car that finds a road block.

Intelligent Agents vs Other Software

- Agents are autonomous
- Agents contain some level of intelligence
- Agents react to environments, and can sometimes take proactive actions
- Agents may have social ability; ie communicate with user and other agents
- Agents may cooperate with other agents
- Agents may migrate from system to system

Structure of Intelligent Agents

Agents are built with programs running on some hardware. AI systems may be general-purpose computers, or special-purpose computers.

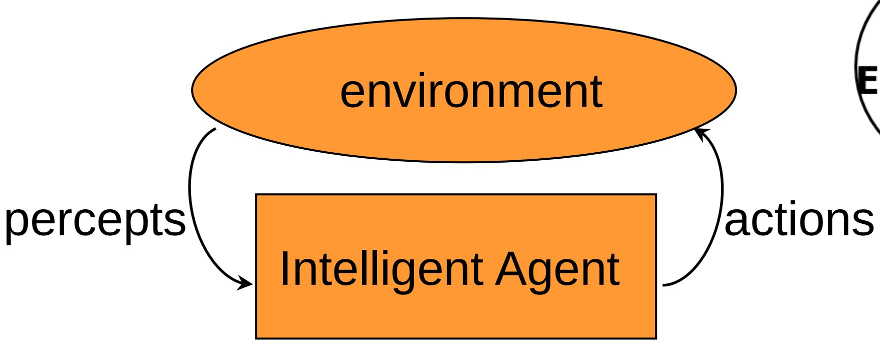
Agent programs are very complex, because intelligence is complex. Simple look up tables mapping percepts into actions will not work.

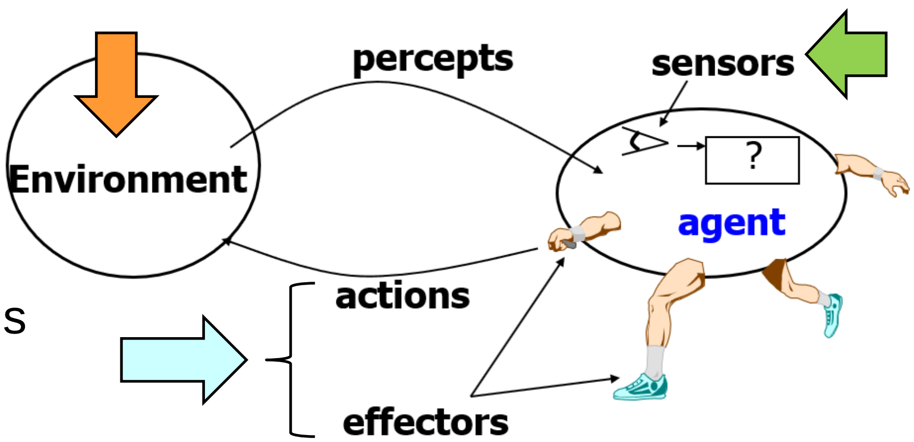
Example: A chess player would require 35^{100} entries.

Specify the TASK environment

PEAS: **P**erformance, **E**nvironment, **A**ctuators, **S**ensors.

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





Agent Types and their Descriptions

PEAS: **P**erformance, **E**nvironment, **A**ctuators, **S**ensors..

Agent Type	Performance Measure	Environment	Actuators	Sensors
Medical diagnosis system	Healthy patient, minimize costs, lawsuits	Patient, hospital, staff	Display questions, tests, diagnoses, treatments, referrals	Keyboard entry of symptoms, findings, patient's answers
Satellite image analysis system	Correct image categorization	Downlink from orbiting satellite	Display categorization of scene	Color pixel arrays
Part-picking robot	Percentage of parts in correct bins	Conveyor belt with parts; bins	Jointed arm and hand	Camera, joint angle sensors
Refinery controller	Maximize purity, yield, safety	Refinery, operators	Valves, pumps, heaters, displays	Temperature, pressure, chemical sensors
Interactive English tutor	Maximize student's score on test	Set of students, testing agency	Display exercises, suggestions, corrections	Keyboard entry

Properties of ***environments***

- **<u>Fully vs partially observable</u>**: **Fully observable** if the sensors give access to the environment at each point in time. **Partially observable** if the sensors are noisy or inaccurate or parts of the environment states are missing. If the agent has no sensors, the environment is **unobservable**.

- **<u>Deterministic vs. stochastic:</u>** if the next state of the environment is completely determined by the *<u>current state + the action executed by the agent</u>* – it is **deterministic**. Otherwise, it is **stochastic**.
  - we say that an environment is **uncertain** if it is not fully observable or not deterministic.
  - A **nondeterministic** environment is one in which the actions are characterized by their possible outcomes, but no probabilities are attached to them.

*Properties of **environments** - 2*

- **Episodic vs sequential**: in an **episodic environment** the agent's *experience* is divided into atomic episodes. In each episode, the agent receives a percept and then performs a single action. Crucially, the next episode does not depend on the action taken in the previous episode. In **sequential environments** the current action decision could affect all future action decisions.

- **Static vs. dynamic:** if the environment can change while the agent is deliberating on what action to take, then we have a **dynamic environment**, otherwise it is **static**.
- Discreet vs. continuous
- Single-agent vs. multi-agent
  - Competitive vs. cooperative

Properties of ***environments*** - 3

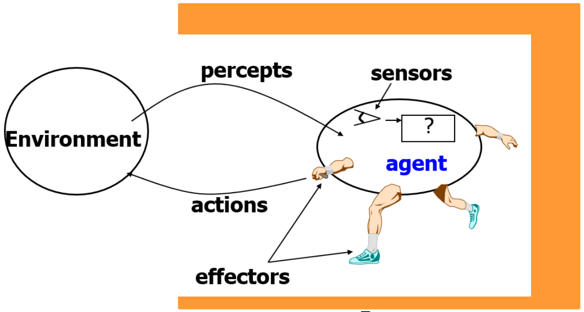
- **<u>Discreet vs continuous</u>**: the discreet/continuous distinction applies to <u>the state of the environment : </u>(a) the way time is handled; and (b) to the percepts and actions of the agent. IF we have a finite number of actions and percepts – it is a **discrete** environment, otherwise it is a **continuous** one.
- **<u>Known vs. unknown: </u>**it refers to the agent's state of knowledge about the "laws of physics" of the environment.
- **<u>Single vs. Multi-agent</u>**: in case of multi-agent:
  - Competitive vs. cooperative

Properties of environments: examples

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

What is the structure of an intelligent agent?

agent = architecture + program



program:

a function that maps percepts to actions

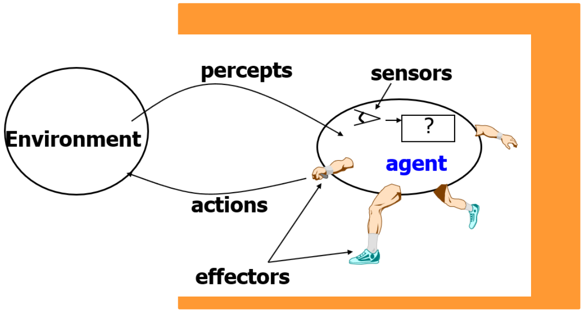
architecture:

- where the program runs: HW+SW
- makes percepts available to program
- runs program
- feeds actions from program to effectors

Agent Programs

agent program→

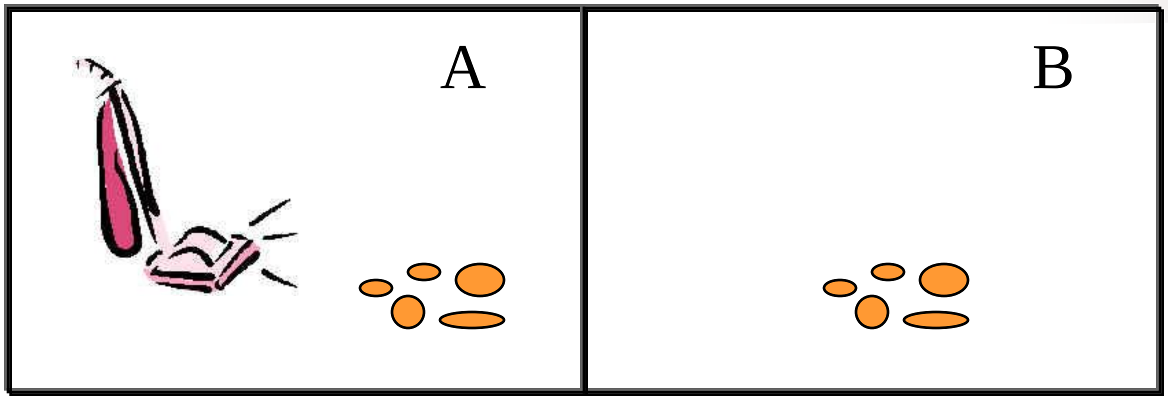
Takes the *current percept* as input from the sensors & returns an *action to the actuators* (effectors).



Example of agent program: **The table-driven-Agent**

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

Two simple examples:



Vacuum-cleaner world with 2 locations

Percepts: Clean, Dirty

Percept Sequence:
[A, Clean]
[B, Dirty]
[B, Clean]
[A, Dirty]
[A, Clean],
[B, Clean],
....
[A, Clean]

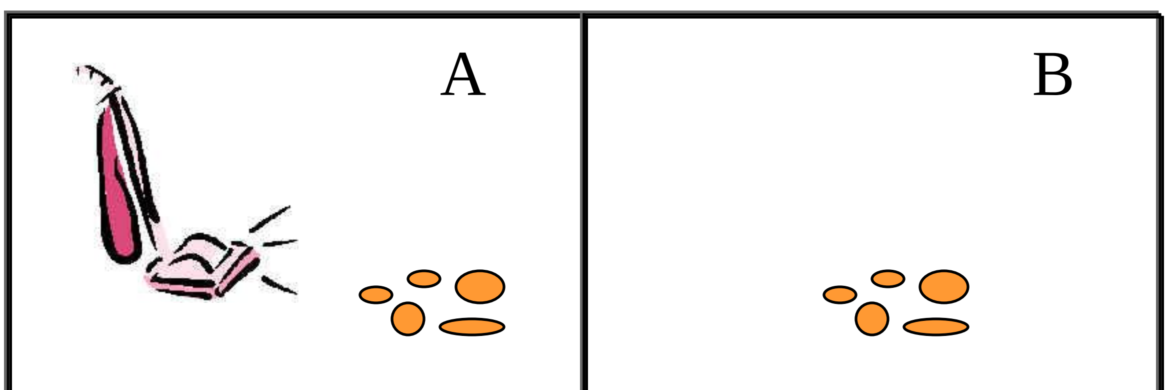
Actions: Move-Left, Move-Right,
Suck-dirt, Do-nothing

Simple agent function:
If (dirty) Suck-dirt
Else move-other-location

Agent Tabulation

Percept Sequence:
[A, Clean]
[B, Dirty]
[B, Clean]
[A, Dirty]
[A, Clean],
[B, Dirty]
....
[A,Dirty]

Action Sequence:
Move-Right
Suck-Dirt
Move-Left
Suck-Dirt
Move-Right
Suck-Dirt
....
Suck-Dirt



Kitty-robot

The Techno Kitty is a true robot.

She meows, walks, cries, wiggles her tail, purrs

Has state-of-the-art sensor technology

Hello Kitty Robot

Kitty Robot as table-driven agent

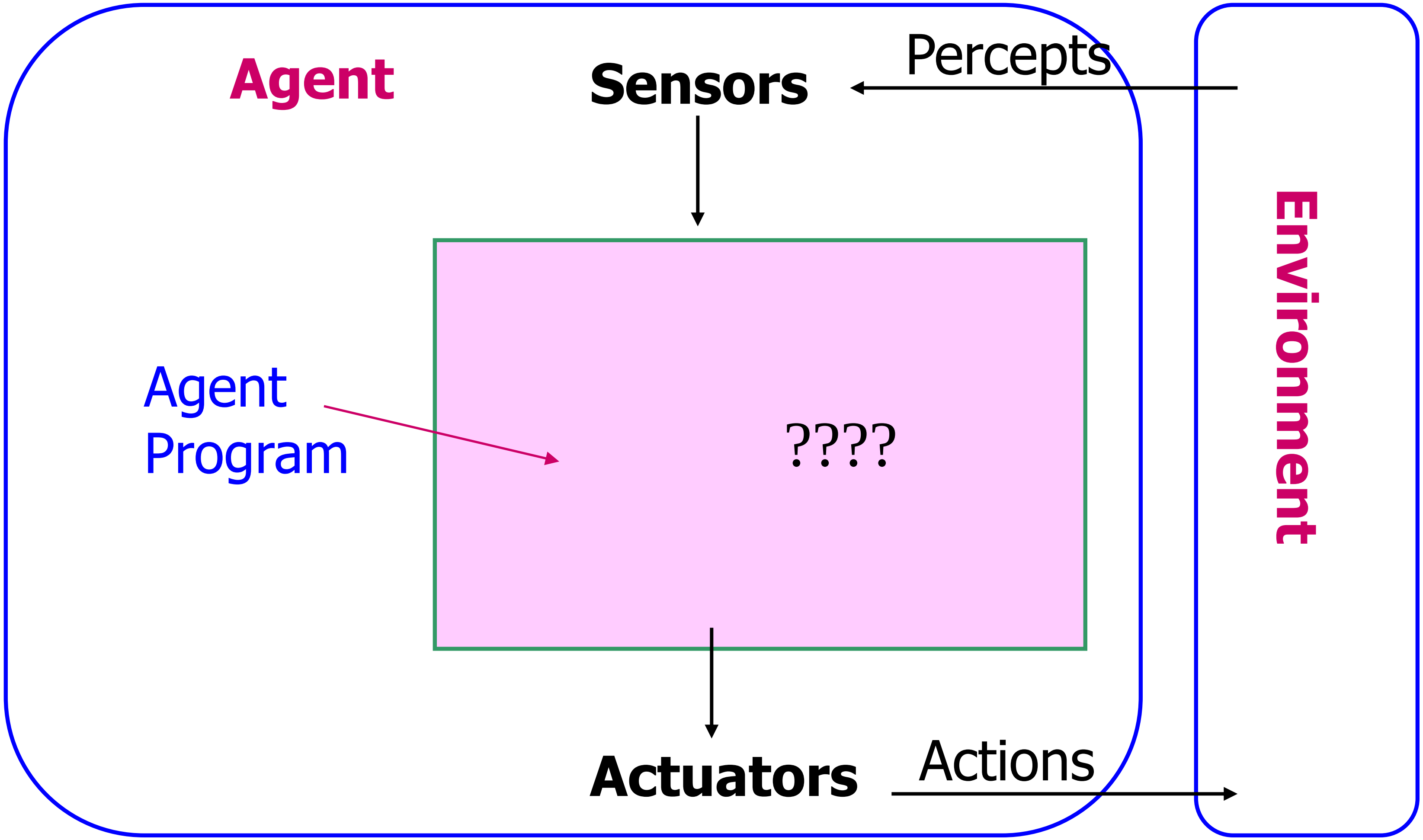
Percepts: Clap, Pet, Bump

Actions: Walk, Purr, Meow, Blink, Stop

Percept Sequence:	Actions:
[Clap]	Meow
[Clap, Clap]	Walk
[Pet][Pet]	Purr
[Clap][Bump]	Walk, Blink, Stop
[Clap][Clap][Bump]	Walk, Meow, Stop
[Clap][Clap][Pet]	Walk, Stop, Purr, Walk
[Clap][Bump][Pet]	Walk, Blink, Purr, Stop

Table of actions

Agents



Agent
Sensors
Percepts
Environment
Agent
Program
????
Actuators
Actions

Table-driven agent

function TABLE-DRIVEN-AGENT(*percept*)
returns action
static: *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 ← CHOOSE-BEST-ACTION(*memory*)
return *action*

How is  working?

function TABLE-DRIVEN-AGENT(*percept*)
returns action
static: *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 ← CHOOSE-BEST-ACTION(*memory*)
return *action*

Percept Sequence:	Actions:
1. [Clap] weight=1	Meow
2. [Clap, Clap] w=3	Walk
3. [Pet][Pet] w=13	Purr
4. [Pet] w=4	Stop
5. [Clap][Pet] w=8	Purr, Walk
6. [Clap][Clap][Bump] w=12	Walk, Meow, Stop
7. [Clap][Clap][Pet] w=10	Walk, Stop, Purr, Walk
8. [Clap][Bump] w=15	Walk, Blink, Stop

← TABLE

Percepts:
1| Clap
2| Clap
3| Pet
4| Clap
5| Bump
6| Pet
7| Pet
8| Clap

Choose Best Action:
Walk,Stop,Purr,Walk; w=10
Walk,Blink,Stop; w=15
Purr; w=13
Meow; w=1; TOTAL=39

The Structure of Intelligent Agents

The key challenge in AI is to find out how to write programs that, to the extend possible, produce rational behavior from a smallish program rather than from a vast table!!!!!

The question facing the designer is: **How to structure (architect) an agent?**

Four types of agents (in increasing complexity order)

- Simple reflex agents
- Model-based reflex agents
- Goal-based agents
- Utility-based agents

A Simple Reflex agent

Agent Type 1

Selects actions based on the current percept

Uses condition-action rules, productions.

In humans, condition-action rules are both learned responses and innate reflexes (e.g. blinking)

if light-is-green then accelerate

if light-is-red then brake

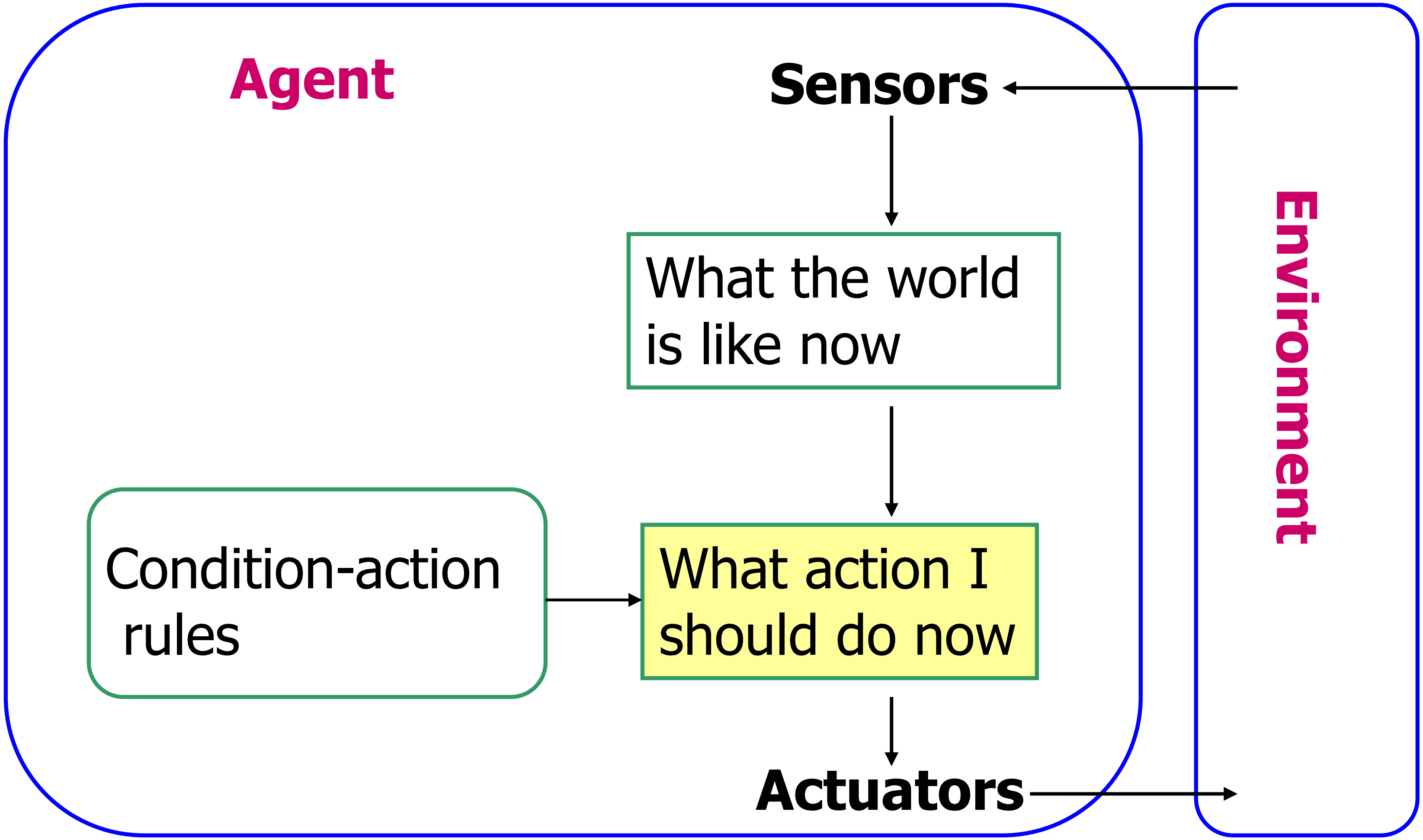
Real life examples

Example:

- Is driving purely a matter of reflex?
  What happens when making a lane change?
  Must remember what you saw in the
  rear-view window

- Grocery shopping: if milk-carton then put in basket.
  What happens when you get to the dairy section?
  Can't remember what is already in the basket.

Reflex agent architecture



Reflex Agent program

```
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
```

Vaccum-Agent

function REFLEX-VACUUM-AGENT(*location,status*)
returns an action
if *status = Dirty* **then return** *Suck-dirt*
else if *location = A* **then return** *Move-Right*
else if *location = B* **then return** *Move-Left*

Percept Sequence:	Action Sequence:
[A, Clean]	*Move-Right*
[B, Dirty]	*Suck-Dirt*
[B, Clean]	*Move-Left*
[A, Dirty]	*Suck-Dirt*
[A, Clean],	*Move-Left*
[B, Dirty]	*Suck-Dirt*
……………	*……………*
[A,Dirty]	*Suck-Dirt*

Only the current percept is considered!!!!

Give rules to Kitty

Percepts: Clap, Pet, Bump

Actions: Walk, Purr, Meow, Blink, Stop

If percept = Clap THEN walk or [Meow,Walk]

If percept = Pet THEN [Purr] or [Purr,Meow]

If percept=Bump THEN [Blink, Stop] or [Meow, Walk, Stop]

Percepts:	Actions:
1| Clap	1| Meow, Walk
2| Clap	2| Meow, Walk
3| Pet	3| Purr
4| Clap	4| Meow, Walk
5| Bump	5| Blink, Stop
6| Pet	6| Purr
7| Pet	7| Purr, Meow
8| Clap	8| Meow, Walk

Model-based Reflex Agents

Keep track of the part of the world it can't see now
- maintain some sort of internal state

Internal state depends on percept history, reflects some unobservable aspects of the current state (because of past action)

Model-based agents

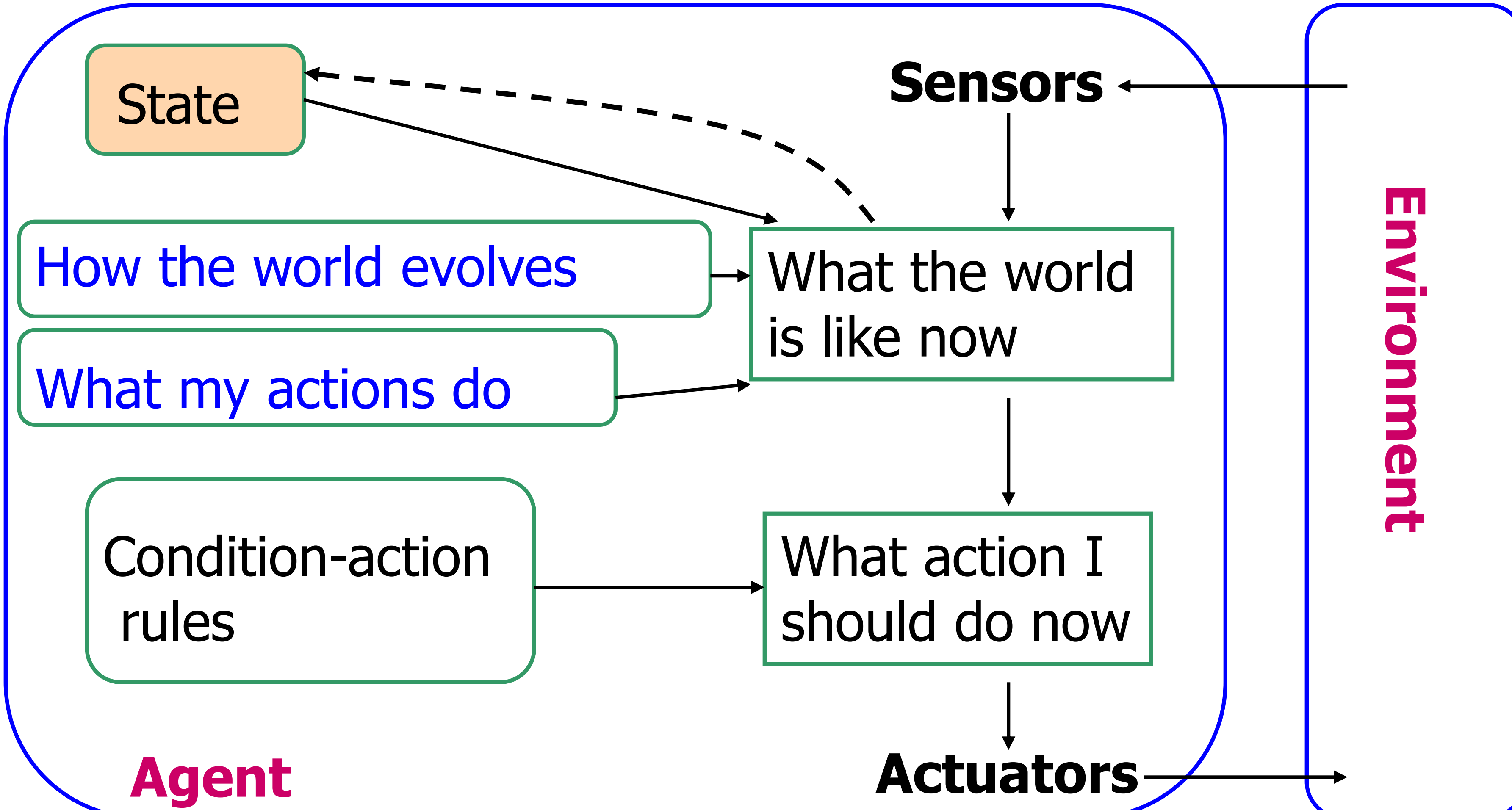
Agent Type 2

Updating the state information as time goes by requires 2 kinds of knowledge:

1. information about how the world <u>evolved</u> independently of the agent
2. information about how the agent's actions <u>affect</u> the world

- Knowledge about how the world works is called a model of the world

Model-Based Reflex Agents
The architecture



Agent Program-
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
action, the most recent action, initially none

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

Kitty with a state

- Let us model the state by two variables:
  - The position of Kitty
  - The "happiness" of Kitty

1	2	3	4	5	6

The position

Very happy	Happy	Sad	Curious

The happiness

Initial state: [6, Curious]
Number of states = 24

State Update for Kitty

1	2	3	4	5	6

The position

Very happy	Happy	Sad	Curious

The happiness

Percepts: Clap, Pet, Bump

Actions: Walk, Purr, Meow, Blink, Stop

state ←UPDATE-STATE(*previous-state, percept, action*)

Happiness +1 ←UPDATE-KITTY[Happines,Clap]
Happiness +2 ←UPDATE-KITTY[Happines,Pet]
Happiness -1 ←UPDATE-KITTY[Happines,Bump]

Position-1 ←UPDATE-KITTY[Position,Walk]
Position ← UPDATE-KITTY[Position,Purr]
Position ← UPDATE-KITTY[Position,Meow]
Position ← UPDATE-KITTY[Position,Blink]
Position ← UPDATE-KITTY[Position,Stop]

Because Kitty's position changes only when she walks – the update-state function needs to consider the previous action!!!

Kitty has a model

The state: {1,2,3,4,5,6} × {VH, H, S, C}

S0=[6,C]

Percepts: Clap, Pet, Bump

Actions: Walk, Purr, Meow, Blink, Stop, Start

STATE-UPDATE

state ←UPDATE-STATE(*previous-state, percept, action*)

Happiness +1 ←UPDATE-KITTY[Happines,Clap]
Happiness +2 ←UPDATE-KITTY[Happines,Pet]
Happiness -1 ←UPDATE-KITTY[Happines,Bump]
Position-1 ←UPDATE-KITTY[Position,Walk]
Position ← UPDATE-KITTY[Position,Purr|Meow|Blink|Stop|Start]

RULES:

If percept = Clap THEN
  if hapiness>S THEN walk
    else [Meow,Walk]

If percept = Pet THEN
  if hapiness>S THEN [Purr]
    ELSE [Purr,Meow]

If percept=Bump THEN
  if happiness = VH
    THEN [Blink, Stop]
    ELSE [Meow, Walk, Stop]

Percepts:		Action:		New State:
1| Clap		1| Meow, Walk		1| [5,S]
2| Clap		2| Meow, Walk		2| [4,H]
3| Pet		3| Purr		3| [4,VH]
4| Clap	→	4| Walk	→	4| [3,VH]
5| Bump		5| Blink, Stop		5| [3,H]
6| Pet		6| Purr		6| [3,VH]
7| Pet		7| Purr		7| [3,VH]
8| Clap		8| Walk		8| [2,VH]

Goal-based agents

Agent Type 3

Knowing current state is not always enough
- agents needs to know also the goal

State allows an agent to keep track of unseen parts of the world, but the agent must update state based on knowledge of changes in the world and of effects of own action

Goal = description of desired situation

Goal-based agents

Another enhancement is to give the agent a goal to look for. The agent actions constitute a sequence that leads to the goal.

– Application examples: Searching, Planning.

- Goal information describes desirable situations.

- The actions now are not provided by if-then rules, but they are selected such that will bring the system closer to the goal.

- States are still necessary. Note that the way actions are selected to get closer to the goal, may still use if-then rules (but not exclusively).

Examples of Goal-based agents

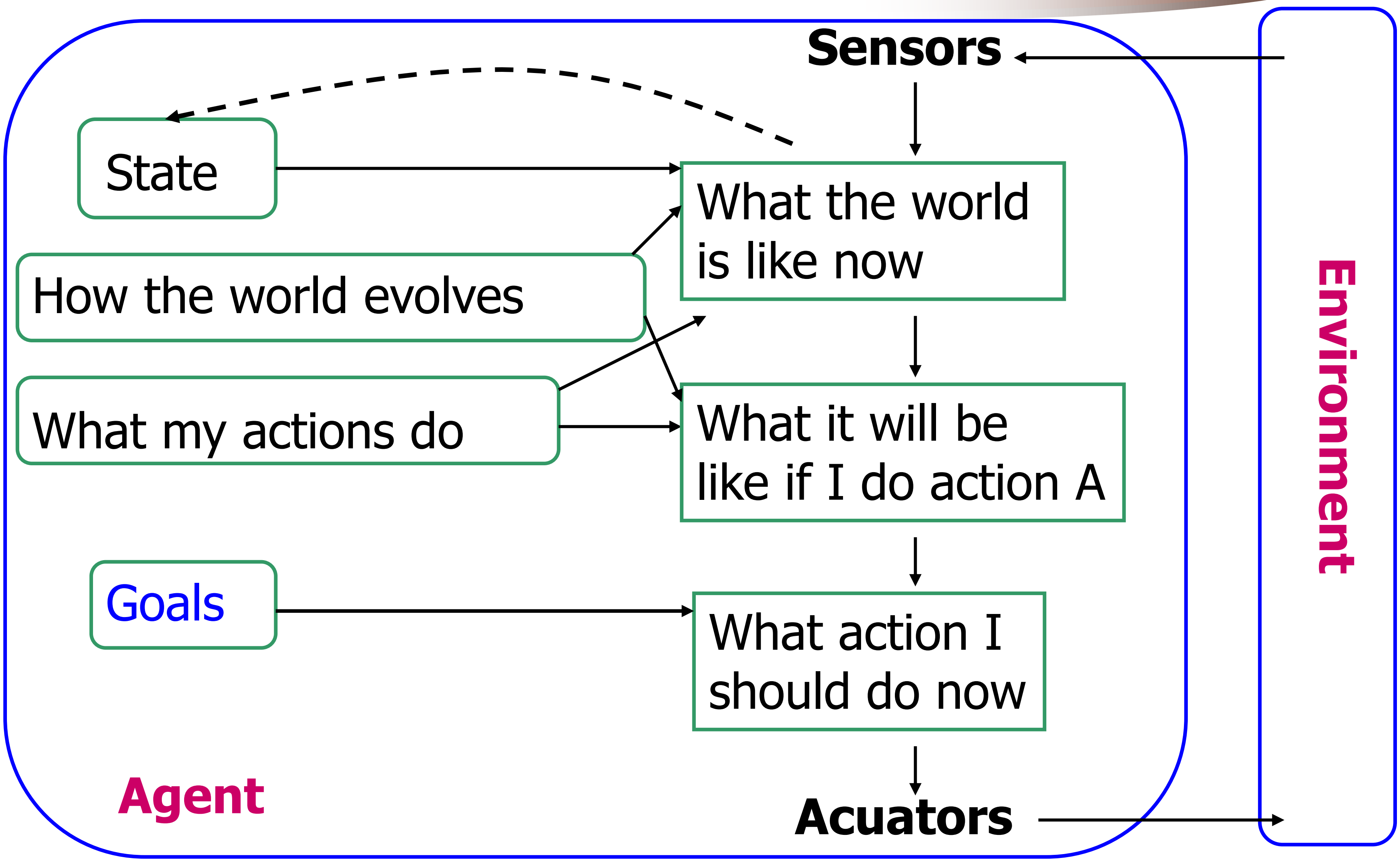
Self-driving cars:

- decision to change lanes depends on the goal to go somewhere (and other factors)

Alexa shopping agent:

- shopping well depends on a shopping list, knowledge of menu, funds available

Diagram of Goal-based agents



Utility-based agents

Agent Type 4

goal = preferred state, *- it is not enough for high-quality behavior*

utility= weight options, *(degree of preference) by mapping a state into a number*

Example: quicker, safer, more reliable ways to get (robot taxi-driver)

Preferred world state has higher utility for agent = quality of being useful

Utility-based agents (more)

Utility function: state $\rightarrow$ U(state)

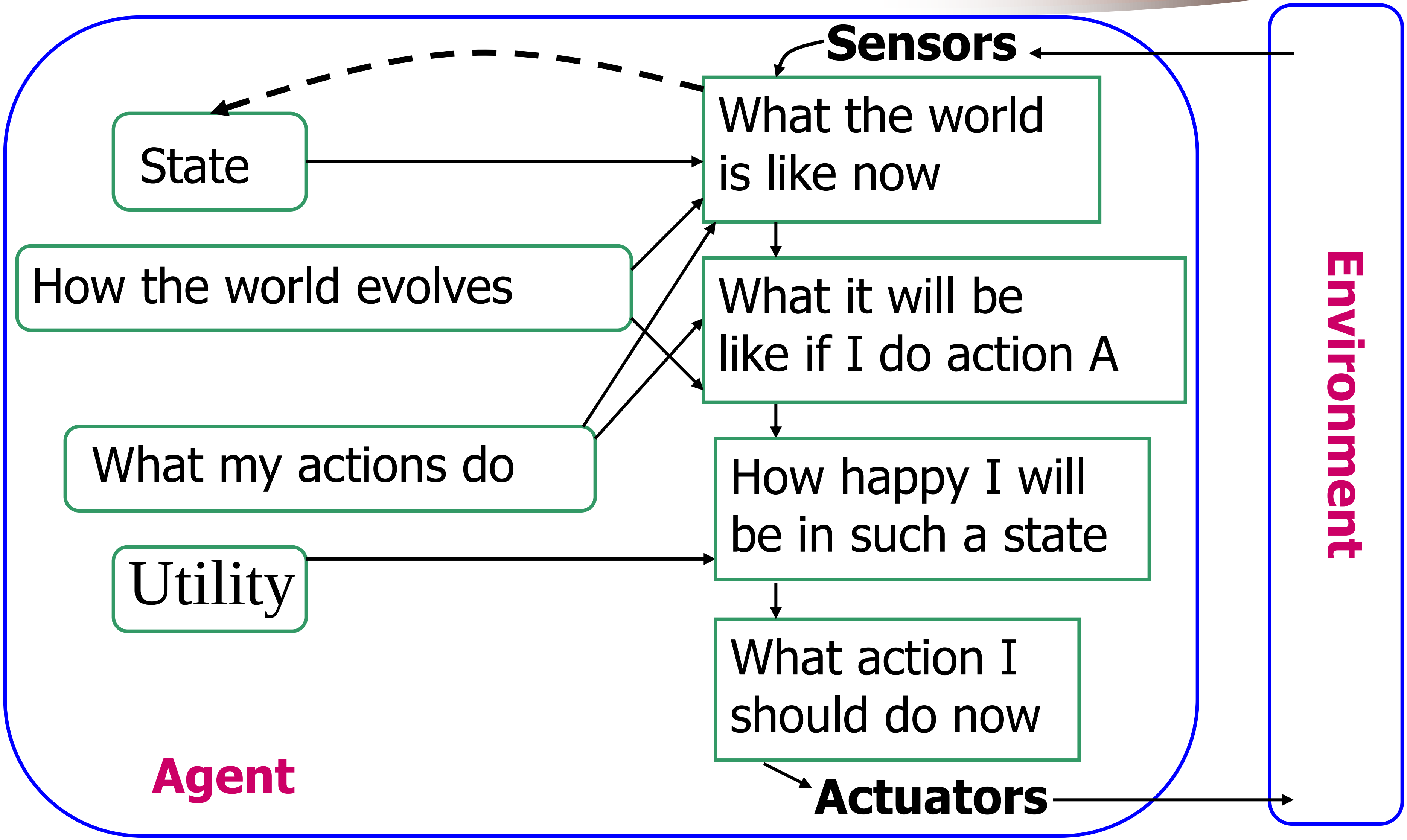
= measure of happiness

Kinds of decisions allows:

- choice between conflicting goals and
- choice between likelihood of success and importance of goal (if achievement uncertain)

Search (goal-based) vs. games (utilities)

Utility-based architecture



Learning agents

Agent Type 5

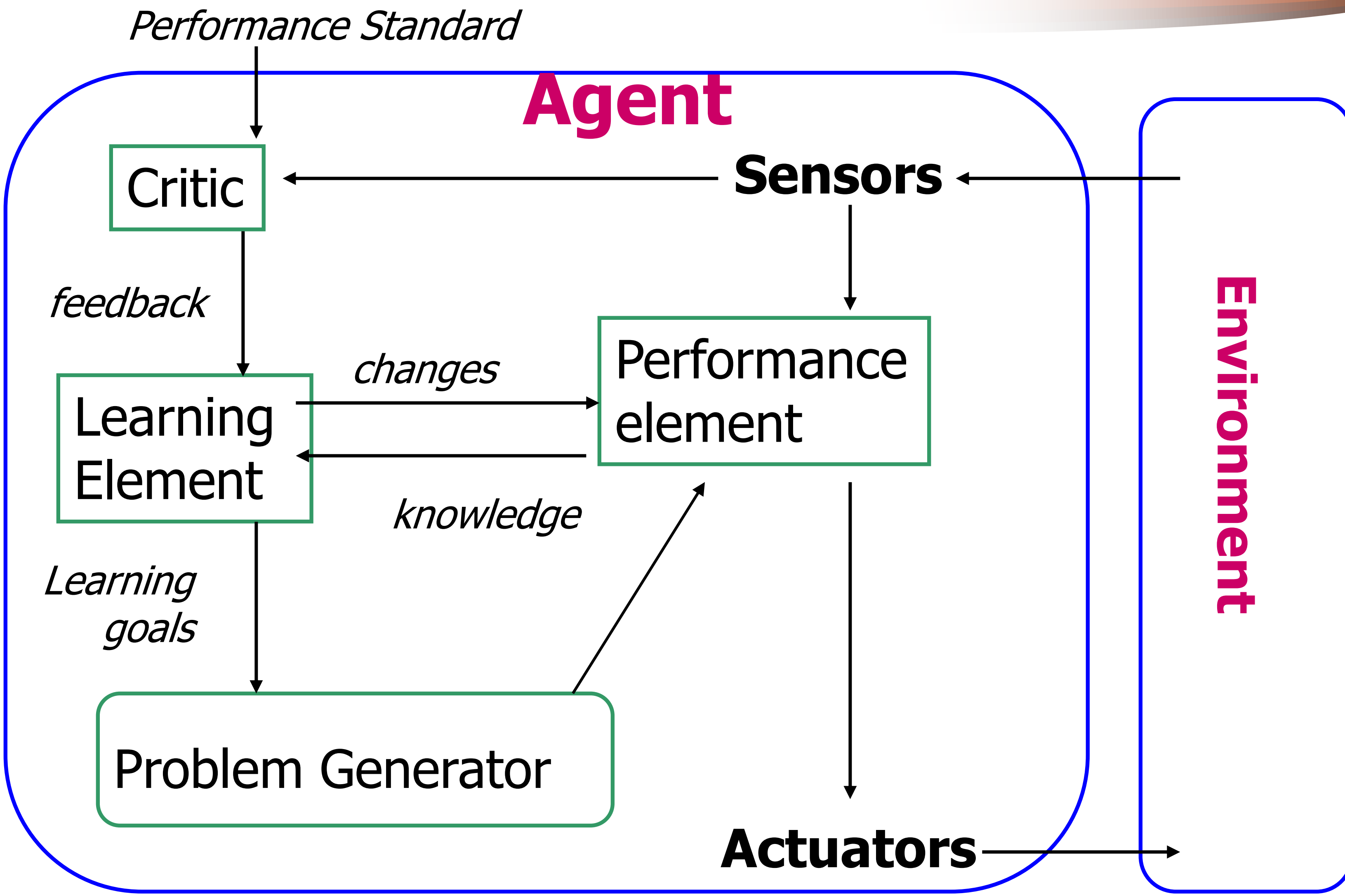
Learning element = *it is responsible for making improvements of the agent*

Performance element= *responsible for selecting external actions*

Critic= *gives feedback to the learning element on how the agent is doing, with respect to a performance standard*

Performance generator= *responsible for suggesting action that will lead to informative experiences*

Learning agent architecture



Next generation agents

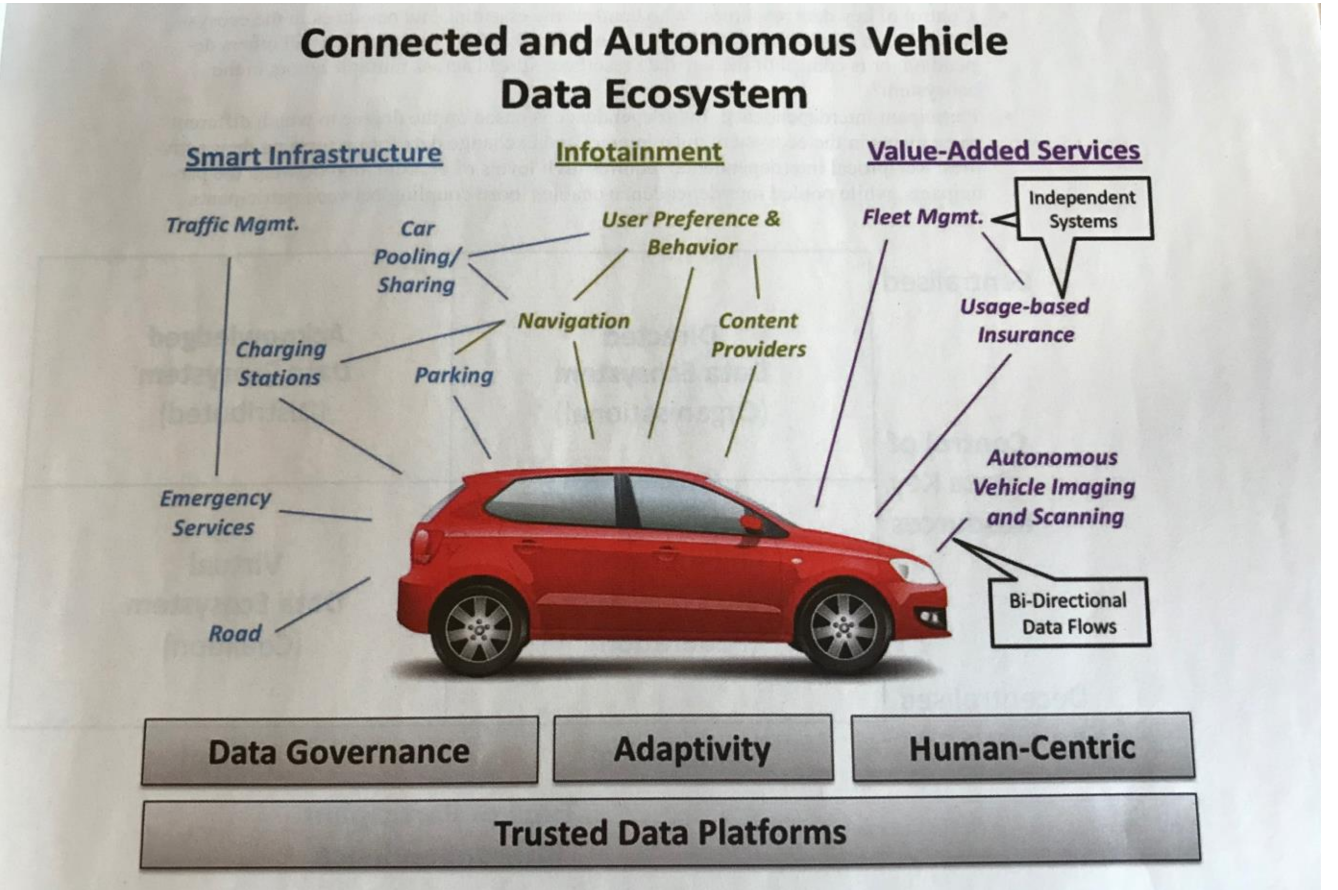


Figure 1. Connected and Autonomous Vehicle Data Ecosystem

Take-home Quiz 2

- due on 9/4 in eLearning :
  - Each of you will read section 2.4.7 from the Textbook and represent Kitty world as
    - Atomic representation
    - Factored representation
    - Structured representation
  - Write the pseudo-code of an utility-based agent for each of these representations.
- Where can I play with some agents?
  - STARTING POINT: AIMA code