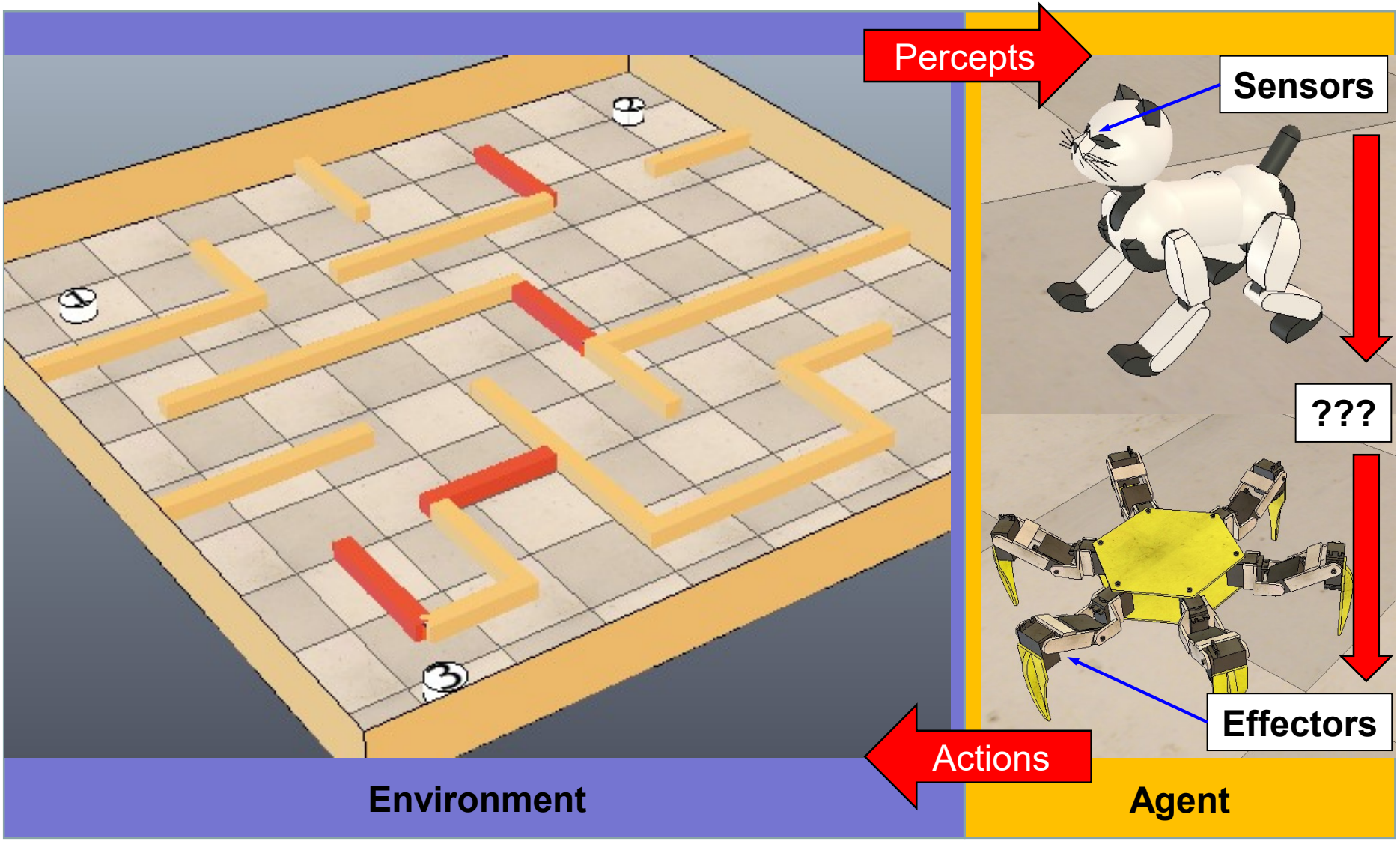
Artificial Intelligence

Intelligent Agents

Chapter 2, AIMA

An Agent



An Agent

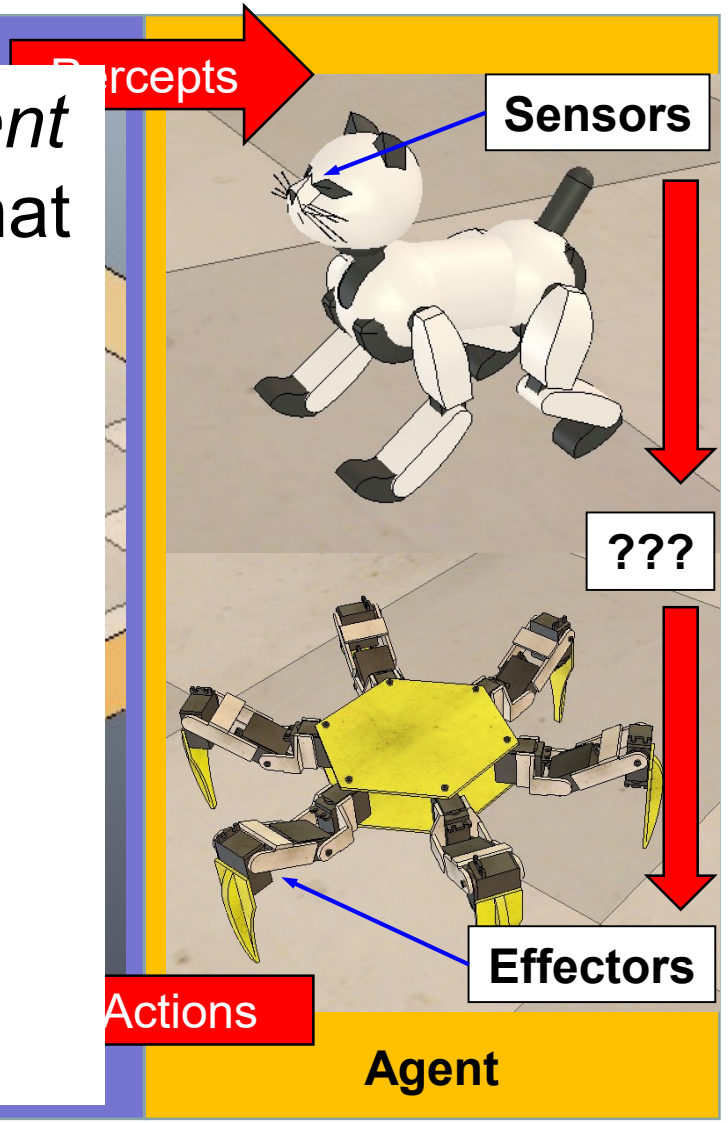
An *agent* perceives its *environment* through *sensors* and acts upon that environment through *effectors*

$$\mathbf{x}(t) = \begin{pmatrix} x_1(t) \\ x_2(t) \\ \vdots \\ x_D(t) \end{pmatrix} \qquad \boldsymbol{\alpha}(t) = \begin{pmatrix} \alpha_1(t) \\ \alpha_2(t) \\ \vdots \\ \alpha_K(t) \end{pmatrix}$$

Percepts Actions

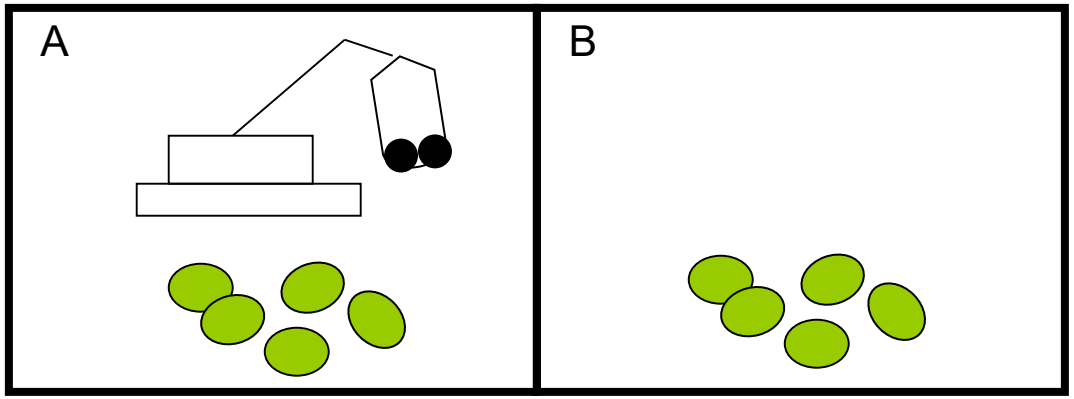
Agent function:

$$\boldsymbol{\alpha}(t) = f[\mathbf{x}(t), \mathbf{x}(t-1), \ldots, \mathbf{x}(0)]$$



Example: Vacuum Cleaner World

Image borrowed from V. Pavlovic, Rutgers

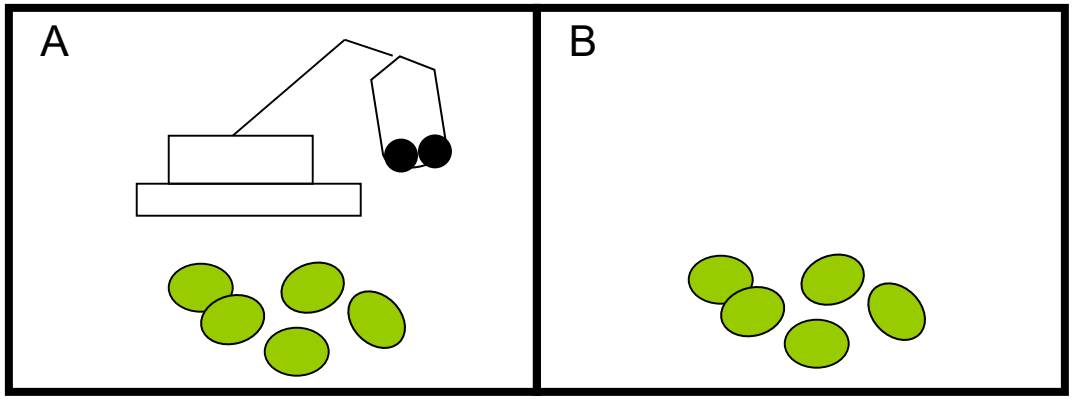


Percepts:

Actions:

Example: Vacuum Cleaner World

Image borrowed from V. Pavlovic, Rutgers

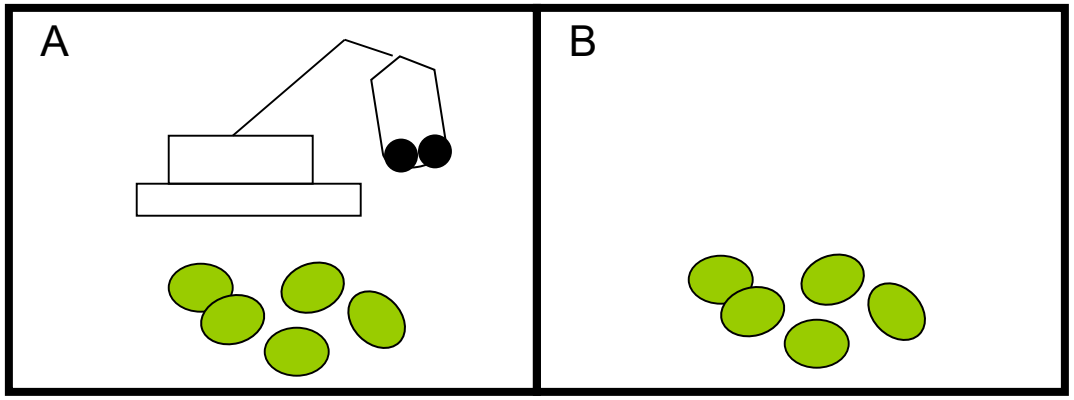


Percepts: $x_1(t)$ $x_2(t)$

Actions: $\alpha(t)$

Example: Vacuum Cleaner World

Image borrowed from V. Pavlovic, Rutgers

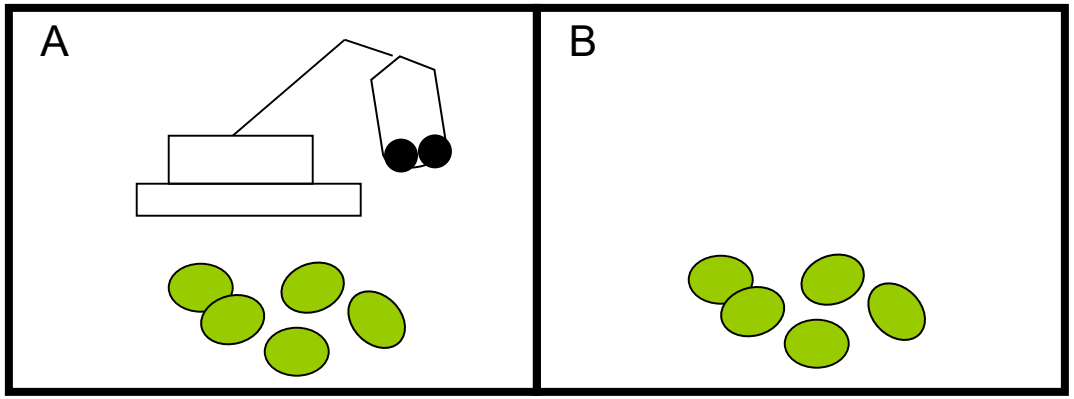


Percepts: $x_1(t) \in \{\text{A}, \text{B}\}, \quad x_2(t) \in \{\text{clean}, \text{dirty}\}$

Actions: $\alpha(t)$

Example: Vacuum Cleaner World

Image borrowed from V. Pavlovic, Rutgers

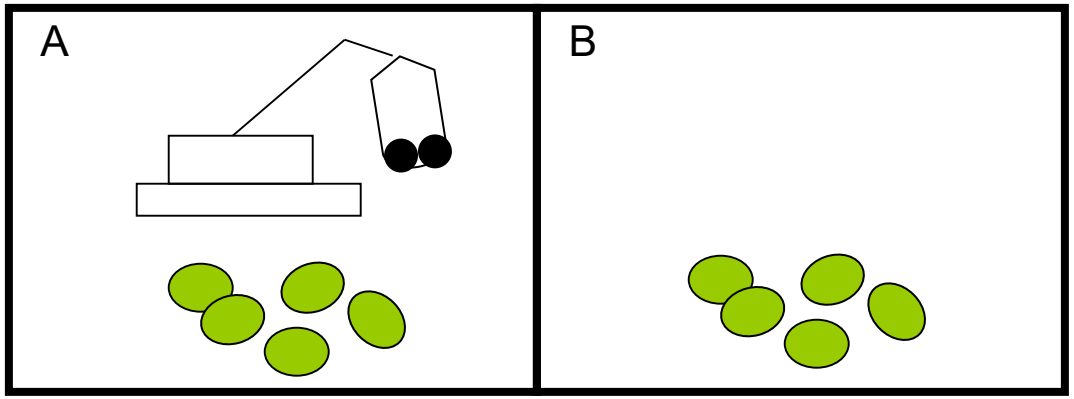


Percepts: $x_1(t) \in \{\text{A, B}\}, \quad x_2(t) \in \{\text{clean, dirty}\}$

Actions: $\alpha(t) \in \{\text{left, right, suck}\}$

Example: Vacuum Cleaner World

Image borrowed from V. Pavlovic, Rutgers



Percepts: $x_1(t) \in \{\text{A, B}\}, \quad x_2(t) \in \{\text{clean, dirty}\}$

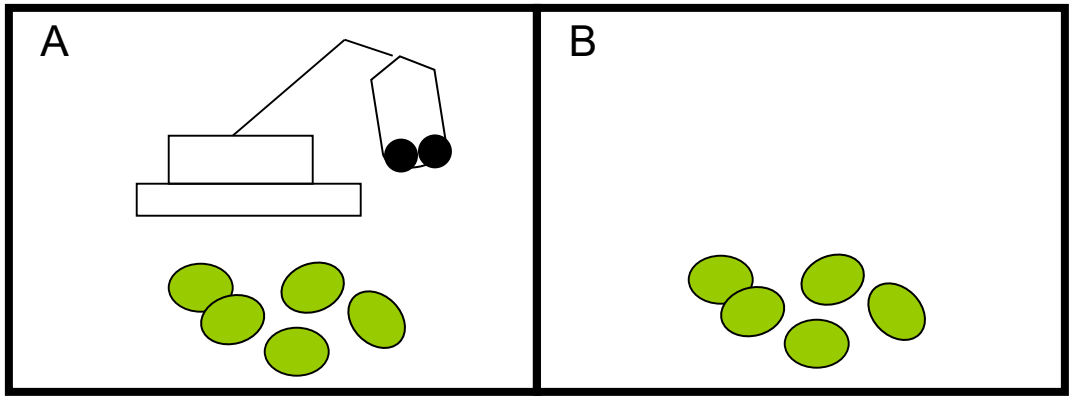
Actions: $\alpha(t) \in \{\text{left, right, suck}\}$

$\mathbf{x}(t) = \begin{pmatrix} * \\ dirty \end{pmatrix} \Rightarrow \boldsymbol{\alpha}(t) = suck$	$\mathbf{x}(t) = \begin{pmatrix} A \\ clean \end{pmatrix} \Rightarrow \boldsymbol{\alpha}(t) = right$	$\mathbf{x}(t) = \begin{pmatrix} B \\ clean \end{pmatrix} \Rightarrow \boldsymbol{\alpha}(t) = left$

This is an example of a reflex agent

Example: Vacuum Cleaner World

Image borrowed from V. Pavlovic, Rutgers

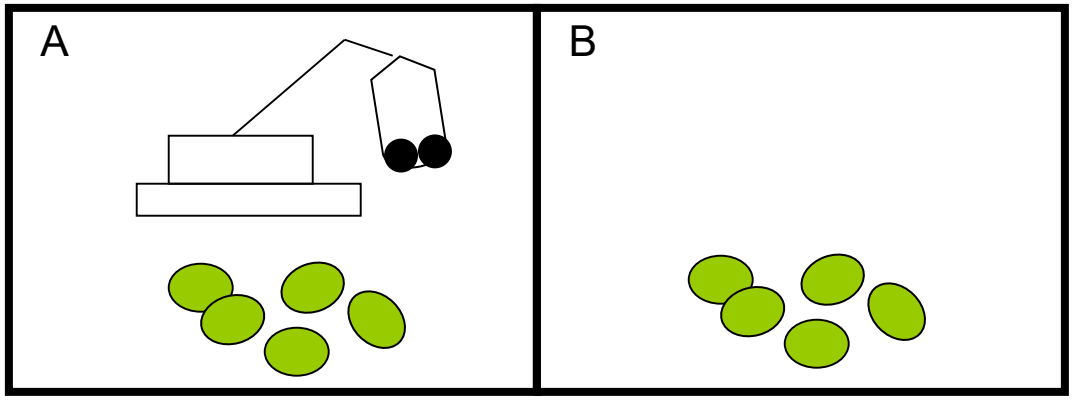


Percepts: $x_1(t) \in \{\text{A, B}\}, \quad x_2(t) \in \{\text{clean, dirty, filthy}\}$

Actions: $\alpha(t) \in \{\text{left, right, suck}\}$

Example: Vacuum Cleaner World

Image borrowed from V. Pavlovic, Rutgers



Percepts: $x_1(t) \in \{\text{A, B}\}, \quad x_2(t) \in \{\text{clean, dirty, filthy}\}$

Actions: $\alpha(t) \in \{\text{left, right, suck}\}$

$\mathbf{x}(t) = \begin{pmatrix} A \\ clean \end{pmatrix} \Rightarrow \boldsymbol{\alpha}(t) = \begin{pmatrix} - \\ right \\ - \end{pmatrix}$	$\mathbf{x}(t) = \begin{pmatrix} A \\ dirty \end{pmatrix} \Rightarrow \boldsymbol{\alpha}(t) = \begin{pmatrix} suck \\ right \\ - \end{pmatrix}$	$\mathbf{x}(t) = \begin{pmatrix} A \\ \text{filthy} \end{pmatrix} \Rightarrow \boldsymbol{\alpha}(t) = \begin{pmatrix} suck \\ - \\ - \end{pmatrix}$

A Rational Agent

A rational agent does "the right thing"

For each possible *percept sequence*, $\mathbf{x}(t)...\mathbf{x}(0)$, a rational agent should select the action that is *expected* to maximise its *performance measure* (given the evidence provided by the percept sequence and whatever built-in knowledge the agent has)

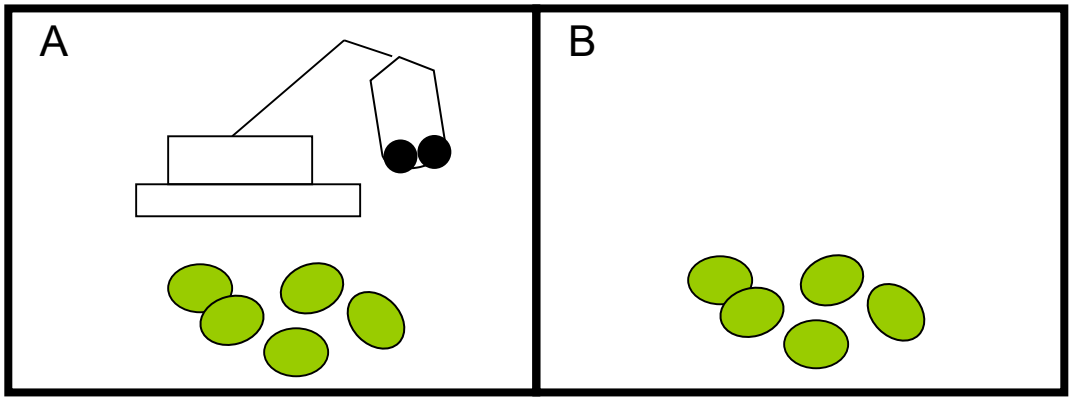
How to design a good performance measure?

Optimality

- Rationality ≠ Optimality
  - rational decision depends on the agent's percepts in the past (up to now)
  - and the *expected* experiences in the future
- Not on future observations
  - nor experiences of others (unknown to the agent)
- Optimality = Rationality + Omniscience
  - omniscience is perfect knowledge
  - rationality is optimising *expected* performance

Vacuum Cleaner Performance Measure

Image borrowed from V. Pavlovic, Rutgers



State defined performance measure

$$S = \begin{cases} +1 & \text{For each clean square at time } t \\ -1 & \text{For each dirty square at time } t \\ -1000 & \text{If more than } N \text{ dirty squares} \end{cases}$$

Action defined performance measure

$$S = \begin{cases} +100 & \text{For each piece of dirt vacuumed up} \\ -1 & \text{For each move left or right} \end{cases}$$

Task Description

A problem to which the agent is a solution

P	Performance measure	*Maximize number of clean cells & minimize number of dirty cells.*
E	Environment	*Discrete cells. Each is either dirty or clean. Partially observable environment, static, deterministic, and sequential. Single agent.*
A	Actuators	*Mouthpiece for sucking dirt.* *Engine & wheels for moving.*
S	Sensors	*Dirt sensor & position sensor.*

Environment Types

- Single vs multi-agent
- Accessible vs inaccessible
  - partially observable
- Deterministic vs nondeterministic
  - apparently nondeterministic
  - strategic environment
- Episodic vs sequential
- Static vs dynamic
- Discrete vs continuous

Basic Classes of Agents

- Random agent
- Fixed (sequential) agent
- Reflex agent
  - no internal memory
  - can be simple (table lookup) or very complex
- Model-based agent
  - knowledge about how the world works
- Goal-based agent
- Utility-based agent
- Learning agent

The random agent:

$$\boldsymbol{\alpha}(t) = \mathrm{rnd}$$

The action $\boldsymbol{\alpha}(t)$ is selected purely at random, without any consideration of the percept $\mathbf{x}(t)$

The sequential agent:

$$\boldsymbol{\alpha}(t) = f[t]$$

$\boldsymbol{\alpha}(t)$ are selected in a fixed sequence, also without any consideration $\mathbf{x}(t)$

Neither is very intelligent.

The reflex agent

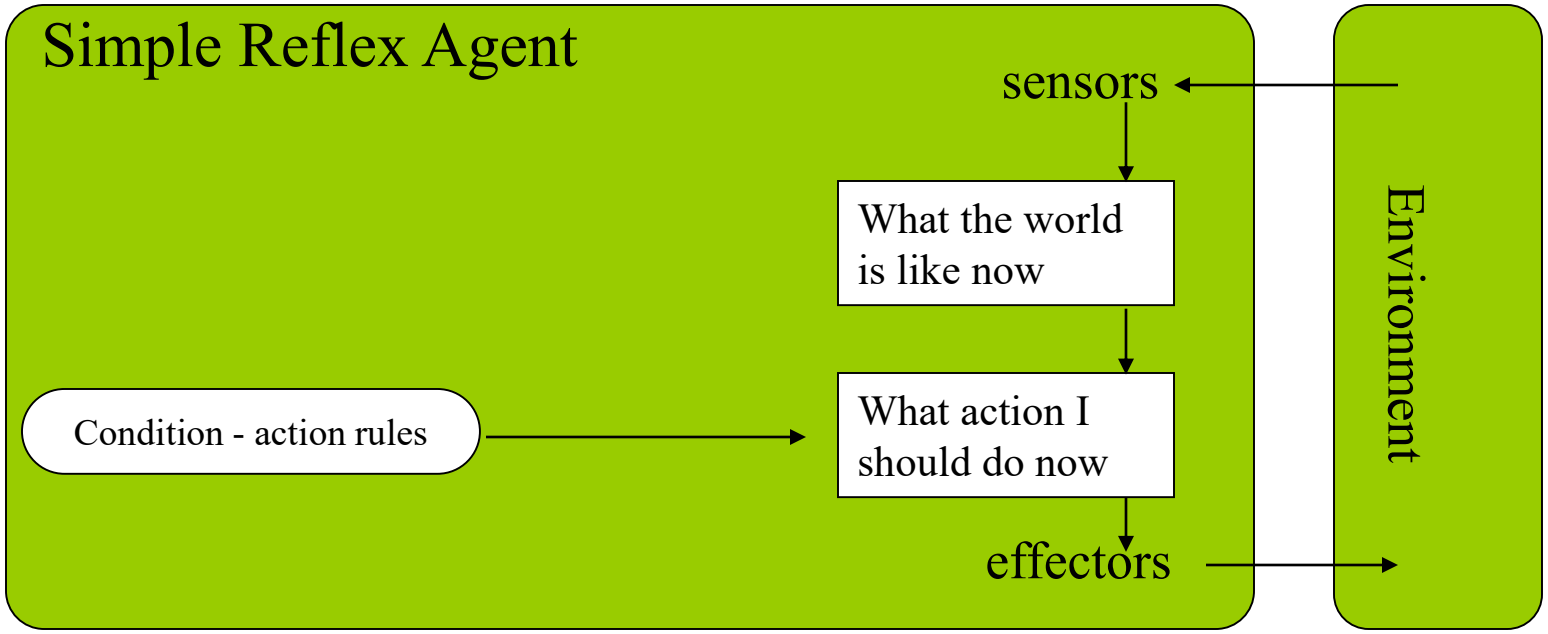
$$\boldsymbol{\alpha}(t) = f[\mathbf{x}(t)]$$

The action $\boldsymbol{\alpha}(t)$ is selected based on only the most recent percept $\mathbf{x}(t)$

No consideration of percept or action history.

In many domains will end up in infinite loops of various kinds.

Appears intelligent when it woks, but often ends up trying to do the same thing over and over again.



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*

First match.
No further matches sought.
Only one level of deduction.

A simple reflex agent works by finding a rule whose condition matches the current situation (as defined by the percept) and then doing the action associated with that rule.

Slide borrowed from Sandholm @ CMU

Simple Reflex Agent

- Table lookup of condition-action pairs defining all possible condition-action rules necessary to interact with the environment
  - e.g. IF *car-in-front-is-breaking* THEN *initiate-breaking*

- **Problems**
  - Table is often too big to generate/store (e.g. a self-driving car)
  - Takes long time to build the table
  - No knowledge of non-perceptual parts of the current state
  - Not adaptive to changes in the environment & requires entire table to be updated if changes occur
  - No looping capability: can't make actions conditional

- Complex reflex agents avoid the need to explicitly generate and store full lookup table

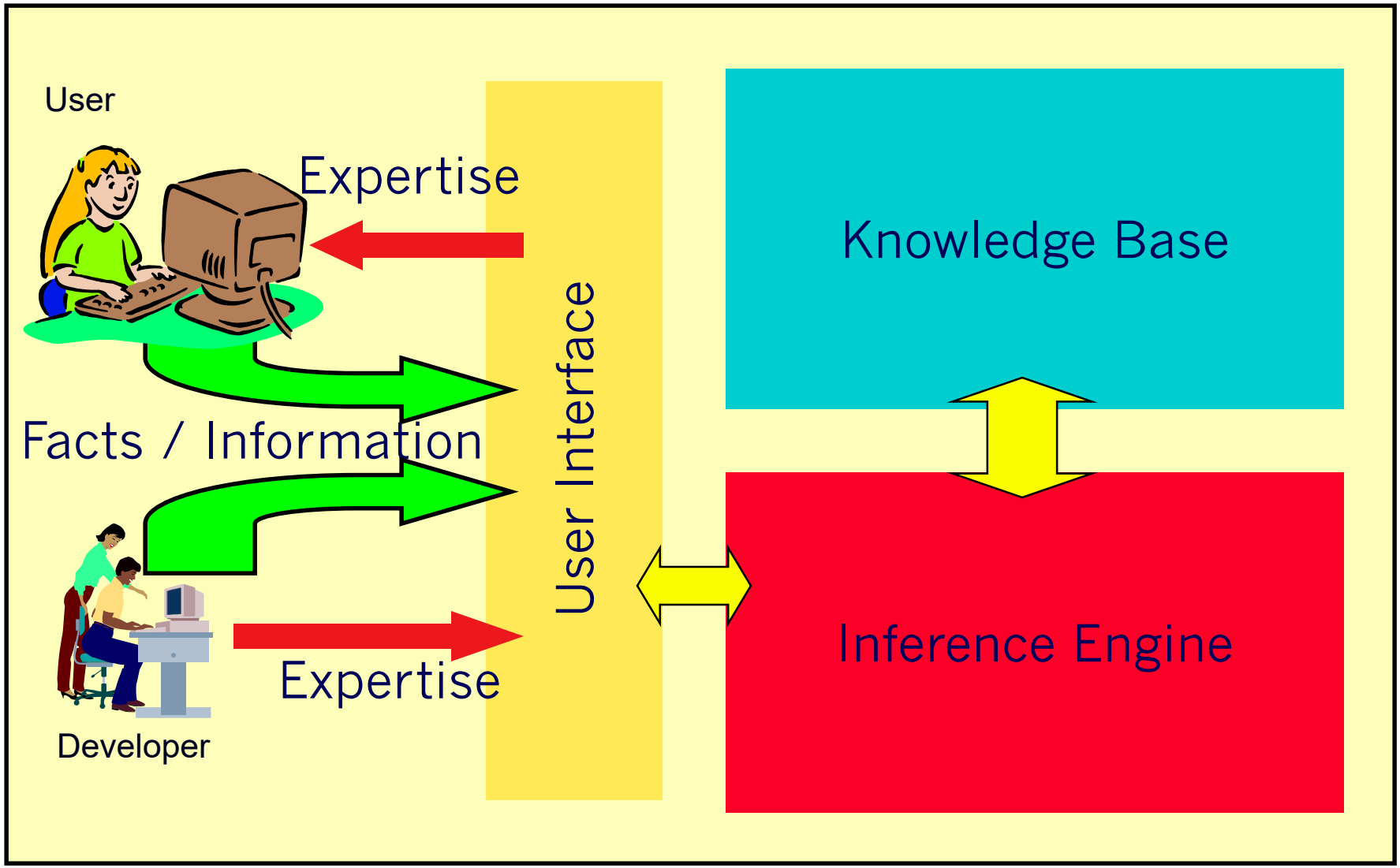
Slide based on Sandholm @ CMU

Complex Reflex Agent

- Conceptually equivalent to lookup table
  - how to make it easier to create and understand?
  - classical programming paradigm:

    IF … THEN …
  - it is very natural for people to think this way
  - but it is hard to express this knowledge precisely
- Knowledge-based systems (rule-based expert systems)
  - separate the domain-dependent knowledge
  - from domain-independent inference engine

Main Components of a KBS



Knowledge Base

- Consists of static and dynamic parts
  - static knowledge are rules and facts compiled as a part of the creation of the system
  - dynamic knowledge consists of facts related to a particular consultation (query) of the system
- Initially, the dynamic knowledge base, often called working memory, is empty
  - as consultation progresses, dynamic knowledge base grows and is used in decision making

Inference Engine

- Performs reasoning in a cycle
  - until no further reasoning can be done

1. Rule Matching
   - figure out which rules are applicable
2. Conflict resolution
   - select the active rule with highest priority
3. Execution
   - perform the actions described by the consequent of the selected rule

Forward and Backward Chaining

- Forward chaining is data-driven
  - reasoning from facts to the conclusion
  - as soon as facts are available, they are used to match antecedents of rules
  - often used for monitoring and control
- Backward chaining is query-driven
  - starts from a hypothesis (query)
  - searches for supporting rules and facts
  - until it can either confirm or reject the hypothesis
  - often used in diagnostic and consultation

Basic Classes of Agents

- Fixed (sequential) agent
- Random agent
- Reflex agent
  - no internal memory
  - can be simple (table lookup) or complex
- Model-based agent
  - knowledge about how the wold works
- Goal-based agent
- Utility-based agent
- Learning agent

The model based agent

$$\boldsymbol{\alpha}(t) = f[\mathbf{x}(t), \theta(t)]$$

$$\theta(t) = \theta[\mathbf{x}(t-1), \ldots, \mathbf{x}(0), \boldsymbol{\alpha}(t-1), \ldots, \boldsymbol{\alpha}(0)]$$

The action $\boldsymbol{\alpha}(t)$ is selected based on the percept $\mathbf{x}(t)$ and the current state $\theta(t)$.

The world model or state $\theta(t)$ keeps track of past actions and the percept history.

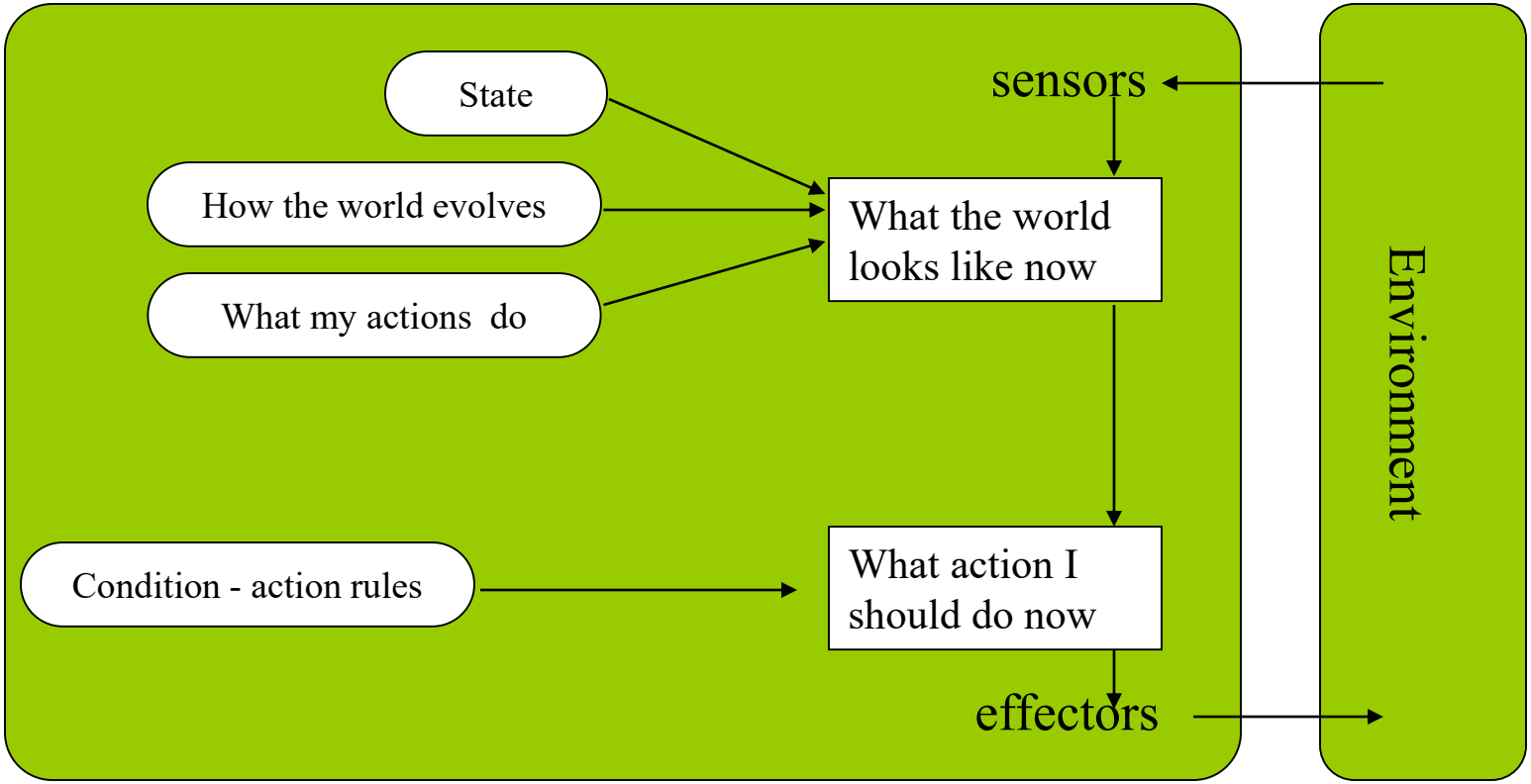
The goal based agent

$$\boldsymbol{\alpha}(t) = f[\mathbf{x}(t), \theta(t+T), \ldots, \theta(t), \ldots, \theta(0)]$$

The action $\boldsymbol{\alpha}(t)$ is selected based on the percept $\mathbf{x}(t)$, the current state $\theta(t)$, and the future *expected* set of states $\theta(t+1)$, $\theta(t+2)$, ...

One or more of the states are considered to be the goal state(s).

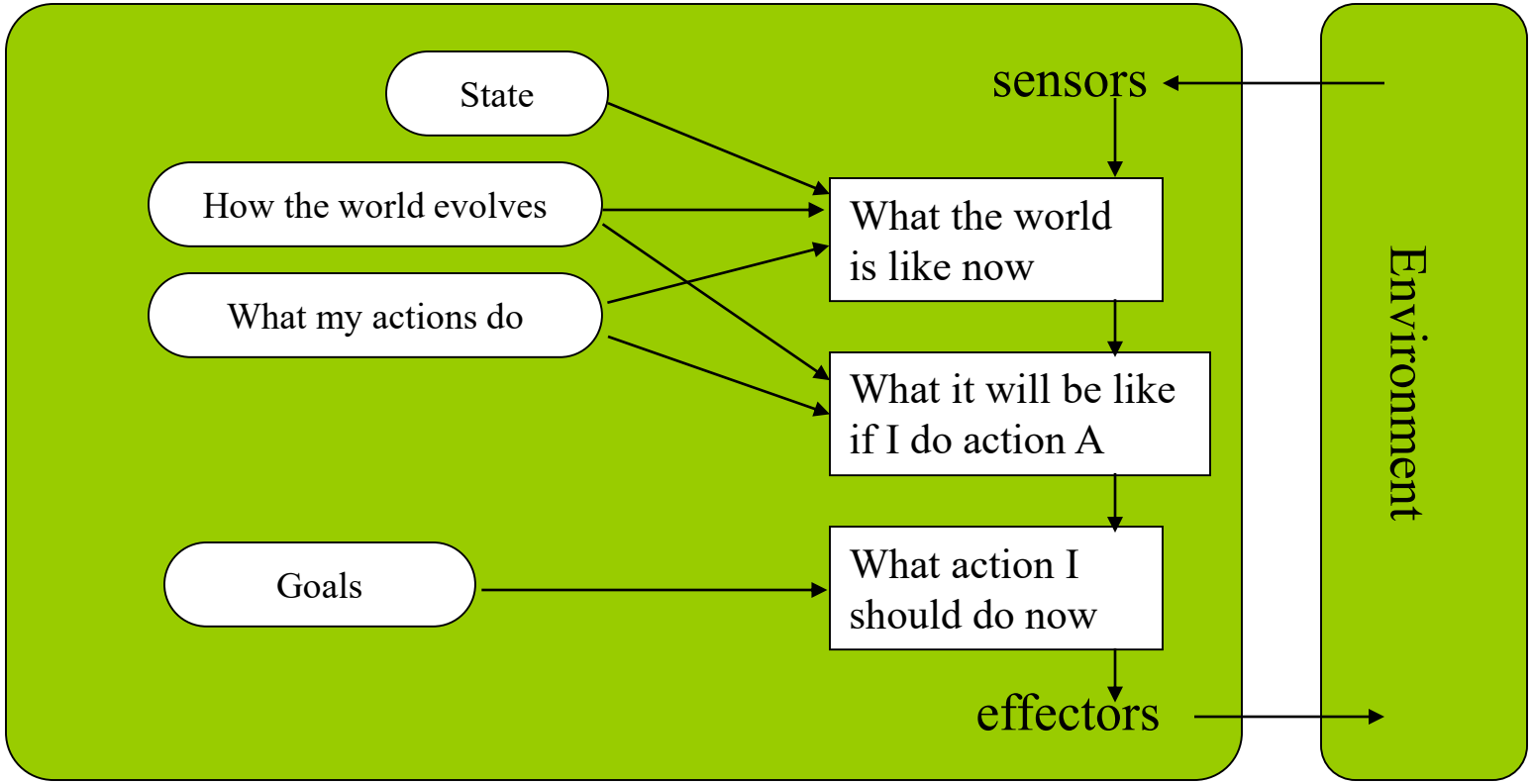
Agent With Internal State



Model based agent

Slide borrowed from Sandholm @ CMU

Agent With Explicit Goals



Goal based agent

The utility based agent

$$\boldsymbol{\alpha}(t) = f[\mathbf{x}(t), U(\theta(t+T)), \ldots, U(\theta(t)), \ldots, U(\theta(0))]$$

The action $\boldsymbol{\alpha}(t)$ is selected based on the percept $\mathbf{x}(t)$, and the utility of future, current and past states $\theta(t)$

The utility function $U(\theta(t))$ expresses the benefit the agent has from being in state $\theta(t)$

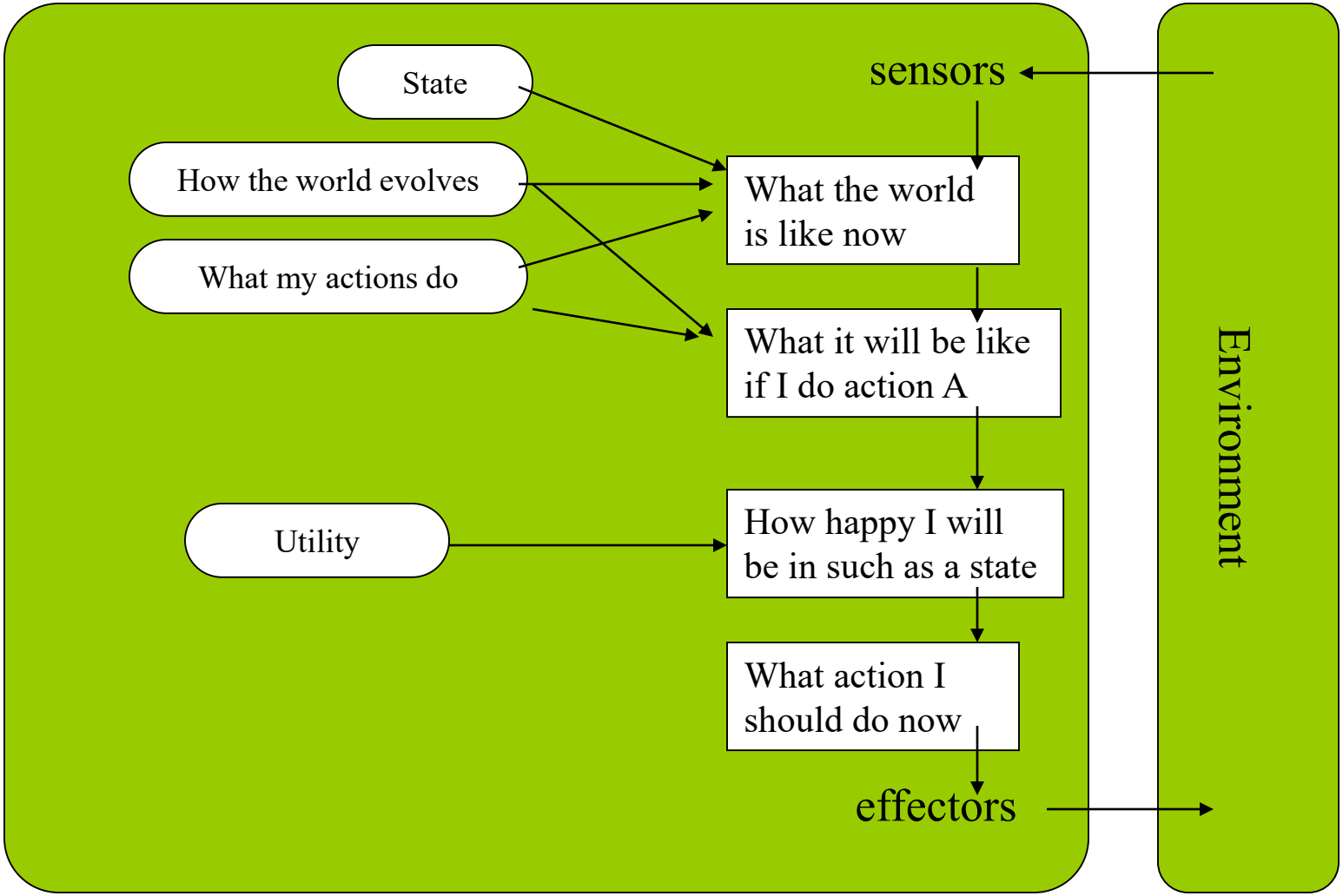
The learning agent

$$\boldsymbol{\alpha}(t) = \hat{f}[\mathbf{x}(t), \hat{U}(\hat{\theta}(t+T)), \ldots, \hat{U}(\theta(t)), \ldots, \hat{U}(\theta(0))]$$

The learning agent is similar to the utility based agent.

The difference is that the knowledge parts (i.e. the prediction of future states, the utility, ...etc.) can now adapt and improve with experience.

Utility-Based Agent



Slide borrowed from Sandholm @ CMU

Basic Classes of Agents

- Fixed (sequential) agent
- Random agent
- Reflex agent
  - no internal memory
  - can be simple (table lookup) or complex
- Model-based agent
  - knowledge about how the wold works
- Goal-based agent
- Utility-based agent
- Learning agent

Questions?

Discussion

Exercise 2.2:

Both the performance measure and the utility function measure how well an agent is doing. Explain the difference between the two.

Discussion

Exercise 2.2:

Both the performance measure and the utility function measure how well an agent is doing. Explain the difference between the two.

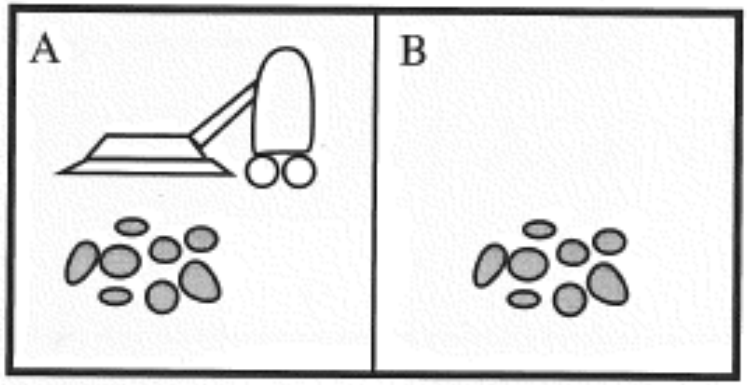
They can be the same but do not have to be. The performance function is used externally to measure the agent's performance. The utility function is used internally (by the agent) to measure (or estimate) it's performance. There is always a performance function but not always an utility function (cf. random agent).

Exercise

Exercise 2.4:

Let's examine the rationality of various vacuum-cleaner agent functions:

a. Show that the simple vacuum-cleaner agent function described in figure 2.3 is indeed rational under the assumptions listed on page 36.
b. Describe a rational agent function for the modified performance measure that deducts one point for each movement. Does the corresponding agent program require internal state?
c. Discuss possible agent designs for the cases in which clean squares can become dirty and the geography of the environment is unknown. Does it make sense for the agent to learn from its experience in these cases? If so, what should it learn?



What should be the performance measure?

Figure 2.2 A vacuum-cleaner world with just two locations.

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.

Possible states of the world:

[A, Clean] & [B, Clean] ⇒ world is clean after 0 steps
[A, Clean] & [B, Dirty] ⇒ world is clean after 2 steps if agent is in A and...
[A, Dirty] & [B, Dirty] ⇒ world is clean after 3 steps
[A, Dirty] & [B, Clean] ⇒ world is clean after 1 step if agent is in A and...

Can any agent do it faster (in fewer steps)?

Figure 2.2 A vacuum-cleaner world with just two locations.

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*
⋮	⋮

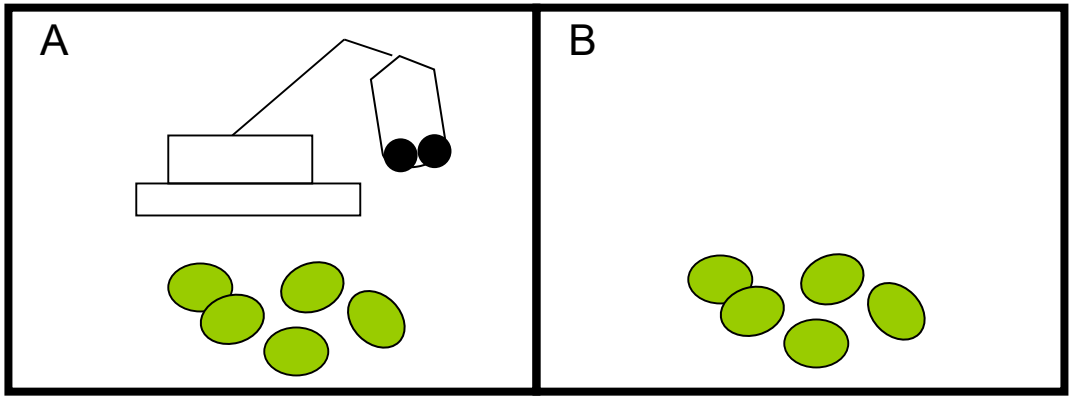
Figure 2.3 Partial tabulation of a simple agent function for the vacuum-cleaner world shown in Figure 2.2.

Exercise 2.4

a. If (square A dirty & square B clean) then the world is clean after one step. No agent can do this quicker.
If (square A clean & square B dirty) then the world is clean after two steps. No agent can do this quicker.
If (square A dirty & square B dirty) then the world is clean after three steps. No agent can do this quicker.

The agent is rational (elapsed time is our performance measure).

Image borrowed from V. Pavlovic, Rutgers



Exercise

Exercise 2.4:

Let's examine the rationality of various vacuum-cleaner agent functions:

a. Show that the simple vacuum-cleaner agent function described in figure 2.3 is indeed rational under the assumptions listed on page 36.

b. Describe a rational agent function for the modified performance measure that deducts one point for each movement. Does the corresponding agent program require internal state?

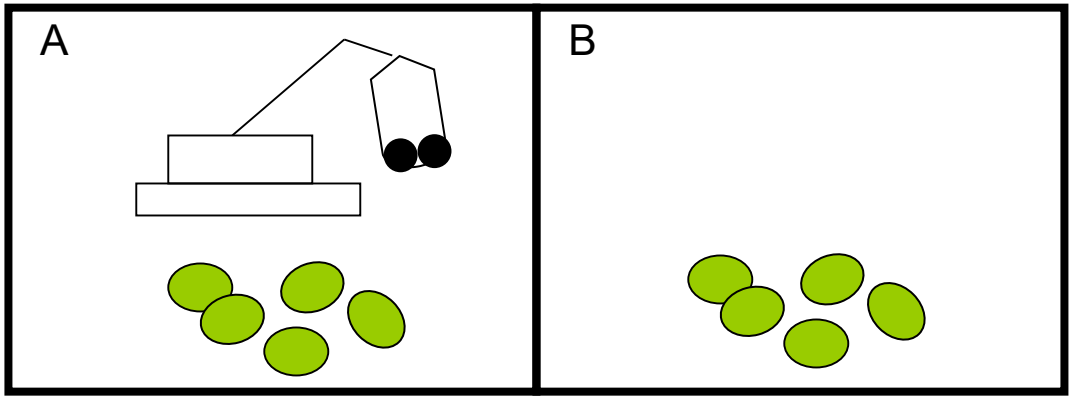
c. Discuss possible agent designs for the cases in which clean squares can become dirty and the geography of the environment is unknown. Does it make sense for the agent to learn from its experience in these cases? If so, what should it learn?

Exercise 2.4

b. The reflex agent will continue moving even after the world is clean. An agent that has memory would do better than the reflex agent if there is a penalty for each move. Memory prevents the agent from visiting squares where it has already cleaned.

(The environment has no production of dirt; a dirty square that has been cleaned remains clean.)

Image borrowed from V. Pavlovic, Rutgers



Exercise

Exercise 2.4:

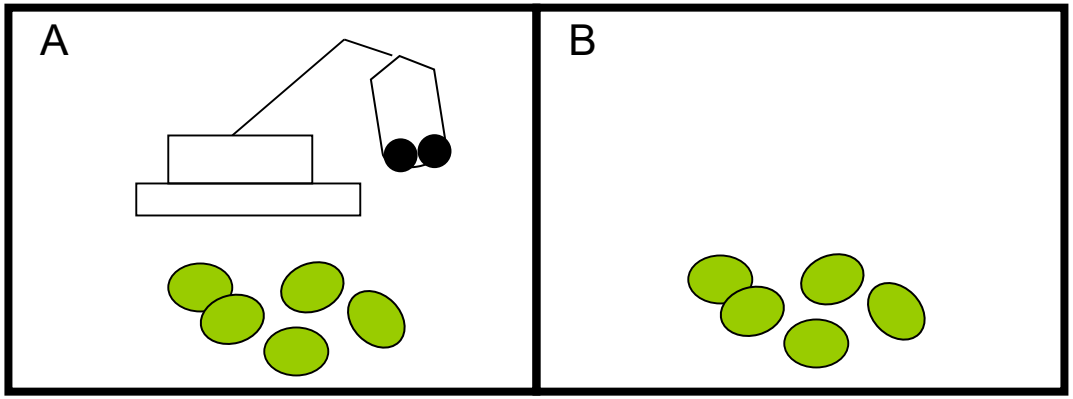
Let's examine the rationality of various vacuum-cleaner agent functions:

a. Show that the simple vacuum-cleaner agent function described in figure 2.3 is indeed rational under the assumptions listed on page 36.
b. Describe a rational agent function for the modified performance measure that deducts one point for each movement. Does the corresponding agent program require internal state?
c. Discuss possible agent designs for the cases in which clean squares can become dirty and the geography of the environment is unknown. Does it make sense for the agent to learn from its experience in these cases? If so, what should it learn?

Exercise 2.4

c. If the agent has a very long lifetime (infinite) then it is better to learn a map. The map can tell where the probability is high for dirt to accumulate. The map can carry information about how much time has passed since the vacuum cleaner agent visited a certain square, and thus also the probability that the square has become dirty.
If the agent has a short lifetime, then it may just as well wander around randomly (there is no time to build a map).

Image borrowed from V. Pavlovic, Rutgers



Lab 1

- Implementation of different kinds of agents
  - random agent
  - sequential agent
  - reflex agent
  - agent with memory
- Similar technology, two example domains
  - poker player & mobile robot
- Python https://www.python.org/
  - V-REP http://www.coppeliarobotics.com/