CS 4700:
Foundations of Artificial Intelligence

Bart Selman

Structure of intelligent agents and environments

R&N: Chapter 2

Outline

Characterization of agents and environments

    Rationality

    PEAS (Performance measure, Environment, Actuators, Sensors)

Environment types

Agent types

Intelligent agent-view provides framework to integrate the many subareas of AI.

Warning: Chapter 2 is somewhat high-level and abstract. Much of the technical framework of how intelligent agents are actually built is introduced later.

Agents

Definition: An agent perceives its environment via sensors and acts upon that environment through its actuators
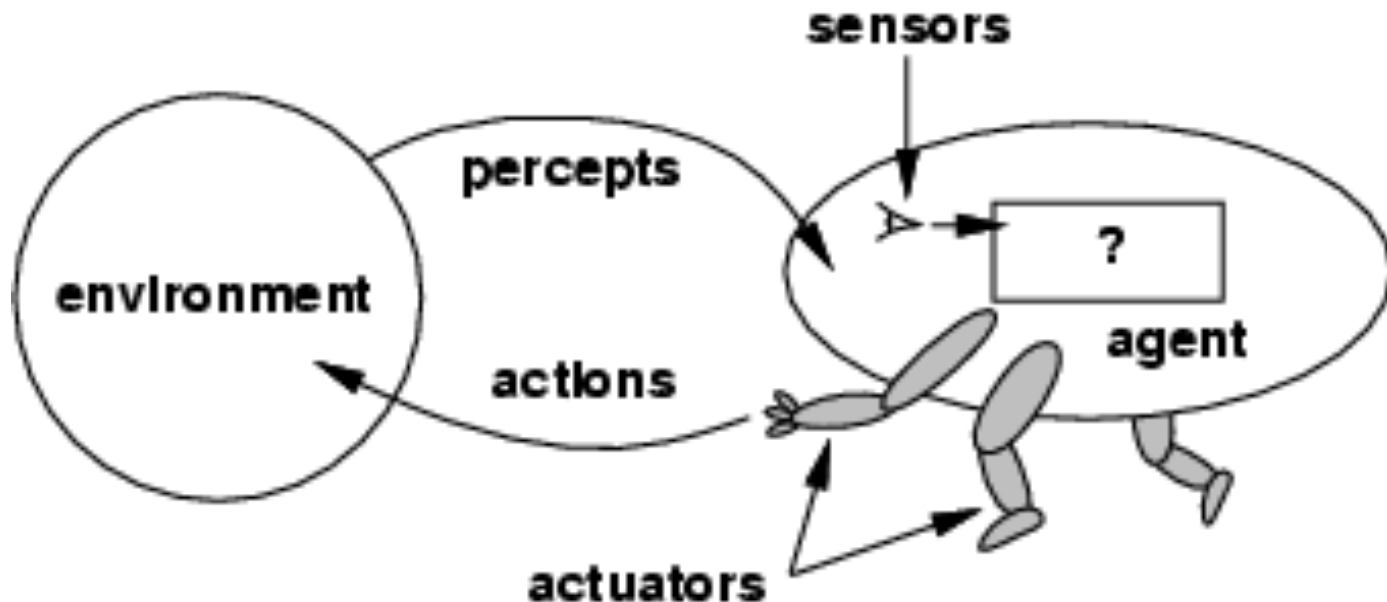
The agent view is really quite generic.

p. 36 R&N:

In a sense, all areas of engineering can be seen as designing artifacts that interact with the world. AI operates at the end of the spectrum, where the artifacts use significant computational resources and the task and environment requires non-trivial decision making.

But, the definition of "agents" does technically also include, e.g., calculators or cars, artifacts with very limited to no intelligence. (Self-driving cars come closer to what we view as intelligent agents, because of non-trivial sensing and real-time decision making under uncertainty.)

Next: definition of rational agents. Sufficiently complex rational agents can be viewed as "intelligent agents."

Agent / Robot

E.g., vacuum-cleaner world
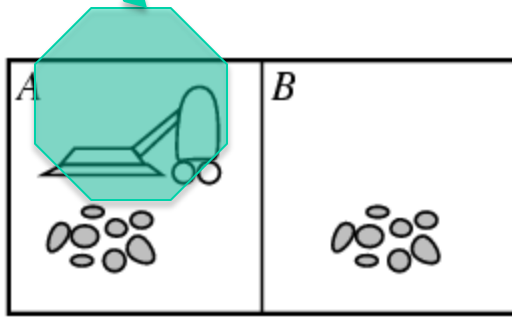
Percepts: location and contents,
e.g., [A, Dirty]

Actions: *Left*, *Right*, *Suck*, *NoOp*

iRobot Roomba® 400
Vacuum Cleaning Robot

iRobot Corporation

Founder Rodney Brooks (MIT)

- Powerful suction and rotating brushes
- Automatically navigates for best cleaning coverage
- Cleans under and around furniture, into corners and along wall edges
- Self-adjusts from carpets to hard floors and back again
- Automatically avoids stairs, drop-offs and off-limit areas
- Simple to use— just press the Clean button and Roomba does the rest

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

Performance measure: An *objective criterion ("utility function") 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 car: 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.

Definition of Rational Agent:

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.

Why "in expectation"?

Captures actions with stochastic / uncertain effects or actions performed in stochastic environments.
We can then look at the expected value of an action.

In high-risk settings, we may also want to limit the worst-case behavior.

Rational agents

Notes:

Rationality is distinct from **omniscience ("all knowing")**. We can behave rationally even when faced with incomplete information.

Agents can perform actions in order to modify future percepts so as to obtain useful information: **information gathering, exploration.**

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

Characterizing a Task Environment

Must first specify the setting for intelligent agent design.

PEAS: Performance measure, Environment, Actuators, Sensors

Example: the task of designing a self-driving car

- **Performance measure** Safe, fast, legal, comfortable trip
- **Environment** Roads, other traffic, pedestrians
- **Actuators** Steering wheel, accelerator, brake, signal, horn
- **Sensors** Cameras (but use limited), LIDAR (light/radar), speedometer, GPS, odometer, engine sensors, keyboard

1) Fully observable / Partially observable

– If an agent's sensors give it access to the **complete state of the environment** needed to choose an action, the environment is **fully observable.**

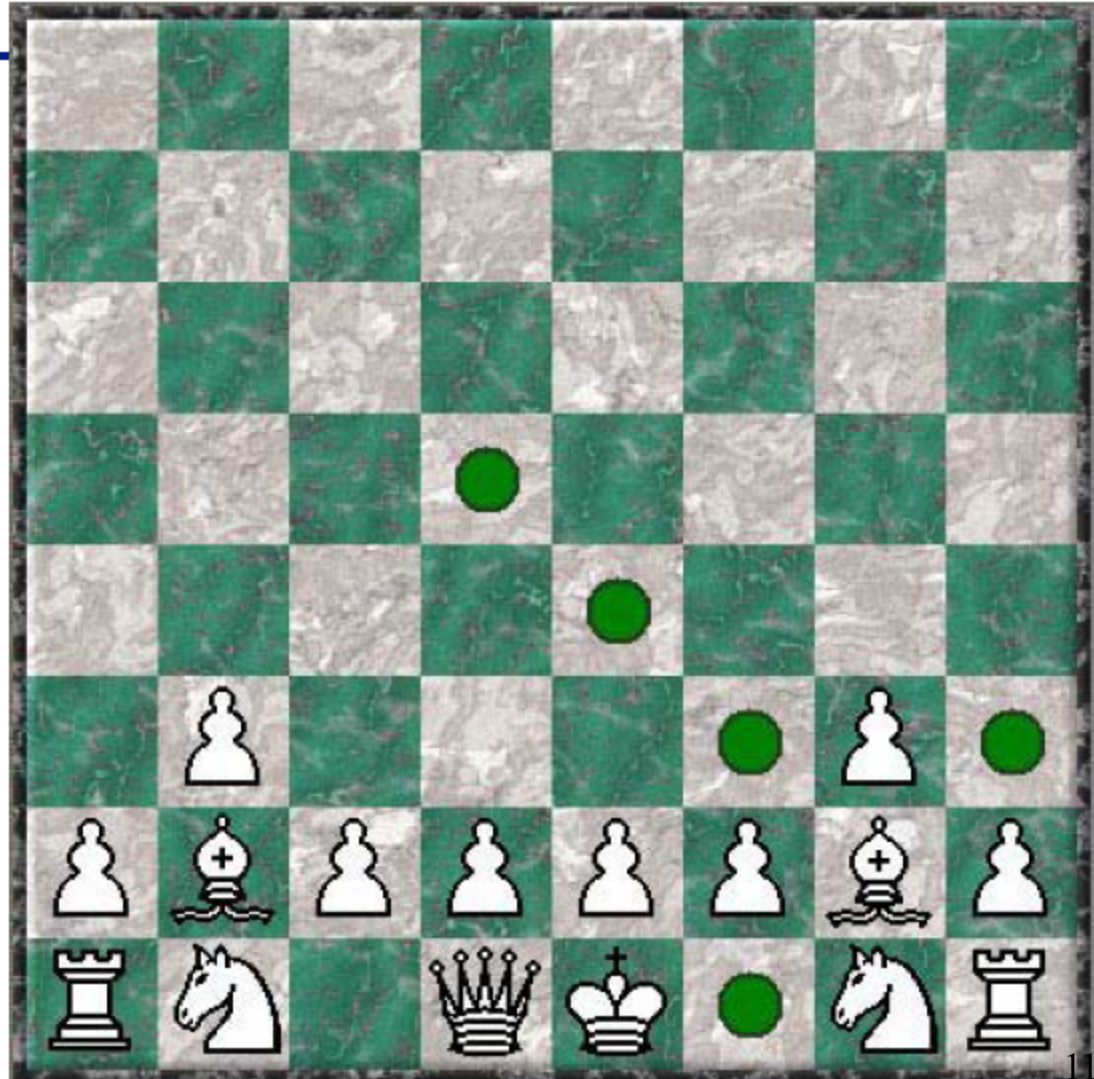(e.g. chess – what about Kriegspiel?)

Incomplete / uncertain information inherent in the game.

Balance exploitation (best move given current knowledge) and exploration (moves to explore where opponent's pieces might be).

Use probabilistic reasoning techniques.

Making things a bit more challenging…
Kriegspiel --- you can't see your opponent!

2) Deterministic / Stochastic

- An environment is **deterministic** if the next state of the environment is completely determined by the <u>current state</u> of the environment and the <u>action</u> 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**).

In a fully observable, deterministic environment, the agent need not deal with uncertainty.

Note: Uncertainty can also arise because of computational limitations. E.g., we may be playing an **omniscient** ("all knowing") opponent but we may not be able to compute his/her moves.

In fact, when considering reinforcement learning of games, even when game is deterministic (e.g. chess), a useful approach is to consider the opponent's move as part of the (only partially known) environment. Can introduce non-determinism. So, your next state is after your move opponent's move.

3) Episodic / Sequential

- In an **episodic** environment, 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 decision can affect future decisions. **(E.g., chess and driving)**

4) Static / Dynamic

- A **static** environment does not change while the agent is thinking.

- The passage of time as an agent deliberates is irrelevant.

- The environment is **semidynamic** if the environment itself does not change with the passage of time but the agent's performance score does.

5) Discrete / Continuous

- If the number of distinct percepts and actions is limited, the environment is discrete, otherwise it is continuous.

6) Single agent / Multi-agent

- 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.

Example Tasks and Environment Types

|	Chess with a clock	Chess without a clock	Taxi driving
Fully observable	Yes	Yes	No
Deterministic	Strategic	Strategic	No
Episodic	No	No	No
Static	Semi	Yes	No
Discrete	Yes	Yes	No
Single agent	No	No	No

The environment type largely determines the agent design

The real world is (of course) partially observable, stochastic, sequential, dynamic, continuous, multi-agent

How to make the right decisions?　　　**Decision theory**
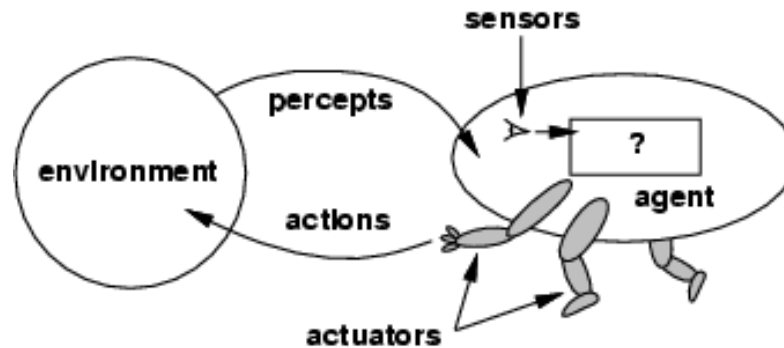
The agent function maps from percept histories to actions

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

The agent program runs (internally) on the physical architecture to produce f

agent = architecture + program

our focus

Design an agent program assuming an architecture that will make the percepts from the sensors available to the program.
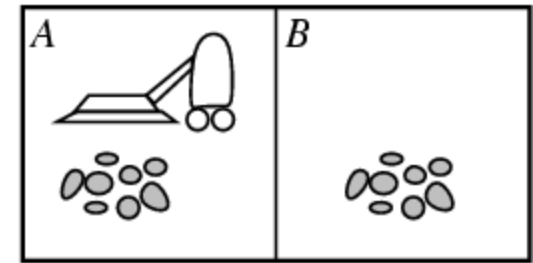
Types of Agents

I) --- Table-lookup driven agents

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/learn the table

Toy example:
Vacuum world.

Percepts: robot senses it's location and "cleanliness."

    So, location and contents, e.g., [A, Dirty], [B, Clean].

    With 2 locations, we get **4 different possible sensor inputs.**

Actions: *Left, Right, Suck, NoOp*

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
⋮ [Perceptual input sequence]	⋮
[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.

Table lookup

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.

In more real-world scenarios, one would have many more different
percepts (eg 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.

So, 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.
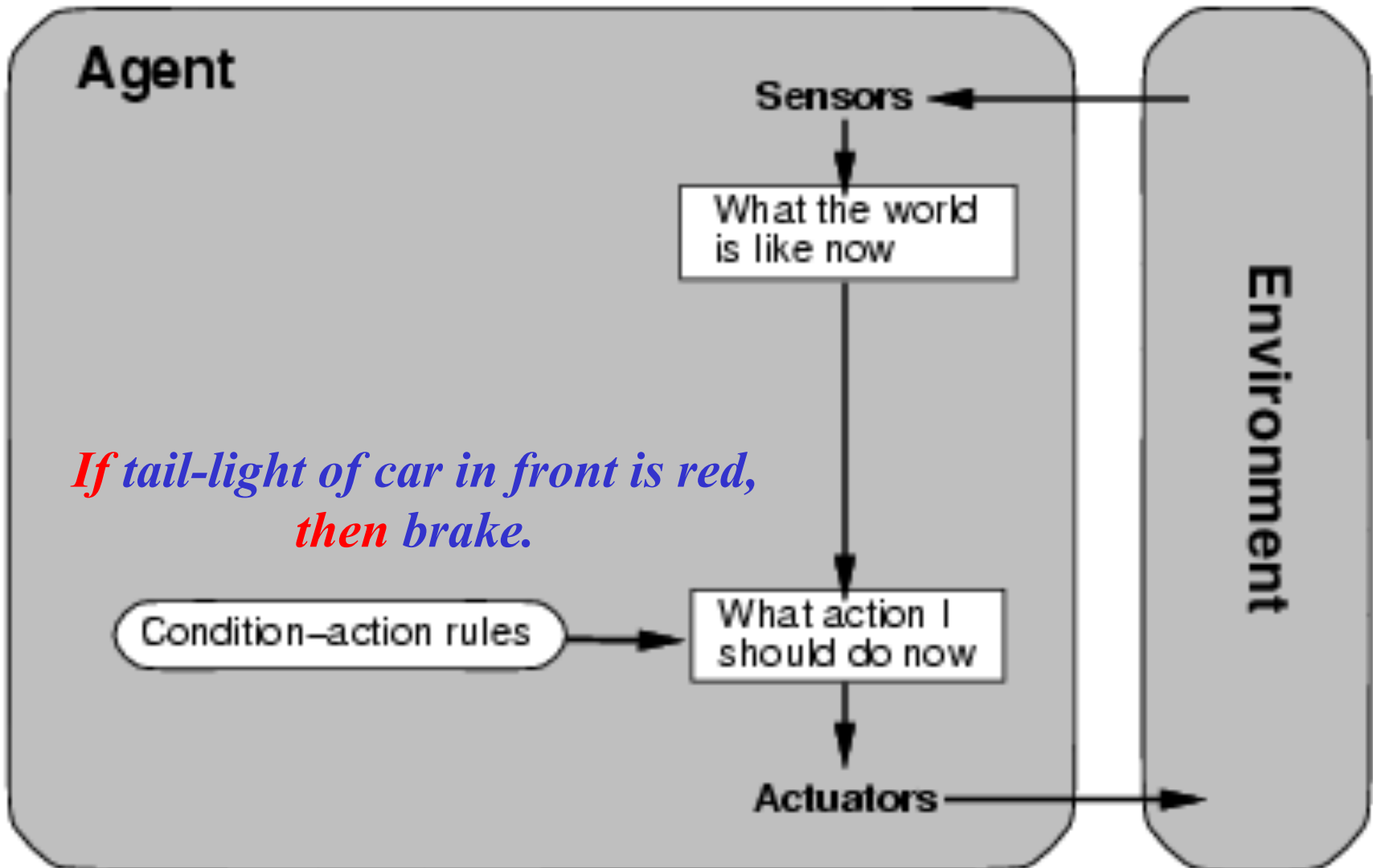
II) --- Simple reflex agents

Agents **do not have memory** of past world states or percepts.
   So, actions depend solely on **current percept**.
   Action becomes a "reflex."

Uses **condition-action rules**.

Agent selects actions on the basis of *current* percept only.


If tail-light of car in front is red, then brake.

Closely related to "behaviorism" (psychology; quite effective in explaining lower-level animal behaviors, such as the behavior of ants and mice).

The Roomba 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. ants colonies and bee hives are quite complex.

A. Simple rules in a diverse environment can give rise to surprising complexity.

See A-life work (artificial life) community, and Wolfram's cellular automata.

Click to LOOK INSIDE!

STEPHEN WOLFRAM

A NEW KIND OF SCIENCE
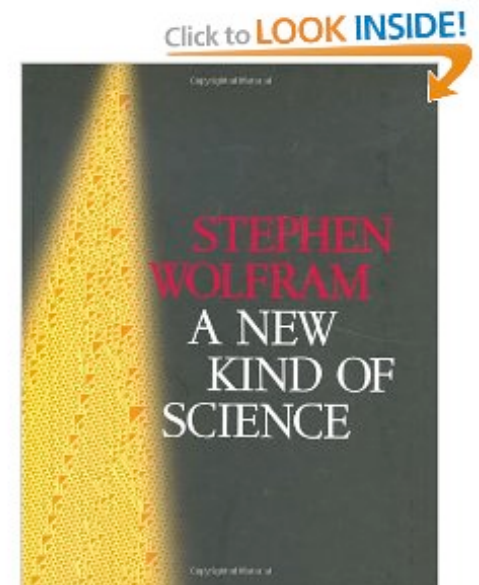
III) --- Model-based reflex agents

Key difference (wrt 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: Rodney Brooks' Subsumption Architecture
--- behavior based robots.

Model-based reflex agents

How detailed?

"Infers potentially dangerous driver in front."

If "dangerous driver in front," then "keep distance."

State

How the world evolves

What my actions do

Sensors

What the world is like now

Condition–action rules

What action I should do now

Agent

Actuators

Environment
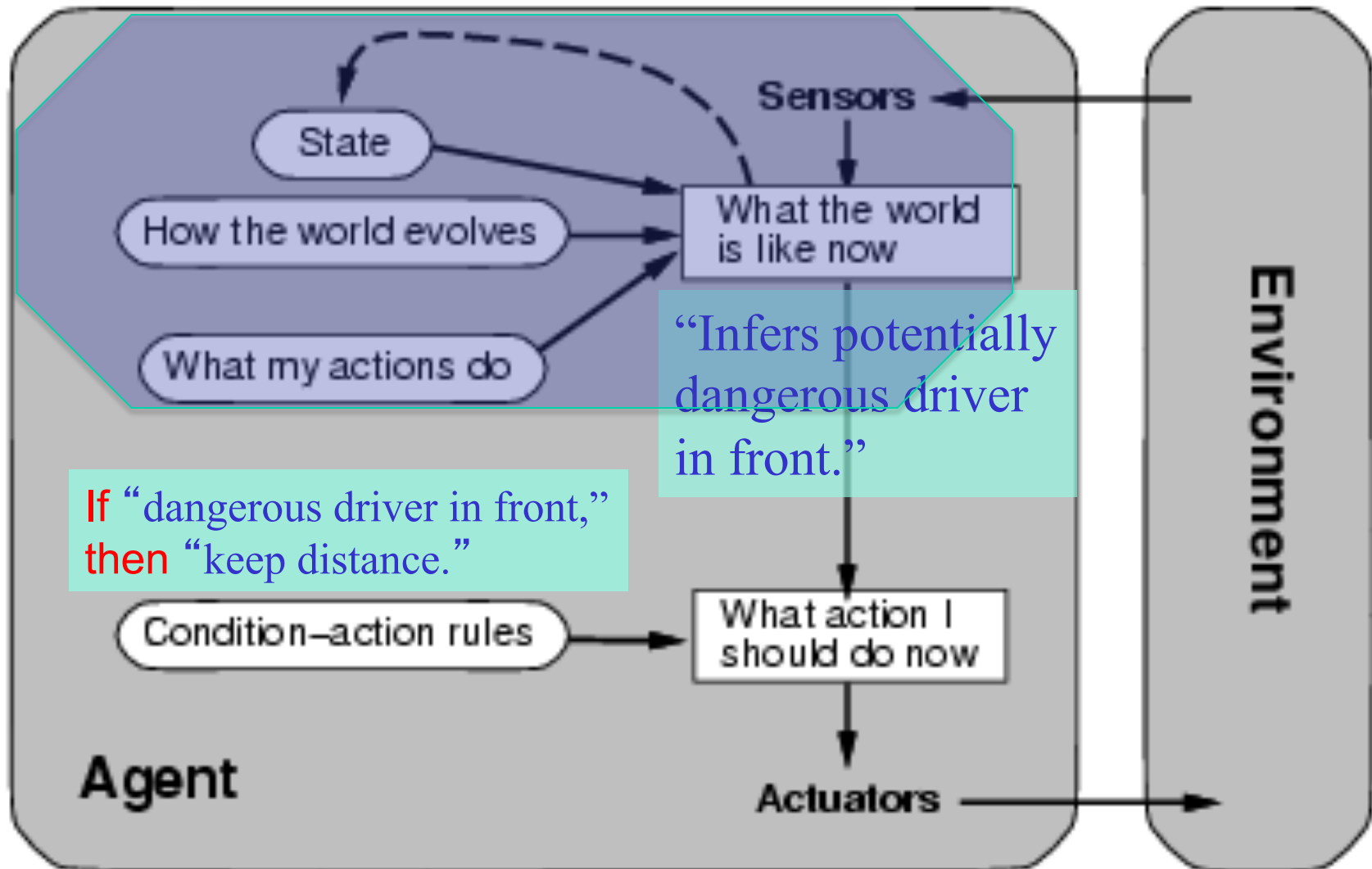
An example:
Brooks' Subsumption Architecture

Main idea: build complex, intelligent robots by decomposing behaviors into a hierarchy of skills, each defining a percept-action cycle for one very specific task.

Examples: collision avoidance, wandering, exploring, recognizing doorways, etc.

Each behavior is modeled by a finite-state machine with a few states (though each state may correspond to a complex function or module; provides internal state to the agent).

Behaviors are loosely coupled via asynchronous interactions.

Note: minimal internal state representation.

p. 1003 R&N

Subsumption Architecture, cont.

In subsumption architecture, increasingly complex behaviors arise from the combination of simple behaviors.

The most basic simple behaviors are on the level of reflexes: • avoid an object; • go toward food if hungry • move randomly.

A more complex behavior that sits on top of simple behaviors may be "go across the room."

The more complex behaviors subsume the less complex ones to accomplish their goal.

How much of an internal model of the world?

Planning in and reasoning about our surroundings appears to require some kind of internal representation of our world. We can "try" things out in this representation. Much like an running a "simulation" of the effect of actions or a sequence of actions in our head.

General assumption for many years:

The more detailed internal model, the better.

Brooks (mid 80s and 90s) challenged this view:

The philosophy behind Subsumption Architecture is that *the world should be used as its own model*. According to Brooks, storing models of the world is dangerous in dynamic, unpredictable environments because representations might be incorrect or outdated. What is needed is the ability to react quickly to the present. So, use minimal internal state representation, complement at each time step with sensor input.

Debate continues to this day: How much of our world do we (should we) represent explicitly? Subsumption architecture worked well in robotics.

IV) --- Goal-based agents

Key difference wrt Model-Based Agents:

In addition to state information, have **goal information** that
describes desirable situations to be achieved.

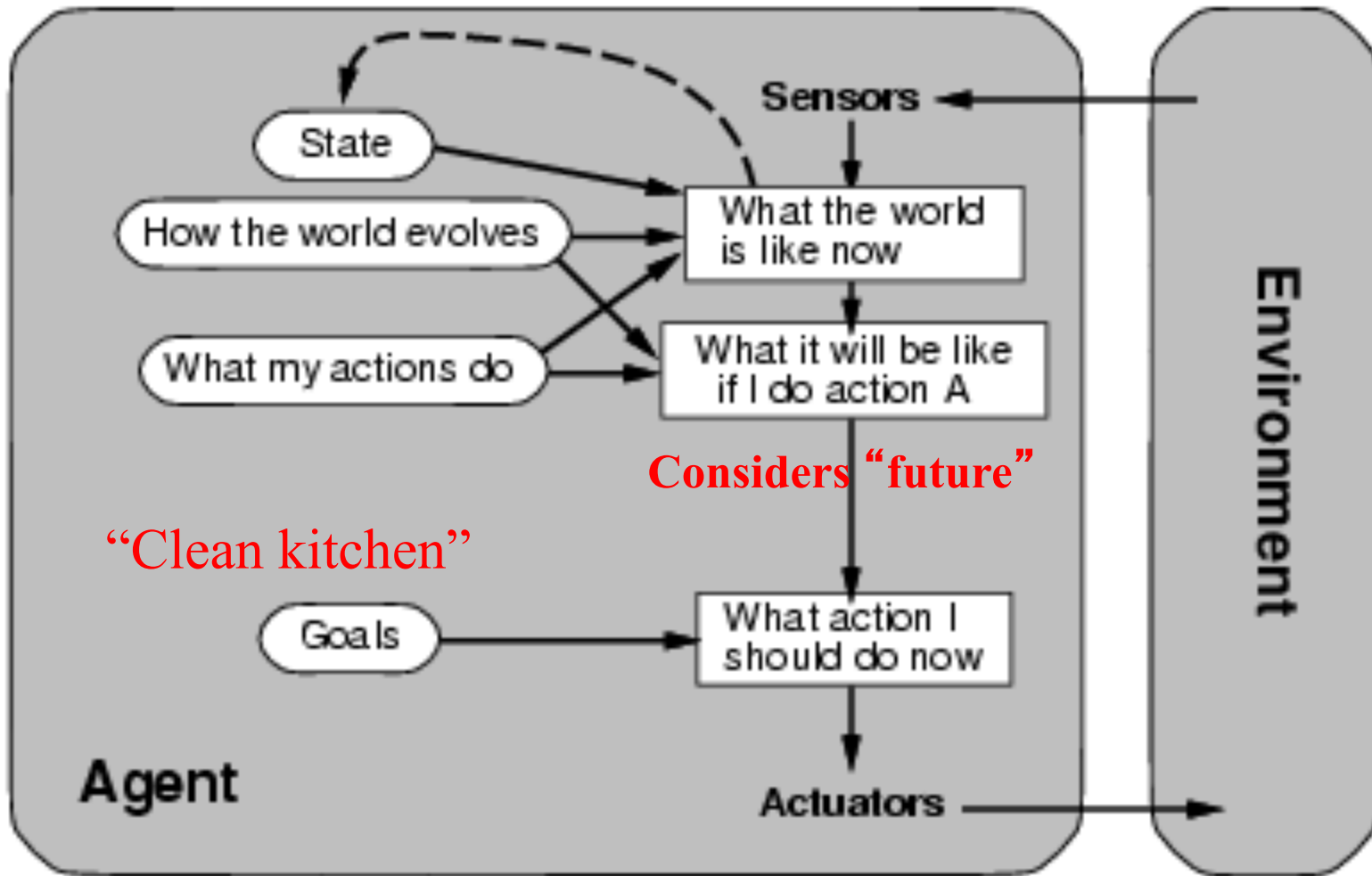Agents of this kind take future events into consideration.
What *sequence* of actions can I take to achieve certain goals?

Choose actions so as to (eventually) achieve a (given or computed) goal.

→ *problem solving and search! (R&N --- Part II, chapters 3 to 6)*

Module:
Problem Solving

Goal-based agents

Inside the diagram:
- State
- How the world evolves
- What my actions do
- Sensors
- What the world is like now
- What it will be like if I do action A
- **Considers "future"**
- **"Clean kitchen"**
- Goals
- What action I should do now
- Agent
- Actuators
- Environment

Agent keeps track of the world state as well as 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

V) --- 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/win" and "unhappy/loss" state, but often need a more general performance measure that describes "degree of happiness."
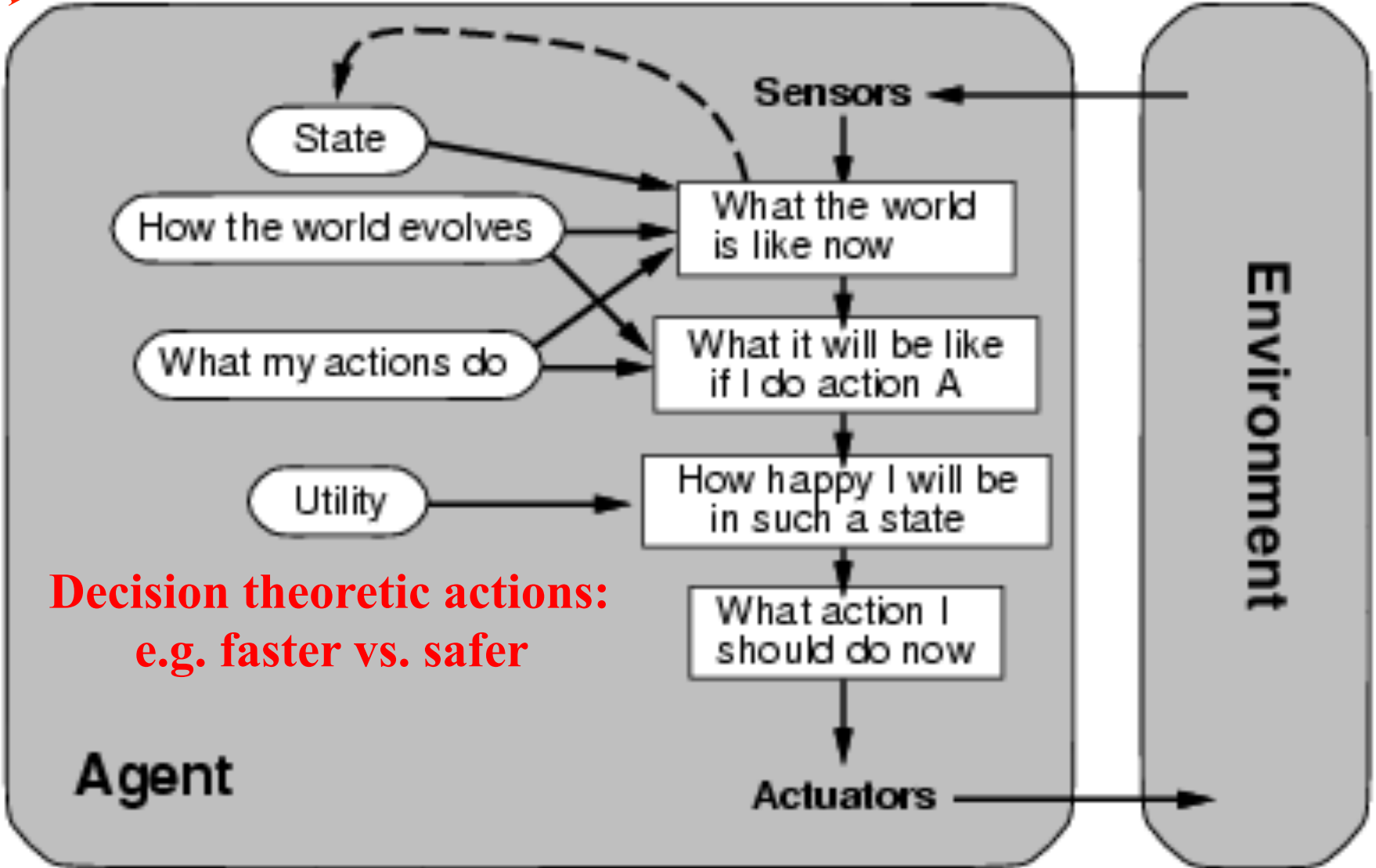
Utility function U: State → R indicating a measure of success or happiness when at a given state.

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.

Utility-based agents
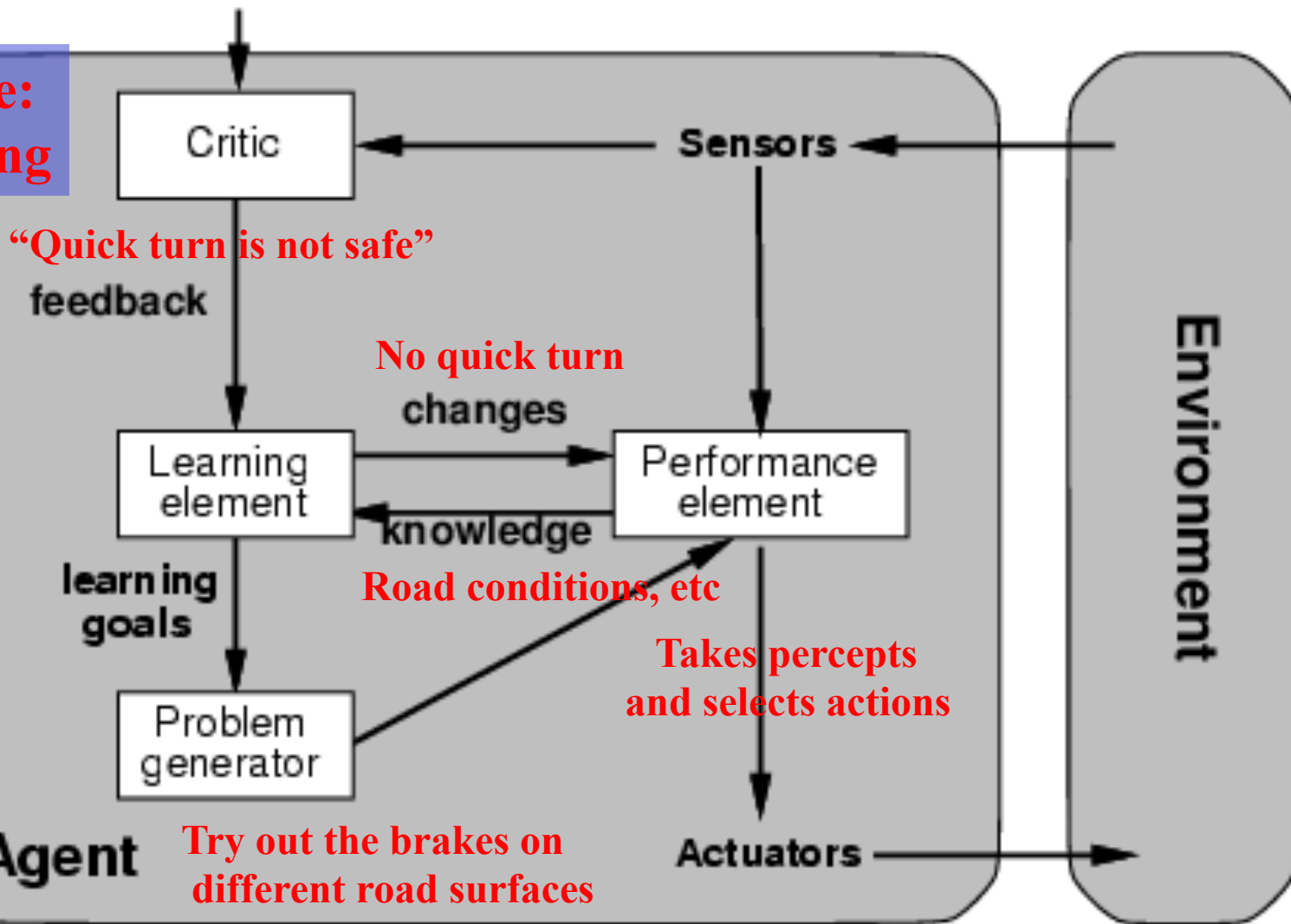

**Decision theoretic actions:
e.g. faster vs. safer**

More complicated when agent needs to learn utility information: Reinforcement learning (based on action payoff)

Module: Learning

Performance standard

Critic ← Sensors ←

"Quick turn is not safe"

feedback

No quick turn changes

Learning element → Performance element

knowledge

Road conditions, etc

learning goals

Takes percepts and selects actions

Problem generator

Agent

Try out the brakes on different road surfaces

Actuators ←

Environment

Summary: agent types

(1) Table-driven agents

– use a percept sequence/action table in memory to find the next action. They are implemented by a (large) lookup table.

(2) Simple reflex agents

– are based on condition-action rules, implemented with an appropriate production system. They are <u>stateless devices which do not have memory of past world states.</u>

(3) Agents with memory - Model-based reflex agents

– have internal state, which is used to keep track of past states of the world.

(4) Agents with goals – Goal-based agents

– are agents that, in addition to state information, have goal information that describes desirable situations. Agents of this kind take future events into consideration.

(5) Utility-based agents

– base their decisions on classic axiomatic utility theory in order to act rationally.

(6) Learning agents

– they have the ability to improve performance through learning.

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.

– **Reflex agents (simple / model-based)** respond immediately to percepts.

– **Goal-based agents** act in order to achieve their goal(s), possible sequence of steps.

– **Utility-based agents** maximize their own utility function.

– **Learning agents** improve their performance through learning.

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.**

Reading: Chapter 2 R&N