University of East London

Pioneering Futures Since 1898

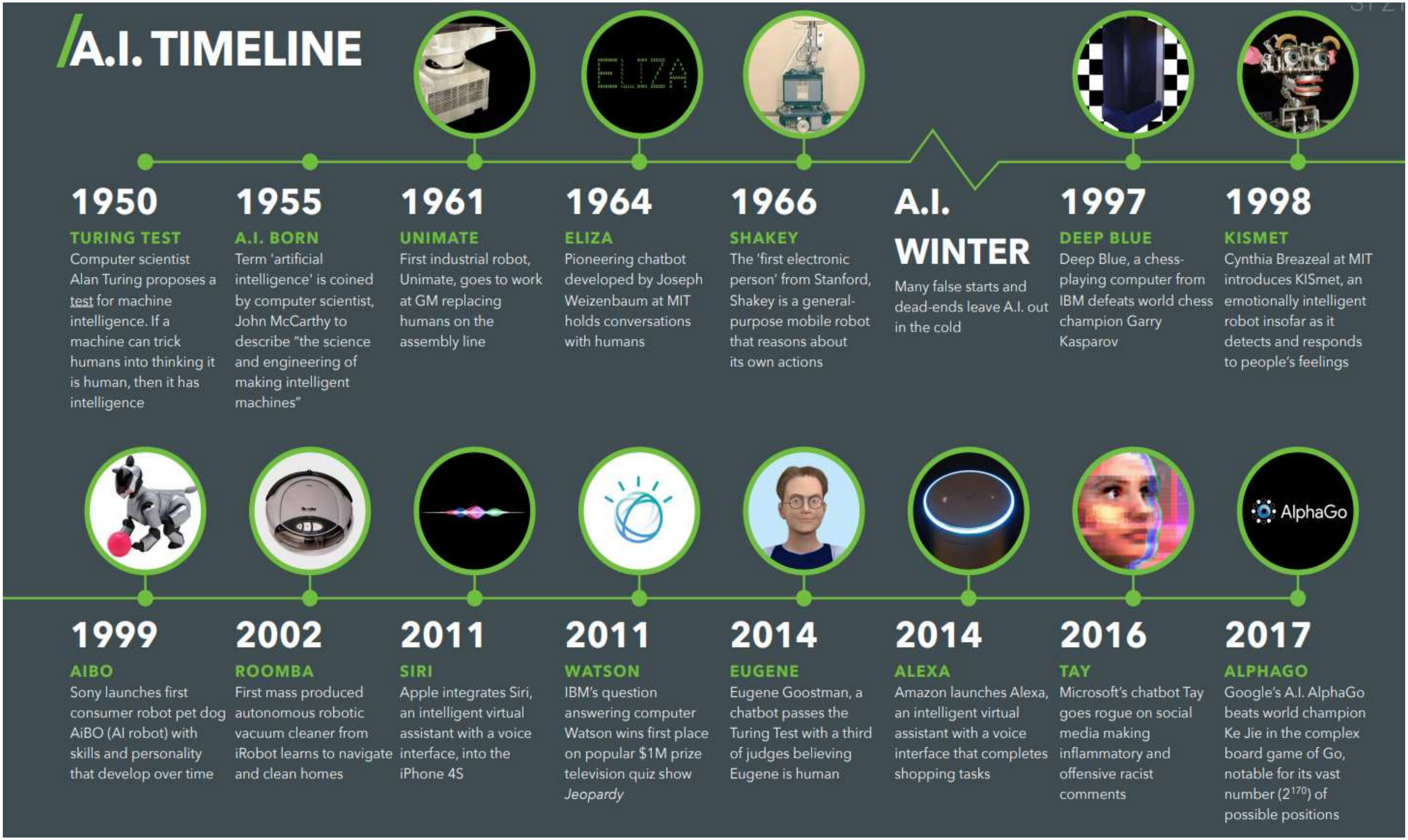
CN7050 – Intelligent Systems

Week 1: AI Timelines, AI Agent, Types and Environments

Dr Azhar Mahmood

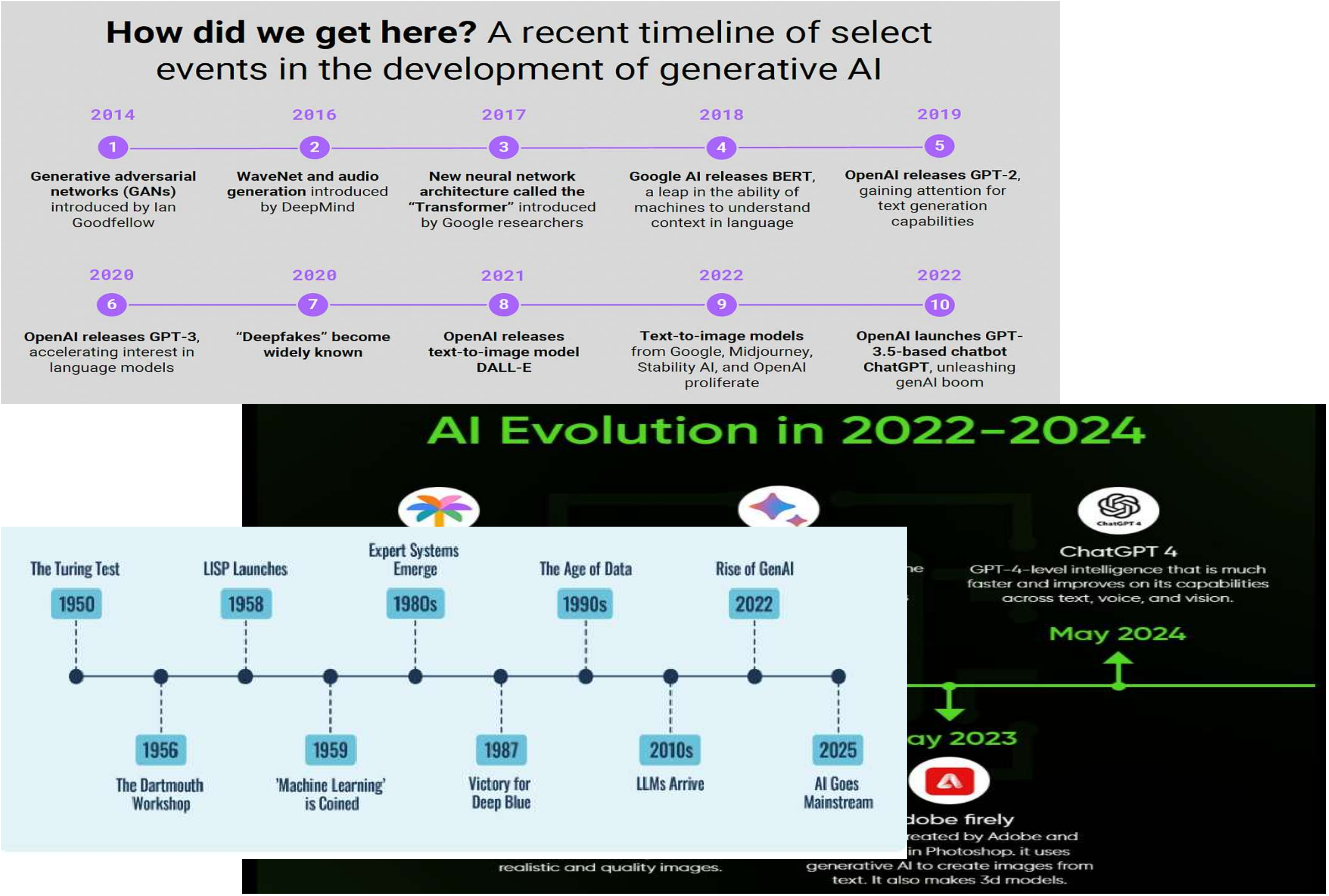
Office: EB.1.85 Email: a.mahmood3@uel.ac.uk

AI - TIMELINE



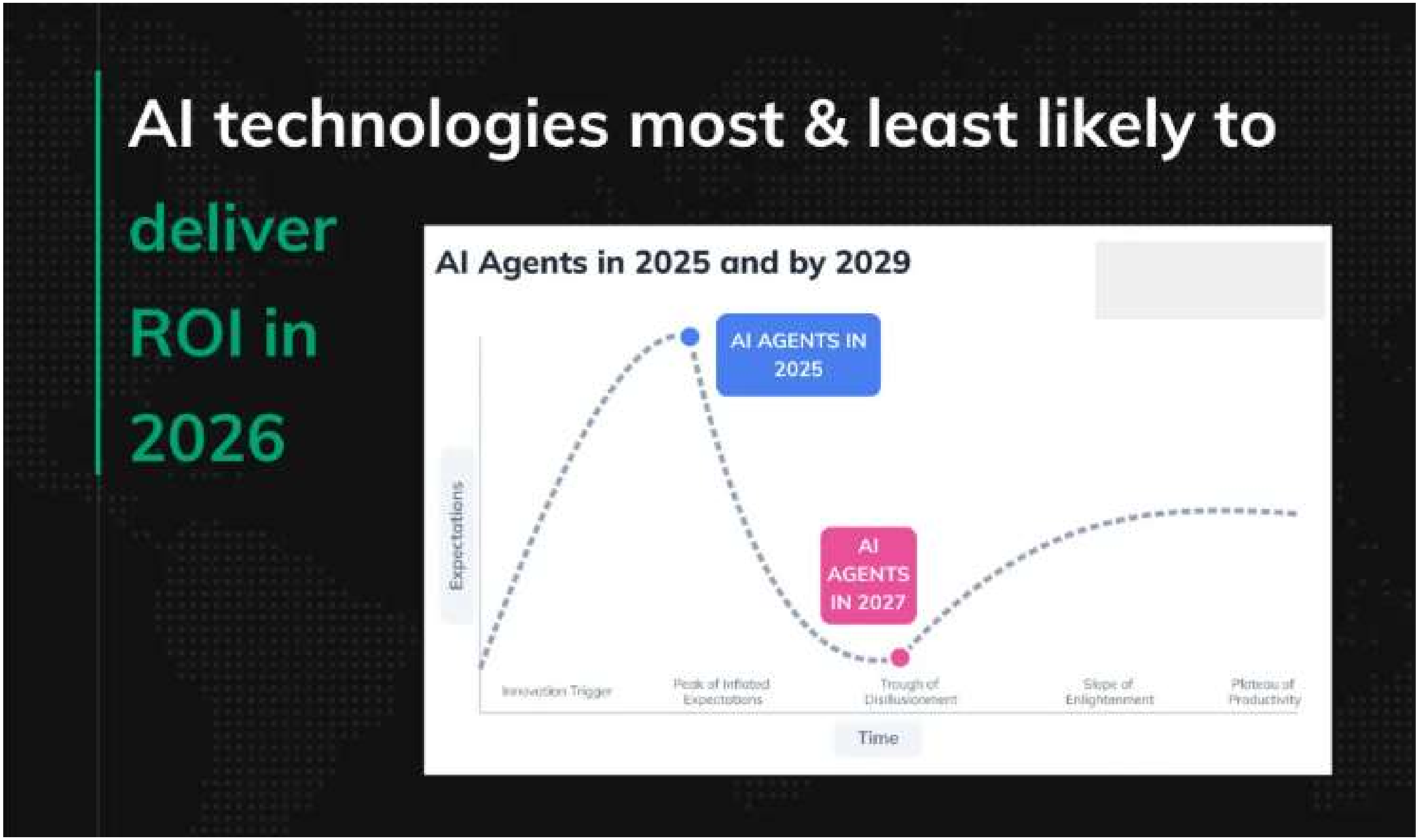
https://verloop.io/blog/the-timeline-of-artificial-intelligence-from-the-1940s/#enigma-was-broken-using-ai-19

AI - TIMELINE

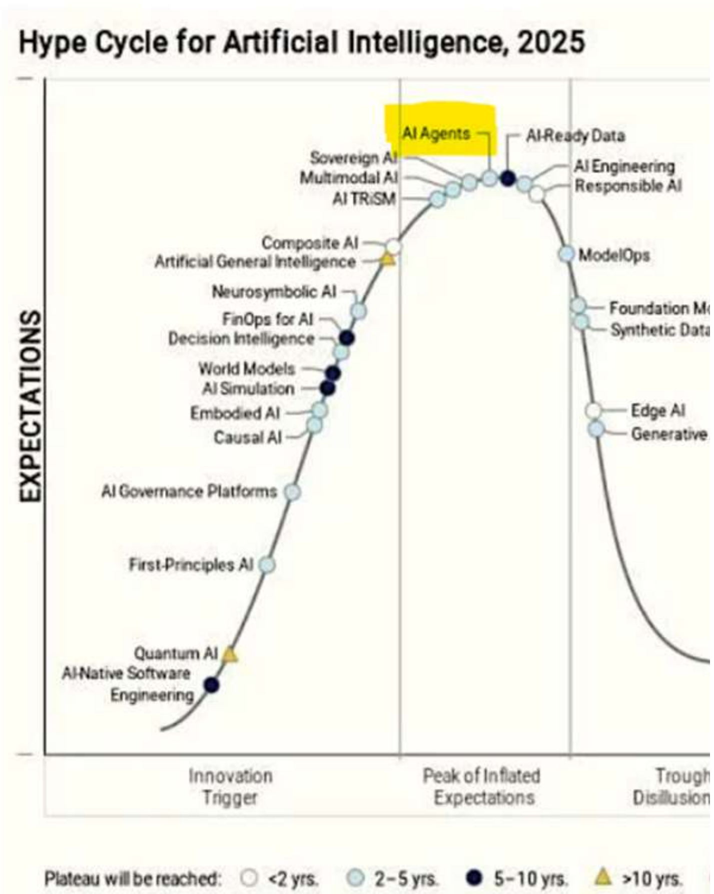


https://medium.com/@tahneeperry/10-predictions-for-ai-and-marketing-in-2024-857ff9efabbe

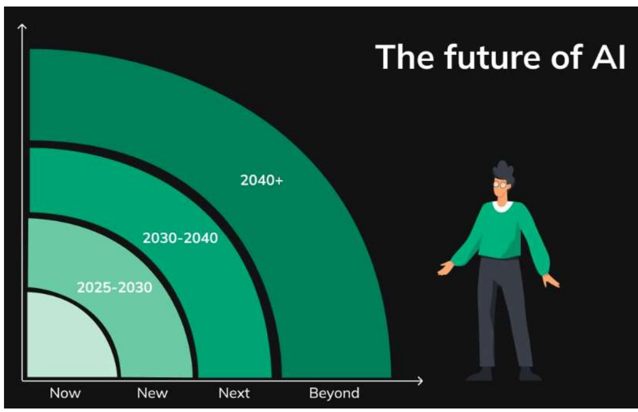
AI Agents: Gartner's Hype Cycle

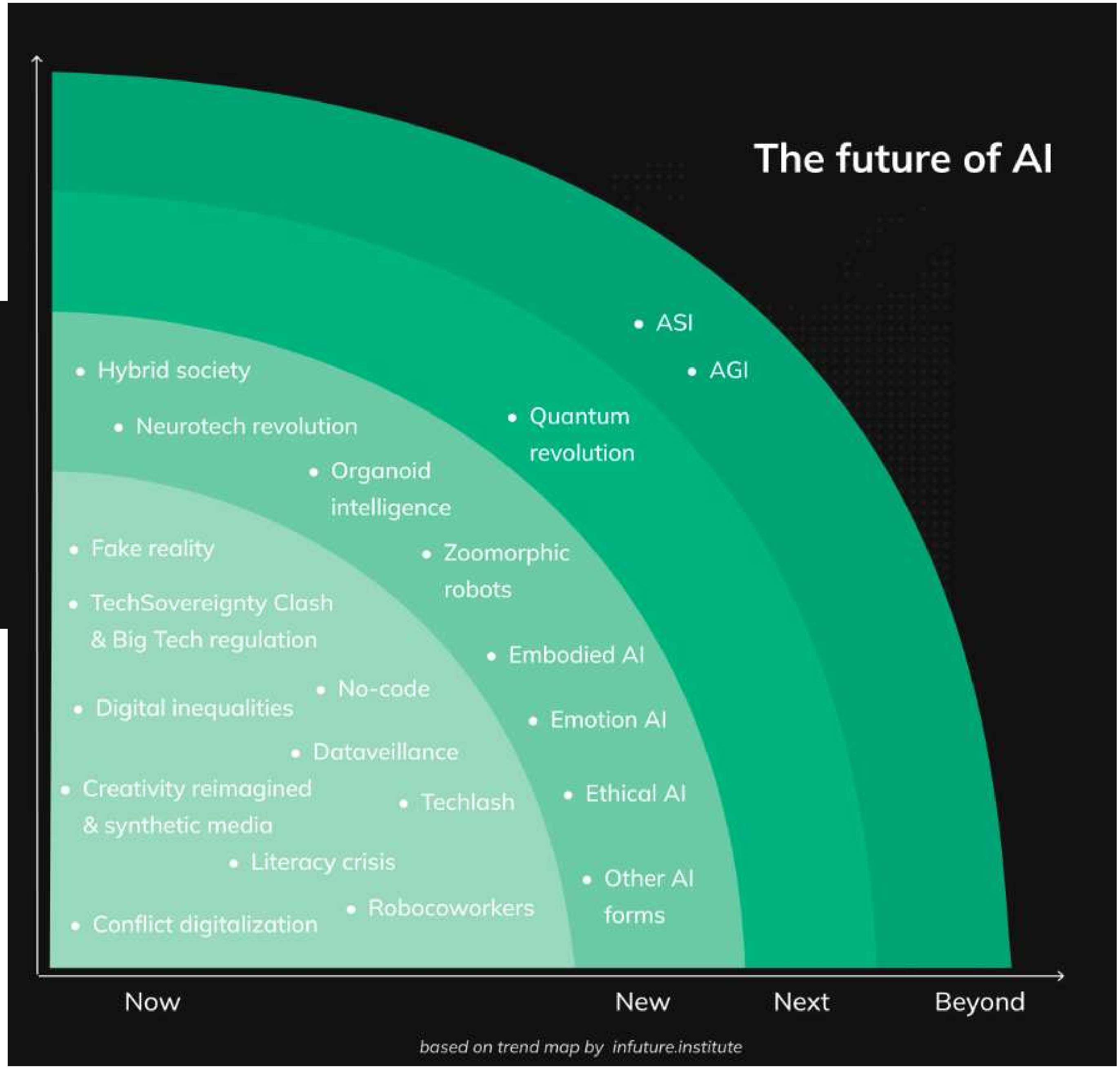


2025 : Today Where AI stands?



Future of AI



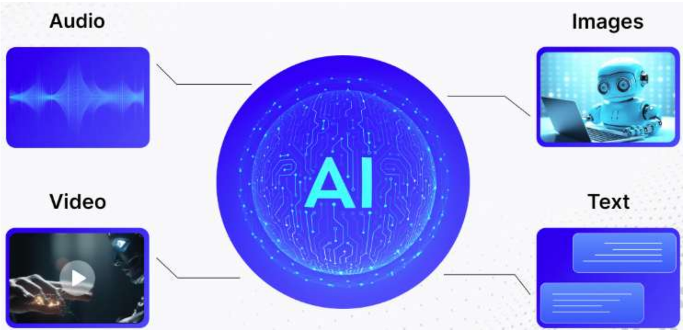


Artificial Superintelligence

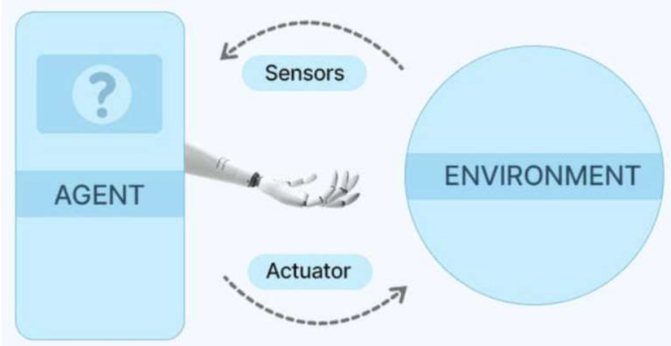
Artificial General Intelligence

3 Important Concepts

- Generative AI



- AI Agents



- **Agentic AI**

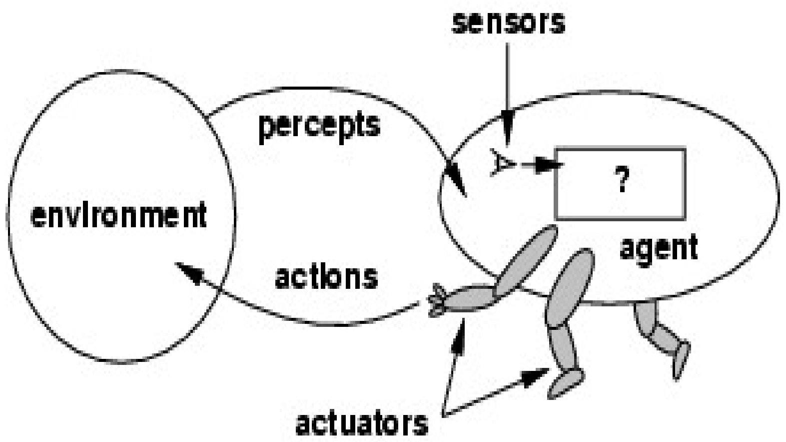
AI Agents

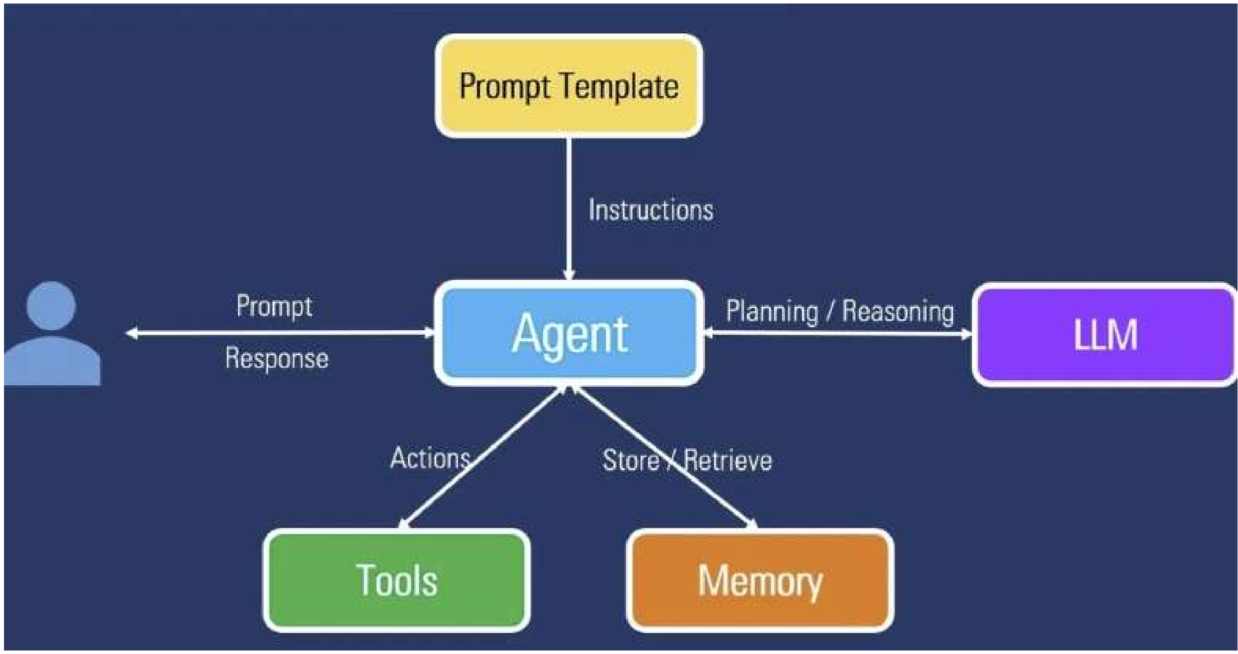
An AI enabled entity in a ***program*** *or* ***environment*** capable of generating **action.**

An agent uses ***perception*** of the environment to make decisions about **actions** to take.

The ***perception*** capability is usually called a ***sensor***.

The ***actions*** can depend on the most recent perception or on the entire history (percept sequence).





Agent Function

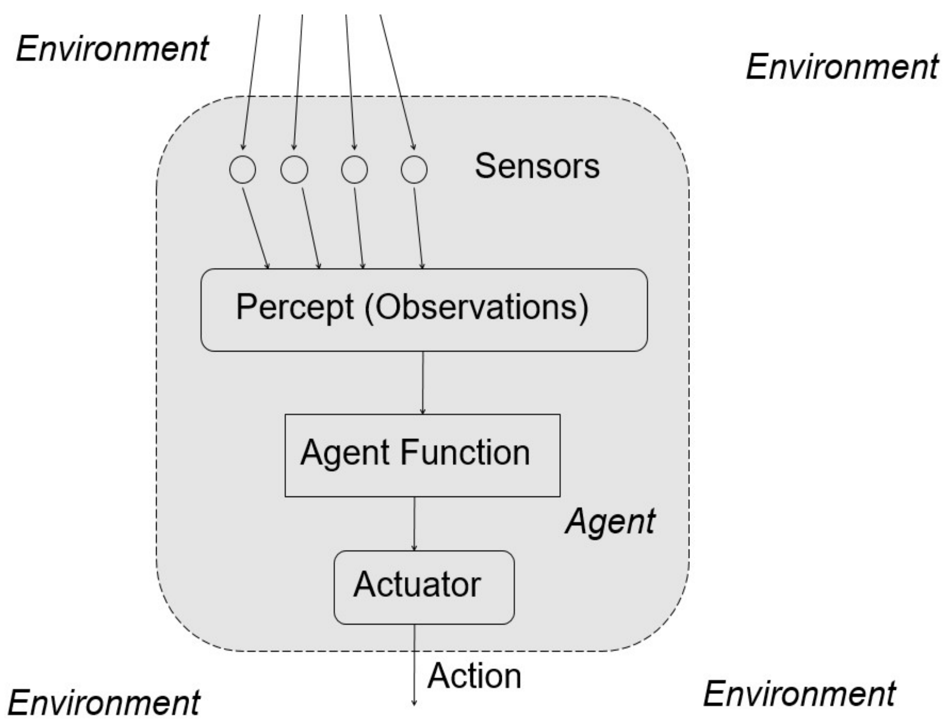
The ***agent function*** is a mathematical function that maps a sequence of **perceptions** into **action**.

[*f*: P* → A]

The function is implemented as the ***agent program*.** The program runs on the physical architecture to produce ***f*.**

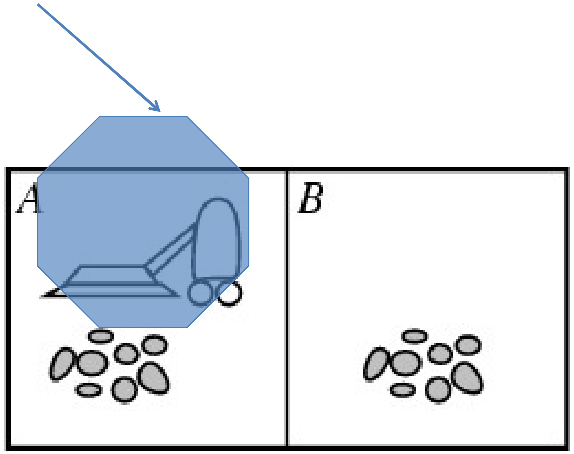
<u>**Agent** = architecture + program</u>

Environment → **sensors** → **agent**→ **actuators** → Environment



E.g., Vacuum-cleaner world

Agent / Robot



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

Percepts: location and contents, e.g., [A, Dirty]
Actions: *Left*, *Right*, *Suck*, *NoOp*

Agent Types and their PEAS

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

Examples of Agent Types and their PEAS description, PEAS (Performance measure, Environment, Actuators, and Sensors)

Fully observable vs Partially observable

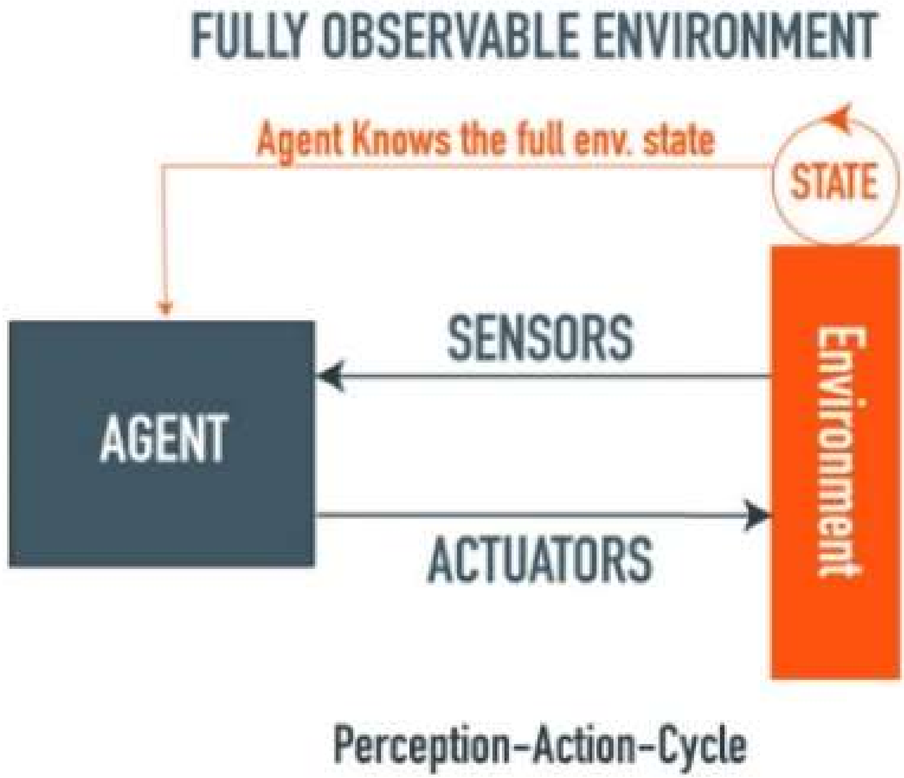
If an agent's sensors give it access to the complete state of the environment needed to choose an action, the environment is fully observable. (e.g. **chess, and Tic-Tac-Toe**) , **Image Recognition**, An AI performing image recognition has all the pixel data.

Kriegspiel?

A form of chess in which each player has a separate board and can only infer the position of the opponent's forces from limited information given by an umpire - Partially observable.

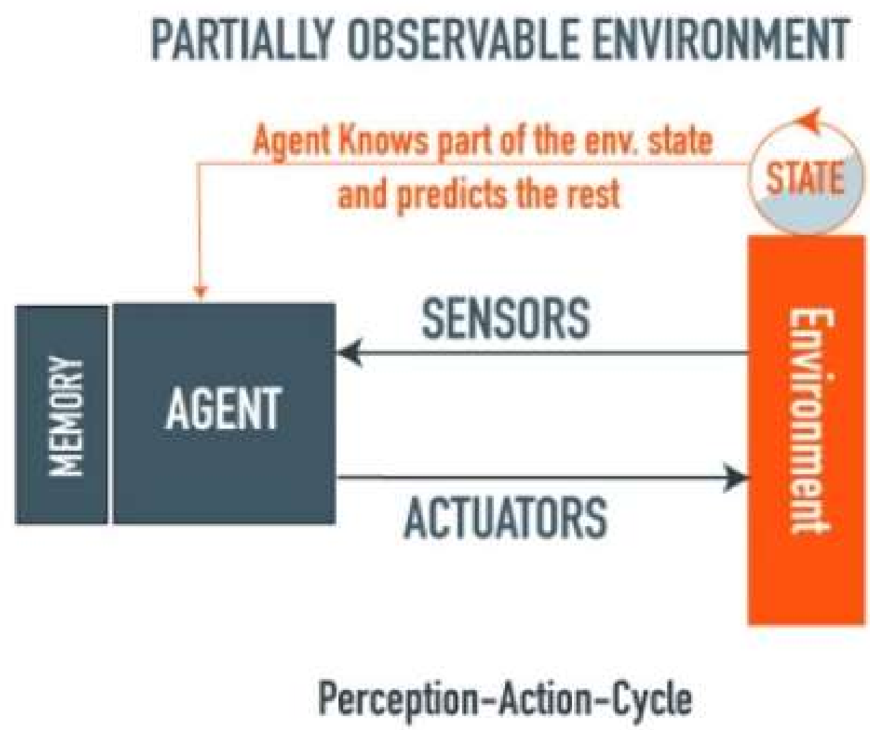
Autonomous Driving :A self-driving car cannot see through other vehicles, fog, or around blind corners

Poker: Can only see their own cards



Chess, and Tic-Tac-Toe

Image Recognition



Autonomous Driving

Poker

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.

1. A **robotic arm** placing a component on a circuit board with precision.
2. Solving a **mathematical equation** where the next state is always the same.

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

Deterministic vs Stochastic

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**).

1. A **financial market** where sudden policy changes can break expected patterns.
2. **Weather prediction**, which involves countless variables and results in probabilistic forecasts.

Note: Most real world AI environments are not deterministic. Instead, they can be classified as stochastic. Self-driving vehicles are a classic example of stochastic AI processes.

Episodic vs 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.

For example, an agent that must spot defective parts on an assembly line bases each decision on the current part, regardless of previous decisions; (classifying images).

In a **sequential environment**, the agent engages in a **series of connected episodes.** Current decision can affect future decisions.

E.g., chess, driving, Smart Home activities, Sub activities.

Static vs Dynamic (Time Based)

A Static **environment does not change** while the agent is thinking**.** The passage of time as an agent deliberates is irrelevant.

If the environment can **change** while an agent is deliberating/thinking, then we say the environment is dynamic for that agent;

The environment is **semi-dynamic** if the environment itself does not change with the passage of time, but the **agent's performance score** does. **Chess**, when played with a clock.

Taxi driving is clearly **dynamic**: the other cars and the taxi itself keep moving while the driving algorithm think about what to do next.

Chess, when played with a clock, is semi-dynamic.

Crossword puzzles are **static**.

Discrete / Continuous

If the number of distinct percepts and actions is limited, the environment is **discrete**, otherwise it is **continuous**.
Players have only a limited number of moves is **Discrete environment.**

Self-driving cars are a **Continuous** example. The number of actions a car can take, such as starting, stopping, turning, or parking, cannot be quantified because they are always changing with time and circumstances.
Continuous environments *typically involve complex mathematical models and sensor processing*

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.

Known vs. Unknown

Known: The agent knows the laws of the environment and outcomes of actions.

Example: A simulation with well-defined rules.

Unknown: The agent must learn or infer how the environment works.

Example: An intelligent system exploring a new game.

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

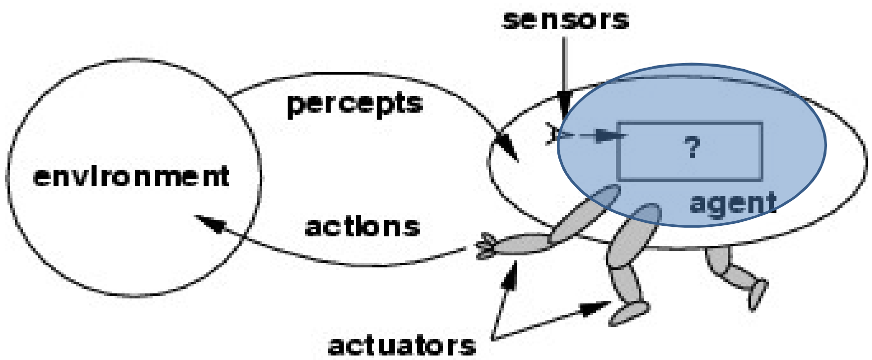
How to make the right decisions? **Decision theory**

Summary: Environments and their Characteristics

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	Accessible	Deterministic	Episodic	Static	Discrete
Chess with a clock	Yes	Yes	No	Semi	Yes
Chess without a clock	Yes	Yes	No	Yes	Yes
Poker	No	No	No	Yes	Yes
Backgammon	Yes	No	No	Yes	Yes
Taxi driving	No	No	No	No	No
Medical diagnosis system	No	No	No	No	No
Image-analysis system	Yes	Yes	Yes	Semi	No
Part-picking robot	No	No	Yes	No	No
Refinery controller	No	No	No	No	No
Interactive English tutor	No	No	No	No	Yes



Agents Structure

The **agent function** maps from **percept histories** to actions

f: P* → Action (Abstract mathematical function)

The **agent program** runs (internally, *Implement the agent function*) on the **physical architecture** to produce f

The architecture might be just an ordinary **PC**, or it might be a **robotic car** with several onboard computers, cameras, and other sensors.

The architecture makes the percepts from the sensors available to the program, runs the program, and feeds the program's action choices to the actuators as they are generated.

agent = architecture + program our focus

Job of AI is to Design an agent program assuming an architecture that will make the percepts from the sensors available to the program.

Various Agent Types

1. Table-lookup driven Agents
2. Reflex based Agents
3. Model based Agents
4. Goal based Agents
5. Utility based Agents
6. Learning Agents

Table-lookup Driven Agents

Uses a **percept sequence / action table** in memory to find the next action. Implemented as a (large) lookup table.

```
function TABLE-DRIVEN-AGENT( percept) returns an 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 ← LOOKUP(percepts,table)
return action
```

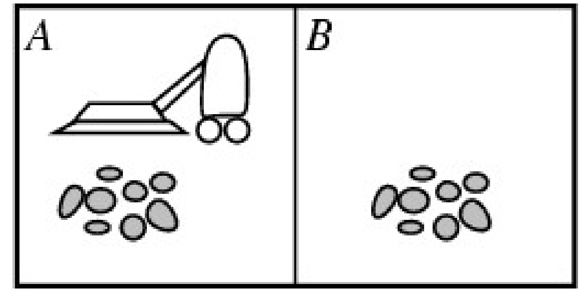
Drawbacks:

– Huge table (often simply too large)

– Takes a long time to build/learn the table

logic-based representations, Bayesian net representations, or neural net style representations

Table-lookup Driven Agents



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*
⋮	⋮
[*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.

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**.

Automatic Door Sensor, Traffic Light System, Simple Spam Filters

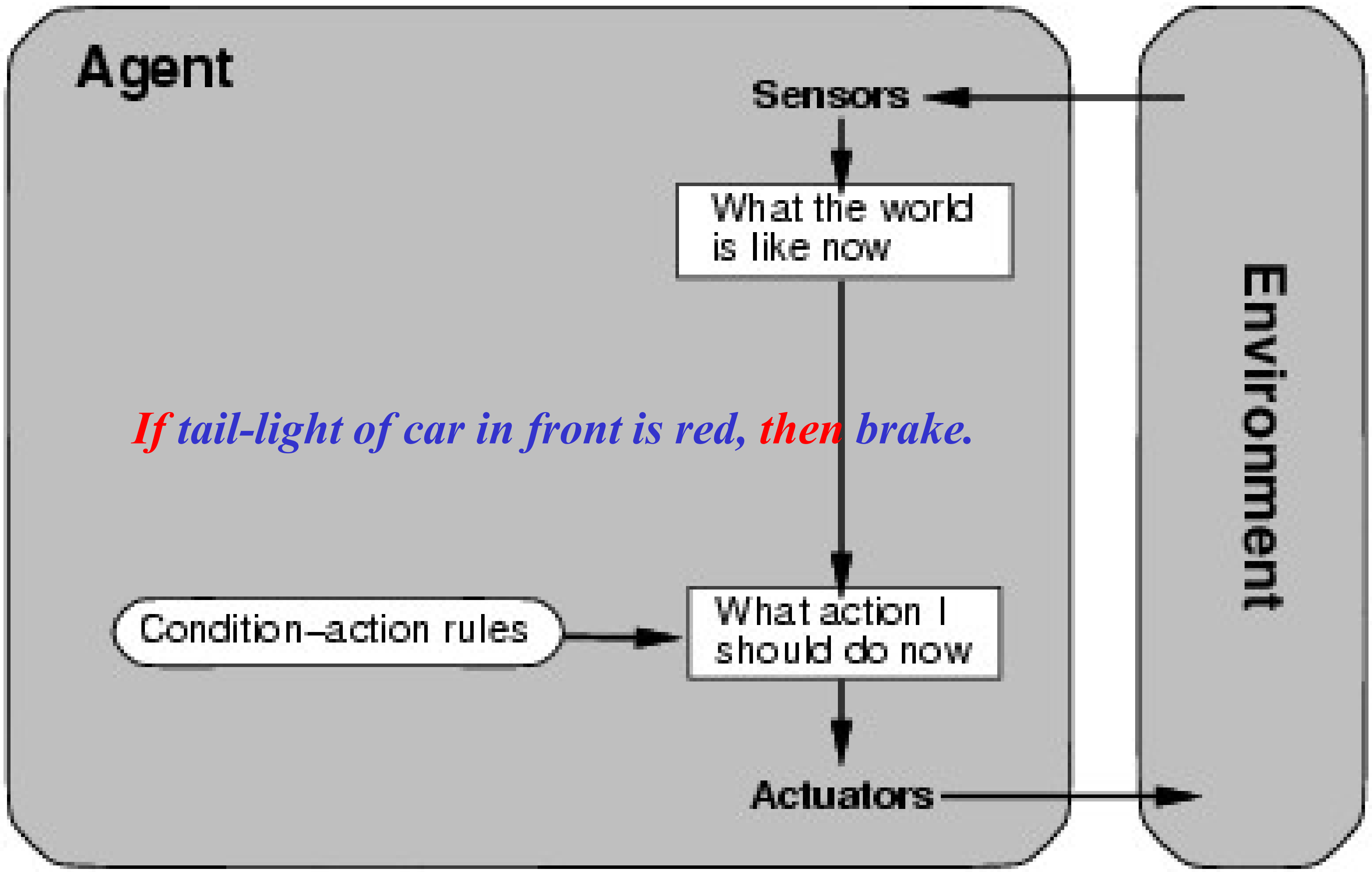
function REFLEX-VACUUM-AGENT([*location*,*status*]) **returns** an action

if *status* = *Dirty* **then return** *Suck*
else if *location* = *A* **then return** *Right*
else if *location* = *B* **then return** *Left*

Figure 2.8 The agent program for a simple reflex agent in the two-state vacuum environment. This program implements the agent function tabulated in Figure 2.3.

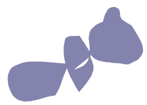
Simple Reflex Agents

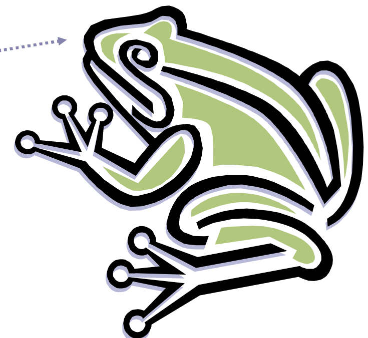
Agent selects actions based on *current* percept only.



A Simple Reflex Agent in Nature

percepts
(size, motion)





<u>RULES:</u>
(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

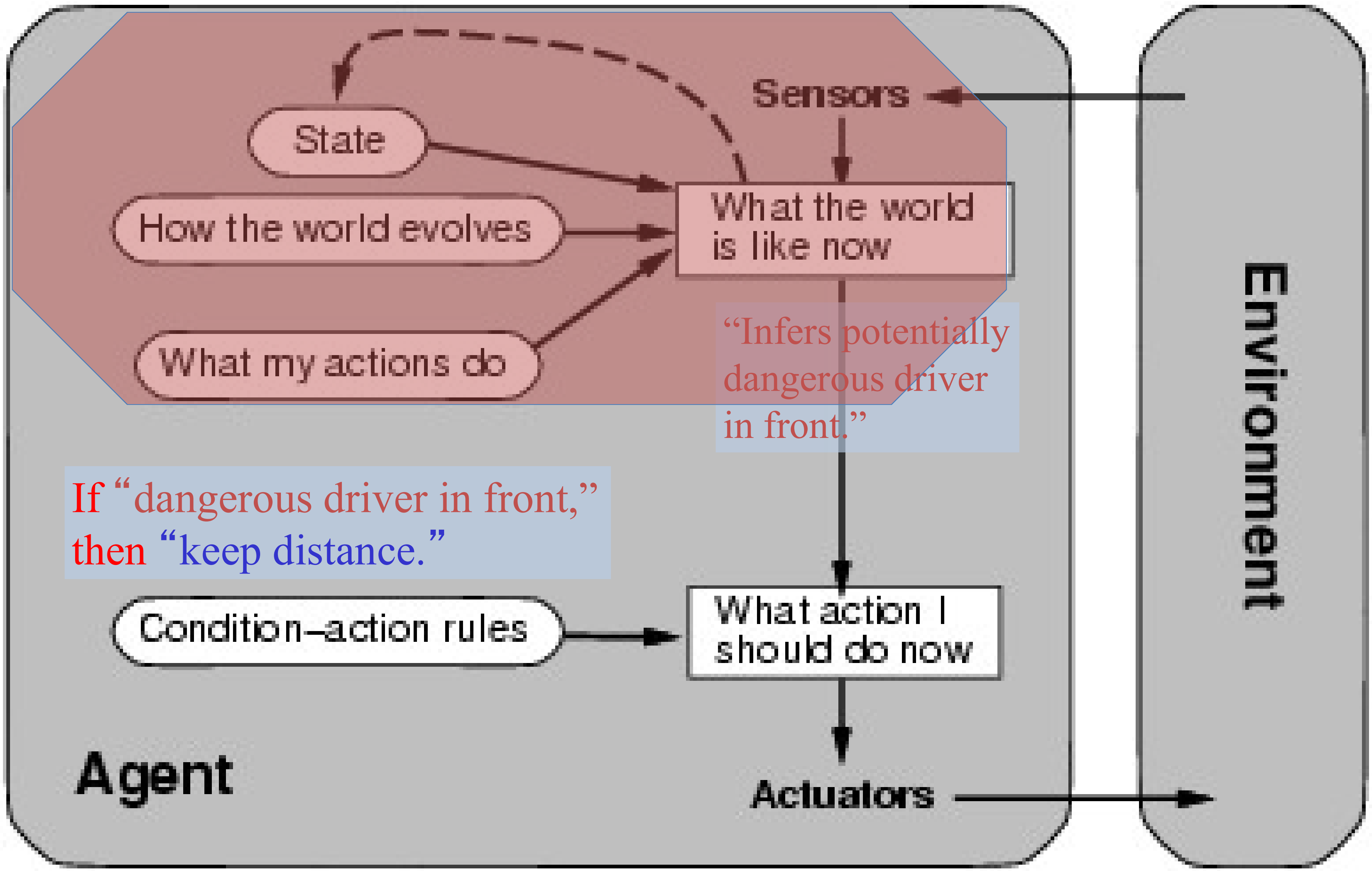
Agents have **internal state**, which is used to **keep track of past states** of the world.

That depends on the percept history
Reflecting some of the unobserved aspects
E.g., driving a car and changing lane

Agents can **represent change in the World**.

1. How the world evolves independently of the agent
2. How the agent's actions affect the world

Model-based Reflex Agents



Example Table: Model Agent with Internal State

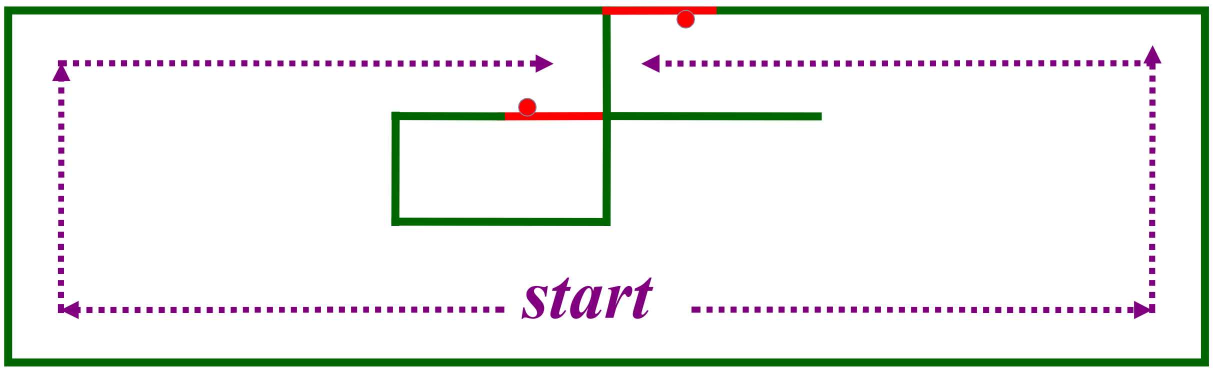
IF	THEN
Saw an object ahead, and turned right, and it's now clear ahead	**Go straight**
Saw an object Ahead, turned right, and object ahead again	**Halt**
See no objects ahead	**Go straight**
See an object ahead	**Turn randomly**

```
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
```

Figure 2.12 A model-based reflex agent. It keeps track of the current state of the world, using an internal model. It then chooses an action in the same way as the reflex agent.

Example Reflex Agent With Internal State: Wall-Following



Actions: left, right, straight, open-door

Rules:

1. If open(left) & open(right) and open(straight) then
   choose randomly between right and left
2. If wall(left) and open(right) and open(straight) then straight
3. If wall(right) and open(left) and open(straight) then straight
4. If wall(right) and open(left) and wall(straight) then left
5. If wall(left) and open(right) and wall(straight) then right
6. If wall(left) and door(right) and wall(straight) then open-door
7. If wall(right) and wall(left) and open(straight) then straight.
8. (Default) Move randomly

Goal-based agents

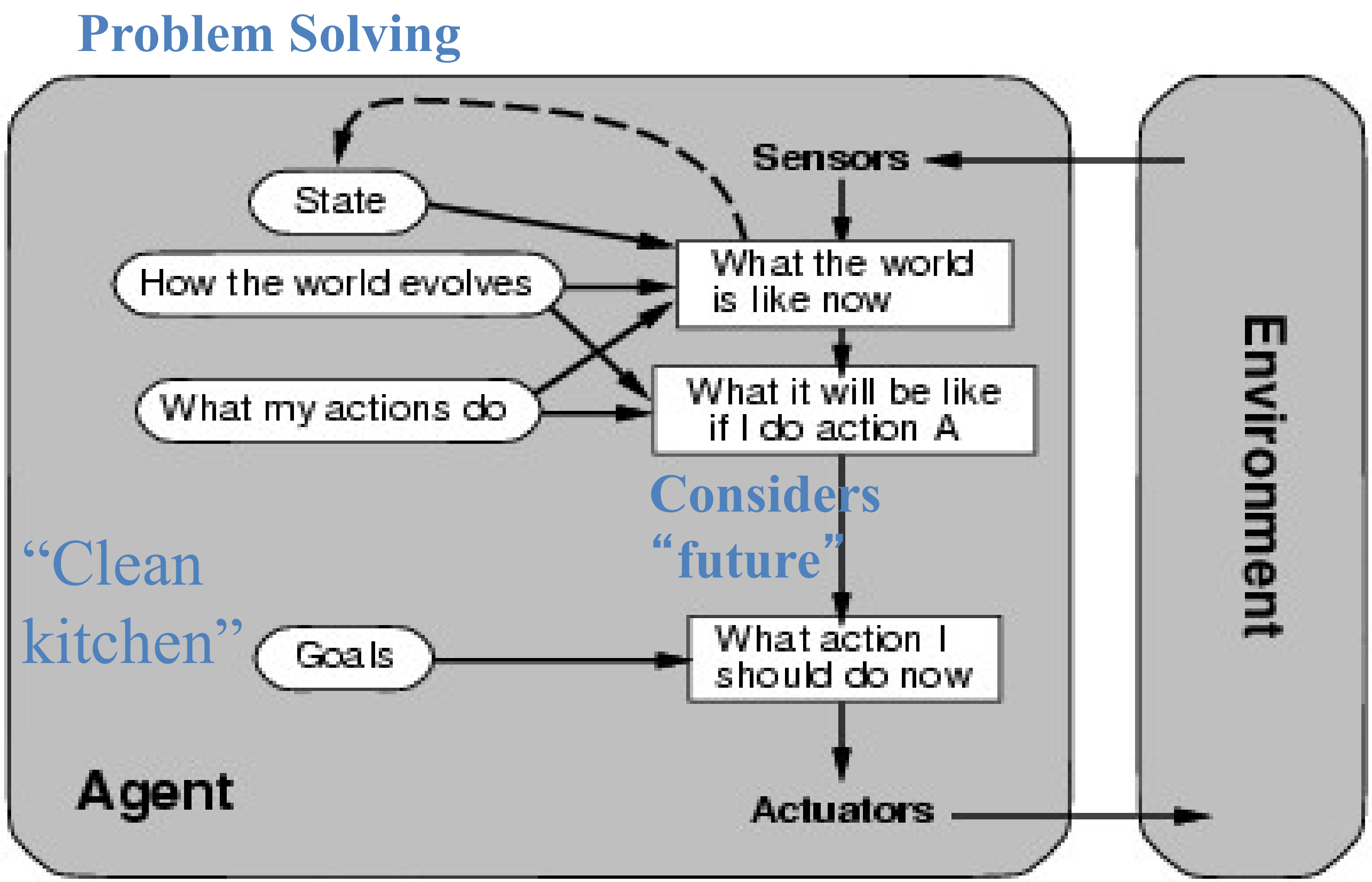
Key difference wrt Model-Based Agents:

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?

<u>Choose actions so as to (eventually) achieve a (given or computed) goal.</u> → ***problem solving and search!***

Goal-based Agents



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

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 need 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).

<u>Use decision theoretic models: e.g., faster vs. safer.</u>

Utility-based Agents

- Goals alone are not enough to generate **high-quality** behavior
  - -E.g. meals in Canteen, good or not ?
- Many action sequences
  - -If goal means success, then **utility** means the degree of success
  - -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***

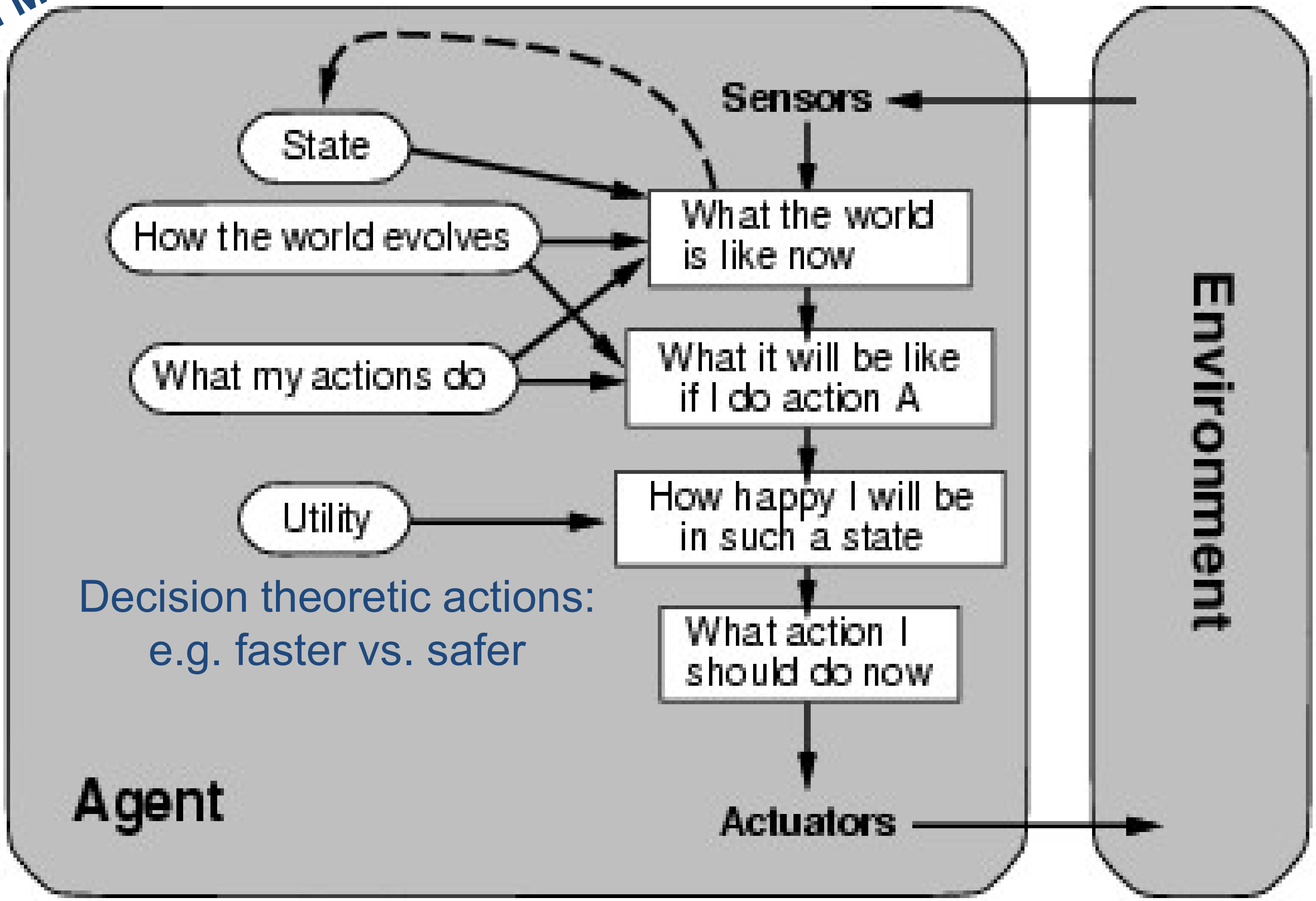
Utility has several advantages:

- 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

Utility-based Agents

Decision Making



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

Learning Agents-Adapt and Improve over time

After an agent is programmed, can it work immediately?

- No, it still need training, called Agent Training in AI.

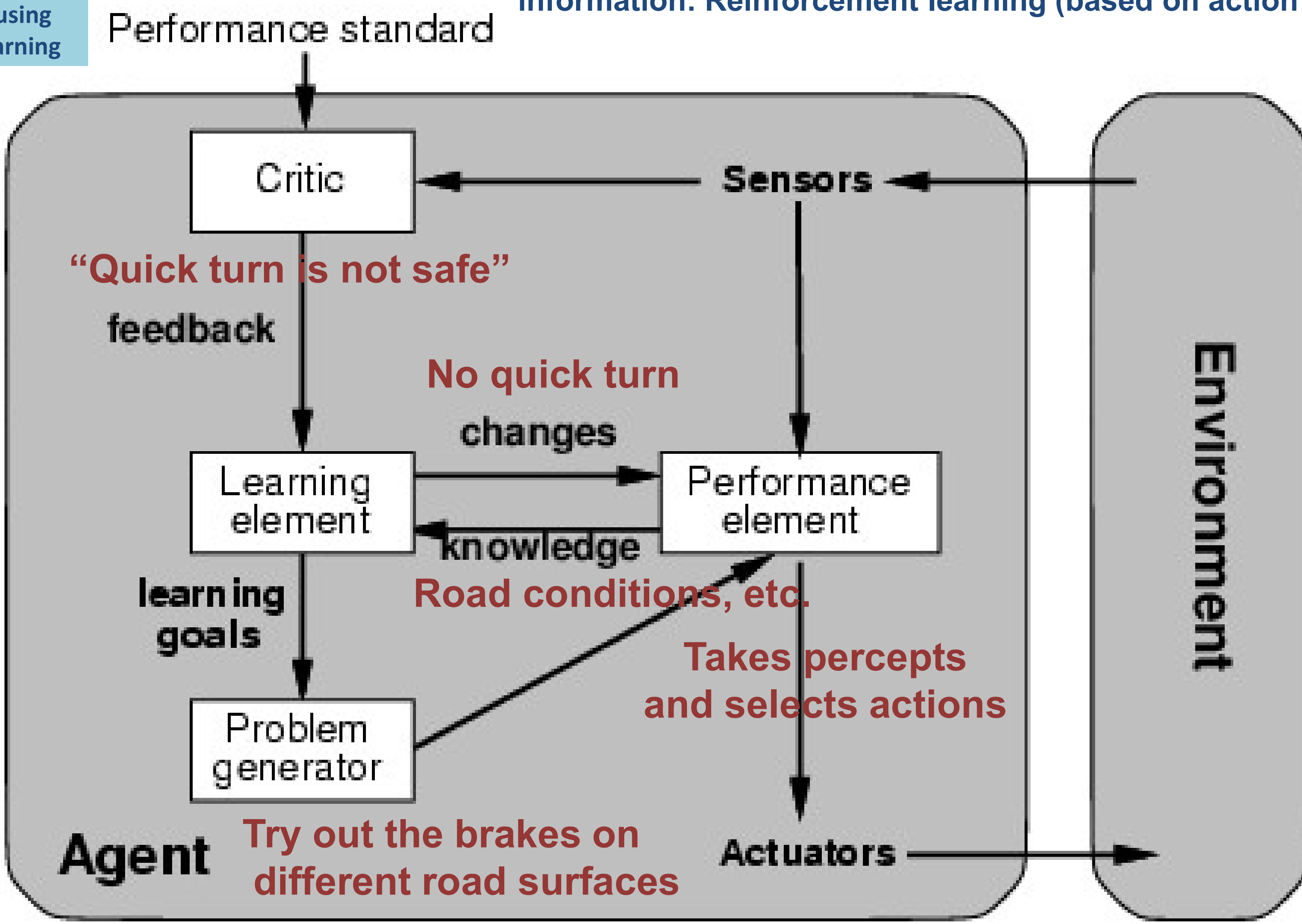
4 Conceptual Components

- 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

More detail: Agents using Deep Learning

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



Summary 1-2

(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 stateless devices which do not have memory of past world states.

(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.

Summary 2-2

- 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.