Intelligent Agents (RN Ch. 1 + Ch. 2)

Nicholas Mattei, *Tulane University*

CMPS3140 – Introduction to Artificial Intelligence

Spring 2020

https://nmattei.github.io/cmps3140/

Many Thanks

Slides are based off contributions from many including

- Anca Dragan, Dan Klein, and Pieter Abbeel from UC Berkeley CS188 Intro to AI at UC Berkeley (ai.berkeley.edu).
- Jihun Hamm (Tulane University)
- Lirong Xia (RPI)
- K. Brent Venable (IHMC)

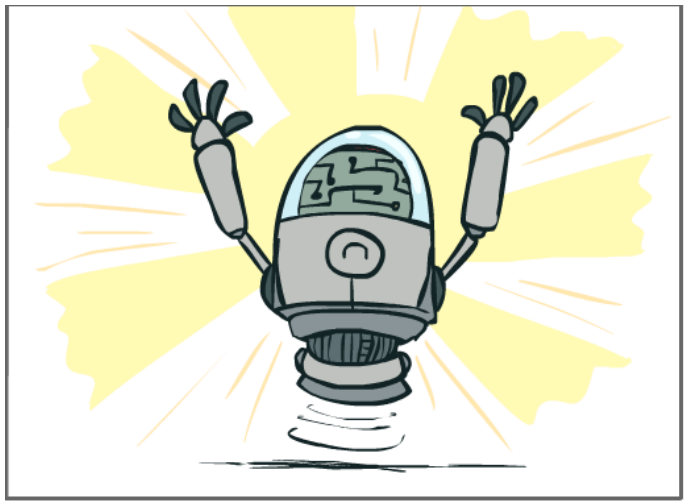
All About Agents

Tulane University

1. Example Agents
    1. Vacuum world
    2. Rationality

2. Task environments
    1. PEAS (Performance measure, Environment, Actuators, Sensors)
    2. Properties of task environments

3. Agent types / architecture



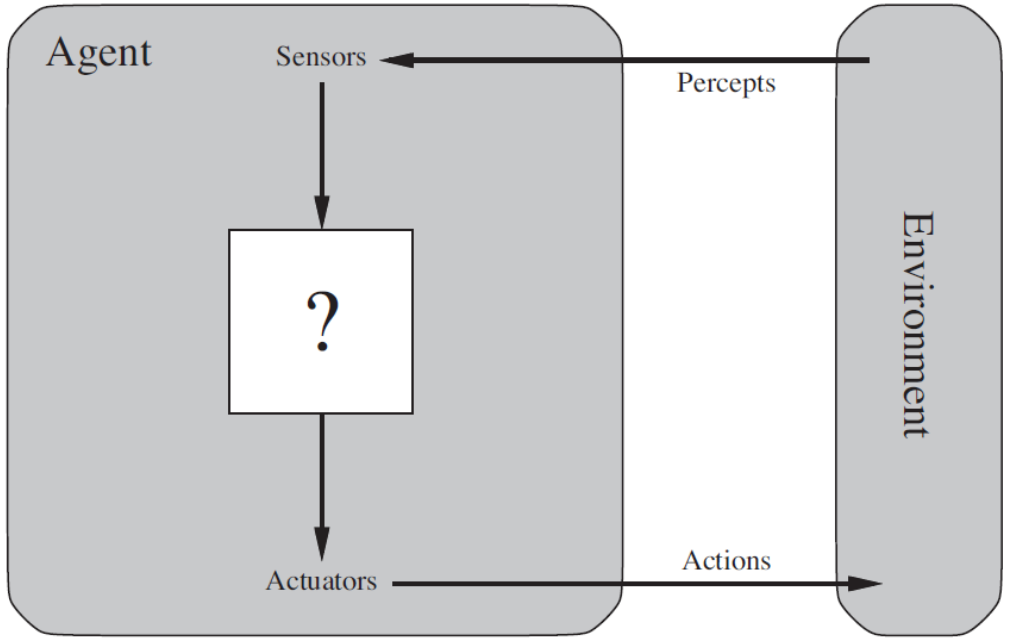
1/3. Agent

Tulane University

- Agent: Entity that *perceives* and *acts*.

- Formally: An **agent** is anything that can be viewed as **perceiving** its **environment** through **sensors** and **acting** upon that environment through **actuators.**

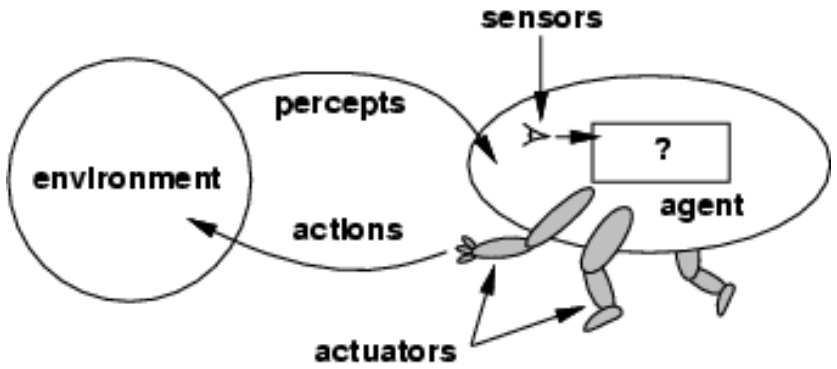
- What is a human agent?
  - Sensors: eyes, ears, touch..
  - Actuators: hands, legs, mouth…
- What is a robotic agent?
  - Sensors: cameras and infrared range finders
  - Actuators: motors, arms, legs..
- What is a software agent?
  - Sensors: image from screen, text search…
  - Actuators: screen, game moves..



1/3. Agent

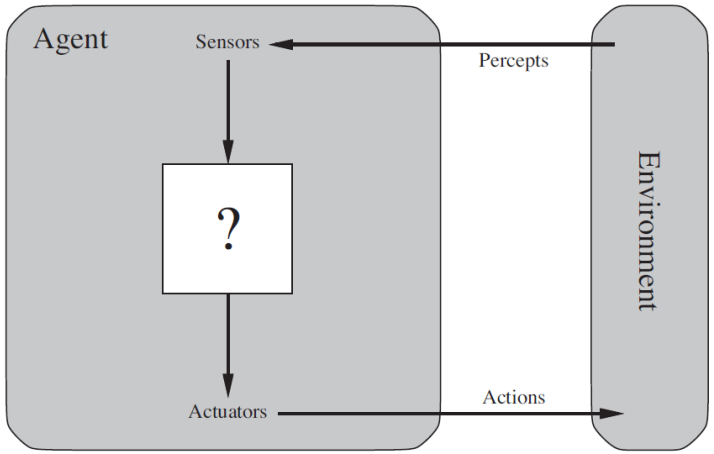
- Agent: Entity that *perceives* and *acts*.

- The **agent function** maps from histories or sequences of percept to actions:
  - *f: P* -> A*



- The **agent program** runs on the physical architecture to produce *f*.

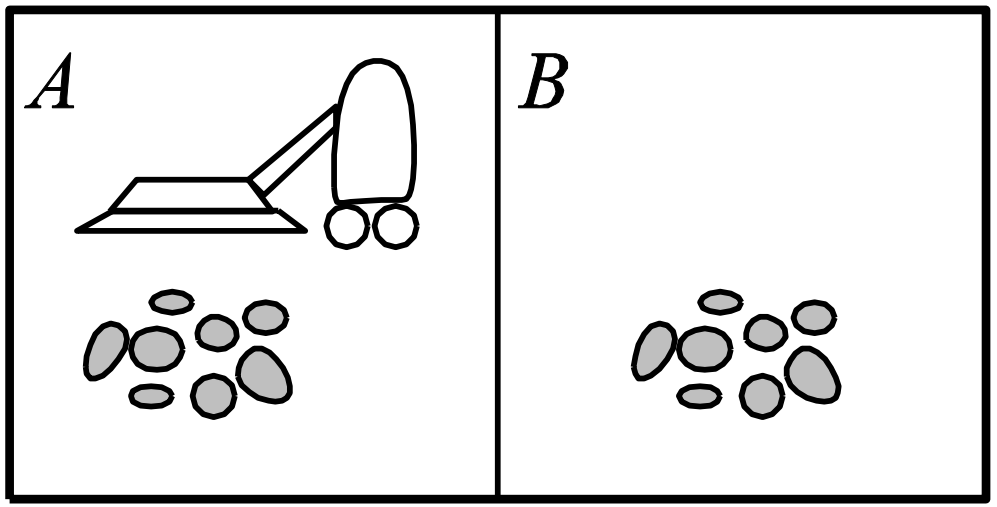
- agent = architecture + program



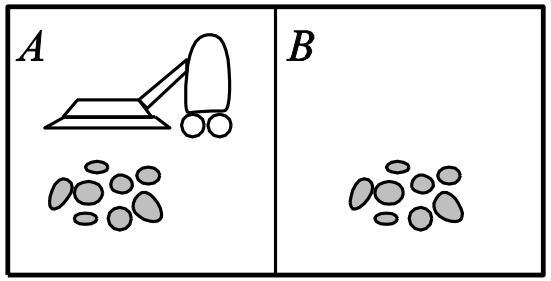
Vacuum Cleaner World

Tulane University

- Two locations (A & B)
- Vacuum agent perceives the location and the presence of dirt at the location
- Vacuum agent can move *left*, *right*, or *suck* dirt



Agent function



Percept sequence	Action
$[A, Clean]$	$Right$
$[A, Dirty]$	$Suck$
$[B, Clean]$	$Left$
$[B, Dirty]$	$Suck$
$[A, Clean], [A, Clean]$	$Right$
$[A, Clean], [A, Dirty]$	$Suck$
⋮	⋮

- An agent is fully characterized by the agent function: *percept sequence* → *action*
- This is an example of a Table Lookup Agent

Agent Functions and Programs

Tulane University

- An agent is completely specified by the agent function mapping percept sequences to actions
- Aim of AI: design programs that implement the agent functions concisely
  - **Agent = architecture + program**
- Architecture:
  - feeds the percepts from the sensors to the program
  - runs the program
  - feeds the actions to the actuators

```
function TABLE-LOOKUP_AGENT(percept)
returns an action
persistent: 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:
  - Invoked for each new percept
  - Huge table (chess 10^{150} entries)
  - Takes a long time to build the table
  - No autonomy
  - Even with learning, need a long time to learn the table entries
  - Who knows how to build it?!

Evaluating the basic agent

Tulane University

- 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

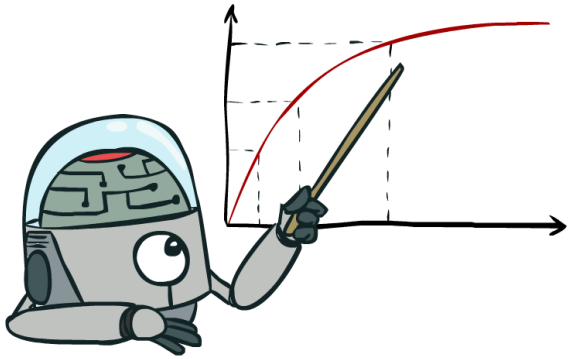
- Need a performance measure to evaluate agent's behavior…
  - Maximize amount of dirt picked up?
  - Maximize ratio of dirt to energy expended?
  - Maximize ration of dirt to combination of energy and time?

- Wrong performance measures lead to rational but unwanted outcome
  - Vacuum: Maximize amount of dirt sucked → ?
  - The Matrix/Skynet: End pain and bring peace to the world → ?
    (By the way, Elon Musk donated $10M to keep AI beneficial to humanity, Jan 15, 2015)

Rationality

Tulane University

- *Rational Agent: For each possible percept sequence, a rational agent should select an action that is expected to maximize its (expected) performance measure, given the evidence provided by the percept sequence and whatever built-in knowledge the agent has.*
  - i.e., maximize it's expected utility given everything it knows!



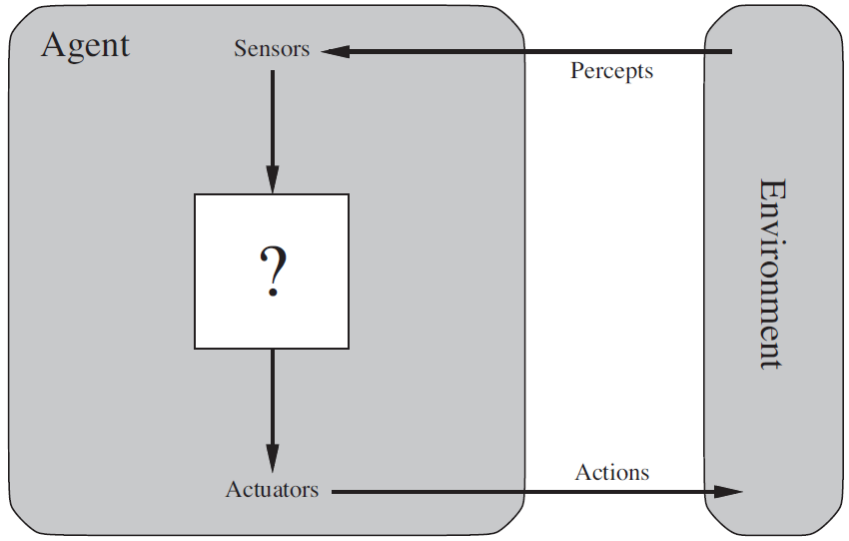
- Rational ≠ omniscient nor clairvoyant
  - Percepts may not supply all relevant information
  - Action outcomes may not be as expected
  - Hence, rational ≠ successful (e.g., lottery)

- The definition of rationality implies that a rational agent should be able to:
  - explore the world to gather more information – use precepts to do new things (**information gathering, exploration**)
  - Ideally: equip an agent with some prior knowledge and the ability to learn
  - autonomous rather than follow what it's designed to do (i.e., that it should learn and adapt with it's experience).

2/3. Task environments

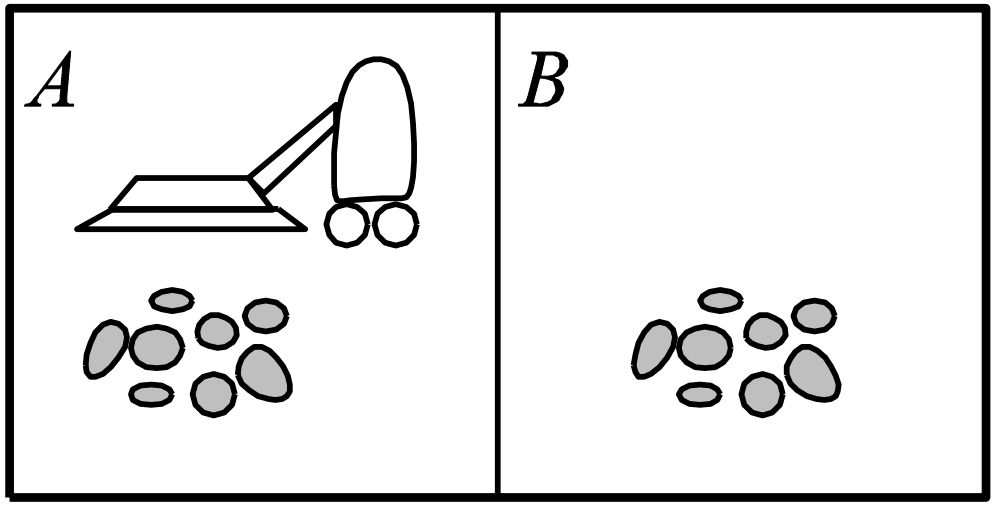
- "Task environment", or simply "task"
  - is the problem we would like to solve
  - involves both the agent and the environment
  - What we design agents as solutions for!

- Task environments are best described by PEAS description
  - P: Performance measure
  - E: Environment
  - A: Actuators (= effectors)
  - S: Sensors



PEAS for vacuum world

- Performance measure
  - Maximize ratio of dirt to energy/time usage
- Environment
  - Two squares A&B, with/without dirt
- Actuators
  - Wheels, vacuum
- Sensors
  - Location, dirt sensor



PEAS for Automated Taxi

- Performance measure
  - Safe, fast, legal, economical, and comfortable trip
- Environment
  - Road, traffic, pedestrian, passenger
- Actuators
  - Steering, accelerator, brake, signal, horn, display
- Sensors
  - Cameras, rangefinder, speedometer, GPS, odometer, accelerometer, etc.

PEAS: Medical Diagnosis System

- Performance measure
  - Healthy patient, minimize costs, lawsuits
- Environment
  - Patient, hospital, staff
- Actuators
  - Screen display for questions, tests, diagnoses, treatments, referrals
- Sensors
  - Keyboard (entry of symptoms, findings, patient's answers)

Take out a Piece of Paper

Tulane University

- Get a partner or two around you…
  - Put everyone's name on the paper!

- Determine PEAS for the following "known" intelligent agents
  - Airplane autopilot
  - Recommender System (e.g. Amazon's)
  - Siri
  - Watson (Jeopardy)
  - Deepblue (Chess)

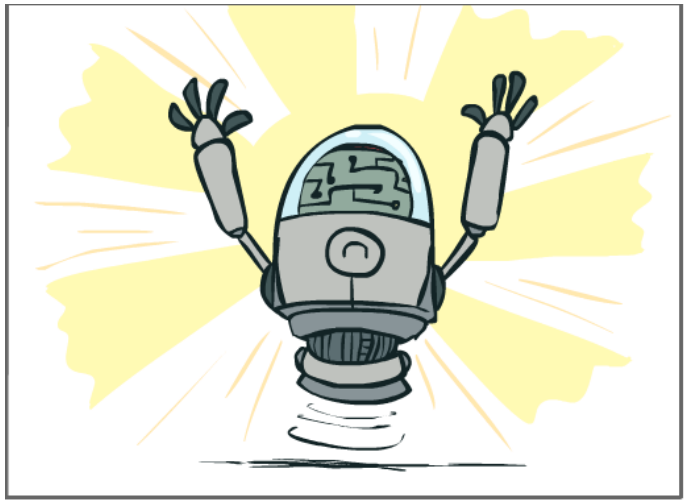
Properties of task environments

Tulane University

- PEAS describes the basics components of a task environment

- To design the agent, the properties of the task environment should be further analyzed:

  - Fully, partially, or not observable
  - Deterministic or stochastic
  - Single or multi-agent
  - Episodic or sequential
  - Static or dynamic
  - Discrete or continuous
  - Known or unknown



Properties of task environments

Tulane University

- Observability - Can the agent get all **relevant** information from the environment using its sensors? (Not all information)

- If an agent's sensors give it access to the complete state of the environment at each point in time.
  - No need for an internal state of the world

- Fully observable
  - Chess: Positions of all the playing pieces
  - AI enemy in first-person shooter video game
- Partially observable
  - Poker: Only the cards in your hand
  - "You" in first-person shooter game
- Unobservable
  - No sensor at all

Properties of task environments

Tulane University

- Deterministic vs Stochastic: Is the next state of the world **determined completely** by the current state of the world and the **action** executed by the agent?

- **Deterministic:** only determined by current state and action of the agent.
- **Stochastic:** There is uncertainty on the next state and it is expressed with probabilities.

- Vacuum world
  - Deterministic: suction and motor wheels always work
  - Stochastic: unreliable suction and/or motor wheels
- Chess: deterministic, except for the actions of other agents
- Automated taxi: stochastic

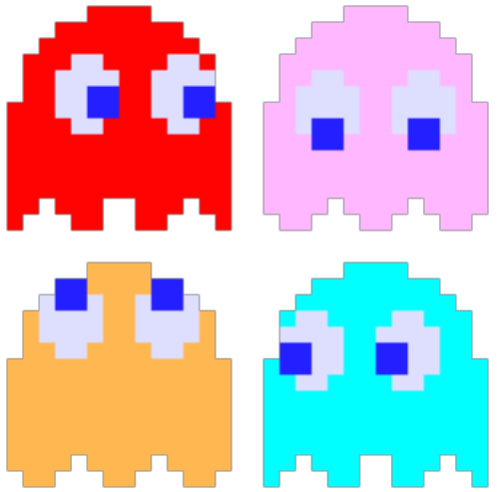
- **Non-Deterministic:** there is uncertainty but no probabilities (e.g., all possible goals).
- **Uncertain:** not fully observable and non-deterministic.

Properties of task environments

- Single vs Multi-agent: Is the agent operating by itself in an environment or are there others?

- Competitive: two agents are competing against each other.
- Cooperative: two agents are working together for a common goal.
- Mixed: ??

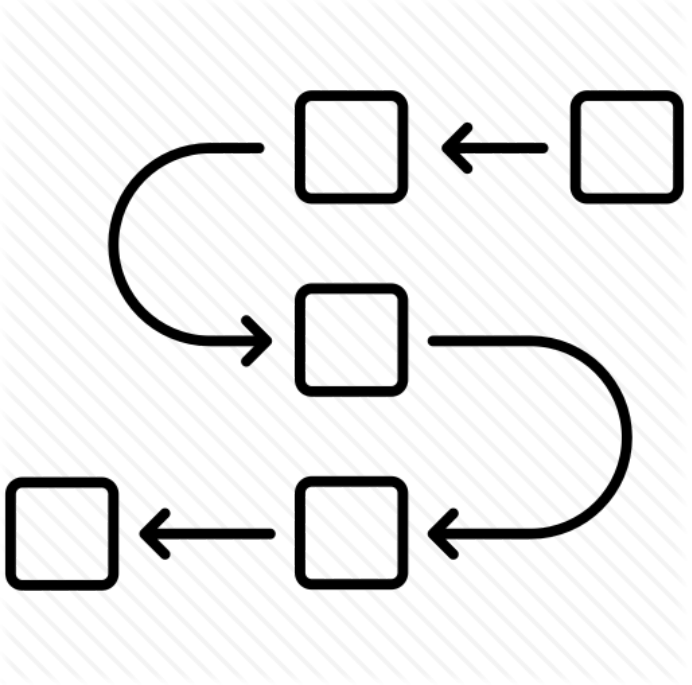
- We sometimes call an environment **strategic** if everything about the environment is **deterministic except for the actions of other agents.**
  - Example??



Properties of task environments

- Episodic V. Sequential

- The agent's experience is divided into **atomic "episodes"** (each episode consists of the agent **perceiving** and then performing a single **action**), and the choice of action in each episode depends only on the episode itself.
- In sequential environments a current decision may affect future decisions



- Episodic: current action affects only the current episode.
  - E.g., a single recommendation.

- Sequential: future episodes as well
  - e.g., chess, or taxi driving, or pacman

Properties of Task Environments

Tulane University

- Static vs. Dynamic

- What happens to the environment during deliberating?

- **Static:** time halts while agent makes a decision – no need to keep looking at the world or worry about time.
  - Turn-based games (e.g., Monopoly)

- **Dynamic:** world changes on while agent decides
  - Pac-man
- Semi-dynamic: world stands still, but decision time has an effect on performance score
  - Tournament-level chess

Properties of Task Environments

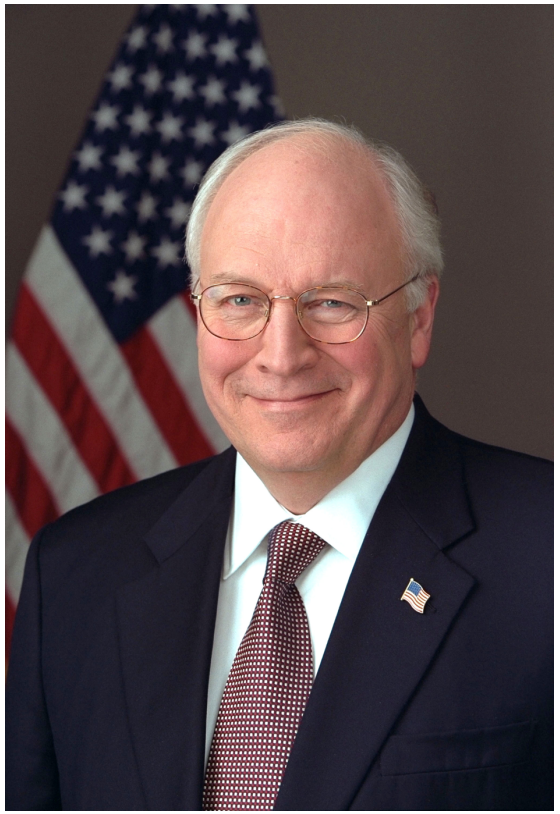
Tulane University

- Discrete vs. Continuous: How is time handled in the environment?
- Discrete: A **finite number** of distinct, clearly defined states, percepts and actions.
- Continuous: Sweep through a range of values smoothly over time.
  - Is time handled in chunks or continuously?
  - Is a state … ?
  - Is an action … ?
  - Examples: chess vs automated taxi

Properties of task environments

Tulane University

- Known vs. Unknown.
  - What is the knowledge of the agent or the designer of the agent with respect to the rules governing the environment.



- Known: agent knows how its action affects the state of the world (e.g., chess)
  - Deterministic: known outcome
  - Stochastic: probability distribution

- Unknown: environment is unknown to agent (more in Part 3, reinforcement learning)
  - We show it a game but don't tell it the rules.

- An environment can also be
  - known and partially observable (for example, a card game)
  - unknown and fully observable (a videogame)

Examples of task environments

Task Environment	Observable	Deterministic	Episodic	Static	Discrete	Agents
Robot soccer						
Internet book-shopping						
Autonomous Mars rover						

Examples

Task Environment	Observable	Deterministic	Episodic	Static	Discrete	Agents
Robot soccer	Partially	Stochastic	Sequential	Dynamic	Continuous	Multi
Internet book-shopping						
Autonomous Mars rover						

- Internet book-shopping: buying a used AI textbook online
  - P: quality/price
  - E: used book sales websites (Ebay, Amazon, Craigslist, etc)
  - A: clicking links or buttons
  - S: web-browser

Examples

Tulane University

Task Environment	Observable	Deterministic	Episodic	Static	Discrete	Agents
Robot soccer	Partially	Stochastic	Sequential	Dynamic	Continuous	Multi
Internet book-shopping	Partially	Deterministic*	Sequential	Static*	Discrete	Single
Autonomous Mars rover						

- Internet book-shopping
  - Sequential? Episodic?
  - Static? Dynamic?
  - Single-agent? Multi-agent?
- No absolutely correct answer; depends on how to formulate the task

Examples

Tulane University

Task Environment	Observable	Deterministic	Episodic	Static	Discrete	Agents
Robot soccer	Partially	Stochastic	Sequential	Dynamic	Continuous	Multi
Internet book-shopping	Partially	Deterministic*	Sequential	Static*	Discrete	Single
Autonomous Mars rover	Partially	Stochastic	Sequential	Dynamic	Continuous	Single

Examples

Tulane University

Task Environment	Observable	Deterministic	Episodic	Static	Discrete	Agents
Robot soccer	Partially	Stochastic	Sequential	Dynamic	Continuous	Multi
Internet book-shopping	Partially	Deterministic*	Sequential	Static*	Discrete	Single
Autonomous Mars rover	Partially	Stochastic	Sequential	Dynamic	Continuous	Single

- Real world problems:
  - partially observable, stochastic, sequential, dynamic, continuous, multi-agent
- Properties of the task environment largely determines the agent design

3/3. Agent architecture

Tulane University

- Five different designs of agents
  - Reflex agent
  - Model-based reflex agent
  - Goal-based agent
  - Utility-based agent
  - Learning agent

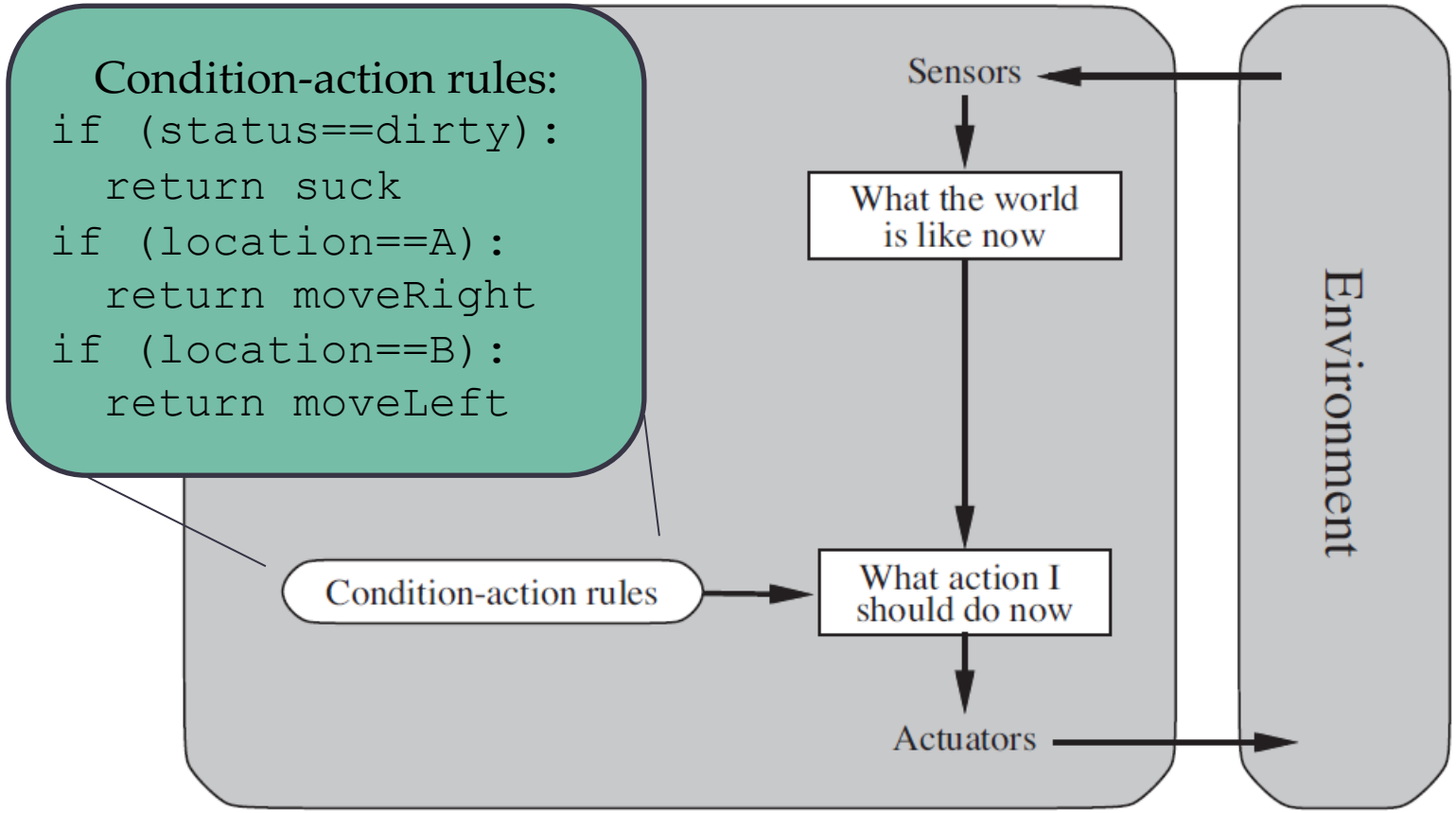
Reflex agent

Tulane University

- Simplest kind of agents
- Choose action based on the current percept, ignoring the percept history
- Useful as a sub-agent of a bigger agent
  - Biological examples:
  - Closing eyes when something approaches eyes
  - Move the hand away when it touches something hot

- They work if the choice of the action **depends only on the current percept**
- That is, if the environment is fully observable
- Example: What happens to the vacuum deprived of the location sensor?

Reflex agent



function SIMPLE-REFLEX-AGENT(percept) **returns** an action
persistent: *rules*, a set of condition-action pairs

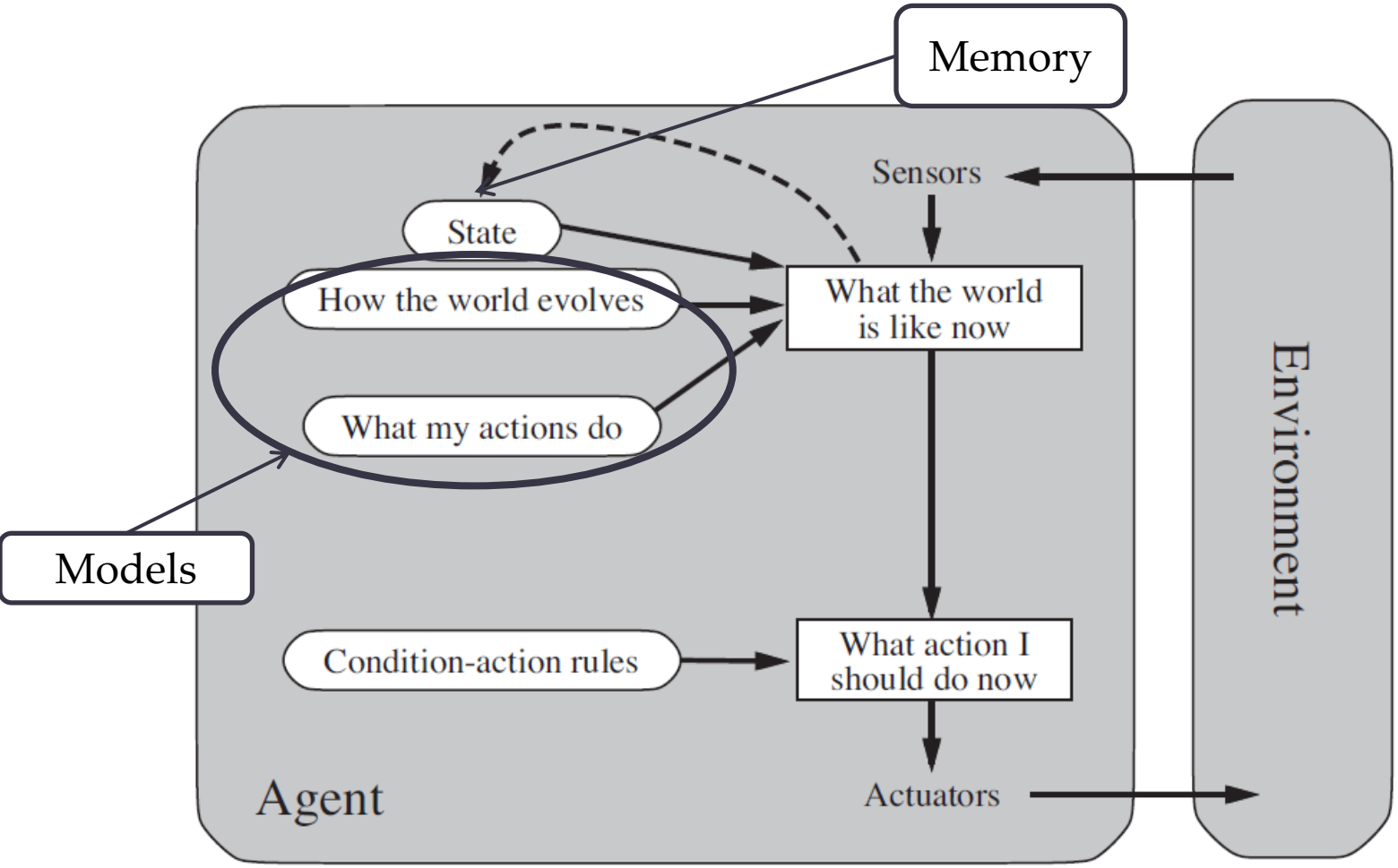
state ← INTERPRET-INPUT(percept)
rule ← RULE-MATCH(state,rules)
action ← *rule*.ACTION
return *action*

Model-based reflex agent

Tulane University

- In partially observable task environments
  - Need to keep track of unseen portions of the world
- Memory
  - Internal state: depends on the sequence of percepts
  - guess of unobserved portions
- Models
  - How the world evolves
  - What my actions do
- Examples
  - Navigating through a maze
  - Passing a car in a single-lane road
- Current state of a partially observable environment is usually not exact
- May be described succinctly

Model-based reflex agent



function MODEL-BASED-REFLEX-AGENT (*percept*)
returns an action

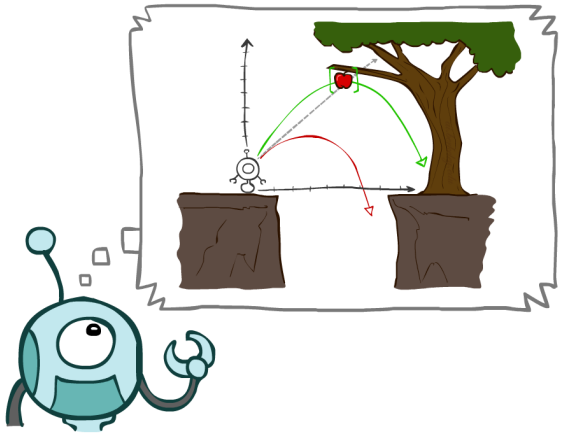
persistent: *state*, the agent's current conception of the world state
model, a description of how the next state depends on current state and action
rules, set of condition-action pairs
action, the most recent action, initially none

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

Goal-based agent

Tulane University

- Sometimes knowing the state of the environment isn't enough.
  - Information on goal is necessary.

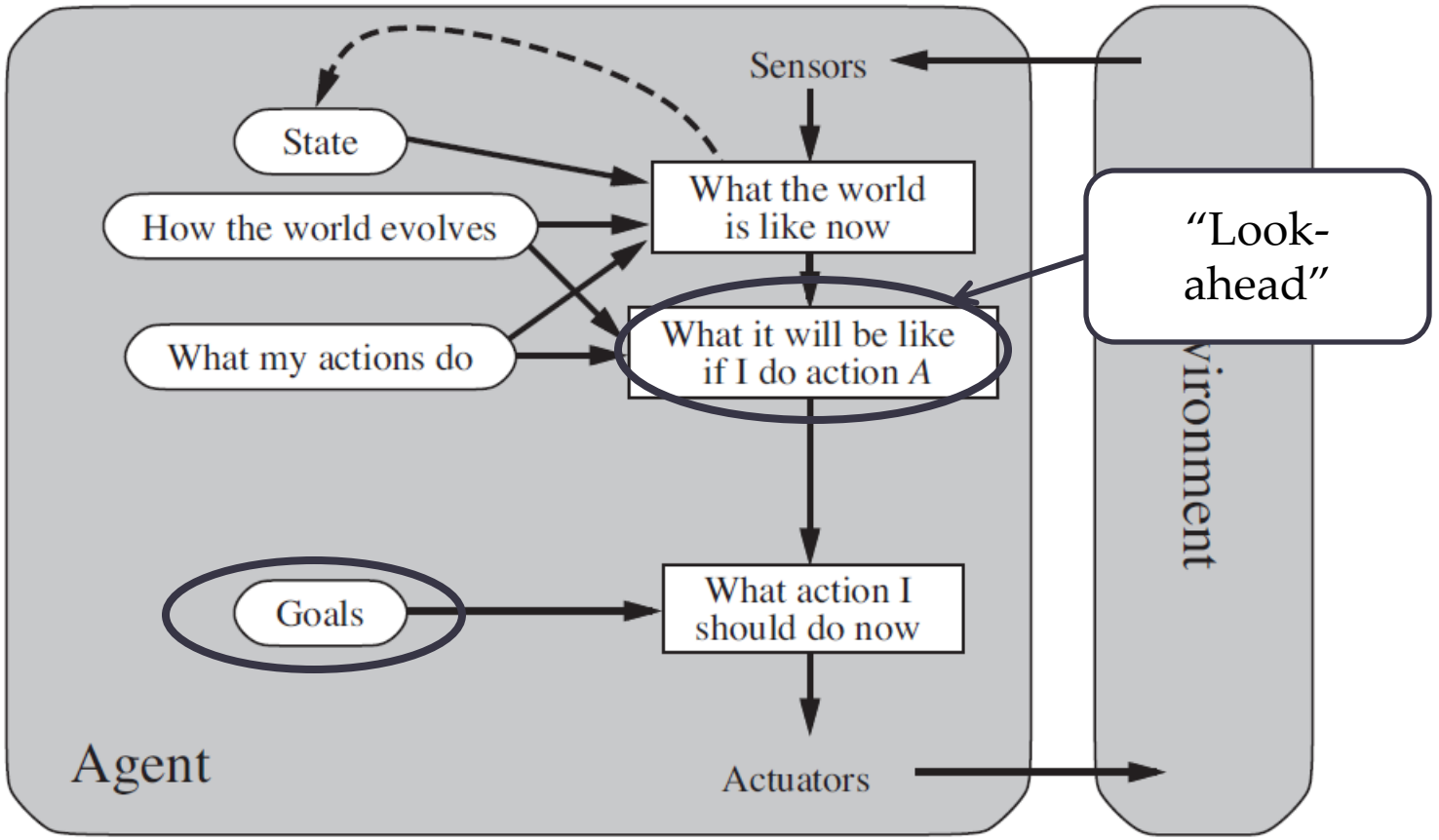


- Best current action is chosen by higher-level goals, not just the current percept
  - Goal: "arriving at destination x" for automated taxi
- Find a sequence of actions to achieve the goal
  - Actions: "left turn, drive 0.5 mile, right turn, drive 1.2 miles,…"

- Main design for Part 1 of the course – search and planning: find an action sequence that achieves a goal.

- Goal driven agents are flexible as they don't require rewriting with the condition rules or goal or space change.

Goal-based agent



Utility-based agent

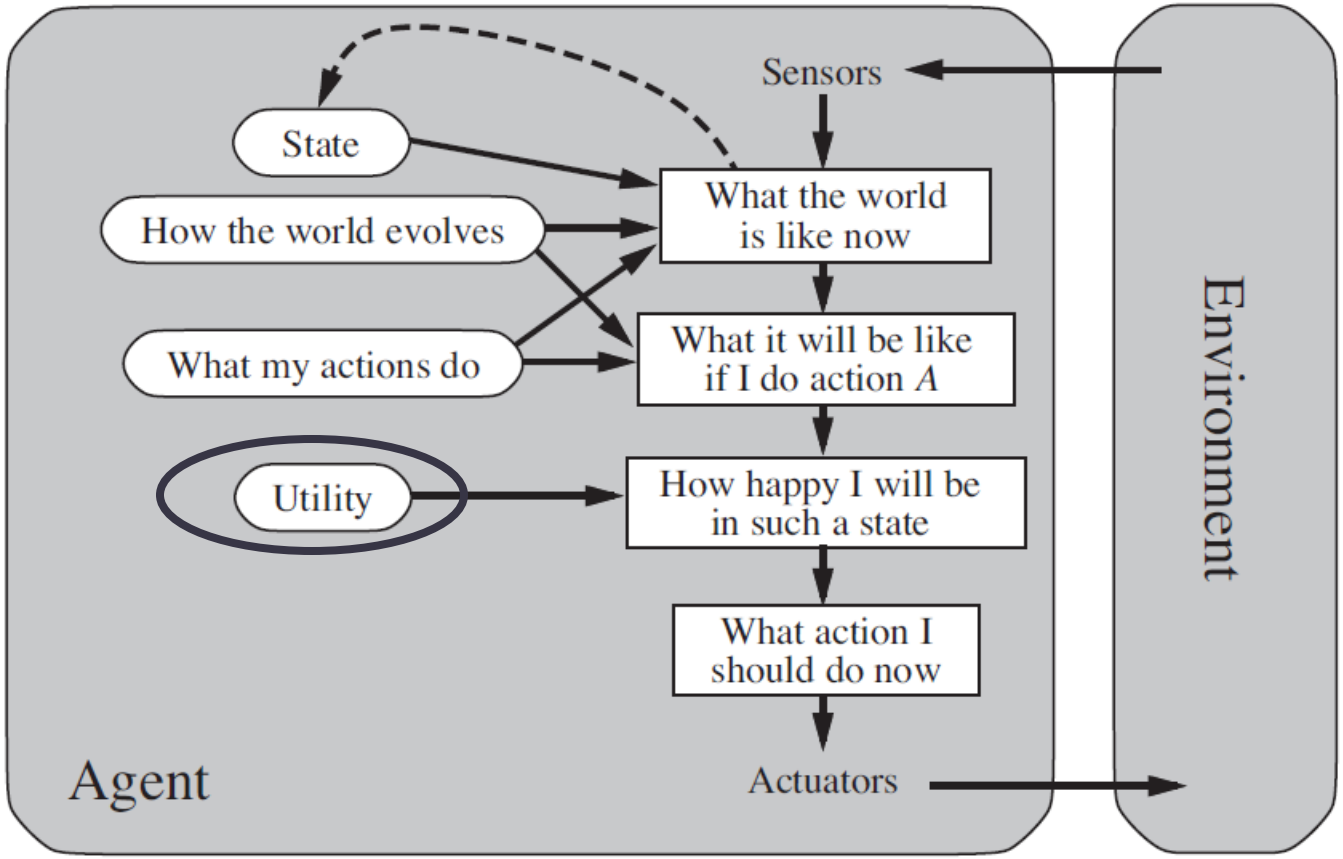
Tulane University

- Goal-based agent: successful or not
  - But not all ways to achieve goals are equally good!
  - How satisfied? --> utility function: represents an performance measure of the level

- Multiple goals: taxi
  - Goal 1: arrive at destination
  - Goal 2: minimize time/fuel consumption
  - Goal 3: passenger safety/comfort
- Utility can resolve multiple conflicting goals
  - Maximize the utility as the weight sum of multiple goals
- Furthermore, if the environment task is stochastic, maximize the *expected* utility

- More in depth in Part 3 Utility theory

Utility-based agent

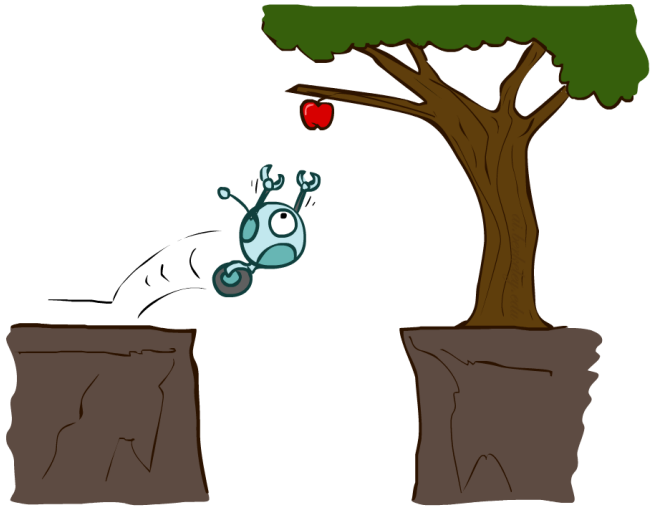


Learning agent

- From the early stages of AI, it became clear that programming agents able to learn and to teach them might often be a better option that programming intelligence directly

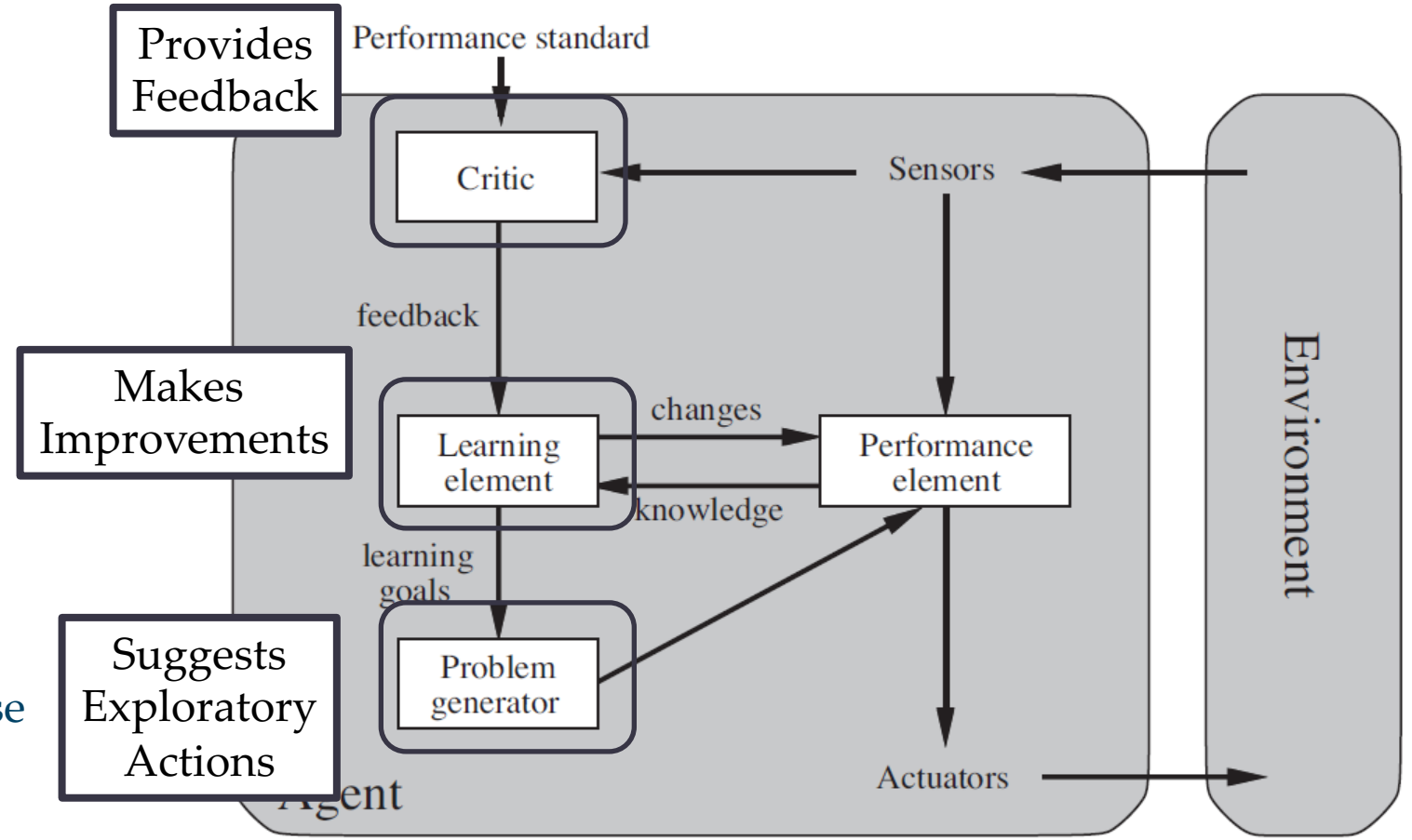
- If we have intermediate feedbacks (in addition to final performance measure) on the agent's performance, use it to refine the agent function
- Components of a learning agent
  - Performance element (=non-learning part)
  - Critic
  - Learning element: adjust behavior
  - Problem generator: exploration

- More in depth in Part 3 on RL!



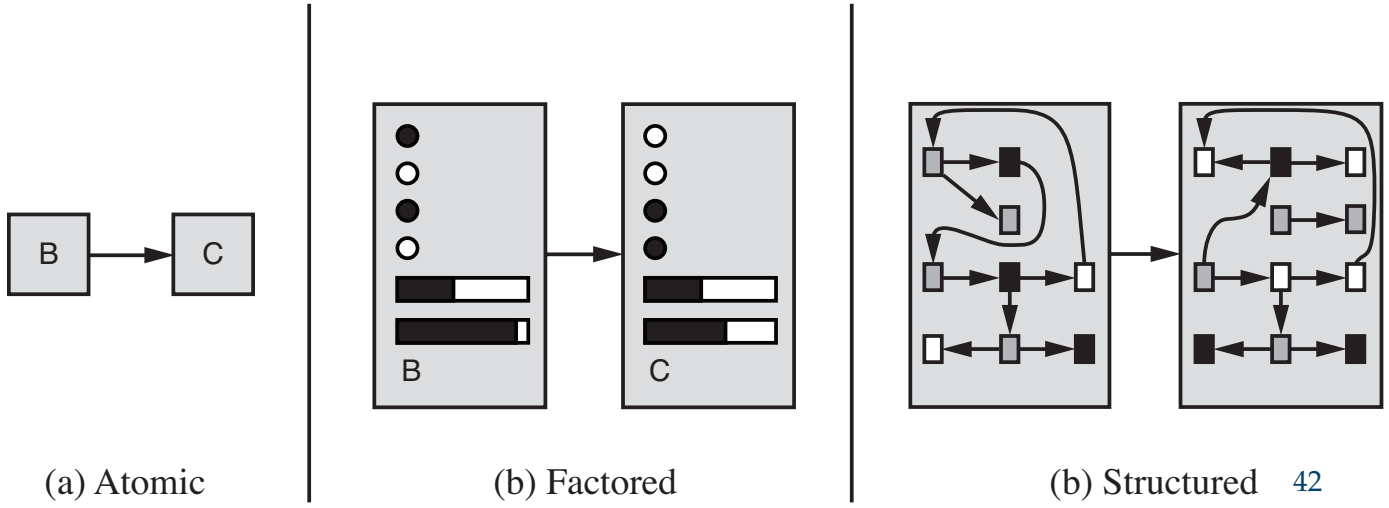
Learning agent

- Components of a learning agent
  - **Learning element**: responsible for making improvements
    - can change any component of the agent
  - **Performance element**: the actual agent
  - **Critic**: assesses how well the agent is performing w.r.t. a fixed performance standard (e.g. checkmate is good)
    - May be adapted to take utility into account and to enable the learning of utility information
  - **Problem generator**: generates actions that yield an information gain (otherwise the ones that are best given the current knowledge would always be chosen)
    - May suggest suboptimal actions which lead to find better actions in the long run



A Note About Problem Representations

- **Atomic representation:** each state of the world is indivisible and has no internal structure (possibly only from the agent view).
  - E.g., directions only by the name of a location.
  - mainly used by search agents and can be powerful (also MPSs and HMMs).
- **Factored representation:** each state has a fixed set of variables or attributes each with a value.
  - E.g., same GPS location but gas/no gas.
  - Used for many things classification, CSPs, propositional logic, planning.
- **Structured representation:** each state is has variables and attributes and we know relations between these objects.
  - E.g., cow in road in front of truck
  - mainly used by planning agents but also first order logic and relationships!
- Tradeoff: expressivity v. space!



(a) Atomic (b) Factored (b) Structured