COS30019 - Introduction to Artificial Intelligence
Tutorial Problems Week 2

Focus (objectives):

- Learn to characterize agent's environment (which is essential for making the right choice of agent's design)
- Learn different agent's types and the differences between them

Task 1:

Let us examine the rationality of various vacuum-cleaner agent functions.

A. Show that the simple vacuum-cleaner agent function described in the lecture (under the given assumptions) is indeed rational.

What is a rational agent? Such agent selects a course of action that maximizes its expected performance measure, given the prior environment knowledge and perceived information to date.

Function: described in the lecture

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
…	…

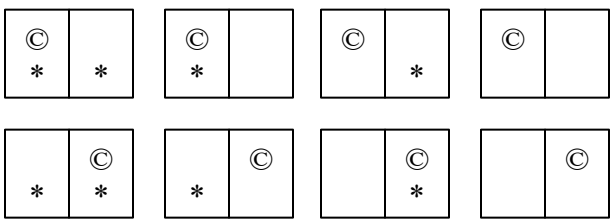
Considered assumptions:

- Performance measure: 1 point for each clean square over the lifetime of 1000 steps
- Geography (squares A and B only) is known
- Dirt distribution, initial position: not known
- Agent can perceive its location and whether that location contains dirt (using its sensors)
- Available actions: Left, Right, Suck, NoOp
- Clean squares stay clean
- Sucking operation cleans the current square
- Left and Right don't take agent outside the environment

We need to prove that under these assumptions **this agent's expected performance is at least as high as the expected performance of any other agent.**

We consider all the possible **initial configuration**:

- initial location is either A or B
- four different cases of initial dirt content in the squares (i.e., dirt in *both squares*, *only in A*, *only in B*, *in no square*)



We can demonstrate that, regardless of the initial configuration, the given agent function (which is equivalent to the behaviour of the following agent program:
procedure REFLEX-VACUUM-AGENT ([*location, status*]) return an *action*
if *status* == *Dirty* then return *Suck*
else if *location* == *A* then return *Right*
else if *location* == *B* then return *Left*)
ensures that the performance measure it achieves is at least as high as any other behaviour as it keeps as many square clean as early as possible.

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?

Our agent above would keep oscillating between the two squares A and B, resulting in low performance measure due to losing point for each movement. It's problem is that it does not remember that it was in a square (A or B) and kept that square clean.
Agent's internal state can "remember" the past states (trace the past observations).
Given the new performance function, a rational agent function will also have to minimize the number of movements. We need an internal state in order to know which squares were cleaned.

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?

Yes, the agent should learn at least two main things: the geography of the environment to build a map, and should learn the time that it takes for particular squares to get dirty (learn distribution of dirt in the environment). It will allow building the most optimal routes for cleaning.

Task 2: Develop a PEAS (performance measure, environment, actuators, sensors) description of the task environment for:

a) Robot soccer player
b) Internet book-shopping agent

	a) Robot soccer player	**b) Internet book-shopping agent**
Performance	Maximize (Goals For – Goals Against) Obey soccer rules No yellow/red card	Search accuracy (right books) Minimize cost Minimize search time
Environment	Ball, Soccer field Other players, Referees Weather	Internet Bookshops DB **Optional**: Other agents (traders)/This depends on your assumptions!
Actuators (for action)	Robotic legs, arms, head (Or wheels) Communication devices (e.g. transmitter, to speak, shout)	APIs for placing orders (e.g. for updating bookstore DBs) and for sending information to the users (e.g., via voice or displayed text)
Sensors (for perception)	Cameras Speedometer LiDAR/RADAR sensors Communication devices (e.g. to receiver, to hear)	APIs for searching (the internet, DBs), and for receiving user's instructions

Actuators – set of devices that the agent can use to perform actions
Sensors – allow the agent to collect percept sequence that will be used for deliberating on the next action

Task 3: For each agent type above, characterize the environment and select a suitable agent design.

Fully vs. Partially observable: can the sensors detect all aspects that are relevant (depends on the performance measure) to the choice of action at each point in time
Deterministic vs. Stochastic: is the next environment state completely determined by the current state and the action executed by the agent?
Episodic vs. Sequential: is the agent's experience divided into atomic episodes? does the next episode depend on the actions taken in previous episodes?
Static vs. Dynamic: can the environment change while the agent is choosing an action? Semi-dynamic – the agent's performance changes even when the environment is the same
Discrete vs. Continuous: can be applied to the state of the environment, to the way time is handled, and to the percepts and actions of the agent.
Single-agent vs. Multi-agent: are there any other agents besides our agent?

	a) Robot soccer player	**b) Internet book-shopping agent**
Observable	Partially	Fully Observable Under some strict assumptions about the agent's APIs, we can assume that the agent has very powerful sensors to allow the environment to be fully observable to the agent – but this is very unlikely. Thus, a more general setting is Partially observable!
Deterministic/Stochastic	Stochastic (there is uncertainty)	Deterministic (again, under some strong assumptions). In a more general setting, we may need to assume that there are non-zero probability that an action may not correctly executed (the network fails, bugs in the program APIs, actions taken by other agents, etc.) and thus the env is stochastic!
Episodic/Sequential	Sequential (depends on the outcome of the previous action)	Sequential (the agent will need to perform a sequence of actions to achieve its goal e.g., search for books, compare options, place orders, etc.)
Static/Dynamic	Dynamic	Static (again, under a very strong assumption that while the agent is calculating its next action, the whole world stops and wait for its decision)/ Dynamic (the book may be sold while the agent searches, need to consider other options)/Semi-dynamic(the agent has to minimise the search time)
Discrete/Continuous	Continuous (space and time)	Discrete (it is moving through discrete items)
Single/Multi-agent	Multi-agent	Single-agent(again, strong assumption)/More generally, it is a multi-agent env (there are other agents who may compete to get the same books, seller agents who may want to maximise their profits and change prices as they see a book become popular)

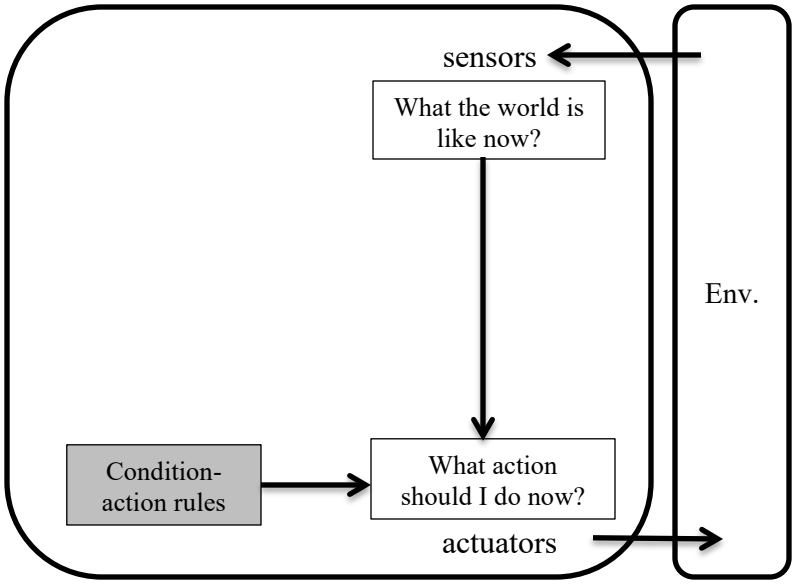
Suitable agent design:

What agent types do we know?

- **Reflex-agents** (respond immediately to percepts (**simple-reflex** – suitable for fully observable environments; **state-based reflex**– more efficient for partially observable environments)
- **Goal-based agents** act in order to achieve their goal(s)
- **Utility-based agents** maximize their own utility function
  Agent **with learning capability**: Agent can improve its performance over time as it learns from its experiences.

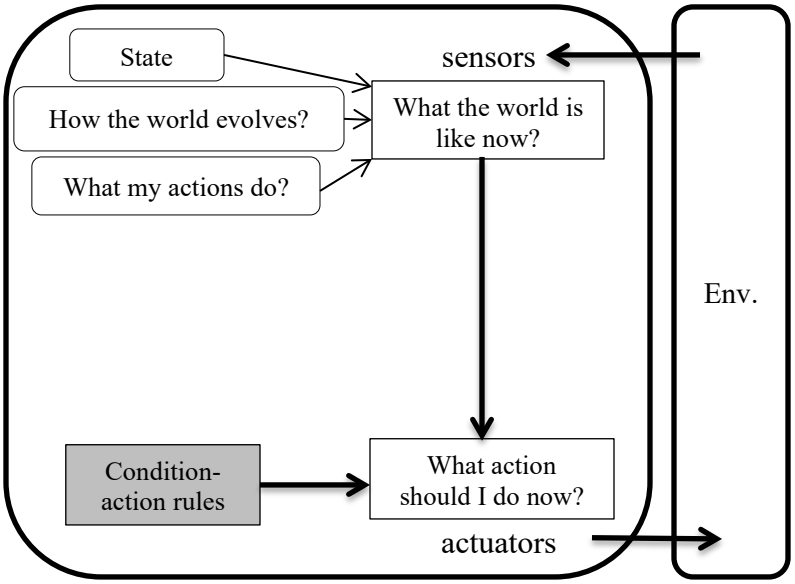
Utility-based agent with learning capability will be a good agent type for robot soccer player because the level of satisfaction increases with increased number of goals and agents will need to learn to adapt to the specific environment and opponents.

Utility-based agents will suit (simple) internet book shopping agent. The measured utility can be the saved amount of money compared to the initial price that the buyer wanted to pay for a particular book, or the amount of time saved while searching for the book compared to the manual search time. Similarly, the bigger the amount of saved money, the higher the satisfaction is; the less time spent for searching, the higher the satisfaction. For the complex internet book shopping agent, it requires learning capability so that it can deals with changes in the environment and other agents.



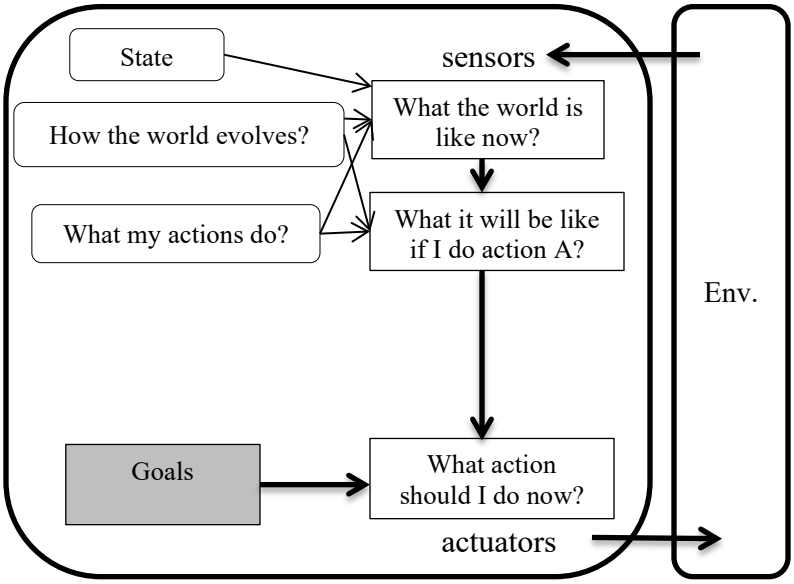
Simple reflex

Single percept
Check condition-action (if dirty, then suck)
For fully observable environments (otherwise input loops may occur)



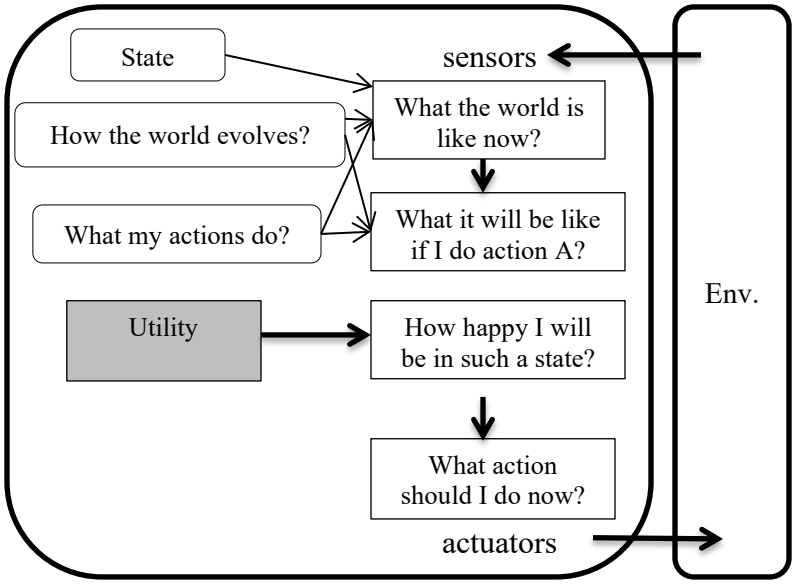
Reflex & State

For partially observable environments
Builds its model of world



Goal-based

Goal indicates which situations are desirable
Used in search and planning research (e.g. chess)
Main difference: future is taken into account



Utility-based

Utility function maps a sequence of states into a real number

Improves on goals:
Selecting between conflicting goals (e.g. attend a lecture or continue to sleep)
Select appropriately between several goals based on likelihood of success

Task 4: Referring to the utility-based agents described in the lecture, both the performance measure and the utility function measure how well an agent is doing. Explain the difference between the two.

Utility function – is an internal measurement of the agent for itself.
Performance measure – an external observer perspective.

Ideally, the agent's utility function (which defines the agent's behaviour) is the same as (or, fully aligned with) the performance measure. However, this is not necessarily always the case, especially in agents that can learn and self-improve and evolve itself (cf. Strong AI). Such an agent may develop a mind of its own and thus will have a utility function that may not be aligned with the intended performance measure. Eg: A robot designed to assist and protect its human owner develops its own mind and decide that its human owner is a "bad" person who needs to be destroyed!