Rational agent

Remember vacuum problem?

Agent program:
  if [Dirty], return [Suck]
  if at [room A], return [move right]
  if at [room B], return [move left]
Agent models

Can also classify agents into four categories:

1. Simple reflex
2. Model-based reflex
3. Goal based
4. Utility based

Top is typically simpler and harder to adapt to similar problems, while bottom is more general representations.
Agent models

A simple reflex agents acts only on the most recent part of the percept and not the whole history.

Our vacuum agent is of this type, as it only looks at the current state and not any previous.

These can be generalized as: “if state = _____ then do action _____” (often can fail or loop infinitely)
Agent models

A model-based reflex agent needs to have a representation of the environment in memory (called internal state).

This internal state is updated with each observation and then dictates actions.

The degree that the environment is modeled is up to the agent/designer (a single bit vs. a full representation).
Agent models

This internal state should be from the agent's perspective, not a global perspective (as same global state might have different actions)

Consider these pictures of a maze:
Which way to go?  Pic 1  Pic 2
Agent models

The global perspective is the same, but the agents could have been doing different things

Pic 1

Pic 2

(Goals are not global information)
Agent models

We also saw this when we were talking about agent functions (also from agent’s perspective, not global)

<table>
<thead>
<tr>
<th>Percept sequence</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>[A, Clean]</td>
<td>Right</td>
</tr>
<tr>
<td>[A, Dirty]</td>
<td>Suck</td>
</tr>
<tr>
<td>[B, Clean]</td>
<td>Left</td>
</tr>
<tr>
<td>[B, Dirty]</td>
<td>Suck</td>
</tr>
<tr>
<td>[A, Clean], [A, Clean]</td>
<td>Right</td>
</tr>
<tr>
<td>[A, Clean], [A, Dirty]</td>
<td>Suck</td>
</tr>
<tr>
<td>:</td>
<td>:</td>
</tr>
</tbody>
</table>
Agent models

For the vacuum agent if the dirt does not reappear, then we do not want to keep moving.

The simple reflex agent program cannot do this, so we would have to have some memory (or model).

This could be as simple as a flag indicating whether or not we have checked the other state.
Agent models

The goal based agent is more general than the model-based agent.

In addition to the environment model, it has a goal indicating a desired configuration.

Model-based reflex only use the internal state to find the immediate next action, while goal based plan multiple actions in advance.
Agent models

A utility based agent maps the sequence of states (or actions) to a real value.

Goals can describe general terms as “success” or “failure”, but there is no degree of success.

In the maze example, a goal based agent can find the exit. But a utility based agent can find the shortest path to the exit.
Agent models

What is the agent model of particles?

Think of a way to improve the agent and describe what model it is now.
Environment classification

Environments can be further classified on the following characteristics:

1. Fully vs. partially observable
2. Single vs. multi-agent
3. Deterministic vs. stochastic
4. Episodic vs. sequential
5. Static vs. dynamic
6. Discrete vs. continuous
7. Known vs. unknown
Environment classification

In a **fully observable** environment, agents can see every part.

Agents can only see part of the environment if it is **partially observable**.
Environment classification

If your agent is the only one, the environment is a single agent environment

More than one is a multi-agent environment (possibly cooperative or competitive)
Environment classification

If your state+action has a known effect in the environment, it is **deterministic**

If actions have a distribution (probability) of possible effects, it is **stochastic**
Environment classification

An episodic environment is where the previous action does not effect the next observation (i.e. can be broken into independent events)

If there is the next action depends on the previous, the environment is sequential

sequential episodic

episodic sequential
Environment classification

If the environment only changes when you make an action, it is static.

A dynamic environment can change while your agent is thinking or observing.
Environment classification

Discrete = separate/distinct (events)
Continuous = fluid transition (between events)

This classification can apply to: agent's percept and actions, environment's time and states
Environment classification

**Known** = agent's actions have known effects on the environment

**Unknown** = the actions have an initially unknown effect on the environment (can learn)

**do not know how to stop**
1. Fully vs. partially observable = how much can you see?
2. Single vs. multi-agent
   = do you need to worry about others interacting?
3. Deterministic vs. stochastic
   = do you know (exactly) the outcomes of actions?
4. Episodic vs. sequential
   = do your past choices effect the future?
5. Static vs. dynamic = do you have time to think?
6. Discrete vs. continuous
   = are you restricted on where you can be?
7. Known vs. unknown
   = do you know the rules of the game?
Environment classification

Some of these classifications are associated with the state, while others with the actions

**State:**
1. Fully vs. partially observable
2. Single vs. multi-agent
3. Deterministic vs. stochastic
4. Episodic vs. sequential
5. Static vs. dynamic
6. Discrete vs. continuous
7. Known vs. unknown

**Actions:**

---

---
Environment classification

Pick a game/hobby/sport/pastime/whatever and describe both the PEAS and whether the environment/agent is:
1. Fully vs. partially observable
2. Single vs. multi-agent
3. Deterministic vs. stochastic
4. Episodic vs. sequential
5. Static vs. dynamic
6. Discrete vs. continuous
7. Known vs. unknown
Environment classification

<table>
<thead>
<tr>
<th>Agent type</th>
<th>Performance</th>
<th>Environment</th>
<th>Actuators</th>
<th>Sensors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Particles</td>
<td>time alive</td>
<td>boarder, red balls</td>
<td>move mouse</td>
<td>screenshot</td>
</tr>
</tbody>
</table>

Fully observable, single agent, deterministic, sequential (halfway episodic), dynamic, continuous (time, state, action, and percept), known (to me!)
An atomic state has no sub-parts and acts as a simple unique identifier.

An example is an elevator:
Elevator = agent (actions = up/down)
Floor = state

In this example, when someone requests the elevator on floor 7, the only information the agent has is what floor it currently is on.
Another example of an atomic representation is simple path finding: If we start at Koffman, how would you get to Keller's CS office?

Go E. -> Cross N @ Ford & Amundson -> Walk to E. KHKH -> K. Stairs -> CS office

The words above hold no special meaning other than differentiating from each other
State structure

A **factored** state has a fixed number of variables/attributes associated with it.

You can then reason on how these associated values change between states to solve problems.

Can always “un-factor” and enumerate all possibilities to go back to atomic states, but might be too exponential or lose efficiency.
Structured states simply describe objects and their relationship to others.

Suppose we have 3 blocks: A, B and C. We could describe: A on top of B, C next to B.

A factored representation would have to enumerate all possible configurations of A, B and C to be as representative.
State structure

We will start using structured approaches when we deal with logic:

Summer implies Warm
Warm implies T-Shirt

The current state might be:
!Summer (¬Summer)
but the states have intrinsic relations between each other (not just actions)