 Agents: definition and formal architectures

CS7032: AI for IET

Lecturer: Liliana Mamani Sanchez
mamanisl@cs.tcd.ie

September 29, 2015

Outline

• What is an agent?
  • Properties one would like to model
  • Mathematical notation
• A generic model
• A purely reactive model
• A state-transition model
• Modelling concrete architectures
• References: (?, ch 1 and ch 2 up to section2.6), (?, ch 2)

Abstract architectures will (hopefully) help us describe a taxonomy (as it were) of agent-based systems, and the way designers and modellers of multi-agent systems see these artificial creatures. Although one might feel tempted to see them as design tools, abstract architectures are really descriptive devices. Arguably their main purpose is to provide a framework in which a broad variety of systems which one way or another have been identified as “agent systems” can be classified in terms of certain formal properties. In order to be able to describe such a variety of systems, the abstract architecture has to be as general as possible (i.e. it should not commit the modeller to a particular way of, say, describing how an agent chooses an action from its action repertoire). One could argue that such level of generality limits the usefulness of an abstract architecture to the creation of taxonomies. This might explain why this way of looking at multi-agent systems hasn’t been widely adopted by agent developers. In any case, it has the merit of attempting to provide precise (if contentious) definitions of the main concepts involved thereby moving the discussion into a more formal domain.
What is an “agent”?

- “A person who acts on behalf of another” (Oxford American Dictionaries)

- Action, action delegation: state agents, travel agents, Agent Smith

- Action, simply (autonomy)

- “Artificial agents”: understanding machines by ascribing human qualities to them (?) (humans do that all the time)

- Interface agents: that irritating paper-clip

An example: Interface agents

- Embodied life-like characters REA: real estate agent (?), speech recognition and synthesis, multi-modal interaction (gestures, gaze, life-size graphics etc)

- Not all UI designers like the idea, though (?)

AI’s Nobel prize winner

- Scientific and philosophical background:
  - Natural Sciences
    - Conformance to “natural laws”
    - An air of necessity about them
  - Sciences of the artificial (?):
    - Prescription vs. description
    - A science of the contingent
    - A science of “design”
From a prescriptive (design) perspective...

- Agent properties:
  - Autonomy: acting independently (of user/human intervention)
  - Situatedness: sensing and modifying the environment
  - Flexibility:
    * re-activity to changes
    * pro-activity in bringing about environmental conditions under which the agent’s goals can be achieved, and
    * sociability: communicating with other agents, collaborating, and sometimes competing

In contrast with the notion of abstract architecture we develop below, these properties of multi-agent systems are useful from the point of view of conceptualising individual systems (or prompting the formation of a similar conceptual model by the user, in the case of interface agents). Abstract architectures, as descriptive devices, will be mostly interested in the property of “situatedness”. The starting point is the assumption that an agent is essentially defined by its actions, and that an agent’s actions might be reactions to and bring about changes in the environment. Obviously, there is more to agents and environments than this. Agents typically exhibit goal-oriented behaviour, assess future actions in terms of the measured performance of past actions, etc. The architectures described below will have little to say about these more complex notions.

However, before presenting the abstract architecture, let’s take a look at a taxonomy that does try to place goals in context:

**PAGE Descriptions**

- P.A.G.E.: Percepts, Actions, Goals and Environment

<table>
<thead>
<tr>
<th>Agent Type</th>
<th>Percepts</th>
<th>Actions</th>
<th>Goals</th>
<th>Environment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Medical diagnosis</td>
<td>Symptoms, findings, patient’s</td>
<td>Questions, tests, treatments</td>
<td>Healthy patient, minimize costs</td>
<td>Patient, hospital</td>
</tr>
<tr>
<td>system</td>
<td>answers</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Satellite image</td>
<td>Pixels of varying intensity,</td>
<td>Print a categorization of frame</td>
<td>Correct categorization</td>
<td>Images from orbiting satellite</td>
</tr>
<tr>
<td>analysis system</td>
<td>color</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Part-picking robot</td>
<td>Pixels of varying intensity</td>
<td>Pick up parts and sort into bins</td>
<td>Place parts in correct bins</td>
<td>Conveyor belt with parts</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Interactive English</td>
<td>Typed words</td>
<td>Print exercises, suggestions,</td>
<td>Maximize student’s score on test</td>
<td>Set of students</td>
</tr>
<tr>
<td>tutor</td>
<td></td>
<td>corrections</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

As in (Russell and Norvig, 1995)

...alternatively, P.E.A.S. descriptions
### Agent Type

<table>
<thead>
<tr>
<th>Agent Type</th>
<th>Performance measure</th>
<th>Environment</th>
<th>Actuators</th>
<th>Sensors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Medical diagnosis system</td>
<td>Healthy patient, minimize costs</td>
<td>Patient, hospital</td>
<td>Questions, tests, treatments</td>
<td>Symptoms, findings, patient’s answers</td>
</tr>
<tr>
<td>Satellite image analysis system</td>
<td>Correct categorization</td>
<td>Satellite link</td>
<td>Print a categorization of acne</td>
<td>Pixels of varying intensity, color</td>
</tr>
<tr>
<td>Part-picking robot</td>
<td>% parts in correct bins</td>
<td>Conveyor belt with parts</td>
<td>Pick up parts and sort into bins</td>
<td>Pixels of varying intensity, color</td>
</tr>
<tr>
<td>Interactive English tutor</td>
<td>Student’s score on test</td>
<td>Set of students; testing agency</td>
<td>Print exercises, suggestions, corrections</td>
<td>Keyboard input</td>
</tr>
<tr>
<td>Plant controller</td>
<td>partially stochastic</td>
<td>partially stochastic</td>
<td>partially stochastic dynamic continuous</td>
<td>single</td>
</tr>
</tbody>
</table>

**From (7)**

### Environment properties

- Fully vs. partially observable: whether agent’s can obtain complete and accurate information about the environment
- Deterministic vs. stochastic: whether the next state of the environment is fully determined by the current state and action performed by the agent
- Episodic vs. sequential: whether agent’s next action depends only on the current state of the environment (episodic), or on assessment of past environment states (sequential)
- Static vs. dynamic: whether the environment changes independently of the agent’s actions
- Discrete vs. continuous: whether the possible actions and percepts on an environment are finite (discrete environment) or not (continuous environment)
- Single vs. multiple agents

### Types of environments

<table>
<thead>
<tr>
<th>Environment</th>
<th>Observable</th>
<th>Deterministic</th>
<th>Episodic</th>
<th>Static</th>
<th>Discrete</th>
<th>Agents</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chess w/ clock</td>
<td>partly</td>
<td>strategic</td>
<td>sequential</td>
<td>static</td>
<td>discrete</td>
<td>multi</td>
</tr>
<tr>
<td>Backgammon</td>
<td>partly</td>
<td>stochastic</td>
<td>sequential</td>
<td>static</td>
<td>discrete</td>
<td>multi</td>
</tr>
<tr>
<td>Car driving</td>
<td>partly</td>
<td>stochastic</td>
<td>sequential</td>
<td>static</td>
<td>discrete</td>
<td>multi</td>
</tr>
<tr>
<td>Medical diagnosis</td>
<td>partly</td>
<td>stochastic</td>
<td>sequential</td>
<td>dynamic</td>
<td>continuous</td>
<td>single</td>
</tr>
<tr>
<td>Robot arm</td>
<td>partly</td>
<td>stochastic</td>
<td>episodic</td>
<td>static</td>
<td>dynamic continuous</td>
<td>single</td>
</tr>
<tr>
<td>Plant controller</td>
<td>partly</td>
<td>stochastic</td>
<td>sequential</td>
<td>dynamic</td>
<td>continuous</td>
<td>single</td>
</tr>
</tbody>
</table>

### From (7)

Chess with clock is semi-static (semi-dynamic) because even though the environment changes (time passes, players are penalised if they fail to play by a certain deadline) independently of the agent’s actions, it does so in a regular and predictable way. The environment a taxi driver operates in, on the other hand, may change in unpredictable ways, being therefore a dynamic environment. Medical diagnosis typically depends on patient history, so its environment is sequential. In contrast, a robot which examines various parts which pass on a conveyor belt in front of the robot’s camera, one at the time, and moves defective parts off the conveyor belt does not have to consider its past decisions in order to decide whether the part it is currently examining is defective. The robot is therefore said to operate in an episodic environment.

### Abstract agent architectures

- Why?
- What we would like to describe:
  - agent
environment

- their interactions

**Notation**

- A few tools from discrete maths and logic:
  - $A, S, \ldots$ : sets
  - $\wp(S)$ : the powerset of $S$
  - $S^*$ : all sequences of elements (i.e. ordered subsets) of $S$
  - $\Rightarrow, \land, \lor, \neg$ : material implication, conjunction, disjunction and negation
  - Quantifiers: $\forall$ and $\exists$

Refreshing your memory: The powerset of a set $S$ is a set containing all of $S$’s subsets. So, if $S = \{s_0, s_1\}$, its powerset $\wp(S) = \{\emptyset, \{s_0\}, \{s_1\}, \{s_0, s_1\}\}$. Sequence sets $S^*$ are similar to power sets except that the order of their elements matters. So, if $S = \{s_0, s_1\}$, we have $S^* = \{<>, s_0, s_1, s_0, s_1\}$. The logical connectives behave in the familiar way. $X \land Y$ is true iff both $X$ and $Y$ are true; $X \lor Y$ is true if $X$ is true, $Y$ is true, or both; $X \Rightarrow Y$ is false if $X$ is true and $Y$ is false, and true otherwise. Since these connectives are used here simply to illustrate particular examples of agent architectures, we will deliberately leave their semantics underspecified. Feel free to interpret them as they are interpreted in, say, PROLOG. One could also add existential and universal quantifiers as needed.

**Defining an architecture**

- A standard architecture is a 4-tuple: $\text{Arch}_s = \langle S, A, \text{action}, \text{env} \rangle$
  where
    - $S = \{s_1, s_2, \ldots, s_n\}$ is a finite set describing all (possible) environment states, and
    - $A = \{a_1, a_2, \ldots, a_n\}$ is the set of all actions an agent is capable of performing
    - $\text{action}$ is a function describing the agents behaviour, and
    - $\text{env}$ is a function describing the “behaviour” of the environment

An example: consider the “blocks world” below:
Assume the robotic arm is capable of performing a set of actions $A = \{\text{move}(a,b), \text{move}(c, \text{floor}), \ldots \}$. Environment states could be represented by an $S = \{s_1, s_2, \ldots, s_n\}$, where we could have

$$s_1 = \{\text{left}(a,b), \text{on}(\text{floor},a), \text{on}(a,c), \ldots \}$$  \hspace{1cm} (1)

and (after a \text{move}(c, \text{floor})):

$$s_2 = \{\text{left}(a,b), \text{on}(\text{floor},a), \text{on}(\text{floor},c), \ldots \}$$  \hspace{1cm} (2)

One could also, for instance, encode general properties such as

$$\forall x, y, z. \text{left}(x, y) \land \text{left}(y, z) \Rightarrow \text{left}(x, z)$$  \hspace{1cm} (3)

etc, to remain constant in all environment states.

“Agenthood”

- An agent’s behaviour will be characterised by the following function:

$$\text{action} : S^* \rightarrow A$$

- Does having $S^*$ as the domain of $\text{action}$ make the function most naturally suited to modelling episodic or sequential environments? Why?

- Requirement captured: the current action may depend on the interaction history (i.e. the sequence of environment states)

Actions consider an essentially sequential environment, since their arguments are sequences. In our blocks world example, one could have $\text{action}((s_1, s_2)) = \text{move}(a,b)$, assuming that the purpose of moving block $c$ off block $a$ was to be able to place $a$ on $b$. Note, however, that the formalism does not explicitly represent purpose, planning, or any such notion.

Environment dynamics

- Changes in the environments will be characterised by the following function:

$$\text{env} : S \times A \rightarrow \wp(S)$$

- Intuition: $\text{env}(s_j, a_k) = S'$ performing an action $a_k$ on an environment whose state is $s_j$ results in a number of scenarios ($S'$)

- In the general case, what type of environment does $\text{env}$ model? Answer: $\text{env}$ models a non-deterministic environment.

- If $|S'| = 1$, then the environment is deterministic.

In the example above, if nothing else is said, action $\text{move}(c, \text{floor})$ would affect the environment as follows:

$$\text{env}(s_1, \text{move}(c, \text{floor})) = \{\{\text{left}(c,a), \ldots \}, \{\text{left}(a,c), \ldots \}\}$$  \hspace{1cm} (4)

If the relative positions of the blocks were irrelevant, $\text{env}(s_1, \text{move}(c, \text{floor}))$ would be a singleton.
Interaction history

- The agent-environment interaction will be characterized as follows:
  \[ h : s_0 \xrightarrow{a_0} s_1 \xrightarrow{a_1} ... \xrightarrow{a_{u-1}} s_u \xrightarrow{a_u} ... \]

- \( h \) is a possible history of the agent in the environment iff:
  \[ \forall u \in \mathbb{N}, a_u = \text{action}(s_0, ..., s_u) \] (5)
  and
  \[ \forall u > 0 \in \mathbb{N}, s_u \in \text{env}(s_{u-1}, a_{u-1}) \] (6)

  Condition (5) guarantees that all actions in a history apply to environment states in that history.
  Condition (6) guarantees that all environment states in a history (except the initial state) “result” from actions performed on environment states.

Characteristic behaviour

- Characteristic behaviour is defined as a set of interaction histories
  \[ \text{hist} = \{ h_0, h_1, ..., h_n \} \]
  where
  - each \( h_i \) is a possible history for the agent in the environment

Invariant properties

- We say that \( \phi \) is an invariant property of an agent architecture iff
  - For all histories \( h \in \text{hist} \) and states \( s \in S \phi \) is a property of \( s \) (written \( s \models \phi \))

  - We will leave the relation \( \models \) underspecified for the time being.
    - In architectures where agents have minimal symbolic reasoning abilities (e.g. some concrete reactive architectures), \( \models \) could be translated simply as set membership, that is \( s \models \phi \Leftrightarrow \phi \in s \), in architectures where reasoning isn’t performed

Behavioural equivalence

- Equivalence of behaviours is defined as follows (in an abstract architecture)
  - An agent \( ag1 \) is regarded as equivalent to agent \( ag2 \) with respect to environment \( env_i \) iff \( \text{hist}(ag1, env_i) = \text{hist}(ag2, env_i) \)

  - When the condition above holds for all environments \( env_i \), then we simply say that \( ag1 \) and \( ag2 \) have equivalent behaviour

7
Modelling reactive agents

- Reactive architectures
  - production rules
  - a scheme for defining priorities in the application of rules (e.g., subsumption (?))

- no reasoning (theorem proving or planning) involved

Abstract reactive agents

- *Purely reactive agents* can be modelled by assuming

  \[ \text{action} : S \rightarrow A \]

- Everything else remains as in the general (standard abstract architecture) case:

  \[ \text{Arch}_r = \langle S, A, \text{action}_r, \text{env} \rangle \]

- Abstract reactive agents operate essentially on an episodic view of environments (i.e., they are memoryless agents). See slide 6

Example: a thermostat.

\[
\begin{align*}
\text{temperature}(\text{cold}) & \Rightarrow \text{do}(\text{heater}(\text{on})) \\
\neg \text{temperature}(\text{cold}) & \Rightarrow \text{do}(\text{heater}(\text{off})) \\
S & = \{\{\text{temperature}(\text{cold})\}, \{\neg \text{temperature}(\text{cold})\}\} \\
A & = \{\text{heater}(\text{on}), \text{heater}(\text{off})\}
\end{align*}
\]

Purely reactive vs. standard agents

- Proposition 1 (exercise):

  Purely reactive agents form a proper subclass of standard agents. That is, for any given environment description \( S \), and action repertoire \( A \):

  (i) every purely reactive agent is behaviourally equivalent to a standard agent, and

  (ii) the reverse does not hold.
Modelling Perception

- Refining the agent’s decision function

![Diagram of Agent, Perception, Environment, See, Action]

- Types and sources of perception

Percepts and actions

- Facts perceived by an agent will be represented as set
  \[ P = \{ p_0, p_1, ..., p_n \} \]

- The decision function becomes
  \[ \text{action} : P^* \rightarrow A \]

- which is then linked to environment states via the perception function
  \[ \text{see} : S \rightarrow P \]

Properties of perception

- If \( \text{see}(s_i) = \text{see}(s_j) \), we say that \( s_i \) and \( s_j \) are indistinguishable, even if \( s_i \neq s_j \).

- Define \( \equiv \), an equivalence relation over \( S \) by saying \( s \equiv s' \) iff \( \text{see}(s) = \text{see}(s') \)

- If \( |\equiv| = |S| \), then we say that the agent is perceptually omniscient

- On the other hand, if \( |\equiv| = 1 \), then the agents perceptual ability is nil

In the following example, four environment states are (rightfully, assuming that a butterfly doesn’t cause changes in the weather) as two:

\( x = \text{weather(cold)} \)

\( y = \text{moving.wings(butterfly)} \)

\( S = \{ \{x, \neg y\}, \{\neg x, y\}, \{x, y\}, \{\neg x, \neg y\} \} \)

\( \text{see}(\{x, \neg y\}) = \text{see}((x, y)) \)

\( \text{see}(\{\neg x, y\}) = \text{see}(\{\neg x, \neg y\}) \)

The equivalence relation \( \equiv \) is then \( \{ \{x, y\}, \{x, \neg y\}, \{\neg x, y\}, \{\neg x, \neg y\} \} \)

For perceptually omniscient agents, \( \text{see}(s_i) = s_i \), for all \( s_i \in S \). In an agent with no perceptual ability, \( \text{see}(s_i) = \text{see}(s_j) \) for all \( s_i, s_j \in S \).
Refining the interaction history

- Representation used so far: history as a sequence of environment states

- The next step: represent history (in the agent architecture) as environment changes as perceived by the agent

State-based decision function

- The state-based architecture will be represented as

  \[ \text{Arch} = \langle S, A, I, \text{action}, \text{env}, \text{see}, \text{next} \rangle \]

  where

  - \( I \) is the set of all internal states of an agent,
  - \( \text{see} : S \rightarrow P \),
  - \( \text{action} : I \rightarrow A \), and
  - \( \text{next} : I \times P \rightarrow I \)

Properties of State-based architectures

- Proposition 2:

  State-based architectures are equivalent to standard architectures with respect to the behaviours they are able to represent.

Exercise

- Prove Proposition 2.

Further information

- (?): overview; this presentation mostly followed the material found there;
- (?): agent = architecture + program
- (?): foundations of a theory of rational agency
References