Intelligent Agents

Based on “An Introduction to MultiAgent Systems” and slides by Michael Wooldridge

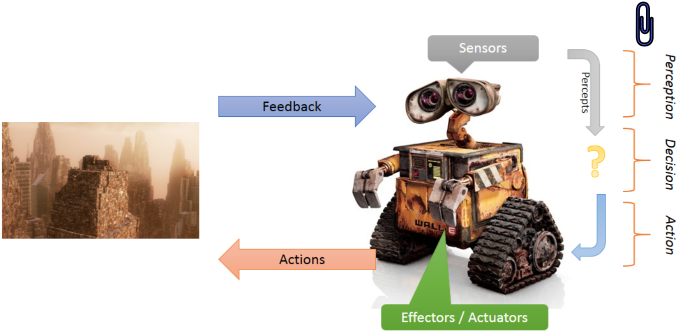
Definition of an Agent

An agent is a **computer system** capable of **autonomous action** in some **environment**, in order to achieve its **delegated goals**.

An agent is in a close-coupled continual interaction with its environment

sense $\rightarrow$decide $\rightarrow$act $\rightarrow$sense $\rightarrow$decide $\rightarrow$...

Agent and Environment



Agent and Environment

- In most domains an agent will not have a *complete control* over its environment (at best partial control)
- The same action performed twice in (apparently) identical circumstances may have entirely different effects, and in particular it may *fail*
- Agents must be prepared for the possibility of *failure*

Properties of Environments

Accessible vs. inaccessible

- accessible environment is one where the agent can obtain complete, accurate, and up-to-date information about the environment's state
- moderately complex environments are inaccessible
- the more accessible environment, the simpler it is to build agents operating in it

Properties of Environments

Deterministic vs. non-deterministic

- a deterministic environment is one where any action has a single guaranteed effect (no uncertainty about the state resulting from an action)
- the physical world can be regarded as non-deterministic
- non-deterministic environments pose a greater challenge for agent designers

Properties of Environments

Static vs. dynamic

- a static environment remains unchanged except the performance of actions by the agent
- a dynamic environment has other processes operating on it, and it changes in ways beyond the agent's control
- the physical world is a highly dynamic environment

Properties of Environments

Discrete vs. continuous

- an environment is discrete if there are a fixed, finite number of actions and percepts in it
- Chess game is a discrete environment, while car driving is a continuous one

Varieties of Autonomy

Autonomy as a spectrum:

- Humans: freedom with respect to beliefs, goals and actions
- Software services: no freedom, they do what they are told

Adjustable autonomy

The control of decision making is transferred from the agent to a person when certain conditions are met

- when the agent beliefs that the human will make a “better” decision
- when there is a degree of uncertainty about the environment
- when the decision may cause harm
- when the agent lacks the capability to make the decision itself

Decisions and Actions

- An agent has a set of available actions ($\rightarrow$ability to modify the environment)
- Actions have associated preconditions, which define the possible situations in which they can be applied
- The key problem for an agent is to decide which of its action to perform in order to best satisfy its (design) objectives

Agent Architectures

Software architectures for decision-making systems that are embedded in an environment

Examples of (Simple) Agents

- Control systems
  - Any control system can be viewed as an agent
  - A simple example: a thermostat (goal: to maintain certain temperature, actions: heat on or off)
  - More complex systems with richer decision structures
- Software daemons
  - Any software, which monitors a software environment and performs actions to modify it
  - An example: anti-virus (or anti-spam) software

Intelligent Agents

Intelligent agent (should) exhibit 3 types of behavior:

1. **Reactivity** – ability to perceive their environment, and to respond (quickly) to changes in order to satisfy their objectives
2. **Pro-activeness** – ability to take the initiative (goal-directed behavior) in order to satisfy their objectives
3. **Social ability** – ability to interact with other agents (and possibly humans) in order to satisfy their objectives

Reactivity and Pro-activeness

It is important (and difficult) to achieve proper balance between reactivity and pro-activeness

- Agents should achieve their goals systematically, e.g., by constructing and following complex plans
- However, they should not follow blindly these plans, if it is clear they will not work or when the goal is no longer valid
- In such cases, agents should be able to react to the new situation
- However, agents should not be continually reacting in order not to loose the overall goal

Social Ability

- The real world is a multi-agent environment and some goals can be only achieved by interacting with others
- Social ability is the ability to interact with other agents via
  - **cooperation** – working together as a team to achieve a shared goal
  - **coordination** – managing the inter-dependencies between activities
  - **negotiation** – the ability to reach agreements on matters of common interest
- In the simplest case, it means the ability to communicate!

Agents and Objects

Are agents just objects by another name?

- Objects encapsulate some state
- Objects communicate via message passing
- Objects have methods corresponding to operations that may be performed on this state

Agents and Objects

- Agents embody a stronger notion of autonomy than objects – in particular they decide whether or not to perform an action on request from another agent

Objects do it for free; agents do it because they want to (or do it for “money”)...

- Agents are capable of flexible (reactive, proactive, social) behavior (standard object model does not deal with it)
- A multi-agent system is inherently multi-threaded – each agent is assumed to have at least one thread of control

Agents as Intentional Systems

- When explaining human activity, we use statements like the following:

Jane took her umbrella because she *believed* it was raining and she *wanted* to stay dry.

- Such statements make use of a folk psychology, where human behavior is predicted and explained by attributing attitudes such as believing, wanting, hoping, fearing...

Intentional System

An entity, whose behavior can be predicted by attributing belief, desires and rational acumen

Agents as Intentional Systems

- As software systems become more complex, we need more powerful abstractions and metaphors to explain their operations (low level explanations become impractical)
- Most important developments in computing are based on new abstractions
  - procedural abstraction
  - abstract data types
  - objects

Agents (as intentional systems) represent a further and more powerful abstraction to describe, explain and predict the behavior of complex systems

Abstract Architectures for Agents

- Assume the environment may be in any of a finite set E of discrete states

$$E = \left\{ e, e^{'}, \ldots \right\}$$

- Agents are assumed to have a set of possible actions, which transform the state of the environment

$$Ac = \left\{ \alpha, \alpha^{'}, \ldots \right\}$$

- A **run** r of an agent in an environment is a sequence of interleaved environment states and actions

$$r : e_0 \xrightarrow{\alpha_0} e_1 \xrightarrow{\alpha_1} e_2 \xrightarrow{\alpha_2} e_3 \xrightarrow{\alpha_3} \cdots \xrightarrow{\alpha_{u-1}} e_u$$

Runs

Let

- $\mathscr{R}$ be the set of all such possible finite sequences (over E and Ac)
- $\mathscr{R}^{Ac}$ be the subset of these runs that end with an action
- $\mathscr{R}^{E}$ be the subset of these runs that end with an environment state

Environments

- A **state transformer** function represents behavior of the environment

$$\tau : \mathscr{R}^{Ac} \rightarrow \mathbf{2}^{E}$$

- Environments are
  - history dependent – the current state is somewhat determined by earlier actions
  - non-deterministic – there is uncertainty about the results of performing an action
- If $\tau(r) = \varnothing$, there are no possible successor states to r, so we say the run has **ended** ("game over")

Environments

An environment *Env* is a triple

$$Env = \langle E, e_0, \tau \rangle$$

where E is a set of environment states, $e_0 \in E$ is initial state, and τ is state transformer function

Agents

- Agent is a function that maps runs to actions

$$Ag : \mathscr{R}^E \to Ac$$

- Agent makes a decision about what (deterministic) action to perform based on the history of the system that it has witnessed to date
- Let $\mathscr{AG}$ be the set of all agents

Systems

- A system is a pair containing an agent and an environment
- Any system is associated with a set of possible runs – we denote the set of runs of agent Ag in environment Env by $\mathscr{R}(Ag, Env)$
- Let $\mathscr{R}(Ag, Env)$ contain only runs that have ended (terminated or finite runs only!)

Formally, a sequence

$$(e_0, \alpha_0, e_1, \alpha_1, e_2 \cdots)$$

represents a run of an agent Ag in environment $Env = \langle E, e_{0,} \tau \rangle$if

1. e_0 is the initial state of Env
2. $\alpha_0 = Ag\,(e_0)$
3. for $u > 0$, $e_u \in \tau\,(e_0, \alpha_0, \cdots, \alpha_{u-1})$ and $\alpha_u = Ag\,(e_{0,} \alpha_0, \cdots, e_u)$

Purely Reactive Agents

Such agents decide what to do without reference to their history – they base their decision making entirely on the present, with no reference to the past

- Formally, a purely reactive agent can be presented as

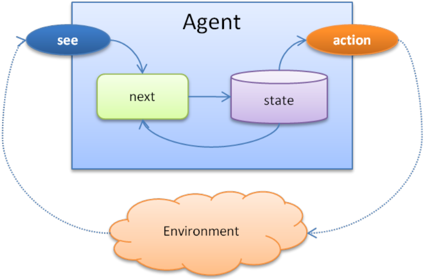
$$Ag : E \to Ac$$

- A thermostat is a purely reactive agent

$$action(e) = \begin{cases} off & \text{if} e = \text{temperature OK} \\ on & \text{otherwise} \end{cases}$$

Agents with State

- In order to construct real agents, the abstract model needs to refined by breaking it into subsystems – data and control structures
- These subsystems constitute an **agent architecture**



Agents with State

- Such agents have some data structure (state) which is used to record information about the environment state and history. Let I be the set of all internal states of the agent
- The perception function *see* represents the agent's ability to obtain information from its environment and transform it into a perceptual input. It is defined as

$$see : E \rightarrow Per$$

Agents with State

- The action-selection function is defined as

$$action : I \rightarrow Ac$$

- An additional function *next* is introduced to map an internal state and percept into a new internal state

$$next : I \times Per \rightarrow I$$

Agent Control Loop

1. Agent starts in some initial internal state i_0
2. Observe environment state and generate a percept $see(e)$
3. Update the internal state via the *next* function – set the state to $i_1 = next(i_0, see(e))$
4. Select action via the *action* function – select $action(i_1)$
5. Perform the selected action
6. Go to 2

Tasks for Agents

- We build agents in order to carry out **tasks** for us
- The tasks must be **specified** by us...
- But we want to tell agents what do to **without** telling them how to do it

Utility Functions over States

- One approach: utilities are associated with individual states – the task of the agent is to achieve states that maximize utility
- A task specification is (simply) a function

$$u : E \to \mathbb{R}$$

- This function associates a real number with every environment state

Utility Functions over States

- Defining the overall utility of an agent in some particular environment
  - pessimistic approach – the utility of the worst state within a run
  - optimistic approach – the utility of the best state within a run
  - sum of utilities of states from a run
  - average utility of states from a run
- Disadvantage: difficult to specify a long term view when assigning utilities to individual (isolated) states

Utilities over Runs

- Another possibility: assign a utility not to individual states, but to runs

$$u : \mathscr{R} \to \mathbb{R}$$

- This approach focuses on a long term view

Utility in Tileworld

- Simulated two dimensional grid environment on which there are agents, tiles, obstacles, and holes
- An agent can move in four directions, up, down, left, or right, and if it is located next to a tile, it can push it
- Holes have to be filled up with tiles by the agent. An agent scores points by filling holes with tiles – the aim is to fill as many holes as possible
- Tileworld changes with the random appearance and disappearance of holes

Utility in Tileworld

- The performance of an agent is measured by running Tileworld for a predetermined number of time steps and counting holes filled by the agent
- The performance of an agent on some particular run is defined as

$$u(r) = \frac{\text{number of holes filled in } r}{\text{number of holes that appeared in } r}$$

Changes and Opportunities in Tileworld