CHAPTER 2: INTELLIGENT AGENTS

An Introduction to Multiagent Systems

http://www.csc.liv.ac.uk/~mjw/pubs/imas/
Chapter 2
An Introduction to Multiagent Systems

What is an Agent?

The main point about agents is they are autonomous.

Thus: capable independent action.

We think of an agent as being in a close-coupled,
continual interaction with its environment:
autonomous action in some environment, in
order to achieve its delegated goals.

• Themainpointaboutagentsistheyareautonomous:
capableindependentaction.

• Themainpointaboutagentsistheyareautonomous:

Thus: capable independent action.

Sense – decide – act – sense – decide – decide •

...
Agent and Environment
Simple (Uninteresting) Agents

• Thermostat
  – delegated goal is maintain room temperature
  – actions are heat on/off

• UNIX biff program
  – delegated goal is monitor for incoming email and flag it
  – actions are GUI actions.

They are trivial because the decision making they do is trivial.
Intelligent Agents

We typically think of an intelligent agent as exhibiting 3 types of behaviour:

- reactive;
- pro-active;
- social.

Chapter 2
An Introduction to Multiagent Systems 26

http://www.csc.liv.ac.uk/~mjw/pubs/tmas/
Changes that occur in it (in time for the response to be useful).

A reactive system is one that maintains an ongoing interaction with its environment, and responds to changes that occur in it.

The real world is not like that. Most environments are not fixed.

If a program’s environment is guaranteed to be fixed,

A program can just execute blindly.

A reactive system asks itself whether it is worth executing.
Proactiveness

- Reacting to an environment is easy (e.g., stimulus → response rules).
- But we generally want agents to do things for us.
- Hence goal directed behaviour.
- Pro-activeness = generating and attempting to achieve goals; not driven solely by events; taking the initiative.
- Recognising opportunities.

Hence goal directed behaviour.

Pro-activeness
The real world is a multi-agent environment: we cannot go around attempting to achieve goals without taking others into account. Some goals can only be achieved by interacting with others. Similarly for many computer environments: witness the INTERNET.

Social Ability
Social ability in agents is the ability to interact with other agents (and possibly humans) via cooperation, coordination, and negotiation.

At the very least, it means the ability to communicate...
Social Ability: Cooperation

Cooperation is working together as a team to achieve a shared goal.

Often prompted either by the fact that no one agent can achieve the goal alone, or that cooperation will obtain a better result (e.g., get result faster).

• Cooperation is working together as a team to achieve a shared goal.

• Cooperation is working together as a team to achieve a shared goal.
Social Ability: Coordination

Coordination is managing the interdependencies between activities.

• For example, if there is a non-sharable resource that you want to use and I want to use, then we need to coordinate.

• Coordination is managing the interdependencies between activities.
Negotiation is the ability to reach agreements on matters of common interest.

Typically involves offer and counter-offer, with compromises made by participants.

A possible deal: watch football tonight, and a movie tomorrow.

For example: You have one TV in your house; you want to watch a movie, your housemate wants to watch football tonight, and you want to watch football tomorrow.
Some Other Properties:

- Mobility
- Veracity
- Benevolence
- Rationality
- Learning/adaption
Are agents just objects by another name?

Object:

- encapsulates some state;
- communicates via message passing;
- has methods corresponding to operations that may be performed on this state.

http://www.csc.liv.ac.uk/~mjw/pubs/imas/
Chapter 2

An Introduction to Multiagent Systems

Differences between Agents & Objects

- Agents are active:

  • Agents are active:

    - such types of behavior: capable of flexible (reactive, proactive, social)

- Agents are smart:

  - another agent:

    - whether or not to perform an action on request from

- Agents are autonomous:

  Differences between Agents & Objects

http://www.csc.liv.ac.uk/~mjw/pubs/imas/
Objects do it for free...

- agents do it for money.
- agents do it because they want to.

Objects do it for free...
Aren't agents just expert systems by another name?

Expert systems typically embody disembodied expertise.

Example: MYCIN knows about blood diseases in humans.

It has a wealth of knowledge about blood diseases in humans.

A doctor can obtain expert advice about blood diseases by giving MYCIN facts, answering questions, and posing queries.

Expert systems embody disembodied expertise.

Example: MYCIN knows about blood diseases in humans.

Aren't agents just expert systems by another name?
Differences between Agents & Expert Systems

- **Agents are** agents:
  - act
  - situated in an environment

  MYCIN does not operate on patients.

- **Some real-time** (typically process control) expert systems are agents:

  MYCIN is not aware of the world — only information obtained is by asking the user questions.

http://www.csc.liv.ac.uk/~mjw/pubs/imas/
Aren't agents just the AI project?

Isn't building an agent what AI is all about?

Aren't agents just the AI project?

 Aren't agents just the AI project?
When building an agent, we simply want a system that can choose the right action to perform, typically in a limited domain.

We do not have to solve all the problems of AI to build a useful agent.

A limited domain. That can choose the right action to perform, typically in a domain.

When building an agent, we simply want a system that can do intelligence goes a long way!

Oren Etzioni, speaking about the commercial experience of NETBOT, Inc:

We made our agents dumber and dumber... until finally they made money.

http://www.csc.liv.ac.uk/~mjw/pubs/tmas/
Properties of Environments

- Accessible vs Inaccessible.

An accessible environment is one in which the agent can obtain complete, accurate, up-to-date information about the environment's state. An accessible environment is one in which the agent can obtain complete, accurate, up-to-date information about the environment's state. Most moderately complex environments (including, for example, the everyday physical world and the Internet) are inaccessible.

The more accessible an environment is, the simpler it is to build agents to operate in it. The more accessible an environment is, the simpler it is to build agents to operate in it.
As we have already mentioned, a deterministic environment is one in which any action has a single guaranteed effect — there is no uncertainty about the state that will result from performing an action.

Non-deterministic environments present greater problems for the agent designer.

The physical world can to all intents and purposes be regarded as non-deterministic. As we have already mentioned, a deterministic environment is one in which any action has a single guaranteed effect — there is no uncertainty about the state that will result from performing an action.

Non-deterministic vs non-deterministic.

•

http://www.csc.liv.ac.uk/~mjw/pubs/imas/
Episodic vs non-episodic.

In episodic environments, the performance of an agent in a number of discrete episodes, with no link between the performance of an agent in different scenarios, is dependent on the current episode — it need not reason about the interactions between this and future episodes. In an episodic environment, the performance of an agent is dependent on a number of discrete episodes, with no link between the performance of an agent in different scenarios.

Episodic environments are simpler from the agent developer’s perspective because the agent can decide what action to perform based only on the current episode — it need not reason about the interactions between this and future episodes. Episodic environments are simpler from the agent developer’s perspective because the agent can decide what action to perform based only on the current episode — it need not reason about the interactions between this and future episodes.

In episodic environments, the performance of an agent in a number of discrete episodes, with no link between the performance of an agent in different scenarios, is dependent on the current episode — it need not reason about the interactions between this and future episodes.
Chapter 2: An Introduction to Multiagent Systems

Static vs. dynamic.

A static environment is one that can be assumed to remain unchanged except by the performance of actions by the agent. A dynamic environment is one that has other processes operating on it, and which hence changes in ways beyond the agent's control. The physical world is a highly dynamic environment.

http://www.csc.liv.ac.uk/~mjw/pubs/imas/
An environment is discrete if there are a fixed, finite number of actions and percepts in it. Russell and Norvig give a chess game as an example of a discrete environment, and taxi driving as an example of a continuous one.
Agents as Intentional Systems

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

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

These statements make use of a folk psychology, by attributing attitudes such as believing, wanting, hoping, fearing, hoping, wanting, believing, wanting, hoping, fearing, ....

When explaining human activity, we use statements like the following:
Daniel Dennett coined the term *intentional system* to describe entities whose behavior can be predicted by the method of attributing beliefs and desires. A first-order intentional system has beliefs and desires, and no doubt other intentional states. A second-order intentional system has beliefs and desires about beliefs and desires (and other intentional states), but no beliefs and desires about beliefs and desires (etc.). A *second-order intentional system* is more sophisticated; it has beliefs and desires (and no doubt other intentional states) about beliefs and desires (and other intentional states) — both those of others and its own.
Can We Apply the Intentional Stance to Machines?

An Introduction to Multiagent Systems 2e

Known. (John McCarthy)

operating systems, but is most useful when applied to entities whose structure is incompletely
operational for machines of known structure such as thermostats and computer
in a simpler setting than for humans, and later applied to humans. Ascription of mental qualitites
isomorphiC to them, Theories of belief, knowledge and wanting can be constructed for machines
about the state of the machine in a particular situation may require mental qualitites or qualitites
logically required even for humans, but expressing reasonably briefly what is actually known
the machine, its past or future behaviour, or how to repair or improve it. It is perhaps never
expresses about a person. It is useful when the ascription helps us understand the structure of
legitimate when such an ascription expresses the same information about the machine that it
To ascribe beliefs, free will, intentions, consciousness, abilities, or wants to a machine is

http://www.csc.liv.ac.uk/~mjw/pubs/imas/
What can be described with the intentional stance?

Consider a light switch:

'It is perfectly coherent to treat a light switch as a (very cooperative) agent with the capability of transmitting current at will, who invariably transmits current when it believes that we want it to, and not otherwise; flicking the switch is simply our way of communicating our desires.'

(Yoav Shoham)
Most adults would find such a description absurd! The more we know about a system, the less we need description of its behaviour. (Yoav Shoham)

While the intentional stance description is consistent, it does not buy us anything, since we essentially understand the mechanism sufficiently to have a simpler, mechanistic description of its behaviour.

While an animistic, intentional explanation of its behaviour.

The more we know about a system, the less we need to rely on animistic, intentional explanations of its behaviour. (Yoav Shoham)
But with very complex systems, a mechanistic explanation of its behaviour may not be practicable.

As computer systems become ever more complex, we need more powerful abstractions and metaphors to explain their operation — low level explanations become impractical.

The intentional stance is such an abstraction.

http://www.csc.liv.ac.uk/~mjw/pubs/imas/
The intentional notions are thus abstraction tools, which provide us with a convenient and familiar way of describing, explaining, and predicting the behaviour of complex systems.

Agents, and agents as intentional systems, represent:

- objects;
- abstract data types;
- procedural abstraction;
- complex systems.

The intentional notions are thus abstraction tools.

Remember: most important developments in computing are based on new abstractions.
Points in favour of this idea:

- It provides us with a familiar, non-technical way of understanding & explaining agents.

- It offers us the potential to specify systems that include representations of other systems.

- Nested Representations are essential for agents that must cooperate with other agents.

- Characterising Agents:
  
  It is widely accepted that such nested representations include representations of other systems.
With some built-in theory of rational agency, knowing what it will act in accordance with a high-level description of the delegated goal, and let the control mechanism figure out what to do, knowing that it will act in accordance with some built-in theory of rational agency, give the system general information about the relationships between objects, and let a built-in control mechanism (e.g., goal-directed theorem proving) figure out what to do.

In declarative programming, we state something that we want to achieve, give the system general information about what a system should do.

In procedural programming, we say exactly what a procedural control mechanism should do.

Post-Declarative Systems
The message $m_2$.

\text{IF} \quad \text{process i knows process j has received message } m_1 \\quad \text{THEN} \quad \text{process i should send process j the message } m_2.

\text{IF} \quad \text{process i knows process j has received message } m_1 \quad \text{THEN} \quad \text{process i should send process j the message } m_2.

An aside...
Abstract Architectures for Agents

• Assume the environment may be in any of a finite set of discrete, instantaneous states: $E$
• Agents are assumed to have a repertoire of possible actions available to them, which transform the state of the environment.
• A run, $r$, of an agent in an environment is a sequence of interleaved environment states and actions:

$$r: e_0 \overset{\alpha_0}{\rightarrow} e_1 \overset{\alpha_1}{\rightarrow} e_2 \overset{\alpha_2}{\rightarrow} \cdots$$

• Assume the environment is a finite set of discrete, instantaneous states: $E$

The environment.

$$\{ \ldots, e, e' \} = \mathcal{E}$$

$\mathcal{E}$
Let... runs

- $R_F$ be the subset of these that end with an environment state.
- $R_A$ be the subset of these that end with an action.
- $R_{AE}$ be the subset of these that end with an action (over $F$ and $Ac$).

And

- $R_E$ be the subset of all such possible finite sequences.

Let...
An Introduction to Multiagent Systems

Chapter 2

Environments

• A state transformer function represents behaviour of
  the environment:

\[ (E)^c \leftarrow \exists \forall c \]
An environment \( \mathcal{E} \) is then a triple \( \langle \mathcal{E}, \mathcal{E}^0, \tau \rangle \) where \( \mathcal{E} \) is set of environment states, \( \mathcal{E}^0 \) is initial state, and \( \tau \) is state transformer function.
Agents

Let $\mathcal{A}$ be the set of all agents.

Thus an agent makes a decision about what action to perform based on the history of the system that it has witnessed to date.

Let $\mathcal{A}_g$ be the set of all agents.

$\mathcal{A}_g : \mathcal{R} \rightarrow \mathcal{A}_c$

Agent is a function which maps runs to actions.
A system is a pair containing an agent and an environment.

Assume $R(\text{Ag}_i, \text{Env})$ contains only runs that have ended.

- Any system will have a set of possible runs, we denote the set of runs of agent $\text{Ag}_i$ in environment $\text{Env}$ by $R(\text{Ag}_i, \text{Env})$. 
- Any system will have a set associated with it.

系是系 is a pair containing an agent and an environment.
Formally, a sequence

\((e_0, \alpha_0, e_1, \alpha_1, e_2, \alpha_2, \ldots)\)

represents a run of an agent \(Ag\) in environment \(Env\).

1. \(e_0\) is the initial state of \(Env\).
2. \(\alpha_0 = Ag(e_0)\); and
3. for \(n < 0\),

\[\forall n \in \mathbb{N}, \langle e_0, \alpha_0, \ldots, \alpha_{n-1}, e_n \rangle \in Env\]
• Some agents decide what to do without reference to their history—they base their decision making entirely on the present, with no reference at all to the past.

- A thermostat is a purely reactive agent.

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

We call such agents purely reactive:

- Some agents decide what to do without reference to their history—purely reactive agents.
Now introduce perception system:
The **see** function is the agent's ability to observe its environment, whereas the **action** function represents the agent's decision-making process.

Output of the **see** function is a **percept**: percepts, which maps sequences of percepts to actions.

\[
\text{action : Per}^* \rightarrow A
\]

The **see** function is the agent's ability to observe its environment, whereas the **action** function represents the agent's decision-making process.
We now consider agents that maintain state:

Agents with State
These agents have some internal data structure, which is typically 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 $\text{see}$ for a state-based agent is unchanged:

$$\text{see} : E \rightarrow \text{Per}$$
The action-selection function action is now defined as a mapping from internal states to actions.

\[ \text{action} : I \rightarrow A_c \]
A function $\text{next}$ is introduced, which maps an internal state and percept to an internal state:

\[
\text{next} : I \times \text{Per} \rightarrow I
\]

Next State Function
Agent control loop

1. Agent starts in some initial internal state $i_0$.
2. Repeat forever:
   - Observe environment state, and generate a percept through $\text{see}(\cdot)$.
   - Update internal state via $\text{next}(\cdot)$.
   - Select action via $\text{action}(\cdot)$.
   - Perform action.

Agent control loop
Tasks for Agents

- We build agents in order to carry out tasks for us.
- The task must be specified by us.
- But we want to tell agents what to do without telling them how to do it.
Utilities are functions over states. One possibility: associate utilities with individual states that maximise utility. A task specification is a function $\mathbb{R} \leftarrow E : n$ — the task of the agent is then to bring about states with individual utilities maximised. One possibility: associate utilities utilities functions over states.
But what is the value of a run...

(One possibility: a *discount* for states later on.)

Disadvantage: difficult to specify a long term view

- average?
- sum of utilities of states on run?
- maximum utility of state on run?
- minimum utility of state on run?

But what is the value of a run...

An Introduction to Multiagent Systems 2e
Another possibility: assigns a utility not to individual states, but to runs themselves:

\[
\mathbb{R} \leftarrow \mathbb{R}^n
\]

Such an approach takes an inherently long term view.

Other variations: incorporate probabilities of different states emerging.

Utilities over runs
Problems with Utility-based Approaches

- "Wheredothenumberscomefrom?" (Peter Cheeseeman)
- People don’t think in terms of utility — it’s hard for people to specify tasks in these terms.
- Nevertheless, works well in certain scenarios...
Utility in the Tile world

• Simulated two dimensional grid environment on which there are agents, tiles, obstacles, and holes.

• An agent can move in four directions, up, down, left, right, and 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, with the aim being to fill as many holes as possible.

• Simultaneously changes with the random appearance and disappearance of holes.

• There are agents, tiles, obstacles, and holes.

• An Introduction to Multiagent Systems 2e

http://www.csc.liv.ac.uk/~mjw/pubs/imas/
Utility function defined as follows:

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

Thus:

\[ \frac{\text{number of holes filled in } r}{\text{number of holes that appeared in } r} = (\lambda)^n \]

Utility function defined as follows:

Utility in the Tileworld
The expected utility of agent $Ag$ in environment $Env$ is then:

$$EU(Ag, Env) = \sum_{r \in R} u(r) p(r | Ag, Env)$$

Note: Write $p(r | Ag, Env)$ to denote probability that run $r$ occurs when agent $Ag$ is placed in environment $Env$. 

\[ \sum_{r \in R} p(r | Ag, Env) = 1 \]
Consider the environment defined as

$$L_0 \vdash \langle E, \alpha_0 \rangle, \Gamma$$

There are two agents possible with respect to this environment:

$$\{ e_0, e_1, e_2, e_3 \} = L_0 \vdash \langle 0, \alpha_0 \rangle, \Gamma$$

$$\{ e_0, e_1, e_2, e_3, e_4 \} \subseteq \{ e_0, e_1, e_2, e_3 \}$$

Chapter 2

An Introduction to Multiagent Systems 2e

http://www.csc.liv.ac.uk/~mjw/pubs/imas/
The probabilities of the various runs are as follows:

\[ P(e_0^{\alpha_0} \rightarrow e_1 | Ag_1, Env_1) = 0.4 \]
\[ P(e_0^{\alpha_0} \rightarrow e_2 | Ag_1, Env_1) = 0.6 \]
\[ P(e_0^{\alpha_1} \rightarrow e_3 | Ag_2, Env_1) = 0.1 \]
\[ P(e_0^{\alpha_1} \rightarrow e_4 | Ag_2, Env_1) = 0.2 \]
\[ P(e_0^{\alpha_1} \rightarrow e_5 | Ag_2, Env_1) = 0.7 \]

Assume the utility function \( u_1 \) is defined as follows:

\[ u_1(e_0^{\alpha_0} \rightarrow e_1) = 8 \]
\[ u_1(e_0^{\alpha_0} \rightarrow e_2) = 11 \]
\[ u_1(e_0^{\alpha_1} \rightarrow e_3) = 70 \]
\[ u_1(e_0^{\alpha_1} \rightarrow e_4) = 9 \]
\[ u_1(e_0^{\alpha_1} \rightarrow e_5) = 10 \]

What are the expected utilities of the agents for this utility function?
The optimal agent $A_{opt}$ in an environment $Env$ is the one that maximizes expected utility:

$$\forall g \in \mathcal{AG} \max_{Env} EU(g, Env) = ido$$

The optimal agent $A_{opt}$ in an environment $Env$ is the one that will do best; only that on average we can expect it to do best.

Of course, the fact that an agent is optimal does not mean that it will be best; only that on average we can expect it to do best.
Some agents cannot be implemented on some computers.

\[ AG_{\text{opt}} = \arg\max_{AG \in AG_m} EU(AG, Env) \]

The bounded optimal agent, \( AG_{\text{opt}} \), with respect to \( m \): Write \( AG_m \) to denote the agents that can be implemented on some computer.

Some agents cannot be implemented on some computers.
• As a special case of assigning utilities to histories, assign 0 (false) or 1 (true) to a run.

- Call these predicate task specifications.
- Denote predicate task specification by $\Psi$.

$\{0, 1\} \leftarrow \forall \Psi$

If a run is assigned 1, then the agent succeeds on that run, otherwise it fails.

A special case of assigning utilities to histories is to assign 0 (false) or 1 (true) to a run.

**Predicate Task Specifications**
A task environment specifies:

- the properties of the system the agent will inhabit;
- the properties of the system the agent will inhabit;
- the criteria by which an agent will be judged to have
  either failed or succeeded.

Let $\mathcal{T}$ be the set of all task environments.

is a predicate over runs.

$$\{1,0\} \leftarrow \forall : I$$

environment, and

A task environment is a pair $\langle \text{Env}, \Psi \rangle$, where $\text{Env}$ is an

An Introduction to Multiagent Systems 2e

Chapter 2

http://www.csc.liv.ac.uk/~mjw/pubs/tmas/
\( R^\Psi(\text{Ag}, \text{Env}) \)

\( R^\Psi(\text{Ag}, \text{Env}) = \{ r \mid r \in R(\text{Ag}, \text{Env}) \text{ and } \Psi(r) = 1 \} \)

We then say that an agent \( \text{Ag} \) succeeds in task \( \langle \text{Env}, \Psi \rangle \) if \( R^\Psi(\text{Ag}, \text{Env}) = R(\text{Ag}, \text{Env}) \).

\( R(\text{Ag}, \text{Env}) \) denotes the set of all runs of the environment \( \text{Env} \) that satisfy the agent \( \text{Ag} \) in the environment \( \text{Env} \).
Let \( P(r | Ag, Env) \) denote the probability that run \( r \) occurs if agent \( Ag \) is placed in environment \( Env \).

Then the probability \( P(\Psi | Ag, Env) \) that \( \Psi \) is satisfied by \( Ag \) in \( Env \) would then simply be:

\[
\sum_{r \in R} P(r | Ag, Env) \cdot P(\Psi | Ag, Env) \cdot P(\Phi | Ag, Env) \]

\( \sum_r \) by \( Ag \) in \( Env \) that \( \Phi \) is satisfied by \( Ag \) in \( Env \).

Let \( P(r | Ag, Env) \) denote the probability that run \( r \) occurs if agent \( Ag \) is placed in environment \( Env \).

The Probability of Success
• Two most common types of tasks are achievement tasks and maintenance tasks:
An achievement task is specified by a set of "good" or "goal" states: $G \subseteq E$. The agent succeeds if it is guaranteed to bring about at least one of these states (we do not care which one).

A maintenance goal is specified by a set of "bad" states: $B \subseteq E$. The agent succeeds if it manages to avoid all states in $B$ — it never performs actions which result in any state in $B$ occurring.

Maintenance goals are considered equally good — they are all considered equally good. An achievement task is specified by a set of "good" or "goal" states: $G \subseteq E$. The agent succeeds if it is guaranteed to bring about at least one of these states.
Agent synthesis is automatic programming: goal is to have a program that will take a task environment, and from this task environment automatically generate an agent that succeeds in this environment.

\[ \text{syn} : (\{ \top \} \cap \delta A) \mapsto 3T \]

Think of \( \top \) as being like \texttt{null} in Java.

Agent synthesis is automatic programming.
Soundness and Completeness

• Synthesis algorithm is:

- complete if it is guaranteed to return an agent whenever there exists an agent that will succeed in the task environment given as input; and
- sound if, whenever it returns an agent passed as input, that agent succeeds in the task environment given as input; then this

http://www.csc.liv.ac.uk/~mjw/pubs/imas/
Synthesis algorithm \( \text{syn} \) is sound if it satisfies the following condition:

\[
\text{syn} (⟨\Phi, \text{Env}⟩) = \text{Ag} \implies R(\text{Ag}, \text{Env}) = R(\text{Ag}, \text{Env}),
\]

and complete if:

\[
\forall \text{Ag} \in \text{Ag} \text{'s.t. } R(\text{Ag}, \text{Env}) = R(\text{Ag}, \text{Env}) \implies \text{syn}(⟨\Phi, \text{Env}⟩) \neq ⊥.
\]