Utility functions and Concrete architectures: deductive agents

CS7ET03: AI for IET

Lecturer: S Luz
luzs@cs.tcd.ie

September 29, 2014
Abstract Architectures (ctd.)

- An agent's behaviour is encoded as its history:

\[ h : s_0 \xrightarrow{a_0} s_1 \xrightarrow{a_1} \ldots \xrightarrow{a_{u-1}} s_u \xrightarrow{a_u} \ldots \]

- Define:
  - \( H^A \): set of all histories which end with an action
  - \( H^S \): histories which end in environment states

- A state transformer function \( \tau : H^A \rightarrow \wp(S) \) represents the effect an agent has on an environment.

- So we may represent environment dynamics by a triple:

\[ Env =< S, s_0, \tau > \]

- And similarly, agent dynamics as

\[ Ag : H^S \rightarrow A \]
Utility functions

- Problem: how to “tell agents what to do”? (when exhaustive specification is impractical)

- Decision theory (see [Russell and Norvig, 1995, ch. 16]):
  - associate *utilities* (a performance measure) to states:
    \[ u : S \rightarrow \mathbb{R} \]
  - Or, better yet, to *histories*:
    \[ u : H \rightarrow \mathbb{R} \]
Example: The Tileworld

The utility of a course of action can be given by:

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

When the utility function has an upper bound (as above) then we can speak of optimal agents.
Example: The Tileworld

The utility of a course of action can be given by:

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

When the utility function has an upper bound (as above) then we can speak of optimal agents.
Optimal agents

Let $P(h|Ag, Env)$ denote the probability that $h$ occurs when agent $Ag$ is placed in $Env$.

Clearly

$$\sum_{h \in H} P(h|Ag, Env) = 1$$

An optimal agent $Ag_{opt}$ in an environment $Env$ will maximise expected utility:

$$Ag_{opt} = \arg \max Ag \sum_{h \in H} u(h)P(h|Ag, Env)$$
From abstract to concrete architectures

- Moving from abstract to concrete architectures is a matter of further specifying action (e.g. by means of algorithmic description) and choosing an underlying form of representation.

- Different ways of specifying the action function and representing knowledge:
  - Logic-based: decision function implemented as a theorem prover (plus control layer)
  - Reactive: (hierarchical) condition $\rightarrow$ action rules
  - BDI: manipulation of data structures representing Beliefs, Desires and Intentions
  - Layered: combination of logic-based (or BDI) and reactive decision strategies
Logic-based architectures

- AKA *Deductive Architectures*
- Background: symbolic AI
  - Knowledge representation by means of logical formulae
  - “Syntactic” symbol manipulation
  - Specification in logic $\Rightarrow$ executable specification
- “Ingredients”:
  - Internal states: sets of (say, first-order logic) formulae
    - $\Delta = \{ \text{temp}(\text{roomA}, 20), \text{heater}(\text{on}), \ldots \}$
  - Environment state and perception,
  - Internal state seen as a set of beliefs
  - Closure under logical implication ($\Rightarrow$):
    - $\text{closure}(\Delta, \Rightarrow) = \{ \varphi | \varphi \in \Delta \lor \exists \psi. \psi \in \Delta \land \psi \Rightarrow \varphi \}$
  - (is this a reasonable model of an agent's beliefs?)
Representing deductive agents

We will use the following objects:

- \( L \): a set of sentences of a logical system
  - As defined, for instance, by the usual wellformedness rules for first-order logic
- \( D = \wp(L) \): the set of databases of \( L \)
- \( \Delta_0, ..., \Delta_n \in D \): the agent’s internal states (or beliefs)
- \( \models_\rho \): a deduction relation described by the deduction rules \( \rho \) chosen for \( L \):
  We write \( \Delta \models_\rho \varphi \) if \( \varphi \in \text{closure}(\Delta, \rho) \)
Describing the architecture

- A logic-based architecture is described by the following structure:

\[ \text{Arch}_L = \langle L, A, P, D, \text{action}, \text{env}, \text{see}, \text{next} \rangle \]  

(3)

- The update function consists of additions and removals of facts from the current database of internal states:
  - \( \text{next} : D \times P \rightarrow D \)
  - \( \text{old} \): removal of “old” facts
  - \( \text{new} \): addition of new facts (brought about by \textit{action})
Pseudo-code for \textit{action}

1. function \textit{action}(\Delta : D) : A
2. begin
3. for each \(a \in A\) do
4. \hspace{1em} if \(\Delta \models^\rho \text{do}(a)\) then
5. \hspace{1em} \hspace{1em} return \(a\)
6. \hspace{1em} end if
7. \hspace{1em} end for
8. for each \(a \in A\) do
9. \hspace{1em} if \(\Delta \not\models^\rho \neg\text{do}(a)\) then
10. \hspace{2em} return \(a\)
11. \hspace{1em} end if
12. \hspace{1em} end for
13. return \textit{noop}
14. end function \textit{action}
Environment and Belief states

- The environment change function, $env$, remains as before.
- The belief database update function could be further specified as follows

\[
\text{next}(\Delta, p) = (\Delta \setminus \text{old}(\Delta)) \cup \text{new}(\Delta, p)
\]  

(4)

where $\text{old}(\Delta)$ represent beliefs no longer held (as a consequence of action), and $\text{new}(\Delta, p)$ new beliefs that follow from facts perceived about the new environmental conditions.
Example: The Vacuum world
Example: The Vacuum world
Example: The Vacuum world
Example: The Vacuum world

[Russell and Norvig, 1995, Weiss, 1999]
Describing The Vacuum world

- Environment
  - perceptual input: dirt, null
  - directions: (facing) north, south, east, west
  - position: coordinate pairs (x, y)

- Actions
  - move_forward, turn_90°_left, clean

- Perception:
  
  \[ P = \{ \{ \text{dirt}(x, y), \text{in}(x, y), \text{facing}(d), \ldots \}, \ldots \} \]
Deduction rules

- Part of the decision function
- Format (for this example):
  - $P(...) \Rightarrow Q(...)$
- PROLOG fans may think of these rules as Horn Clauses.
- Examples:
  - $in(x, y) \land dirt(x, y) \Rightarrow do(clean)$
  - $in(x, 2) \land \neg dirt(x, y) \land facing(north) \Rightarrow do(turn)$
  - $in(0, 0) \land \neg dirt(x, y) \land facing(north) \Rightarrow do(forward)$
  - ...

Updating the internal state database

- $\text{next}(\Delta, p) = (\Delta \setminus \text{old}(\Delta)) \cup \text{new}(\Delta, p)$
- $\text{old}(\Delta) = \{(P(t_1, \ldots, t_n) | P \in \{\text{in, dirt, facing}\} \land P(t_1, \ldots, t_n) \in \Delta\}$
- $\text{new}(\Delta, p)$:
  - update agent’s position,
  - update agent’s orientation,
  - etc
The “Wumpus World”

- See [Russell and Norvig, 2003, section 7.2]
- BTW, Ch 7 is available online (last accessed Oct 2012) at http://aima.cs.berkeley.edu/newchap07.pdf
Shortcomings of logic-based agents

- Expressivity issues: problems encoding percepts (e.g. visual data) etc
- *Calculative rationality* in dynamic environments
- Decidability issues
- Semantic elegance vs. performance:
  - loss of “executable specification”
  - weakening the system vs. temporal specification
- etc
Existing (??) logic-based systems

- MetameM, Concurrent MetameM: specifications in temporal logic, model-checking as inference engine (Fisher, 1994)
- CONGOLOG: Situation calculus
- Situated automata: compiled logical specifications (Kaelbling and Rosenschein, 1990)
- AgentSpeak, ...
- (see [Weiss, 1999] or [Wooldridge, 2002] for more details)
BDI Agents

- Implement a combination of:
  - deductive reasoning (*deliberation*) and
  - planning (*means-ends reasoning*)
Planning formalisms describe actions in terms of (sets of) \textit{preconditions} \((P_a)\), \textit{delete lists} \((D_a)\) and \textit{add lists} \((A_a)\):

\[
\langle P_a, D_a, A_a \rangle
\]

E.g. Action encoded in STRIPS (for the “block’s world” example):

Stack\((x,y)\):

\begin{itemize}
  \item \textbf{pre:} clear\((y)\), holding\((x)\)
  \item \textbf{del:} clear\((y)\), holding\((x)\)
  \item \textbf{add:} armEmpty, on\((x,y)\)
\end{itemize}
A planning problem $\langle \Delta, O, \gamma \rangle$ is determined by:

- the agent's beliefs about the initial environment (a set $\Delta$)
- a set of operator descriptors corresponding to the actions available to the agent:
  $$O = \{ \langle P_a, D_a, A_a \rangle \mid a \in A \}$$
- a set of formulae representing the goal/intention to be achieved (say, $\gamma$)

A plan $\pi = \langle a_1, \ldots, a_n \rangle$ determines a sequence $\Delta_0, \ldots, \Delta_{n+1}$ where $\Delta_0 = \Delta$ and $\Delta_i = (\Delta_{i-1} \setminus D_{a_i}) \cup A_{a_i}$, for $1 \leq i \leq n$. 
Suggestions

- Investigate the use of BDI systems and agents in games.
- See, for instance, [Norling and Sonenberg, 2004] which describe the implementation of interactive BDI characters for Quake 2.
- And [Wooldridge, 2002, ch. 4], for some background.
- There are a number of BDI-based agent platforms around. ‘Jason’, for instance, seems interesting: 
  
  http://jason.sourceforge.net/


