G53DIA: Designing Intelligent Agents

Lecture 7: Deliberative Architectures

Brian Logan
School of Computer Science & IT
`bsl@cs.nott.ac.uk`

Outline of this lecture

- deliberative architectures
- the role of representation
- advantages of deliberation
- examples
  - Shakey the robot
  - a simple planning agent

Agent architectures

The *architecture* of an agent

- defines the atomic operations of the agent program, and implicitly determines the components of the agent
- determines which operations happen automatically, without the agent program having to do anything

The atomic operations together with which operations happen automatically determine whether an architecture is reactive or deliberative, e.g., a deliberative architecture would typically include automatic comparison of alternatives.

Limitations of reactive architectures

There are many things reactive systems can't do

- they can't represent or reason about remote objects and times;
- they don't do well in domains where plausible actions can't be ignored or undone if they prove to be unwise;
- it is difficult for purely reactive agents to organise their own activities over time or to coordinate their behaviour with that of other agents in a non-trivial way.

It is therefore difficult to get intelligent behaviour from a purely reactive system.

Deliberation

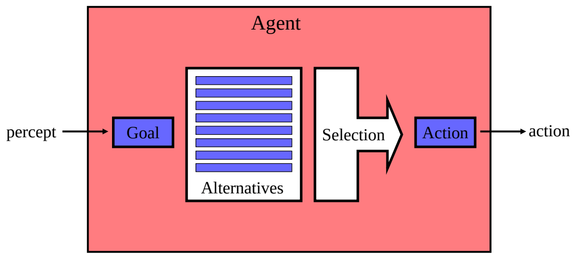
Deliberation is the explicit consideration of alternative courses of action:

- generating alternatives
- choosing an alternative

An agent can deliberate about both

- ends—whether to intend a goal
- means—how to achieve a goal

Deliberative architecture



The role of representations

Deliberation requires representations of the world and courses of action:

- deliberation involves the manipulation of a model of the world and possible courses of action, rather than the world itself
- it requires the ability to represent actions and derive the consequences of actions without actually performing them
- the result of deliberation is a description of an action or actions to be performed, rather than the performance of the actions in the world

Counterfactual representations

To represent desired states and the consequences of actions:

- some states of the agent are *counterfactual* in the sense of referring to hypothetical past or future states (goals) or as yet unexecuted actions (plans)
- some of the basic operations of the architecture produce/read/write such counterfactual states
- such states are influential in the choice of actions

To represent hypothetical situations, a deliberative agent requires representations with compositional semantics.

Advantages of deliberation

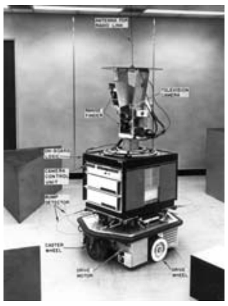
Deliberation

- is useful when the penalty for incorrect actions is high, e.g., when the environment is hazardous
- allows us to code a *general procedure* for finding a solution to a class of problems
  - may be better than reactive systems at coping with novel problems
  - we may be able to get a correct or even an optimal answer, e.g., decision theoretic approaches.

Shakey the robot (1966–1972)

Shakey was the first mobile robot to reason about its actions.

- multiple sensors (TV camera, a triangulating range finder, and bump sensors)
- connected to DEC PDP-10 and PDP-15 computers via radio and video links
- programs for perception, world-modeling, and acting (simple motion, turning, and route planning)



The planning problem

Planning in the real world is a hard problem:

- the agent has imperfect knowledge of the world;
- the world is dynamic;
- the world is non-deterministic;
- the agent may make errors executing the plan;
- planning itself takes time.

Simplifying assumptions

We therefore make the following simplifying assumptions:

- the agent has perfect knowledge about the world;
- the world doesn't change unless the agent changes it;
- the world is deterministic;
- plan execution is error-free; and
- computation is free.

Classical planning is the production of a complete, totally ordered set of actions, which, when executed in a given initial situation, will achieve a goal.

Shakey's planning

- Shakey inhabited a specially constructed simplified environment containing blocks of different colours and shapes
- the environment was static—there were no other agents and nothing moved unless Shakey moved it
- it planned actions such as pushing an object from one place to another and then executed the plan
- Shakey planned using the STRIPS planning system, which used information stored in a symbolic world model to determine what actions to take to achieve the robot's goal
- perception provided the information to maintain the representations in the world model

Example: playing Hide-and-Seek

- plays a version of the children's game, Hide-and-Seek in a 3D environment consisting of hills, valleys etc.
- uses classical problem solving (ABC algorithm)
- uses hierarchical planning to limit problem complexity
- plans then acts

The agent's environment

The agent's model

The agent's goals

The agents play Hide-and-Seek in the terrain models:

- plan a route which allows the agent to observe a series of locations;
- plan a route which is concealed from a number of observers whose locations are known;
- plan a route to observe a series of locations while remaining concealed from observers at those locations;

Search problems

- A search problem consists of:
  - a set of *states*: complete descriptions of the world for the purposes of problem-solving;
  - a set of *operators*: actions that transform one state into another state;
  - the *state space* is the set of all states reachable from the initial state by any sequence of actions.
- a *path* in the state space is any sequence of actions leading from one state to another.
- we use a *path cost function, g(n),* to assign a quality measure to a path.
- a *solution* is a path from an initial state to a goal state.

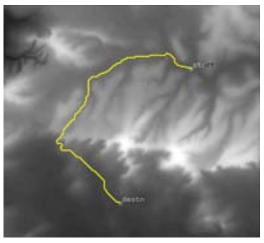
Route planning

In the route planning problem, the states and operators are straightforward:

- the states are just the possible locations of the agent, e.g., the cells in our terrain model
- the operators are then just the actions of moving from a cell to each of its neighbours
- the cost function is the hard part

An example plan

The output of the planner is a sequence of actions to be performed:



The Next Lecture

Further Deliberation

Suggested reading:

- Russell & Norvig (1995), chapter 13;
- Wooldridge (2002), chapter 4.