EH2750 Computer Applications in Power Systems, Advanced Course.

Lecture 3

Professor Lars Nordström, Ph.D.
Dept of Industrial Information & Control systems, KTH
larsn@ics.kth.se

Acknowledgement

• These slides are based largely on a set of slides provided by:
  
  Professor Rosenschein of the Hebrew University Jerusalem, Israel
  and
  Dr. Georg Groh, TU-München, Germany.

• Available at the Student companion site of the Introduction to Multi Agent Systems book

Outline of the Lecture

• Repeating where we are right now
• Practical reasoning Agents (Ch 4)
• Reactive Agents (Ch 5)
• Agent concepts and JACK

What is an Intelligent Agent?

• The main point about agents is they are autonomous: capable of acting independently, exhibiting control over their internal state
• Thus: an intelligent agent is a computer system capable of flexible autonomous action in some environment in order to meet its design objectives

input

System

Environment

output
Agent components

Beliefs, Desires & Intentions - BDI

- When we describe Intelligent Agents it is convenient to talk about them as if they have:
  - Beliefs
    - Some image of the environment
    - E.g. Temperature measurement
  - Desires
    - Goals they wish to achieve
    - E.g. Keep the temperature at the setpoint
  - Intentions
    - Actions that the agent can take
    - E.g. Increase Temperature!
    - Subsequent step to an Intention is a Plan (set of actions)
      - Open hot water valve

Where are we right now?

- Chapter 2 describes the idea of agents that perform tasks in an environment and sets some definitions
- Chapters 3, 4, & 5 describe three different approaches to describing and developing the apparent Intelligence in the agents.
  - Chapter 3 – Deductive Reasoning Agents
  - Chapter 4 – Practical Reasoning Agents
  - Chapter 5 - Reactive (and Hybrid Agents)

So, how do we make the agent think?

- One straightforward way is to use logic
- Program the agent to be completely logical and use deduction to prove it’s way to choosing which action to perform.

```java
function action(i:I) returns α:A {
    for each α in A do {
        if(i using ρ proves Do(α)) {
            return α
        }
    }
    for each α in A do {
        if(i using ρ does not prove NOT(Do(α))) {
            return α
        }
    }
    return null
}
```
Example: The Vacuum World

Agents database-rules:

Objective:

\[ \text{In}(x, y) \land \text{Dirt}(x, y) \rightarrow \text{Do(suck)} \]

Traversal:

\[ \text{In}(0, 0) \land \text{Facing(north)} \land \neg\text{Dir}(0, 0) \rightarrow \text{Do(forward)} \]
\[ \text{In}(0, 1) \land \text{Facing(north)} \land \neg\text{Dir}(0, 1) \rightarrow \text{Do(forward)} \]
\[ \text{In}(0, 2) \land \text{Facing(north)} \land \neg\text{Dir}(0, 2) \rightarrow \text{Do(turn)} \]
\[ \text{In}(0, 2) \land \text{Facing(east)} \rightarrow \text{Do(forward)} \]

and for all other rows accordingly

Deductive Agents – does that work?

- The idea of proving theorems as a way of making decisions is logically sound and rigorous

Two challenges remain:
1. It is time consuming to program
2. It is time consuming to execute

- Applied in a human setting it is also rather rigid.
  Imagine a theorem:
  - I will buy the cheapest copy of Wooldridge’s book.
  - Requires you to find a copy, check the price
    - Find next copy check price
    - Etc. until you have found all copies of the book
  - People tend to use Practical reasoning

Outline of the Lecture

- Repeating where we are right now
- Practical reasoning Agents (Ch 4)
- Reactive Agents (Ch 5)
- Agent concepts and JACK

Practical Reasoning

- Human practical reasoning consists of two activities:
  - deliberation
deciding what state of affairs we want to achieve
  - means-ends reasoning
deciding how to achieve these states of affairs
- The outputs of deliberation are intentions

What are possible things I could do? What is the best way to do it? Plans
What is deliberations?

• Beliefs, Desires, Intentions: Symbolically represented:
  \[ \text{Bel} = \{B, B', B'', \ldots\} \quad \text{Des} = \{D, D', D'', \ldots\} \quad \text{Int} = \{I, I', I'', \ldots\} \]

• Deliberation = \(<\text{option, filter}>\)
  \[ \text{options} : 2^{\text{Bel}} \times 2^{\text{Int}} \rightarrow 2^{\text{Des}} \]
  \[ \text{filter} : 2^{\text{Bel}} \times 2^{\text{Des}} \times 2^{\text{Int}} \rightarrow 2^{\text{Int}} \]

• Option generation function \text{option} generates desires (goals)
• Filtering function \text{filter} selects intentions (commitments)
• Belief revision function \text{brf} updates beliefs
  \[ \text{brf} : 2^{\text{Bel}} \times Per \rightarrow 2^{\text{Bel}} \]

What is Means-End Reasoning?

• Basic idea is to give an agent:
  - representation of goal/intention to achieve
  - representation actions it can perform
  - representation of the environment
  and have it generate a \text{plan} to achieve the goal
• Essentially, this is \text{automatic programming}

Planning is a big thing in AI

Planning

• Question: How do we represent... 
  - goal to be achieved 
  - state of environment 
  - actions available to agent 
  - plan itself

Plan to achieve goal

Diagram of planner

goals/ intention/ task
state of environment
possible action

plan to achieve goal
The Blocks World

- We'll illustrate the techniques with reference to the blocks world
  Contains a robot arm, 3 blocks (A, B, and C) of equal size, and a table-top

The Blocks World Ontology

- To represent this environment, need an ontology
  \( \text{On}(x, y) \)  \( \text{obj} \ x \text{ on top of obj} \ y \)
  \( \text{OnTable}(x) \)  \( \text{obj} \ x \text{ is on the table} \)
  \( \text{Clear}(x) \)  \( \text{nothing is on top of obj} \ x \)
  \( \text{Holding}(x) \)  \( \text{arm is holding} \ x \)

- The closed world assumption is implicitly valid.

The Blocks World

- Here is a representation of the blocks world described above:
  \( \text{Clear}(A) \)
  \( \text{On}(A, B) \)
  \( \text{OnTable}(B) \)
  \( \text{OnTable}(C) \)

- Use the closed world assumption: anything not stated is assumed to be false

The Blocks World

- A goal is represented as a set of formulae
  - Here is a goal:
    \( \text{OnTable}(A) \land \text{OnTable}(B) \land \text{OnTable}(C) \)
The Blocks World

- **Actions** are represented using a technique that was developed in the STRIPS planner.
- Each action has:
  - a **name** which may have arguments
  - a **pre-condition list** list of facts which must be true for action to be executed
  - a **delete list** list of facts that are no longer true after action is performed
  - an **add list** list of facts made true by executing the action

Each of these may contain **variables**

The Blocks World Operators

- **Example 1:**
  The *stack* action occurs when the robot arm places the object *x* it is holding is placed on top of object *y*.

  \[
  \text{Stack}(x, y) \quad \text{pre} \quad \text{Clear}(y) \land \text{Holding}(x) \\
  \text{del} \quad \text{Clear}(y) \land \text{Holding}(x) \\
  \text{add} \quad \text{ArmEmpty} \land \text{On}(x, y)
  \]

- **Example 2:**
  The *unstack* action occurs when the robot arm picks an object *x* up from on top of another object *y*.

  \[
  \text{UnStack}(x, y) \quad \text{pre} \quad \text{On}(x, y) \land \text{Clear}(x) \land \text{ArmEmpty} \\
  \text{del} \quad \text{On}(x, y) \land \text{ArmEmpty} \\
  \text{add} \quad \text{Holding}(x) \land \text{Clear}(y)
  \]

- **Example 3:**
  The *pickup* action occurs when the arm picks up an object *x* from the table.

  \[
  \text{Pickup}(x) \quad \text{pre} \quad \text{Clear}(x) \land \text{OnTable}(x) \land \text{ArmEmpty} \\
  \text{del} \quad \text{OnTable}(x) \land \text{ArmEmpty} \\
  \text{add} \quad \text{Holding}(x)
  \]

- **Example 4:**
  The *putdown* action occurs when the arm places the object *x* onto the table.

  \[
  \text{Putdown}(x) \quad \text{pre} \quad \text{Holding}(x) \\
  \text{del} \quad \text{Holding}(x) \\
  \text{add} \quad \text{Clear}(x) \land \text{OnTable}(x) \land \text{ArmEmpty}
  \]
**A Plan**

- What is a plan?
  - A sequence (list) of actions, with variables replaced by constants.

### Outline of the Lecture

- Repeating where we are right now
- Practical reasoning Agents (Ch 4)
- Reactive Agents (Ch 5)
- Agent concepts and JACK

### Reactive Architectures

- There are many unsolved (some would say insoluble) problems associated with symbolic AI
- These problems have led some researchers to question the viability of the whole paradigm, and to the development of reactive architectures
- Although united by a belief that the assumptions underpinning mainstream AI are in some sense wrong, reactive agent researchers use many different techniques
- In this presentation, we start by reviewing the work of one of the most vocal critics of mainstream AI: Rodney Brooks
Purely Reactive Agents (Repeat)

• 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
• We call such agents purely reactive:

A thermostat is a purely reactive agent

\[
\text{action : } E \rightarrow Ac
\]

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

The Subsumption Architecture

• A subsumption architecture is a hierarchy of task-accomplishing behaviors
• Each behavior is a rather simple rule-like structure
• Each behavior ‘competes’ with others to exercise control over the agent
• Lower layers represent more primitive kinds of behavior (such as avoiding obstacles), and have precedence over layers further up the hierarchy
• The resulting systems are, in terms of the amount of computation they do, extremely simple
• Some of the robots do tasks that would be impressive if they were accomplished by symbolic AI systems

Layered Control in the Subsumption Architecture

Steels’ Mars Explorer

• Steels’ Mars explorer system, using the subsumption architecture, achieves near-optimal cooperative performance in simulated ‘rock gathering on Mars’ domain:

The objective is to explore a distant planet, and in particular, to collect sample of a precious rock. The location of the samples is not known in advance, but it is known that they tend to be clustered.
Steels’ Mars Explorer Rules

- For individual (non-cooperative) agents, the lowest-level behavior, (and hence the behavior with the highest “priority”) is obstacle avoidance:
  \[ \text{if detect an obstacle then change direction} \] (1)

- Any samples carried by agents are dropped back at the mother-ship:
  \[ \text{if carrying samples and at the base then drop samples} \] (2)

- Agents carrying samples will return to the mother-ship:
  \[ \text{if carrying samples and not at the base then travel up gradient} \] (3)

- Agents will collect samples they find:
  \[ \text{if detect a sample then pick sample up} \] (4)

- An agent with “nothing better to do” will explore randomly:
  \[ \text{if true then move randomly} \] (5)

Outline of the Lecture

- Repeating where we are right now
- Practical reasoning Agents (Ch 4)
- Reactive Agents (Ch 5)
- Agent concepts and JACK

What is JACK

*JACK Intelligent Agents* is an *environment* for building, running and integrating commercial *Java-based* multi-agent software using a *component-based* approach.
Outline of the Lecture

- Repeating where we are right now
- Practical reasoning Agents (Ch 4)
- Reactive Agents (Ch 5)
- Agent concepts and JACK