



## 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](mailto: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



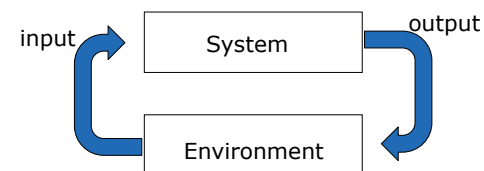
## Outline of the Lecture

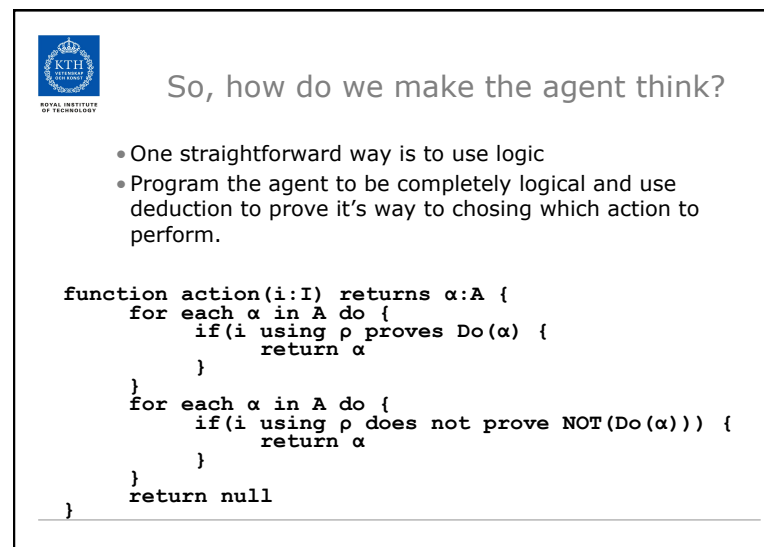
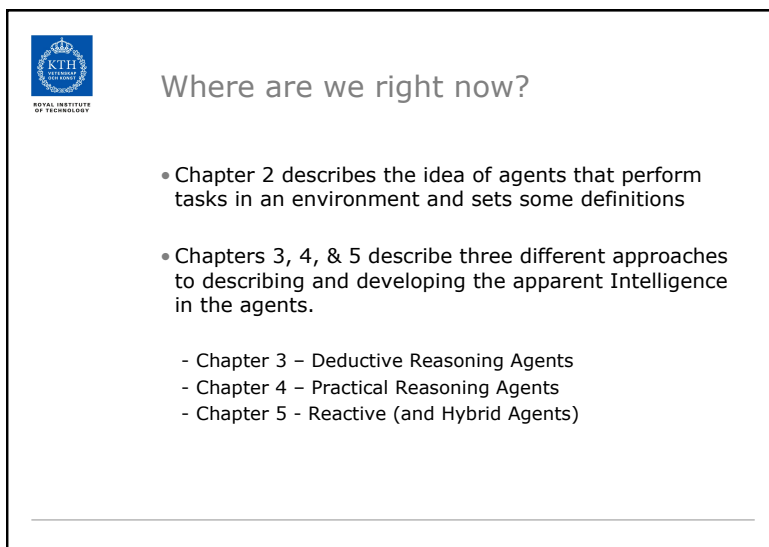
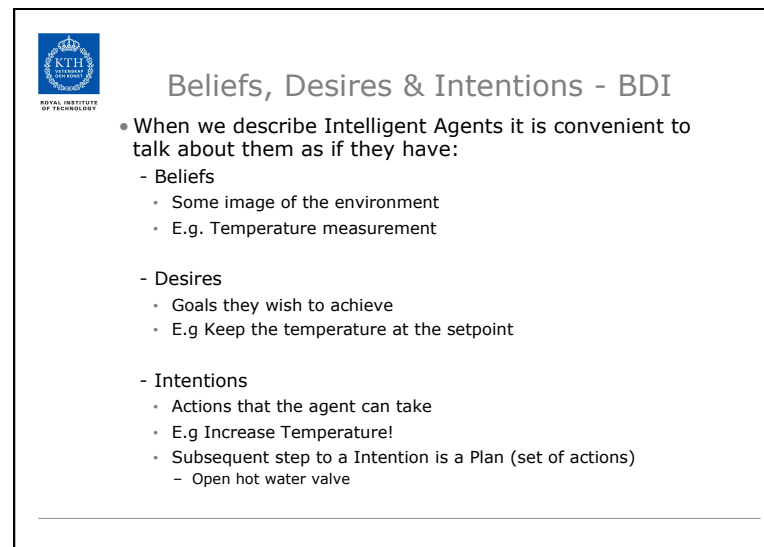
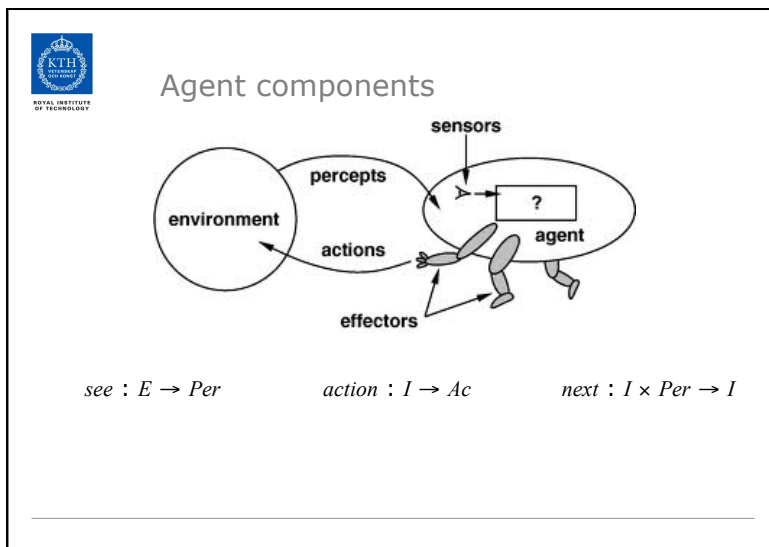
- Repeating where we are right now
- Practical reasoning Agents (Ch 4)
- Reactive Agents (Ch 5)
- Agent concepts and JACK




## 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*





 Example: The Vacuum World

Agents database-rules:

Objective:

$$\underbrace{In(x, y) \wedge Dirt(x, y)}_p \rightarrow Do(suck)$$

Traversal:

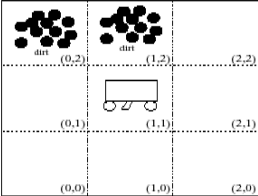
$$In(0, 0) \wedge Facing(north) \wedge \neg Dirt(0, 0) \rightarrow Do(forward)$$


$$In(0, 1) \wedge Facing(north) \wedge \neg Dirt(0, 1) \rightarrow Do(forward)$$

$$In(0, 2) \wedge Facing(north) \wedge \neg Dirt(0, 2) \rightarrow Do(turn)$$

$$In(0, 2) \wedge Facing(east) \rightarrow Do(forward)$$

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:

- It is time consuming to program
- It is time consuming to execute

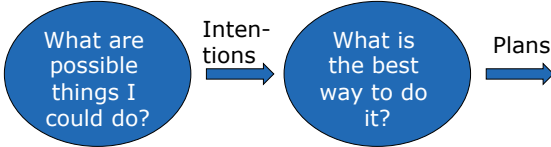
- Applied in a human setting it is also rather rigid. Imagine a theorem:
  - I will buy the cheapest copy of Wooldridge's book.
- Requires you to find a copy, check the price
  - Find next copy check price
  - Etc. until you have found all copies of the book
- People tend to use Practical reasoning

 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 is deliberations?

• Beliefs, Desires, Intentions: Symbolically represented:  
 $Bel = \{B, B', B'', \dots\}$      $Des = \{D, D', D'', \dots\}$      $Int = \{I, I', I'', \dots\}$

- Deliberation =  $\langle option, filter \rangle$

$$options : 2^{Bel} \times 2^{Int} \rightarrow 2^{Des}$$

$$filter : 2^{Bel} \times 2^{Des} \times 2^{Int} \rightarrow 2^{Int}$$

- Option generation function *option* generates desires (goals)
- Filtering function *filter* selects intentions (commitments)
- Belief revision function *brf* updates beliefs

$$brf : 2^{Bel} \times Per \rightarrow 2^{Bel}$$

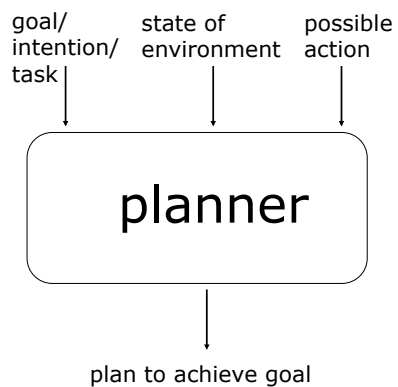


## What is Means-End Reasoning?

- 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 *plan* to achieve the goal
- Essentially, this is *automatic programming*




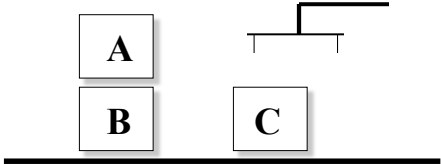
## Planning is a big thing in AI




## Planning

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


 The Blocks World



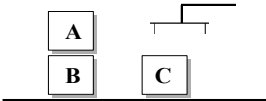
- We'll illustrate the techniques with reference to the *blocks world*. Contains a robot arm, 3 blocks (A, B, and C) of equal size, and a table-top.


 The Blocks World Ontology

- To represent this environment, need an *ontology*
  - $On(x, y)$  obj  $x$  on top of obj  $y$
  - $OnTable(x)$  obj  $x$  is on the table
  - $Clear(x)$  nothing is on top of obj  $x$
  - $Holding(x)$  arm is holding  $x$
- *The closed world assumption is implicitly valid.*

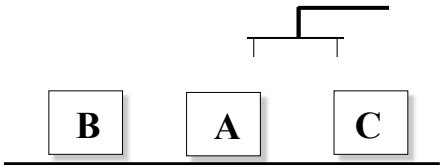
 The Blocks World

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



4-20



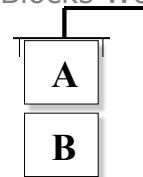
## The Blocks World

- **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  $y$ .

```

Stack(x, y)
pre  Clear(y) ∧ Holding(x)
del  Clear(y) ∧ Holding(x)
add  ArmEmpty ∧ On(x, y)

```



## The Blocks World Operators

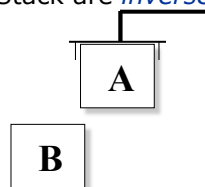
- **Example 2:**  
The *unstack* action occurs when the robot arm picks an object  $x$  up from on top of another object  $y$ .

```

UnStack(x, y)
pre  On(x, y) ∧ Clear(x) ∧ ArmEmpty
del  On(x, y) ∧ ArmEmpty
add  Holding(x) ∧ Clear(y)

```

Stack and UnStack are *inverses* of one-another.



## The Blocks World Operators

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

```

Pickup(x)
pre  Clear(x) ∧ OnTable(x) ∧ ArmEmpty
del  OnTable(x) ∧ ArmEmpty
add  Holding(x)

```

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

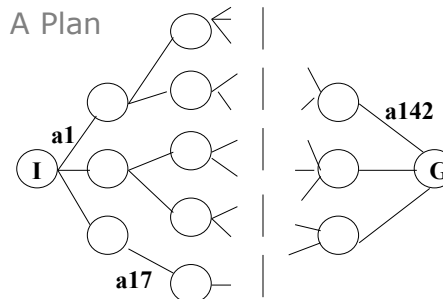
```

Putdown(x)
pre  Holding(x)
del  Holding(x)
add  Clear(x) ∧ OnTable(x) ∧ 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

$$\text{action} : E \rightarrow Ac$$

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

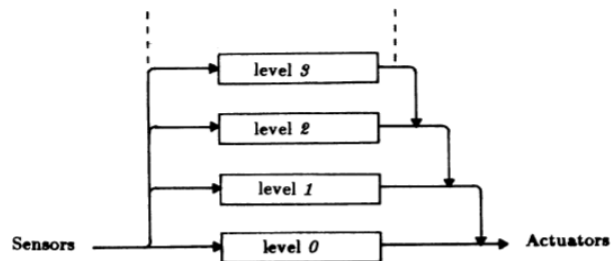


## The Subsumption Architecture

- 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



From Brooks, “A Robust Layered Control System for a Mobile Robot”, 1985

5-31



## 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:  
*if detect an obstacle then change direction* (1)
- Any samples carried by agents are dropped back at the mother-ship:  
*if carrying samples and at the base then drop samples* (2)
- Agents carrying samples will return to the mother-ship:  
*if carrying samples and not at the base then travel up gradient* (3)



## Steels' Mars Explorer Rules

- Agents will collect samples they find:  
*if detect a sample then pick sample up* (4)
- An agent with “nothing better to do” will explore randomly:  
*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.

