Distributed Intelligent Systems

MultiAgent Systems

Ulises Cortés

2018

Origins

- Trends in Computer Science
- Agents and Multiagent Systems
- 2 views of the Field

Computing now-a-days

- **Internet Technology**
  - **IoT, Internet 2.0, Broadband access, exploding usage…**
- **Mobile *Telephony* Technology**
  - **5G, 4G, 3G, iMode, WAP, Wireless PDAs, Bluetooth…**
- **Software Technology**
  - **JavaBeans, Soap, UDDI, JINI…**
- **Web Technology**
  - **XML, RDF, Servlets, JavaBeans, *Semantic Web (OWL)***
- **AI**
  - **Reasoning, Knowledge Representation, Deep Learning, Computational Intelligence, Agents…**

Origins of MAS

- **Five ongoing trends have marked the history of computing [M. Wooldridge]:**
  - **Ubiquity;**
  - **Interconnection;**
  - **Intelligence;**
  - **Delegation; and**
  - **Human-orientation**

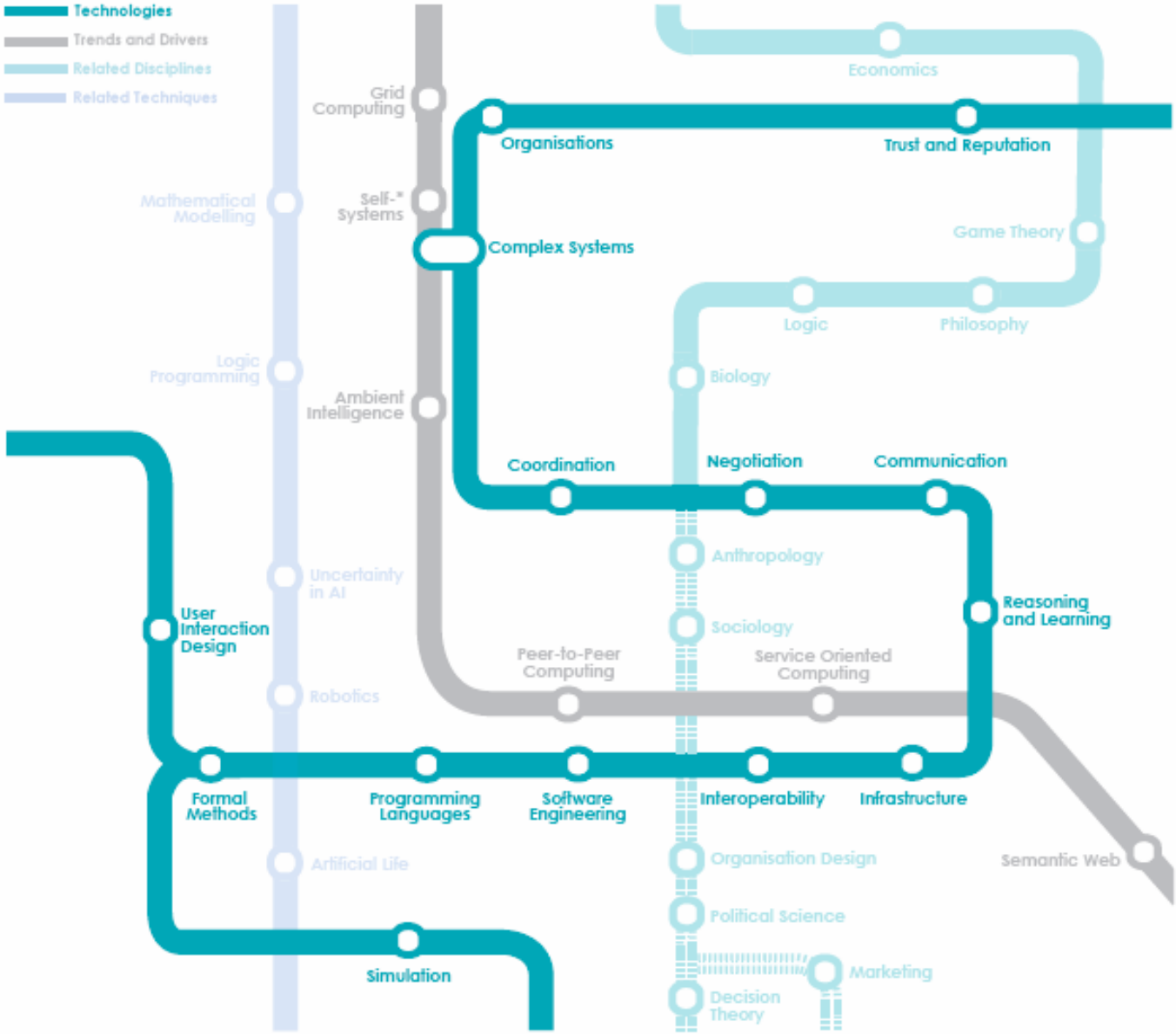
5 trends (1 of 3)

- **Ubiquity**
  - The continual reduction in cost of computing capability has made it possible to introduce processing power into places and devices that would have once been uneconomic
  - As processing capability spreads, computation (and intelligence of a sort) becomes ubiquitous

- **Interconnection**
  - Computer systems today no longer stand alone, but are networked into large distributed systems
  - Since distributed and concurrent systems have become the norm, some researchers are putting forward theoretical models that portray computing as primarily a process of interaction

Agent Technology: Computing as Interaction

A Roadmap for Agent Based Computing



5 trends (2 of 3)

- **Intelligence**
  - The complexity of tasks that we are capable of automating and delegating to computers has grown steadily, to the limits that we can define as *intelligent*.
- **Delegation**
  - Computers are doing more for us – without our intervention
  - We are *giving control* to computers, even in safety critical tasks
- **Human orientation**
  - The movement away from machine-oriented views of programming toward concepts and metaphors that more closely reflect the way we ourselves understand the world
  - Programmers conceptualize and implement software in terms of higher-level – more human-oriented – abstractions

5 trends (3 of 3)

- Delegation and Intelligence imply the need to build computer systems that can act effectively on our behalf
- This implies:
  - The ability of computer systems to act *independently*
  - The ability of computer systems to act in a way that *represents our best interests* while interacting with other humans or systems
- Interconnection and ***Distribution*** have become core motifs in Computer Science
- But ***Interconnection*** and ***Distribution***, coupled with the need for systems to represent our best interests, implies:
  - Systems that can *cooperate* and *reach agreements* (or even *compete*) with other systems that have different interests (much as we do with other people)

Ambient Intelligence

- Aimed at seamless delivery of services and applications.
- Relies on *ubiquitous computing*, *ubiquitous communication* and *intelligent user interfaces*
- ***A Vision***
  - An environment of potentially thousands of embedded and mobile devices (or software components) interacting to support user-centred goals and activity.
  - Suggests a component-oriented view of the world in which the components are independent and distributed.
  - ***Autonomy, distribution, adaptation, responsiveness***, and so on, are key characteristics of these components, and in this sense they share the same characteristics as agents.
- Requires agents to be able to interact with numerous other agents in the environment around them in order to achieve their goals.

Computer Science progression

- These issues were not studied in Computer Science until recently (less than 20 years)
- All of these trends have led to the emergence of a new field in Computer Science: *multiagent systems/computational Intelligence*
- Wooldridge says that programming has progressed through:
  - machine code;
  - assembly language;
  - machine-independent programming languages;
  - sub-routines;
  - procedures & functions;
  - abstract data types;
  - objects;

to ***agents***.

Agents and Multiagent Systems

- An agent is a computer system that is capable of ***independent*** action on behalf of its user or owner (figuring out what needs to be done to satisfy design objectives, rather than constantly being told)
- A multiagent system is one that consists of a number of agents, which ***interact*** with one-another
- In the most general case, agents will be acting on behalf of users with different goals and motivations
- To successfully interact, they will require the ability to ***cooperate***, ***coordinate***, and ***negotiate*** with each other, much as people do

Multiagent Systems (working definition)

- A multiagent system is one that consists of a number of agents, which ***interact*** with one-another
- In the most simple case, all agents are programmed by the same team and they collaborate to complete a task
- In the most general case, agents will be acting on behalf of users with different goals and motivations
- To successfully interact, they will require the ability to ***cooperate***, ***coordinate***, and ***negotiate*** with each other, much as people do

Agents and Multiagent Systems

- **Building Agents, we address questions such as:**
  - How do you state your preferences to your agent?
  - How can your agent compare different deals from different vendors? What if there are many different parameters?
  - What algorithms can your agent use to negotiate with other agents (to make sure you get a good deal)?

- **In Multiagent Systems, we address questions such as:**
  - How can cooperation emerge in societies of self-interested agents?
  - What kinds of languages can agents use to communicate?
  - How can self-interested agents recognize conflict, and how can they (nevertheless) reach agreement?
  - How can autonomous agents coordinate their activities so as to cooperatively achieve goals?

Agent Design, Society Design

- **Two key problems:**
  - **How** do we build agents capable of independent, autonomous action, so that they can successfully carry out tasks we delegate to them?
  - *How* do we build agents that are capable of interacting (*cooperating, coordinating, negotiating*) with other agents in order to successfully carry out those delegated tasks, especially when the other agents cannot be assumed to share the same interests/goals?
    - The first problem is *agent design*
    - The second is *society design* (micro/macro)

Multiagent Systems is Interdisciplinary

- The field of Multiagent Systems is influenced and inspired by many other fields:
  - Philosophy
  - Logic
  - Game Theory
  - Economics
  - Social Sciences
  - Ecology
- This can be both a strength (infusing well-founded methodologies into the field) and a weakness (there are many different views as to what the field is about)

2 Views of the Field

- ***Agents as a paradigm for software engineering***: Software engineers have derived a progressively better understanding of the characteristics of complexity in software. It is now widely recognized that *interaction* is probably the most important single characteristic of complex software

- Over the last two decades, a major Computer Science research topic has been the development of tools and techniques to model, understand, and implement systems in which *interaction* is the *norm*

2 Views of the Field

- ***Agents as a tool for understanding human societies*:**
  Multiagent systems provide a novel new tool for simulating societies, which may help shed some light on various kinds of social processes.

- This has analogies with the interest in *theories of the mind* explored by some artificial intelligence researchers and cognitive scientists

Standards: FIPA (*www.fipa.org*)

- **International Agent Standard**
  - Started in 1996 to provide agent technology specifications.
  - Part of IEEE (since 2005) as 11th standards committee.
- **Includes standards for**
  - Communication: Agent Communication Languages, Content Languages, Semantic Framework
  - Infrastructure: directories, message transport, naming, etc…
- **Recent trends**
  - Moved toward web technology (OWL, XML, RDF, HTTP)
  - Plug and Play architectures
  - Moves for Java standard
- **Next phase**
  - Verification
  - Significant take-up
  - Demonstration of Value

Hot topic: Open Service Environments

- Explosion of Agent technology with new uses for Open Service Environments
- Automation of Services
  - Proactive, responsible, intelligent, peer to peer
- Dynamic Composition of Services
  - Automated discovery, automated coordination, *Just in Time* Enterprises, Virtual Companies
- Semantics
  - HTML won't do anymore
  - Semantic Web
  - Service-level semantics
  - Semantics for E-commerce
  - Service-Oriented Architectures' frameworks

Agent types and architectures

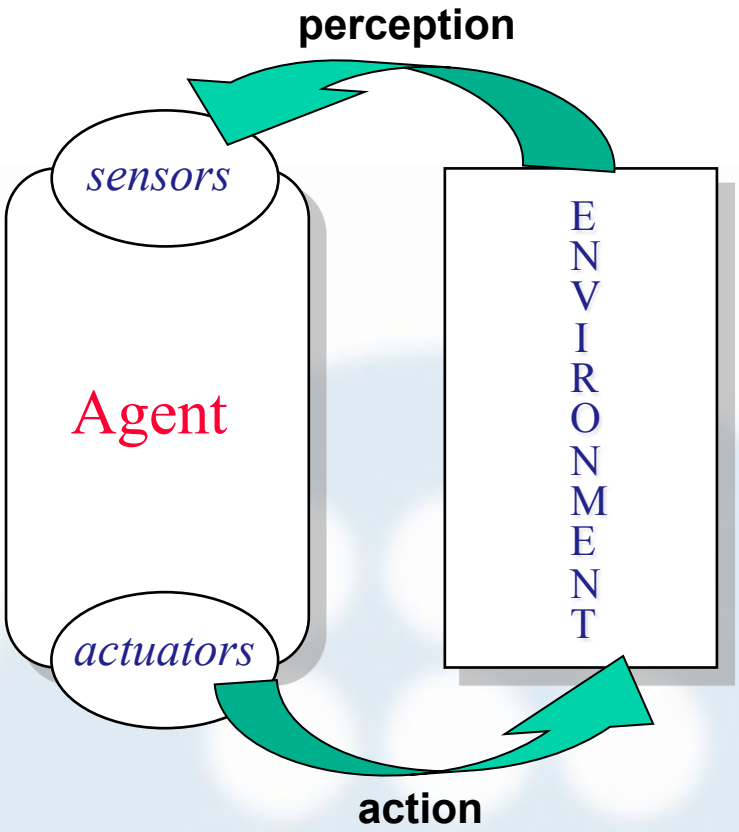
- **Agent properties**
- **Environment properties**
- **Agent types**
- **Abstract architecture**

Agent Properties

Autonomy

An agent *is a computer system capable of* autonomous action *in some environment in order to meet its* design objectives

Usually the environment is *complex* and *dynamic*, and agents should interact with it in real time.



Main property:

Autonomy capable of acting independently, exhibiting control over their ***internal state***

Agent Definitions

- Autonomous agents are computational systems that inhabit some complex dynamic environment, sense and act autonomously in this environment, and by doing so realize a set of goals or tasks for which they are designed.

- An autonomous agent is a system situated within and a part of an environment that senses that environment and acts on it, over time, in pursuit of its own agenda and so as to affect what it senses in the future

Agent Properties

Autonomy, Flexibility

Trivial (non-interesting) agents: ***thermostat***

Definition 3: *An* intelligent agent *is a computer system capable of* flexible *autonomous action in some environment*

By *flexible*, we mean:

- *reactive* (response capability)
- *pro-active* (taking initiative)
- *social* (interacting with others)

Agent Properties: Reactivity

- If a program's environment is guaranteed to be fixed, the program need never worry about its own success or failure – program just executes blindly
  - Example of fixed environment: compiler
- The real world is not like that: things change, information is incomplete. Many (most?) interesting environments are *dynamic*

Agent Properties: Reactivity

- Software is hard to build for dynamic domains: program must take into account possibility of failure – *ask itself whether it is worth executing!*

- A *reactive* system is one that maintains an ongoing interaction with its environment, and responds to changes that occur in it (in time for the response to be useful)

Agent Properties: 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 behavior*
- ***Pro-activeness*** = generating and attempting to achieve goals; not driven solely by events; taking the initiative
- Recognizing opportunities

Agent Properties: Social Ability

- 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 with the *cooperation* of others
- Similarly for many computer environments: witness the Internet
- *Social ability* in agents is the ability to interact with other agents (and possibly humans) via some kind of *agent-communication language*, and perhaps cooperate with others

Agent Properties

Balancing Reactive and Goal-Oriented Behavior

- We want our agents to be reactive, responding to changing conditions in an appropriate (timely) fashion
- We want our agents to systematically work towards long-term goals
- These two considerations can be at odds with one another

Reactivy vs. Deliberation balance

- Designing an agent that can balance reactivity and deliberation (reason about long term goals) remains an open research problem

Other Agent Properties
(desireable, not mandatory)

- ***mobility***
  - the ability of an agent to move around an electronic network
- ***veracity***
  - an agent will not knowingly communicate false information
- ***benevolence***
  - agents do not have conflicting goals, and that every agent will therefore always try to do what is asked of it
- ***rationality***
  - **agent will act in order to achieve its goals, and will not act in such a way as to prevent its goals being achieved — at least insofar as its beliefs permit**
- ***learning/adaption***
  - agents improve performance over time

Environment properties

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
- 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

Environment properties

Deterministic vs. non-deterministic

- 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
- The physical world can to all intents and purposes be regarded as *non-deterministic*
- Non-deterministic environments present greater problems for the agent designer

Environment properties

Episodic vs. non-episodic

- In an episodic environment, the performance of an agent is dependent on a number of discrete episodes, with no link between an agent's performance 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

Environment properties

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
- Other processes can interfere with the agent's actions (as in concurrent systems theory)
- The physical world is a highly dynamic environment

Environment properties

Discrete vs. continuous

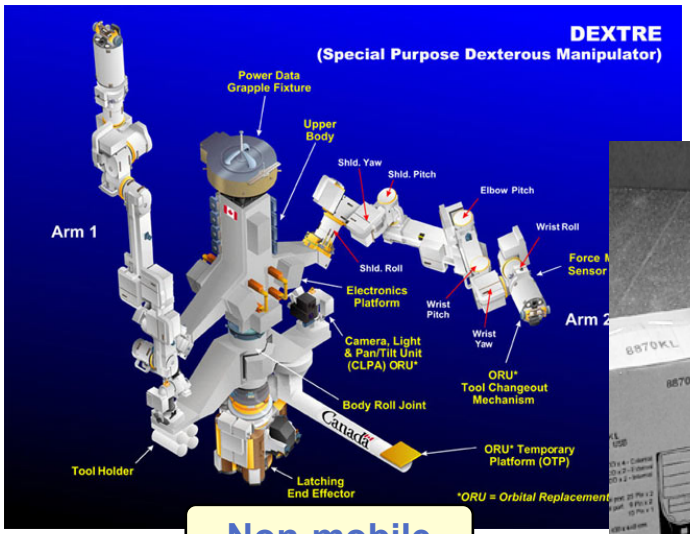
- An environment is discrete if there are a fixed, finite number of actions and percepts in it
- Russell and Norvig give a chess game as an example of a discrete environment, and taxi driving as an example of a continuous one
- Continuous environments have a certain level of mismatch with computer systems
- Discrete environments could *in principle* be handled by a kind of lookup table

Agent types

Physical (embodied) Agents vs. Software Agents

- Software agents' environment is a virtual one
  - Single machine, intranet, internet
  - Interact with other software agents, with software modules, services
  - Interact with humans through human interfaces
- Physical agents or embodied agents
  - Interact with real world (sensors, actuators connected to real world)
  - Problems of perception and action
  - Best known example: *Robots*.

Agent types
Robots



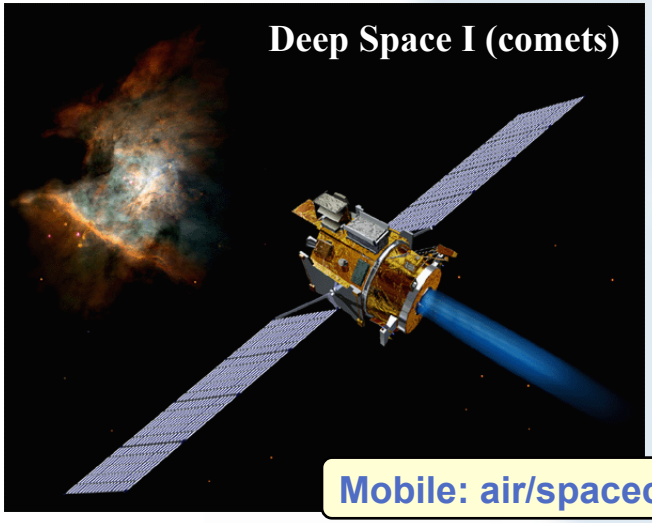
Non-mobile

Mobile: weeled

Mobile: legged

SONY aibo

https://www.kemlg.upc.edu

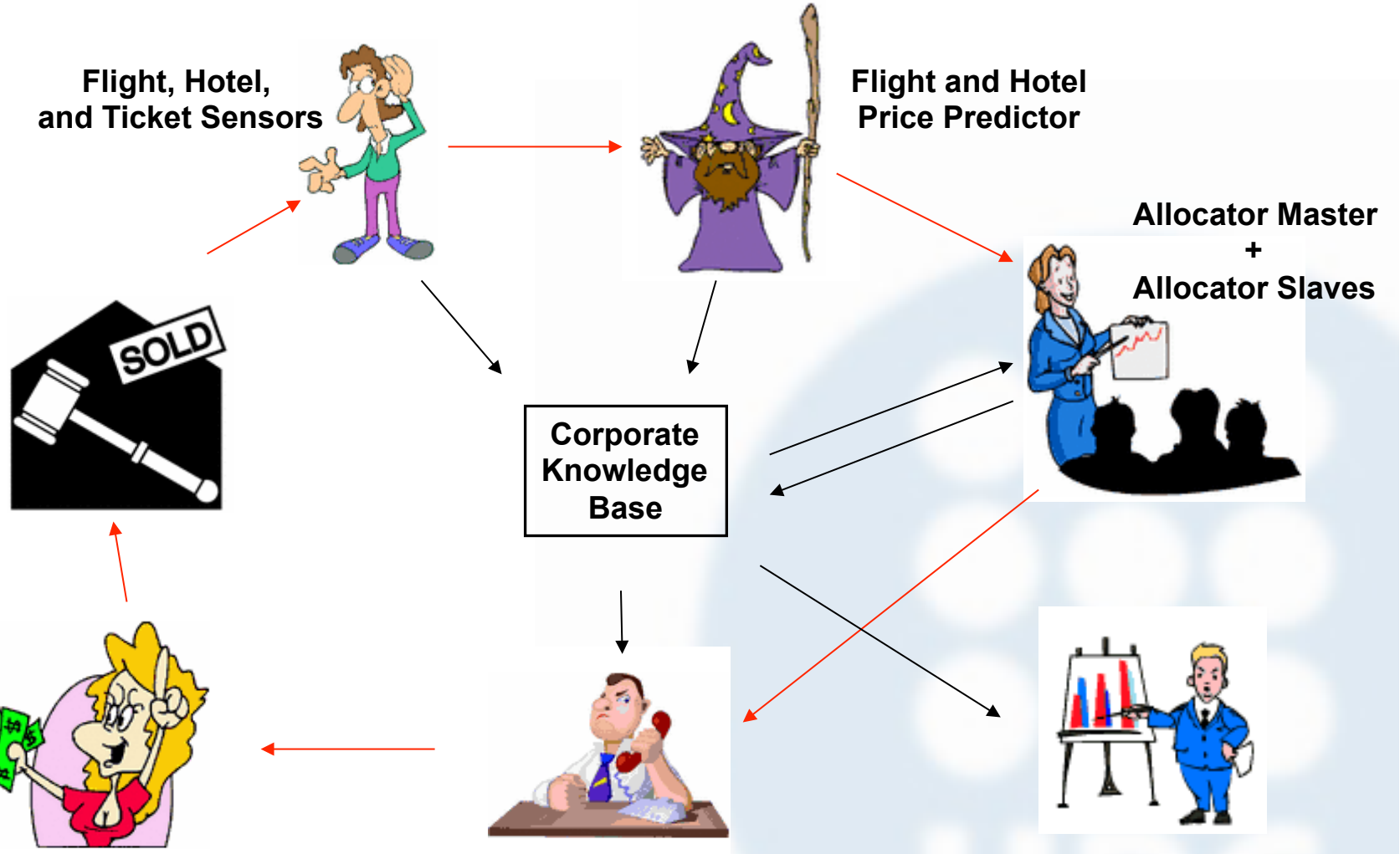


Mobile: air/spacecrafts

Agent Types
Software agents

- **Internet agents** search and information extraction/ management from Internet
- **Collaborative agents** they coordinate with other agents to solve a common task
  - To solve problems too complex for a single agent
  - To solve problems distributed in nature
  - To interconnect already existing, heterogeneous systems (→ **Agentification**)
- **Interface agents** they collaborate with a human user to solve a task, or to act on behalf of the user.
- **Mobile SW agents** they can move from one computer to another

A Multi-Agent System (Trading Agent Competition)



Agent types
Internal architecture

- **Purely Reactive Agents** (with no internal state)
- **Reactive Agents with internal state**
- **Delliberative Agents** (goal-oriented behaviour)
- **Hybrid Agents** (combine reactive and delliberative behaviour)

Agent architectures

- **Abstract architecture for agents**
- **Architectures for Multiagent systems**

Agent Architectures

- An agent architecture proposes a particular methodology for building an autonomous agent.
- How the construction of the agent can be decomposed into the construction of a set of component modules
- How these modules should be made to *interact*
- These two aspects define how the sensor data and the current internal state of the agent determine the *actions* (effector outputs) and future internal state of the agent

Purely Reactive Agents

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

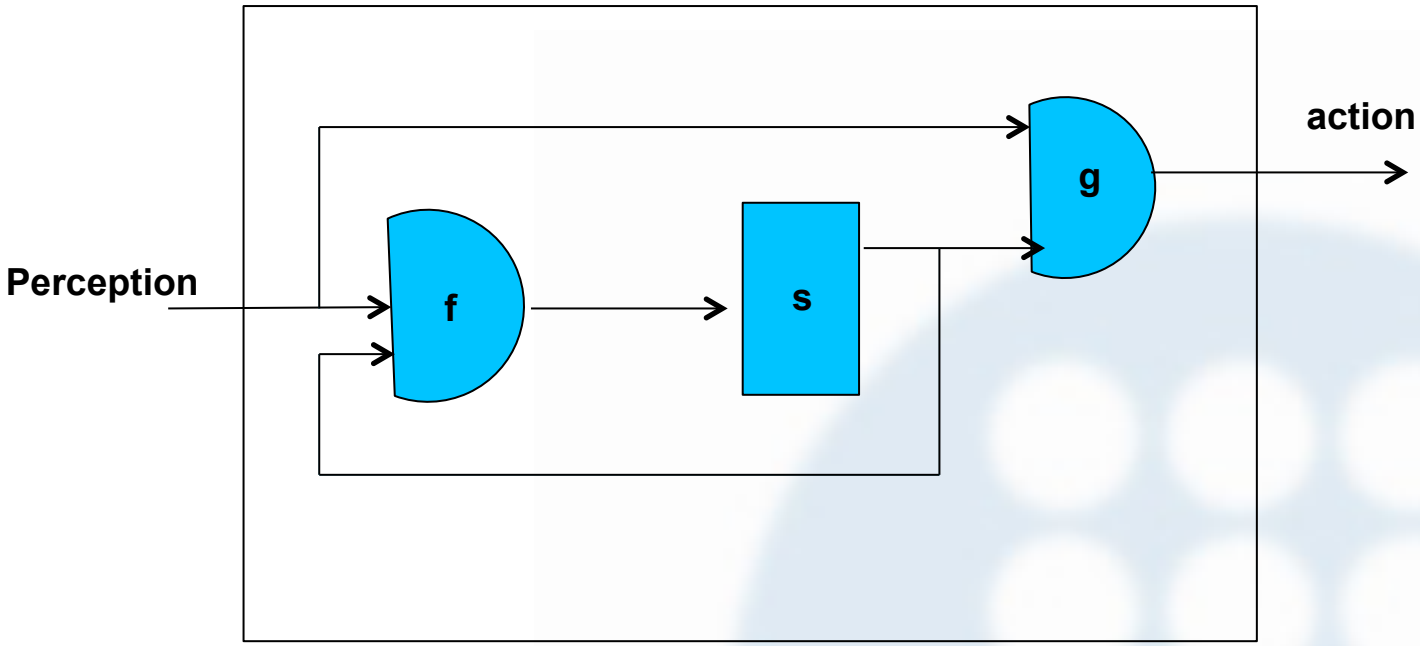
$$action : E \rightarrow Ac$$

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

Situated Automata components

- An agent is specified in terms of two components: *perception* and *action*
- Two programs are then used to synthesize agents
  - RULER is used to specify the perception component of an agent
  - GAPPS is used to specify the action component

From perception to action



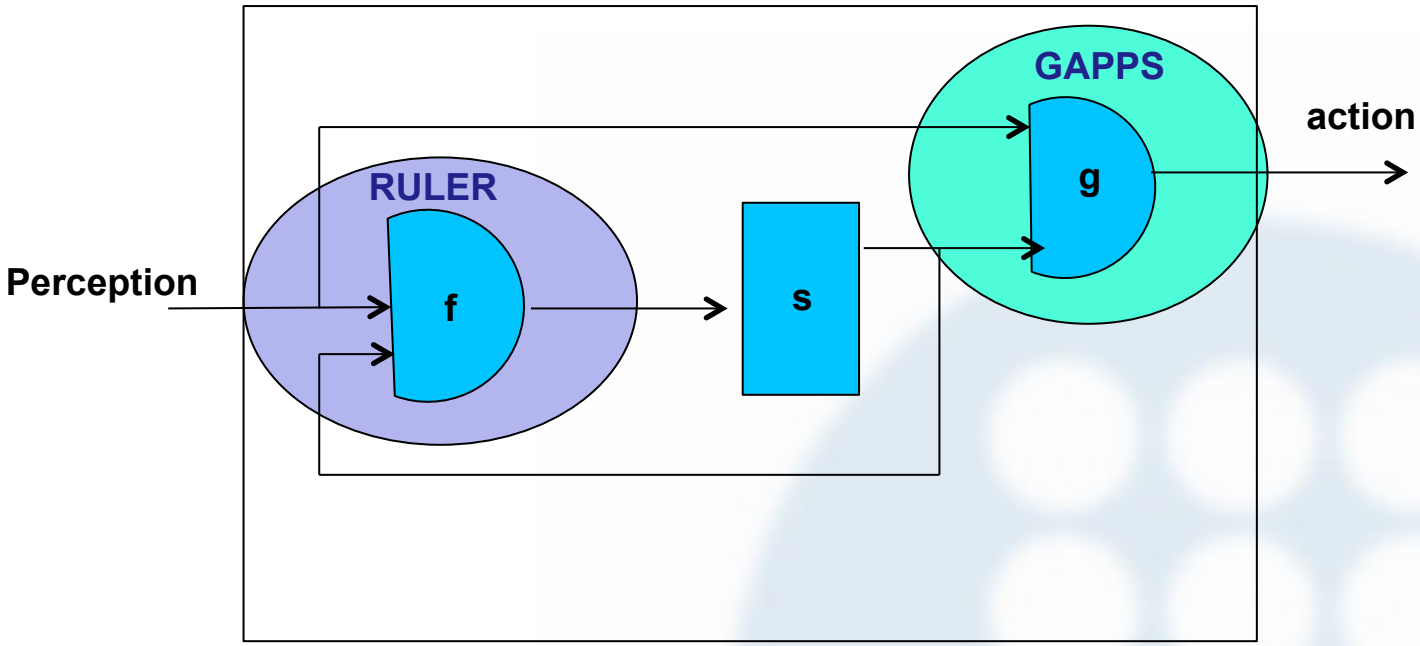
f = State update function

s = Internal state

g = Output function

From Rosenschein and Kaelbling, "*A Situated View of Representation and Control*", 1994

From perception to action



f = State update function

s = Internal state

g = Output function

RULER – Situated Automata

- RULER takes as its input three components
  - The semantics of the agent's inputs (‘whenever bit 1 is on, it is raining’)
  - A set of static facts (‘whenever it is raining, the ground is wet’)
  - A specification of the state transitions of the world (‘if the ground is wet, it stays wet until the sun comes out’).
- The programmer then specifies the desired semantics for the output (‘if this bit is on, the ground is wet’)
- The compiler designs a circuit whose output will have the correct semantics

GAPPS – Situated Automata

The GAPPS program takes as its input

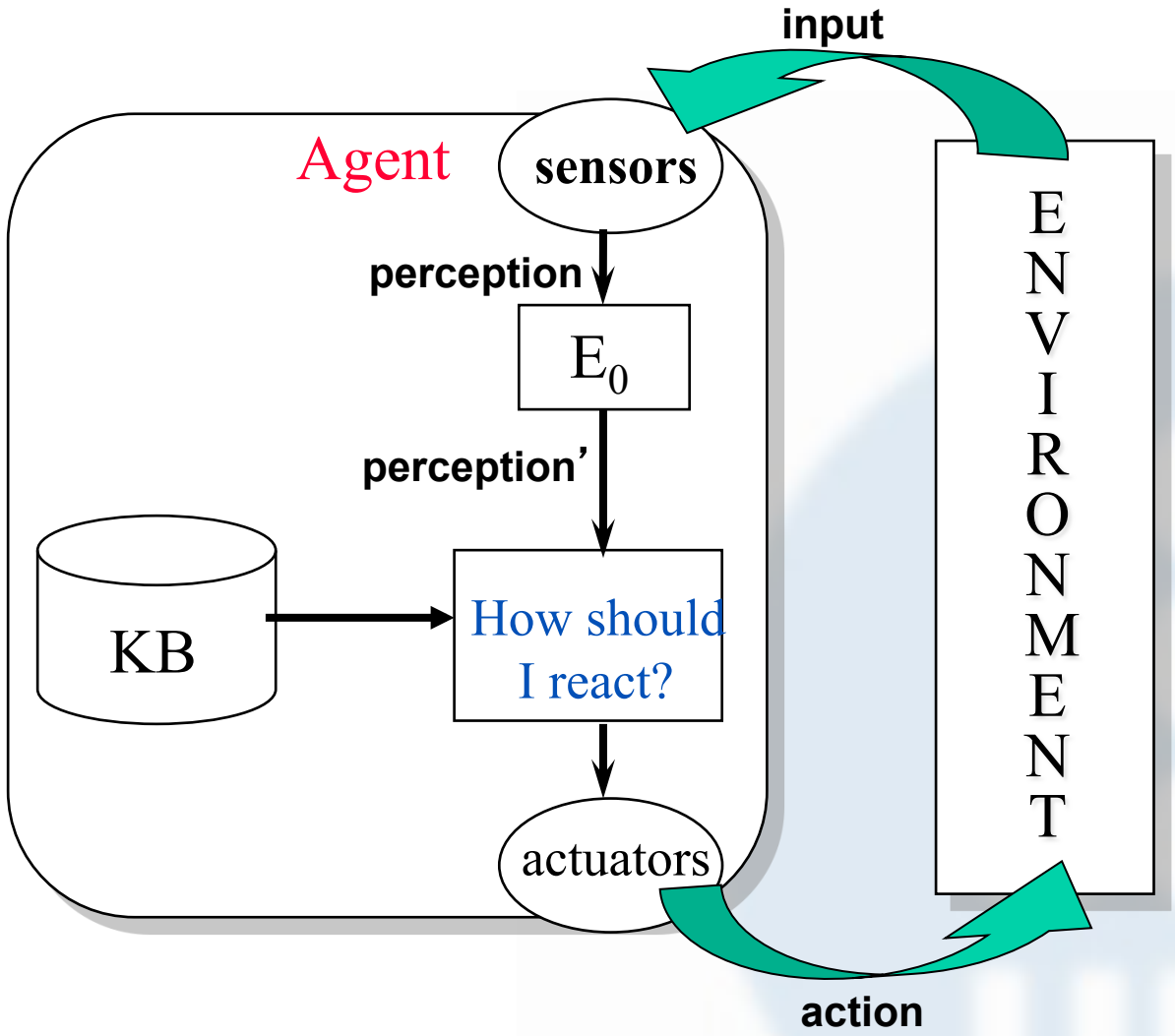
- A set of *goal reduction rules*,
  - Rules that encode information about *how* goals can be achieved in a given state
- A top level goal

- Then it generates a program that can be translated into a digital circuit in order to realize the goal
- The generated circuit does not represent or manipulate symbolic expressions; all symbolic manipulation is done at compile time

Advantages of Reactive Agents

- **Simplicity** of individual agents
- **Flexibility**, adaptability
  - Ideal in very dynamic and unpredictable environments
- Computational tractability
  - Avoiding complex planning/reasoning procedures
  - Avoiding continuous model update
- **Robustness** against failure
  - No central planning component (*e.g.* ant colony)
- **Elegance**

Purely Reactive Agents

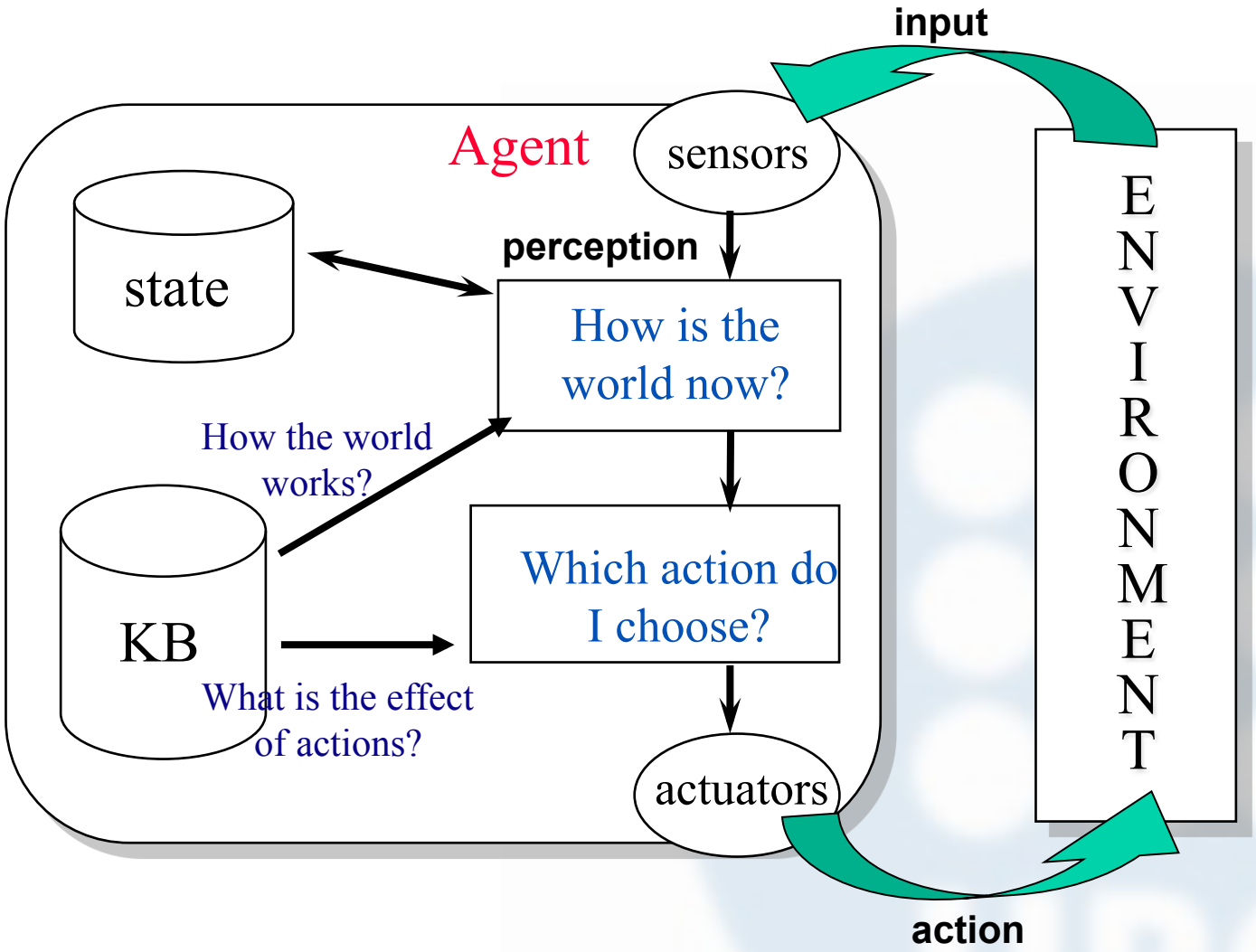


Purely Reactive Agents

function pra(percept) **returns** (action)
static rules

state ← **interpret-input**(percept)
rule ← **rule-match**(state,rules)
action ← **rule-action**[rule]
return action

Reactive Agents with internal state



Reactive Agents with internal state

Function reactive-agent-with-state(percept) **returns** action
Static state ;a world description
rules ;a set of, *e.g.*, *if-then* rules

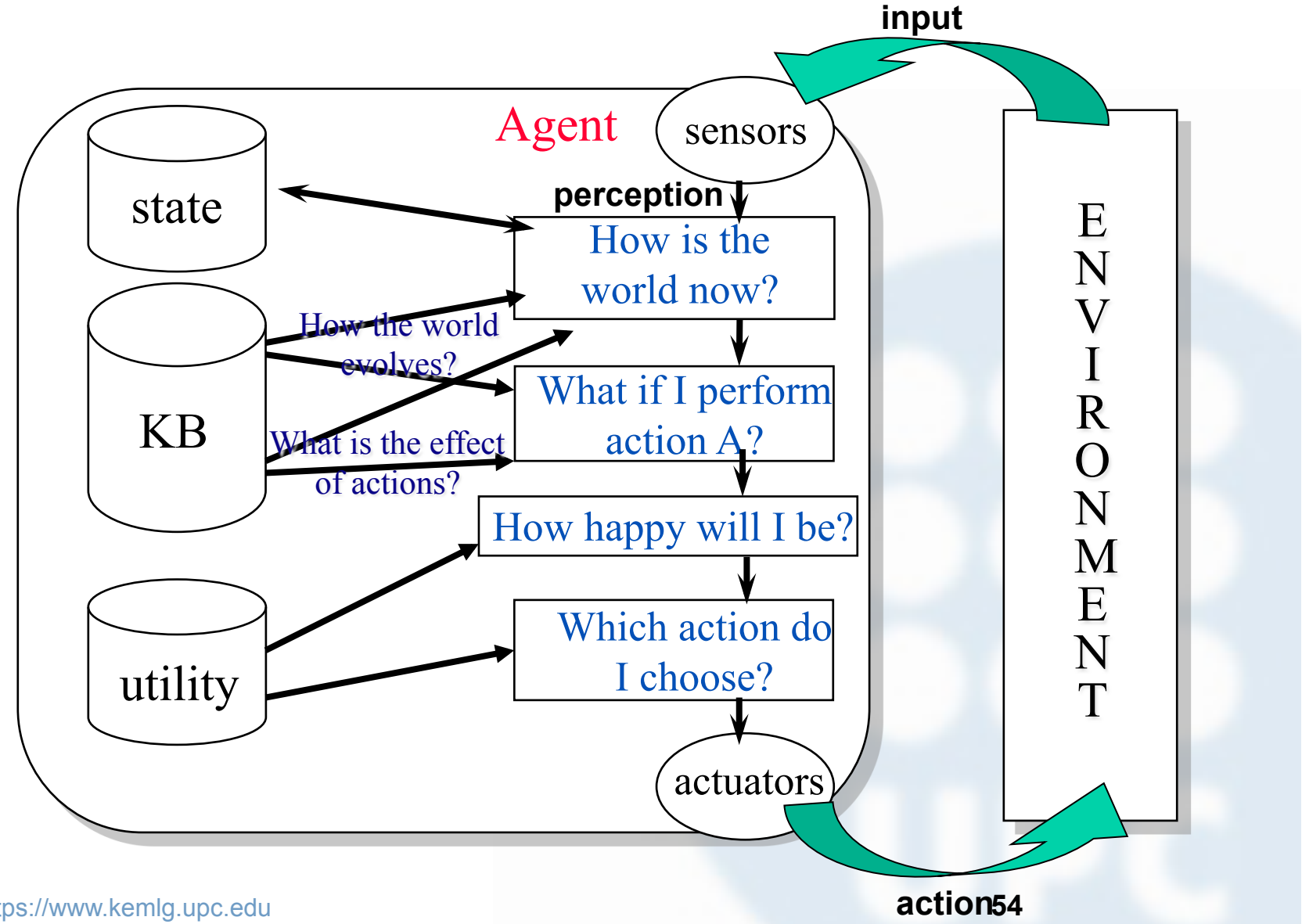
state ⟵ update-state(state,percept)
rule ⟵ rule-match(state,rules)
action ⟵ rule-action[rule]
state ⟵ update-state(state,action)

return ⟵ action

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
- One possibility: associate *utilities* with individual states — the task of the agent is then to bring about states that maximize utility

Delliberative Agents (with expected utilities)



Utility Functions over States

- A task specification is a function

$u : E \rightarrow \#$

which associates a real number with every environment state

- But what is the value of a *run*…
  - minimum utility of state on run?
  - maximum utility of state on run?
  - sum of utilities of states on run?
  - average?

- **Disadvantage**: difficult to specify a *long term* view when assigning utilities to individual states (One possibility: a *discount* for states later on.)

Utilities over Runs

- Another possibility: assigns a utility not to individual states, but to runs themselves:

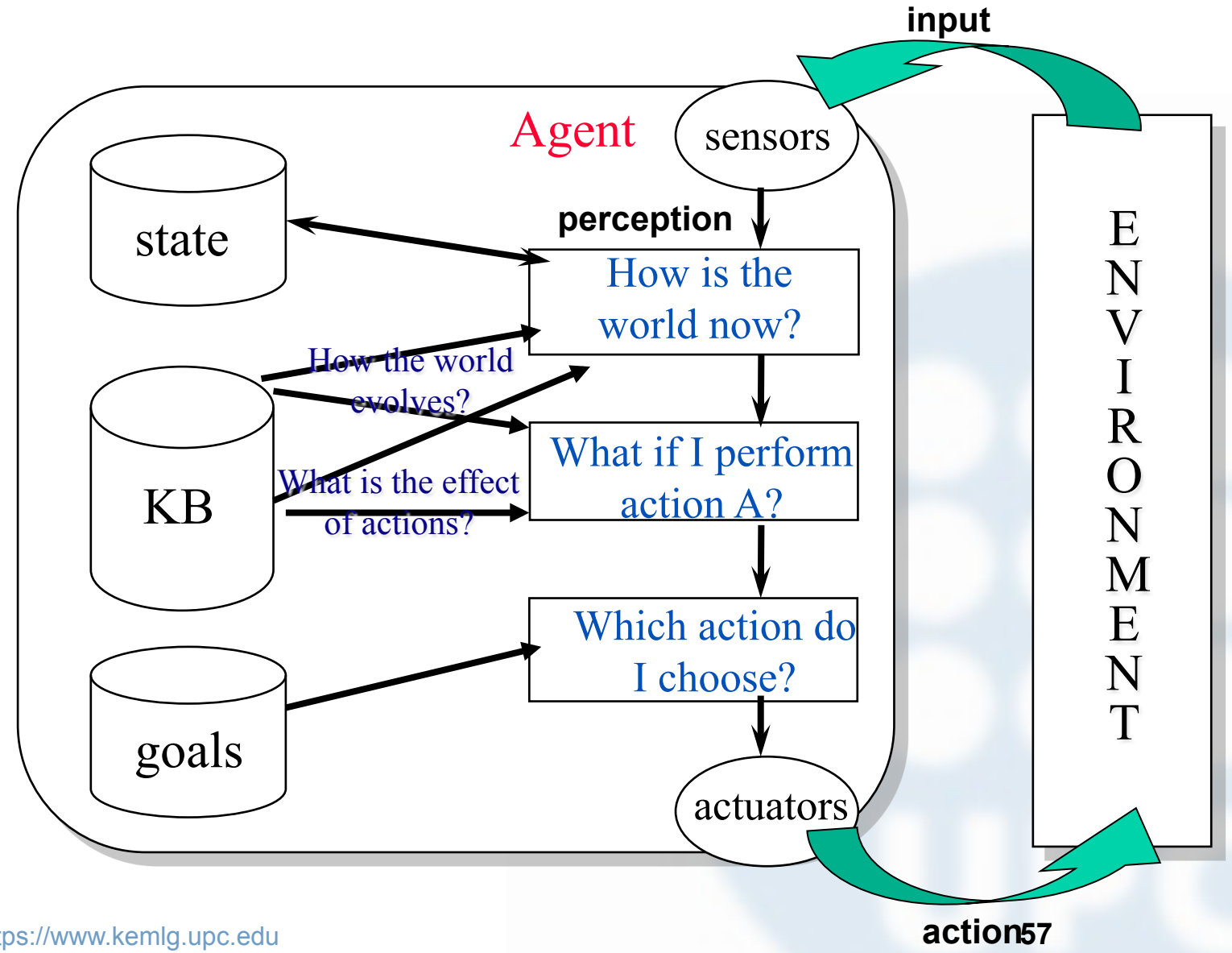
$$u : \mathrm{R} \rightarrow \#$$

- Such an approach takes an inherently *long term* view

- Other variations: incorporate probabilities of different states emerging

- Difficulties with utility-based approaches:
  - where do the numbers come from?
  - we (normally?) do not think in terms of utilities!
  - hard to formulate tasks in these terms

Delliberative Agents (with explicit goals)



Delliberative Agents (with explicit goals)

Function reactive-agent-with-goals(percept) **returns** action
Static state ; a world description
rules ;a set of, e.g., *if-then* rules
goals ;a list of *goal states*

state ← **update-state(state,percept)**
appliable-rules ← **rule-match(state,rules)**
possible-actions ← **rule-action[rule]**
action ← **goal-oriented-selection[possible-actions]**
state ← **update-state(state,action)**

return **action**

Agents' interactions

- **Interaction between agents is unavoidable**
  - To achieve own goals,
  - To manage interdepencies
- **It should occur at *Knowledge-level***
  - *Which* goals?, *When*?, *Who* executes what?
- **Flexibility to start and to give answers.**
  - Synchronic, programs, *etc*

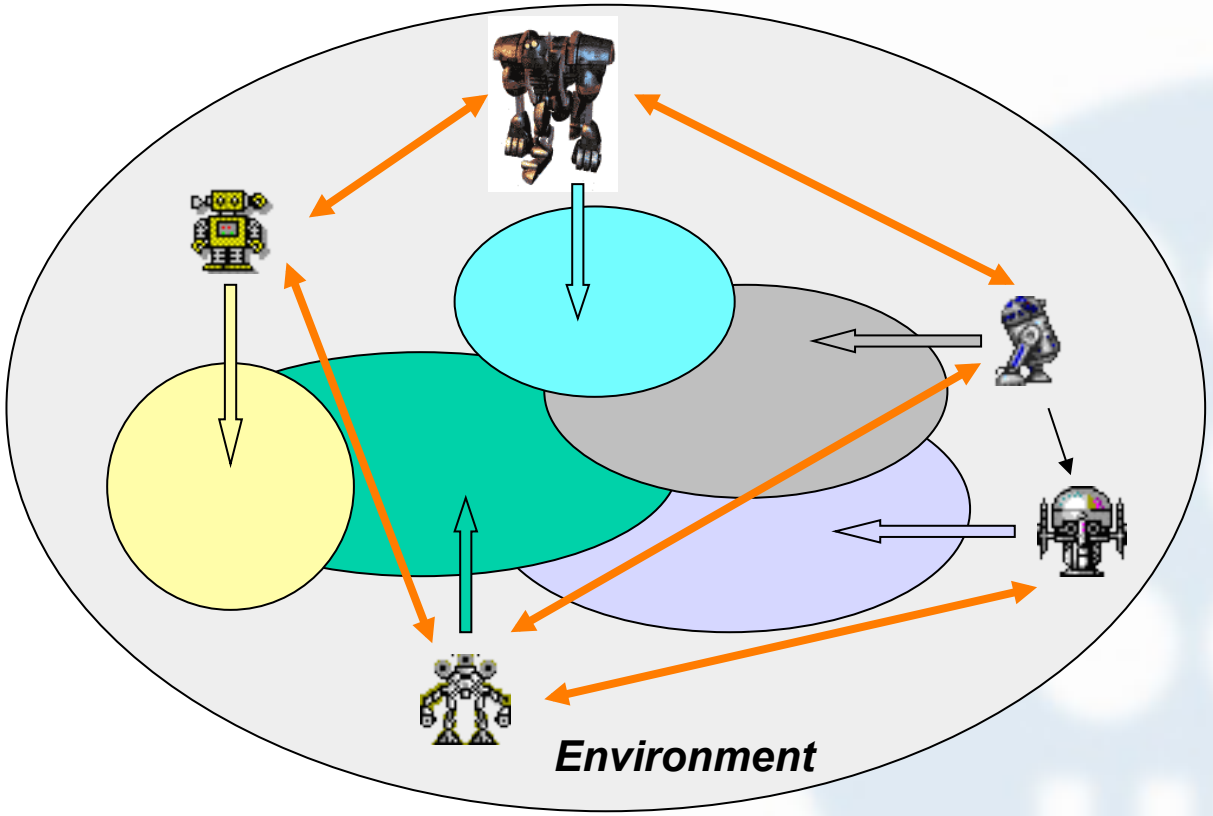
This implies a radical change in the way programs usually interact

Multiagent Systems' architecture

- Agents in a multiagent system tend to interact through a middleware layer
- This middleware provides connectivity between agents, solving low-level connectivity issues
  - Communication methods
- Sometimes this middleware is called *agent platform*

Multiagent Systems

Many entities (*agents*) in a common environment



Influence area

Interactions

MAS - many agents in the same environment

Interactions among agents

- high-level interactions

Interactions for
- coordination
- communication
- organization

❑ Coordination

➔ collectively motivated / interested

➔ self interested

- own goals / indifferent
- own goals / competition / competing for the same resources
- own goals / competition / contradictory goals
- own goals / coalitions

MAS - many agents in the same environment

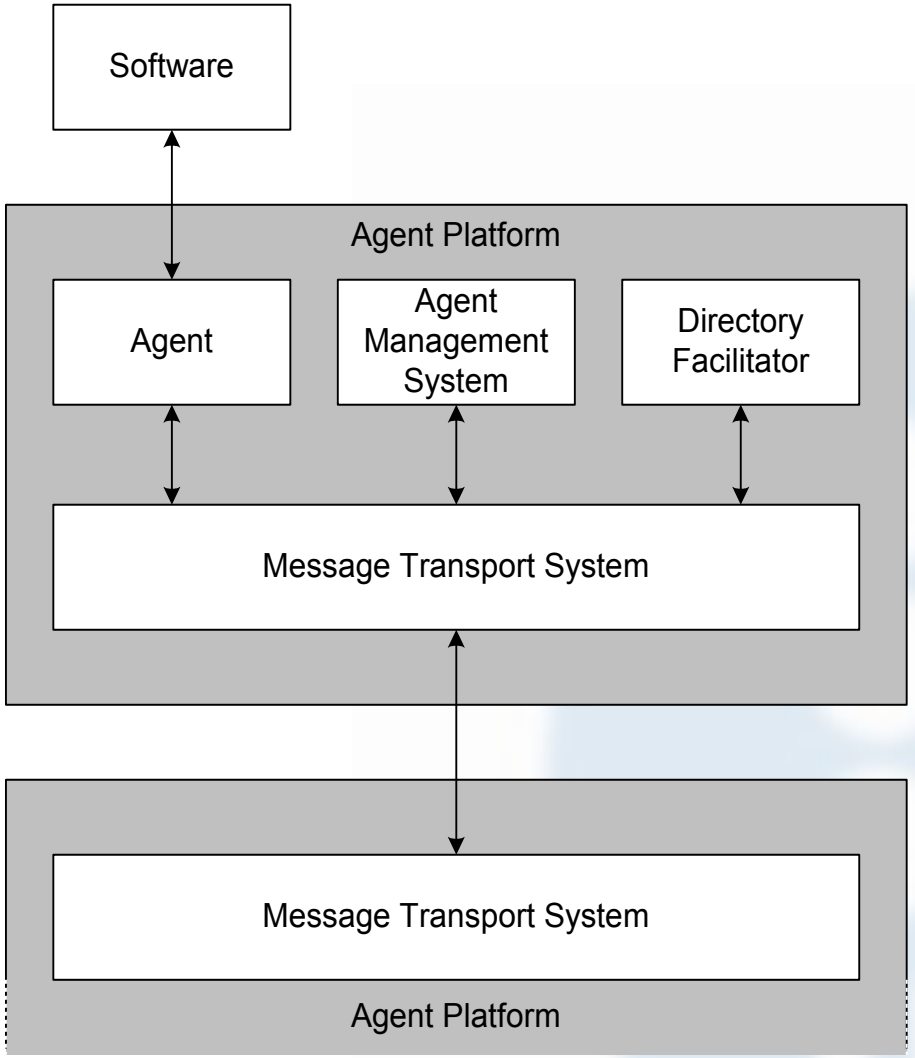
- **Communication**
  - ➔ communication protocol
  - ➔ communication language
  - negotiation to reach agreement
  - Ontology
- **Organizational structures**
  - ➔ centralized vs decentralized
  - ➔ hierarchical/ markets

cognitive agent approach

Communication methods

- *Blackboard systems*
  - Agents communicate information through a common data structure, accessible by everybody
  - Problem: if there is no middleware to provide some concurrency, it tends to become a bottleneck.
- *Message passing*
  - Agents communicate directly by means of messages
  - The agent platform usually acts as message router
  - Common communication language (*e.g.* FIPA-ACL)
  - Common communication protocols (message format, steps in a communication)

FIPA Architecture for Agent Platforms



Components of an Agent Platform

- **Agent:** a program providing a list of services
- **Directory Facilitator (DF)** is an agent which provides a Yellow Pages service within the platform (knows the services that agents within the platform provide)
  - *register, deregister, modify, search*
- **Agent Management System (AMS)** is an agent controlling access and usage of the agent platform. It knows the platform and agents' addresses and provides a White Pages service (knows the routing addresses for agents within and in other platforms)
- **Message Transport Service (MTS)** is used to enable communication between agents in different platforms.

Agent Platform tasks

- Suspend temporally an agent execution
- Stop an agent execution
- Resume/continue agent execution
- Start an agent
- Platform resource management

How do agents acquire intelligence?

Cognitive agents

The model of human intelligence and human perspective of the world ➔ characterise an intelligent agent using symbolic representations and ***mentalistic notions:***

- **knowledge** – Mary knows humans are mortal
- **beliefs** - Mary took his umbrella because she believed it was going to rain
- **desires, goals** - Mary wants to possess a PhD
- **intentions** - Mary intends to work hard in order to have a PhD
- **choices** - Mary decided to apply for a PhD
- **commitments** - Mary will not stop working until getting his PhD
- **obligations** - Mary has to work to make a living

(Shoham, 1993)

Premises

Such a mentalistic or intentional view of agents - a kind of *folk psychology* - is not just another invention of computer scientists but is a useful paradigm for describing complex distributed systems.

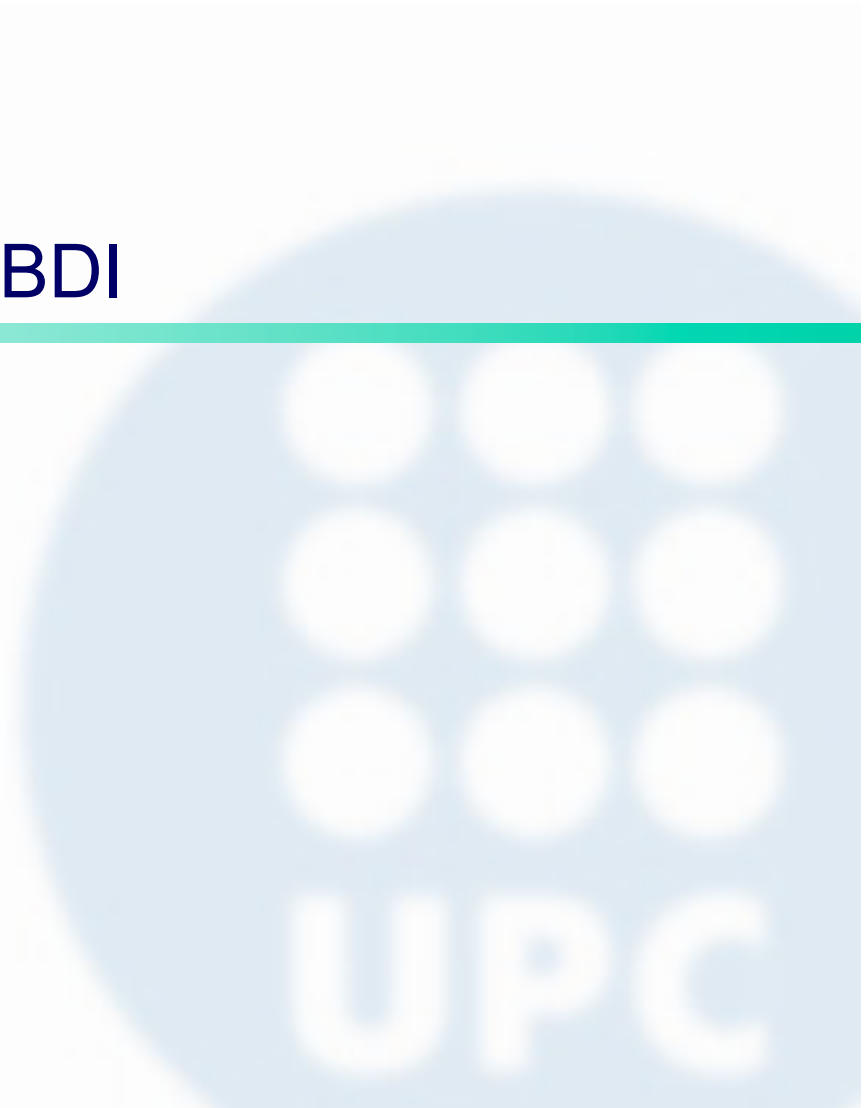
The complexity of such a system or the fact that we can not know or predict the internal structure of all components seems to imply that we must rely on animistic, intentional explanation of system functioning and behavior.

Is this the only way agents can acquire intelligence?

Reactive agents

- Simple processing units that perceive and react to changes in their environment.
- Do not have a symbolic representation of the world and do not use complex symbolic reasoning.
- The advocates of reactive agent systems claims that intelligence is not a property of the active entity but it is distributed in the system, and steams as the result of the interaction between the many entities of the distributed structure and the environment.

BDI



Practical Reasoning Agents

Beliefs and preferences: towards BDI Agents

One can define an agent's solving problem mechanism in terms of its beliefs, how it proceeds to achieve its goals, its capabilities and relate its behaviour with its individual characteristics and attributes as its *preferences*.

[Haddadi96]

BDI Agents

Agent = Intentional System

Intentionality is a property of many *mental states* and *Events* which can then be mapped into *states of affairs*.

Example: If I have desire, this (desire) should be of doing something or willing that something happens (in the world) in a way

***Beliefs* and *Desires* are intentional.**

BDI Agents
Agent = Intentional System

Intentionality **refers to the agent's intentions as** ***rationally***
implied by its beliefs (about the world).

Example: If I assume that $Agent_i$ **believes** $p,q,r\ldots$
I assume that the agent believes all that it can be deduced from propositions $p,q,r\ldots$

[Dennett,81]

BDI Agents

Agent = Intentional System

- **Intentions direct goal-based *reasoning***
- **Intentions constraint future *deliberation***
- **Intentions are *persistent***
- **Intentions are *beliefs* upon which pragmatic reasoning is built.**

BDI Agents
Intentional Attitudes

- **Intentional Attitudes play different roles in the Agent's *personality* definition.**
  - ***Cognitives* refer to epistemological aspects**
  - ***Volitives* refer to action and control. These denote an attempt to perform an action (Conative).**
  - ***Affectives* refer to the Agent's dynamics.**

BDI Agents
Intentional Attitudes (1)

Cognitives	**Beliefs, Knowledge, Recognition**
Volitives	**Intentions, Plan, Compromises**
Affectives	**Goal, Desire, Preferences**

BDI Agents
Intentional Attitudes (2)

Informatives	**Beliefs, Knowledge, Recognition**
Motivation	**Intention, Choice, Plan, Goal, Desire, Compromise, Preference, (whish, want)**
Social	**Obligation, Permission**

BDI Agents
BDI Architecture

- **BDI = Belief, Desires and Intentions**
- **BDI architectures are aimed to model the rational or the intentional *agenciality***
- **Symbols representing the world do correspond with *mental attitudes***

BDI Agents
BDI Architecture

- **Beliefs**: Are used to model the state of the world
- **Desires**: Allow the selection of possible states of the world.
- **Intentions**: Are compromises to achieve a given state.

BDI Agents
Knowledge and Beliefs

The problem is that intentional notions --such as beliefs and desires-- are referentially opaque, in that they set up opaque contexts in which the standard substitution rules of first-order logic do not apply. In classical (propositional and first-order) logic, the denotation, or semantic value, of an expression is dependent solely on the denotation of its sub-expressions. For example, denotation of the propositional logic formula p∧q is a function of the truth-values of p and q. In contrast, the intentional notions are not truth functional.
It is surely not the case that the truth-value of the sentence: "Jaime believes p" is dependent solely on the truth-value of p.

[Wooldrige, 1992]

BDI Agents

Possible World Semantics and Modal Logic

- **The language to express the semantics of a possible world is the modal logic.**
- **Modal logic was developed to formalize expressions (arguments) that include *necessity* and *possibility*.**
- **A *necessary proposition* is a truthful proposition that cannot be false.**
- **A *possible proposition* is one that can be true.**
- ***w.r.t* the possible worlds a *necessary proposition* is that that is true in all the possible worlds and a *possible proposition* is that that at least is true in one world.**

BDI Agents

What is a Modality?

- **A modality is an operator that if applied to a formula produces a new formula with a new meaning.**
- **For example: the operator □ applied to the formula Ψ produces a new formula □Ψ**
- **The only difference with the classic logic is that truth value of □Ψ is only determined by the value of Ψ.**
- **Modalities do not maintain the truthness.**

BDI Agents

Examples of unary modal operators

- $\Box\Psi$ Ψ is necessary
- $\Diamond\Psi$ Ψ is possible
- $F\Psi$ At some point in the future Ψ will be true
- $P\Psi$ At some point in the past Ψ was true
- $B_i\Psi$ Agent i believes Ψ
- $K_i\Psi$ Agent i knows Ψ
- $O_i\Psi$ Agent i is obliged to Ψ

BDI Agents

Examples of unary modal operators

• $\Box\Psi$	Ψ is necessary	• Modal Logic
• $\Diamond\Psi$	Ψ is possible	• Modal Logic
• FΨ	At some point in the future Ψ will be true	• Temporal logic
• PΨ	At some point in the past Ψ was true	• Temporal logic
• BiΨ	Agent i believes Ψ	• Doxastic Logic
• KiΨ	Agent i knows Ψ	• Doxastic Logic
• OiΨ	Agent i is obliged to Ψ	• Deontic Logic

BDI Agents
Modal Logic

The syntax of Modal logic is the one of Classic Logic with the addition of two modal operators $\Box$ necessity and $\Diamond$ possibility.

$$\Diamond\varphi = \neg\Box\neg\varphi$$

The semantic of ML is defined as a model $M=\langle W,R,\pi \rangle$

- **W is a set of possible worlds**
- **R is a binary function, called accessibility function**
- **π is an assignation function that determines for each $w \in W$, the true values of the propositions in W**

It can be seen as a graph (W, R) with a truth value assignation function π that indicates which propositional functions are true and in which node

BDI Agents
Modal Logic

In a Modal Logic all *valid formulas* are interpreted according to a pair $\langle M, w \rangle$ using the satisfaction function ($M \models p$).

A formula is *satisfactible* if it is satisfied in at least one pair model/world.

A formula is *true* if satisfactible in each of the worlds of the model.

A formula is *valid* if it is true in each of the model/ world pairs.

BDI Agents

Modal Logic: semantics

$\langle M, w \rangle \models$ **true**

$\langle M, w \rangle \models p$ where $p \in Prop$, iff $p \in \pi(w)$

$\langle M, w \rangle \models \neg\varphi$ iff $\langle M, w \rangle \not\models \varphi$

$\langle M, w \rangle \models \varphi \vee \psi$ iff $\langle M, w \rangle \models \varphi$ ó $\langle M, w \rangle \models \psi$

$\langle M, w \rangle \models \Box\varphi$ iff $\forall w' \in W.\ (w,w') \in R$ then $\langle M, w \rangle \models \varphi$

$\langle M, w \rangle \models \Diamond\varphi$ iff $\exists\, w \in W.\ (w,w') \in R \wedge \langle M, w \rangle \models \varphi$

BDI Agents

Modal Logic: basic equivalences

- $\Psi_1 \vee \Psi_2 \quad = \quad \neg \Psi_1 \rightarrow \Psi_2$
- $\Psi_1 \wedge \Psi_2 \quad = \quad \neg (\Psi_1 \rightarrow \neg \Psi_2)$
- $\Psi_1 \leftrightarrow \Psi_2 \quad = \quad (\Psi_1 \rightarrow \Psi_2) \wedge (\Psi_2 \rightarrow \Psi_1)$
- $\Diamond \Psi = \neg \Box \neg \Psi$

BDI Agents

Modal Logic: properties

K:	$\Box(\varphi \Rightarrow \psi) \Rightarrow (\Box\varphi \Rightarrow \Box\psi)$
T:	$\Box\,\varphi \Rightarrow \varphi$
D:	$\Box\,\varphi \Rightarrow \Diamond\varphi$
4:	$\Box\varphi \Rightarrow \Box\Box\varphi$
5:	$\Diamond\varphi \Rightarrow \Box\Diamond\varphi$
NEC:	IF $\models \varphi$ then $\models \Box\varphi$

BDI Agents

Epistemic logic

K: An agent's knowledge is closed under implication, that is an agent knows all consequences of its ***valid beliefs***.

T: It says that if an agent knows facts, the facts must be true. This has often been taken as the major distinguishing feature between knowledge and belief. While you can ***believe*** something that is false, you cannot ***know*** something that is false.

D: An agent's valid beliefs are not contradictory.

4: The ***Positive Introspection Axiom*** says that agents know what they do know.

5: The ***Negative Introspection Axiom*** says that agents know what they do not know.

BDI Agents
Systems in Modal Logics

- **Usual attributes of knowledge and beliefs are referred to *truthness* (the truth of knowledge, the consistency of beliefs) and *introspection.***
- **T = NKT**
- **S4 = NKT4**
- **S5 = NKT5**
- **$S5^{+}$ = KD45**

BDI Agents

Systems on Modal Logics: The K system

K: $\Box(\varphi \Rightarrow \psi) \Rightarrow (\Box\varphi \Rightarrow \Box\psi)$

N: **if** $\models\varphi$ **then** $\models \Box\varphi$

K + N $\Rightarrow$: $\Box\varphi \vee \Diamond\psi \vee \Diamond(\neg\varphi \wedge \neg\psi)$

If φ is a theorem (of any system invoking N), then $\Box\varphi$ is likewise a theorem.

BDI Agents
Knowledge *vs* Beliefs

Knowledge **can be distinguished from** ***beliefs*** **considering that knowledge is a belief that is true:**

$$K_i \varphi \Leftrightarrow B_i \varphi \wedge \varphi$$

$K_i \varphi$ denotates that "an agent *i* knows φ"

BDI Agents

Agents and actions

- **What it means that an *agent* can do action *a*?**
- **What it means that an agent has the *ability* to do action *a*?**
  - **Control: The action *a* should be in the control of A_i or the truth of φ should be under A_i 's control.**
  - **Repeatability: A_i should be able to repeatedly do action *a* or repeatedly bring about formula φ.**
  - **Avoidability: A_i should be able to avoid doing action *a*.**
  - **Free-will: A_i should be free to decide whether or not to do action *a*.**
  - **Causality: A_i should cause action a to take place or the formula φ to become true.**
  - **Intentionality: A_i should intentionally do action a or bring about φ.**

BDI Agents

Agents and actions (2)

- ***Do*(*a*) "The Agent does *a*"**
  - **K** $Do(a \Rightarrow b) \Rightarrow (Do(a) \Rightarrow Do(b))$
  - **T** $Do(a) \Rightarrow a$
  - **Nec** *if* $\models a$ *then* $\models Do(a)$ **($\models$: entails)**

T says "if the Agent does *a* then *a* is the case"

BDI Agents
Agents and actions (3)

- **The Agent can achieve φ in a situation *s***
  - ***Result (s,a) = s'*** *(the result of executing action a in situation s is situation s')*
  - ***Can(s, φ) := ∃a (Result (s,a) satisfies φ)***
- **What does it mean for an Agent to know *how* a plan will achieve its goal?**
  1. **If K(*s*, φ) is true**
  2. **If ∃ a K(Result (s,a), Can(Result (s,a) φ))**

BDI Agents
Logical Omniscience

The necessity rule and the Knowledge axiom together imply that agent *believes* all the valid formulae and that it *knows* all the logical consequences of its valid beliefs

***Logical consistency*: an agent cannot believe that φ and ψ at the same time if $\varphi \Rightarrow \neg\psi$**

***Logical equivalence*: given two propositions i) φ and, ii) $\varphi \wedge \psi$, if ψ is true then i) and ii) are logically equivalent.**

BDI Agents

Logical Omniscience Problem (1)

- **If epistemic logic is to be interpreted as describing actual knowledge of realistic (though idealized) agents, then the discussed closure properties require agents to be very powerful reasoners whose computational capacities cannot be achieved by real (human or artificial) agents, who are simply not logically omniscient.**

- **Logical omniscience poses a problem because it contradicts the fact that agents are limited in their reasoning powers. They are inherently resource-bounded and therefore cannot handle an unlimited amount of information. Agents may establish immediately certain logical truths or simple consequences of what they consciously assented to.**

BDI Agents

Logical Omniscience Problem (2)

- **However, there are highly remote dispositional states which could only be established by complex, time-consuming reasoning. The modal framework cannot distinguish between a sentence that an agent consciously assented to and a piece of potential knowledge which could never be made actual by the agent and is therefore not suited to model resource-bounded reasoning**

BDI Agents
Logical Omniscience Problem (3)

- **An agent believes all valid formulas**
- **The agent's beliefs are a closure under logical implication**
- **Equivalent propositions are equivalent beliefs**
- **If an agent is inconsistent then it believes all (anything)**
- **In the worst of the cases automation is not possible.**

BDI Agents

Logical Omniscience Problem: Solutions

- **The LOP shows that the view that modal epistemic logic describes actual knowledge of idealized agents is not tenable.**
- **Consistency should be a rigorous property for an ideal reasoner: it is enough not to be contradictory.**
- **Equivalent propositions are not necessarily equivalent beliefs.**

BDI Agents
Intentions and Beliefs

- ***Consistency intentions-beliefs***
  - **An agent ought to believe that its objectives are possible**
  - **An agent cannot believe that it will not reach its objectives**
- ***Incompleteness intentions-beliefs***
  - **Intentions should be consistent with beliefs but not necessarily to support them.**
  - **It is possible to have incomplete beliefs about the own intentions.**
- ***Side-effect Problem***
  - **An agent that tries to do a and believes that doing a requires doing b does not have to have the intention of making b**

BDI Agents
Desires and Intentions

- ***Internal Consistency***
  - **An agents should avoid to have intentions in conflict but it may have conflicting desires**
- ***Means-end Analysis***
  - **Intentions may create problems for future deliberations, desires not necessarily.**
- ***Keep memory of successes and failures (of intentions)***
  - **To keep the compromise with a given action; There is no compromise associated to a desire. Intentions are desires + compromises to act.**
- ***Consistency with beliefs***
  - **Intentions ought to be consistent with the beliefs but not necessarily with the desires**

BDI Logic

A logic framework

- **The formulation language is FOL, multimodal with equality.**
- **Standard FOL operators + *Happens, Done,* BEL, and GOAL**
- **The world is defined as sequence of discrete events that extends infinitely towards the past and future.**
- ***Happens* defines a sequence of events that will occur**
- ***Done* defines a sequence of events that already happen**
- ***e;e'* (*e* followed by *e'*)**
- ***e|e'* (*e* or *e'*)**
- ***e?* (test) *e** (iteration)**

BDI Logic

A logic framework(2)

- **The temporal operators are: □ (always) ◊ (sometimes), LATER, y BEFORE. Those symbols should not be identified with the *necessity* and *possibility*.**

- **The semantics of BEL, and GOAL is given by their accessibility relations to the possible worlds.**

- **It is assumed that agents have only a restricted set of accessibility relations. This is known as *realism constraint*. This avoids that agents choose worlds that are outside of their beliefs**

BDI Logic

A logic framework (3)

- **If an agent believes that φ is true (now), it cannot desire it become false (now), an agent cannot chose what it cannot modify**

$$\models (\mathrm{BEL}_x\ \varphi) \Rightarrow \neg\ (\mathrm{GOAL}x\ \varphi)$$

- **An agent assumes that all its goals are false (not achieved) to possible satisfy them**

$$(\mathrm{A\_GOAL}_x\ \varphi) = (\mathrm{GOAL}_x\ (\mathrm{LATER}\ \varphi)) \wedge (\mathrm{BEL}_x\ \neg\varphi)$$

BDI Agents

Mental States of an Agent

- ***Beliefs*** **represent some knowledge about other agents. Agent a_1 believes φ about a_2.**
  - $\mathbf{BEL}_{a1}(\mathbf{DO}_{a2}\,\varphi)$
- ***Capacities*** **correspond to the abilities to act.**
- **Goals are generated by the evaluation of other agent's messages and/or correspond to the agent's aims.**
- ***Intentions*** **correspond to the decision made when adopting a *goal*. An agent only adopts goals that he knows he can achieve.**

BDI Agents

Mental states of an Agent

Intention: Is the will to accomplish a desire or to perform an action

$$Intend(x,a) \Leftrightarrow \exists P \,/\, Goal(x,P)$$

$$Bel(x,a \supset P)$$

$$Bel(x,\neg P)$$

$$Bel(x,Do(x,a))$$

$$Bel(x,a) \supset Bel(x,P)$$

BDI Agents
Mental States of an Agent

- **An intention is fulfilled by the actions oriented to achieve a goal.**
- **A goal assigned by another agent is a *mission*.**
- **An agent *remembers* all assigned goals:**

$$\mathbf{MIS}\ a_1\ a_2\ \varphi$$
$$\mathbf{G}_a\ \varphi = \mathbf{MIS}\ a,a\ \varphi$$

BDI Agents

Beliefs about competences

Who can do what?

* **By task delegation**

* **By collaboration**

$$\boldsymbol{Intend}\,(a, Do\,(a, T\,))$$
$$\wedge\, Bel\,(a, \neg Do\,(a, T\,))$$
$$\wedge\, Bel\,(a, \Diamond Do\,(b, T\,))$$
$$\mapsto\ request\,(a, b, Intend\,(b, Do\,(b, T\,)))$$

BDI Agents
Beliefs about competences

- **Introspective view**
  - **Bel(a, (Bel (a, Bel (a, Bel(.... Bel(a,a))...)**
- **Own model**

$$Bel(a, Bel(b, \varphi))$$

$$Bel(a, \varphi Do(b, \varphi))$$

Implementing BDI Agents

Agent Control Loop

- **This is the abstract algorithm for the BDI reasoning cycle:**

```
B := B0;
I := I0;
while true do
      get next percept ρ;
      B := brf(B, ρ);
      D := options(B, I);
      I := filter(B, D, I);
      π := plan(B, I);
      execute(π)
end while
```

Implementing BDI Agents

Deliberation

- **How does an agent deliberate?**
  - **begin by trying to understand what the *options* available to you are**
  - ***choose between them*, and *commit* to some**
- **Chosen options are then intentions**
- **Deliberation can be decomposed into two distinct functional components:**
  - ***option generation***
    in which the agent generates a set of possible alternatives; Represent option generation via a function, *options*, which takes the agent's current beliefs and current intentions, and from them determines a set of options (= *desires*)
  - ***filtering***
    in which the agent chooses between competing alternatives, and commits to achieving them.
    In order to select between competing options, an agent uses a *filter* function.

Implementing BDI Agents
Problems

- **Overcommitment**
  - **An agent has commitment both to *ends* (i.e., the wishes to bring about), and *means* (i.e., the mechanism via which the agent wishes to achieve the state of affairs)**
  - **Our agent control loop is overcommitted, both to means and ends**
  - **Solution:**
    - ***replan* if ever a plan goes wrong**
    - ***Reconsider the intentions* every now and then, to check if they are still achievable**
- **A dilemma on intention reconsideration:**
  - **an agent that does not stop to reconsider its intentions sufficiently often will continue attempting to achieve its intentions even after it is clear that they cannot be achieved, or that there is no longer any reason for achieving them**
  - **an agent that *constantly* reconsiders its attentions may spend insufficient time actually working to achieve them, and hence runs the risk of never actually achieving them**
  - **Solution: incorporate an explicit *meta-level control* component, that decides whether or not to reconsider**

```
B := B_0;
I := I_0;
while true do
    get next percept ρ;
    B := brf(B, ρ);
    D := options(B, I);
    I := filter(B, D, I);
    π := plan(B, I);
    while not (empty(π)
            or succeeded(I, B)
            or impossible(I, B)) do
        α := hd(π);
        execute(α);
        π := tail(π);
        get next percept ρ;
        B := brf(B, ρ);
        if reconsider(I, B) then
            D := options(B, I);
            I := filter(B, D, I);
        end-if
        if not sound(π, I, B) then
            π := plan(B, I)
        end-if
    end-while
end-while
```

Glosary

- **Competences:** A specific range of skill, knowledge, or ability.
- **Deontic logic:** is the field of logic that is concerned with obligation, permission, and related concepts.
- **Doxastic logic:** is a modal logic that is concerned with reasoning about beliefs
- **Intuitionistic logic:** The system preserves ***justification***, rather than ***truth***, across transformations yielding derived propositions. From a practical point of view, there is also a strong motivation for using intuitionistic logic, since it has the existence property, making it also suitable for other forms of mathematical constructivism.
- **Stance:** intellectual or emotional attitude
- **Volition**: the capability of conscious choice and decision and intention

References

[1.] Jennings, N. & Luck, M. **"Introduction to Autonomous Agents and Multi-Agent Systems"** The 5th Int. Conf. On Autonomous Agents. 2001.

[2.] Wooldridge, M**. "Introduction to Multiagent Systems".** John Wiley and Sons, 2002.

[3.] Y. Shoham**, "An Overview of Agent-Oriented Programming",** in J. M. Bradshaw, ed, Software Agents, pp 271–290. AAAI Press / The MIT Press, 1997.

[4.] Haddadi, A**. "Communication and Cooperation in Agent Systems: A Pragmatic Theory"** LNAI #1056. Springer-Verlag. 1996**.**

[5.] Weiss, G. **"Multiagent Systems: A modern Approach to Distributed Artificial Intelligence".** MIT Press. 1999. ISBN 0262-23203

[6.] Dennett, D**. "Brainstorm".** Harvester Press. 1981

[7.] Thijsse, E. **"Partial Logic and Knowledge Representation"** EBURON. 1992. ISBN 90-5166-267-X

8. C. I. Lewis, `**Implication and the Algebra of Logic**', Mind (1912) 12: 522{31};

These slides are based mainly in [5] and material from M. Wooldridge, J. Padget and M. de Vos

Discussion about Agents

- **Agents vs. Objects**
- **Agents vs. Expert Systems**

Agents vs. Objects

- **Are agents just objects by another name?**
- **Object**:
  - It encapsulates some state
  - It communicates via message passing
  - It has methods, corresponding to operations that may be performed on this state

Agents vs. Objects

- Main differences:
  - *agents are autonomous:*
    agents embody stronger notion of autonomy than objects, and in particular, they decide for themselves whether or not to perform an action on request from another agent
  - *agents are smart:*
    capable of flexible (reactive, pro-active, social) behavior, and the standard object model has nothing to say about such types of behavior
  - *agents are active:*
    a multi-agent system is inherently multi-threaded, in that each agent is assumed to have at least one thread of active control
- As Wooldridge says:
  - objects do it for *free*…
  - …agents do it because they *"want"*
  - …agents do it for *"money"* (aka *utility*)

Agents vs. Expert Systems

- Aren't agents just expert systems with another name?
  - Expert systems are deliberative
  - *e.g.* MYCIN

- Main differences:
  - agents *situated in an environment:*
    MYCIN is not aware of the world — only information obtained is by asking the user questions
  - agents *act:*
    MYCIN does not operate on patients

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

Technological Challenges

- **Implementing multi-agent systems is still a complex task**.
  - Lack of maturity in both methodologies and programming tools.
  - Lack of specialized debugging tools.
  - Lack of skills needed to move from analysis and design to code.
  - Problems associated with awareness of the specifics of different agent platforms.
  - Problems in understanding the nature of what is a new and distinct approach to systems development

Technological Challenges

- **Increase quality of agent software to industrial standard**
  - Agent oriented design methodologies, tools and development environments.
  - Seamless integration with existing technologies.
    - Web, information systems
  - Non-functional properties: scalability, performance, reliability, and robustness.
  - Analysis point of view
    - Agent technology requires dedicated basic concepts and languages:
      - Dynamic aspects, such as time and action.
      - Locality aspects, such as position in a space.

References

[1] **Luck, M., McBurney, P., Shehory, Onn, Willmott, S. "Agent Technology: Computing as interaction. A Roadmap to Agent Based Computing". Agentlink, 2005. ISBN 085432 845 9**

[2] Wooldridge, M. "Introduction to Multiagent Systems". John Wiley and Sons, 2002.

[3] Russell, S. & Norvig, P. "Artificial Intelligence: A Modern Approach" Prentice-Hall Series in Artificial Intelligence. 2002 ISBN 0-13-103805-2

[4] Haddadi, A. "Communication and Cooperation in Agent Systems: A Pragmatic Theory" Lecture Notes in Artificial Intelligence #1056. Springer-Verlag. 1996. ISBN 3-540-61044-8

[5] Rosenschein, J. & Zlotkin, G. "Rules of Encounter. Designing Conventions for Automated Negotiation among Computers". MIT Press. 1994 ISBN 0-262-18159-2

[6] Weiss, G. "Multiagent Systems: A modern Approach to Distributed Artificial Intelligence". MIT Press. 1999. ISBN 0262-23203

These slides are based mainly in material from [2] and [1], with some additions from material by U. Cortés, J.Padget, A. Moreno and Steve Willmott