MULTIAGENT SYSTEMS

Ubi-Comp Lecture 6

PART 1

Multi-Agent Systems (MAS)
(Intelligent Agent Systems)

"Software systems in which program modules are given autonomy and intelligence and collaborate to attain system objectives" [Nwana, '97]

- MAS are major part of Distributed Artificial Intelligence (DAI)
- Fastest growing area in Computer Sciences
- Rapidly being applied to real-life technological problems (air traffic control, battle simulation, manufacturing, supply chain management, etc.)

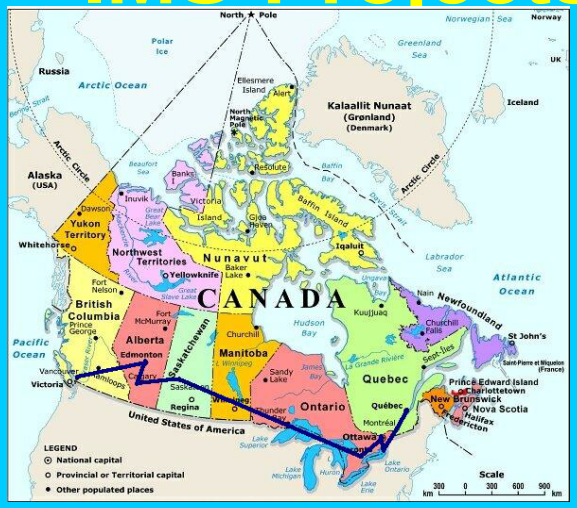
Prediction 2009+, Forrester Research, Inc

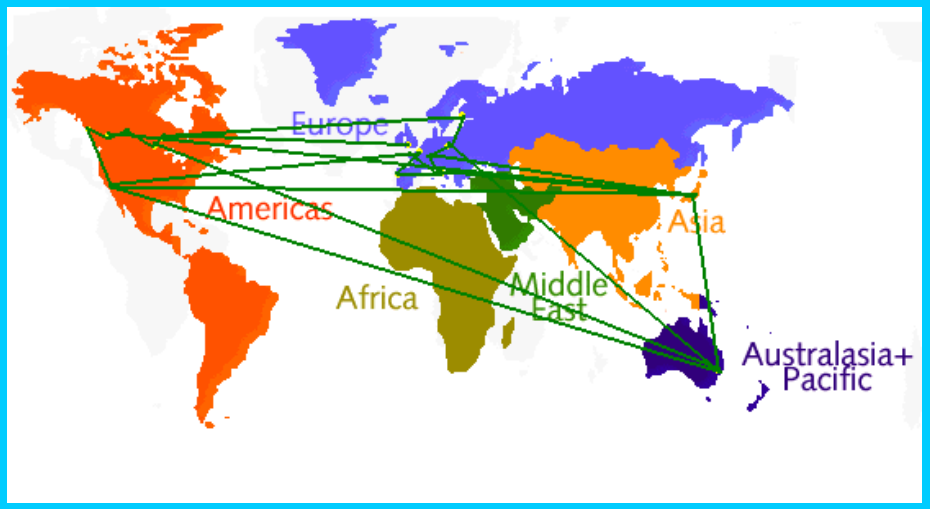
	Phase 1 (2003-2005)	Phase 2 (2006-2008)	Phase 3 (2009+)
	Agent-monitored	Agent-managed	Agent-optimized
Main purpose of using agents	• Sense/interpret environmental data	• Coordinate activities with partners	• Decentralize and automate decisions
Killer app	• Multi-agent simulation systems	• Multi-agent execution systems	• Multi-agent optimization systems
Agent attributes exploited	• Goal-driven • Cooperation ability • Reactivity	• Mobility • Proactivity	• Autonomy • Adaptability • Learning
Application domains	• Financial risk management • Bioinformatics • National security	• Store-level replenishment • Predictive maintenance	• Adaptive supply networks • Domotics • Nanotechnology
Enabling sciences/ standards	• Complexity science • Grid computing • Organic IT	• Web services (BPEL4WS) • AUML	• X Internet (e.g., RFID) • Semantic Web • Swarm intelligence
Key vendors to watch	• IBM, Lost Wax, NuTech Solutions, Searchspace, Tryllian	• Agentis Software, BTexact Technologies, CGE&Y, HP, IBM, living systems, Microsoft, SAP	• Fujitsu, IBM, Microsoft, Oracle, SAP, Sony, Wipro

Source: Forrester Research, Inc.

CYBERINFRASTRUCTURE

- Cyberspace as a
  Dynamic Service Environment
- **Agentcities, FIPA, PlanetLab**
- **IMS-Projects**





Dynamic Service Environment

- COALITION OF SERVICES MODELED THROUGH
- Deployment of automated, intelligent software services (e.g.,, automatic negotiations, financial transactions, advertising and bidding; order placement/delivery, etc.)

Integrated Framework Enabling:

- Complex interactions between such services (e.g.,, compliance policies; argumentation and persuasion via complex conversation protocols, etc.);
- **Dynamic discovery and composition of services to create new compound value added services** (e.g.,, dynamic virtual clustering of synergetic partnerships of collaborative entities aiming to achieve a common goal).

Agents, a (borader) Definition

- 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)
- Wooldridge, M. and Jennings, N.R., “Intelligent Agents: Theory and Practice”, Knowledge Engineering Review, Vol. 10, No. 2, pp. 115-152, 1995
- According to this – eGadgets are agents

Multiagent Systems, a Definition

- 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

COORDINATION

- *Coordination* is a mechanism for ensuring that agents' activities retain some desired relationship/s (sequence, complementarity, etc.).
- *Control* is the extent to which coordination information *must* be implemented by a recipient agent.
- The range of control is from *none* to *total*. Control is inversely related to autonomy (for the recipient agent, no control corresponds to *total* autonomy, and being totally controlled corresponds to *zero* autonomy.

- The fundamental coordination mechanisms are:
  - Mutual adjustment

    Agents share information and resources to achieve a common goal, adjusting their behaviour according to the behaviour of the other agents. No agent has prior control and decision making is joint. Coordination in peer groups and markets is usually by mutual adjustment.

  - Direct supervision

    One agent has some degree of control over others and can control information, resources, and behaviour. Often established through mutual adjustment (e.g. following acceptance of employment or a contract).

- Coordination by standardization

  Standard procedures are established for agents to follow. In mutual adjustment, are implemented by acceptance. In direct supervision, are implemented through (mandatory) requests.

- Mediated coordination

  A mediator acts as a facilitator (e.g. finds/routes information, etc.), a broker (a ‘go-between’ and advisor on resource negotiations, etc.), and a supervisor (exercising some degree of direct supervision). The first role is mandatory, but the others are optional. A mediator facilitates or brokers mutual adjustment between agents and may also use direct supervision.

- Coordination by reactive behaviour

  Agents react to particular stimuli (“situations) with specific behaviours (“actions”). With appropriately selected or evolved stimuli-behaviour groupings and distributions, system-level patterns of coordinated behaviour emerge which contribute to the achievement of common or system goals.

COOPERATION

Of the five coordination mechanisms considered, only two (mutual adjustment, mediated coordination) allows for agents to have autonomy, i.e. the freedom to make choices and determine their own actions. Agents which have some degree of autonomy can *cooperate* with other such agents on tasks and activities, for their own or mutual benefit.

Advantages of Cooperation

- Complete tasks quicker through shared effort
- By sharing resources, achieve tasks otherwise not possible
- Make use of complementary capabilities
- Avoid harmful interactions

Modes of Cooperation

- accidental: not intended
- unilaterally intended: one agent intentionally helps another
- mutual cooperation: two or more agents intentionally collaborate

Degrees of Cooperation

- Fully cooperative (benevolent): Agents always attempt to assist other agents that request or need their help
- Antagonistic: Agents do not cooperate with others and may even try to block their goals
- Partly cooperative: Agents sometimes or to some extent will assist other agents

NEGOTIATION

The process of improving agreement (reducing inconsistency and uncertainty) on common viewpoints or plans through the structured exchange of relevant knowledge (Durfee, 1989)

- Bargaining (promising something in exchange for something else), bidding (offering a service or capability at a specific ‘price’), and contracting (committing to provide a service or capability at a specific ‘price’) may be part of the negotiation process.
- Prior to negotiation: an agent needs to identify a resource or service it requires but cannot supply and then it needs to identify an agent/s which it believes potentially could.

- Negotiation: Agent contacts other agents and identifies its need or requests resource or service (often under specified conditions); other agent/s identify what can be supplied under what conditions; initiating agent/s accepts conditions; supplying agent/s commit to provide resource or service.
  - In *single-stage* negotiation, initiating agent makes a request and the respondent accepts or rejects this.
  - In *multi-stage* negotiation, agents iterate through more than one stage of offer/counter-offer.
  - Negotiation protocols: a structured procedure for one or more stages of the negotiation process.

Agents and Objects

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

From Objects to Agents

- **Object-oriented paradigm** models systems focusing on the structural, static characteristics of their parts which are defined through encapsulation and inheritance. (PASSIVE)

- **Agent paradigm** models systems focusing on the underlining dynamics defined by the ***interactions*** between their parts (PROACTIVE).

- Software Agents are capable to **initiate** communication and decide (like a human) when and how to respond to external stimuli (e.g.,, manifested upon them as requests from other agents).

Agents and 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

Objects do it because they are ‘told’ to…

- *agents do it because they **want to***

Agents as Intentional Systems

- When explaining human activity, it is often useful to make statements such as the following:

  Janine took her umbrella because she *believed* it was going to rain.

  Michael worked hard because he ***wanted*** to possess a PhD.

- These statements make use of a *folk psychology*, by which human behavior is predicted and explained through the attribution of *attitudes*, such as believing and wanting (as in the above examples), hoping, fearing, and so on
- The attitudes employed in such folk psychological descriptions are called *intentional drives/states*

Agents as Intentional Systems

- The philosopher Daniel Dennett coined the term *intentional system* to describe entities 'whose behavior can be predicted by the method of attributing belief, desires and rational acumen'
- Dennett identifies different 'grades' of intentional system:
  'A *first-order* intentional system has beliefs and desires (etc.) but no beliefs and desires *about* beliefs and desires. …A *second-order* intentional system is more sophisticated; it has beliefs and desires (and no doubt other intentional states) about beliefs and desires (and other intentional states) — both those of others and its own'

Agents as Intentional Systems

- Is it legitimate or useful to attribute beliefs, desires, and so on, to computer systems?

Agents as Intentional Systems

- McCarthy argued that there are occasions when the *intentional stance* is appropriate:

‘To ascribe *beliefs*, *free will*, *intentions*, *consciousness*, *abilities*, or *wants* to a machine is *legitimate* when such an ascription expresses the same information about the machine that it expresses about a person. It is *useful* when the ascription helps us understand the structure of the machine, its past or future behavior, or how to repair or improve it. It is perhaps never *logically required* even for humans, but expressing reasonably briefly what is actually known about the state of the machine in a particular situation may require mental qualities or qualities isomorphic to them. Theories of belief, knowledge and wanting can be constructed for machines in a simpler setting than for humans, and later applied to humans. Ascription of mental qualities is *most straightforward* for machines of known structure such as thermostats and computer operating systems, but is *most useful* when applied to entities whose structure is incompletely known’.

Agents as Intentional Systems

- What objects can be described by the intentional stance?
- As it turns out, more or less anything can. . . consider a light switch:

  ‘It is perfectly coherent to treat a light switch as a (very cooperative) agent with the capability of transmitting current at will, who invariably transmits current when it believes that we want it transmitted and not otherwise; flicking the switch is simply our way of communicating our desires’. (Yoav Shoham)

- But most adults would find such a description absurd!
  Why is this?

Agents as Intentional Systems

- The answer seems to be that while the intentional stance description is consistent,

  > . . . it does not *buy us anything*, since we essentially understand the mechanism sufficiently to have a simpler, mechanistic description of its behavior.
  >
  > (Yoav Shoham)

- Put crudely, the more we know about a system, the less we need to rely on animistic, intentional explanations of its behavior
- But with very complex systems, a mechanistic, explanation of its behavior may not be practicable
- ***As computer systems become ever more complex, we need more powerful abstractions and metaphors to explain their operation — low level explanations become impractical. The intentional stance is such an abstraction***

Agents as Intentional Systems

- The intentional notions are thus *abstraction tools*, which provide us with a convenient and familiar way of describing, explaining, and predicting the behavior of complex systems
- Remember: most important developments in computing are based on new *abstractions*:
  - procedural abstraction
  - abstract data types
  - objects

  Agents, and agents as intentional systems, represent a further, and increasingly powerful abstraction
- So agent theorists start from the (strong) view of agents as intentional systems: one whose simplest consistent description requires the intentional stance

Agents as Intentional Systems

- This *intentional stance* is an *abstraction tool* — a convenient way of talking about complex systems, which allows us to predict and explain their behavior without having to understand how the mechanism actually works
- Now, much of computer science is concerned with looking for abstraction mechanisms (witness procedural abstraction, ADTs, objects,…)

  So why not use the intentional stance as an abstraction tool in computing — to explain, understand, and, crucially, program *computer systems?*

- This is an important argument in favor of agents

Agents as Intentional Systems

- Other 3 points in favor of this idea:
- Characterizing Agents:
  - It provides us with a familiar, non-technical way of *understanding & explaining* agents
- Nested Representations:
  - It gives us the potential to specify systems that *include representations of other systems*
  - It is widely accepted that such nested representations are essential for agents that must cooperate with other agents

Agents as Intentional Systems

- Post-Declarative Systems:
  - This view of agents leads to a kind of post-declarative programming:
    - In procedural programming, we say exactly *what* a system should do
    - In declarative programming, we state something that we want to achieve, give the system general info about the relationships between objects, and let a built-in control mechanism (e.g., goal-directed theorem proving) figure out what to do
    - With agents, we give a very abstract specification of the system, and let the control mechanism figure out what to do, knowing that it will act in accordance with some built-in theory of agency (e.g., the well-known Cohen-Levesque model of intention)

Agents and Expert Systems

- Aren’t agents just expert systems by another name?
- Expert systems typically disembodied ‘expertise’ about some (abstract) domain of discourse (e.g., blood diseases)
- Example: MYCIN knows about blood diseases in humans
  - It has a wealth of knowledge about blood diseases, in the form of rules
  - A doctor can obtain expert advice about blood diseases by giving MYCIN facts, answering questions, and posing queries

Agents and Expert Systems

- 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

Intelligent Agents and AI

- Aren’t agents just the AI project? Isn’t building an agent what AI is all about?
- AI aims to build systems that can (ultimately) understand natural language, recognize and understand scenes, use common sense, think creatively, etc. — all of which are very hard
- So, don’t we need to solve all of AI to build an agent…?

Intelligent Agents and AI

- When building an agent, we simply want a system that can choose the right action to perform, typically in a limited domain
- We *do not* have to solve *all* the problems of AI to build a useful agent:

  a little intelligence goes a long way!

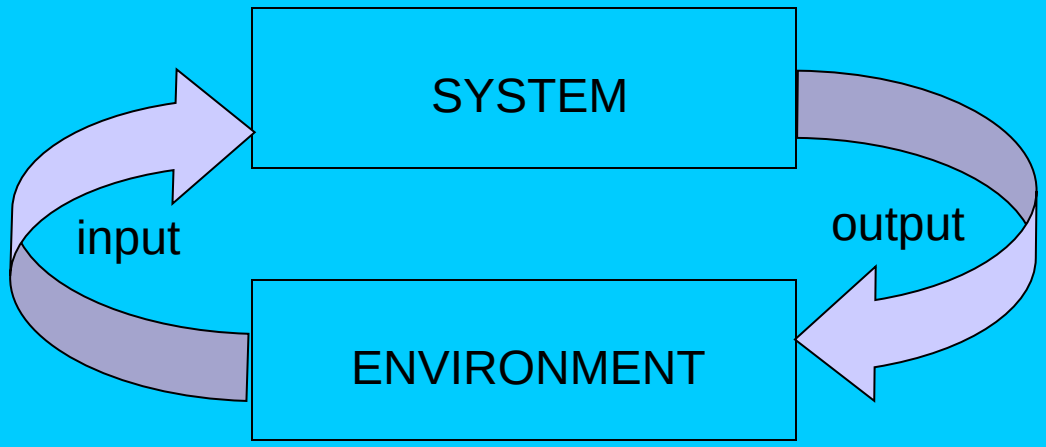
- Oren Etzioni, speaking about the commercial experience of NETBOT, Inc:
  "We made our agents dumber and dumber and dumber…until finally they made money."

What is an Agent?

- The main point about agents is they are *autonomous*: capable of **acting** independently, exhibiting control over their internal **state**

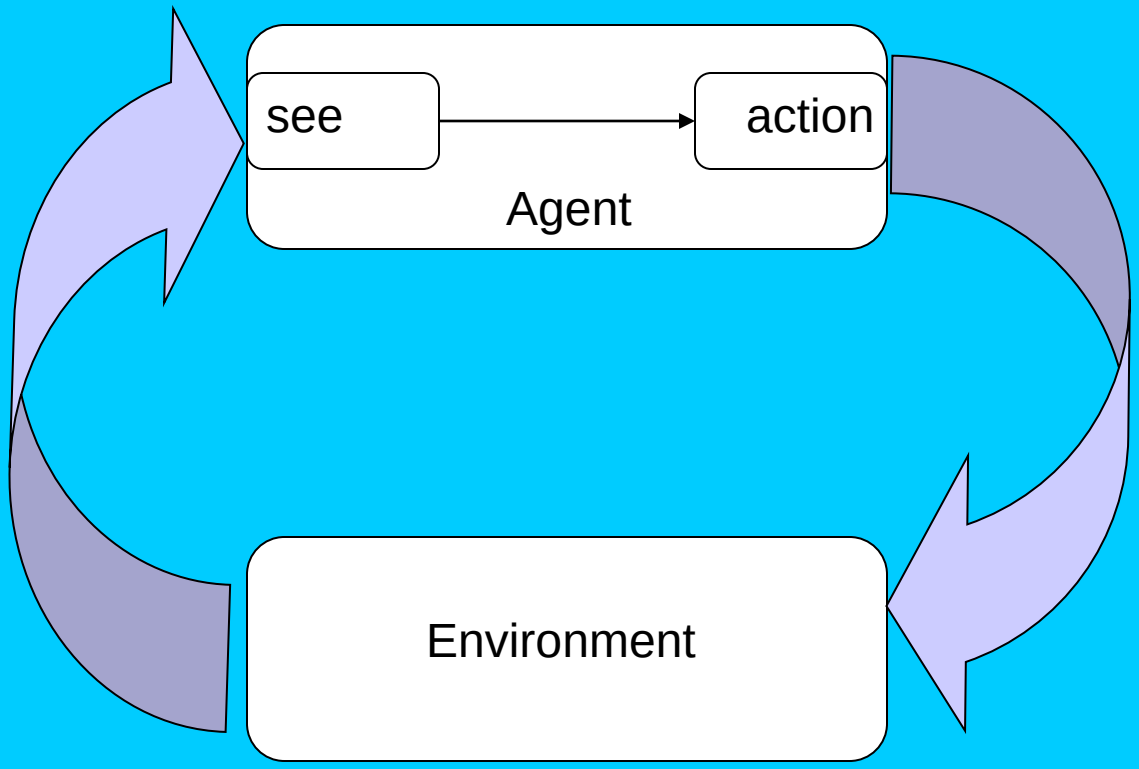
Environment

- Thus: *an* agent *is a computer system capable of* autonomous action *in some environment in order to meet its* design objectives



Perception

- P*erception* system:



Perception

- The *see* function is the agent's ability to observe its environment, whereas the *action* function represents the agent's decision making process
- *Output* of the *see* function is a *percept*:

$$see : E \rightarrow Per$$

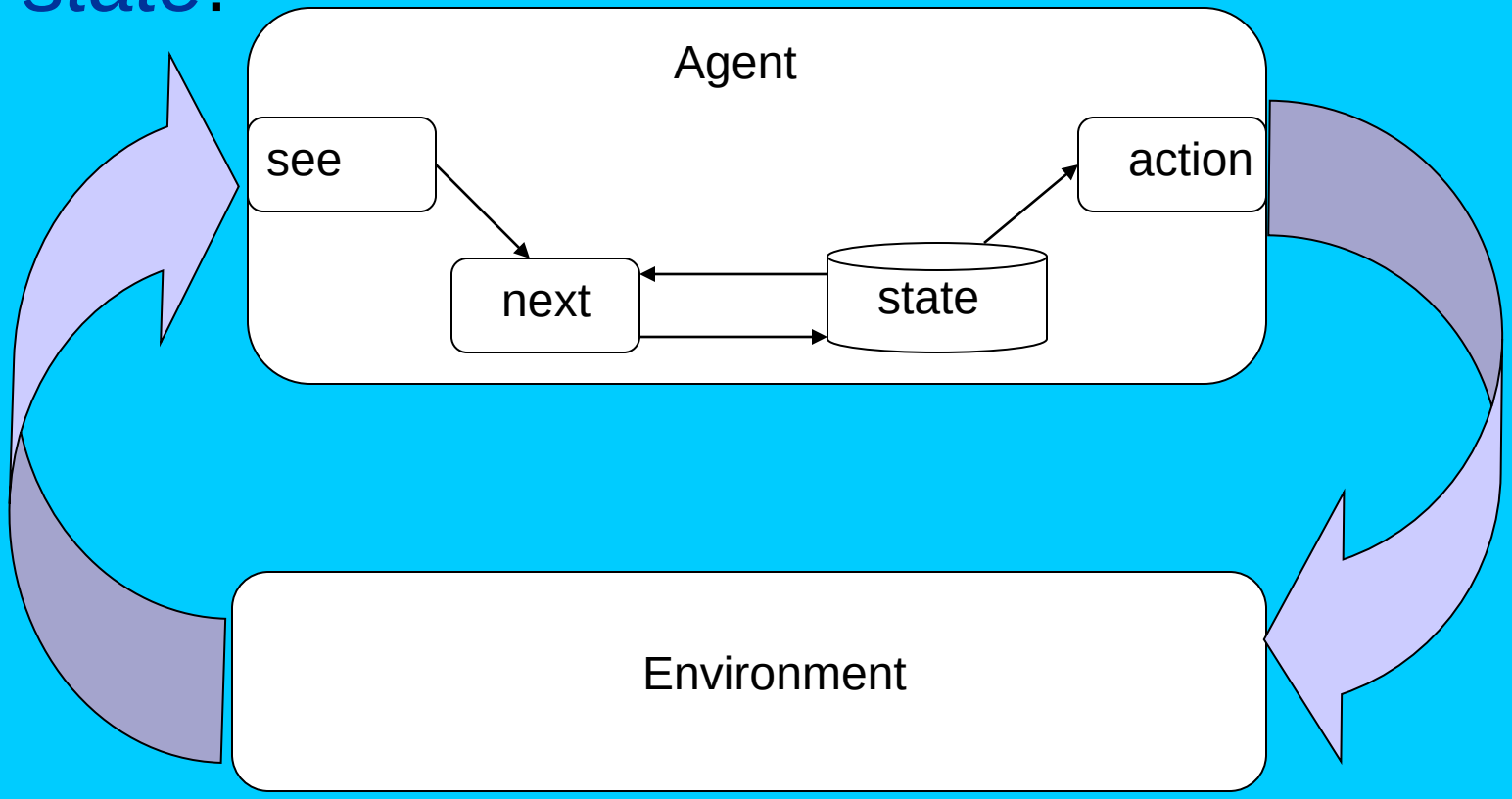
which maps environment states to percepts, and *action* is now a function

$$action : Per^* \rightarrow A$$

which maps sequences of percepts to actions

Agents with State

- We now consider agents that *maintain state*:



Agents with State

- These agents have some internal data structure, which is typically used to record information about the environment state and history.
  Let I be the set of all internal states of the agent.
- The perception function see for a state-based agent is unchanged:

$$see : E \rightarrow Per$$

  The action-selection function $action$ is now defined as a mapping

$$action : I \rightarrow Ac$$

  from internal states to actions. An additional function $next$ is introduced, which maps an internal state and percept to an internal state:

$$next : I \times Per \rightarrow I$$

What is an Agent?

- Trivial (non-interesting) agents:
  - thermostat
  - UNIX daemon (e.g., biff)
- *An* intelligent agent *is a computer system capable of* flexible *autonomous action in some environment*
- By *flexible*, we mean:
  - *reactive*
  - *pro-active*
  - *social*

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*
- 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)

Proactiveness

- Reacting to an environment is easy (e.g., stimulus $\rightarrow$ 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

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
- Designing an agent that can balance the two remains an open research problem

Agent-Oriented Programming

- [Shoham]
- AOP a ‘new programming paradigm, based on a societal view of computation’
- The key idea in AOP is that of directly programming agents in terms of intentional notions like belief, commitment, and intention
- The motivation behind such a proposal is that, as we humans use the intentional stance as an *abstraction* mechanism for representing the properties of complex systems,
  In the same way we use the intentional stance to describe humans, it might be useful to use the intentional stance to program machines.

AGENT0

- Shoham suggested that a complete AOP system will have 3 components:
  - a logic for specifying agents and describing their mental states
  - an interpreted programming language for programming agents
  - an ‘agentification’ process, for converting ‘neutral applications’ (e.g., databases) into agents
- Results only reported on first two components.
- Relationship between logic and programming language is *semantics*
- We will skip over the logic(!), and consider the first AOP language, AGENT0

AGENT0

- AGENT0 is implemented as an extension to LISP
- Each agent in AGENT0 has 4 components:
  - a set of capabilities (things the agent can do)
  - a set of initial beliefs
  - a set of initial commitments (things the agent will do)
  - a set of ***commitment rules***
- The key component, which determines how the agent acts, is the commitment rule set

Commitment Strategies

- The following *commitment strategies* are commonly discussed in the literature of rational agents:
  - *Blind commitment*

    A blindly committed agent will continue to maintain an intention until it believes the intention has actually been achieved. Blind commitment is also sometimes referred to as *fanatical* commitment.
  - *Single-minded commitment*

    A single-minded agent will continue to maintain an intention until it believes that either the intention has been achieved, or else that it is no longer possible to achieve the intention.
  - *Open-minded commitment*

    An open-minded agent will maintain an intention as long as it is still believed possible.

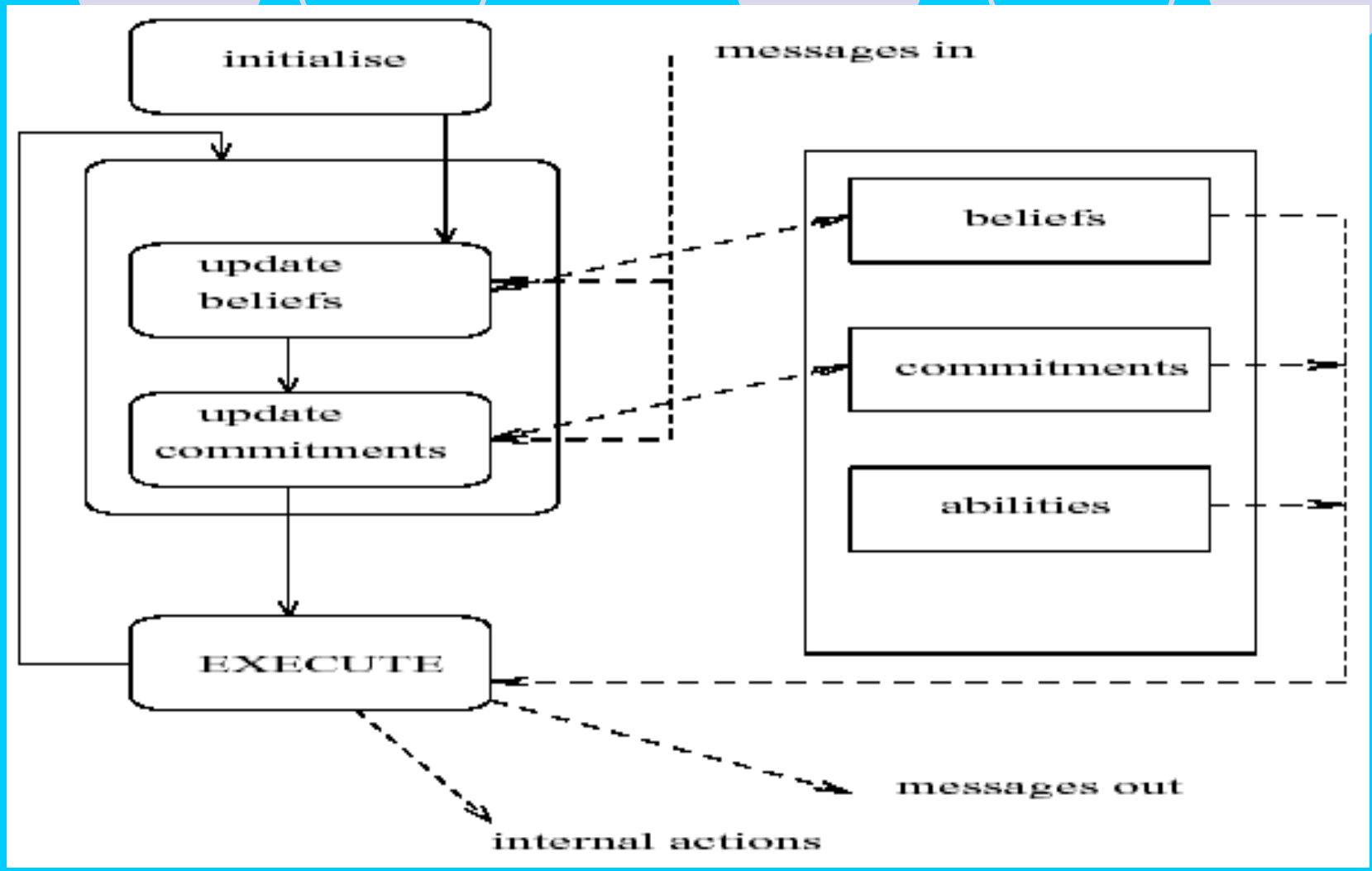
AGENT0 – Commitment Rules

- Each commitment rule contains
  - a *message condition*
  - a *mental condition*
  - an action
- On each ‘agent cycle’…
  - The message condition is matched against the messages the agent has received
  - The mental condition is matched against the beliefs of the agent
  - If the rule fires, then the agent becomes committed to the action (the action gets added to the agent’s commitment set)

AGENT0

- Actions may be
  - *private*:
    an internally executed computation, or
  - *communicative*:
    sending messages
- Messages are constrained to be one of three types:
  - “requests” to commit to action
  - “unrequests” to refrain from actions
  - “informs” which pass on information

AGENT0


initialise
messages in
update beliefs
update commitments
beliefs
commitments
abilities
EXECUTE
messages out
internal actions

AGENT0 Commitment Rule Example

```
COMMIT(
     ( agent, REQUEST, DO(time,
  action)
     ), ;;; msg condition
     ( B,
          [now, Friend agent] AND
          CAN(self, action) AND
          NOT [time, CMT(self,
  anyaction)]
     ), ;;; mental condition
     self,
     DO(time, action)
)
```

AGENT0

- This rule may be paraphrased as follows: if I receive a message from *agent* which requests me to do *action* at *time*, and I believe that:
  - *agent* is currently a friend
  - I can do the action
  - At *time*, I am not committed to doing any other action

  then commit to doing *action* at *time*

AGENT0 and PLACA

- AGENT0 provides support for multiple agents to cooperate and communicate, and provides basic provision for debugging…
- …it is, however, a *prototype*, that was designed to illustrate some principles, rather than be a production language
- A more refined implementation was developed by Thomas, for her 1993 doctoral thesis
- Her Planning Communicating Agents (PLACA) language was intended to address one severe drawback to AGENT0: the inability of agents to plan, and communicate requests for action via high-level goals
- Agents in PLACA are programmed in much the same way as in AGENT0, in terms of *mental change* rules

AGENT0 and PLACA

- An example mental change rule:

```
(((self ?agent REQUEST (?t (xeroxed ?x)))
  (AND (CAN-ACHIEVE (?t xeroxed ?x)))
     (NOT (BEL (*now* shelving)))
     (NOT (BEL (*now* (vip ?agent))))
  ((ADOPT (INTEND (5pm (xeroxed ?x)))))
  ((?agent self INFORM
     (*now* (INTEND (5pm (xeroxed ?x)))))))
```

- Paraphrased:
  if someone asks you to xerox something, and you can, and you don't believe that they're a VIP, or that you're supposed to be shelving books, then
  - adopt the intention to xerox it by 5pm, and
  - inform them of your newly adopted intention

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

Other Properties of Agency

- *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

Intelligent Agents

An intelligent agent is a software entity which exhibits, in some significant measure, autonomy, intelligence, and environmental awareness, and which interacts with its environment to achieve internal goals.

- Co-operation with other agents
- Software + Hardware?

  (Agents or Holons or Autonomous XXX?)

Why Agents?

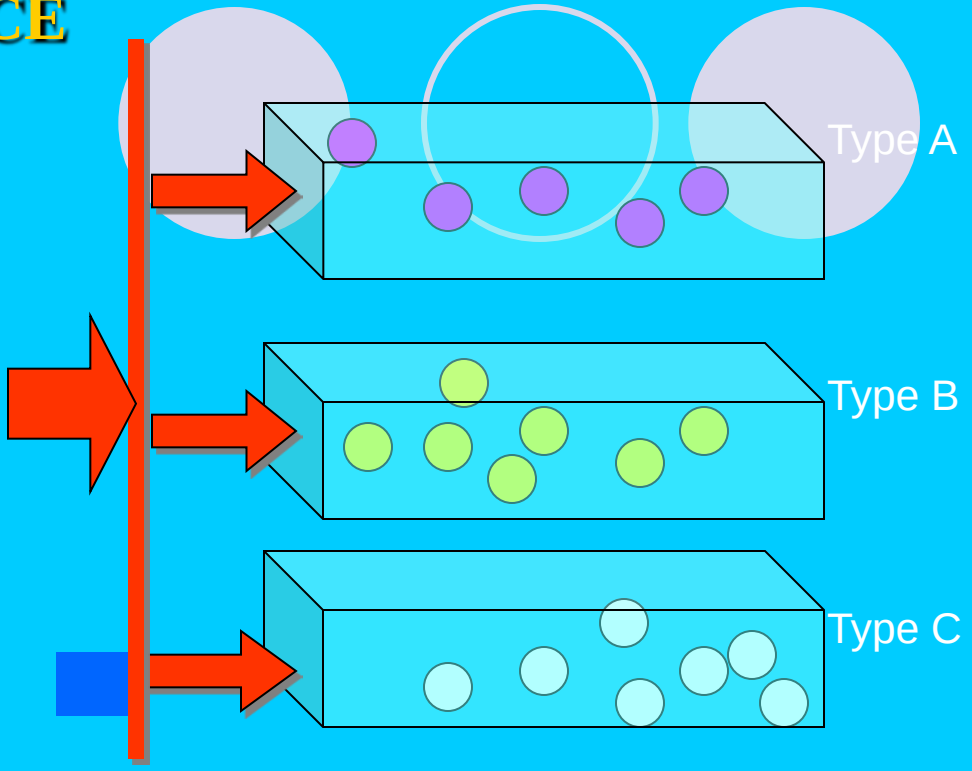
- More usable and understandable structure (e.g. spaghetti code vs structure modules).
- (Almost) essential in large distributed systems where all “subsystems” need to be continually interchanging information to collectively achieve or to maintain some desired state

  (e.g. highly coupled subsystems)

  (cf. human case of multi-disciplinary research team)

DISTRIBUTED INTELLIGENCE

A World of Agents



- An agent is any entity that can send or receive a message.
- An agent can be hardware or software or both (and can be a human)
- An agent can be simple or complex with any desired functionality

What Kinds of Agents?

Knowledge Agent

Expert in an area of knowledge (human or artificial)

Knowledge Server

Artificial agent with major capabilities for storing and retrieving knowledge (eventually reasoning).

Interface Agent

Intelligent assistant.

Coach or Tutor Agent

Intelligent coach or tutor.

Mediator Agent

Coordinates other agents and resolves conflicts.

Knowledge Management Agent

High-level coordination of knowledge activities for an individual or collaborative group (e.g. ‘Mediator’).

Information Search Agent

‘Travelling’ searcher, e.g. knowbots, infobots.

Directory Agent

‘Points’ when queried ‘Where is XXX?’

Mentor Agent

For higher level expertise or strategy.

Autonomous Agents

Hardware + Software

e.g. Autonomous vehicles, robots, etc.

Other Agents

Citation and document retrieval

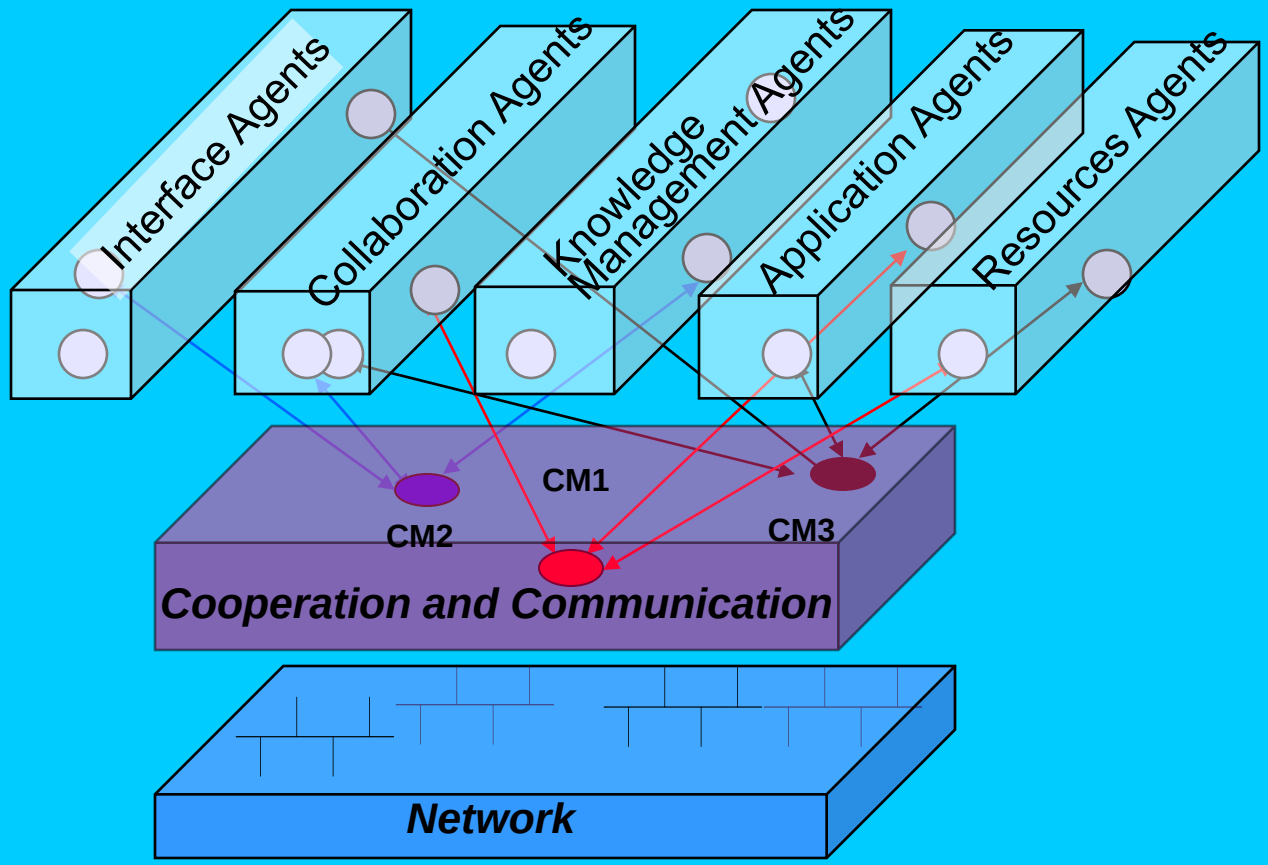
Dictionaries

Atlases (Geographic Information Systems)

etc.

etc.

EXAMPLE OF MULTI-AGENT SYSTEMS ARCHITECTURE



THE COOPERATION-COMMUNICATION LAYER

- URL: [protocol://] [id@] host : port [/path]