G53DIA: Designing Intelligent Agents

Lecture 10: Multi-Agent Systems I

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

Outline of this lecture

- multi-agent systems
- designing multi-agent systems
- example: explorer robots on Mars
- task allocation
- example: contract net protocol
- example: Witness Narrator Agents

Defining “multi-agent system”

- like the notion of an ‘*agent*’, a ‘*multi-agent system*’ is an analysis tool
- it is pointless trying to pin down which systems are *really* multi-agent systems
- the key point is whether we gain by looking at a system as a multi-agent system
- many distributed systems can be viewed as multi-agent systems, but it may not useful to do so

Multi-agent systems

- a *multi-agent system* is a system in which several agents share a common task environment and *cooperate* at least part of the time
- the *environment* may not *appear* the same to the agents if they are different, e.g., if they have different sensors and actions
- the *agents* can have any of the architectures we have seen so far, e.g., reactive or deliberative or hybrid
- all the agents may have the same architecture or they may have different architectures

Interactions in multi-agent systems

- if the agents are not aware of or simply *ignore* each other, there isn't very much interesting to say
- if they always *compete* with each other, it is more interesting, but the agents don't form a *system* in anything other than the ecological sense (e.g., artificial life)
- for a multi-agent system to be possible the agents must *cooperate* about some things – there must be some overlap in their task environments
- e.g., even if the agents compete for resources, they must cooperate about how the resources are to be allocated

Competition & cooperation in MAS

- the balance between competition and cooperation depends on the degree to which the goals of the agents overlap

- e.g., agents representing different organisations in an electronic market will typically have competing goals (to maximise the profit of their organisation)

- however they must cooperate to ensure that the market (e.g., auction) works fairly

- *mechanism design* is concerned with designing interaction protocols in which the agents have no incentive to cheat

Co-operation in multi-agent systems

- agents are *self-interested* and do not share a common goal
  - e.g., they are designed to represent the interests of different individuals or organisations
  - agents co-operate because it helps them achieve their own goals
- agents implicitly or explicitly share a *common goal*
  - benevolently work to achieve the overall objectives of the system, even when these conflict with the agent's own goals
  - e.g., when the agents are 'owned' by the same organisation or individual

Shared goals

- we will focus on the special case in which all the agents in the MAS cooperate to achieve one or more system or *organisational goals*
- the agents co-operate to perform some task that a single agent can't do on its own
  - because a single agent doesn't have all the capabilities or knowledge required to perform the task
  - because a single agent would be too slow
- note that there may still be elements of competition, e.g., if the agents compete for the organisation's resources
- mechanisms are still required to ensure that resources and tasks are allocated appropriately

Applications of multi-agent systems

- *distributed problem solving*
  - each agent has only restricted capabilities or knowledge in relation to the (shared) problem to be solved
  - e.g., scheduling meetings, design of industrial products
- *solving distributed problems*
  - the agents have similar capabilities but the problem is distributed
  - e.g., controlling a communications or energy distribution network

Designing multi-agent systems

- more complex than designing a single agent
- the *types* of agents to use:
  - should they be identical or specialised?
  - how many agents should there be of each type (redundancy)?
- what *architecture*(s) should they have?
- how the agents *communicate* with each other, e.g., by signalling or sending messages

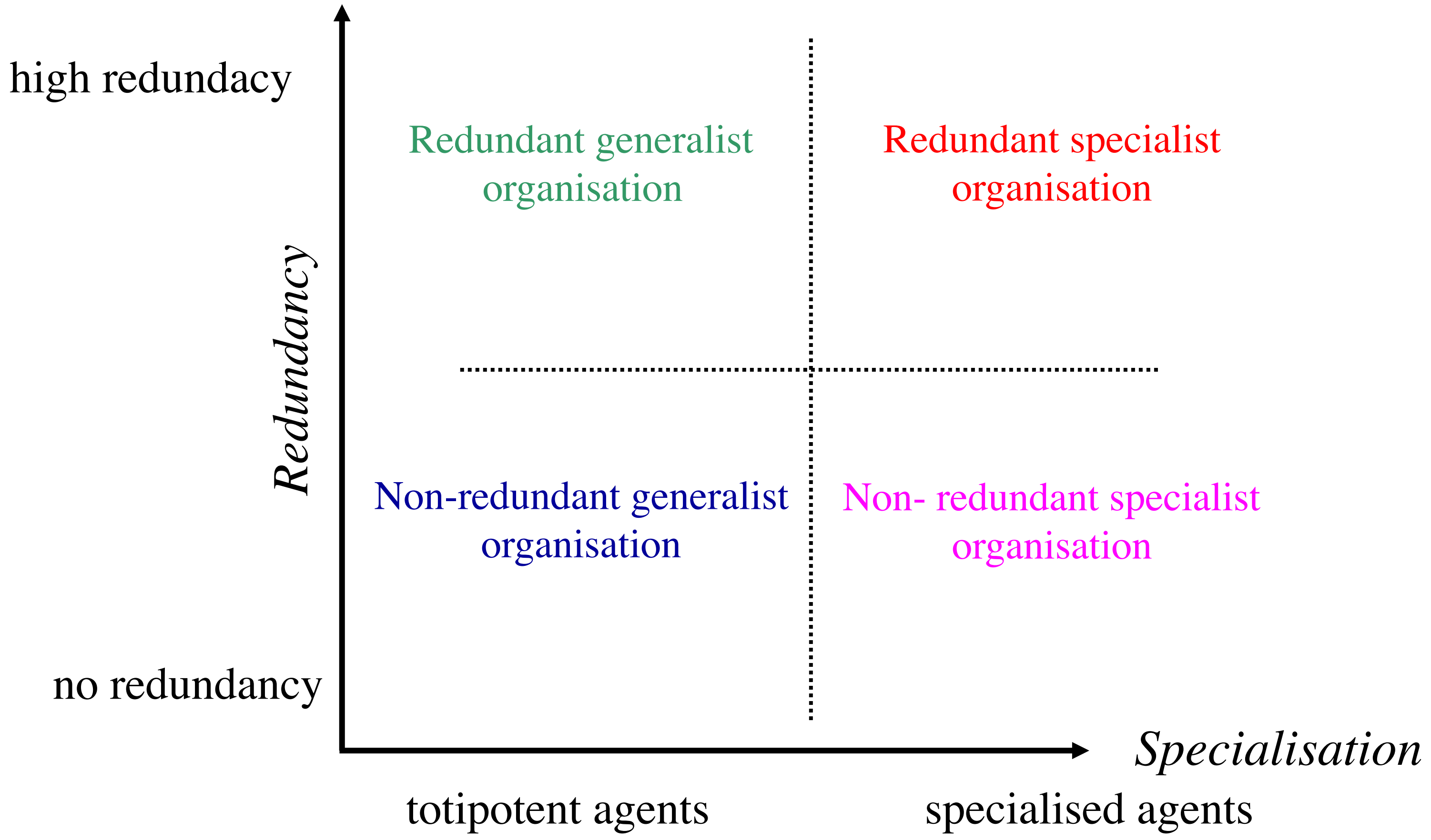
Designing multi-agent systems 2

- what type of organisational structure should be used:
  - *predefined*: relationships are determined in advance by the designer of the system
  - *emergent*: the structure is entirely the result of the interactions between the agents
- how should the organisational structure be implemented:
  - should *control* be hierarchical or distributed?
  - if distributed, what *mechanisms* are there for ensuring co-operation between agents–e.g., sharing tasks and resources, co-ordination of actions, arbitration and negotiation

Specialisation & redundancy

- the degree of *specialisation* indicates the number of actions an agent can perform in relation to the number of actions necessary to perform the task
- the degree of *redundancy* indicates the proportion of agents capable of performing a given action
- for simplicity, we assume that all (basic) actions can be carried out by a single agent

Specialisation vs redundancy



Specialisation vs redundancy

- *non-redundant generalist organisation*: each agent can perform many actions and each action is performed by only a few agents
- *redundant specialist organisation*: each agent can perform only a few actions and each action is carried out by many agents
- *redundant generalist organisation*: each agent can perform many actions and each action can be performed by many agents
- *non-redundant specialist organisation*: each agent can perform only a few actions and each action is performed by only a few agents

Control

- control structure determines the way in which agents can cause other agents to perform certain tasks:
  - *hierarchical structures*: control is organised around a branching tree, with agents nearer the leaves subordinate to those nearer the root of the tree
  - *distributed structures*: any agent can ask any other agent to carry out a task which it may or may not agree to perform

Example: explorer robots on Mars

- from a fixed base several mobile robots explore an unknown environment in order to find and recover ore and transport it back to the base
- the agents must perform three actions to gather ore:
  - find some ore
  - drill down to bring it to the surface
  - transport the ore back to base
- each action can be accomplished independently of the others by a single agent
- robots can be rendered inoperative for various reasons, e.g., being hit by a meteorite, breakdown etc.

Designing the robots

To solve the problem we have to determine:

- the *types* of robots to use: should they be identical or specialised?
- the *architecture(s)* of the agents: should all the agents have the same architecture or should they have different architectures? should they be reactive or deliberative or hybrid?
- the kind of *communication* to use: signals or messages? (this interacts with the architecture(s) of the agents)
- the *co-operation mechanisms* and *interaction protocols* to use: what happens when two robots discover a deposit of ore at the same time?
- the *organisation* of the agents: should they work as a group or on their own? are the teams fixed or dynamic? can agents ask for assistance?

Solution 1

- *hierarchical*, *predefined* organisational structure
- each agent performs a single action (detecting, drilling and transporting), and several agents can perform the same action
- agents are organised into fixed teams with a hierarchical subordination structure
- each detector robot commands a (fixed) set of driller robots and each driller commands a (fixed) set of transporter robots

Solution 2

- *egalitarian*, *predefined* organisational structure:
- robots can be *totipotent* or *specialised*
- each can ask the others for help as the need arises:
  - if a detector agent finds ore, it can relay the position to the other agents
  - if the agents are specialised, then drillers can ask detectors to find some ore
- one way to do this is the *contract net protocol* (later)

Solution 3

- *egalitarian*, *emergent* organisational structure:
- each robot can detect, drill and gather ore on its own
- the system has a great deal of redundancy, but can be inefficient
- to improve performance, the robots can start to specialise while they are working
- e.g., those who have transported ore become more likely to transport ore in the future

Task sharing

- *task sharing* is the problem of determining how tasks are allocated to individual agents in a multi-agent system
- for homogeneous (e.g., totipotent) agents this is straightforward–only concern is load balancing
- if the agents are heterogeneous (have differing capabilities) and/or are autonomous (can refuse tasks), then task sharing involves reaching agreements between agents

Contract net protocol

- *contract net protocol* is a way of achieving efficient co-operation through task sharing in networks of (possibly heterogeneous, autonomous) agents

  – *task announcement*: an agent which generates (or receives) a task broadcasts a description of the task to some or all of the agents

  – *bid response*: agents respond to the task announcement with a bid

  – *task allocation*: the agent which announced the task allocates it to one or more of the bidding agents

  – *expediting*: the agent to which the task was allocated carries it out

Task annoucement

- *task manager* sends a task announcement to some or all agents
- task announcement contains information about the task to be performed:
  - *eligibility specification*: the criteria an agent must meet in order to be eligible to submit a bid
  - *task abstraction*: brief description of the task to allow potential bidders to evaluate level of interest
  - *bid specification*: description of the expected form of a bid for the announced task

Bidding

- on receipt of a task announcement, an agent determines if it is *eligible* for the task based on:
  - the task's eligibility specification
  - the agent's hardware and software resources
  - its current commitments
- eligible agents send a *bid* to the task manager containing the information in the bid specification, e.g., when they will be able to complete the task, how much it will cost, etc.

Task allocation

- bids are stored by the task manager until a deadline is reached
- if no (acceptable) bids are received by the deadline, task is re-announced
- otherwise the manager then awards the task to one or more bidders
- bidders who have be awarded the task confirm that they are still able to undertake it (situation may have changed between bid and award)
- otherwise part or all of the task is re-announced

Task processing

- award messages contain a complete specification of the task to be executed
- successful bidder(s) (contractors) must attempt to expedite the task
- this may result in the generation of new *sub-tasks* which the bidder then manages ...
- when the task is complete, contractors send their manager a report message containing the result of the task

Applications

- contract net has become one of the most popular frameworks for task sharing in multi-agent systems (e.g., FIPA-OS)
  - originally used to allocate tasks over a distribute network of sensors (benevolent agents)
  - later extended to self-interested agents in electronic markets
- many variants—e.g., agents respond with offers of tasks to *swap* for the announced task

Example: Witness Narrator Agents

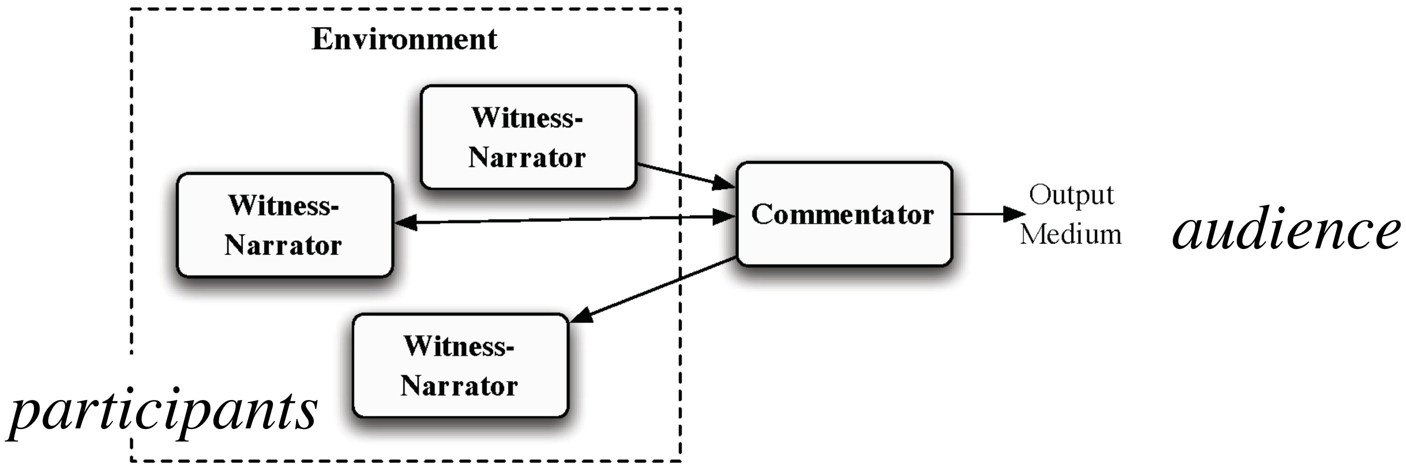
- MMORPGs provide diverse interactive experiences for large numbers of simultaneous participants
- scale of the environment and activities of other participants makes it difficult to involve players in an overarching narrative experience
- interaction with other players precludes the possibility of an omniscient narrator whose story structures the user's experience
- much of the experience is driven by the (unknowable) thoughts and feelings of other players

Witness-narrator framework

- agent-based approach
- agents embodied in the environment generate narrative from observations of participant's actions
- narrative is published to external audiences, e.g., community websites, SMS messages
- or fed back into the environment in real-time to embellish the ongoing experience

Narrative production

- ***participants*** are the subject of the narrative—interact with *witness narrator agents* in the environment
- external ***audience*** are not (currently) embodied in the world but read accounts of the action—interact with non-embodied *commentator agents*
- both participants and audience make requests for information about past, present or future events



Example output

Dragon slain in Etum Castle District

An ancient dragon was slain in Etum Castle District within the last hour. Lance Bannon, a powerful mage, delivered the fatal blow by casting a fireball at the dragon.

It all started when Jim Fingers, a young rogue, attacked the dragon with a sword. The ancient dragon slashed Jim Fingers with its talons. Lance Bannon, a powerful mage, cast invisibility. Oliver Ranger, a fighter, stabbed the dragon with a dagger. The dragon cast a fireball at Jim Fingers. Lance Bannon cast a fireball at the dragon. Finally, the ancient dragon died.

Embodiment & control

- witness narrator agents are embodied in the environment and have (approximately) the same capabilities as a human participant
- participants can determine when they are being observed and the information an agent can obtain given its position
- can also try to avoid agents or modify their behaviour when they are around
- allows participants (some) control over what gets reported
- important when reporting events to an external audience

User requests

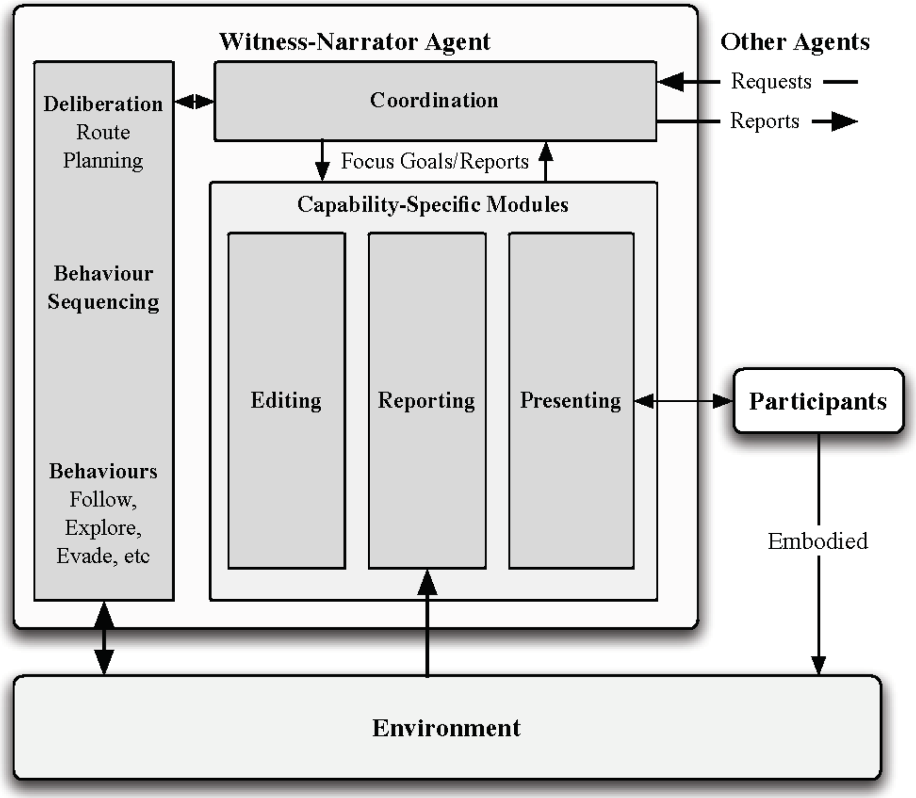
- information requests give rise to *focus goals* which direct the activities of the witness-narrator agents:
  - partial description of events, e.g., “what are my friends doing now”?
  - area of the environment and the time(s) at which the events occur, e.g., “what has happened at this location in the past”?
  - interval specifying how frequently to generate reports
- focus goals determine which events observed by the agents are considered ‘interesting’

Autonomous goal generation

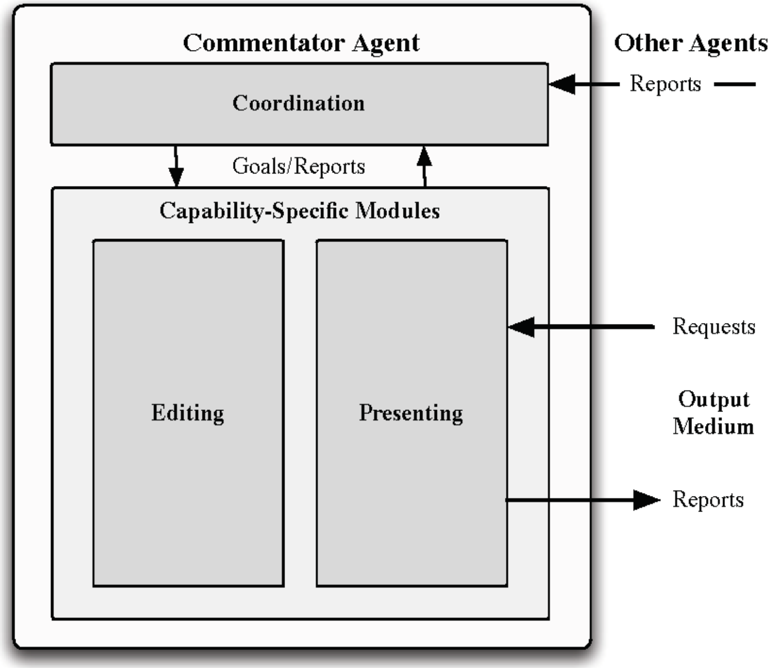
- witness-narrator agents can generate focus goals *autonomously* in response to observed events
  - always refer to current or future events
  - always specialisations of existing focus goals
- WNAs have *a priori* high-level goals which are used as a basis for autonomous goal generation, e.g., death of a player

Agent architecture

Witness-Narrator Agent



Commentator Agent



Agent coordination

- a focus goal specifying past or current events which cannot be satisfied by the agent that generated it is *broadcast* to all WNAs, e.g.
  - "what happened yesterday"
  - "what are my friends doing right now"
- reports matching the focus goal are forwarded to the originating agent

Team formation

- focus goals which specify future events result in the formation of a *team of agents* coordinated by the agent which generated the goal
- coordinator broadcasts a call for participation which includes the focus goal
- agents which can attend to a focus goal at any point during the time it is active will offer to join the team, stating when they are available
- coordinator assigns roles to team members based on a set of ideal role requirements, so as to ensure the maximum coverage of the goal
- team formation is on a best-effort basis

Implementation

- agents are implemented in AgentSpeak (Jason)—each module is a collection of Jason plans and rules
- event ontologies are developed in OWL-DL using Protégé and compiled into Jason rules
- coordination layer builds on Jason's contract net implementation
- also draws on a number of other Jason extensions (multiagent communication, persistent database etc)
- NWN gameserver plugin for sensing

The next lecture

Multi-Agent Systems II

Suggested reading:

- Ferber (1999), chapter 1