19-0: Outline

- What is an agent?
- What are MAS/DAI problems?
- Communication languages
- Coordination
  - Contract Net
  - Distributed Planning
- Societies of agents

19-1: Distributed AI

- Distributed AI (more often called Multiagent Systems these days) is the study of how multiple agents can and do interact with each other.
  - May include both human or computational agents.
  - May be a closed or an open system
  - Agents may be cooperative or self-interested.

19-2: Qualities of an agent

- Autonomy
- Adaptation
- Goal-directed behavior
- Has “beliefs” and “intentions”
- Proactive
- Situated within an environment

19-3: Autonomy

- Autonomy is a quality often attributed to agents.
- An autonomous agent is able to rely on its percepts and past experience to make decisions, rather than asking a human for help.
- This is a thorny area - most agents will not have complete autonomy.
  - When might we not want an agent to have complete autonomy?
- Challenge: Designing an agent that can reason about its own autonomy and know when to ask for help.

19-4: Agent-oriented Programming

- We can also think of agents as a programming paradigm.
  - The next logical step after objects.
  - “Objects do it for free, agents do it for money.”
- Objects are receivers of actions, agents are actors.
- It’s less useful to think of agent as an objective label than as a subjective description.
- Agency is a useful abstraction for us as programmers.
  - Allows us to think about a program at a higher level.
- Treat something as an agent if that helps to understand, predict, or explain its behavior.
  - Thermostats as agents
19-5: Usefulness of the agent metaphor

- Why bother with all this? We already know how to write programs.
- Agents tend to be open-ended programs
  - Difficult to specify in advance what they should do in all cases.
- It's helpful to talk about them as if they were intelligent.
- "The robot wants to find the power supply."
- "The server believes that the client has reset."
This assigning of mental states to programs is called the intentional stance.

19-6: Goals and beliefs and intentions

- Treating a program as an agent lets us work with it at the knowledge level.
- We can program in terms of what an agent 'knows' or 'wants' or is 'trying to achieve'.
  - We need a software layer that gives us this flexibility.
  - See the AI class for more on building this layer.
- This can help us:
  - Manage complexity in our own programs
  - Interact with other programs whose internal operations are opaque to us.

19-7: Why solve DAI problems?

- So a DAI or MAS problem involves a number of agents interacting with each other.
- Why not construct a centralized system?
  - Complexity
  - Geographic or temporal separation of components
  - Separate ownership
  - Dynamic or changing system

19-8: Some examples

- NASA
  - Coordinating Mars rovers
  - Multiple orbiting satellites
- Scheduling
  - Robots in factories
  - Power generation and distribution
  - Troop movement and deployment
- Financial markets
- P2P networks
- etc

19-9: Communication Languages

- Working with agents at the knowledge level creates the need for a communication language and protocol at this level.
  - Built on top of lower-level protocols: HTTP, SOAP, etc.
- Goal: describe a sequence of messages declaratively.
  - What is said, rather than the format of the message.

19-10: Layers of communication

- We can distinguish between several layers at which communication must be specified.
  - Method: this refers to the underlying layer that will be used to transport messages.
    - TCP, HTTP, RMI, UDP, etc.
  - Syntax: this refers to the way in which a message is structured.
    - XML, KQML
  - Semantics: This refers to the meaning of a message, usually as related to other messages.
    - RDF, OWL, ontologies.
  - Pragmatics: This refers to the way that a message is used as part of a larger protocol or dialogue.
    - KQML.
19-11: *Speech acts*

- A common approach in the agent communication language (ACL) community has been to develop protocols based on **speech acts**.
- KQML is an example of this approach.
  - Note: KQML has been largely superseded by the FIPA ACL, but they work similarly.
- Provides a framework which separates the transmission of facts from the representation of those facts.
- Allows agents with different capabilities to implement different subsets of the language.

19-12: *Tell and Ask*

- Most speech-act-based protocols can be divided into two types of communicative actions: **tell** and **ask**.
- These actions are called **performatives**.
  - Tell allows one agent to provide information to another.
  - This may be unsolicited, or in response to an ask.
  - Can be subdivided into inform, explain, confirm, refuse, bid, reply
  - Each of these have their own particular semantics regarding the **commitment** that an agent is making to this communication.

19-13: *Performatives*

- **advertise**
- **ask and tell**
- **deny**
- **evaluate, ask-if, ask-about**
- **request and reply**
- **achieve/unachieve**
- **register/unregister**
- **forward, broadcast**

19-14: *Commitment*

- Commitment is a key concept in agent design.
- Commitment is used (in some cases) to separate **goals** from **intentions**
  - Goals are things you’d like to have
  - Intentions are things you’ve set out to do.
- When an agent makes a commitment to a goal, it becomes an intention.
- Agents may make commitments to other agents, thereby adopting intentions on their behalf. These commitments are binding.

19-15: *Tell and Ask*

- Ask allows one agent to request information or services of another.
- This can be subdivided into request, command, query
- Each message in KQML (or FIPA ACL) has an associated protocol.
- Compliant agents must agree to follow this protocol.

19-16: *Example*

- In FIPA ACL, an agent must respond to a request.
- It can either agree, or refuse.
  - (An organization may choose to place further restrictions on this)
- The receiving agent may also place further conditions on how a task will be accomplished.
- In this example, A asks B to deliver a box. B agrees, but indicates that the task will have a low priority.
19-17: Example

(request
  :sender (agent-identifier :name A)
  :receiver (set (agent-identifier :name B))
  :content
    "((action (agent-identifier :name B)
               (deliver box017 (loc 12 19)))")
  :protocol fipa-request :language fipa-sl
  :reply-with order 567)

(agree
  :sender (agent-identifier :name B)
  :receiver (set (agent-identifier :name A))
  :content
    "((action (agent-identifier :name B)
               (deliver box017 (loc 12 19)))
     (priority order 567 low))"
  :in-reply-to order 567
  :protocol fipa-request :language fipa-sl)

19-18: Points to Emphasize

- Uses a lisp-style syntax.
  - Like most protocols, wrappers exist in most high-level languages.
- The message is structured as a set of key-value pairs in a list, rather than a fixed size and order.
- The language in which the content is represented is independent of KQML or FIPA ACL.
  - Treated as a string to be interpreted.
  - Might be a logical representation, XML/RDF, SOAP, or executable code.

19-19: ACL questions

- The development of languages for agent communication has received a great deal of research attention (and funding!)
- Issues:
  - How many performatives are needed? How rich does communication have to be?
  - How can agents exchange knowledge (ontologies) stored in different formats?
  - What level of interaction is needed in a general agent language?
  - What tradeoffs exist between generality and expressivity? Can domain-specific needs be removed from an ACL?

19-20: Coordination

- Coordination turns out to be one of the most fundamental problems in MAS.
- Coordination can occur because:
  - Agents must work together to solve some larger problem.
  - There is the possibility of conflict.
  - There is the possibility of shared goals.

19-21: Mechanisms for Coordination

- Organizational structure
  - Little communication needed
  - Agents do not need to know about the entire population
  - Structure must be agreed on in advance
  - Can be brittle, or inefficient.

19-22: Mechanisms for Coordination

- Negotiation mechanisms
  - More communication needed
  - Agents do not have to be cooperative
  - Challenges: designing a negotiation protocol that is fair and efficient.
19-23: Mechanisms for Coordination

- Markets and auctions
  - Can deal with relatively large numbers of agents.
  - Agents can be self-interested
  - Under certain conditions, efficient outcomes are reached.
  - If those conditions are not met, outcomes are less predictable.
  - Can include NP-hard problems, which limits the size of the population.
  - Potential expressivity problems.

19-24: Mechanisms for Coordination

- Social norms/laws
  - Designed for Internet-level population sizes
  - Rules are defined for how one “should act.”
  - Violators are (ideally) discouraged from behaving poorly.
  - Challenge: enforcement.

19-25: Cooperative environments

- We’ll start with cooperative environments.
- Distribution due to spatial/geographic reasons or complexity.
- Agents are assumed to tell the truth, and act to achieve a global goal.

19-26: Contract Net

- Contract Net is one of the oldest and most widely-used coordination mechanisms.
  - Simple, easy to implement.
  - Works best in cooperative environments, but can be used in self-interested environments.
  - Used to allocate tasks.
  - Who will agree to do what?

19-27: Contract Net

- Begin with an agent known as the manager
  - Allocates tasks
- A set of agents act as contractors
  - Potentially perform these tasks
- The manager announces a task to (a subset of) the contractors (a request performative)
- Any agent who can complete the task responds with a bid. (a reply, or bid)
- Manager evaluates all bids and assigns the task. (a tell performative)
- Awardee now has a commitment to accomplish this task.
  - It may further contract this task out.

19-28: Example: distributed sensor management

- A set of cameras located around campus can track people
- Cameras each have an area they can see.
- A camera is tracking a person. As the person leaves its line of sight, it announces a bid to continue tracking.
- The first agent to reply wins the contract and continues tracking.
- Since all agents are cooperative, the ‘bidding’ is quite simple.
19-29: What's interesting about contract net?

- Contract net seems very simple; why is it interesting?
- Decisions about how tasks are allocated are made by both the manager and the contractor.
  - Manager cannot force an agent to take on a task.
- Formal bidding models can easily be added onto the basic protocol.
- Works nicely in environments where tasks appear dynamically.

19-30: Distributed Planning

- Contract Net works nicely for environments that have discrete one-shot tasks.
- In many cases, domains are more complex.
- Agents are specialized
- A 'task' may require a sequence of steps
- These steps may be accomplished in several ways.
- This leads into the realm of distributed planning

19-31: Planning in Five Minutes

- A plan is a sequence of operations meant to accomplish a goal.
- The goal is specified declaratively: at(luggage, airport), at(brooks, airport), at(students, airport)
- Actions are ways of accomplishing parts of a plan
- They have preconditions and effects.
  - Preconditions must be true to perform the action
  - Effects must hold after the action is taken
- PutLuggageInTrunk. pre: holding(luggage) effect: in(luggage, trunk)
- Planning is the process of finding a sequence of actions that accomplishes a goal.

19-32: Planning in Five Minutes

- Many algorithms exist for building and repairing plans.
- Issues:
  - Ordering constraints
  - Dealing with failure
  - Adapting to changes in the world
  - Scaling
- A common way to deal with large planning problems is to construct a hierarchical plan.

19-33: Hierarchical Planning

- Many times, aspects of a problem can be solved independently.
- Example: taking a trip to Peru can be decomposed into:
  - Buying tickets
  - Getting everyone to the airport
  - Getting on the plane and flying there.
- I can figure out how to solve each of these problems more or less independently.
- Each subproblem can be represented as an AND/OR graph
- Some decisions made at runtime.
  - Caveat: decisions made in one subproblem may affect possible choices in other subproblems

19-34: Distributed Planning

- Distributed planning comes about when multiple agents (usually with hierarchical plans) must share resources.
  - A communication channel, a bridge, a power supply
- Alternatively, there may be opportunities for synergy.
  - Both agents plan to deliver packages to the same location.
- How can agents synchronize their plans?
19-35: Distributed Planning

- Solution 1: Submit plans to a centralized coordinator.
  - Doesn’t scale
  - Agents may not be willing to share more information than is needed.
- Solution 2: broadcast top-level constraints to each other.
  - This allows agents to detect whether there is a top-level conflict.
  - Plans will either be totally serialized or totally parallel.

19-36: Distributed Planning

- A better solution:
  - Detect whether there is:
    - No problem: all possible interactions may be interleaved.
    - No solution: plans must be serialized.
    - Some solution: We then ‘step down a level’ in the plans and force agents to commit to particular alternatives.
  - Tradeoff: Deeper level requires more communication and interleaving (an exponential problem), but produces finer-grained coordination.

19-37: Societies of agents

- Contract net and distributed planning work for tens of agents.
- How can we govern environments with thousands (or more) agents?
- These are often referred to as agent societies
  - Still a research area
  - Inspiration drawn from human society, Internet-scale protocols.

19-38: Societies of agents

- Research in this area can be divided into descriptive and proscriptive domains:
  - Descriptive: “Given a structure or behavior on the world, what is the outcome?”
  - Proscriptive: “If a structure or behavior is enforced, what outcomes result?”
- There is also a vigorous debate about whether participants in an Internet-level agent society should be treated as self-interested, cooperative, or a mix of the two.
  - Cooperation potentially allows for more beneficial outcomes, if participants can be trusted.
- Many of the same issues as P2P systems arise.

19-39: Example

- The agent society approach can be used to construct teams of agents, each with very simple behavior, who can collectively solve a difficult task.
- Ant algorithms
  - Problem: Explore an unknown area and locate high-resource areas

19-40: Rules

- Avoid obstacles.
- If you are not holding a resource, wander randomly. If you sense ‘pheromones’, weight random selection towards them.
- If you find resources, pick them up and begin dropping pheromones. Follow a beacon back home.
- If you make it home, drop the resource.
- Over time, pheromone paths are built up between the home and the resource.
19-41: Issues

- Achieving macro-level behavior from microlevel rules.
  - How do you guarantee outcomes? Is there an efficient way to synthesize these sorts of rules?
- Imposition of social norms or laws
  - What outcomes can be guaranteed for a given set of norms or laws? What language is necessary to describe norms or laws?
- Mechanisms for trust and reputation
  - How can noncompliance be enforced?

19-42: Summary

- Multiagent problems arise due to spatial, privacy or scaling constraints.
- General languages for declarative communication are needed in open systems.
- Coordination is a fundamental problem
  - Contract net
  - Distributed planning
  - Ant Algorithms

19-43: Next time

- Lying liars and the lies they tell.
- or,
- What to do when agents are out for themselves.
- Applying economics to coordination and allocation problems.