INTRODUCTION TO MULTIAGENT SYSTEMS, 2012

If you read any of my work before, you already know the drill: this document is a best-effort, brief, inconclusive summary of the course's material that I compiled for my own use. If it happens to help you, that's great, but don't blame me for missing something. Comments are welcome at *sashag@cs*.

This text is organized in the same order the material was presented in class. However, I used the second edition of Wooldridge's "Introduction to Multiagent Systems" book, so the book's chapters are not presented in order.

WOOLDRIDGE, CHAPTER 1: INTRODUCTION

Trends in computing:

- **Ubiquity** – processors everywhere
- **Interconnection** – networks of billions of processors are possible
- **Intelligence** – tasks delegated to computers have increasing complexity
- **Delegation** – we trust computers with safety-critical tasks
- **Human-orientation** – abstractions make HMI more approachable to humans

Challenges for agents:

- Operate **independently**
- Represent our **best interests**
- **Cooperate**, **negotiate** and **reach agreements**

An **agent** is a computer program capable of independent action on behalf of its owner. Figures out what it has to do on itself, does not have to be told. A **multiagent system** consists of several interacting agents that cooperate, coordinate and negotiate.

The field of multiagent systems deals with **agent design** (micro) and **society design** (macro). The field borrows from:

- **Software engineering** – interaction as key in complex software
- **Economics** – self-interested entities interacting, but computational aspects paramount
- **Social sciences** – agents help in simulating human societies, but agents are computational
- **Distributed systems** – but agents are more autonomous and self-interested
- **Artificial intelligence** – but don't need all of AI to build agents, and need social aspects

WOOLDRIDGE, CHAPTER 2: INTELLIGENT AGENTS

An agent is a **computer system** situated in some **environment**, capable of **autonomous** action to meet its delegated **objectives**.
[Trivial examples: thermostat, daemon; better examples: auto-pilot, nuclear reactor control system.]

An agent consists of:

- **Sensors** – perception of the world
- **Deliberation** logic based on **objectives** and **percepts** that leads to **action**
- **Effectors** – influence the environment

Agents must be prepared for **failure** because they have incomplete control of their environment.
Classification of environments:

- **Accessible** vs. **inaccessible** – whether all information is available to the agent
- **Static** vs. **dynamic** – changes without the agent's control
- **Deterministic** vs. **non-deterministic** – whether there is uncertainty about action effects
- **Episodic** vs. **non-episodic** – history matters
- **Discrete** vs. **continuous** – finite number of actions/percepts

Additional challenges:

- Agents are **reactive** (harder to program)
- Agents often make **long-term** considerations (e.g. resource allocation fairness)
- Agents often must make decisions in **real-time**

AN INTELLIGENT AGENT

- **Reactive** – perceives and responds to the environment in a timely fashion
- **Proactive** – takes the initiative to satisfy its goals
- **Social** – interacts with agents/humans to satisfy its goals (negotiation, cooperation &c.)

Agents must strike a **balance** between reactive and proactive behavior; must be **flexible**.

Agents are not objects:

- Agents **request** actions from each other, can **refuse**
- Agents are **autonomous**, **flexible**
- Agents are **active** (concurrent, thread of execution per agent)

Agents are not expert systems:

- Expert systems are not situated in an **environment**
- Expert systems are not capable of **reactive/proactive** behavior
- Expert systems do not have **social ability**

Intentional stance: describe agents using **mental states** – **beliefs**, **desires**, **intentions**:

- Useful when the system is sufficiently complex and its structure is not completely known
- Not useful if we have a simpler, mechanistic description

- An **abstraction mechanism** for managing complexity

FORMAL MODEL

- **Environment states** *E*
- **Agent actions** *Ac*
- **Run** – sequence of environment states and actions
- **State transformer function** τ – maps a run that ends with an agent action to a set of possible environment states (non-deterministic)
- **Environment** *Env* – triple of *E*, initial state, τ
- **Agent** *Ag* – maps runs that end with an environment state to an action (deterministic)
    - **Purely reactive agent** – function from *E* to *Ac* (no history)
    - **Stateful agent** – maintains state, the function is from state to *Ac*
- Set of runs of agent in environment R(*Ag*, *Env*) – terminated runs only

TASK SPECIFICATION

Utility functions specify tasks for agents. Some alternatives:

- Gives utility value to **environment states** (very short-term)
- Gives utility value to **runs** (very long-term), possibly with decay

The **optimal agent** in an environment maximizes the expected utility. **Bounded optimality** introduces the constraints of a given machine into the equation.

It's often hard to reason in terms of utility. Instead, can use **predicate task specifications** that map a run to either 1 or 0. Two common types of tasks:

- **Achievement task** *G* – in every run, bring about one (or more) of the states in *G*
- **Maintenance task** *B* – in every run, stay away from all the states in *B*

Synthesizing agents for task environments: take an environment and task specification and generate an agent. Ideal synthesis is both **sound** and **complete**.

FRANKLIN, GRAESSER: AGENT OR PROGRAM

Some properties by which agents are defined:

- **Autonomous** execution oriented towards **goals**
- Domain-oriented **reasoning**
- **Perceiving** its environment and **acting** upon it
- Situated in a **dynamic, complex environment / real-world**
- **Persistent** in its agenda
- **Delegation** – acts on behalf of a user
- **Social, reactive, proactive** *[see above]*
- Capable of **negotiation** and **coordination**

- **Learning, adaptive, flexible**

Definition: a system situated within an **environment** that **senses** it and **acts** upon it, pursuing an **agenda** so as to **effect** what it senses in the future.

Agents can be **classified** based on these properties – control mechanism, environment, mobility, reasoning, purpose &c.

WOOLDRIDGE, CHAPTER 3: DEDUCTIVE REASONING AGENTS

Symbolic AI – maintain a symbolic representation of the environment and behavior and manipulate it syntactically. In **deductive reasoning agents**, the symbols are **logical formulae** and the manipulation is **theorem proving**.
Problems:

- **Transduction** problem – translate the real world into a symbolic description
- **Representation/reasoning** problem – represent information symbolically and reason on it within reasonable time constraints

PURELY LOGIC-BASED AGENTS

- Try to deduce some action *a* to perform – **prove Do(*a*)** – from the internal database
- If nothing is deduced, try to find an action *a* whose negation ¬Do(*a*) **cannot be deduced** from the internal database
- Perform an action and modify the state of the database accordingly
- Repeat

The agent specification is **directly executed** – transparent, elegant.

Objections – entirely **impractical**:

- First-order logic theorem proving is **undecidable**
- **Number of formulas** is immense
  - **Frame axioms** – work very hard to specify what does *not* change
- **Calculative rationality** – recommended action is for the environment when the deliberation began, which might have changed
- Not obvious how to map environment to symbolic formulae (e.g. images) – **transduction**
- Not obvious how to represent temporal information – **representation**

AGENT-ORIENTED PROGRAMMING [SHOHAM]

Shoham: an agent is any entity **described using mental terms** such as beliefs, capabilities, choices, commitments.
Planning to provide:

- **Formal language** for describing mental state
- **Programming language** for defining agents (LISP-based)

- **Agentifier** – converting neutral programs into agents (not implemented)

Agent-oriented programming embeds **beliefs**, **decisions/commitments** and **capabilities** in agents as programming language constructs. These **mental constructs** are used to **design** (**specify**) the system, not just analyze it.

AGENT0 agents:

- **Capabilities** – $CAN_i^t \varphi$
- Initial **beliefs** - $B_i^t \varphi$, initial **commitments** - $OBL_{i,j}^t \varphi$
- **Commitment rules** – message condition, mental condition (beliefs), action (commitment)
  - Interpreted at runtime, do not require theorem proving
  - To ensure easy belief consistency checking with new facts, facts are very restricted

All of these have a **temporal component** – "*i* believes that at time *t*, *b* will believe…" &c.

AGENT0 supports private (internal) and public **messages**: `request`, `unrequest`, `inform`, `refrain`. Sending messages and acting upon them is **constrained** by the agent's mental state, e.g.:

- **Consistency** of beliefs and obligations
- **Commit** only to what you are **capable** of
- **Awareness** of obligations
- By-default **persistence** of beliefs and obligations

PLACA – subsequent work that introduces the ability to plan and communicate via high-level goals; similar to AGENT0.

CONCURRENT METATEM

Supports direct execution of **temporal logic formulae** in multiple agents **concurrently**. **Concurrent MetateM** agents:

- **Interface** – ID, messages accepted, messages emitted
- **Computational engine** – program rules of the form *past* → *present/future*

Rules are based on **temporal logic operators** such as ~*open*(*door*) U *request*(*you*), meaning the door is not open until you request it, or ⦿*request*(*you*) → ◊*open*(*door*), meaning that if you just requested it, the door will open at some point in the future.

Agent execution revolves around constructing **proofs of satisfiability** for the rules.

SHOHAM: AGENT-ORIENTED PROGRAMMING

[This paper is described in the preceding section.]

WOOLDRIDGE, CHAPTER 4: PRACTICAL REASONING AGENTS

Practical reasoning consists of figuring out what to do – it is directed towards **action** and not theoretical beliefs. Consists of:

- **Deliberation** – what state of affairs to achieve (goals, intentions)
- **Means-ends reasoning** – how to achieve them (plans, actions)

DELIBERATION

Properties of intentions:

- Intentions are **stronger than desires** – they lead to action, although can be overridden
- Intentions **persist over time**, you try again if you fail
- Intentions **can be dropped** if unrealistic or if goals change
- Intentions **constrain future reasoning**, e.g. incompatible intentions
- Intentions **influence future beliefs**, e.g. for reasoning post-achievement of the intention
- Agents must believe their **intentions are achievable**
- Agents will not adopt as intentions something they believe to be **inevitable**
- Agents need not intend all possible consequences of their intentions – **package deal**

Deliberation consists of:

1. **Option generation** – produces new desires from beliefs and intentions
2. **Filtering** – produces new intentions from beliefs, desires and intentions
3. **Belief revision** – produces new beliefs from existing beliefs and perceptions

PLANNING

Means-ends reasoning (planning) produces from beliefs and intentions a plan of concrete actions.

STRIPS planner:

- **World** modeled as first-order logic formulae
- **Action schemata** available to planner – name, preconditions, delete list, add list
- **Initial state**, **goal** – planner produces actions from initial state to goal

STRIPS works using a **goal stack** – this is limiting (also in PRS), can't pursue multiple goals at once: must finish with a certain goal before moving to a sibling.

Plans can be generated from a **plan library** – for each plan, see if it brings about the intentions required.

PRACTICAL REASONING AGENT

1. Generate **beliefs**, **desires**, **intentions** from world perception
2. Generate **plan** from beliefs, indentions, actions available
3. **Execute** first step of the plan
4. **Revise** beliefs from world perception
5. If it's worthwhile [*] to **reconsider** the intentions, **repeat deliberation**

   - **Blind commitment** – maintain intention until it's achieved
   - **Single-minded commitment** – maintain intention until it's achieved / impossible
   - **Open-minded commitment** – maintain intention as long as it's believed possible
6. If plan not sound, **re-plan**
7. If plan done, got to step 1; if plan not done, go to step 3

[*] Constant deliberation is expensive, but blindly executing in a changing world is expensive, too – **cheap meta-level control** determines whether it's worthwhile to reconsider: **bold** vs. **cautious** agents in a world with varying rate of environment change (static/dynamic environment).

BDI LOGIC

Path quantifiers: *A x* means "on all paths, *x* is true", *E x* means "on some paths, *x* is true".

BDI connectives: (Bel *i x*), (Des *i x*), (Int *i x*).

Axioms of **KD45**:

1. $Bel(p \rightarrow q) \rightarrow (Bel\ p \rightarrow Bel\ q)$
2. $Bel\ p \rightarrow \neg Bel\ \neg p$
3. $Bel\ p \rightarrow Bel\ Bel\ p$
4. $\neg Bel\ p \rightarrow Bel\ \neg Bel\ p$
5. If p and $p \rightarrow q$, then q
6. If p is a theorem, then so is $Bel\ p$

CTL* notation:

1. A = for every path
2. E = there exists a path
3. G = globally (on all states on the path)
4. F = future (on some state on the path)

BDI logic:

1. **Belief-goal compatibility** – $(Des\ \alpha) \rightarrow (Bel\ \alpha)$
2. **Goal-intention compatibility** – $(Int\ \alpha) \rightarrow (Des\ \alpha)$
3. **Volitional commitment** – $\big(Int\ does(a)\big) \rightarrow does(a)$
4. **Awareness of goals and intentions** – $(Des\ \phi) \rightarrow \big(Bel\ (Des\ \phi)\big)$, $(Int\ \phi) \rightarrow \big(Bel\ (Int\ \phi)\big)$
5. **No unconscious actions** – $done(a) \rightarrow Bel\big(done(a)\big)$
6. **No infinite deferral** – $(Int\ \phi) \rightarrow AF\big(\neg(Int\ \phi)\big)$
   - The agent will drop the intention or act on it

IRMA – INTELLIGENT RESOURCE-BOUNDED MACHINE

IRMA has a **plan library** and **explicit** representations of **beliefs**, **desires** and **intentions**.
Consists of:

1. **Reasoner** – inference engine

2. **Means-ends analyzer** – which plans might be useful
3. **Opportunity analyzer** – generates new options from the environment
4. **Filtering process** – which options are compatible with intentions
5. **Deliberation process** – which intentions are best to adopt

PROCEDURAL REASONING SYSTEM (PRS)

First **BDI architecture**, applied to real-world multiagent systems.

Planning relies on **BDI database** and a **plan library** constructed by the programmer. Plans in PRS have **goals** (post-conditions), **contexts** (pre-conditions) and **bodies** (actions).

Actions can consist of sub-goals to be achieved, loops. Planning starts with a single top-level goal. Deliberation between competing options is done using **meta-level plans** (or utilities for plans).

PRS executes plans using an **intention stack** – no branching, which means only one goal can be pursued at a time. *[This is limiting, current AI uses graph search algorithms instead.]*

BRATMAN, ISRAEL, POLLACK: PLANS AND RESOURCE-BOUNDED PRACTICAL REASONING

When dealing with **deliberation** and **means-ends reasoning**, it is critical to remember that agents are **resource bounded**. The longer the agent deliberates, the more likely the deliberation results to become **stale**.
Mechanisms for limiting reasoning time:

- **Existing plans** – constrain means-ends reasoning to a clear purpose, narrow deliberation, influence future reasoning (the agent is **committed** to doing what it plans)
- **Partial plans** – shorter deliberation, less need to reevaluate every detail

WOOLDRIDGE, CHAPTER 5: REACTIVE AND HYBRID AGENTS

Reactive agents position, contrary to symbolic AI:

1. Symbolic representations and deductive reasoning are not necessary
2. Intelligent behavior is product of **interaction with the environment**, situated in the world
3. Intelligent behavior **emerges from simpler behaviors**

REACTIVE AGENTS

Advantages:

1. **Computationally cheap** – constant-time decision making instead of intractable deduction
2. **Simple to build**
3. **Robust to failure**

Disadvantages:

1. Difficult to model a **learning** process, long-term view
2. Designed for a **specific environment**, hard to adapt
3. "Emergent" means we **don't understand it well**, no methodology
4. Exponentially harder to build when there are **dozens of layers**

SUBSUMPTION ARCHITECTURE

Brooks' **subsumption architecture** proposes that intelligent behavior emerges without explicit symbolic representations and reasoning.

The system is constructed in **layers** of task-accomplishing behaviors, simple rules of the form *situation* → *action*. Layers can **suppress/replace** inputs or **inhibit** outputs of other layers (e.g. collision avoidance will inhibit output of exploration layer if collision is imminent).
Layers are constructed as **levels of competence**; for a mobile robot:

0. Avoid contact with objects
1. Wander aimlessly around
2. Explore the world by reaching distant places
3. Build a map and plan routes
4. Notice changes in the environment
5. Identify and reason about objects
6. Formulate and execute plans for changing the world state
7. Reason about the behavior of objects and modify plans

Advantages:

- Building the system in layers means a **working system** is available when level 0 is done
- Layers can work in **parallel**
- Failure, errors or slowness at higher levels does not cause the entire system to fail – it merely **degrades**

SITUATED AUTOMATA

Agents are **specified** in a logic of **beliefs** and **goals**, but **compiled** to a **digital circuit** that does no symbolic manipulation. Agents consist of **perception component** and **action component**.

1. **RULER** builds the perception component:
    - Semantics of agent's input bits
    - Set of static facts
    - Set of world state transitions
    - Semantics of perception output bits
    - Generates digital circuit with the correct semantics
2. **GAPPS** builds the action component:
    - Goal reduction rules
    - Top-level goal
    - Generates digital circuit that realizes the goal

The primary problem is around the compilation process, which can be extremely expensive or even theoretically impossible.

OTHER IDEAS

PENGI idea – most decisions encoded into an efficient low-level structure (digital circuit) and periodically updated for new problems. *[PENGI is a simulated computer game of a penguin chased by bees.]*

Competence modules (**agent network**) – modules provide **relevance** in a particular situation; the higher the relevance, the more the module affects the agent's behavior in that situation. Modules are interconnected, similarly to a neural network.

HYBRID AGENTS

Organize the system into layers: some layers capable of **reactive behavior**, some layers capable of **proactive behavior**. Control flow:

- **Horizontal layering** – each layer connected to sensory input and action output, means there are many complex interactions and mediation required
- **Vertical layering** – sensory input and action output dealt with by one layer (each), simpler; can be **one-pass** or **two-pass**

TOURINGMACHINES

Horizontally-layered architecture with three layers:

- **Reactive layer** – *situation* → *action* rules such as obstacle avoidance
- **Planning/proactive layer** using a library of skeleton plans to pursue goals
- **Modeling layer** – generates goals for the planning layer, models other agents and the world

A **control subsystem** decides which layer should have control of the agent – **censorship rules** for inputs (inhibition) and outputs.

INTERRAP

Vertically-layered two-pass architecture with three layers:

- **Behavior-based layer** – reactive behavior
- **Local planning layer** – plans to achieve goals
- **Cooperative planning layer** – deals with social interactions with other agents

Each layer has a **knowledge base** of corresponding information – raw sensory input, plans, models of other agents, &c. – depending on layer.

Bottom-up activation – lower layer passes control to higher layer because it doesn't know what to do. **Top-down execution** – higher layer uses lower layer to execute goals. Input and output are linked only to the lowest layer.

BROOKS: A ROBUST LAYERED CONTROL SYSTEM FOR A MOBILE ROBOT

[This paper is discussed in the preceding section.]

WOOLDRIDGE, CHAPTER 11: MULTIAGENT INTERACTIONS

Agents have multiple and sometimes overlapping **spheres of influence**, and are **linked** by hierarchical and other relationships.

Simultaneous actions by agents on the environment result in a certain **outcome**. Agents have preferences, expressed as **utility functions** over **outcomes**.

The **relationship between money and utility** is complicated: sometimes plenty of money brings little utility; **risk**-related factors also affect the "value" of money vs. utility.

The utility functions introduce a **preference ordering** over outcomes: $\omega \succcurlyeq_i \omega'$ if $u_i(\omega) \geq u_i(\omega')$. Agent preferences can be characterized over **combinations of actions** – e.g. $(C, C) \succcurlyeq_i (C, D) \succ_i (D, C) \succcurlyeq_i (D, D)$.

Payoff matrices characterize games in a standard fashion:

Game 1

	i defects	*i* cooperates
j defects	4 4	1 4
j cooperates	4 1	1 1

Game 2

	i defects	*i* cooperates
j defects	3 2	1 2
j cooperates	4 1	3 3

In **strictly competitive** games, agent preferences are diametrically opposed – an outcome preferred by one agent is feared by another. In **zero-sum** games, the sum of utilities for every outcome is 0.

SOLUTION CONCEPTS

Dominant strategy – a strategy *s* is dominant for player *i* if no matter what player *j* chooses, *i* will do at least as well with that strategy. This is the obvious choice, when it exists.
[In Game 1, (D,D) is a dominant strategy.]

Nash equilibrium – strategies *s*, *t* for players *i*, *j* respectively are in Nash equilibrium if (a) under the assumption *i* plays *s*, *j* can do no better than play *t*, and (b) under the assumption *j* plays *t*, *i* can do no better than play *s*. There might be 0, 1, or several NEs.
[In Game 2, (D,D) is a Nash equilibrium.]

Mixed strategies – introducing an element of randomness: play *s* with probability *p*, play *t* with probability *q*, &c. There is always a NE with mixed strategies.
[In rock-paper-scissors, playing each with probability ⅓ is a strategy that is in NE with itself.]

Sub-game perfect equilibrium – convert the game to **extensive form**, where players move sequentially, in order; a sub-game perfect equilibrium is a NE in every sub-game.
(Or: **work backwards** from all the ends of the game, identify best responses, and move backward to previous player's decision, etc.)

Pareto optimal/efficient – an outcome is Pareto efficient if no other outcome improves some player's utility without hurting some other player. Pareto inefficient outcomes are wasteful in terms of utility – usually we steer clear of them.

Maximizing social welfare – an outcome maximizes social welfare if the sum of gained utilities is maximal over the outcomes. When all players have the same **owner**, this is ideal.

PRISONER'S DILEMMA

This game has the following payoff matrix:

Prisoner's Dilemma

	i defects	*i* cooperates
j defects	2 2	0 5
j cooperates	5 0	3 3

Defection is a dominant strategy for both players, and (*D*,*D*) is the only Nash equilibrium. However, this outcome is not Pareto efficient and does not maximize social welfare – **utility is wasted**.

In an **infinite iterated tournament**, the shadow of punishment encourages cooperation, and the rational behavior is to cooperate.

In a **finite iterated tournament** when the number of rounds is known, defection is still the dominant strategy (backwards induction argument). This can be "fixed" by making the number of rounds unknown in advance.

AXELROD'S TOURNAMENT

1980 tournament: programs competed against each other in Prisoner's Dilemma games, 200 rounds each.
Some strategies:

- **Random** – choose *C* or *D* at random with probability ½
- **All-D** – always defect
- **Tit-for-tat** – cooperate initially, on subsequent rounds do what your opponent did last

Tit-for-tat won the tournament – the key is that it **interacted** with **like-minded strategies** that favored cooperation. *[Interestingly, it also won the second tournament!]*
Characteristic "rules":

1. **Don't have to beat your opponent to do well**
2. **Don't be first to defect** – "nice" programs that didn't defect first fared much better
3. **Reciprocate cooperation and defection** – punish & forgive quickly, but don't exaggerate
4. **Don't be too clever** – makes it hard for others to recognize you and cooperate

Program equilibria – submit a program that plays the game. Programs can compare themselves to each other. In Prisoner's Dilemma – equilibrium formed by "cooperate if the other program is identical to yours, otherwise defect".

OTHER SYMMETRIC 2×2 GAMES

Stag hunt – $(C,C) \succ_i (D,C) \succ_i (D,D) \succ_i (C,D)$ – there are two NEs, with mutual defection and mutual cooperation.

Game of chicken – $(D,C) \succ_i (C,C) \succ_i (C,D) \succ_i (D,D)$ – there are two NEs: if you believe your opponent is going to drive off the cliff, then you should steer away, and vice versa.

AXELROD: THE EVOLUTION OF COOPERATION (CH. 1-3)

Cooperation emerges from the norm of **reciprocity**:

- It can only emerge if the players might meet again – "the **shadow of the future**"
- Typically, a **discount parameter** is applied to payoff from future interactions

Ecological approach to Axelrod's tournament: if the representation of better strategies grows with their success, the "non-nice" programs become extinct and tit-for-tat grows faster than any other.

Evolution of cooperation:

1. Can evolve even in a world of unconditional detection where there are **small clusters** of cooperating individuals
2. Reciprocity can **thrive** in a world of many different strategies
3. Cooperation based on reciprocity can **protect itself from invasion** by other strategies

Invasion: a strategy is invaded by a new strategy if the newcomer gets a higher score with a native than a native gets with another native. *[Idea: the natives will then begin adopting the newcomer's strategy, until the old strategy is evolutionally eliminated.]*

A **collectively stable** strategy cannot be invaded by any other strategy:

- **Tit-for-tat is collectively stable** if the discount parameter is large enough
- If the discount parameter is sufficiently small, any strategy which may be the first to cooperate can be invaded by All-D
- A nice strategy that is collectively stable must retaliate for its opponent's very first defection
- **All-D is always collectively stable**

Invasion in clusters: for a sufficiently large **cluster of newcomers** with a new strategy, new results are possible:

- **All-D can be invaded** by a sufficiently large cluster of tit-fot-tat
- A **nice** strategy that cannot be invaded by one newcomer cannot be invaded by any cluster of newcomers

WOOLDRIDGE, CHAPTER 14: ALLOCATING SCARCE RESOURCES

Desired properties of **mechanism design** (for auctions, negotiation &c.):

- **Convergence/guaranteed success**
- Maximizing **social welfare** – there is no "wasted" utility
- **Pareto efficiency**
- **Individual rationality** – no agent is asked to give up a better outcome
- **Stability** – no one has incentive to defect
- **Simplicity**
- **Distribution** – can be executed in a distributed system, no single point of failure

An **auction** is a mechanism for **reaching agreement** on **allocating scarce resources**.
Characterization of auctions:

- **Public/private value** – whether the item's value is inherent to it, or specific to the bidder + **Correlated value** – depends on what other bidders estimate the item, as well
- **Winner determination** – highest bidder, &c.
- **Payment** – first-price, second-price, everyone pays what they bid
- **Open cry/sealed-bid** – whether agents see each other's bids
- **Bidding mechanism** – one-shot, multiple rounds: ascending, descending

AUCTION TYPES AND STRATEGIES

	Payment	Visibility	Mechanism	Strategy	Notes
English	first-price	open cry	ascending	Bid a small amount more than the current bid; when bid reaches your private valuation, withdraw	Winner's curse: should I be worried because no one else was willing to pay that much?
Vickrey	second-price	sealed-bid	one-shot	Bid exactly your private valuation	Subject to antisocial behavior

Dutch		open cry	descending		Winner's curse
Sealed	first-price	sealed-bid	one-shot	Bid somewhat less than your true valuation	

Expected revenues for the auctioneer:

- **Risk-neutral bidders** – identical in all four types of auctions
- **Risk-averse bidders** – Dutch and first-price sealed-bid leads to higher expected revenue (bidders are willing to pay slightly more than their valuation)
- **Risk-averse auctioneers** – Vickrey and English auctions

LIES AND COLLUSION

All protocols discussed are subject to **bidder collusion**. Collusion can be prevented by modifying the protocol so that bidders can't identify each other.

All protocols discussed are subject to **auctioneer manipulation** (e.g. lie about second highest bid in Vickrey auction). Lies can be prevented by using a trusted third party or signing bids.

Shills (bogus bidders) are a problem in English auctions.

[Sections 14.3 and 14.4 skipped, as they were not discussed in class and are not in the first edition.]

WOOLDRIDGE, CHAPTER 15: BARGAINING

- **Negotiation set** – space of possible **proposals** agents can make
- **Protocol** – legal proposals as a function of prior negotiations (also, whether the negotiation is one-to-one, many-to-one, or many-to-many)
- **Strategies** – usually private, but proposals are usually seen by other parties
- **Rule** – for determining a deal, what it is, and the **conflict deal** if no agreement reached

[When multiple issues (for negotiation) are involved, the problems are considerably more complicated. The chapter examines only single-issue, symmetric, one-to-one negotiation.]

DOMAIN THEORY

Task-oriented domains: agents want to redistribute tasks.

- Postmen delivering letters.
- Clerks sending faxes.

State-oriented domains: agents plan to reach acceptable final states, scheduling.

- Moving blocks to reach a certain configuration (blocks world).

Worth-oriented domains: utility function rates acceptability of states, agents plan to maximize it.
[Note: negotiation is not over a single issue, but over states and how to reach them – plans, goals.]

- Pushing tiles into holes of different value (tile world).

ALTERNATING OFFERS

The protocol: make offers, in turn, until someone accepts. If not accepted – go for conflict deal.

Assuming that the conflict deal is the worst option, when the number of rounds is fixed, the **player in the last round has all the power** – the other player would rather accept than go for the conflict deal.

The same applies to an infinite number of rounds: using the strategy **"always propose that I get everything and reject anything else"** is in NE with accepting that proposal. If you're convinced that is what your opponent is doing, you should accept.

To model impatient players, use a **discount factor** for each agent – the value of the good is reduced by the discount factor with each round. In a fixed or infinite number of rounds, there is still a NE strategy – the more patient the agent, the "bigger slice" he gets.

Negotiation decision function – use a function that tells you how much to bid in every round. Can decrease price quickly or slowly, &c.

TASK ALLOCATION

Formal model:

- Agents negotiate a **set of tasks** T that can be redistributed
- There is a **cost function** c which determines the cost of executing each subset of tasks $c(\emptyset)=0$ and **monotonic**
- The **conflict deal** Θ consists of the tasks originally allocated to the agents
- They try to reach a better deal, where the **utility of the deal** for agent i is the difference between the cost of his original tasks and the cost of his tasks in the deal

Criteria for deal selection:

- A deal d **dominates** another deal e if it is not worse than e for any agent and better than e for at least one agent (in terms of utility)
  - We prefer non-dominated deals, which are **Pareto optimal**
- A deal is **individual rational** if it weakly dominates the conflict deal
  - If there is no such deal, it makes no sense to bargain at all
- The **negotiation set** is the set of individual rational and Pareto optimal deals

MONOTONIC CONCESSION PROTOCOL

1. Both agents propose deals d_1, d_2 from the negotiation set
2. If $u_1(d_2) \geq u_1(d_1)$ or $u_2(d_1) \geq u_2(d_2)$, the deal is accepted
   The exceeding deal is selected; if both exceed, the deal is selected at random
3. Next, no agent is allowed to make a proposal that is less preferred by its opponent
4. If no concession is made, negotiation terminates with the conflict deal

The protocol terminates, but can proceed for an exponential number of rounds. Also, it isn't clear how to start and how to concede.

Zeuthen strategy:

- First proposal – the agent's **most preferred deal**
- On subsequent rounds – the agent with the **smaller risk concedes**
  - Agent *i*'s risk for offer (δ_i, δ_j) is $\left(u_i(\delta_i) - u_j(\delta_j)\right)/u_i(\delta_i)$ – utility lost in conflict
  - If risk is equal, flip a coin to decide who concedes
- Make the **smallest concession necessary** to change the balance of risk, repeat

This protocol terminates with a **Pareto optimal deal**, but still may proceed for an exponential number of rounds. The Zeuthen strategy is **in NE with itself**, which makes it a good "open" protocol.

[Weakness: this requires the agents to know the opponent's risk, and hence its agreement space.]

DECEPTION

Phantom task – pretend you have been allocated a task that you have not; **decoy task** – a phantom task that you can produce on demand, so it is also **verifiable**.

Hidden task – pretend that you have not been allocated a task that you have.

Sub-additive TOD – if for sets of tasks X, Y it is true that $c(X \cup Y) \leq c(X) + c(Y)$.

- In all-or-nothing deals, there is no incentive to hide a task.
- In all-or-nothing or mixed deals, if there is a penalty for discovering a phantom, there is also no incentive to hide a task.
- Decoy tasks are always beneficial.

Concave TOD – for sets of tasks Y, Z and $X \subseteq Y$ it is true that $c(Y \cup Z) - c(Y) \leq c(X \cup Z) - c(X)$. In other words, the cost Z adds to X is more than the cost it adds to Y, when X is a subset of Y.

- In all-or-nothing deals, there is no incentive to lie at all.

Modular TOD – for sets of tasks X, Y it is true that $c(X \cup Y) = c(X) + c(Y) - c(X \cap Y)$.

- In all kinds of deals, there is no incentive to create phantom/decoy tasks.
- There is still incentive to hide tasks in pure or mixed deals.

[Section 15.4 skipped, as it was not discussed in class and does not appear in the first edition.]

WOOLDRIDGE, CHAPTER 16: ARGUING

Argumentation is about **agreeing what to believe** – a rationally justifiable position, resolving inconsistencies between the beliefs of multiple agents.

Why argumentation?

- Justifying a position can be **more powerful** than mere negotiation

- Argumentation allows for **understanding why** an agreement was reached
- Argumentation models **position changes**, e.g. preferences

Types of argument between humans:

- **Logical** mode – deductive, explicit, rational: "If $A \rightarrow B$, and A, then B"
- **Emotional** mode – appeals to feelings: "How would you feel if that were done to you?"
- **Visceral** mode – physical aspect: stamp your foot or pound on the table
- **Kisceral** mode – appeal to the intuitive, religious: "This is against Christian teaching!"

ABSTRACT ARGUMENTS

Abstract arguments can be described as a directed graph, where an edge (a, b) means that a **attacks** b. If c attacks a in this case, then we say c **defends** b.

A set of arguments (**position**) is **admissible** if it is:

- **Conflict free** – no argument in it attacks another, and
- **Mutually defensive** – every attacked element is defended by some element in it

An admissible position is a **preferred extension** if every superset of it is inadmissible.

A **grounded extension** is computed as follows:

- Begin with **"in"** arguments – those that don't have any attackers
- Eliminate arguments attacked by them – they are **"out"**
- Continue until there are no changes in the graph that remains – that's the GE

DEDUCTIVE ARGUMENTS

Deductive arguments are pairs (S, C) where S is in the database of logical formulae known to be true, C can be proved from S, and S is consistent and minimal.

(Γ_1, φ_1) **attacks** (Γ_2, φ_2) in the following cases:

- **Rebuttal** – $\varphi_1 \equiv \neg\varphi_2$
- **Undercut** – $\varphi_1 \equiv \neg\psi$ for some $\psi \in \Gamma_2$

[We tend to prefer arguments that can be undercut to arguments that can be rebutted, because undercuts can be "fixed" by choosing a different support set.]

A **dialogue** between agents is a sequence of arguments that cannot be repeated and that attack one another. It ends when no further moves are possible.
Types of dialogues:

- **Persuasion** – convince the other party
- **Negotiation** – get the best deal for yourself
- **Inquiry** – find a "proof" for a theoretical problem
- **Deliberation** – influence outcome of reaching a decision

- **Information seeking** – spread knowledge, gain knowledge
- **Eristics** – strike the other party (in a conflict), settlement is not the goal

Persuader is a system for labor negotiation, exchanging proposals and counter-proposals over multiple issues (wages, pensions, &c.):

- Determines **violated goals** and tries to find **compensating actions** and generates **arguments**
- Models **argument severity** – e.g. an appeal to status quo is preferred to threats or prevailing practice

JENNINGS, FARATIN, LOMUSCIO, PARSONS, SIERRA, WOOLDRIDGE: AUTOMATED NEGOTIATION: PROSPECTS, METHODS AND CHALLENGES

Because agents are **autonomous**, **negotiation** is central to multiagent systems – an agent has to **convince** other agents to act in a particular way.
Topics in negotiation:

- **Negotiation protocols** – messages, parties, decisions
- **Negotiation objects** – number of resources, dynamic resources
- **Decision making model** – how agents achieve their objectives within the protocol

[Also, it is unclear how users instruct agents to negotiate – attitude, autonomy, duration &c.]

General model:

- Negotiation is a **distributed search** through a space of potential agreements
- Each agent has a **space** in which it is willing to make agreements
- Agents make **proposals** within their space and move the boundaries of their space
- Agents make **counter-offers** or **critiques** of proposals made to them
- Agents may **argue** to justify their position or persuade others (threats, rewards, appeals) *[Argumentation adds overhead, which means it's not always suitable.]*

Heuristic methods:

- Based on **realistic** assumptions
- Agent designers can use **less constrained models** of rationality (than game theory)
- Decision making is based on **tactics** that modify proposals heuristically
    - Include **tradeoff** (modifying the value of the offer)
    - Include **issue manipulation** (adding/removing issues to the negotiation set)
- Might select **sub-optimal** outcomes
- **Hard to predict** actual behavior under varying circumstances

Argumentation is usually combined with negotiation – stop negotiating, go to a round of argumentation, go back to negotiating, &c.

WOOLDRIDGE, CHAPTER 12: MAKING GROUP DECISIONS

In a **voting** setting, a set of agents vote over a set of **candidates**. Can either choose one candidate, or provide a **complete ordering** of their preferences over all candidates.
Then, use voting procedures:

- **Social welfare function** – combines all preferences to an ordering of the candidates
- **Social choice function** – combines all preferences to choose a single candidate

VOTING METHODS

Plurality voting – the winner is the candidate that appears first in the largest number of preference orders.

- Simple to **implement**, easy to **understand**
- Might select an outcome when **another outcome is preferred** by the majority
- Subject to **misrepresenting your preferences** to manipulate the outcome
  (e.g. vote for Bibi because you know Yehimovitz won't win and you don't want Liberman)

Sequential majority – series of pairwise elections, the winner proceeds to the next round; can be organized in a list or a tree (like the World Cup).

- **Depends on the order** in which candidates face each other!
- Can be **manipulated**

Majority graph – directed graph of candidates, an edge (a, b) means a would beat b in direct competition. If the graph is a cycle, we can fix the order so that anyone can win!

Condorcet winner – overall winner in any possible order of pair-wise elections (there is an edge from w to every other node in the majority graph).

- Seems **fair**, easy to compute

Borda count – for each preference tuple of size k, the first candidate gets k-1 points, the second k-2, and so on; the last candidate gets 0.

- Takes into account **all the preference information**, not just the first element

Slater ranking – choose ordering that needs the minimal number of fixes to be consistent with the majority graph (e.g. when the graph is a cycle, there are many orderings that need just one fix).

- **Computing** the Slater ranking is NP-hard

Copeland – a's score is the number of candidates a beats in pairwise election; select the candidate with the highest score.

Kemeny rule – produces a ranking that minimizes the total number of disagreements compared to voter preferences (summed over all voters).

Plurality with run-off – select top two by plurality voting, and then match selected pair.

Black's procedure – if there's a Condorcet winner, choose it; otherwise, use Borda count.

Single transferable vote (STV) – election proceeds in rounds, in each round voters choose the highest ranking candidate and the candidate with least points is eliminated.

Dodgson's rule – elect candidate with minimum number of exchanges between adjacent candidates needed to make it a Condorcet winner.
[Somewhat similar to Slater ranking.]

CRITERIA FOR VOTING METHODS

Pareto condition – if every voter ranks *w* above *u*, the voting method should not choose *u* (should not prefer *u* to *w*).
[Sequential pairwise voting violates this criterion.]

Condorcet winner condition – if there is a Condorcet winner, the voting method should choose it (prefer it to other candidates).
[Borda count violates this criterion.]

Smith's generalized Condorcet criterion – if the candidates can be partitioned into sets *A* and *B* such that every candidate from *A* beats every candidate from *B* pairwise, the rule should not elect a candidate from *B*.

Independence of irrelevant alternatives (IIA) – if the relative ranking of *w* and *u* is not changed (e.g. $w \succ_i x \succ_i u$ changes to $x \succ_i w \succ_i u$), the outcome should still rank *w* and *u* in the same way.
[This is a strong requirement that we might want to lift – e.g. Borda count does not satisfy it.]

Monotonicity – if *x* is a winner under a voting rule and one or more voters change their preferences in a way favorable to *x*, then *x* should still be the winner.
[Plurality with run-off violates this criterion.]

Dictatorship – only one voter's preferences influence the outcome.

IMPOSSIBILITY THEOREMS

Arrow's impossibility theorem: for more than two candidates, any voting procedure satisfying the Pareto condition and IIA is a dictatorship.

Manipulable voting procedure – given the preferences $\omega_1, \dots, \omega_n$ and voting function *f*, the voting function is manipulable if there exists ω_i' such that $f(\omega_1, \dots, \omega_i', \dots, \omega_n) \succ_i f(\omega_1, \dots, \omega_i, \dots, \omega_n)$.

Gibbard-Satterthwaite theorem: for more than two candidates, any voting procedure satisfying the Pareto condition that is not manipulable is a dictatorship.

- Hopefully, finding the appropriate manipulation is computationally intractable
- For second-order Copeland, manipulation is NP-complete (in the worst case, of course)

MANIPULATION TECHNIQUES

- Voters reveal **false preferences** – esp. to manipulate second-preference
- **Center** (running the election) removes candidates, adds candidates, adds voters
- A **coalition** cooperates to manipulate – fortunately, this is NP-hard for most voting rules

SINGLE PEAKED PREFERENCES

- Each voter chooses a **point** to build a grocery store, wants it close to his house.
- **Nonmanipulable** solution: choose the **median** point.

WOOLDRIDGE, CHAPTER 13: FORMING COALITIONS

- **Coalition** – a subset of all the agents.
- **Grand coalition** GC – the set of all agents.
- **Characteristic function** v – assigns to each coalition a payoff to distribute to its members.
- A coalition is **stable** if no member of it can do better by **defecting**.
  - Checking stability is computationally hard.
- An **outcome** for a coalition C is a distribution of $v(C)$ to C's members that is:
  - **Feasible** – does not exceed $v(C)$, and
  - **Efficient** – distributes all of $v(C)$.
- The **core** is the set of feasible distributions of payoff to the grand coalition that **no sub-coalition can object to**
  - A coalition C **objects** to an outcome for GC if some outcome for C makes all members of C strictly better off
  - It is better for them to **defect** and work on their own
- The core is non-empty ↔ the grand coalition is stable
  - Note that outcomes in the core might be **"unfair"** – but no member (or set thereof) can do better on their own

SHAPLEY VALUE

- **Pre-imputation** – efficient payoff vector (does not waste utility).
- **Imputation** – individually rational pre-imputation (gets no less than it could get alone).

Shapley value – the better an agent performs, the higher its reward.
Axiomatically defined:

- **Symmetry** – agents that make the same contribution get the same payoff
- **Dummy player** – a player that contributes to any coalition only what it would make on its own should get exactly that
- **Additivity** – a player does not gain or lose from playing more than once

Shapley value of i is his **marginal contribution** to **any order** in which any coalition was formed:

$$sh_i = \frac{1}{|GC|!} \sum_{\pi \in Perm(GC)} \left[v(C_i(\pi) \cup \{i\}) - v\left(C_i(\pi)\right) \right]$$

…where $C_i(\pi)$ is the set of all agents that appear before *i* in the permutation π.

REPRESENTING THE CHARACTERISTIC FUNCTION

Naïve representation is **exponential** in the number of agents.
Alternatives:

- **Weighted subgraph** – represent game as **weighted graph**, $v(C)$ is the weight of the edges whose components are in C
- **Marginal contribution nets** – represent game as a set of **rules**, *pattern* → *value*, where the pattern is a conjunction of agents such as $i \wedge j \wedge \neg k$
- **Weighted voting game** – see below

[Weighted subgraphs and weighted voting games are incomplete representations; marginal contribution nets are complete, but can have exponentially many rules. Note that in all cases but weighted voting games, checking whether the core is empty or whether a payoff is in the core are hard – NP, co-NP and similar classes.]

VOTING GAMES

The characteristic function is onto "win", "lose" – $v: 2^{Ag} \rightarrow \{0,1\}$; alternatively, there is a set W of **winning coalitions**.

Weighted voting games – each agent has **weight** w_i; if the sum of weights in a coalition exceeds a **quota** q, the coalition wins.

Shapley-Shubik power index – Shapley index adapted to weighted voting games. For agent *i*, it is the fraction of times *i*'s vote is decisive in any permutation of the votes.

- In the game <8; 5, 5, 3> all members have equal power, ⅓, because it takes any two members to decide the vote.
- In the game <8; 5, 3, 1> the member with weight 1 has no power, it is never the decisive vote. Permutations: (5, 3, 1), (5, 1, 3), (3, 5, 1), (3, 1, 5), (1, 3, 5), (1, 5, 3) – both 5 and 3 have voting power of ½, they are each pivotal half the time.
- In the game <8; 5, 3, 3> - permutations: (5, 3, 3), (5, 3, 3), (3, 5, 3), (3, 3, 5), (3, 3, 5), (3, 5, 3) – 5 is pivotal 4 times, each 3 is pivotal once; 5 has weight ⅔ and the 3's have weight ⅙ each.

k-weighted voting games – there are multiple voting games with different weights, and a coalition has to win all of them to win the total game (e.g. Senate + Congress voting on a law, or sub-committee approval followed by the general body voting).

[Sections 13.4.2, 13.5, 13.6 skipped, as they were not discussed in class and do not appear in the first edition.]

CONITZER: MAKING DECISIONS BASED ON THE PREFERENCES OF MULTIPLE AGENTS

A **language** is required for **expressing** voter preferences over **multiple outcomes** (e.g. type of meal, time of meal, cost of meal). **CP-net** is such a language – specifies dependent preferences.

Envy-freeness criterion – every agent prefers the tasks allocated to it than any other agent's bundle. (This still might not be Pareto efficient, and deciding whether such an allocation exists is NP-hard.)

Kidney exchanges – finding **cycles** of sequentially compatible donors for transplants; NP-hard when maximal cycle length is set, and greater than 2.

Combinatorial auction – multiple resources are sold, and bidders indicate a value for **every subset** of the resources. Again, expressive bidding languages are required to prevent exponential explosion of bids, e.g.:

- **XOR language** – ({*a*}, 5) XOR ({*b, c*}, 10) – *a* is worth 5, {*a, b*} is worth 5, {*b*} is worth nothing, {*b*, *c*} is worth 10, {*a*, *b*, *c*} is worth 10
- Winner **determination** becomes NP-hard and inapproximable
- Can proceed in **multiple rounds**, in which case we strive to **minimize communication**

Donations – build a framework that **conditions** a donation on the total amount donated (this makes donors feel better about their donation).

- Can also condition the donation on the total amount donated by a certain **closed social group**, making an even stronger stimulus.

Prediction market – trading securities that pay out if an event occurs; then the price of the security represents the probability of the event (in the audience's eyes).

- The combinatorial explosion when multiple outcomes are involved complicates things.

WOOLDRIDGE, CHAPTER 7: COMMUNICATING

Agents aren't objects – one agent **can't force another** to perform an action. It can, however, **communicate** as to influence another agent's actions.

Austin's **performative verbs**: request, inform, promise &c.

Austin's **speech acts** [1]:

- **Locutionary** – the act of making an utterance that is grammatically correct and meaningful
- **Illocutionary** – action performed by saying something, its meaning (request, inform &c.)
- **Perlocutionary** – effect of the act (action, knowledge, inspiration, fear &c.)

Searle's **speech act conditions**:

- **Normal I/O conditions** – hearer is able to hear, there is no interference &c.
- **Preparatory conditions** – request only things the other party can do, &c.
- **Sincerity conditions** – don't ask for things you don't mean, don't speak untruthfully, &c.

Searle's **types of speech acts**:

- **Representatives** – informing of something
- **Directives** – requesting something
- **Commissives** – promising something
- **Expressives** – thanking for something
- **Declarations** – declaring war

PLAN-BASED THEORY

Speech acts have **preconditions** and **postconditions** based on **beliefs**, **abilities** and **wants** (adapted from STRIPS planner).
Examples:

- Request – preconditions:
  - Speaker **believes** hearer **can** perform the action
  - Speaker **believes** the hearer **believes** he **can** perform the action
  - Speaker **believes** he **wants** the action to be performed
- Request – postconditions:
  - Hearer **believes** the speaker **believes** the speaker **wants** the action to be performed

COMMUNICATION LANGUAGES

Knowledge query and manipulation language (KQML) – envelope format for messages, does not concern the content.

- **Metadata parameters** – sender, receiver, ontology &c.
- **Performative** – `ask-one`, `broadcast`, `tell` &c. (from a predefined set of 41)
- **Content parameter** – message content

Criticism:

- **Performative set** is fluid, ad hoc, making interoperability difficult
- **Transport mechanisms** not precisely defined
- **Semantics of performatives** not rigorously defined
- **Missing commissives** (for making promises, required for coordination)

Knowledge interchange format (KIF) – representation of knowledge about some domain, requires ontology to be meaningful.

FIPA Agent Communication Language (ACL) – defines outer language for messages with 20 performatives and allows any language for the content.
[Actual implementations exist, e.g. Java framework called JADE.]

- Performatives well-defined, rigorous
- Messages and performatives must comply with beliefs, desires and uncertain beliefs to be sent – **feasibility condition**
  - Example: `inform` requires that the agent believes what he informs, and believes that the recipient does not already know it

- Example: `request` requires the agent believes the recipient is capable of the action, and believes the recipient has not yet performed it

[Section 7.2.3 skipped, as it was not discussed in class and does not appear in the first edition.]

COHEN, PERRAULT: ELEMENTS OF A PLAN-BASED THEORY OF SPEECH ACTS

Speech acts are **operators** (actions) that have **preconditions**, **effects** and **bodies** – their effects are on the models speakers and hearers maintain. *[Models – beliefs, intentions &c.]*

A **plan-based theory** takes an initial set of **beliefs** and **goals** and leads to the **generation** of plans for speech acts to issue.

- **Belief** is modeled as an operator *a* Believe(*P*) with **axioms** somewhat similar to KD45.
- **Want** (desire, goal) is modeled as an operator *a* Want(*P*).
- Speech operators have **preconditions** – `cando` and `want`, begin with "speaker believes".
- Speech operators have **effects** – begin with "hearer believes".
- **Planning** involves checking preconditions, inserting sub-goals (e.g. to discover certain information), and so forth.

Description of `request` operator *[see also in the preceding section]*:

- Preconditions:
  - `cando` precondition: *S* Believe(*H* cando *ACT*) ∧ *S* Believe(*H* Believe(*H* cando *ACT*))
    Later refined to just "*H* cando *ACT*", makes composition more flexible
  - `want` precondition: *S* Believe(*S* want *request-instance*)
- Effects:
  - *H* Believe(*S* Believe(*S* want *ACT*))

Description of `inform` operator:

- Preconditions:
  - `cando` precondition: *S* Believe(*P*)
    Later refined to just "*P*"
  - `want` precondition: *S* Believe(*S* want *inform-instance*)
- Effects:
  - *H* Believe(*S* Believe(*P*))

Further operators are defined to plan **wh-questions**: `informref`, which is used to inform another agent; and **yes/no questions** – `informif`. Issuing a **request to inform** is the way of asking such a question.

Finally, **side effects** from operators must also be modeled – e.g. after receiving a `request` the hearer knows that its preconditions hold (on the speaker's side).

WOOLDRIDGE, CHAPTER 8: WORKING TOGETHER

Unlike distributed systems, agents have **different** (possibly contradictory) **goals** and must **dynamically coordinate** their activities.

Cooperative Distributed Problem Solving (CDPS) – the entire system is designed to solve a problem together, common interest (benevolent), but no single node has enough **knowledge**, **expertise**, or **processing power** to solve alone.

- This is **not just parallelism** – in parallelism, nodes are homogenous and "stupid"

Coherence – how well the system behaves as a unit: efficiency, solution quality, performance.

Coordination – no extraneous activity, little need of communication, no interference.

TASK AND RESULT SHARING

- **Decompose** the big problem into sub-problems – may involve several levels
- **Solve** the sub-problems (possibly by different agents) – may involve sharing of information
- **Synthesize** an answer from the sub-solutions – again, may be hierarchical

Task sharing involves **decomposition** and **allocation** of tasks to agents. If agents are self-interested, this may involve negotiation/auctions.

Result sharing involves **sharing information** proactively or reactively.

Contract Net protocol – task sharing:

1. Manager **announces** task – broadcast, multicast or point-to-point
2. Contractors **bid** for tasks (based on the task's **marginal cost** for them)
3. Manager **awards** task to some contractor
4. Contractor **reports** task completion to the manager

[Adaptations: retry if no one bid, revise the announcement e.g. with costs/requirements, decompose the problem in a different way. Another variation is for contractors to announce availability instead of the manager announcing tasks.]

Result sharing makes it possible to:

- **Cross-check** solutions, improving confidence
- Achieve a **better global view** by combining local views
- Increase the **precision** of other agents' solutions
- Derive a result more **quickly**

FELINE – a system of coordinating experts in distinct knowledge areas:

- Each agent maintained **skills** and **interests** (hypotheses requiring inquiry) for itself and its acquaintances in the network
- Messages were of type **request**, **response** and **inform**
- When evaluating hypotheses, agents checked their acquaintances recursively and broadcast information to interested agents

BLACKBOARD

First implementation – **Hearsay II** speech understanding system, required cooperation between multiple nodes (**knowledge sources** – KS) in a distributed system.

Result sharing via **blackboard** – shared data structure:

- Multiple agents can read/write, requires **locking** at multiple levels of granularity
- **Subscribe/notify** over blackboard supported
- Possible **hierarchy**
- Possibly **multiple dimensions**
- Can be a considerable **bottleneck**
- Can be **distributed** to resolve bottleneck – information distribution (blackboard data), processing distribution, control distribution (scheduling, activation)

Blackboard is modeled as a **production system**, *precondition* → *action*, executing in **parallel**. Nodes maintain **local contexts** (history), which are updated by the blackboard when events occur.

INCONSISTENCIES

Inconsistencies arise w.r.t. **beliefs** (information) and **intentions** (goals), and are **inevitable**.

- Contract Net – **do not allow** inconsistencies to occur, centralized management
- **Resolve** inconsistencies through bargaining
- **Degrade gracefully** in the presence of inconsistency

FA/C – functionally accurate/cooperative, do the best you can with partial information:

- Problem solving progresses **opportunistically** and **incrementally**
- Agents exchange **high-level results**
- There are many **redundant** ways of arriving at the solution

COORDINATION TYPOLOGY

- **Positive relationship** between activities
  - **Requested** (explicit ask for help)
  - **Non-requested** (implicit opportunistic cooperation)
    - Only one agent performs an action another intended as well
    - One agent's actions as a side-effect achieve another agent's goals
    - One agent's plan makes another's goals easier to achieve
- **Negative relationship** between activities
  - **Incompatibility** of desires
  - **Resource** bottleneck
    - Consumable resource
    - Non-consumable resource

PARTIAL GLOBAL PLANNING

Agents exchange information to reach common conclusions – **partial** planning because a plan for the entire problem is impossible, **global** planning because agents exchange local plans and construct bigger plans from them.

1. Each agent generates **short-term plans**
2. Agents **exchange information** about plans and goals
3. Agents **alter local plans** to better coordinate

Partial global plan:

- **Objective** – system larger goal
- **Activity map** – what agents are doing and what results to expect
- **Solution construction graph** – how to interact and what information to exchange in order to generate a result

Techniques for coordination:

- **Sharing local information** – all, some, per request
- **Sharing results** – to subscribers, broadcast to everyone, promised to someone
- **Redundancy** – random selection when redundancy detected
- **Handling rescheduling** for negative and positive relationships

JOINT INTENTIONS

A group of agents has a **joint commitment** to the overall goal – this commitment is **persistent**. A commitment has a **social convention** – when it can be dropped, how to behave, &c.

An agent that believes the goal is **impossible** or the **motivation** for it is no longer present starts persuading others that this is the case, to drop the **joint persistent goal (JPG)**.

ARCHON – a system that has explicit rules modeling cooperation, such as "if the goal is satisfied then abandon all related activities and inform others".

TEAMWORK MODEL

1. **Recognition** – some agent recognizes a potential for cooperation w.r.t. one of its goals.
2. **Team formation** – the agent recruits others to help, and they agree on the joint goal.
3. **Plan formation** – the team plans together (possibly negotiating) how to achieve the goal.
4. **Team action** – the agents execute the plan (see JPG).

MUTUAL MODELING

MACE – a system that introduces **acquaintance models**, modeling other agents: class, name, role, skills, goals, plans. This helps agents coordinate.

NORMS AND SOCIAL LAWS

These **conventions** dictate agent **behavior** in certain situations, make it easier to **plan** and **coordinate**.

Offline design – norms are **hardwired** into the agent. Obviously, not everything can be predicted and the dynamic nature of systems calls for greater **flexibility**.

Emergence – conventions emerge from within a group of agents interacting with each other. **Strategy update functions** dictate how agents modify their norms:

- **Majority** – change to another strategy if observed majority of agents following it
- **Majority with communication** – exchange experiences/memories with other agents
- **Majority with success** – communicate success to other agents
- **Highest cumulative reward** – use the strategy that resulted in highest payoff to date

Criteria: **memory restart frequency**, efficiency and **time to convergence**.

PLANNING AND SYNCHRONIZATION

- **Centralized planning** – **master** agent develops and distributes a plan to the **slaves**
- **Distributed planning** – a group cooperates to form a **centralized** plan, they do not necessarily execute it
- **Distributed planning for distributed plans** – a group cooperates to form **individual** plans and coordinate their activities; negotiation may be required

Merging plans:

- Use generalized STRIPS notation (pre-, post-conditions) and a **during** list – conditions that must hold **during** the action's execution
- **Interaction analysis** – identify how plans interact with each other (**dependencies**)
- **Safety analysis** – identify **harmful** interactions
- **Interaction resolution** – apply **mutual exclusion** to unsafe plan interactions

Iterative planning – instead of centralized merging, agents generate **successively more refined plans** and interact with each other if necessary (e.g. if interference is detected).

MOBILE AGENTS

Mobile agents execute on a host on behalf of someone far from it on the network – good for minimizing **latency**, low-**bandwidth** networks, &c.
Problems:

- **Security** of host – sandboxing &c.
- **Integrity** of agent – encryption &c.
- **Heterogeneity** of hosts – requires interpreted/ubiquitous languages

Types of mobile agents:

- **Autonomous** – decide when and where to go, e.g. TELESCRIPT language has `go` instruction, agents can meet at a host and cooperate, have limited resources and travel permits
- **On-demand** – executed when the host demands them (Java applet)
- **Active mail** – piggy backs on email, scripts executed as part of email parsing

DURFEE: DISTRIBUTED PROBLEM SOLVING AND PLANNING

Why **distributed problem solving**?

- **Speedup** of problem solving thanks to parallelism
- Distributed **expertise** – diverse capabilities (actual expertise or location-based &c.)
- Distributed **data** – minimizing communication and centralization
- Acting on **results** in small groups – again, minimizing communication and centralization

Challenges in task distribution:

- **Backtracking** makes it hard to hierarchically divide tasks – need to think of communication
- Parallelization might compromise solution **accuracy**
- Parallelization **potential** might be **limited** by the problem domain
- Sub-problems might have very **different sizes** and require **different expertise**
- The **number of tasks** produced by the decomposition might be >> the number of agents
- The actual decomposition might be very **time consuming**

Alternative to result-sharing blackboard: **assign an agent to each resource** and have it process the contending demands for the resource. Contention issues are settled before any subsequent work is performed – this avoids useless work later.

Reduce communication by using the task decomposition structure – e.g. agents close to each other in a mapping effort communicate, or in general use the **organization structure** for communication:

- Requests for information communicated only to some agents
- Partial results communicated only to some agents
- Degree of skepticism regarding received results/information