Agents and Multi-Agent Systems Notes

Vishnu

Tuesday 14th January, 2020

Contents

1 What is an Agent?

1.1 Basic Definition

An agent is an *autonomous* system situated within an environment, capable of *flexible* behaviour to achieve a set of goals.

Autonomous Can control its own state and behaviour

Flexible Is both:

> **Reactive** Responds to changes in and constantly interacts with the environment
>
> **Proactive** Works towards future goals - keeps track of current and goal states
>
> Finding a balance between reactive/proactive is an ongoing problem

1.1.1 Environment and Multi-Agent Domains

Agents are *situated* within environments, and experience/affect them via sensors/actuators.

These environments may be *multi-agent*, so an agent has to consider other agents while achieving goals. Some goals require co-operation, so an agent requires *social ability* to interact with other agents.

1.2 Agents vs. Objects

Objects (as in OOP) encapsulate a state, communicate with others, and have methods/actions. Agents have the above and:

1. Independence to choose to listen/ignore requests
2. Capable of flexible behaviour
3. Actively sense and work towards goals, rather than passively respond to requests

1.3 Intentional Systems

An **Intentional System** is an entity whose behaviour can be explained/predicted by attributing belief/desire/rationality to it. e.g She *believed* X, He *wanted* Y
This allows us to quickly summarise the behaviour of a system to understand the state, even if it's not actually true. This description is *legitimate* if you can describe a person the same way, and *useful* when it simplifies the understanding. For simple systems (such as a lightbulb), there are easier ways to describe the system.

First Order Intentional Has beliefs/desires .e.g A dog wants to walk

Second Order Intentional Has beliefs/desires about beliefs/desires.
e.g. Wants to know X, Knows it doesn't know Y

1.4 Why Agents?

1. Need for co-operation between devices

   Ubiquity Devices are in everything now. e.g IoT

   Interconnection Distributed/Concurrent/Networked systems are the norm

2. Devices can act independently

   Intelligence Devices can do more complex tasks

3. Devices need to accurately represent our intentions

   Delegation Devices are doing more tasks than before without human intervention

   Human-Orientation Devices are being viewed/think like humans. e.g intentional systems

Agent-based programming is capable of modelling independent action, interaction between devices and representing a set of goals. Multi-Agent Systems can co-operate and negotiate to achieve individual goals.
Agents can also be used in traditional software, to regulate the interactions between different parts of a complex system.
Finally, agents are excellent for *modelling* complex systems (such as human societies), to predict future behaviours.

1.5 Related Fields

Multi-agent systems draws from many other fields, including: economics, social sciences, philosophy, logic and game theory. Lots of ideas, lots of conflicting views.

1.5.1 Distributed Systems

Very similar, but threads aren't autonomous/don't have goals. Co-ordination requires negotiation, as opposed to allocations by some master controller.

1.5.2 Artificial Intelligence

Agents don't need to be strong AI, and instead focuses on the social/co-operative aspects

1.5.3 Economics/Game Theory

Great for modelling perfect theoretical scenarios, but doesn't give us specific solutions and often makes too many/infeasible assumptions

1.5.4 Social Science

No guarantee agents will act like humans societies - plenty of overlap, but still distinct fields

2 Practical Reasoning

2.1 Definition

Practical Reasoning is reasoning on what actions to do.
In humans, there are two steps:

1. Deliberation - Deciding the desired goal
2. Means-End Reasoning - Deciding how to achieve the goal

2.2 Intentions

Intentions are the result of deliberation: a state of affairs that the agent aims to achieve. Once obtained, agents will devote resources to trying to achieve it.
For an intention I chosen by an agent A:

- A will try to achieve I, trying alternative plans if it fails
- A won't aim for any intention that conflicts with I (This helps constrain the search space)
- A *believes* that I is possible under certain circumstances
- Agents won't choose intentions that are impossible or that will happen inevitably (without the agent acting)
- Agents don't always intend the side effects of their intentions (i.e if a is intended and a→b, the agent doesn't necessarily intend b)

Intentions can be thought of as a want/goal with actions being put towards it, rather than just a desire.

2.3 Planning

Means-End reasoning gives a plan of action for the agent to achieve its intention, taking into account the environment/state and possible actions.
The largest problem is that deliberation/reasoning aren't instantaneous: the optimal intention may have changed (change in environment/goals) before the deliberation has even finished (Calculative Rationality).

2.3.1 Deliberation

1. Generate options/desires using state, environment and current goals (to avoid conflict of intentions)
2. Filter these options to generate the new intention set

2.3.2 Commitment

Once an intention has been selected, the agent *commits* to achieving it.

Blind/Fanatical Commits until the intention is achieved

Single-Minded Till achieved or no longer possible to achieve

Open-Minded Commits while it believes the intention to be possible

Agents can commit to ends(goals) or means(the actions taken). If there's a change, the agent needs to check if its plan is valid and potentially re-plan.

2.3.3 Re-consideration Strategies

- Agents can re-check actions/intentions after each action it performs to check they're still possible (open-minded), but this is very costly.
- Checking every iteration might cause the agent to only deliberate and never act
- Checking too infrequently might lead the agent to try to achieve something that's no longer possible

Many systems have a higher-level check to determine if the intention/plan should be reconsidered - this check is less costly than reconsidering.
An environment that changes frequently favours constant re-checking, while static environments favour agents that achieve rather than check.

2.3.4 Belief Desire Intention (BDI) Agent

```
1. B = [initial_beliefs], I = initial_intentions
2. while true:
    (a) Obtain percepts P (info from environment)
    (b) Update beliefs B using P
    (c) D = options(B,I)
    (d) I = filter(B,D,I)
    (e) $\pi$ = plan(B,I)
    (f) while not (empty($\pi$) OR succeded(I) or impossible(I)):
        i. Execute action $\pi[0]$
        ii. Obtain percepts P
        iii. B = update(B,P)
        iv. if reconsider_check(B,I):
            1 D = options(B,I)
            2 I = filter(B,D,I)
        v. if not(sound($\pi$)):
            1 $\pi$ = plan(B,I)
```

2.3.5 Plan Libraries

The IRMA and PRS (Procedural Reasoning System) architectures use a *plan library*: a pre-made set of plans that an agent can use to achieve its intentions.

Each plan has:

1. Invocation Condition - why it would be selected
2. Context - the requirements for the plan
3. Maintenance Condition - things that must remain true for the duration of the plan
4. Body - contains the actions/goals to be achieved (The goals add sub-plans to a stack)

The PRS observes, generates goals by checking the plans' invocation conditions, and selects a plan as an intention. It creates a stack for the intention in its event queue: the system can have multiple intentions running at once.

2.4 Hybrid Agents

It's suggested that agents can't work with only deliberation/reactive capabilities, and that they need both. Such an agent can react to immediate events while maintaining overall goals.

3 Embedded Agents

An agent is a system capable of flexible autonomous action to meet objectives in a (multi-agent) environment. They typically work in a *sense* $\rightarrow$ *decide* $\rightarrow$ *act* loop.

3.1 Types of Environment

3.1.1 Accessibility

Accessible The agent can obtain complete, accurate and up to date information about the relevant parts of the state. e.g Chess

Inaccessible Where it can't get this information. e.g. the real world, the internet

3.1.2 Probability

Deterministic All actions have a guaranteed effect, with no uncertainty. e.g. Chess

Non-Deterministic The outcomes of an action have a probability of occurring, there's no guarantee. e.g the real world

3.1.3 Episodic

Episodic The performance of the agent in the current scenario doesn't affect future scenarios. e.g robotic vacuum cleaner

Non-Episodic The agent has to reason about its actions effects on the current state and future states. e.g Chess

3.1.4 Static/Dynamic

Static Only the agent's actions change the environment. e.g. Solitaire

Dynamic The environment changes beyond the agent's control (external processes, other agents).e.g. Chess, the real world

3.1.5 Complexity

Discrete A finite/fixed number of actions and percepts (can be solved with a perfect lookup table). e.g. Tic Tac Toe

Continuous Countably infinite number of actions and percepts

3.2 Functions and Sets

3.2.1 Sets

A set is a collection of objects, and can be written as S = $\{x|xhasthepropertyP\}$, where s is the collection of all objects that have the property P.

- x $\in$ S means x is part of the set S

- $\emptyset$ is the empty set (no elements)
- 2^S is the *power set* of S, which is the set of all subsets. e.g $2^{(1,2)} = (\emptyset, (1), (2), (1,2))$
- A x B is the cartesian product - the set $\{(a,b)|a \in A, b \in B\}$

3.2.2 Functions

A function maps each element of set A to a single element in set B ($f : A \longrightarrow B$). If x $\in$ A and y $\in$ B, this can be written as $f(x) = y$.

- $\exists$ means 'there exists' e.g. $\exists x$ such that $x < 5$
- $\forall$ means 'for all' e.g. $\forall x < 5, x - 5 < 0$

3.3 Agent and Environment Interaction

- For a discrete environment, it could be in a state e $\in$ E
- Any agent has a set of possible actions Ac, that affect the environment
- A run is a sequence of alternating states and actions. e.g $e^0 \rightarrow a^0 \rightarrow e^2 \rightarrow a^4 \rightarrow e^3 \rightarrow \ldots$
  - R is the set of all possible runs using E and Ac
  - R^{Ac} is the set of runs that ends with an action
  - R^E is the set of runs that ends with a state

3.3.1 Environment Behaviour

A *State Transformer* is a function that takes a run r $\in R^{Ac}$ and returns the set of states that could result from it. i.e $\tau : R^{Ac} \rightarrow 2^E$
The environment state obtained is dependent on the run (**history-dependent**) and non-deterministic. If τ returns $\emptyset$, then the run has ended - there are no possible successor states (These are shown from the last state, rather than the last action).
An environment *Env* can be represented as $< E, e_0, \tau >$ (set of states, initial state, state transformer).

3.3.2 Agent Behaviour

The agent determines its action based on the current and previous states: assuming it's deterministic, we can represent this as $Ag : R^E \rightarrow Ac$.
A system is a pair (Ag, Env). The set of terminating runs for this system is R(Ag,Env). We can say that two agents are behaviourally equivalent for Env iff R(Ag_1,Env) == R(Ag_2,Env). If this holds for all Env, then the two agents are generally behaviourally equivalent.

3.4 Types of Agents

3.4.1 Purely Reactive Agents

These don't take into account the past - they only use the present to make their decisions. i.e action: $E \rightarrow Ac$
Some agents map the environment to a *percept*, then select an action using this internal state. These agents have a set of percepts P, and use the following: see:$E \rightarrow P$, action:$P \rightarrow Ac$.

3.4.2 State-Based Agents

These agents maintain an internal state I between episodes, and update this based on percepts from the environment. This state records information about previous actions and environment states. i.e see:$E \rightarrow P$, next: $IxP \rightarrow I$, action:$I \rightarrow Ac$

3.5 Utility-Based Task Specification

Agents are built to fulfil tasks we specify, but we don't want to tell agents exactly how to complete them (as in a lookup table).

3.5.1 State Utility Functions

One method is to assign each state $\in$ E a utility, and tell the agent to maximise its utility. e.g u(e) = profit made by the agent in state e
This gives the agent a target, but doesn't give a guide for the method - it's difficult to give the agent a long term view with individual utilities.

3.5.2 Run Utility Functions

Assigning a utility to a run forces the agent to take a long term view. e.g u(r) = (profit in final state * number of states not in debt)

If the agent Ag is non-deterministic, then we can say the probability of run r occurring in environment env is $P(R|env, Ag)$. The expected utility of this run is $u(R) \cdot P(R|env, Ag)$.

An optimal agent is one that maximises the expected utility over all possible runs. i.e.

$$\arg\max_{Ag \in Agents} \sum_{r \in R(Ag, Env)} u(R) \cdot P(R|env, Ag)$$

3.5.3 Bounded Optimality

Since we don't have infinite processing power, we can't always have a perfectly optimal agent: instead, we choose the best agent that can run on our machine (AG_m).

3.5.4 Problems with Utility

1. Not all systems are easy to define in terms of numbers. e.g. financial systems are easy, preference systems are mostly arbitrary
2. Humans don't usually think in terms of utilities
3. It's hard to define specific tasks with utilities - we can give a final state to achieve, but not the tasks to complete on the way

3.6 Predicate Task Specification

Runs are assigned a utility of 0(fail) or 1(pass) - $\Psi : R \rightarrow 0, 1$. A task environment is defined as $< \Psi, Env >$ - the properties of the system + the criteria for fail/success.

$R^{\Psi}(Ag, Env)$ is the set of runs by Ag in Env that pass - an agent succeeds in a task environment if

$$R^{\Psi}(Ag, Env) = R(Ag, Env)$$

i.e all terminating runs succeed. For a non-deterministic agent, the probability of success is simply $\forall r \in R^{\Psi}(Ag, Env) \sum P(r|Ag, Env)$.

3.6.1 Achievement Tasks

The agent must achieve one of a given set of goal states G - it doesn't matter which. The agent succeeds if $\forall r \in R(Ag, Env)\ \exists e_i$ such that $e_i \in r$ and $e_i \in G$

3.6.2 Maintenance Tasks

The agent must never enter a set of 'bad' states B .i.e $\forall r \in R(Ag, Env)\ \neg\exists e_i$ such that $e_i \in r$ and $e_i \in B$

4 Linear Programming

4.1 Deductive Reasoning Agents

Agents that represent the state as logical formulae, and use deductive reasoning to determine the optimal actions. A *Symbolic AI* paradigm.

4.1.1 Problems

1. Representation/Reasoning - How does one represent the real world as a set of predicates that are useful?
2. Transduction - How do we make the translation between the real world and our representation fast enough?

The underlying problem is that symbol-based manipulation is complex and untractable.

4.1.2 Algorithm

The agent has a set of rules/predicate statements of the form $state- > action$, where the action is the best action for the given state.

At each time step, the agent chooses the rule whose precondition matches the current state. If there isn't a match, it picks the first statement that doesn't contradict the current state.

In a static environment, if the agent doesn't have a general rule (base case), it might reach a state where it doesn't know what to do - in this case it will stall. This doesn't mean that the goal is impossible, but that the agent doesn't know how to achieve it.

4.2 Planning Agents

Rather than solving the theorem at each state, the planner makes a plan (sequence of actions), that progresses from the initial state to the goal.

4.2.1 Problems

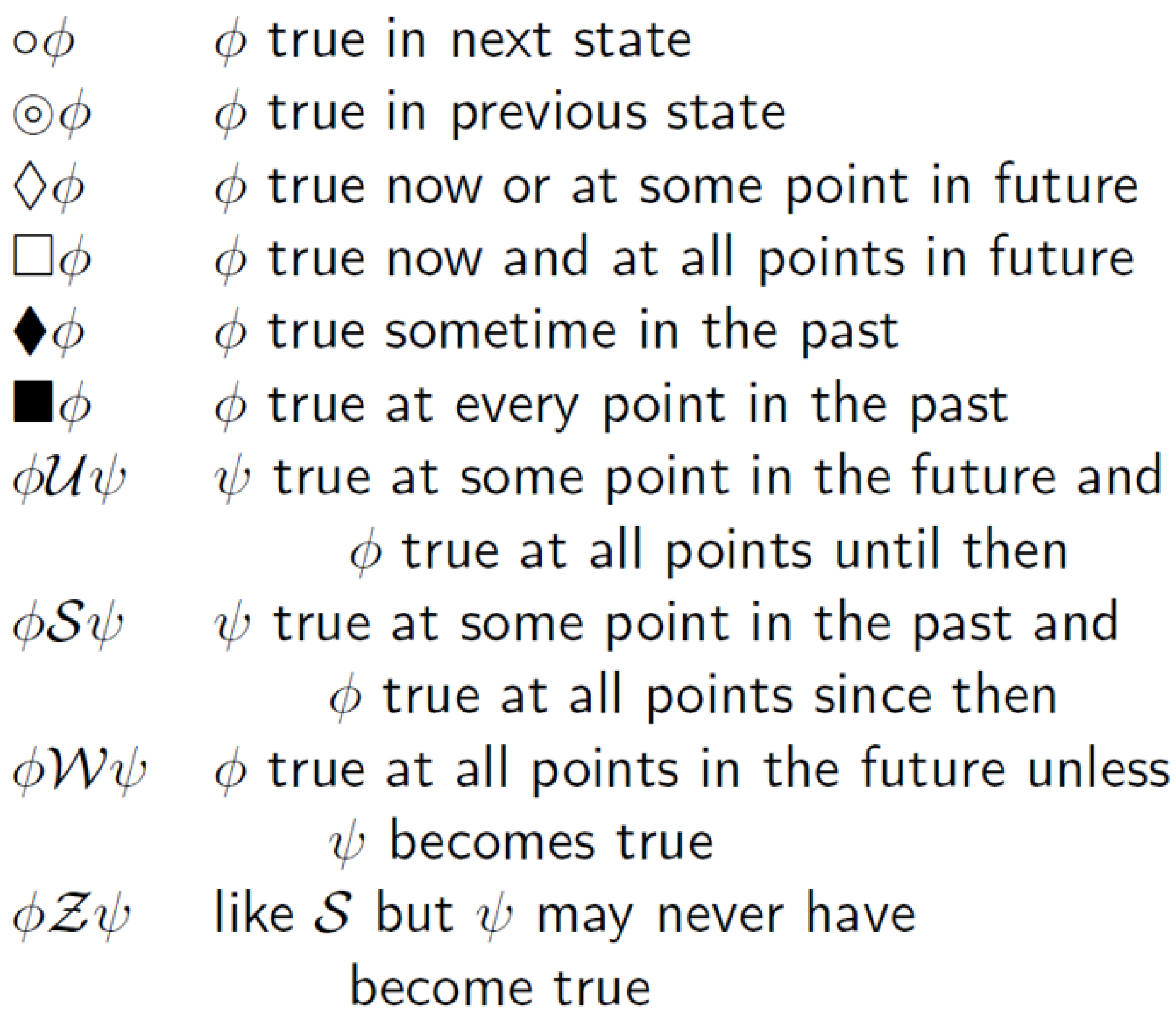
1. Representation/Reasoning - How does one represent the real world as a set of predicates that are useful?
2. Assumes a static environment
3. First-order logic can be NP-complete

4.3 Calculative Rationality

Any agent makes the optimal choice for the state in which it started to make the choice - in dynamic/multi-agent environments, this state might have changed before the decision is made. Such an agent has *calculative rationality.* Therefore, an optimal agent needs to be able to plan for the future while reacting to the current state. To represent this, we can use *Temporal Logic*

4.4 Temporal Logic

4.4.1 Symbols



$\circ\phi$	ϕ true in next state
$\circledcirc\phi$	ϕ true in previous state
$\lozenge\phi$	ϕ true now or at some point in future
$\square\phi$	ϕ true now and at all points in future
$\blacklozenge\phi$	ϕ true sometime in the past
$\blacksquare\phi$	ϕ true at every point in the past
$\phi\mathcal{U}\psi$	ψ true at some point in the future and ϕ true at all points until then
$\phi\mathcal{S}\psi$	ψ true at some point in the past and ϕ true at all points since then
$\phi\mathcal{W}\psi$	ϕ true at all points in the future unless ψ becomes true
$\phi\mathcal{Z}\psi$	like $\mathcal{S}$ but ψ may never have become true

4.4.2 Concurrent MetateM

- Agents are specified as a set of temporal logic rules, permitted incoming arguments, and output types
- Each agent uses its history/incoming information to match against rules - on a match, it performs an action

- Concurrent agents can communicate with asynchronous messages

At each state, the agent:

1. Reads incoming messages to set state booleans
2. Checks which rules fire in the current state
3. Executes rules and fulfils any commitments from actions in previous states (commitments take priority)

This way, the agent separates immediate reactions with long term actions to achieve a goal. The reasoning is still difficult, as is translating the world to symbols.

4.5 Brookes: Subsumption Architecture

4.5.1 Concept

This approach is heavily inspired by *Behaviourism*, that a system can be described by how it reacts to its environment.
It's primary principle is that real intelligence **must be situated in the real world**, and emerges from the interactions with the world. It can't be achieved by an abstract theorem solver.
Brookes put forward three theses:

1. Intelligence can be generated without explicit representations
2. Intelligence can be generated without abstract reasoning
3. Intelligence is an emergent property of a complex system, rather than inherent/quantifiable

4.5.2 Composition

The architecture is composed of sets of modules/rules/behaviours, each of which takes the form:

IF situation THEN action

These behaviours are organised into hierarchies to resolve ties.

4.5.3 Performance

Though simple, it achieves near-optimal performance on multiple domains. However, it can't think ahead to future episodes, and doesn't have a history to reference.

4.6 Hybrid Architectures

Some researchers have suggested an agent with two subsystems: one deliberative, one reactive. Usually the reactive takes priority.
To manage priorities, the system is split into layers:

Horizontal Layering Each layer is connected to the input and output, and each gives a separate suggestion - as if each were a separate agent

Vertical Layering Layers are connected to each other (one connected to input, one to output), and take the results from the previous layer as input to make decision

Horizontal layering gives a wide variety but introduces a bottleneck as every layer must wait for the slowest. Vertical layering isn't tolerant to a single layer failing, but makes decisions across multiple layers than a single choice at the end.

5 Agent Based Modelling

Simulations are an abstracted model of the real world. They can be used to test theories/effects of potential future changes, without actually implementing them.

They're widely used where data analytics fail.i.e in very complex systems, where behaviour is emergent, and where data isn't available. Simulations *generate* data, rather than just analyse it.

5.1 Models

Any simulation uses a model, which is an abstraction of the real world made to be as simple as possible. Models are based on assumptions and theories, often to test the theories.

If a simulation/model explains a real-world phenomenon, then it is a *candidate solution* - further proof is needed to call it a *causal explanation.* The further proof is an ill-posed problem.

One test is to vary the initial parameters - fragile models are very sensitive, so will vary dramatically with a tiny change.

5.2 Bottom-Up Modelling

Agent-based modelling is a *bottom up* approach - the individual parts of the system are defined, and the behaviour of the system as a whole is emergent from the interactions of the parts.

These parts/agents can be heterogeneous - a single agent can then be individually changed, and the effects of that change are then transmitted to the system organically.

5.3 Environment

Agents are embedded into an environment, which define their behaviour.

Spatial Models Agents have co-ordinates on a grid, and can move about. Interactions are based on proximity.

Network Models Links between agents are via edges, like a graph. Interactions are based on link proximity.

Environments can also contain non-autonomous entities, like the weather/ external factors.

5.4 Scale and Feedback

Since agents are non-deterministic, and behaviour emerges from interactions, every run of the system is unique. These can be from lots of simple agents, or a few behaviourally complex agents. e.g. Testing network effects at scale on simple agents vs. testing how a few agents react to a complex environment
The rich data obtained comes from feedback loops: the final state isn't a simple function of the initial state. These loops don't depend on the size/scale of the system, just its structure.

5.5 Cellular Automata

A simpler system that influenced most ABM models.

- The environment is a grid of cells, and each cell has a state
- At each time step, the cell updates based on its neighbours' states
- Each cell is a non-moving, homogeneous agent with no memory in a fixed dynamic environment (i.e the neighbouring cells)

A famous example is Conway's Game of Life.

5.6 Shelling's Segregation Model

A precursor to ABM's, that focused on individuals in the system - their individual decisions had effects they neither intend nor were aware of. e.g people wanting to live amongst their community splits cities on racial lines

5.7 Intervention

An *Intervention* is a change in the model that affects the agents behaviour. It can be explored by comparing the 'before' and 'after' applying the changes.

5.7.1 Influence Maximisation

By identifying which agents have most influence on the system (e.g the most connections in a network model), interventions can be targeted at them - this maximises the amount of change with minimal intervention. In the real world, these could be applying changes to social media influencers or targeted at certain ethnic/political groups.

5.7.2 Influence/Spreading Functions

Influence can be modelled in different ways, such as in peer pressure (an agent is influenced if a certain percentage of its connections are). The spread of disease can be modelled in such a fashion.

5.8 Creation

Usually the agents are relatively simple (e.g. BDI, state machines, subsumption-based), and the behaviour emerges from their interactions. The agents only need to capture the relevant behaviour, so they don't need to be complex.

Data is required for realistic modelling:

- Calibration - to set the initial agent/environment parameters
- Verification - use the data to set limits on the agent behaviour, etc.
- Validation - check the simulation is correct against real data (Face validation is checking against estimates)

5.8.1 Pyschological Agents

For social setting models, we simulate a particular aspect(s) of the psychology of the individuals. e.g wanting to do X

Adding psychological models such as stress and trust can increase the realism of the model - some models even have their agent learn and adapt as the model iterates (i.e. reinforcement learning)

6 Coordination

Agents are autonomous: so in a multi-agent environment, one agent can't *force* another agent to taken an action. Instead, they have to communicate to try to influence other agents.

6.1 Speech Acts

The *Speech Act Theory* (John Austin, 1962) states that communication is an action that aims to achieve an intention. The speech act can be split into three parts:

1. Locutionary - making the communication. e.g. saying 'Open the Door'
2. Illocutionary - conveying intentions (highly context dependent)
3. Perlocution - the effect of the communication. e.g. the door has been opened

Agents make a locutionary act, hope that the illocutionary act has been transmitted (i.e the speaker understood the meaning), and expect the listener to effect the perlocution.

6.1.1 Classes of Speech Acts

According to John Searle (1969), there are five classes:

Representative The speaker says a statement is True/False. e.g. it is raining, it is not 5pm

Directive The agent asks the hearer to do something

Commissive The agent commits to doing something. e.g. I will submit it by 6pm

Expressive The speaker expresses a mental state. e.g. 'Thank you', 'I am sad'

Declaration The speaker says something, which causes an effect. i.e the speech is an action. e.g. 'We are now at war with Mars'

6.1.2 Parts of a Speech Act

Speech acts have two components:

Performative Verb The illocutionary part. e.g. request, inform, commit

Propositional Content The raw content

e.g.

	Verb: Request	Verb: Inform	Verb: Inquire
Content: The door is closed	Please close the door	The door is Closed	Is the door closed?

6.2 Agent Communication Languages

Provide a standardised way for heterogeneous agents to communicate.
e.g. the KQML is an agent communication language that consists of a Knowledge Interchange Format (to denote info/relationships) and a Knowledge Query Manipulation Language (to define performatives). Agents apply the latter to teh former (as in speech acts) to communicate.

6.2.1 Communication as Actions

Communication is treated as a separate action in a plan. e.g Request(X,Y,Z):

Preconditions X believes Y can do Z, X wants Y to do Z

Effect X believes Y wants to do Z

6.2.2 Communication as Mental States

Communications are based around the beliefs of an agent. i.e. if an agent believes a fact to be true
The effect isn't caused as soon as the communication occurs, leading to a potential difference between what the agent believes and what is true.

6.2.3 Communication as Social Commitments

Communications are transmitted as commitments from one agent to another. e.g. If I tells J that $\neg A$, then it can't tell K that A unless it also tells J
These commitments can be publicly verified.

6.2.4 Effectiveness

- Most of these approaches assume the agents share the same semantics/thought processes as well as the semantics - not always feasible
- Limited to the set of 5 performatives - we can expand this list, but where do we stop?
- As a positive, these performatives act as shortcuts to more complicated concepts

6.3 Co-operation

Agents are autonomous. i.e they make decisions at runtime. Therefore, any co-operation between them must be dynamic, rather than hard-wired

6.3.1 Measuring Co-operation

Bond and Gasser [1992] proposed the following criteria to evaluate cooperation:

Coherence How well the system achieves the goal. e.g. outcome measure, time taken

Co-ordination How well the agents work together. e.g. not getting in each other's way, sychronizing routines so they're not waiting for each other

6.3.2 Co-operative Distributed Problem Solving

A type of domain where agents are autonomous and can solve different problems independently, but must work together to complete the problem as a whole.

- No agent has enough knowledge/skills by itself
- The problem is modular (can be split into reasonable blocks)
- More time spent on computation than communication (if the other way, better to make one 'super' agent)

Solving a CDPS problem has 4 main steps:

1. Decompose the problem
   - Which agent decomposes, and how granular is the decomposition?
2. Allocate the subproblems
   - Who decides the allocations? What if an agent refuses its allocation?
3. Solve the subproblems
   - Some subproblems might need scheduling/further co-operation
4. Combine the subproblem solutions as required

6.3.3 Contract Net Protocol

A protocol to allocate subproblems, where the agents bid for the problems they want. The process is:

1. An agent recognises a problem needs co-operation/delegation of parts
2. Agent broadcasts the task: - description, constraints (e.g. deadline) and meta-information (e.g. bid submission deadline)
3. Other agents consider, and maybe return a bid
4. First agent decides who gets the task, and informs all the bidders
5. The winning agent (contractor) performs the given task

In an ACL, this would be: agent calls for proposal, other agents propose, agent accepts/refuses to all other agents, contract informs of failure/success of task.

The collection of agents is the contract net - the only hierarchy is the delegation of tasks, otherwise all agents are equal. e.g A delegates X to B, B splits X into X_1 and X_2, delegates X_1 to A and X_2 to C

The protocol works well for distributed data, heterogeneous agents (with different skills), and tasks large enough to split and delegate.

6.3.4 Co-ordination, Multi-agent Planning and Joint Intentions

Co-ordination is managing dependencies between agents activities.

Positive Agents ask for help with actions, don't plan the same actions, contribute to others' actions

Negative Agents try to use resources at the same time, plan actions that require other agents even if they're busy

One way to manage co-ordination is through various types of *Multi-Agent Planning*:

Centralised Planning A central agent makes plans for all agents

Merging Each agent makes a plan, and a central agent merges them to produce a global plan

Partial Global Planning Each agent receives the global plan, makes its own plan accordingly and adjusts the global to include the new information, then passes the global plan along - no centralised system

Multi-agent planning requires agents to share private info and reduces autonomy. Combining plans is also non-trivial.

Joint Intentions is a teamwork-based approach. Agents have shared goals, and a responsibility towards other agents. Every agent works towards a goal G with a motivation M - if G is no longer possible or M doesn't hold, the agent tries to inform the other agents. If G is achieved, it instead informs the other agents that the goal is over.

6.4 Expectations and Norms

To co-ordinate, agents need to have expectations on how other agents will act. These can be obtained via:

1. Communications and joint intentions
2. Knowledge of the system: what is normal/required, past experiences (trust) and reputation of other agents

6.4.1 Normative Agent Systems

A *norm* is an expected pattern of behaviour - it isn't guaranteed though. They can be *Conventions*(e.g. take a gift when visiting people) or *Prescripted* (e.g. follow the laws).

A prescriptive norm can be formally represented with: who it affects, is it an obligation/permission/prohibition, when it applies, what it is, and what the punishment for breaking it. e.g.

<All Drivers, Obligation, At a red light, Stop, 3 points on licence>

6.4.2 Reasoning with Norms

Norms affect agent behaviour: violating one has a punishment and causes a loss of reputation, so agents will add achievement/maintenance tasks to not break norms. For MAS, agents can expect other agents to generally follow norms - but this isn't guaranteed.

Agents might break norms if: they didn't know/understand, the punishment is too low, another norm/goal is more important, or if the norms aren't being enforced.

6.4.3 Conventions as Emergent Behaviour

Conventions can occur from agents copying other agents - these might not be globally followed, with different subsets of agents following different norms. e.g driving on the right/left

By applying influence maximisation, desired norms can be spread across agents by targeting influencers.

6.5 Trust/Reputation

Agents running a bid will pick the agent it believes will do the best job (maximising expected utility). This belief can either come from previous dealings with the agent in question (**Trust**), or by the trust from other agents (**Reputation**).

Reputation requires trust in the other agents' ratings: the ratings could be on a different scale, or two agents could collude to boost their reputation with other agents.e.g fake amazon reviews.
If the agent only uses trust, it will continuously pick the same agent as it never generates trust scores for unknown agents. A degree of exploration can be built into the agent, so it tries new agents and generates new trusts.

6.5.1 FIRE Model

One example of trust/reputation is the FIRE model - it uses a weighted sum of:

- Direct experience
- Other trusted agent references
- Norm expectations
- References provided by the agent

The weights are based on the source and recency of the information.

7 Negotiation

Self-interested agents need to agree on how to perform/allocate tasks/resources - they do this by negotiating.
An *Agreement Protocol/Mechanism* defines the structure of the negotiation/argument:

1. What moves each agent can make
2. What are the permitted responses/sequence of moves. e.g. can an agent make an offer in response to an offer?
3. How the outcome is determined

Each agents has a *Strategy* to decide what move it makes in a given state.

7.1 Game Theory

A way to analyse negotiations/scenarios where each agent tries to maximise its own utility.

Strategy The action the agent chooses to take

Outcome The result of each agent/player executing their action

Dominant Strategy No matter what the other player(s) do, this is the best strategy for an agent

Nash Equilibrium An outcome where no agent benefits from changing, assuming no other agent changes

Pareto Optimal An outcome where changing to any other outcome to benefit one player would harm another player(s)

Max. Social Welfare The outcome with the max. utility sum across all agents

e.g.

Agents		Agent 2	
	Actions	A	B
Agent	A	2 2	4 0
1	B	0 4	0 0

In the above example of a game with two agents and two actions each: BB is the Nash Equilibrium; AA, AB and BA are Pareto Optimal; AA, AB and BA maximise Social Welfare.

7.2 Properties of Agreement Mechanisms

An agreement mechanism can be any of the following:

1. Individual Rational - No agent is worse off for participating
2. Symmetric - Same rules for everyone
3. Fair - Every agent has an equal chance of affecting the outcome
4. Pareto Optimal - A PO outcome is guaranteed
5. Maximises social utility - An outcome that maximises social utility is guaranteed
6. Stable - A Nash Equilibrium Exists
7. Simple - easy to find the best strategy
8. Guaranteed to terminate - no way the participants interact indefinitely

7.3 Negotiation Definitions

Negotiations usually take place over a series of rounds, and each agent makes one proposal per round. The *Negotiation Set* is the set of possible proposals that an agent can make. Negotiations can be *single-issue* (e.g. the price of X) or *multi-issue* (e.g a contract with multiple clauses). In the single it's obvious where concessions are made, but in the multi it's not obvious how the various issues relate. Multi-issue negotations also can't be simply iterated over, as there are too many possibilities.
Negotiations can be one-to-one, many-to-one (e.g an auction) or many-to-many.

7.4 Alternating Offers Protocol

A one-to-one protocol to split a given amount - in each round, one agent proposes a deal and the other either accepts (and the negotiation terminates) or rejects and continues to a new round with the roles reversed.
The game is not guaranteed to terminate: if the agents can't reach a deal, this is known as the *Conflict Deal* - which is the worst possible outcome for both agents.

7.4.1 Ultimatum Game

Like the AoP, but only a single round. Guaranteed to terminate, but unfair to agent 2 as only agent 1 can affect the outcome (both want to avoid a conflict deal)

7.4.2 Strategies

Given that all agents want to avoid a conflict deal, the agent going last (with a fixed number of rounds) can offer any deal, and the other agent must accept. For an indefinite number of rounds, the first agent will get any deal they want - as they always go first, they can propose the same deal every time and reject whatever the second agent proposes.

Agents are generally impatient - the sooner they get something, the more value it has. This can be represented as the value of X to agent i at round K being $\delta_i^k \cdot X$ - the higher δ_i, the more patient agent i is.
Knowing the patience of the other agent allows you to make an offer they must accept - the longer the game goes on the less the item is worth to them. For an indefinite number of rounds, the deal agent 1 should make (Assuming the split is (agent 1, agent 2)) is:

$$\left(\frac{1-\delta_1}{1-\delta_1\delta_2}, \frac{\delta_2(1-\delta_1)}{1-\delta_1\delta_2}\right)$$

7.5 Task Oriented Domains

Agents have to achieve a set of tasks: they can negotiate to swap tasks to be more efficient. The domain is represented as: $< T, Ag, c >$ = <set of tasks, set of agents, cost of each subset of tasks>.

- Encounter - Initial allocation of tasks (a set $< T_1, T_2, ..., T_i >$ where T_i is the allocation for agent i)
- Deal - The final allocation of tasks(a set $< D_1, D_2, ..., D_i >$ where D_i is the allocation for agent i)
- Cost of deal for agent i - $c(D_i)$
- Utility of deal for agent i - $c(T_i) - c(D_i)$
- Conflict Deal - the initial deal
- Individual Rational Deal - a deal where no agent prefers the conflict deal

The negotiation set is all deals that are individual rational + pareto optimal (i.e utility >= 0 for all agents)

7.5.1 Deception

Agents can lie about the number of tasks they have to get a better deal.

1. Hidden Task - An agent can hide a task from its encounter. This lowers the encounter cost, so reduces the individual rational strategies to give it a better deal
2. Phantom Task - An agent can pretend to have been allocated a new task. This task can make another task (which the agent wants) look worse, so no other agent will bid for it.

7.6 Monotonic Concession Protocol

One-to-one protocol: agents simultaneously propose deals in each round. If neither agent agrees to its opponent's deal, the round continues - in the next round, each agent's proposed deal *cannot be less preferable to it's opponent.* i.e it must propose a deal that is equal/better for the other agent. If neither agent concedes on their deal, the conflict deal is chosen.

7.6.1 Zeuthen Strategy

1. The first deal an agent proposes is its most preferred
2. The agent least willing to risk conflict should conceded
3. An agent should only concede enough to change the balance of risk

This strategy maintains a Nash Equilibrium, so agents can announce this strategy without losing anything.

The closer an agent's deal is to the conflict deal, the more willing to risk (less to lose).

$$\begin{aligned}\text{A's willingness to risk} &= \frac{utility(Deal_a) - utility(Deal_b)}{utility(deal_a)} \\ &= \frac{\text{Utility lost by conceding}}{\text{Utility lost by conflict}}\end{aligned}$$

If $utility(deal_a) = 0$ (i.e. the conflict deal), then the risk = 1.

8 Voting

Social Choice Theory is the study of how group decisions are made. Agents are self-interested and want to influence the outcome in their favour, so they use voting protocols to account for a range of preferences.

8.1 Domain

There are a set of N *voters*, each of which have preferences to a set of *outcomes/candidates.* We assume that each voter has a complete set of preferences($A > BorB > A\forall(A, B)$, and that these preferences are transitive ($A > B, B > C \rightarrow A > C$) - so we can represent them as a ranked list.

8.2 Preference Aggregation

Given a collection of preference orders, combine them into an ordering that reflects the orders as closely as possible. There are two main types of algorithm:

Social Welfare Function combine the data into a new complete ordering

Social Choice Function combine the data to select a single most preferred outcome

8.3 Plurality Voting

A simple social choice function: each outcome gets one point if it's **first** in a voter's preference ranking, and the outcome with the most points is selected.
This ignores the rest of the preference/lower ranks. e.g. voters are split between A and B for first, but everyone has C in second. It also allows for strategic voting: if an agent knows the rankings of other agents, it can submit a false ranking to change the outcome.

8.4 Condorcet Winner

A social choice function: every pair of outcomes has a contest, and the outcome that beats *every* other outcome is chosen. The contest is simply the number of voters that prefer one outcome to the other. i.e. for (a,b), a gets +1 point for each voter with $a > b$ and -1 for each voter otherwise

8.4.1 Condorcet Principle and Paradox

Any system that follows the *Condorcet Principle* will always choose the condorcet winner. However, there might not be a candidate that wins in every case - this is the *Condorcet Paradox.*

8.5 Linear Sequential Majority

A social choice function: outcomes take place in pairwise rounds, and the winner moves onto the next round. e.g. (4,3) $\rightarrow$ winner faces 1 $\rightarrow$ winner faces 2
The order of the contests is called the *agenda* - it determines which outcomes face each

other, and is open to manipulation as some outcomes might not be the strongest, but if they're at the end of the agenda can still win. The contests can be represented as a *majority graph*, with an edge $a \rightarrow b$ implying a beats b in a simple majority.

8.6 Instant Run-Off

A social choice function: in each round, the candidate with the least number of first choice votes is removed from all voters' lists, and this continues until only one outcome remains. If a voter's first choice is removed, then its second choice becomes its first and so on.
This method doesn't account for deeper preferences in candidates that are removed early. e.g. nobody had B first, but everyone had them second.

8.7 Copeland Rule

A social choice function: similar to condorcet, except the outcome with *the most wins* is chosen. i.e. there doesn't need to be an outcome that beats every other outcome. A win gives 1 point, a loss -1 points, and a draw 0.

8.8 Borda Count

A social choice function: similar to plurality but the full ranking is used, with each position being weighted. The outcome with the most points wins. The weighting for m candidates is: (m-1) points for each first place vote, (m-2) for each second place . . . 0 for last place votes.

8.8.1 Spoiler Effect

Candidates who don't win the election under the borda count can still shift the results of the election.e.g.

3 votes for $a > b$, 2 votes for $b > a$ - a wins
3 votes for $a > b > c$, 2 votes for $b > c > a$ - b wins

This happens because B now receives more votes as it's not in last place, while A still is.

8.8.2 Positional Scoring

Rather than strictly (m-1), (m-2), the scores $(S_1, S_2, S_3, \ldots, S_m)$ can be any values as long as the following two rules are maintained:

1. $S_1 \geq S_2 \geq S_3 \geq \cdots \geq 0$. (Non-negative decreasing scores)
2. $S_1 \geq S_m$. (The scores must decrease at some point)

e.g. $S_m = 0, S_{m-1} = 1, S_{m-2} = 3, S_{m-3} = 6, S_1 = m(m+1)/2$ Positional scoring as a whole violates the condorcet principle (i.e. it's not always guaranteed to provide the condorcet winner)

8.9 Voting Rule Properties

Anonymous All votes are counted equally - no voter is more influential than any other

Resolute A unique winner (no ties)

Surjective Every candidate has a chance to win

Dictatorship there is a voter whose first choice is always the winner

Strategy-Proof there isn't an ordering for a voter that gives it a better outcome than its true preferences (no reason for a voter to lie)

Weakly Pareto It won't select an outcome A it there is an outcome B such that $B > A$ for all voters

8.9.1 Gibbard-Satterthwaite Theorem

Any resolute, surjective and strategy-proof rule for 3 or more candidates will be a dictatorship.
Selecting a strategy can be very difficult, so this doesn't mean that all rules are inherently useless.

8.9.2 Independence of Irrelevant Alternatives

The outcome of a preference between A and B should only depend on the preferences between A and B. e.g. if $A > B$, then adding C shouldn't change that. As seen previously, the Borda Count does not hold IIA

8.9.3 Arrow's Impossibility Theorem

Any rule that is surjective and is independent of irrelevant alternatives will be a dictatorship.

8.9.4 Universal Domain Assumption

Both of the above theorems assume that every voter has a complete preference ordering - if we drop this assumption, then the theorems no longer hold.

8.10 Median Voter Rule

8.10.1 Single-Peaked Preference

There is an ordering over the outcomes, and outcomes closer to your first choice are more preferred than those further away. e.g. let the outcomes be 21,17,11 and 4 - if a voter chooses 17, they'll prefer 21 to 11.

8.10.2 Voting Rule

A social choice function: each voter chooses a single candidate, the votes are ordered, then the median candidate is chosen. e.g.

21 gets 10 votes, 17 gets 20 votes, 11 gets 40 votes, 4 gets 30 votes.
There are 100 votes in total, so the median is 50 - this is 11, so 11 is the winner

If there are an odd number of voters, this provides a condorcet winner. The rule is strategy proof (a voter can't shift the median towards their true outcome by voting against it), and is weakly pareto.

9 Auctions

Auctions are a mechanism to allocate **limited** resources that are desired by more than one agent. The resources are allocated based on how much each agent values the resource.

9.1 Structure of an Auction

9.1.1 Types of Agents

Auctioneer an agent that allocates the resources - wants to maximise the selling price

Bidder an agent that wants to minimise the price they pay for the resources - they do this by using a *strategy* to place bids

9.1.2 Valuation of Goods

Public/Common The good has the same value to all agents - but not all agents have perfect information on this value

Private Each agent has their own value on the good. e.g. sentiment

Correlated Private + based on other people's valuations. e.g. buy a good to sell it later, so you need to know how much others want it

9.1.3 Visibility of Bids

Open Cry Bids are common knowledge to all bidders

Sealed Bid Bids are private

9.1.4 Bidding Mechanism

Ascending/Descending Start at price X, subsequent bids increase/decrease the price

One-Shot All agents submit a single bid

9.1.5 Winner Price Determination

First Price Agent with the highest bid pays the highest bid

Second Price Agent with the highest bid pays the second highest bid value

9.2 English Auctions

First price, open cry, ascending. The bidding stops after a set time deadline or a fixed time period in which no bids have been made.

9.2.1 Bidder Strategy

Agents will never bid more than their expected value of the good. Instead, they bid ΔV (the smallest permitted amount) above the current highest bid until they reach their expected value.

9.2.2 Expected Price

The auctioneer will receive the second highest bid + ΔV, as opposed to the winner's true valuation. It therefore loses out that much. The less competition, the higher the loss of revenue for the auctioneer.
Auctioneers (or agents they collude with) can place *shill bids*, which are false bids intended to drive up the value of the second highest bids. Auctioneers will also have a *reserve value*, which is the minimum value they will sell for.

9.3 Dutch Auction

First price, open cry, descending. The auctioneer controls the speed of the descent, and the first bidder to accept the current price wins.

9.4 Bidder Strategy

Agent will never bid more than the expected value. Below this value, the agent needs to balance two risks: decreasing the price vs. losing the item to another agent. This is made harder as the agent doesn't know the valuations of the other agents.

9.4.1 Risk Attitudes

Risk Averse An agent that's less likely to take risks

Risk Seeking An agent that's more likely to take risks

Risk Neutral An agent that bases their choice solely on probabilities. e.g.

9.4.2 Expected Price

If the bidders are risk averse, they will accept as soon as it hits their true value, so the auctioneer will get the maximum valuation across all agents. If they're risk-seeking, then an English Auction provides better results.

9.5 First Price Sealed Bid

A one-shot auction mechanism, where every agent submits one private bid. It's similar to a dutch auction - agents won't pay more than their true valuation, and below that depends on their risk attitude.

All of the mechanisms so far have bidders taking into account the other agent's valuations, rather than giving their true valuations.

9.6 Vickrey Auction

Second price, sealed bid, one shot. As agents only pay the second price, it might be possible to bid more than their true valuation

9.6.1 Bidder Strategy

- If they bid higher than their true value, the second bid could also be higher - this would end in a loss
- If they bid lower than their true value, they have less chance of winning and they still get the second highest valuation

Therefore, it makes sense for the bidders to bid their true values

9.6.2 Expected Price

Auctioneers will get the second highest true valuation - this is slightly less than an english auction.

An *anti-social bid* is a bid placed to intentionally raise the second highest bid, so the winner pays more. This is risky for a bidder this value will be higher than its true valuation, so it might accidentally win and get a loss. However, an auctioneer can place a shill anti-social bid at no risk once it knows the highest bid.

9.7 Winner's Curse

For any auction, the winner always faces the concern that they paid too much/over-valued the good, as no other agent was willing to pay that much. This is especially the case if the good has a public value. For a Vickrey auction this is reduced, as the second highest agent would have also needed to imperfectly value the good.

9.8 Bidder Collusion

Multiple/all of the bidders could collude to buy the good at a low price, then sell it elsewhere for a higher price and split the profits between them.
This is dangerous if the agents cant trust each other, as one could betray and buy it for a slightly higher price then keep it. Hiding agent bids and keeping who wins a secret further dicourages collusion.

9.9 Combinatorial Auctions

Auctions in which multiple goods are auctioned at the same time. For the set of goods Z, each agent i∈Ag has a preference/valuation function over subsets of goods ($v_i : 2^Z \rightarrow R$(real numbers)). The outcome of the auction is the set of allocations of each item ($< z_1^*, z_2^*, \cdots >$) - an item can't be allocated to more than one agent.

9.9.1 Properties of Valuation functions

Free disposal the agent isn't worse off getting more items ($z_1 \subseteq z_2 \rightarrow v_i(z_1) \leq v_i(z_2)$)

Normalised $v_i(\emptyset) = 0$

Substitutable Some goods can replace others, so we don't want to repeat items. e.g. buying a console+game and buying just the game
$z_1 \cap z_2 \neq \emptyset, \quad v_i(z_1) + v_i(z_2) > v_i(z_1 \cup z_2)$

Complementarity Some goods increase in value when together. e.g. 2 halves of a set of collectable items
$z_1 \cap z_2 \neq \emptyset, \quad v_i(z_1) + v_i(z_2) < v_i(z_1 \cup z_2)$

The *exposure problem* is when an agent bids high for all of the items in a complementary set, but fails to obtain one or more - they've now overpaid for the other items.

9.9.2 Maximising Social Welfare

The auctioneer wants to maximise social welfare (give each bidder its highest valuation), but doesn't have access to the valuation functions of each agent. Even if each agent declares its entire value function, this isn't sufficient as:

1. Agents can lie on their value functions
2. The table would be of size numOfAgents x numOfBundles. For 1000 items, there are more possible combinations than atoms in the universe
3. Np-hard problem, even assuming a small number of items/agents

9.9.3 Bidding Language

Agents can submit *atomic bids* - a pair (z, p) of a bundle and the price the agent is willing to pay for it. This defines a valuation function v_i where $v_i(z') = p$ if $z' \supseteq z$, and $v_i(z') = 0$ otherwise.
These can be connected with an XOR to form *composite bids* - agents will pay for one of the atomic bids to be satisfied. If multiple bids are, then the agent pays the highest price. These composite bids can be used to represent the entire set of possible bundles, and only require an exponential amount of atomic bids if the agent has a different valuation for every combination of goods. e.g. if the agent only wants a good g, then it can represent every bundle containing that good with a single atomic bid

9.9.4 Winner Determination Problem

This is a combinatorial optimisation problem, which is NP-Hard. To make the problem easier we can

1. Restrict the number/types of bids - constraints
2. Use different techniques for specific cases - heuristics

3. Accept approximate solutions

One set of constraints is to limit the size of bids, and say that the bids must be on a continuous set of good. e.g. continuous frequency spectrum. If there is no natural ordering, we can impose an artificial one.
An example of a heuristic is a greedy algorithm on bid prices - at each iteration, allocate the bundle with the highest bid price among all agents

9.9.5 Vickery-Clarke-Groves Mechanism

1. Each agent declares their valuation function
2. The auctioneer allocates to maximise social welfare $\rightarrow < z_1^*, z_2^*, \cdots >$
3. For each agent i, the price is p_i where:
    (a) Determine the max. social welfare allocation if i wasn't present $\rightarrow < z_1', z_2', \cdots >$
    (b)

$$\begin{aligned} p_i &= \sum_{k \in (Ag \neq i)} v_k(z_k') - \sum_{k \in (Ag \neq i)} v_k(z_k^*) \\ &= \text{social welfare lost since the agent is present} \end{aligned}$$

10 Argumentation

An *argument* is a reason to believe/do/want something.

- Can be exchanged between agents to resolve differences and structure interactions
- Agents can use them to structure their internal reasoning
- There can be multiple arguments for/against the same thing

An argument is structured as a set of *premises* to support a *claim.* i.e X because Y,F and M

10.1 Definition

Argumentation is the process of constructing/exchanging arguments to interact with other agents.

1. Abstract Argumentation - given a set of arguments that contradict each other, find a consistent set of knowledge
2. Argument-based Dialogue - give a standard structure for interaction in negotiation, persuasion, etc.

10.1.1 Terms

Database a set of formulae/knowledge, might not be internally consistent

Argument (premises,claim) - the premises are present in the database, and the claim follows from the premises (using classical logic). The premises are the minimal required set to prove the claim.

Attack when two arguments conflict with each other

Rebut (p_1, a_1) and (p_2, a_2), where $a_1 = \neg a_2$

Undercut (p_1, a_1) and (p_2, a_2), where $a_1 = \neg p$ for some $p \in p_2$

10.2 Abstract Argumentation

The internal structure of the arguments are ignored, and the focus is on which arguments attack each other - this is represented as an attack graph (A,R) (= (arguments, attacks)).

An argument is *In* if all attacking arguments are *Out*, and *Out* if any argument that attacks it is *In.* Arguments can also be unlabelled. e.g. if $A \to B$ and $B \to A$, without making an assumption/external arguments then neither argument can be labelled.

An argument that isn't attacked by anything is *In* by default.

10.2.1 Semantics

1. Grounded Semantics - strictly follow the rules to minimise node labelling
2. Preferred Semantics - Use assumptions to maximise node labelling
   (a) Preferred Credulous - an argument is marked In under any of the preferred labellings
   (b) Preferred Sceptical - an argument is marked In under all of the preferred labellings

Preferred semantics are preferred as they provide us with more information to use.

10.2.2 Subjective Defeat Relations

For $A \to B$ and $B \to A$, A only *defeats* B if A is subjectively preferred to B - introduces preferences for certain arguments.
These preferences can be explicitly shown in the graph, and can attack attack relations between arguments (e.g. in the above example, a preference for A would attack $B \to A$). Preferences can also attack each other or, or attacks on preferences, or ...

10.2.3 Defeasible Knowledge

Knowledge can be revised, attacked, or rendered void. If counter-evidence is provided, something assumed to be true can be changed to false. The above only applies to facts - opinions can't be objectively defeated.

10.3 Non-Monotonic Reasoning

Agents can change their conclusions as they learn about the world - they can shrink or grow their set of conclusions (can only grow in monotonic).

Classical logic doesn't work well to represent this, but argumentation naturally allows for new knowledge to attack previous knowledge. It also provides simple/intuitive ways for humans to add in knowledge, and clearly shows why agents have taken certain decisions.

10.4 Argument Dialogues

Different agents can engage in dialogues with each other, to share their reasons and try to change the other agents' views. Arguments dialogues can be split into: initial situation/cause, , goal of the argument, and personal aims of the agents.

10.4.1 Dialogue types

Type	Initial Situation	Main Goal	Personal Aims
Persuasion	Conflicting points of view	Resolution of views	Change the other agent's views to their own
Inquiry	Ignorance of multiple agents	Growth of knowledge	Find/destroy a proof/incorrect belief
Deliberation	Need for action	Reach a decision	Influence the outcome towards the 'best'
Neogtiation	Conflict of interests, but co-operation required	Make a deal	Get the best deal
Info-seeking	Ignorance of a single agent	Spread knowledge (general/about other agents)	Gain/Hide/Show personal knowledge

10.4.2 Dialogue System

The system is defined by:

1. The moves each agent can make. e.g. assert, accept, challenge, question
2. Protocol - rules on what moves can/can't be made. e.g. you can't attack an argument you made previously, you must accept arguments you can't attack
3. Strategy for each agent - the 'best' strategy is still a research question
4. When the argument terminates and determining the outcome. e.g. last argument wins