Auctions

▼ Single Item Auctions

An auction is a process for an auctioneer (an agent) to allocate a resource to a bidder (another agent). This is done because the resource is scarce, so multiple agents are interested in it (the demand outstrips the supply). It doesn't need to be a physical thing, it could be processor time in a machine.
An auction mechanism can help agents to achieve their goals (auctioneer: maximise benefit, bidder: lower expenditure) and "reveal some hidden truths" (more on this later on).

▼ Terms

- Common Value: the value of the resource is the same for all bidders.
- Private Value: the value of the resource is private, and can differ between bidders. This is the case more often than not.
- First Price Auction: the winner pays the price of the highest bid.
- Second Price Auction: the winner pays the price of the second highest bid (more on this later).
- Open Cry: all bids are known to all bidders.
- Sealed Bid: bids are private, known only by the bidder and the auctioneer.
- One-Shot Auction: all bids are collected in a single round and then the auction is decided
- Ascending/Descending Auction: bids are made in successive rounds

▼ Mechanisms

▼ English Auction

The most common type of auction for humans, as in Fair Price, Open Cry, Ascending...

It's a First Price, Open Cry and Ascending Auction, so the highest bidder pays the price they name, all the bidders know what bids have been placed and the bids are placed in multiple rounds with the price increasing each round.

An English Auction follows these steps:

1. Auctioneers sets a reserve price as the initial *current price*
2. Agents or bidders bid more than the current price if they want to bid

3. Resource is allocated to the highest bidder, paying the price of their bid

The dominant strategy for this is to bid in small increments, up to your private value.

An English Auction is susceptible to the Winners Curse. That is what happens when a resource of unknown value is auctioned. The private value is assigned based on some estimate, so the winner does not know if they should feel good for winning or bad for overloading or overvaluing the resource.

▼ Dutch Auction

This is the way things have worked for hundreds of years in the Dutch Flower Markets. It is a First Price, Open Cry, Descending Auction. The winner bidder pays the price they stated and all bids are known by all bidders. Let's explain how a Descending Auction works:

1. Auctioneer sets the reserve price at an artificially high amount
2. Auctioneer reduces the price in increments
3. Bidders place a bid when the price reaches what they want to pay.

There is no dominant strategy for this type of auction, and it is also susceptible to the winners curse, because the winner could still think everyone else was willing to pay less for the item.

▼ First Price Sealed Bid Auction

One Shot, First Price, Sealed Bid auction. All bids are collected at once, bidders share their bids privately (in envelopes, for example), and the highest bidder pays the price of their bid.

There are a number of possible strategies:

- Agents bid **their value**, but winning bidder could have bid the amount of (bid number 2) + 1
- Agents bid less than their true value, but how much less? Maybe some kind of function of how much you believe the other agents' values are

There is no dominant strategy, this is strategically equivalent to the Dutch Auction.

▼ Second Price Sealed Bid Auction (Vickrey Auction)

One Shot Auction with sealed bids. The winner is the bidder that pays the highest amount, but they pay the price of the second highest bid.

The dominant strategy is to bid your true value. If you bid more than your true value, you could win but making a loss. If you bid less than your true value, you could lose or win with payment being unaffected. This is because, if you win, since you'll pay the second highest bid, you may as well have bid your true value.

This isn't susceptible to strategic manipulation, and is what we know as Incentive Compatible (more on this later)

▼ Combinatorial Auctions

Combinatorial Auctions handle bundles of items

$$\mathcal{Z} = \{z_1, ..., z_m\}$$

of which we have m items.
The n agents have preferences or valuations over each of the bundles, so a bundle will be worth a value to an agent. The valuation for agent i is defined by

$$v_i : 2^{\mathcal{Z}} \rightarrow \mathbb{R}$$

An **outcome** in a CA is an allocation of goods to agents. This means we give agents particular goods. The valuation we saw before will play a part on determining how good an allocation is. We don't necessarily allocate all goods nor goods to all agents.
The allocation of goods to agents is represented as

$$alloc(\mathcal{Z}, Ag)$$

We have n sets of goods, having one allocated to each agent, so each set of goods i has to be a subset of all the items

$$z_i \subseteq \mathcal{Z}$$

Also, the interception of any two different pairs of goods has to be empty (mutually disjoint), meaning each good has been allocated only once.

$$z_i \cap z_j = \emptyset : i \neq j$$

This means we can't give the same good twice, since different agents will get different goods.

We can think of a combinatorial auction as a social welfare mechanism that takes in bundles of goods and valuations of the agents that get those goods (Z_n and v_n) that sums up all the valuations the agents have of the set of goods they have been allocated.

$$sw(Z_1, ..., Z_n, v_1, ..., v_n) = \sum_{i=1}^{n} v_i(Z_i)$$

▼ Winner Determination

We can think of a combinatorial auction as a function that maximises social welfare.

The winning allocation is going to be the one where all bidders get the most value that they can out of what can be allocated.

$$Z_1^*, ..., Z_n^* = arg_{(Z_1,...,Z_n) \in alloc(Z,Ag)} max\ sw(Z_1, ..., Z_n, v_1, ..., v_n)$$

Computing the optimal allocation is known as the Winner Determination Problem. It's a difficult problem to solve because:

- Agents may not give their true valuation, manipulating the auction
- Representation of the valuation function can be impractical because of the size of it. If you try to map the valuation to each possible bundles of goods, it will grow a lot in size.
- The problem for determining the winner is **NP-HARD.**

▼ Bidding Languages

We can represent an agent's valuation of a particular set of goods using a bidding language. A bidding language can help us deal with the representation issue. The main building block of a BD is atomic bids. These take in a set of items and a price.

$$\beta(Z, p)$$

This means I would pay *p* if I get the items in *Z*. Expressed more formally,

$$if Z \subseteq Z'$$

then I pay *p*, since my bid is **satisfied**. So, if the bundle of goods you get contains everything in Z, and even if it contains more, it is worth *p* for you, since your bid is satisfied.

▼ Example

If you say you'd bid 4 for items a and b

$$\beta(\{a, b\}, 4)$$

and you're presented with the bundle a, b and c

$$\{a, b, c\}$$

that's satisfied, because you'll get all the items that you wanted.

However, if you say you'll happily pay 5 for the set of goods c and d

$$\beta(\{c, d\}, 5)$$

and you're presented with the set of goods b, d and e

$$\{b, d, e\}$$

you'll bid 0, since it doesn't contain all the items you want.

Expressed more formally, the valuation of an atomic bid for a particular set of goods Z' is p if what you're bidding for Z is a subset of Z'

$$V\beta(Z') = p \; if \; Z \subseteq Z'$$

$$V\beta(Z') = 0 \; otherwise$$

▼ Combining bids

We can use logical operators to combine multiple bids.

▼ Logical XOR operator

$$\beta_1 = (\{a, b\}, 3) XOR (\{c, d\}, 5)$$

In the last expression, we use XOR (one but not all), meaning we will pay either 3 for a bundle of goods that contains a and b or 5 for a bundle of goods that contains c and d, but we will not pay both.

▼ Example

$$v_\beta 1(\{a\}) = 0$$

$$v_\beta 1(\{a, b\}) = 3$$

$$v_\beta 1(\{c, d\}) = 5$$

$$v_\beta 1(\{a, b, c, d\}) = 5$$

Expressed more formally

$$v_\beta(Z') = max\{p_i | Z_i \subseteq Z'\}$$

$$v_\beta(Z') = 0 \; if \; Z_i \nsubseteq \forall Z_i$$

If our bid is not satisfied at all we will bid 0, but if it is satisfied, we will bid the maximum possible, since that is what it is truly worth to us.

▼ Logical OR operator

$$\beta_1 = (\{a, b\}, 3) OR (\{c, d\}, 5)$$

In the last expression, we can have one, the other or both.

▼ Example

$$v_\beta 1(\{a\}) = 0$$

$$v_\beta 1(\{a, b\}) = 3$$

$$v_\beta 1(\{c, d\}) = 5$$

$$v_\beta 1(\{a, b, c, d\}) = 8$$

To find

$$v_\beta(Z')$$

for OR-bids, the process is a little bit different.

We have to find a set of atomic bids W, such that

- Every bid is satisfied by Z'
- Each pair of bids has mutually disjoint goods as in

$$Z_i \cap Z_j = \emptyset \; \forall i, j$$

- There is no other set of bids W' that has a greater total value

▼ VCG Mechanism

In Combinatorial Auctions agents can misrepresent their true valuation to gain an unfair advantage, making them easy to manipulate We want a mechanism where agents who act rationally will be able to maximise their social welfare. We want the dominant strategy to be to tell the truth.

A mechanism with these properties is known as incentive-compatible. The Vickrey-Clarke-Groves Mechanism provides this incentive-compatibility. It is actually a generalisation of the Vickrey Auction we saw before.

▼ How it works

There is a special valuation that agents can have, called an indifferent valuation, represented as

$$v^0(Z) = 0 \forall Z \subseteq \mathcal{Z}$$

For any subset of items the agent bids for, the indifferent valuation is 0. We use this to represent when an agent is going to be uninterested in an item.

We can also calculate social welfare for all agents except agent i. This is, how much does a given allocation benefit the agents except agent i.

$$sw_{-i}(Z_1, ..., Z_n) = \sum_{j \in Ag - i} v_j(Z_j)$$

The VCG mechanism goes as follows:

1. Each agent declares a valuation function (this could not be their true valuation)

$$\hat{v}_i$$

2. We compute the allocation

$$Z_i^*, ..., Z_n^*$$

that maximises social welfare using

$$\hat{v_1}, ..., \hat{v_n}$$

3. To disincentivize lying, each agent pays compensation into the system. We look at the allocation chosen if the agent declared 0 valuation. To calculate the compensation of a particular agent, we compare the social welfare of the rest of the group as if agent i wasn't there (declared 0), and we subtract it the social welfare of the group as if i declared a valuation.

$$p_i = sw_{-i}(Z_1', ..., Z_n', \hat{v_1}, \hat{v_i}, \hat{v_n}) - sw_{-i}(Z_1', ..., Z_n', \hat{v_1}, ..., \hat{v_i}, ..., \hat{v_n})$$

That way we know how much agent i would be costing the group in case of being a lier.

▼ Example

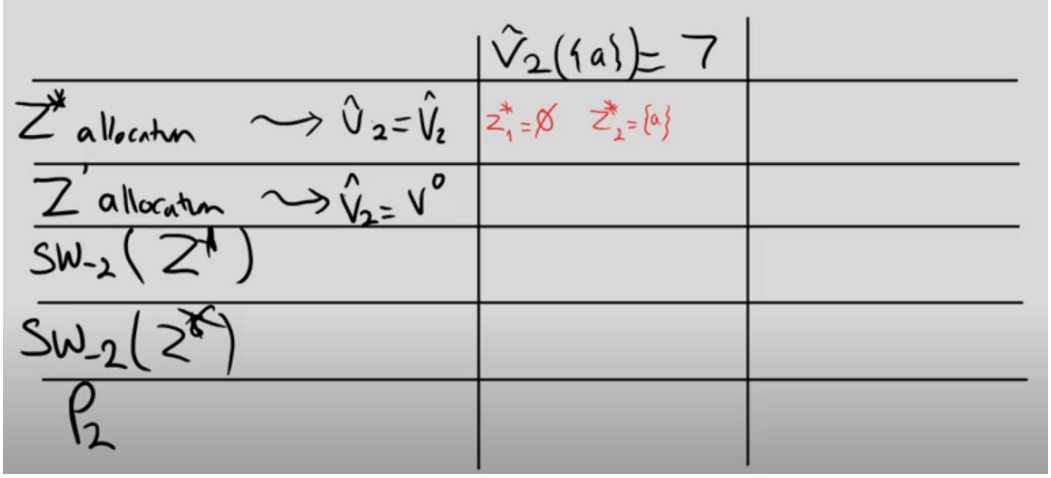
We have one item "a" and two agents 1 and 2. Agent 1's true valuation of a is 5, and agent 2's true valuation of a is 3.

$$\mathcal{Z} = \{a\}; Ag = \{1, 2\}; v_1(\{a\}) = 5; v_2(\{a\}) = 3$$

We assume agent 1 is telling the truth, and we'll see what happens when agent 2 lies and agent 1 tells the truth.

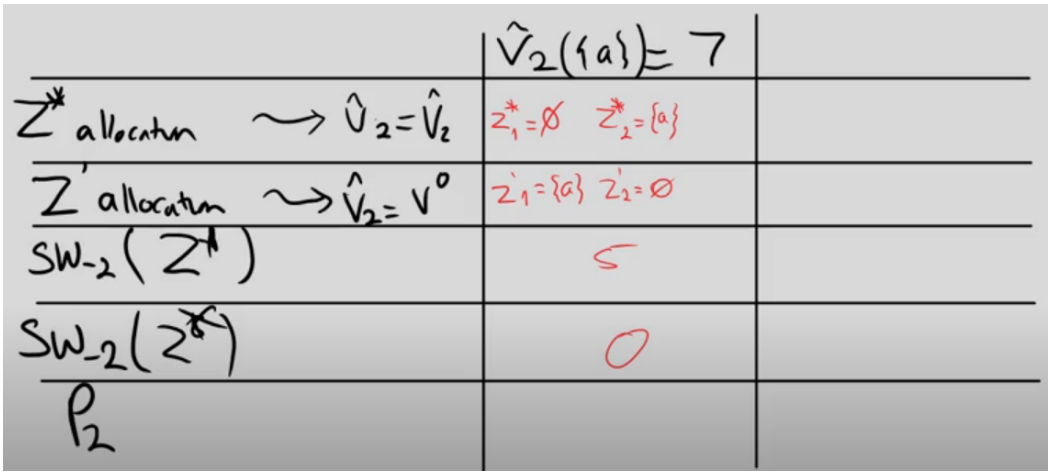
▼ When agent 2 lies

We calculate the Z* allocation of goods to agents



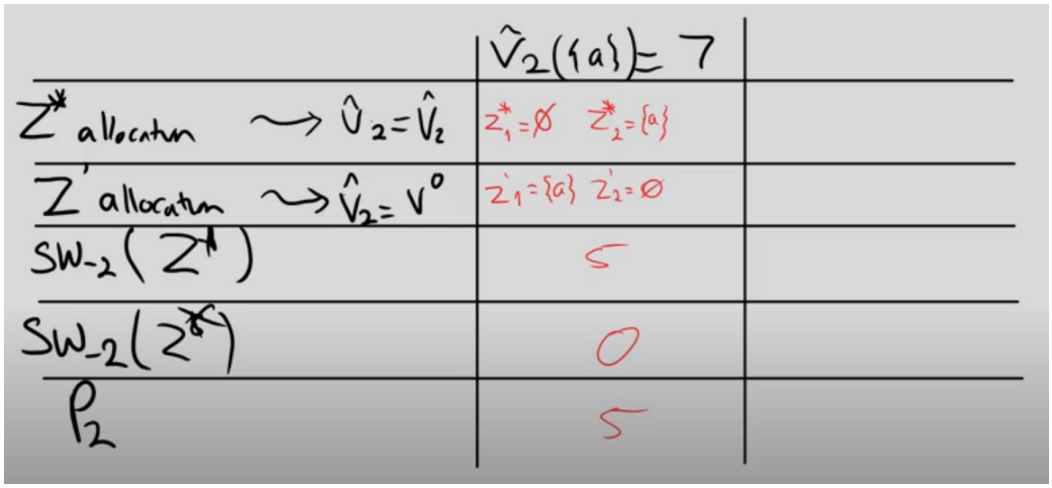
That means agent 2 is going to win the item, since their bid was higher.

If agent 2 declared 0, Agent 1 would get it, since their bid would be higher.



Now we look at the social welfare for all the agents except agent 2 for these 2 different allocations. We can see that for the Z' allocation, SW is 5 (only agent 1's), but for Z* (everyone except 2), it's 0.

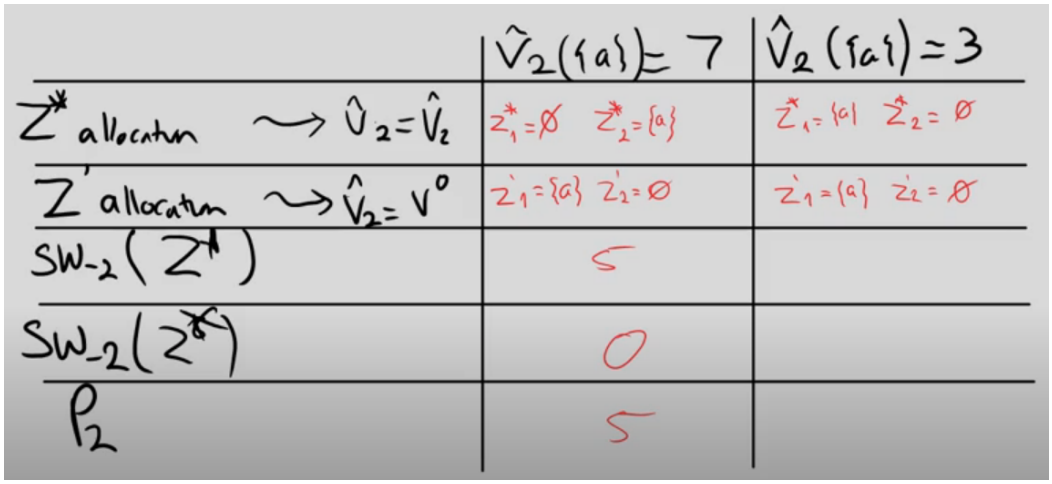
We subtract both and get 5.



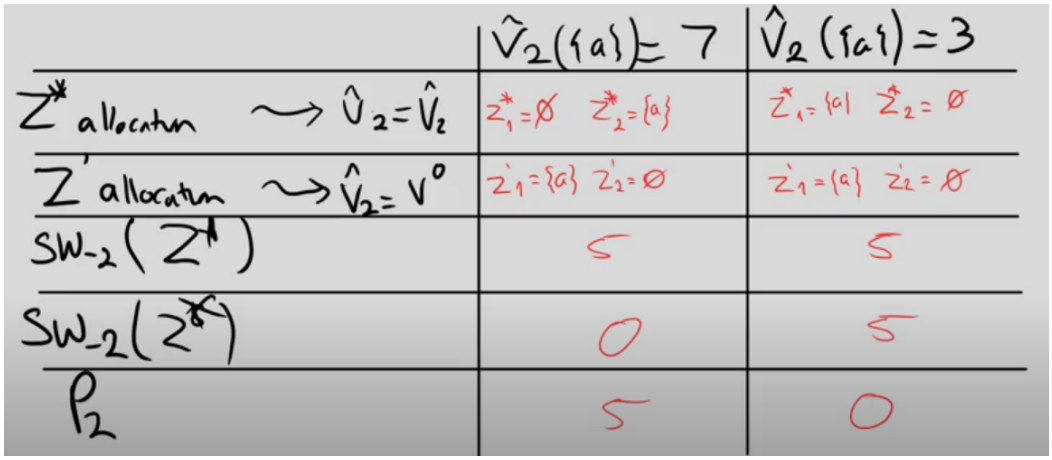
This shows that, because agent 2 has lied, he has deprived the group of 5 social welfare, and now has to pay in 5. When comparing that to when he uses his true valuation (3), we see he's making a loss

▼ When agent 2 tells the truth

This time, agent 2 tells the truth and agent 1 wins the auction



Because agent 2 didn't get the good, the fact he declares 0 does not affect the allocation



Now agent 2 does not have to pay anything.

▼ Bargaining

Negotiation can be a complex task. They often happen in rounds of proposals. Parties will accept or reject until some agreement is reached.

Single attribute negotiating is simple to analyse, and it's easy to spot when the other party makes a concession (for example, the price of an item, and a concession would be lowering or raising the proposed price).

Multiple issue negotiating is harder to analyse, and concessions can be more difficult to recognise. For example, when buying a car may include negotiations on price, features or warranty, and the salesman might "make concessions" that are not really a true concession but give the impression of being one (like giving something for free).

Multiple issue negotiating leads to exponential blow up in the number of possible deals, and also the number of agents that negotiate and the way of interaction (1:1, n:1, n:m) can add to that complexity. We will focus on single issue negotiation.

▼ Terms

- Negotiation set: all possible proposals that can be made by agents
- Protocol: defines a legal proposal, often a function of negotiation history
- Strategies: determine which proposal an agent can make
- Rule: determines when a deal has been struck and what the agreement deal is (what is going to happen once an agreement has been made)

▼ Alternating Offers Protocol

It's a 1-1 protocol in a sequence of rounds.

Agent 1 will make a proposal and agent 2 will think about it and either accept it or reject it.

If they accept agent 1's proposal, they are going to implement it.

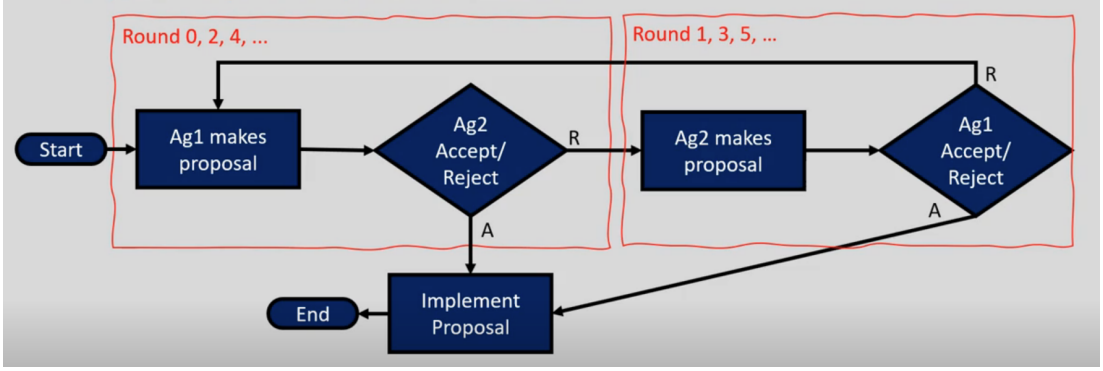
If they reject it, they are going to move on to the next round.

The making of a proposal and the opposite agent accepting or rejecting it is a round.

In the next round, it's the agent 2 that makes a proposal (they alternate) and agent 1 decides if they accept or reject it.

If they accept, they implement it.

If they reject, they move to the next round, where agent 1 will make a counteroffer.



We need some strategies for these negotiations in case agents keep on rejecting

Conflict happens when agents can't agree, and it's the worst outcome, the ultimate disagreement.

$$\Theta$$

Agents want to maximise utility, so an agent would rather get a bad deal as opposed to not deal at all.

We model the strategies as if we were dividing an apple pie, so agents will give their proposal in this form

$$(x, 1 - x)$$

Agent 1 will get a slice of the pie of size x, so agent 2 will get a slice of size 1 - x.

The value of x will be between 0 and 1.

▼ Strategies for alternating proposals: Patient Players

Let's see it having one round only:

1. Agent 1 puts forward a proposal. Because it is one round, agent 2 has to accept agent 1's proposal. Otherwise, we get conflict and that is worse than anything. Since agent 1 wants to maximise their utility, proposes (1, 0). Agent 2 has to accept since getting none of the pie is worst than getting no deal at all.

Let's see it having two rounds:

1. Same as in last example except agent 2 rejects it
2. Agent 2 proposes (0, 1), and agent 1 must accept, because, if he doesn't accept it, they'll have conflict.

This is called an **ultimatum game**, where the last mover has all the power. This will work the same for any fixed number of rounds. If it's an odd number 1 will be the final mover and agent 2 will be when the number of rounds is even.

If there is infinite rounds, it could go on forever, because agent 1's strategy will be to propose (1,0) and reject everything else. In this case, agent 2's best response is to accept, and that is in Nash Equilibrium. This means it would be better for agent 1 to share their strategy with agent 2, in order to "make them understand" accepting is their best option. Otherwise the negotiation could last a lifetime.

▼ Strategies for alternating proposals: Impatient Players

▼ Discount Factor

For outcome X, and times t1 and t2, the agents prefer X at t1. The agents are impatient, this is represented by a discount factor, an adversarial feelings towards time between 1 and 0, but not including 1

$$0 \leq \delta_i < 1$$

This makes the value of the "pie slice" fall as time progresses. The agent that has a higher discount factor is going to have more tolerance of time. For example, if it's 0.5, the value of the pie is going to half each time step. If it were 0.75, it's going to reduce by 3/4, so the lower discount factor means the agent is more impatient.

At time step t0, the value of the pie is x

$$d_i^0 x = x$$

At time step t1, it will be

$$d_i^1 x = d_i x$$

At time tk it will be

$$d_i^k x = d_i^k x$$

The value reduces by the discount factor each time step.

In a negotiation with only one round, this strategy behaves the same way as the Patient Players one. Agent 2 has to accept agent 1's proposal otherwise conflict occurs, therefore agent 1 proposes (1, 0).

With 2 rounds, the value of the pie is reduced by the discount factors in the second round. So in the first round, ag1 proposes (1,0) and agent 2 rejects. In round 2, ag2 proposes (0,1). Agent 1 has to accept, since this is an **ultimatum game**. At the same time, the pie is only worth the value of agent 2's discount factor to it.

Agent 1 could take into account agent 2's discount factor, so if ag1 proposes (1-d2, d2), ag2 should accept since it will not benefit from rejecting ag1's proposal, so this is in Nash Equilibrium.

▼ Bargaining for task allocation

Given a set of tasks, the set of agents gets an initial allocation of those tasks, and can then bargain with each other to divide them differently. We write this task-oriented domain as a triple of a set of tasks T, a set of agents Ag and a cost function c

$$\langle T, Ag, c \rangle$$

The function c is the cost of executing some tasks. It takes in a power set of tasks and maps it to a real number

$$C : 2^T \rightarrow \mathbb{R}$$

The function must be monotonic, such that when you take away tasks, the cost does not increase.

$$T_1, T_2 \subseteq T; T_1 \subseteq T_2; c(T_1) < c(T_2)$$

Also

$$c(\emptyset) = 0$$

▼ Encounter

An encounter is a Task Environment where agents are allocated tasks in a default setting

$$\langle T_1, T_2, ..., T_n \rangle$$

T1 to Tn are subsets of the set of tasks T.

▼ Deals

Agents can try to strike deals with each other. Imagine we have two agents, 1 and 2, and an encounter with t1 and t2

$$Ag = \{1, 2\}; \langle T_1, T_2 \rangle$$

meaning 1 is initially going to do the tasks in T1 and 2 is initially going to do the tasks in T2.

A deal is an allocation of these tasks that have been allocated collectively

$$T_1 \cup T_2; \delta = (D_1, D_2)$$

the deal has two parts, D1 and D2. D1 are the tasks agent 1 said they could do, and D2 is the same for agent 2.

A Pure Deal is a in which that all tasks that were allocated initially will be done after the deal.

$$T_1 \cup T_2 = D_1 \cup D_2$$

The cost of Agent i doing some tasks is

$$c(T_i)$$

The cost to Agent i for a deal is

$$cost_i(\delta)$$

The Utility of a deal tells us how much an agent gains from doing the deal. It's the difference between the agent's existing task minus the cost of the deal.

$$utility_i(\delta) = c(T_i) - cost_i(\delta)$$

If it were negative, we know they would be worse off doing the deal than they would be without doing it.

If there is no agreement of a deal, the conflict deal happens.

$$\Theta$$

In this case we do the original allocation

$$\langle T_1, T_2 \rangle$$

▼ Dominance and Pareto Optimality

We must think of how deals compare to each other.

Consider 2 deals delta1 and delta2. The deal delta1 is dominant if it's at least as good to all the agents as delta2 is. That means deal 1 is dominant if every agent is going to gain the same amount or more than they would have won if they had gone for deal 2, no one is going to suffer for going to deal 1 over deal 2

$$\forall i \in \{1, 2\} : utility_i(\delta_1) \geq utility_i(\delta_2)$$

And at least one agent is strictly better off by doing deal 1.

$$\exists i \in \{1, 2\} : utility_i(\delta_1) > utility_i(\delta_2)$$

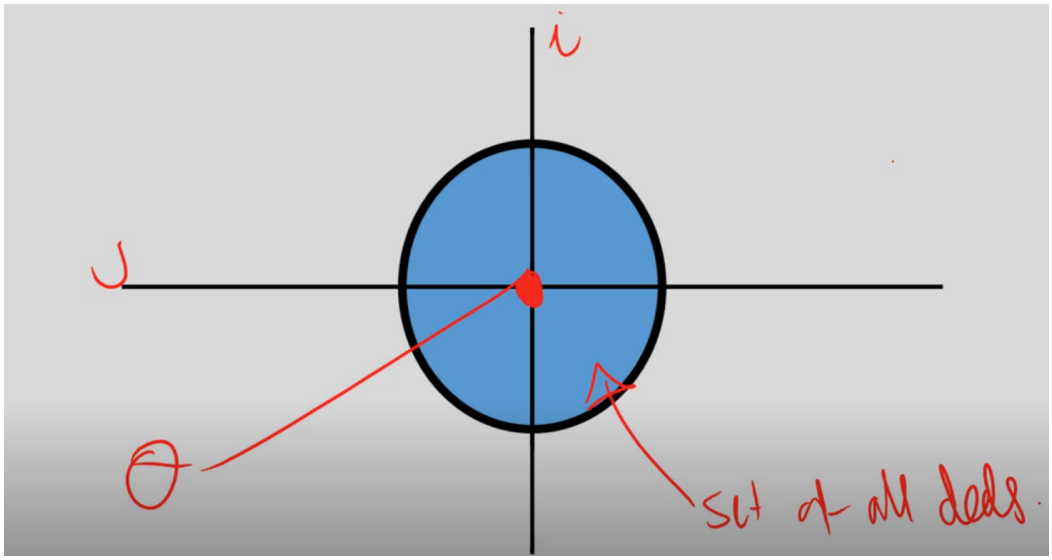
If only the first one applies, but it's not strictly better for one of them, then it is only a weak dominance.

If a deal is not dominated by another deal, then it is pareto optimal.

If a deal weakly dominates the conflict deal, then it is individual rational.

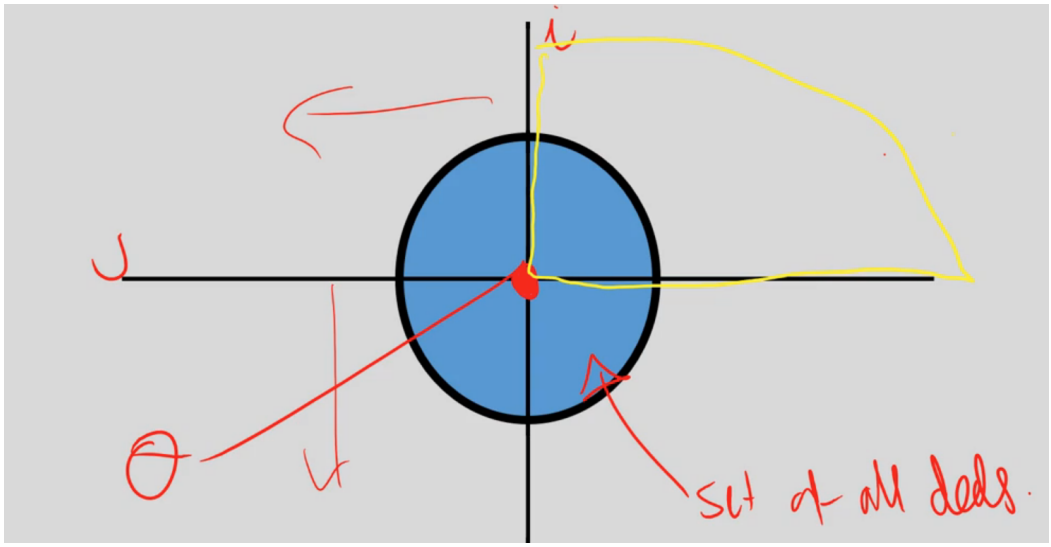
▼ The negation set

The negotiation set is the set of all possible deals that can be done. We can use a diagram that plots utilities for agents i and j to show what deals are possible for both of them

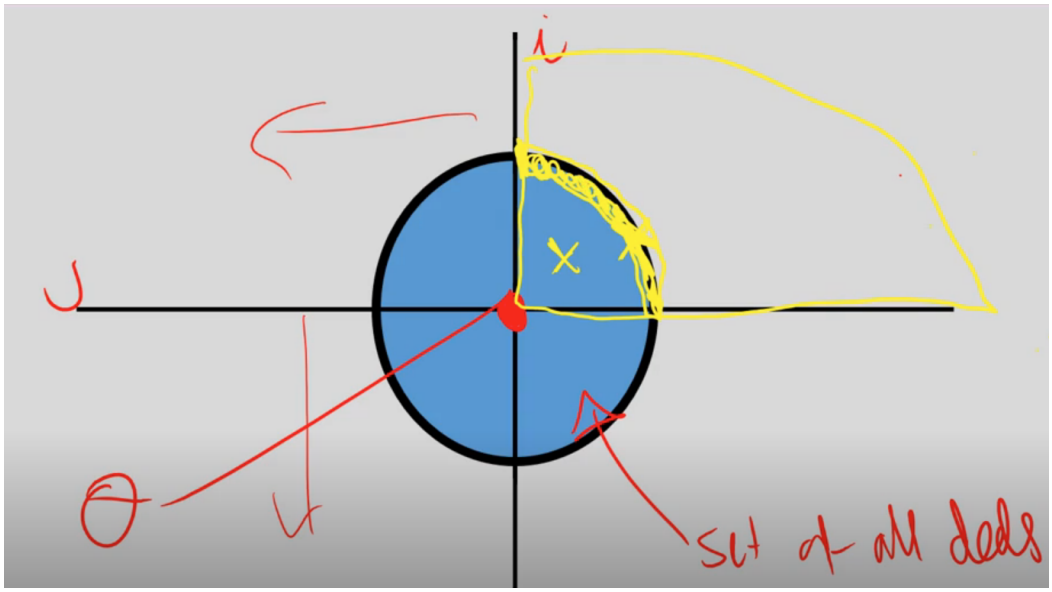


This means there is no point in proposing a deal when the conflict deal is going to be preferred to it (red dot).

Anywhere left of the line is out of the question for agent j, and everything underneath the horizontal line is out of the question for agent i. So any deal in the top right quadrant is individual rational.



We also want the deals to be pareto optimal. These are the deals present in the black circumference's section in that quadrant (painted over in yellow)



Where this circumference crosses the axis, the utility of one of the agents will be equal to the one they'd get in the conflict set, but it's still individually rational to do that, since they don't lose anything.

▼ Monotonic Concession Protocol

It's a bargaining protocol consisting of negotiations in rounds. Both agents simultaneously propose a deal from the negotiation set.

An agreement will occur if either agent 1 isn't worse off doing agent 2's deal or 2 isn't worse off doing 1's deal.

$$utility_1(\delta_2) \geq utility_1(\delta_1) \vee utility_1(\delta_1) \geq utility_2(\delta_2)$$

If the agreement does occur, we implement the deal from the true side of the argument (or random if both are true). So if the left side is true, we implement the deal delta2, if the right side is true, delta1.

If there is no agreement reached, we repeat and propose simultaneous deals from te negotiation set. Agents can not propose a deal that is "worse" than the one they just proposed, so both agents will consistently make things not as worse for the other.

If there is no agreement after a fixed amount of rounds, conflict deal occurs. With this, we can't guarantee a quick agreement will happen, since there is an exponential number of deals.

▼ Zeuthen Strategy

This tells agents how to behave when they are in a Monotonic Concession Protocol. It tells agents:

- What should their 1st proposal be: their most preferred deal.

- Who should concede: the agent with most to lose if conflict were to occur. This willingness to concede is characterised with Risk

$$Risk = \frac{Loss\ caused\ by\ conceding\ and\ accepting\ other\ agent's\ deal}{Loss\ caused\ by\ conflict}$$

The larger the difference between what I would lose by conceding and what I would lose by having a conflict deal, the more I have to lose. This value goes between 0 and 1.

- How much should they concede by: just enough to tip the balance of risk to the other agent so they're the one who concedes but without wasting their own utility. in the case of equal risks, it will be chosen at random.

This strategy guarantees termination and that every agreement reached is pareto optimal, as it's going to be better than the conflict. It's also in Nash Equilibrium, since agents can do no better than to use this strategy if they know the other agent is also using it.

▼ Argumentation

How do agents decide what to believe?

A dialogue in a court room gives a judge a lot of information. The jury has to establish a rationally justifiable position with regard to the arguments. The evidence put forward could have inconsistencies or contradictions (if there aren't any of these, there is no argument needed).

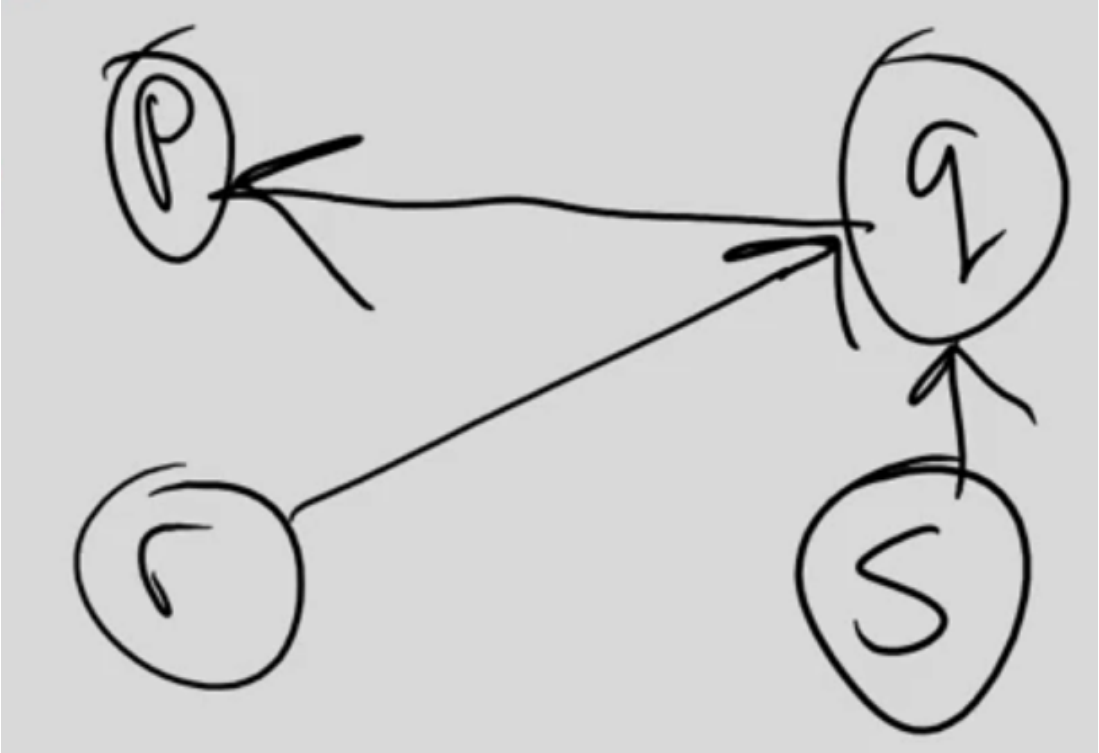
Argumentation provides principal techniques for handling this inconsistency in order to know what to believe.

There are two types of argumentation: abstract argumentation or deductive argumentation.

▼ Abstract Argumentation (Dung Style)

Arguments can interact with one another, so accepting one may mean having to reject another, like accepting the grass is green means having to reject the grass is blue.

The interaction between arguments can be represented as a directed graph. Each node is an argument and a directed edge from node q to node p indicates accepting q means having to reject p, expressed as "q attacks p".



This can be used to know if, given a set of arguments, the position you take is going to be rationally justifiable or not.

Using that graph we can compute it's Preferred Extension. To do that, we must analyse the it's arguments.

- A set is conflict-free if no member attacks another member. In the graph from before, this could be

$$\emptyset, p, q, r, \{p, r\}, \{r, s\}, \{p, s\}, \{r, s, p\}$$

- An argument is defended if it's attacker is attacked (for example, r defends p from q). A set is mutually defensive if all attacked members are defended by a member. In the graph from before, these would be

$$\emptyset, r, s, \{r, s\}, \{p, r\}, \{p, s\}, \{p, r, s\}$$

- A set is admissible if it's both conflict-free and mutually defensive. That means the argumental position we are holding is, because none of our arguments contradict each other and if there is any argument in our position that is attacked by something, we have backup to help us out

$$\emptyset, r, s, \{r, s\}, \{p, r\}, \{p, s\}, \{p, r, s\}$$

A Preferred Extension is an admissible set which can't have any other members added without becoming inadmissible. With the graph from before,

this can only be

$$\{p, r, s\}$$

if we only took one of them, we could still add the others, but we can't add q because it would stop being conflict-free.

▼ Skeptical/Credulous acceptance

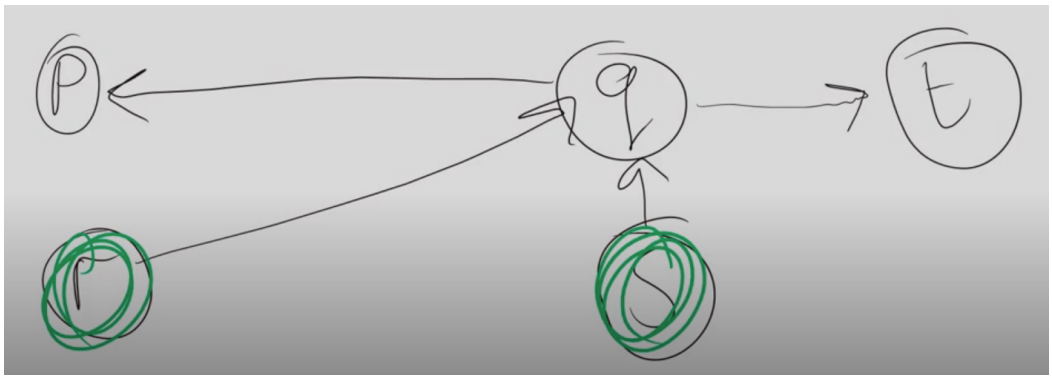
It could be the case that we have multiple preferred extensions of a set of arguments.

We want to pick one based on how strong their arguments are. Stronger arguments will show up in more preferred extensions. If an argument appears in all preferred extensions, it is skeptically accepted (a skeptic is harder to convince). If it appears in fewer preferred extensions, it is credulously accepted.

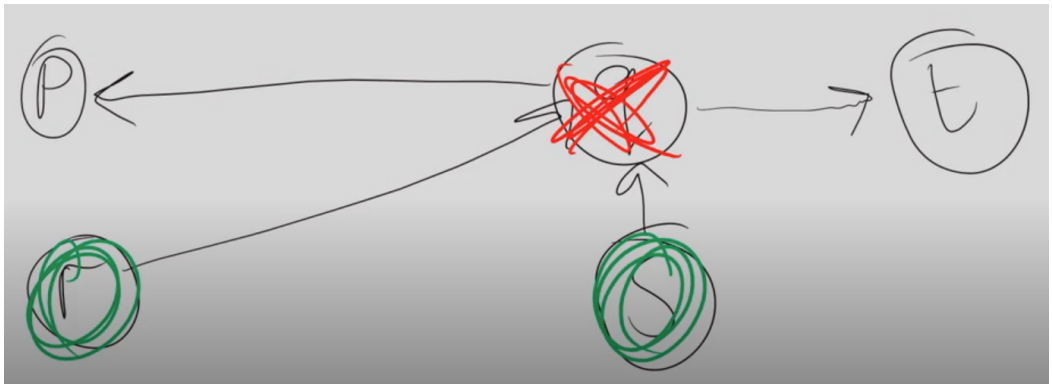
▼ Grounded extension

It's another way to generate a rationally justifiable position. Every argument set will have a GE, but it might be empty. An argument is either IN or OUT of the extension.

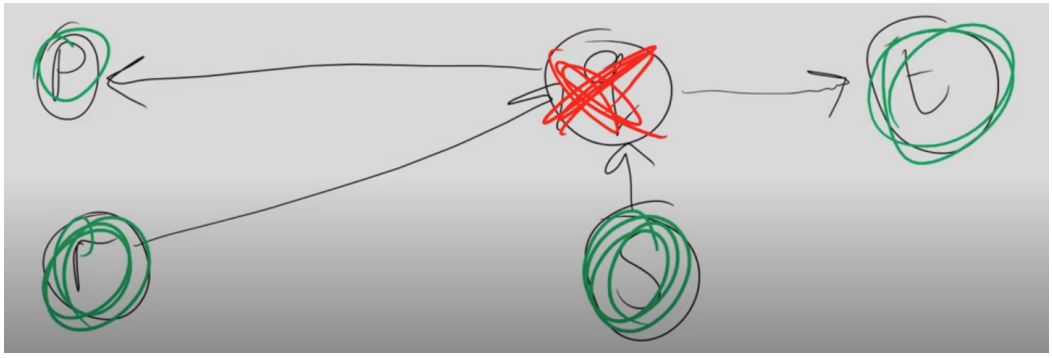
We start with IN arguments that are not attacked. For example



Then, we remove anything that they attack. In this case, q is OUT.



Now we can add arguments that are not being attacked



In this case, our grounded extension is

$$GE = \{p, r, s, t\}$$

▼ Deductive Argumentation

This type of argumentation contains a Database of Evidence expressed as classical logical formulae DB. It is agreed between the set of agents Ag as "common ground".

A deductive argument is a pair (support, conclusion)

Support is a collection of arguments in the database that are logically consistent that support the conclusion the agent is trying to make. The agent is trying to convince other agents of this. No other subset satisfies this, we just want the bare minimum arguments to convince people.

We say an argument X attacks Y if X contradicts Y

$$X \equiv \neg Y$$

Imagine we have two arguments A1=(s1, c1), A2=(s2, c2)

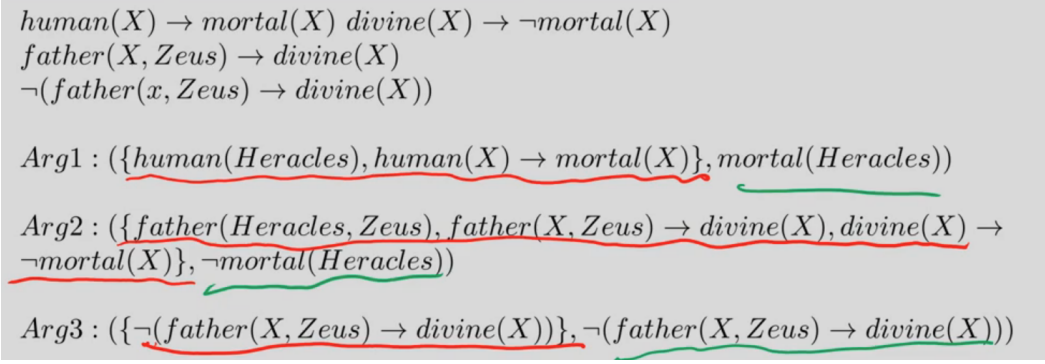
An argument can be defeated in 2 different ways:

- Rebuttal: the conclusion of A1 contradicts directly the conclusion of A2. Then we say A2 has been rebutted by A1.
- Undercut: if a conclusion that came from A1 brings into question a supporting statement of A2, argument 1 undercuts A2. So c1 attacks something in s2.

If the conclusion attacks another conclusion, it's rebuttal. If the conclusion attacks another argument's support, it's an undercut.

▼ Example

We have the following database (supports underlined in red, conclusions underlined in green)



This brings into question Heracles' bloodline, saying Zeus is not his father.

We can see that the conclusion of Arg2 is directly contradicting the conclusion of Arg1.

Arg2 rebuts Arg1

The conclusion of Arg3 contradicts the support of Arg2

Arg3 undercuts Arg2

We can rank arguments by their acceptability. In this case Arg1 will be the most acceptable, Arg2 would be the next, and Arg3 would be the least.

The key thing about argumentation is that, if you're trying to negotiate with an agent to carry out a task for you, you have to convince the agent that your argument "you should do this for me" is acceptable.