Introduction to Multi-Agent Systems

Yoav Shoham
(Written with Trond Grenager)

April 30, 2002

Chapter 7

Mechanism Design

7.1 Overview

In the preceding chapters we presented essential elements of game theory. Throughout the discussion the issue was framed as follows. Given an interaction among a set of agents, first we need to decide how to represent this interaction, and second, given this representation, we need to predict or prescribe the outcome of this interaction. The representations included the normal and extensive forms (as well as several others), and the analysis consisted of investigating the Nash equilibrium and various refinements of it. Essential, however, was the we started with a *given* strategic interaction.

We now turn to what is sometimes called "inverse" game theory. Rather than investigate a given strategic interaction, we start with certain desired behaviors on the part of agents, and ask what strategic interaction among these agents might give rise to these behaviors. Roughly speaking, from the technical point of view this will translate to the following: We will assume unknown individual utility functions, and ask whether we can design a game such that in the equilibrium of that game the agents exhibit a certain desired behavior no matter what their secret utility functions actually are. This area, called *mechanism design* or *implementation theory*, is perhaps the most "computer scientific" part of game theory, since it concerns itself with designing effective protocols for distributed systems. The key difference from the traditional work in distributed systems is that in the current setting the distributed elements are not necessarily cooperative, and must be motivated to play their part. For this reason one can think of mechanism design as an exercise in "incentive engineering."

Mechanism design has many applications. The most famous of these is the design of auctions, such as the popular online consumer auctions or the more somber government auctions of electromagnetic spectrum. We return to the topic of auctions later, but here are two different examples that illustrate the problem of mechanism design.

Example 7.1.1 Strategic Voting You are taking four children – Will, Liam, Vic and Ray – to play in a schoolyard, and need to decide on what sport they will all play. You can choose among basketball, soccer, and volleyball. You don't know the kids well, so you ask them all to tell you their preferred choice, announcing (reasonably enough) that you will pick the sport that the majority of kids voted for (breaking ties at random). What will happen?

Consider the following situation, in which the true preferences of the kids are as as follows (each column describes the preferences of the child in descending order):

	Will	Liam	Vic	Ray
1	V	V	S	B
2	B	B	B	S
3	S	S	V	V

Will, Liam, and Vic are regular kids and tell you their true preferences. But little Ray goes through the following reasoning process. Since he does know his friends, he knows what sport each will vote for. He thus knows that if he votes for his true passion – basketball – he'll end up playing volleyball with certainty. So he votes for soccer, ensuring that he has a 50% probability of avoiding the detested volleyball.

Is there anything you can do to prevent such manipulation by little Ray?

A rather different example is taken from the networking domain.

Example 7.1.2 Shortest-Path Routing With Selfish Agents You wish to route a message between two nodes in a communication network. Each link in the network is owned by a different company, which experiences a certain cost for transmitting the message. Assume that this cost is private knowledge of the companies, and that they wish to maximize their revenues. You wish to route the message along the least-cost route (note that this is different from wishing to pay the least amount of money; you care about the total costs of the companies, not your total expenditures). Your task would be easy if the companies revealed their true costs, since then you'd need to compute a simple shortest path in a weighted graph; but how can you ensure that the companies indeed reveal the truth? [[[expand example]]]

The remainder of this chapter is organized as follows. In the next section, we present a formal model for the mechanism design problem, and present some

general mechanism design results. In Section **??**, we discuss auctions, which are a key application of mechanism design. Finally, in Section 7.5.3 we discuss some other applications of mechanism design in computer science.

7.2 A Formal Model And Some General Results

Before we begin to answer the questions posed in the preceding examples, let us set things up formally.

Definition 7.2.1 (Mechanism Design Problem) *A mechanism design problem M is a tuple (N, O, U, C), where*

- *N is a set of agents,*
- *O is a set of outcomes,*
- *$U = U_1 \times \cdots \times U_n$, where U_i is the set of possible utility functions for agent $i \in N$. Each $u_i \in U_i$, $u_i : O \to \Re$, is a possible utility function for agent i,[1] and*
- *$C : U \to 2^O$ is a function mapping agents' utilities to subsets of outcomes, those desired by the mechanism designer.*

We can use this problem to formalize the voting example above. In this problem, there are four agents, three possible outcomes (soccer, volleyball and basketball), the set of utility function for each child consists of all possible mappings from outcomes to the real numbers, and for every 4-tuple of utility functions the desired outcomes consist of those in which the outcome has the maximal utility for the largest set of agents. In the particular instance of utility examples given there (which is really a set of instances, since we specify only the qualitative preferences of each child and not the numerical utility), there is one desired outcome – volleyball.

Now that we have a definition of a mechanism-design problem, we need to define a mechanism.

Definition 7.2.2 (Mechanism) *Given a mechanism design problem $M = (N, O, U, C)$,* a mechanism for *M is a pair (A, μ), where*

- *$A = A_1 \times \cdots \times A_n$, where A_i is the set of actions available to agent $i \in N$, and*
- *$\mu : A \to \Pi(O)$ maps each action profile to a distribution over outcomes. For convenience, when $\mu(a)(o) = 1$ we write $\mu(a) = o$.*

[1] The function u_i in an instance of the problem is often called the agent's *type*.

Note that a problem and a mechanism as above together define a set of games (N, A, O, μ, u), one for each $u \in U$.[2]

Informally speaking, a solution to a mechanism design problem is a mechanism which always defines a game in which every equilibrium necessarily leads to one of the desired outcomes. More formally, we have the following definition.

Definition 7.2.3 (Mechanism Design Solution) *Given a mechanism design problem $M = (N, O, U, C)$, a mechanism (A, μ) for M is a* Nash-solution *of M iff it is the case that for any preference profile $u \in U$ and action profile $a^* \in A$, if a^* is a Nash equilibrium of the game (N, A, O, μ, u) then for all $o \in O$, if $\mu(a^*)(o) > 0$ then $o \in C(u)$.*

In the sports example above, the pair consisting of "each child votes one choice" and "the sport selected is one with most votes" is a well-formed mechanism for the problem, since it specifies the actions available to each child and the outcome depending on the choices made. But it is clearly not a solution, since in the particular instance described there it is an equilibrium for all kids but Ray to vote their first choice, and for Ray to vote his second, leading to a 50% probability that the outcome will not be the one desired by the mechanism designer.

The above definition is specific to the Nash equilibrium, but clearly we could define similar definitions in terms of alternative solution concepts. In general, given a mechanism design problem M and a solution concept $\mathcal{S}$, we will speak about a mechanism (A, μ) forming an *$\mathcal{S}$-solution of M*, or an *S-implementation of M*. When talking about an $\mathcal{S}$-solution, we will assume that the mechanism always gives rise to games in which that solution exists. Of course, when $\mathcal{S}$ is the Nash equilibrium, this is not a substantive assumption, since Nash equilibria are guaranteed to exist. But it is a substantive assumption when speaking, for example, about dominant-strategy solutions.

Finally, we need to extend the concept of mechanism design to Bayesian settings. Given our understanding of Bayesian games (see Section 5.6), this extension is also straightforward. A Bayesian mechanism design problem is a distribution over a set of utility functions and a partition over this set for each agent; a mechanism remains as before, but it gives rise to a Bayesian game instead of a normal form game. Finally, a mechanism is a solution to the problem if the Bayes-Nash equilibria of any (Bayesian) game it creates always lead to desired outcomes.

A classical example of Bayesian mechanism design is auction design. While we devote a more lengthy discussion to auctions in Section **??**, the basic idea is as follows. The designer wishes (for example) to ensure that the bidder with the highest valuation for a given item win the auction, but the valuations of the agents are all private. The outcomes consist of allocating the item (in the case of a simple, single-item auction) to one of the agents, and having the agents make

[2]Note that this is more general than the formulation of a game presented in Chapter 5; it does not equate action vectors directly with outcomes but rather maps the former to a distribution over the latter.

or receive some payments. The auction rules define the actions available to the agents (the "bidding rules"), and the mapping from action vectors to outcomes ("allocation rules" and "payment rules": who wins and who pays what as a function of the bidding). If we assume that the valuations are drawn from some known distribution, each particular auction design and particular set of agents define a Bayesian game, in which the signal of each agent is (for example) his own valuation. A typical goal of the auction designer, in this case, is to ensure that in all such games the the winner of the auction is the person with the highest valuation.

With this we are ready for some of the main results in the theory of mechanism design.

7.2.1 A Positive Result: The Revelation Principle

A *direct* mechanism is one in which the only action available to agents is to announce a preference function; that is, $A_i = U_i$. For example, in a simple, single-item auction setting, the only action available is to announce one's value for the item. Since the set of actions is the set of all preference functions, agents may lie and announce a preference function $\hat{u}_i$ that is different from his true preference function u_i.[3] A direct mechanism is said to be *truthful* or *incentive compatible*[4] if, for any preference vector u, in the game defined by the mechanism it is a dominant strategy for every agent i to announce his true preference function, such that $\hat{u}_i = u_i$. Sometimes the term used is *incentive compatibility in dominant strategies*, to distinguish from the case in which the agents are truthful only in equilibrium (called *Nash incentive compatibility.*)

Of course, it may not be possible to find a truthful solution for every mechanism design problem. The following powerful result, however, assures us that under many conditions we will be able to find one.

Theorem 7.2.1 (Revelation Principle) *Given a mechanism design problem M, if there exists a dominant-strategy (resp., Nash) solution to M then there exists a solution to M that is direct mechanism that is incentive compatible in dominant strategies (resp., Nash incentive compatible).*

In other words, any solution to a mechanism design problem can be converted into one in which agents always reveal their true preferences. The proof is by construction, and can be explained informally. The new mechanism accepts the agents' truthful utility functions, and "lies for them." That is to say, its mapping to outcomes mimics the mapping that would occur had the old mapping been in place and the agents would bid their dominant strategies (or their Nash equilibria strategies). This is arguably the most powerful result in mechanism design. It means that, while one might have thought a priori that a particular mechanism design problem calls for an arbitrarily complex strategy space, in

[3]The action chosen is sometimes called the *revealed type*, to be contrasted with the agent's *true type*.

[4]Some authors also use the term *strategy proof*.

fact one can restrict one's attention to direct mechanisms – and even incentive compatible ones.

7.2.2 A Negative Result: The Gibbard–Satterthwaite Theorem

We use the term *dictatorial* to describe mechanisms that always adopt the preferred outcome of one particular agent, regardless of the preference vector. While such mechanisms seem undesirable, the following result states that under certain conditions truthful mechanisms are necessarily dictatorial.

Theorem 7.2.2 (Gibbard-Satterthwaite Impossibility Theorem) *Given a mechanism design problem $M = (N, O, U, C)$ such that*

- $|O| \geq 3$*, and*
- *C is an onto function; that is, for all outcomes $o \in O$ there exists an agent preference profile $u \in U$ such that $C(u) = \{o\}$,*

then if a dominant-strategy solution to M exists then the solution (and hence C) are dictatorial; that is, there exists $i \in N$ such that for all $u \in U$ it is the case that $C(u) = \arg\max_{o \in O} u_i(o)$.

7.2.3 A Positive Result: The Vickrey-Clarke-Groves Mechanism

If we are to design a truthful mechanism that is not dictatorial, we are going to have to relax some of the conditions of the Gibbard-Satterthwaite theorem. The obvious candidate for relaxation is the the final condition, that the mechanism be onto. And indeed when we do that we are still left with a vast class of mechanisms, including a very general one called the *Vickrey-Clarke-Groves mechanism,* or VCG for short. We begin by defining a class of mechanism design problems called the *quasi-linear problems.*

Definition 7.2.4 (Quasi-Linear Problem) *A* quasi-linear mechanism design problem *is a mechanism design problem (N, O, U, C) with the following structure:*

- *$O = X \times \Re^n$, where X is some finite set.*
- *For each agent $i \in N$ and each $u_i \in U_i$ there exists a function $v_i : X \to \Re$ such that the utility for each agent i is* quasi-linear*: $u_i(x, r_1, \ldots, r_n) = v_i(x) + r_i$. For slight abuse of notation, we expand the domain of v_i to include outcomes as follows: $v_i((x, r_1, \ldots, r_n)) = v_i(x)$.*
- *The goal of the designer is to maximize the so-called* social welfare:

$$C(u) = \arg\max_{o \in O} \sum_{i \in N} u_i(o)$$

Intuitively, X represents a set of non-monetary outcomes (for example, the allocation of an object to one of the bidders in an auction), and r_i is the (possibly negative) payment received by agent i (for example, a payment to the auctioneer). The quasi-linearity assumption means that the agent's overall utility can is the sum of his value for the non-monetary outcome (for example, his value for the auction item) and his payment. Maximizing social welfare means maximizing the sum of the total utilities of the agents; notice that, under the assumption of quasi-linearity, payments among agents don't impact the social welfare.

Technically speaking, the quasi-linear problem fixes the set of agents. However we generally consider families of quasi-linear problems, for any set of agents. For example, consider a voting game of the sort discussed earlier. You would want to be able to speaking a voting problem and a voting solution in a way that is not dependent on the number of agents. So in the following we assume that a quasi-linear problem is still defined when any one agent is taken away. In this case the set of non-monetary outcomes must be updated (for example, in an auction setting the missing agent cannot be the winner), and is denoted by O_{-i}. Similarly, the utility functions u_i and the choice function C must be updated accordingly.

Definition 7.2.5 (VCG Mechanism) *Given a quasi-linear mechanism design problem $M = (N, O, U, C)$ and functions v_i such that for each agent $i \in N$ and outcome $o \in O$ $u_i(o) = v_i x + r_i$, the VCG mechanism for M is (A, μ) such that $A = U$, and $\mu(\hat{u}) = (x, p_1, \ldots, p_n)$, where*

- $x = \arg\max_{x \in X} \sum_{i \in N} \hat{v}_i(x)$, *and*
- $p_i = \sum_{j \neq i} \hat{v}_j(x) - \max_{o \in O_{-i}} \sum_{j \neq i} \hat{v}_j(o)$

In other words, VCG is a direct mechanism in which agents can bid any valuation function $\hat{v}$ (and thus any utility function $\hat{u} \in U$, given the quasi-linear structure). The center then optimizes the choice of outcome assuming that the agents disclosed their true utility function, and charges agent i his "social cost": the difference between the declared social welfare of the remaining agents in the current situation, and their declared welfare in the hypothetical situation in which agent i did not exist.

The remarkable property of the VCG mechanism is that it is a dominant-strategy solution to the quasi-linear problem:

Theorem 7.2.3 *Given a quasi-linear mechanism design problem M, the VCG mechanism for M is a direct dominant-strategy solution to M.*

In other words, in the VCG mechanism it is a dominant strategy for agents to report their true utility functions. Mysterious as this may sound, it is an immediate consequence of the definitions. Recall that the VCG mechanism chooses the x which maximizes the quantity

$$\sum_{i \in N} \hat{v}_i(x) = \hat{v}_i(x) + \sum_{j \neq i} \hat{v}_j(x)$$

where $\hat{v}_i$ is the declared utility function of agent i. At the same time, the true utility function of agent i is

$$v_i(x) + \sum_{j \neq i} \hat{v}_j(x) - \max_{o \in O_{-i}} \sum_{j \neq i} \hat{v}_j(o)$$

Agent i has no control over the third term $\max_{o \in O_{-i}} \sum_{j \neq i} \hat{v}_j(o)$, and can only influence the remaining sum $v_i(x) + \sum_{j \neq i} \hat{v}_j(x)$. But this is identical to the term maximized by the mechanism, other than the use of the function $\hat{v}_i$ by the mechanism and v_i by the agent; thus the agent can only lose by selecting $\hat{v}_i \neq v_i$.

7.2.4 A Negative Result: The Myerson–Satterthwaite Theorem

The VCG mechanism seems almost too good to be true; where's the catch? We will discuss some computational catches in the section in which we discuss combinatorial auctions, but here is one economic shortcoming. Recall that a mechanism is *incentive compatible* (in dominant strategies) if it leads to a game in which it is a dominant strategy for each agent to disclose his true utility, and that a mechanism is (economically) *efficient* if it always maximized the sum of utilities for all agents. In addition, a mechanism for a quasi-linear problem is said to be *budget balanced* if the sum of payments to the agents is always exactly zero (thus in a budget-balanced auction the auctioneer neither makes nor loses money). The following theorem shows that these three conditions cannot be achieved simultaneously.

Theorem 7.2.4 (The Myerson-Satterthwaite Impossibility Theorem) *No mechanism is simultaneously incentive compatible, efficient, and budget balanced.*

In particular, it follows that the VCG mechanism cannot be all three; indeed, is is incentive compatible and efficient, but not budget balanced.

7.3 A Key Application: Auctions

Auctions constitute an interesting and well known application of mechanism design. In the most familiar types of auction there is one good for sale, one seller, and multiple potential buyers. Each buyer has his own valuation for the good, and each wishes to purchase it at the lowest possible price. Our task is to design a protocol for this auction that satisfies certain desirable global criteria. For example, we might want an auction protocol that maximizes the expected revenue of the seller. Or, we might want an auction that is economically efficient, that is, one that guarantees that the potential buyer with the highest valuation ends up with the good.

The auction setting is important because auctions are widely used in consumer, corporate, and computer science settings. Millions of people use auctions daily on Internet consumer websites to trade goods. More complex types of auctions have been used by governments around the world to sell important public resources such as access to electromagnetic spectrum. Indeed, all financial markets constitute a type of auction (one of the family of so-called *double auctions*). Auctions are also often used in computer science applications to efficiently allocate bandwidth and processing power to applications and users.

In the remainder of this section we first survey the space of auction types, go on to discuss how auctions are modelled as (Bayesian) games, and then present some of the central results of auction theory.

7.3.1 Auction Types

It is important to realize that the most familiar type of auction – the ascending-bid, English auction – is a drop in the ocean of auction types. Indeed, in a precise sense, there is an infinite number of auction types. To give a feel for this broad space, we start with a taxonomical survey of several auction families, and conclude with a broader discussion of the space of auction

The taxonomy of auction types we discuss is depicted graphically in Figure 7.3.1.

```
auction
    one sided
        single dimensional
            sealed bid
                1st-price, 2nd-price, etc
            open outcry
                English
                Dutch
                Japanese
        multi-dimensional
            multi-attribute
            multi-good
                combinatorial
                composite
    two sided ("double auction")
        continuous double auction (CDA)
        periodic double auction (call market)
```

Figure 7.1: A partial auction taxonomy.

Let us say a few words about this taxonomy; we will concentrate primarily on one-sided auctions[5].

[5]It must be emphasized again that this taxonomy is not exhaustive; in particular, there

Single-sided, single-dimensional auctions

In a single dimensional setting there is only one type of good for sale. There could be only one copy of the item, in which the auction is called *single unit*, or multiple interchangeable items, in which case the auction is called *multi-unit*. In both cases, in a single-dimensional auction bidders care only about the number of goods they receive and the price they pay, and their bids can mention only price and (in the case of multi-unit auctions) quantity.

The best known one-sided, single-dimensional auction families are the English auction and the sealed-bid auction, followed closely by the Dutch and Japanese families. Let us briefly review each of them.

The English auction is perhaps the best-known family of auctions, since in form or another they are used in the venerable old-guard auction houses as well as most of the online consumer auction sites. In a single-unit English auction, the auctioneer sets a starting price for the good, and agents then have the option to announce successive bids, each of which must be higher than the previous bid (usually by some minimum increment set by the auctioneer). The rules for when the auction closes vary; in some instances the auction ends at a fixed time, in others it ends after a fixed period during which no new bids are made, in others at the latest of the two, and in still other instances at the earliest of the two. The final bidder, who by definition is the agent with the highest bid, must purchase the good for the amount of his final bid.

Multi-unit English auctions are less straightforward. For one thing, they vary in the payment rules. If there are 3 items for sale, the top 3 bids win one item each. In general, these bids will be for different amounts; the question is what each bidder should pay. In the *pay-your-bid* scheme (the so-called *discriminatory* pricing rule) each of the three top bidders pays a different amount, namely his own bid. In the *uniform* pricing rule all winners pay the same amount; this is usually set to be lowest among the winning bids (though it can be others; for example, the highest among the losing bids).[6]

The extension of the English auction to the multi-unit case is mostly straightforward; a bid for five units at $10/unit is interpreted as five different bids. One subtlety that arises regards minimum increments. Consider the following example, in which there is a total of 10 units available, and two bids: one for 5 units at $1/unit, and one for 5 units at $4/unit. What is the lowest acceptable next bid? Intuitively, it depends on the quantity – a bid for 3 units at $2/unit can be satisfied, but a bid for 7 units at $2/unit cannot. Of course, the latter bid can be *partially* satisfied – is that allowed, or is the bid for 7 units all-or-nothing? This must be specified, but note that all-or-nothing bids give rise to subtle tie-breaking problems. For example, imagine that at the end of the previous auction the highest bids are as follows, all of them all-or-nothing: 5 units

---

certainly exist two-sided combinatorial auctions, as well as auctions that fall outside this taxonomy.

[6] Confusingly, the English auction in conjunction with the uniform pricing rule is sometimes called *Dutch auction*. This is a practice to be discouraged; the correct use of the term is in connection with the descending outcry auction, discussed below.

for $20/unit, 3 units for $15/unit, 5 units for $15/unit, and 1 unit for $15/unit. Presumably the first bid is satisfied, as well as two of the remaining three – but which? Here one sees different tie-breaking rules – by quantity (larger bids win over smaller ones), by time (earlier bids win over later bids), and combinations thereof.

The *Japanese auction*[7] is similar to the English auction in that it is ascending-bid auction, but different otherwise. Here the auctioneer sets a starting price for the good, and each agent must choose whether or not to be "in", that is, whether they are willing to purchase the good at that price. The auctioneer then calls out successively increasing prices in a regular fashion[8], and after each call each agents must announce whether they are still in. When they drop out it is irrevocable, and they cannot re-enter the auction. The auction ends when there is exactly one agent left in; the agent must then purchase the good for the current price.

The extension of the Japanese auction to the multi-unit case is again mostly straightforward. Now after each price increase each agent calls out a number rather than the simple in/out declaration, signifying the number of units he is willing to buy at the current price. A common restriction is that the number decrease in time; the agent cannot ask to buy more at a high price than he did at a lower price. The auction is over when the supply equals or exceeds the demand. If, as is typical in practice, the supply strictly exceeds the demand, one encounters the same pricing options as in the English auction, as well as the subtleties regarding tie-breaking.

In a *Dutch auction*[9] the auctioneer begins by announcing a high price, and then proceeds to announce successively lower prices in a regular fashion. The auction ends when the first agent signals the auctioneer; the signaling agent must then purchase the good for that price. Again, extension to the multi-unit case is mostly straightforward, with some twists. Here agent signal the quantity they wish to buy. If that is not the entire available quantity the auction continues. Here there are several options – the price can continue to descend from the current level, can be reset to a set percentage above the current price, or can be reset to the original high price.

All the auctions discussed so far are considered *open outcry* auctions, in that in all the bidding is done by calling out the bids in public (however, as we'll discuss shortly), in the case of the Dutch auction this is something of an optical illusion). The family of *sealed bid auctions* is different. In a single-unit sealed-bid auction each agent submits to the auctioneer a secret, "sealed" bid for the good which is not accessible to any of the other agents. The agent with the highest bid must purchase the good, but the price at which she does so depends on the type of sealed bid auction. In a first-price sealed bid auction (or simply *first-price auction*) the winning agent pays an amount equal to her own bid. In

[7]Unlike the terms *English* and *Dutch*, the term *Japanese* is not used universally; however, it is commonly used, and there is no competing name for this family of auctions.

[8]In the theoretical analyses of this auction the assumption is usually that they prices rise continuously.

[9]So called because it is the auction method used in the Amsterdam flower market.

a *second-price auction* she pays an amount equal to the next highest bid (that is, the highest rejected bid). The second-price auction is also called the *Vickrey auction*. In general, in a *kth-price auction* the winning agent purchases the good for a price equal to the kth highest bid.[10]

Sealed-bid auctions can be extended to apply to the multi-unit case. In this case, when there are m units for sale, one sometime speaks of mth-price auction and $m+1$-price auction, which play the roles analogous to first- and second-price auctions in the single-unit case. Here too there are issues of tie breaking, which are dealt with similarly to the auctions discussed above.

Two-sided, single-dimensional auctions

In *two-sided auctions*, otherwise known as *double auctions*, there are many buyers and sellers. A typical example is the stock market, where there are many buyers and sellers of any given stock. It is important to distinguish this setting from certain marketplaces (such as popular consumer auction sites) in which there are multiple separate single-sided auctions.

We will not have much to say about double auctions, in part because the relative dearth of theoretical results about them. However, let us mention two primary models of single-dimensional double markets, that is, markets in which there are many potential buyers and sellers of many units of the same good (for example, the shares of a given company). We distinguish here between two kinds of markets, the *continuous double auction* (or CDA) and the *periodic double auction* (otherwise known as the *call market*).

In both the CDA and the call market agents bid at their own pace and as many times as they want. Each bid consists of a price and quantity, where the quantity is either positive (signifying a 'buy' order) or negative (signifying a 'sell' order). There are no constraints on what the price or quantity might be. Also in both cases, the bids received are put in a central repository, the *order book*. Where the CDA and call market diverge is on when a trade is decided on. In the CDA, as soon as the bid is received, at attempt is made to match it with one or more more bids on the order book; for example, a new sell order for 10 units may be matched with one existing buy bid for 4 units and another buy bid for 6 units, so long as both the buy-bid prices are higher than the sell price. In cases of partial matches, the remaining units (either of the new bid or of one of order-book bids) is put back on the order book. For example, if the new sell order is for 13 units and the only buy bids on the order book with a higher price are the ones described (one buy bid for 4 units and another buy bid for 6 units), two trades are arranged – one for 4 units, and one for 6 – and the remaining 3 units of the new bid are put on the order book as a sell order. (We have not mentioned the price of the trades arranged; obviously they must

[10] The reader who has no previous acquaintance with these auction types may be puzzled about the merit of kth-price auction for any $k > 1$. We return to this shortly, but remind the reader that the VCG mechanism employs a rule similar to second-price auction; indeed, the VCG is a generalization of the second-price auction, and for this reason is often called the *Generalized Vickrey Auction*, or GVA for short, in the context of auctions.

lay in the interval between the price in the buy bid and the price in the sell bid – the so called bid-ask spread – but are unconstrained otherwise, and indeed could be lower for the seller than for the buyer, allowing a commission for the exchange or broker.)

In contrast, when a bid arrives in the call market, it is simply placed in the order book. No trade is attempted. Then, at some predetermined time, an attempt is made to arrange maximal amount of trade possible. This is done simply by ranking the sell bids in ascending order, the buy bids in descending order, and finding the point at which supply meets demand. Figure 7.3.1 depicts

before	Sell:	5@$1	3@$2	6@$4	2@$6	4@$9	
	Buy:	6@$9	4@$5	6@$4	3@$3	5@$2	2@$1
				↑			
after	Sell:	2@$6	4@$9				
	Buy:	2@$4	3@$3	5@$2	2@$1		

Figure 7.2: A call-market order book, before and after market clears.

a typical call market. In this example 14 units are traded when the market clears, after which the order book is left with the follow bids awaiting the next market clear.

Multi-dimensional auctions

Multi-dimensional auctions are ones in which each bid mentions more that only the price and quantity of one good. Single-dimensional auctions are used almost universally in consumer auction, primarily because of their relative simplicity. However, multi-dimensional auctions play a critical role in commercial settings: in governmental auctions for the electromagnetic spectrum, in energy auctions, and in corporate procurement auctions.

One can break down multi-dimensional auctions into two families: multi-attribute and multi-good. In multi-attribute auctions, each good has multiple features. For example, each good might be a car with a particular engine size, color, five different options. A potential buyer might have different values for the car, depending which features it has. In most cases, the multi-attribute problem is reduced to the single-dimensional case; each agent has a scoring function for the car as a function of its features, which determines his value for it.

Much more complex is the issue of multi-good auctions. In these auctions there are multiple goods for sale, and somehow the auction process ties them together. The reason to tie them together in the first place is that bidders might have *non-additive* utility functions. For example, the value of a bidder for the pair (TV, DVD player) may be different for the sum of his values for each item alone (in this case the items are *complementary*, and thus presumably the utility function would be *super additive*). The bidder would hate to bid on the DVD player and win it, only to find out that he got outbid on the TV and cannot

display the DVD movies. Conversely, a bidder might be willing to pay $100 for one TV and $90 for another, but still only $100 for the pair (in this cases they are *substitutes*, and the utility function is presumably *sub-additive*).

There are in principle two ways to tie the goods together in an auction. One way is to run essentially separate auctions for the different goods, but to connect them at in certain ways. For example, one way is to have a multi-round (e.g., Japanese) auction, but to synchronize the rounds in the different auctions so that as one bids in one auction one has a reasonably good indication of what is transpiring in the other auctions of interest. Another way to tie auctions together is to institute certain constraints on bidding that span all the auctions (so-called *activity rules*). One example is a budget constraint; a bidder may not exceeds a certain total commitment across all auctions. Both these ideas can be seen in some government auctions for electromagnetic spectrum (where the so-called *simultaneous ascending auction* was used) as well as in some energy auctions.

Perhaps the most straightforward way to tie goods together is to allow bidders to bid on combinations of goods. For example, to allow a bidder to offer $100 for the pair (TV, DVD player), or a disjunctive offer "either $100 for TV1 or $90 for TV2." This is precisely the nature of *combinatorial auctions*. This important class of auctions has received much attention in both economics and computer science, and thus we devote Section 7.4 to it later in the chapter.

Beyond taxonomy

While it is useful to have reviewed the best known auction types, we have emphasized all along that the taxonomy presented is not exhaustive. Many other auctions have been proposed and tried, even single-dimensional ones. For example, consider the following auction, consisting of a series of sealed bids. In the first round the lowest bidder drops out; his bid is announced, and becomes the minimum bid in the next round for the remaining bidders. The process continues until only one bidder remains, who is the winner at that final price. This auction, called the *elimination* auction, is different from any of the above, and yet makes perfect sense. Or consider a procurement reverse auction, in which an initial sealed-bid is conducted among the interested suppliers, and then a reverse English auction is conducted among the three cheapest suppliers (the "finalists") to determine the ultimate supplier. This two-phased auction, which actually is not uncommon in industry, is again not on the standard menu.

Indeed, the taxonomical perspective obscures the elements common to all auctions, and thus the infinite nature of the space. What is an auction? At heart it is simply a structured framework for negotiation. Each such negotiation has certain rules, which can be broken down into three categories:

1. Bidding rules: How are offers made (by whom, when, what can their content be).

2. Clearing rules: When are trades decided on, or what are those trades (who gets which goods, and what money changes hands) as a function of the

bidding.

3. Information rules: Who knows what and when about the state of negotiation.

The different auctions discussed make different choices in this regard, but it is clear that other rules can be instituted. Indeed, when viewed this way, it becomes clear that what seem like three radically different commerce mechanisms – namely the hushed purchase of a Matisse at a high-end auction house in London, the mundane purchase of groceries at the local supermarket, and the one-on-one horse trading in a Middle Eastern *souk* – simply make different choices along these three dimensions.

7.3.2 Elements of Auction Theory

When analyzing different auction mechanisms, one tries to answer basic questions such as whether the auction will maximize the revenue to the seller, as compared to any other auction that might be used. Or alternatively, one might ask if the auction is (economically) efficient, in that it maximizes the social welfare.

Given the popularity of auctions on the one hand, and the diversity of auction mechanisms on the other, it is not surprising that the literature on the topic is vast. In this section we provide a taste for this literature, concentrating on single-dimensional, one-sided, single-unit auctions. We begin with some simple observations, and then provide enough of a formal model of auctions as Bayesian mechanisms to be able to present some formal results.

Initial observations

The first observation is that the Dutch auction and the first-price sealed bid auction, while quite different in appearance, are actually the same auction (in the technical jargon, they are *strategically equivalent*). In both auctions each agent must select an amount without knowing about the other agents' selections; the agent with the highest amount price wins the auction, and must purchase the good for that amount.

A similar relationship exists between the Japanese auction and the second-price sealed bid auction. In both cases the bidder must select a number (in the sealed bid case the number is the one written down, and in the Japanese case it is the price at which the agent will drop out); the bidder with highest amount wins, and pays the amount selected by the second-highest bidder. However the connection is not as tight as the relationship between the Dutch and first price auctions, since here the information disclosure is different. In the sealed bid auction the amount is selected without knowing anything about the amounts selected by others, whereas in the Japanese auction the amount can be updated based on the prices observed at which lower bidders dropped out. This matters in certain cases, in particular the cases of *common value* discussed below.

Obviously, the Japanese and English auctions are also closely related. The main difference is that in the English auction successive bids can be so-called *jump bids*, or bids that are greater than the previous high bid by more than the minimum increment. Although it seems relatively innocuous, this feature complicates analysis of such auctions, and indeed when an ascending auction is analyzed it is almost always the Japanese one, not the English.

Auctions as Bayesian mechanisms

In order to analyze auctions beyond these basic observations we need to be more formal. First note that an auction setting defines a (Bayesian) mechanism-design problem (N, O, U, C). The possible outcomes O consist of all possible ways to allocate the good and to charge the bidders. The choice function C depends on the objective of the auction. If it is to maximize efficiency, it is defined in a straightforward way. If it is to maximize revenue, we must add the auctioneer as one of the agents, with no choice of strategy but with a decided preference over the various outcomes (namely, preferring the outcomes in which the total payments to the auctioneer are maximal), a preference that defines the C function.

However, each Bayesian problem includes two more ingredients that we need to specify – the common prior, and the private signals of the agents. Here we distinguish between two settings, called the *independent private value* (IPV) setting and the *common value* (CV) setting. In the IPV setting all agents' valuations are drawn independently from the same (commonly known) distribution, and the signal of the agent consists only of his own valuation (and thus gives him no information about the valuation of the others). An example where the IPV setting is appropriate is in auctions consisting of bidders with personal tastes who aim to buy a piece of art purely for their own enjoyment. In contrast, in the CV setting all agents have an identical value which is drawn from some distribution, but the agents get different signals about the value. An example where the CV setting is appropriate is in auctions for oil drilling rights. In these auctions there is a certain amount of oil to be found, the cost of extraction will be about the same no matter who wins the contract, and the price of oil will be what it will be. The only difference is that the different companies have different geologists and financial analysts, and thus different assessments for these quantities.[11]

The difference between the IPV and CV setting is substantial. Consider, for example, the question of whether the second-price sealed-bid auction, which is a direct mechanism, is incentive compatible (that is, does it provide incentive the agents to bid their true value). It is not hard to see that in the CV case it does. Indeed, the second-price auction is a special case of the VCG mechanism discussed earlier, but in this special case the proof is even more immediate; here it is immediate to see that the bidder's bid amount determines whether he wins, but has no impact on his payment. Clearly the bidder would want to win at

[11] There is also an intermediate setting called *affiliated values*, but we do not discuss it here.

any amount up to his true valuation, and will only lose by bidding either higher or lower. But this analysis depends crucially on the assumption that the bidder knows his precise valuation, which is true in the IPV setting but not in the CV setting.

It is interesting to contrast this with the analysis of the first-price auction in the CV setting. Here we do not have the luxury of having dominant strategies, and must resort to (Bayesian) equilibrium analysis. We will consider the two-player case, in which the bidders' valuations are drawn uniformly from some interval, say [0..10], and the bidders are risk neutral.[12]

In what follows we use s_i to refer to the bid of player i, and v_i to refer to the true valuation of player i. Thus if player i wins, his payoff is $u_i = v_i - s_i$; if he loses, it is $u_i = 0$. Now we prove that there is an equilibrium in which each player bids half of their true valuation (it also happens to be the unique symmetric equilibrium, but we do not discuss that here). In other words, we prove that $(\frac{1}{2}v_1, \frac{1}{2}v_2)$ is an equilibrium strategy profile. We begin by calculating the expected payoff of player 1, assuming that player 2 is bidding $\frac{1}{2}v_2$. Since player 1 believes that all possible valuations to player 2 are equally likely, we do this by integrating over all possible valuations of player 2.

$$\mathcal{E}(u_1) = \int_0^{10} u_1 dv_2$$

Note that this integral can be broken up into two smaller integrals that differ on whether or not player 1 wins the auction. Because player 2 is bidding half of her true valuation, player 1 wins when player 2's valuation is less than twice his own bid, s_1, and he loses otherwise. Then player 1's utility is simply $(v_1 - s_1)$ when he wins, and 0 otherwise.

$$\begin{aligned}
\mathcal{E}(u_1) &= \int_0^{2s_1} u_1 dv_2 + \int_{2s_1}^{10} u_1 dv_2 \\
&= \int_0^{2s_1} (v_1 - s_1) dv_2 + \int_{2s_1}^{10} 0 dv_2 \\
&= \int_0^{2s_1} (v_1 - s_1) dv_2 \\
&= (v_1 - s_1) v_1 |_0^{2s_1} \\
&= 2v_1 s_1 - 2s_1^2
\end{aligned}$$

Now we have a closed form function which represents the expected payoff of player 1, in terms of his valuation and bid. We would like to find the bid value which maximizes this expected payoff. We find the maximum by finding the point where the derivative with respect to s_1 is zero, and then solving for s_1 in

[12] Risk neutral agents are indifferent between a certain event with a particular payoff and a lottery among events with the same expected outcome. In contrast, risk-averse agents have a higher utility to the former, and risk-seeking to the latter.

terms of v_1.

$$\begin{aligned} \frac{\partial}{\partial s_1}(2v_1 s_1 - 2s_1^2) &= 0 \\ 2v_1 - 4s_1 &= 0 \\ s_1 &= \frac{1}{2}v_1 \end{aligned}$$

Thus when player 2 is bidding half her valuation, player 1's best strategy is to bid half his valuation. The calculation of the optimal bid for player 2 is analogous, given the symmetry of the game and the equilibrium. We have proven that $(\frac{1}{2}v_1, \frac{1}{2}v_2)$ is an equilibrium strategy profile of this game.

More generally, we have the following theorem.

Theorem 7.3.1 *In a first-price sealed bid auction with n risk-neutral agents whose valuations are independent and identically distributed over a finite interval, the unique symmetric equilibrium is given by the strategy profile* $(\frac{n-1}{n}v_1, \ldots, \frac{n-1}{n}v_n)$.

In other words, the unique equilibrium of the auction occurs when each player bids $\frac{n-1}{n}$ of their valuation. Thus the first-price sealed-bid auction protocol is not incentive compatible.

Revenue maximization

The final topic that we discuss in connection with auction theory is arguably what auctioneers care most about: revenue maximization. If you have an item to sell and wish to get top dollar, which of the many auction types should you use?

The most prominent result here is the following theorem.

Theorem 7.3.2 (Revenue Equivalence Theorem) *Given an IPV setting with risk-neutral bidders*[13]*, if an auction has the following two properties:*

- *The auction is efficient, that is, it always awards the good to the bidder with the highest valuation, and*
- *The bidder with the lowest valuation never has to pay anything*

then the auction maximizes the seller's expected revenue.

Thus under the specified conditions, all the auctions we have spoken about so far – English, Japanese, Dutch, and all sealed bid auction protocols – are revenue equivalent, and optimal.

The primary difference between the IPV and the common value (CV) environments is that in the CV environment, the English and first-price sealed bid auction protocols are no longer revenue equivalent. One way to understand this is to note that agents in sealed bid auctions are susceptible to the so-called

[13] And certain conditions on the distribution of valuations, which are not discussed here.

winner's curse – by definition, the agent who has overestimated the value the most is the winner. In such an environment the English auction protocol can be expected to give higher revenue than the first-price sealed bid auction protocol, because in an English by seeing other agents' bids the bidder is somewhat immune from this curse. However, the Dutch auction and the first-price sealed bid auction are still revenue equivalent, because in neither protocol do the buyers receive information about the valuations of other buyers.

Because these findings can be confusing, they are summarized in table 7.1.

IPV	Risk-neutral	Jap	=	Eng	=	2nd	=	1st	=	Dutch
	Risk-averse		=		=		<		=	
	Risk-seeking		=		=		>		=	
CV	Risk-neutral		=		>		>		=	

Table 7.1: Relationships between revenues of various auction protocols.

7.4 Combinatorial auctions

As mentioned briefly above, *combinatorial auctions* are auctions in which multiple goods are being auctioned simultaneously. In a combinatorial auction, bidders are allowed to place bids on arbitrary combinations, or *bundles* of these goods. For example, imagine that you visit a popular consumer auction website, and find a wide variety of household goods for sale. You might like to submit a bid of the following form: "I bid $100 for the TV, VCR, and couch." Of course, your bid may be more complex , such as: "I bid $100 for the TV and VCR, or $150 dollars for the TV and DVD player, but not both."

Let's begin by giving a precise formulation of a combinatorial auction problem. A combinatorial auction problem is a tuple $(N, X, v_1, \ldots, v_n)$, where N is a set of n agents, X is a set of m goods, and for each agent $i \in N$, $v_i : 2^X \rightarrow \Re$ is a valuation function. Most commonly, the combinatorial auction problem is to select an allocation $a : 2^X \rightarrow N$ of goods to agents that maximizes some measure such as total revenue to the auctioneer, or efficiency.

Combinatorial auctions pose a number of interesting computational problems. In the consumer auction example above, there are number of questions you might ask. First, as a bidder you might want to know *what you can bid*; in other words, what kinds of bids are you permitted to submit. While this is trivial in single-unit auctions, in a combinatorial auction a bid may consist of an arbitrary valuation of every possible subset of goods. When there are m goods, there are 2^m such subsets, and thus the size of bids can easily be exponential in the number of goods. We will discuss possible *bidding languages* in section 7.4.1 below.

Second, as in single-dimensional auctions, you might want to know *what you should bid*. What strategy is most likely to maximize your welfare? If the combinatorial auction mechanism is *incentive compatible*, you will want to

submit your true valuation for the good as your bid. As we will see in section 7.4.2, we can use a generalized form of the Vickrey auction to as an incentive compatible mechanism.

Finally, if you are the auctioneer in this auction, you might want to know *how you should allocate the goods* after you have collected all of the bids. Although this is straightforward in single-unit auctions, in combinatorial auctions it is not at all trivial. In section 7.4.3 we will see that the problem is in general very difficult.

7.4.1 Expressing a Bid: Bidding Languages

Before we can consider the computation of an allocation, we must find a way for the bidders in the auction to express their bids. In a combinatorial auction, a bid may consist of an arbitrary valuation on every possible subset of the goods. Since there are an exponential number of such subsets, the length of a particular bid may in general be exponential. If we are to have any hope of finding tractable mechanisms for general combinatorial auctions, we must first find a way for bidders to express their bids in a more succinct manner. In this section we will present a number of bidding languages that have been proposed for encoding bids.

As we will see, these languages differ in the ways that they express different classes of bids. We can state some desirable properties that we might like to have in a bidding languages. First, we want our language to be *expressive* enough to represent all possible valuation functions. Second, we want our language to be *concise*, so that expressing commonly used bids doesn't take space that is exponential in the number of goods. Third, we want our language to be *natural* for humans to both understand and create; thus the structure of the bids should reflect the way in which we think about them naturally. Finally, we want our language to be *tractable* for the auctioneer algorithms to process when computing an allocation.

In the discussion that follows, for convenience we will often speak about bids as valuation functions. Indeed, in the most general case a bid will contain a valuation for every possible combination of goods. However, be aware that the bid valuations may or may not reflect the players' true underlying valuations. We also limit the scope of our discussion to valuation functions in which the following properties hold.

- **No externalities.** The bidder's valuation depends only on the set of goods he wins, so that the valuation function is $v_i : 2^X \rightarrow \Re$ where the domain is just the set of goods that she wins.
- **Free disposal.** Goods have non-negative value, so that if $S \subseteq T$ then $v_i(S) \leq v_i(T)$.
- **Nothing-for-nothing.** In other words, $v_i(\emptyset) = 0$.

There are two important properties that a valuation function may or may not satisfy. They concern the way in which the valuation of one good may be affected

by the presence or absence of another good. The first is *complementarity*. A valuation function v has complementarities if there exist two sets of goods $S, T \subseteq X$, for which $v(S \cup T) > v(S) + v(T)$. The second property is *substitutability*. A valuation function v has substitutability if there exist two sets of goods $S, T \subseteq X$, such that $S \cap T = \emptyset$, for which $v(S \cup T) < v(S) + v(T)$.

Before we define specific bidding language, let us consider some types of bids that we may commonly want to express. We can divide these bids into *symmetric* and *asymmetric* valuations. Symmetric valuations are those in which all goods are identical from the point of view of the bidder, and for this reason we sometimes use the term *multiple units of good*. A few common symmetric valuations are the following.

- **Additive valuation.** The bidder's valuation of a set is directly proportional to the number of goods in the set, so that $v_i(S) = c|S|$ for some constant c.
- **Single item valuation.** The bidder desires any single item, and only a single item, so that $v_i(S) = c$ for some constant c for all $S \neq \emptyset$.
- **Fixed budget valuation.** Similar to the additive valuation, but the bidder has a maximum budget of B, so that $v_i(S) = \min(c|S|, B)$
- **Majority valuation.** The bidder values equally any majority of the goods, so that
$$v_i(S) = \begin{cases} 1 & \text{if } |S| \geq m/2 \\ 0 & \text{otherwise} \end{cases}$$

We can generalize all of these symmetric valuations to a general symmetric valuation.

- **General symmetric valuation.** Let $p_1, p_2, \ldots, p_m$ be arbitrary non-negative prices, so that p_j specifies how much the bidder is willing to pay of the jth item won. Then
$$v_i(S) = \sum_{j=1}^{|S|} p_j$$
- **Downward sloping valuation.** A downward sloping valuation is a symmetric valuation in which $p_1 \geq p_2 \geq \cdots \geq p_m$.

Many common types of bids are not symmetric, however. Often there are different classes of goods, and valuations of sets of goods are a function of the classes of goods in the set. For example, imagine that our set X consists of two classes of goods: some red items and some green items, and the bidder requires only items of the same color. Alternatively, it could be the case that the bidder wants exactly one item from each class.

Now that we have seen some common bid valuations, let's begin to build up some languages for expressing these bids. Perhaps the most basic thing we

might do is bid on one particular subset of goods. We call such a bid an *atomic bid*. An atomic bid is a pair (S, p) which indicates that the agent is willing to pay a price of p for the subset of goods S. Note that an atomic bid implicitly represents an AND operator between the different goods in the bundle. We stated an atomic bid above when we wanted to bid on the couch, the TV, *and* the VCR for $100.

Of course, many simple bids cannot be expressed as an atomic bid; for example, it is easy to verify that an atomic bid cannot represent even the additive valuation defined above. In order to represent this valuation, we will need to be able to bid on disjunctions of atomic valuations. An *OR bid* is a disjunction of atomic bids (S_1, p_1) OR (S_2, p_2) OR $\cdots$ OR (S_k, p_k) which indicates that the agent is willing to pay a price of p_1 for the subset of goods S_1, or a price of p_2 for the subset of goods S_2, etc.

Note that we have used the OR operator informally in the definition of the OR bid. The OR operator is actually an operator for combining valuation functions, and we can define its semantics more precisely. Let V be the space of possible valuation functions, and $v_1, v_2 \in V$ be arbitrary valuation functions. Then we have that

$$(v_1 \text{ OR } v_2)(S) = \max_{R,T \subseteq S, R \cap T = \emptyset} (v_1(R) + v_2(T)).$$

It is easy to verify that an OR bid can express the additive valuation. As the following result shows, its power is still quite limited however; for example it cannot express the single item valuation described above.

Theorem 7.4.1 *OR bids can express all bids that have no substitutability, and only them.*

For example, in the consumer auction example given above, we wanted to bid on either the TV and the VCR for $100, or the TV and the DVD player for $150, but not both. It is not possible for us to express this using OR bids.

For this reason, we present the *XOR bid*. An XOR bid is an exclusive OR of atomic bids (S_1, p_1) XOR (S_2, p_2) XOR $\cdots$ XOR (S_k, p_k) which indicates that the agent is willing to accept one but no more than one of the atomic bids.

Once again, the XOR operator is actually defined on the space of valuation functions. We can define its semantics precisely as follows. Let V be the space of possible valuation functions, and $v_1, v_2 \in V$ be arbitrary valuation functions. Then we have that

$$(v_1 \text{ XOR } v_2)(S) = \max(v_1(S), v_2(S)).$$

We can use XOR bids to express our example from above:

$$(\{\text{TV}, \text{VCR}\}, 100) \text{ XOR } (\{\text{TV}, \text{DVD}\}, 150)$$

It is easy to see that XOR bids have unlimited representational power, since it is possible to construct a bid for an arbitrary valuation using an XOR of the atomic valuations for every possible subset $S \subseteq X$.

Theorem 7.4.2 *XOR bids can represent all possible valuation functions.*

However, this doesn't imply that XOR bids represent every valuation function efficiently. In fact, as the following result states, there are simple valuations that can be represented by short OR bids but which require XOR bids of exponential size.

Theorem 7.4.3 *Additive valuations can be represented by OR bids in linear space, but require exponential space if] represented by XOR bids.*

Note that for the purposes of the present discussion, we consider the size of a bid to be the number of atomic formulas that it contains. The reader can verify that the additive valuation requires just this.

Now we present bidding languages that result from combining the OR and XOR operators on valuation functions. Consider a language which allows bids that are of the form of an OR of XOR of atomic bids. We call these bids *OR-of-XOR bids*. An *OR-of-XOR bid* is a set of XOR bids, as defined above, such that the bidder is willing to obtain any number of these bids.

Of course, like XOR bids, OR-of-XOR bids have unlimited representational power. However, unlike XOR bids, they can generalize to plain OR bids, which affords greater simplicity of expression, as we have seen above. As a specific example, OR-of-XOR bids can express any downward sloping symmetric valuation on m items in size of only m_2. However, its expressive power is still limited. For example, even simple assymetric valuations require size of at least $2^{m/2+1}$ to express in the OR-of-XOR language.

It is also possible to define a language of XOR-of-OR bids, and even a language allowing arbitrary nesting of OR and XOR statements here (we refer to the latter as generalized OR/XOR bids). These languages vary in their expressivity.

Now we turn to a slightly different sort of bidding language that is powerful enough to simulate all of the preceding languages with a relatively succinct representation. This language results from the insight that it is possible to simulate the effect of an XOR by allowing bids to include *dummy items*. The only difference between an OR and a XOR is that the latter is exclusive; we can enforce this exclusivity in the OR by ensuring that all of the sets in the disjunction share a common item. We call this language *OR**. Given a set of dummy items X_i for each agent $i \in N$, an *OR* bid* is a disjunction of atomic bids (S_1, p_1) OR (S_2, p_2) OR $\cdots$ OR (S_k, p_k), where for each $l = 1, \ldots, k$, the agent is willing to pay a price of p_l for the set of items $S_l \subseteq X \cup X_i$.

Let's give an example to help make this clearer. If we wanted to express our TV bid from above using dummy items, we would create a single dummy item D, and express the bid as follows.

$$(\{\text{TV}, \text{VCR}, \text{D}\}, 100) \text{ OR } (\{\text{TV}, \text{DVD}, \text{D}\}, 150)$$

The following results show us that the OR* language is surprisingly expressive and simple.

Theorem 7.4.4 *Any valuation that can be represented by OR-of-XOR bids of size s can also be represented by OR* bids of size s, using at most s dummy items.*

Theorem 7.4.5 *Any valuation that can be represented by XOR-of-OR bids of size s can also be represented by OR* bids of size s, using at most s^2 dummy items.*

Theorem 7.4.6 *Any valuation that can be represented by OR/XOR bids of size s can also be represented by OR* bids of size s, using at most s^2 dummy items.*

Let us briefly summarize the properties of the languages we have discussed. The XOR, OR-of-XORs, XOR-of-OR, OR/XOR, and OR* languages are all powerful enough to express all valuations. Second, the efficiencies of the OR-of-XOR and XOR-of-OR languages are incomparable: there are bids that can be expressed succinctly in one but not the other, and vice-versa. Third, the OR* language is strictly more expressive than both the OR-of-XOR and XOR-of-OR languages: it can efficiently simulate both languages, and succinctly express some valuations that require exponential size in both of them.

Recall that in the auction setting these languages are used for communicating bids to the auctioneer. It is the auctioneer's job to first interpret these bids, and then calculate an allocation of goods to agents. Thus it is natural to be concerned about the computational complexity of a given bidding language. In particular, we may want to know how difficult it is to take an arbitrary bid in some language and compute the valuation of some arbitrary subset of goods according to that bid. We call this the *interpretation complexity*. The interpretation complexity of a bidding language is the minimum time required to compute the valuation $v(S)$, given input of an arbitrary subset $S \subseteq X$ and arbitrary bid v in the language.

Not surprisingly, the atomic bidding language has interpretation complexity that is polynomial in the size of the bid; to compute the valuation of some arbitrary subset S, just check to see whether every member of S is in the atomic bid; if they are, the valuation of S is just that given in the bid (because of free disposal) and if they are not, then the valuation of S is 0. The XOR bidding language also has interpretation complexity that is polynomial in the size of the bid; just perform the above procedure for each of the atomic bids in turn. However, all of the other bidding languages mentioned above have interpretation complexity that is exponential in the size of the bid. For example, given the OR bid (S_1, p_1) OR (S_2, p_2) OR $\cdots$ OR (S_k, p_k), computing the valuation of S requires checking all possible combinations of the atomic bids, and there are 2^k such possible combinations.

One might ask why we even consider bidding languages that have exponential interpretation complexity. Simply stated, the answer is that the language with only polynomial interpretation complexity are not expressive enough. This brings us to a more relaxed criterion. It may be enough to require that our bid's valuation of a set is *verifiable* in polynomial time. We define the *verification complexity* of a bidding language as the minimum time required to verify the

valuation $v(S)$, given input of an arbitrary subset $S \subseteq X$, an arbitrary bid v in the language, and a proof of the proposed valuation $v(S)$.

Note that the relationship between verification complexity and interpretation complexity is analogous to the relationship between the complexity classes P and NP in theoretical computer science. As it turns out, all of the bidding languages mentioned above are polynomially verifiable.

7.4.2 Achieving Incentive Compatibility: The Generalized Vickrey Auction

In this section we address the problem of making auctions incentive compatible. Recall that a mechanism is incentive compatible if it is a dominant strategy for each player to reveal their true valuation function (or type).

In general, there are a few different measures that auction mechanisms might try to optimize when selecting an allocation.

1. **Revenue maximization.** The allocation selected by the auction protocol maximizes the total revenue to the seller.

2. **Efficiency.** The auction protocol allocates the goods to the bidders who value them the highest.

3. **Incentive compatibility.** The auction protocol gives every bidder incentive to reveal his true valuation functions.

One might imagine that a seller designing an auction really only cares about the first criterion: maximizing the revenue that he will receive. Auction protocols that satisfy this property are sometimes called *optimal*, and optimal auctions are not well understood. Thus, instead we will discuss protocols that satisfy the second property, *efficiency*. We usually achieve this second property using mechanisms that are also *incentive compatible*; if we know the agents' true valuations, then it is straightforward for us to assign the goods to the agents who value them the highest.

Note that in a naive combinatorial auction mechanism there is ample incentive for bidders *not* to reveal their true valuation in their bids. Consider the following simple valuations in a combinatorial auction setting. Here v_i is intended to represent the bidder's true valuation.

$$v_1(x, y) = 100$$

$$v_1(x) = v_1(y) = 0$$

$$v_2(x) = v_2(x) = 75$$

$$v_2(x, y) = 0$$

$$v_3(x) = v_2(x) = 40$$

$$v_3(x, y) = 0$$

Given these valuations, it is clear that players 2 and 3 are better off not revealing their true type. As it stands, the auctioneer will maximize welfare by assigning x and y to 1, and 2, respectively, and this would still be the case if 1 and 2 lowered their valuations for x and y a bit.

However, recall from section 7.2 that the revelation principle assures us that if there is any mechanism which is a solution, then there is a mechanism which is incentive compatible. Thus we get the third desideratum "for free."

Perhaps not surprisingly, the auction which satisfies properties two and three is an instance of the general Vickrey-Clarke-Groves (VCG) mechanism discussed in section 7.2. We will formalize the VCG combinatorial auction in the remainder of this section. Note that it would be difficult to design an auction protocol which maximized efficiency without being incentive compatible: the auctioneer would not have any information about the true valuations of the bidders!

Given an auction problem (N, X, v), the VCG combinatorial auction works as follows. We use notation that is slightly different from that given above. Here, we let $a_{S,i}$ be 1 if the subset $S \in 2^X$ was allocated to agent $i \in N$ and 0 otherwise.

1. Each bidder $i \in N$ reports a bid valuation v_i. (We will see below that this bid valuation is their true valuation, since they have no incentive to misreport it.)

2. The auctioneer chooses an allocation $a = (a_{S,i})_{S \in 2^X, i \in N}$ that solves the following integer program.

$$V = \max \sum_{i \in N} \sum_{S \subseteq X} v_i(S) a_{S,i}$$

$$\text{s.t.} \sum_{S \ni j} \sum_{i \in N} a_{S,i} \leq 1 \ \forall j \in X$$

$$\sum_{S \subseteq X} a_{S,i} \leq 1 \ \forall i \in N$$

$$a_{S,i} = \{0, 1\} \ \forall S \subseteq X, i \in N$$

   Call this optimal allocation a^*.

3. The auctioneer computes for each bidder $k \in N$ the following.

$$V^{-k} = \max \sum_{i \in (N-k)} \sum_{S \subseteq X} v_i(S) a_{S,i}$$

$$\text{s.t.} \sum_{S \ni j} \sum_{i \in (N-k)} a_{S,i} \leq 1 \ \forall j \in X$$

$$\sum_{S \subseteq X} a_{S,i} \leq 1 \ \forall i \in (N-k)$$

$$a_{S,i} = \{0, 1\} \ \forall S \subseteq X, i \in (N-k)$$

4. Finally, the payment that bidder i makes is

$$V^{-i} - [V - \sum_{S \subseteq X} v_i(S) a^*_{S,i}].$$

Notice that the payment made by each bidder is non-negative.

Just as in the general case, each bidder pays the auctioneer the difference between the social welfare of the other agents when he is part of the allocation, and the social welfare of the other agents in the hypothetical case that he is not part of the allocation. Note that V is the maximum social welfare of all the agents, and that V^{-i} is the maximum social welfare of all of the agents except i when i is not considered in the allocation.

Now, let us show that the VCG combinatorial auction is incentive compatible. Note that the true utility function of bidder i is

$$\sum_{S \subseteq X} v_i(S) a^*_{S,i} - V^{-i} + [V - \sum_{S \subseteq X} v_i(S) a^*_{S,i}]$$

$$= V - V^{-i}$$

In the first expression, the first term is the value that bidder i places on the goods that he receives, and the remainder is the payment that he must make to the auctioneer. In the simplified expression, the bidder has no influence over the second term, and can only benefit by trying to maximize the first term. But this is precisely what the auctioneer is trying to maximize. Thus the bidder has incentive to report his true valuation function.

It is more difficult to show that the VCG combinatorial auction maximizes revenue to the auctioneer. However, consider the following informal argument. The total revenue of an auctioneer employing the VCG auction can be calculated as follows.

$$\sum_{i \in N} V^{-i} - \sum_{i \in N} [V - \sum_{S \subseteq X} v_i(S) a^*_{S,i}]$$

$$= \sum_{i \in N} \sum_{S \subseteq X} v_i(S) a^*_{S,i} + \sum_{i \in N} (V^{-i} - V)$$

$$= V + \sum_{i \in N} (V^{-i} - V)$$

Note that if there were a large number of bidders, then no single bidder could have a significant effect. That is, one would expect that on average, V is close in value to V^{-i} for all $i \in N$. Thus, the revenue to the seller would be close to V, which is of course the largest possible revenue that any auction could extract.

The VCG combinatorial auction mechanism has a few shortcomings. The most important of these is shown by the Myerson-Satterthwaite impossibility theorem, which states that no mechanism can be simultaneously incentive compatible, efficient, and budget balanced. In particular, it follows that the VCG

auction cannot be all three; indeed, is is incentive compatible and efficient, but not budget balanced.

The VCG auction is clearly impractical to implement for most applications, since it requires computing the solution to an integer program, a problem known to be NP-complete (and thus requiring time exponential in n and m). Of course it is possible to approximate the optimal solution to the integer program, but this may not preserve incentive compatibility. Another way to make the problem tractable is to restrict the classes of bids that bidders may submit. We will cover these issues in more detail in the next section.

7.4.3 Computing an Allocation

After the valuations have been expressed in some language and communicated to the auctioneer, the problem of computing the allocation still remains. For the purposes of this section, we consider the problem of computing an allocation that is efficient, in that it maximizes the total social welfare. We begin by formalizing this problem as the following *integer program* (IP).

$$\text{maximize} \sum_{i \in N} \sum_{S \subseteq X} v_i(S) a_{S,i}$$

$$\text{s.t.} \sum_{S \ni j} \sum_{i \in N} a_{S,i} \leq 1 \; \forall j \in X$$

$$\sum_{S \subseteq X} a_{S,i} \leq 1 \; \forall i \in N$$

$$a_{S,i} = \{0, 1\} \; \forall S \subseteq X, i \in N$$

The first line states that we want to maximize the sum of the values to the agents of the goods that they are assigned in the allocation. The next two lines give the constraints on this optimization. The first of these ensures that all of the subsets in the allocation are non-overlapping, that we don't allocate any goods more than once. The second ensures that each bidder receives at most one subset of goods. Finally, the last constraint is what makes this an integer program rather than a general *linear program* (LP): no subset can be partially assigned to an agent.

Readers familiar with the theory of computation will recognize that the combinatorial auction allocation problem expressed above is an instance of the more general *set packing problem* (SPP) that has long been studied by theoreticians. In the set packing problem, we are given a set of elements X and a set Y of possible subsets of X, each of which is assigned a weight w_k. We wish to select the set of subsets that maximizes the sum of the weights of the subsets. In the program that follows, let a_k be 1 if the set $k \in Y$ is selected in the allocation, and 0 otherwise. Also, let $b_{j,k}$ be 1 if the element $j \in X$ is in the subset $k \in Y$ Then the problem can be expressed formally as the following integer program.

$$\text{maximize} \sum_{k \in Y} w_k a_k$$

$$\text{s.t.} \sum_{k \in Y} b_{j,k} a_k \leq 1 \ \forall j \in X$$

$$a_k = \{0, 1\} \ \forall k \in Y$$

Let a^* be the optimal allocation. Note that in order to make this a combinatorial auction problem, we need only to divide the weight function into the many valuation functions of the different bidders.

Unfortunately, it is well known that the SPP, and IPs in general, are NP-complete. In other words, they are known to be in a class of problems for which no polynomial time algorithm is known. Thus we believe that in the worst case it may take exponential time to compute this efficient allocation.

If we want to compute an allocation in a combinatorial auction, we must use a strategy other than solving the IP. There are a few different approaches we might take. The first thing we might try is to find a way to approximate the solution. A straightforward way to approximate the solution to the integer program is to relax the integer constraint, thereby transforming the problem into a linear program, which is solvable by known methods in polynomial time. However, note that such a solution may result in "fractional" allocations, in which fractions of bundles of goods are allocated to different bidders. If we are lucky, our solution to the LP will happen to be integral anyway.

Fortunately, this is the case for certain special instances of the auction problem. Mathematically, these are instances in which the extreme points of the polyhedron $P(A) = \{a | \sum_{j \in Y} b_{j,k} a_k \leq 1 \forall j \in X; a_k > 0 \ \forall k \in Y\}$ representing the solution space are composed only of 0 or 1 values. Such a polyhedron is said to be *integral*. As it turns out, it is not trivial to define conditions that are sufficient to ensure an integral polyhedron. In general these conditions comprise restrictions on the kinds of subsets that bidders may bid on. In the following discussion we will present a few special cases that are relevant to combinatorial auctions.

The most common of these is called *total unimodularity* (TU). In general terms, a matrix A is TU if the determinant of every square submatrix is 0, 1, or -1. Since every extreme point of the polyhedron $P(A)$ corresponds to a square submatrix of A, and it is easy to see that the polyhedron of a TU matrix will be integral.

How do we find out if a particular matrix (of possible bids, for instance) is TU? There are many ways. First, there exists a polynomial time algorithm to decide whether an arbitrary matrix is TU. Second, we can characterize important subclasses of TU matrices. One important subclass of TU matrices are the *network matrices*, which are matrices in which each column contains at most two non-zero entries of opposite sign and absolute value 1. It is not clear what class of bids the network matrices correspond to.

Another important subclass of TU matrices is the class of 0-1 matrices with the *consecutive ones property*. In this subclass, all nonzero entries in each column must appear consecutively. One might ask what classes of bids the consecutive ones property corresponds to in auction problems. This corresponds roughly to contiguous single-dimensional goods, such as time intervals or parcels

of land along a shoreline, where bids can only be made on bundles of contiguous goods.

Another subclass of auction problems that have integral polyhedra, and thus can be easily solved using linear programming, corresponds to the set of *balanced matrices*. A 0-1 matrix is balanced if it has no square submatrix of odd order with exactly two 1's in each row and column. One class of auction problems that is known to have a balanced matrix are those which allow only *tree-structured* bids. Consider that the set of goods for sale are the vertices of a tree, connected by some set of edges. All bids must be on bundles of the form (j, r), which represents the set of vertices which are within distance r of item k. The constraint matrix for this set of possible bundles is indeed balanced, and so the corresponding polyhedron is integral, and the solution can be found using linear programming.

An unrelated class of auction problems that can be solved easily is that in which we allow bundles of no more than two items. It is possible to show that for this sort of auction problem an optimal allocation can be computed in quadratic time.

In many cases the solutions to the associated linear program will not be integral. In these cases we must resort to using *heuristic methods* to find solutions to the auction problem. We can distinguish between *complete* heuristic methods, which are guaranteed to find an optimal solution if one exists, and *incomplete* methods, which are not guaranteed to find optimal solutions. As an example, one obvious incomplete heuristic method is the *greedy method*, in which we iteratively allocate the bundle which maximizes the ratio of the valuation of the bundle to the number of goods in the bundle.

Unfortunately, in general there doesn't exist an algorithm that can guarantee that you reach even an approximate solution that is within a fixed fraction of the optimal solution, no matter how small the fraction. However, there does exist an algorithm that guarantees a solution that is within $1/\sqrt{k}$ of the optimal solution, where k is the number of goods.

In recent years we have seen an explosion of specialized search algorithms for combinatorial auctions. The complete methods guarantee optimal results, but not rapid convergence (and of course in the worst case they take exponential time). Incomplete, greedy-search methods, such as the one described above can perform an order of magnitude faster. As we move forward, we will need a uniform means of testing and evaluating the performance the different heuristic algorithms.

7.5 Other Topics in Combinatorial Auctions

7.5.1 Approximation

7.5.2 Ascending Combinatorial Auctions

7.5.3 Communication Complexity

7.5.4 Selfish Routing

7.5.5 Congestion Control

7.5.6 Fault-Tolerant Mechanism Design

7.5.7 Rational Computation

7.6 History and References