10

Protocols for Strategic Agents: Mechanism Design

As we discussed in the previous chapter, social choice theory is nonstrategic; it takes the preferences of the agents as given, and investigates ways in which they can be aggregated. But of course those preferences are usually not known. What you have, instead, is that the various agents *declare* their preferences, which they may do truthfully or not. Assuming the agents are self interested, in general they will not reveal their true preferences. Since as a designer you wish to find an optimal outcome with respect to the agents' true preferences (e.g., electing a leader that truly reflects the agents' preferences), optimizing with respect to the declared preferences will not in general achieve the objective.

10.1 Introduction

Mechanism design is a strategic version of social choice theory, which adds the assumption that agents will behave so as to maximize their individual payoffs. For example, in an election agents may not vote their true preference.

10.1.1 *Example: strategic voting*

Consider again our babysitting example. This time, in addition to Will, Liam, and Vic you must also babysit their devious new friend, Ray. Again, you invite each child to select their favorite among the three activities—going to the video arcade (a), playing basketball (b), and going for a leisurely car ride (c). As before, you announce that you will select the activity with the highest number of votes, breaking ties alphabetically. Consider the case in which the true preferences of the kids are as follows:

$$\begin{aligned} \text{Will:} &\quad b \succ a \succ c \\ \text{Liam:} &\quad b \succ a \succ c \\ \text{Vic:} &\quad a \succ c \succ b \\ \text{Ray:} &\quad c \succ a \succ b \end{aligned}$$

Will, Liam, and Vic are sweet souls who always tell you their true preferences. But little Ray, he is always figuring things out and so he goes through the

following reasoning process. He prefers the most sedentary activity possible (hence his preference ordering). But he knows his friends well, an in particular he knows which activity each of them will vote for. He thus knows that if he votes for his true passion—slumping in the car for a few hours (c)—he will end up playing basketball (b). So he votes for going to the arcade (a), ensuring that this indeed is the outcome. Is there anything you can do to prevent such manipulation by little Ray?

mechanism design

implementation theory

This is where *mechanism design*, or *implementation theory*, comes in. Mechanism design is sometimes colloquially called "inverse game theory." Our discussion of game theory in Chapter 3 was framed as follows: Given an interaction among a set of agents, how do we predict or prescribe the course of action of the various agents participating in the interaction? In mechanism design, 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 preferences, and ask whether we can design a game such that, no matter what the secret preferences of the agents actually are, the equilibrium of the game is guaranteed to have a certain desired property or set of properties.[1] Mechanism design 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."

10.1.2 *Example: buying a shortest path*

auction theory

Like social choice theory, the scope of mechanism design is broader than voting. The most famous application of mechanism design is *auction theory*, to which we devote Chapter 11. However, mechanism design has many other applications. Consider the transportation network depicted in Figure 10.1.

In Section 6.4.5 we considered a selfish routing problem where agents selfishly decide where to send their traffic in a network that responded to congestion in a predictable way. Here we consider a different problem. In Figure 10.1 the number next to a given edge is the cost of transporting along that edge, but these costs are the private information of the various shippers that own each edge. The task here is to find the shortest (least-cost) path from S to T; this is hard because the shippers may lie about their costs. Your one advantage is that you know that they are interested in maximizing their revenue. How can use you that knowledge to extract from them the information needed to compute the desired path?

1. Actually, as we shall see, technically speaking what we design is not a game but a mechanism that together with the secret utility functions defines a Bayesian game.

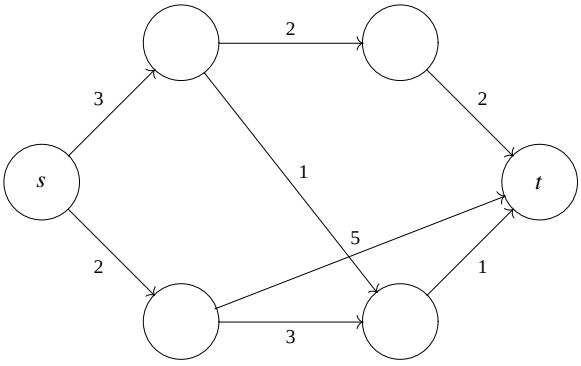


Figure 10.1 Transportation network with selfish agents.

10.2 Mechanism design with unrestricted preferences

We begin by introducing some of the broad principles of mechanism design, placing no restriction on the preferences agents can have. (We will consider such restrictions in later sections.) Because mechanism design is most often studied in settings where agents' preferences are unknown, we start by defining a Bayesian game setting, basing it on the epistemic types definition of Bayesian games that we gave in Section 6.3.1. The key difference is that the setting does not include actions for the agents, and instead defines the utility functions over the set of possible outcomes.[2]

Bayesian game setting

Definition 10.2.1 (Bayesian game setting) *A* Bayesian game setting *is a tuple* (N, O, Θ, p, u)*, where:*

- N *is a finite set of* n *agents;*
- O *is a set of outcomes;*
- $\Theta = \Theta_1 \times \cdots \times \Theta_n$ *is a set of possible joint type vectors;*
- p *is a (common-prior) probability distribution on* Θ*; and*
- $u = (u_1, \ldots, u_n)$*, where* $u_i : O \times \Theta \mapsto \mathbb{R}$ *is the utility function for each player* i*.*

Given a Bayesian game setting, we can define a mechanism.

mechanism

Definition 10.2.2 (Mechanism) *A* mechanism *(for a Bayesian game setting* (N, O, Θ, p, u)*) is a pair* (A, M)*, where:*

- $A = A_1 \times \cdots \times A_n$*, where* A_i *is the set of actions available to agent* $i \in N$*; and*
- $M : A \mapsto \Pi(O)$ *maps each action profile to a distribution over outcomes.*

2. Recall from our original discussion of utility theory in Section 3.1.2 that utility functions always map from outcomes to real values; we had previously assumed that $O = A$. We now relax this assumption, and so make explicit the utility functions' dependence on the chosen outcome.

A mechanism is *deterministic* if for every $a \in A$, there exists $o \in O$ such that $M(a)(o) = 1$; in this case we write simply $M(a) = o$.

10.2.1 *Implementation*

Together, a Bayesian game setting and a mechanism define a Bayesian game. The aim of mechanism design is to select a mechanism, given a particular Bayesian game setting, whose equilibria have desirable properties. We now define the most fundamental such property: that the outcomes that arise when the game is played are consistent with a given social choice function.

implementation in dominant strategies

Definition 10.2.3 (Implementation in dominant strategies) *Given a Bayesian game setting* (N, O, Θ, p, u)*, a mechanism* (A, M) *is an* implementation in dominant strategies *of a social choice function* C *(over* N *and* O*) if for any vector of utility functions* u*, the game has an equilibrium in dominant strategies, and in any such equilibrium* a^* *we have* $M(a^*) = C(u)$.[3]

strategy-proof

A mechanism that gives rise to dominant strategies is sometimes called *strategy-proof*, because there is no need for agents to reason about each others' actions in order to maximize their utility.

In the aforementioned babysitter example, the pair consisting of "each child votes for one choice" and "the activity selected is one with the most votes, breaking ties alphabetically" is a well-formed mechanism, since it specifies the actions available to each child and the outcome depending on the choices made. Now consider the social choice function "the selected activity is that which is the top choice of the maximal number of children, breaking ties alphabetically." Clearly the mechanism defined by the babysitter does not implement this function in dominant strategies. For example, the preceding instance of it has no dominant strategy for Ray.

This suggests that the above definition can be relaxed, and can appeal to solution concepts that are weaker than dominant-strategy equilibrium. For example, one can appeal to the Bayes–Nash equilibrium.[4]

implementation in Bayes–Nash equilibrium

Definition 10.2.4 (Implementation in Bayes–Nash equilibrium) *Given a Bayesian game setting* (N, O, Θ, p, u)*, a mechanism* (A, M) *is an* implementation in Bayes–Nash equilibrium *of a social choice function* C *(over* N *and* O*) if there exists a Bayes–Nash equilibrium of the game of incomplete information*

3. The careful reader will notice that because we have previously defined social choice functions as deterministic, we here end up with a mechanism that selects outcomes deterministically as well. Of course, this definition can be extended to describe randomized social choice functions and mechanisms.

4. It is possible to study mechanism design in complete-information settings as well. This leads to the idea of Nash implementation, which is a sensible concept when the agents know each other's utility functions but the designer does not. This last point is crucial: if the designer did know, he could simply select the social choice directly, and we would return to the social choice setting studied in Chapter 9. We do not discuss Nash implementation further because it plays little role in the material that follows.

(N, A, Θ, p, u) such that for every $\theta \in \Theta$ and every action profile $a \in A$ that can arise given type profile θ in this equilibrium, we have that $M(a) = C(u(\cdot, \theta))$.

A classical example of Bayesian mechanism design is auction design. While we defer a lengthier discussion of auctions to Chapter 11, the basic idea is as follows. The designer wishes, for example, to ensure that the bidder with the highest valuation for a given item will 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 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 his own valuation.

Finally, there exist implementation concepts that are satisfied by a larger set of strategy profiles than implementation in dominant strategies, but that are not guaranteed to be achievable for any given social choice function and set of preferences, unlike Bayes–Nash implementation. For example, we could consider only symmetric Bayes–Nash equilibria, on the principle that strategies that depend on agent identities would be less likely to arise in practice. It turns out that symmetric Bayes–Nash equilibria always exist in symmetric Bayesian games. A second implementation notion that deserves mention is *ex post* implementation. Recall from Section 6.3.4 that an *ex post* equilibrium has the property that no agent can ever gain by changing his strategy even if he observes the other agents' types, as long as all the other agents follow the equilibrium strategies. Thus, unlike a Bayes–Nash equilibrium, an *ex post* equilibrium does not depend on the type distribution. Regardless of the implementation concept, we can require that the desired social choice function is implemented in the only equilibrium, in every equilibrium or in at least one equilibrium of the underlying game.

10.2.2 *The revelation principle*

truthfulness

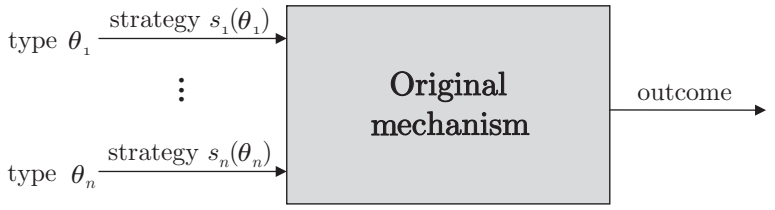
One property that is often desired of mechanisms is called *truthfulness*. This property holds when agents truthfully disclose their preferences to the mechanism in equilibrium. It turns out that this property can always be achieved regardless of the social choice function implemented and of the agents' preferences.

direct mechanism

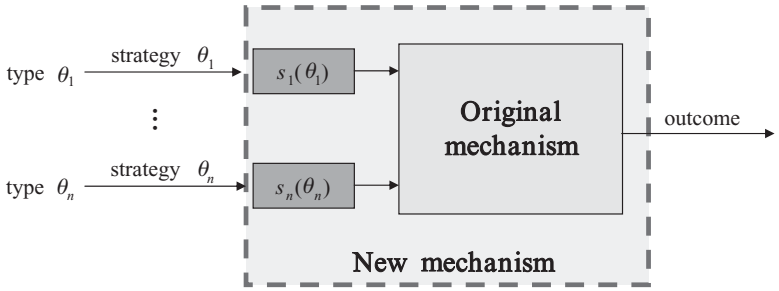
More formally, a *direct mechanism* is one in which the only action available to each agent is to announce his private information. Since in a Bayesian game an agent's private information is his type, direct mechanisms have $A_i = \Theta_i$. When an agent's set of actions is the set of all his possible types, he may lie and announce a type $\hat{\theta}_i$ that is different from his true type θ_i. A direct mechanism is said to

truthful

be *truthful (*or *incentive compatible)* if, for any type vector θ, in equilibrium



(a) Revelation principle: original mechanism



(b) Revelation principle: new mechanism

Figure 10.2 The revelation principle: how to construct a new mechanism with a truthful equilibrium, given an original mechanism with equilibrium $(s_1, \ldots, s_n)$.

of the game defined by the mechanism and under the appropriate equilibrium concept every agent i's strategy is to announce his true type, so that $\hat{\theta}_i = \theta_i$. We can thus speak about *incentive compatibility in dominant strategies* and *Bayes–Nash incentive compatibility*. Our claim that truthfulness can always be achieved implies, for example, that the social choice functions implementable by dominant-strategy truthful mechanisms are precisely those implementable by strategy-proof direct mechanisms. This means that we can, without loss of coverage, limit ourselves to a small sliver of the space of all mechanisms.

incentive compatibility in dominant strategies

Bayes–Nash incentive compatibility

revelation principle

Theorem 10.2.5 (Revelation principle) *If there exists any mechanism that implements a social choice function C in dominant strategies then there exists a direct mechanism that implements C in dominant strategies and is truthful.*

Proof. Consider an arbitrary mechanism for n agents that implements a social choice function C in dominant strategies. This mechanism is illustrated in Figure 10.2a. Let $s_1, \ldots, s_n$ denote the dominant strategies for agents $1, \ldots, n$. We will construct a new mechanism which *truthfully* implements C. Our new mechanism will ask the agents for their utility functions, use them to determine $s_1, \ldots, s_n$, the agents' dominant strategies under the original mechanism, and then choose the outcome that would have been chosen by the original mechanism for agents following the strategies $s_1, \ldots, s_n$. This new mechanism is illustrated in Figure 10.2b.

Assume that some agent i would be better off declaring a utility function u_i^* to the new mechanism rather than his true utility function u_i. This implies

that i would have preferred to follow some different strategy s_i^* in the original mechanism rather than s_i, contradicting our assumption that s_i is a dominant strategy for i. (Intuitively, if i could gain by lying to the new mechanism, he could likewise gain by "lying to himself" in the original mechanism.) Thus the new mechanism is dominant-strategy truthful. ■

In other words, any solution to a mechanism design problem can be converted into one in which agents always reveal their true preferences, if the new mechanism "lies for the agents" in just the way they would have chosen to lie to the original mechanism. The revelation principle is arguably the most basic 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 fact one can restrict one's attention to truthful, direct mechanisms.

As we asserted earlier, the revelation principle does not apply only to implementation in dominant strategies; we have stated the theorem in this way only to keep things simple. Following exactly the same argument we can argue that, for example, a mechanism that implements a social choice function in a Bayes–Nash equilibrium can be converted into a direct, Bayes–Nash incentive compatible mechanism.

The argument we used to justify the revelation principle also applies to original mechanisms that are indirect (e.g., ascending auctions). The new, direct mechanism can take the agents' utility functions, construct their strategies for the indirect mechanism, and then simulate the indirect mechanism to determine which outcome to select. One caveat is that, even if the original indirect mechanism had a unique equilibrium, there is no guarantee that the new revelation mechanism will not have additional equilibria.

Before moving on, we finally offer some computational caveats to the revelation principle. Observe that the general effect of constructing a revelation mechanism is to push an additional computational burden onto the mechanism, as is implicit in Figure 10.2b. There are many settings in which agents' equilibrium strategies are computationally difficult to determine. When this is the case, the additional burden absorbed by the mechanism may be considerable. Furthermore, the revelation mechanism forces the agents to reveal their types completely. There may be settings in which agents are not willing to compromise their privacy to this degree. (Observe that the original mechanism may require them to reveal much less information.) Finally, even if not objectionable on privacy grounds, this full revelation can sometimes place an unreasonable burden on the communication channel. For all these reasons, in practical settings one must apply the revelation principle with caution.

10.2.3 *Impossibility of general, dominant-strategy implementation*

We now ask what social choice functions can be implemented in dominant strategies. Given the revelation principle, we can restrict our attention to truthful mechanisms. The first answer is disappointing.

Theorem 10.2.6 (Gibbard–Satterthwaite) *Consider any social choice function C of N and O. If:*

1. $|O| \geq 3$ *(there are at least three outcomes);*
2. C *is* onto*; that is, for every $o \in O$ there is a preference profile $[\succ]$ such that $C([\succ]) = o$ (this property is sometimes also called* citizen sovereignty*); and*
3. C *is dominant-strategy truthful,*

then C is dictatorial.

If Theorem 10.2.6 is reminiscent of the Muller–Satterthwaite theorem (Theorem 9.4.8) this is no accident, since Theorem 10.2.6 is implied by that theorem as a corollary. Note that this negative result is specific to dominant-strategy implementation. It does not hold for the weaker concepts of Nash or Bayes–Nash equilibrium implementation.

10.3 Quasilinear preferences

If we are to design a dominant-strategy truthful mechanism that is not dictatorial, we are going to have to relax some of the conditions of the Gibbard–Satterthwaite theorem. First, we relax the requirement that agents be able to express any preferences and replace it with the requirement that agents be able to express any preferences in a limited set. Second, we relax the condition that the mechanism be onto. We now introduce our limited set of preferences.

quasilinear utility functions

quasilinear preferences

Definition 10.3.1 (Quasilinear utility function) *Agents have* quasilinear utility functions *(or* quasilinear preferences*) in an n-player Bayesian game when the set of outcomes is $O = X \times \mathbb{R}^n$ for a finite set X, and the utility of an agent i given joint type θ is given by $u_i(o, \theta) = u_i(x, \theta) - f_i(p_i)$, where $o = (x, p)$ is an element of O, $u_i : X \times \Theta \mapsto \mathbb{R}$ is an arbitrary function and $f_i : \mathbb{R} \mapsto \mathbb{R}$ is a strictly monotonically increasing function.*

Intuitively, we split outcomes into two pieces that are linearly related. First, X represents a finite set of nonmonetary outcomes, such as the allocation of an object to one of the bidders in an auction or the selection of a candidate in an election. Second, p_i is the (possibly negative) payment made by agent i to the mechanism, such as a payment to the auctioneer.

What does it mean to assume that agents' preferences are quasilinear? First, it means that we are in a setting in which the mechanism can choose to charge or reward the agents by an arbitrary monetary amount. Second, and more restrictive, it means that an agent's degree of preference for the selection of any choice $x \in X$ is independent from his degree of preference for having to pay the mechanism some amount $p_i \in \mathbb{R}$. Thus an agent's utility for a choice cannot depend on the total amount of money that he has (e.g., an agent cannot value having a yacht more if he is rich than if he is poor). Finally, it means that agents care only about the choice selected and about their own payments: in particular, they do not care about the monetary payments made or received by other agents.

Strictly speaking, we have defined quasilinear preferences in a way that fixes the set of agents. However, we generally consider families of quasilinear problems, for any set of agents. For example, consider a voting game of the sort discussed earlier. You would want to be able to speak about 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 quasilinear utility function is still defined when any one agent is taken away. In this case the set of nonmonetary outcomes must be updated (e.g., 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.

10.3.1 *Risk attitudes*

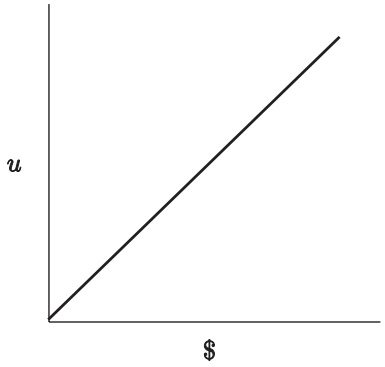
There is still one part of the definition of quasilinear preferences that we have not discussed—the functions f_i. Before defining them, let us consider a question that may at first seem a bit nonsensical. Recall that we have said that p_i denotes the amount an agent i has to pay the mechanism. How much does i value a dollar? To make sense of this question, we must first note that utility is specified in its own units, rather than in units of currency, so we need to perform some kind of conversion. (Recall the discussion at the end of Section 3.1.2.) Indeed, this conversion can be understood as the purpose of f_i. However, the conversion is nontrivial because for most people the value of a dollar depends on the amount of money they start out with in the first place. (For example, if you are broke and starving then a dollar could lead to a substantial increase in your utility; if you are a millionaire, you might not bend over to pick up the same dollar if it was lying on the street.) To make the same point in another way, consider a fair lottery in which a ticket costs $\$x$ and pays off $\$2x$ half of the time. Holding your wealth constant, your willingness to participate in this lottery would probably depend on x. Most people are willing to play for sufficiently small values of x (say \$1), but not for larger values (say $\$10,000$). However, we have modeled agents as expected utility maximizers—how can we express the idea that an agent's willingness to participate in this lottery can depend on x, when the lottery's expected value is the same in both cases?

These two examples illustrate that we will often want the f_i functions to be nonlinear. The curvature of f_i gives i's *risk attitude*, which we can understand as the way that i feels about lotteries such as the one just described.

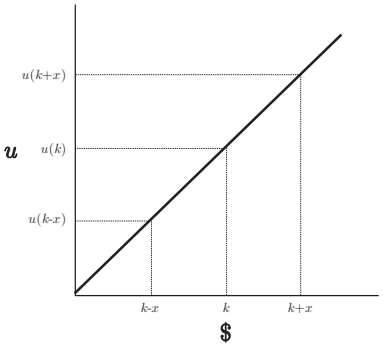
risk attitude

If an agent i simply wants to maximize his expected revenue, we say the agent is *risk neutral*. Such an agent has a linear value for money, as illustrated in Figure 10.3a. To see why this is so, consider Figure 10.3b. This figure describes a situation where the agent starts out with an endowment of $\$k$, and must decide whether or not to participate in a fair lottery that awards $\$k + x$ half the time, and $\$k - x$ the other half of the time. From looking at the graph, we can see that $u(k) = \frac{1}{2}u(k - x) + \frac{1}{2}u(k + x)$—the agent is indifferent between participating in the lottery and not participating. This is what we would expect, as the lottery's expected value is k, the same as the value for not participating.

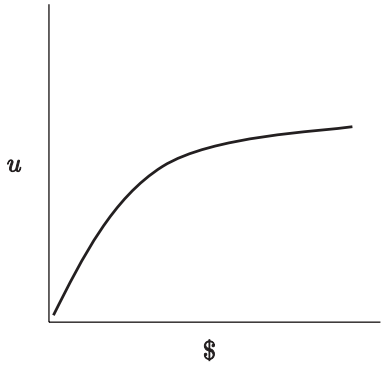
risk neutral



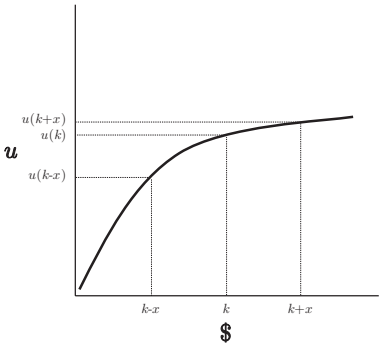
(a) Risk neutrality



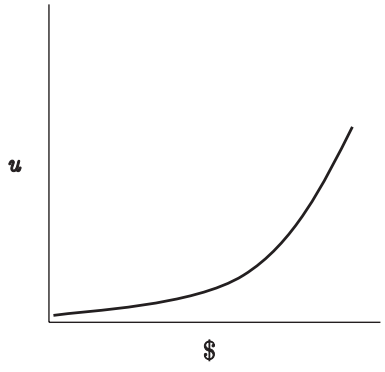
(b) Risk neutrality: fair lottery



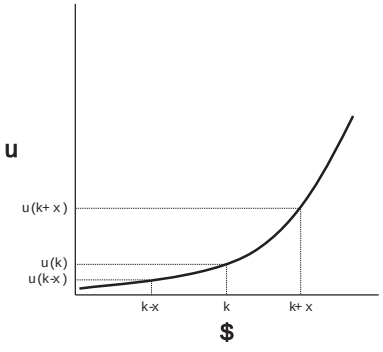
(c) Risk aversion



(d) Risk aversion: fair lottery



(e) Risk seeking



(f) Risk seeking: fair lottery

Figure 10.3 Risk attitudes: risk aversion, risk neutrality, risk seeking, and in each case, utility for the outcomes of a fair lottery.

risk averse

In contrast, consider the value-for-money curve illustrated in Figure 10.3c. We call such an agent *risk averse*—he has a sublinear value for money, which means that he prefers a "sure thing" to a risky situation with the same expected value. Consider the same fair lottery described earlier from the point of view

of a risk-averse agent, as illustrated in Figure 10.3d. We can see that for this agent $u(k) > \frac{1}{2}u(k-x) + \frac{1}{2}u(k+x)$—the marginal disutility of losing $\$x$ is greater than the marginal utility of gaining $\$x$, given an initial endowment of k.

risk seeking

Finally, the opposite of risk aversion is *risk seeking*, illustrated in Figure 10.3e. Such an agent has a superlinear value for money, which means that the agent prefers engaging in lotteries to a sure thing with the same expected value. This is shown in Figure 10.3f. For example, an agent might prefer to spend \$1 to buy a ticket that has a $\frac{1}{1,000}$ chance of paying off \$1, 000, as compared to keeping the dollar.

The examples above suggest that people might exhibit different risk attitudes in different regions of f_i. For example, a person could be risk seeking for very small amounts of money, risk neutral for moderate amounts and risk averse for large amounts. Nevertheless, in what follows we will assume that agents are risk neutral unless indicated otherwise. The assumption of risk neutrality is made partly for mathematical convenience, partly to avoid making an (often difficult to justify) assumption about the particular shape of agents' value-for-money curves, and partly because risk neutrality is reasonable when the amounts of money being exchanged through the mechanism are moderate. Considerable work in the literature extends results such as those presented in this chapter and the next to the case of agents with different risk attitudes.

Even once we have assumed that agents are risk neutral, there remains one more degree of freedom in agents' utility functions: the slope of f_i. If every agent's value-for-money curve is linear and has the same slope ($\forall i \in N,\ f_i(p) = \beta p$, for $\beta \in \mathbb{R}_+$), then we say that the agents have *transferable utility*. This name reflects the fact that, regardless of the nonmonetary choice $x \in X$, one agent can transfer any given amount of utility to another by giving that agent an appropriate amount of money. More formally, for all $x \in X$, for any pair of agents $i, j \in N$ and for any $k \in \mathbb{R}$, i's utility is increased by exactly k and j's utility decreased by exactly k when j pays i the amount $\frac{k}{\beta}$. We will assume that this property holds for the remainder of this chapter and throughout Chapter 11, except where we indicate otherwise.

transferable utility

10.3.2 *Mechanism design in the quasilinear setting*

Now that we have defined the quasilinear preference model, we can talk about the design of mechanisms for agents with these preferences. As discussed earlier, we assume that agents are risk neutral and have transferable utility. For convenience, let $\beta_i = 1$, meaning that we can think of agents' utilities for different choices as being expressed in dollars. We concentrate on Bayesian games because most mechanism design is performed in such domains.

First, we point out that since quasilinear preferences split the outcome space into two parts, we can modify our formal definition of a mechanism accordingly.

mechanism in the quasilinear setting

Definition 10.3.2 (Quasilinear mechanism) *A* mechanism in the quasilinear setting *(for a Bayesian game setting $(N, O = X \times \mathbb{R}^n, \Theta, p, u)$) is a triple $(A, \chi, \wp)$, where*

- $A = A_1 \times \cdots \times A_n$, *where A_i is the set of actions available to agent $i \in N$;*
- $\chi : A \mapsto \Pi(X)$ *maps each action profile to a distribution over choices; and*
- $\wp : A \mapsto \mathbb{R}^n$ *maps each action profile to a payment for each agent.*

choice rule

payment rule

In effect, we have split the function M into two functions χ and $\wp$, where χ is the *choice rule* and $\wp$ is the *payment rule*. We will use the notation $\wp_i$ to denote the payment function for agent i.

A direct revelation mechanism in the quasilinear setting is one in which each agent is asked to state his type.

direct quasilinear mechanism

Definition 10.3.3 (Direct quasilinear mechanism) *A* direct quasilinear mechanism *(for a Bayesian game setting $(N, O = X \times \mathbb{R}^n, \Theta, p, u)$) is a pair $(\chi, \wp)$. It defines a standard mechanism in the quasilinear setting, where for each i, $A_i = \Theta_i$.*

conditional utility independence

In many quasilinear mechanism design settings it is helpful to make the assumption that agents' utilities depend only on their own types, a property that we call *conditional utility independence*.[5]

conditional utility independence

Definition 10.3.4 (Conditional utility independence) *A Bayesian game exhibits* conditional utility independence *if for all agents $i \in N$, for all outcomes $o \in O$ and for all pairs of joint types θ and $\theta' \in \Theta$ for which $\theta_i = \theta'_i$, it holds that $u_i(o, \theta) = u_i(o, \theta')$.*

valuation

We will assume conditional utility independence for the rest of this section, and indeed for most of the rest of the chapter. When we do so, we can write an agent i's utility function as $u_i(o, \theta_i)$, since it does not depend on the other agents' types. We can also refer to an agent's *valuation* for choice $x \in X$, written $v_i(x) = u_i(x, \theta)$. v_i should be thought of as the maximum amount of money that i would be willing to pay to get the mechanism designer to implement choice x—in fact, having to pay this much would exactly make i indifferent about whether he was offered this deal or not.[6] Note that an agent's valuation depends on his type, even though we do not explicitly refer to θ_i. In the future when we discuss direct quasilinear mechanisms, we will usually mean mechanisms that ask agents to declare their valuations for each choice; of course, this alternate definition is equivalent to Definition 10.3.3.[7] Let V_i denote the set of all possible valuations for agent i. We will use the notation $\hat{v}_i \in V_i$ to denote the valuation that agent

5. This assumption is sometimes referred to as *privacy*. We avoid that terminology here because the assumption does not imply that agents cannot learn about others' utility functions by observing their own types.

6. Observe that here we rely upon the assumption of risk neutrality discussed earlier. Furthermore, observe that it is also meaningful to extend the concept of valuation beyond settings in which conditional utility independence holds; in such cases, we say that agents do not know their own valuations. We consider one such setting in Section 11.1.10.

7. Here we assume, as is common in the literature, that the mechanism designer knows each agent's value-for-money function f_i.

i declares to such a direct mechanism, which may be different from his true valuation v_i. We denote the vector of all agents' declared valuations as $\hat{v}$ and the set of all possible valuation vectors as V. Finally, denote the vector of declared valuations from all agents other than i as $\hat{v}_{-i}$.

Now we can state some properties that it is common to require of quasilinear mechanisms.

truthful

Definition 10.3.5 (Truthfulness) *A quasilinear mechanism is* truthful *if it is direct and* $\forall i \forall v_i$, *agent i's equilibrium strategy is to adopt the strategy* $\hat{v}_i = v_i$.

Of course, this is equivalent to the definition of truthfulness that we gave in Section 10.2.2; we have simply updated the notation for the quasilinear utility setting.

strict Pareto efficiency

efficiency

Definition 10.3.6 (Efficiency) *A quasilinear mechanism is* strictly Pareto efficient, *or just* efficient, *if in equilibrium it selects a choice* x *such that* $\forall v \forall x', \sum_i v_i(x) \geq \sum_i v_i(x')$.

That is, an efficient mechanism selects the choice that maximizes the sum of agents' utilities, disregarding the monetary payments that agents are required to make. We describe this property as *economic efficiency* when there is a danger that it will be confused with other (e.g., computational) notions of efficiency. Observe that efficiency is defined in terms of agents' true valuations, not their declared valuations. This condition is also known as *social welfare maximization*.

social welfare maximization

The attentive reader might wonder about the relationship between strict Pareto efficiency as defined in Definitions 3.3.2 and 10.3.6. The underlying concept is indeed the same. The reason why we can get away with summing agents' valuations here arises from our assumption that agents' preferences are quasilinear, and hence that agents' utilities for different choices can be traded off against different payments. Recall that we observed in Section 3.3.1 that there can be many Pareto efficient outcomes because of the fact that agents' utility functions are only unique up to positive affine transformations. In a quasilinear setting, if we include the operator of the mechanism[8] as an agent who values money linearly and is indifferent between the mechanism's choices, it can be shown that all Pareto efficient outcomes involve the mechanism making the same choice and differ only in monetary allocations.

budget balance

Definition 10.3.7 (Budget balance) *A quasilinear mechanism is* budget balanced *when* $\forall v, \sum_i \wp_i(s(v)) = 0$, *where* s *is the equilibrium strategy profile.*

In other words, regardless of the agents' types, the mechanism collects and disburses the same amount of money from and to the agents, meaning that it makes neither a profit nor a loss.

weak budget balance

Sometimes we relax this condition and require only *weak budget balance*, meaning that $\forall v, \sum_i \wp_i(s(v)) \geq 0$ (i.e., the mechanism never takes a loss, but it may make a profit). Finally, we can require that either strict or weak budget

8. For example, this would be a seller in a single-sided auction, or a market maker in a double-sided market.

balance hold *ex ante*, which means that $\mathbb{E}_v\left[\sum_i \wp_i(s(v))\right]$ is either equal to or greater than zero. (That is, the mechanism is required to break even or make a profit only on expectation.)

ex interim individual rationality

Definition 10.3.8 (*Ex interim* individual rationality) *A quasilinear mechanism is* ex interim individually rational *when*

$$\forall i \forall v_i,\ \mathbb{E}_{v_{-i}|v_i}\left[v_i(\chi(s_i(v_i), s_{-i}(v_{-i}))) - \wp_i(s_i(v_i), s_{-i}(v_{-i}))\right] \geq 0,$$

where s is the equilibrium strategy profile.

This condition requires that no agent loses by participating in the mechanism. We call it *ex interim* because it holds for *every* possible valuation for agent i, but averages over the possible valuations of the other agents. This approach makes sense because it requires that, based on the information that an agent has when he chooses to participate in a mechanism, no agent would be better off choosing not to participate. Of course, we can also strengthen the condition to say that no agent *ever* loses by participation.

ex post individual rationality

Definition 10.3.9 (*Ex post* individual rationality) *A quasilinear mechanism is* ex post individually rational *when* $\forall i \forall v,\ v_i(\chi(s(v))) - \wp_i(s(v)) \geq 0$, *where s is the equilibrium strategy profile.*

We can also restrict mechanisms based on their computational requirements rather than their economic properties.

tractability

Definition 10.3.10 (Tractability) *A quasilinear mechanism is* tractable *when* $\forall a \in A,\ \chi(a)$ *and* $\wp(a)$ *can be computed in polynomial time.*

Finally, in some domains there will be many possible mechanisms that satisfy the constraints we choose, meaning that we need to have some way of choosing among them. (And as we will see later, for other combinations of constraints no mechanisms exist at all.) The usual approach is to define an optimization problem that identifies the optimal outcome in the feasible set. For example, although we have defined efficiency as a constraint, it is also possible to soften the constraint and require the mechanism to achieve as much social welfare as possible. Here we define some other quantities that a mechanism designer can seek to optimize.

First, the mechanism designer can take a selfish perspective. Interestingly, this goal turns out to be quite different from the goal of maximizing social welfare. (We give an example of the differences between these approaches when we consider single-good auctions in Section 11.1.)

revenue maximization

Definition 10.3.11 (Revenue maximization) *A quasilinear mechanism is* revenue maximizing *when, among the set of functions χ and $\wp$ that satisfy the other constraints, the mechanism selects the χ and $\wp$ that maximize $\mathbb{E}_v\left[\sum_i \wp_i(s(v))\right]$, where $s(v)$ denotes the agents' equilibrium strategy profile.*

Conversely, the designer might try to collect as *little* revenue as possible, for example if the mechanism uses payments only to provide incentives, but is not intended to make money. The budget balance constraint is the best way to solve this problem, but sometimes it is impossible to satisfy. In such cases, one approach is to set weak budget balance as a constraint and then to pick the revenue minimizing mechanism, effectively softening the (strict) budget balance constraint. Here we present a *worst-case* revenue minimization objective; of course, an average-case objective is also possible.

revenue minimization

Definition 10.3.12 (Revenue minimization) *A quasilinear mechanism is* revenue minimizing *when, among the set of functions x and $\wp$ that satisfy the other constraints, the mechanism selects the x and $\wp$ that minimize* $\max_v \sum_i \wp_i(s(v))$ *in equilibrium, where $s(v)$ denotes the agents' equilibrium strategy profile.*

The mechanism designer might be concerned with selecting a *fair* outcome. However, the notion of fairness can be tricky to formalize. For example, an outcome that fines all agents $100 and makes a choice that all agents hate equally is in some sense fair, but it does not seem desirable. Here we define so-called *maxmin fairness*, which says that the fairest outcome is the one that makes the least-happy agent the happiest. We also take an expected value over different valuation vectors, but we could instead have required a mechanism that does the best in the worst case.

maxmin fairness

Definition 10.3.13 (Maxmin fairness) *A quasilinear mechanism is* maxmin fair *when, among the set of functions x and $\wp$ that satisfy the other constraints, the mechanism selects the x and $\wp$ that maximize* $\mathbb{E}_v[\min_{i \in N} v_i(x(s(v))) - \wp_i(s(v))]$*, where $s(v)$ denotes the agents' equilibrium strategy profile.*

Finally, the mechanism designer might not be able to implement a social-welfare-maximizing mechanism (e.g., in order to satisfy a tractability constraint) but may want to get as close as possible. Thus, the goal could be minimizing the *price of anarchy* (see Definition 6.4.11), the worst-case ratio between optimal social welfare and the social welfare achieved by the given mechanism. Here we also consider the worst case across agent valuations.

price-of-anarchy minimization

Definition 10.3.14 (Price-of-anarchy minimization) *A quasilinear mechanism* minimizes the price of anarchy *when, among the set of functions x and $\wp$ that satisfy the other constraints, the mechanism selects the x and $\wp$ that minimize*

$$\max_{v \in V} \frac{\max_{x \in X} \sum_{i \in N} v_i(x)}{\sum_{i \in N} v_i\left(x(s(v))\right)},$$

where $s(v)$ denotes the agents' equilibrium strategy profile in the worst *equilibrium of the mechanism—that is, the one in which $\sum_{i \in N} v_i(x(s(v)))$ is the smallest.*[9]

9. Note that we have to modify this definition along the lines we used in Definition 6.4.11 if $\sum_{i \in N} v_i(x(s(v))) = 0$ is possible.

10.4 Efficient mechanisms

Efficiency (Definition 10.3.6) is often considered to be one of the most important properties for a mechanism to satisfy in the quasilinear setting. For example, whenever an inefficient choice is selected, it is possible to find a set of side payments among the agents with the property that all agents would prefer the efficient choice in combination with the side payments to the inefficient choice. (Intuitively, the sum of agents' valuations for the efficient choice is greater than for the inefficient choice. Thus, the agents who prefer the efficient choice would still strictly prefer it even if they had to make side payments to the other agents so that each of them also strictly preferred the efficient choice.) Consequently, a great deal of research has considered the design of mechanisms that are guaranteed to select efficient choices when agents follow dominant or equilibrium strategies. In this section we survey these mechanisms.

10.4.1 *Groves mechanisms*

The most important family of efficient mechanisms are the Groves mechanisms.

Groves mechanism

Definition 10.4.1 (Groves mechanisms) Groves mechanisms *are direct quasilinear mechanisms* $(x, \wp)$*, for which*

$$x(\hat{v}) = \arg\max_x \sum_i \hat{v}_i(x),$$

$$\wp_i(\hat{v}) = h_i(\hat{v}_{-i}) - \sum_{j \neq i} \hat{v}_j(x(\hat{v})).$$

In other words, Groves mechanisms are direct mechanisms in which agents can declare any valuation function $\hat{v}$ (and thus any quasilinear utility function $\hat{u}$). The mechanism then optimizes its choice assuming that the agents disclosed their true utility function. An agent is made to pay an arbitrary amount $h_i(\hat{v}_{-i})$ which does not depend on his own declaration and is paid the sum of every other agent's declared valuation for the mechanism's choice. The fact that the mechanism designer has the freedom to choose the h_i functions explains why we refer to the *family* of Groves mechanisms rather than to a single mechanism.

The remarkable property of Groves mechanisms is that they provide a dominant-strategy truthful implementation of a social-welfare-maximizing social choice function. It is easy to see that if a Groves mechanism is dominant-strategy truthful, then it must be social-welfare-maximizing: the function x in Definition 10.4.1 performs exactly the maximization called for by Definition 10.3.6 when $\hat{v} = v$. Thus, it suffices to show the following.

Theorem 10.4.2 *Truth telling is a dominant strategy under any Groves mechanism.*

Proof. Consider a situation where every agent j other than i follows some arbitrary strategy $\hat{v}_j$. Consider agent i's problem of choosing the best strategy

$\hat{v}_i$. As a shorthand, we write $\hat{v} = (\hat{v}_{-i}, \hat{v}_i)$. The best strategy for i is one that solves

$$\max_{\hat{v}_i} \left(v_i(x(\hat{v})) - \wp(\hat{v}) \right).$$

Substituting in the payment function from the Groves mechanism, we have

$$\max_{\hat{v}_i} \left(v_i(x(\hat{v})) - h_i(\hat{v}_{-i}) + \sum_{j \neq i} \hat{v}_j(x(\hat{v})) \right).$$

Since $h_i(\hat{v}_{-i})$ does not depend on $\hat{v}_i$, it is sufficient to solve

$$\max_{\hat{v}_i} \left(v_i(x(\hat{v})) + \sum_{j \neq i} \hat{v}_j(x(\hat{v})) \right).$$

The only way in which the declaration $\hat{v}_i$ influences the maximization above is through the term $v_i(x(\hat{v}))$. If possible, i would like to pick a declaration $\hat{v}_i$ that will lead the mechanism to pick an $x \in X$ that solves

$$\max_{x} \left(v_i(x) + \sum_{j \neq i} \hat{v}_j(x) \right). \tag{10.1}$$

The Groves mechanism chooses an $x \in X$ as

$$x(\hat{v}) = \arg\max_{x} \left(\sum_{i} \hat{v}_i(x) \right) = \arg\max_{x} \left(\hat{v}_i(x) + \sum_{j \neq i} \hat{v}_j(x) \right).$$

Thus, agent i leads the mechanism to select the choice that he most prefers by declaring $\hat{v}_i = v_i$. Because this argument does not depend in any way on the declarations of the other agents, truth telling is a dominant strategy for agent i. ■

Intuitively, the reason that Groves mechanisms are dominant-strategy truthful is that agents' externalities are internalized. Imagine a mechanism in which agents declared their valuations for the different choices $x \in X$ and the mechanism selected the efficient choice, but in which the mechanism did not impose any payments on agents. Clearly, agents would be able to change the mechanism's choice to another that they preferred by overstating their valuation. Under Groves mechanisms, however, an agent's utility does not depend only on the selected choice, because payments *are* imposed. Since agents are paid the (reported) utility of all the other agents under the chosen allocation, each agent becomes just as interested in maximizing the other agents' utilities as in maximizing his own. Thus, once payments are taken into account, all agents have the same interests.

Groves mechanisms illustrate a property that is generally true of dominant-strategy truthful mechanisms: an agent's payment does not depend on the amount of his own declaration. Although other dominant-strategy truthful mechanisms exist in the quasilinear setting, the next theorem shows that Groves mechanisms

are the *only* mechanisms that implement an efficient allocation in dominant strategies among agents with arbitrary quasilinear utilities.

Theorem 10.4.3 (Green–Laffont) *An efficient social choice function* $C : \mathbb{R}^{Xn} \mapsto X \times \mathbb{R}^n$ *can be implemented in dominant strategies for agents with unrestricted quasilinear utilities only if* $\wp_i(v) = h(v_{-i}) - \sum_{j \neq i} v_j(x(v))$.

Proof. From the revelation principle, we can assume that C is *truthfully* implementable in dominant strategies. Thus, from the definition of efficiency, the choice must be selected as

$$x = \arg\max_x \sum_i v_i(x)$$

We can write the payment function as

$$\wp_i(v) = h(v_i, v_{-i}) - \sum_{j \neq i} v_j(x(v)).$$

Observe that we can do this without loss of generality because h can be an arbitrary function that cancels out the second term. Now for contradiction, assume that there exist some v_i and v_i' such that $h(v_i, v_{-i}) \neq h(v_i', v_{-i})$.

Case 1: $x(v_i, v_{-i}) = x(v_i', v_{-i})$. Since C is truthfully implementable in dominant strategies, an agent i whose true valuation was v_i would be better off declaring v_i than v_i':

$$\begin{aligned} v_i(x(v_i, v_{-i})) - \wp_i(v_i, v_{-i}) &\geq v_i(x(v_i', v_{-i})) - \wp_i(v_i', v_{-i}), \\ \wp_i(v_i, v_{-i}) &\leq \wp_i(v_i', v_{-i}). \end{aligned}$$

In the same way, an agent i whose true valuation was v_i' would be better off declaring v_i' than v_i:

$$\begin{aligned} v_i'(x(v_i', v_{-i})) - \wp_i(v_i', v_{-i}) &\geq v_i'(x(v_i, v_{-i})) - \wp_i(v_i, v_{-i}), \\ \wp_i(v_i', v_{-i}) &\leq \wp_i(v_i, v_{-i}). \end{aligned}$$

Thus, we must have

$$\begin{aligned} \wp_i(v_i, v_{-i}) &= \wp_i(v_i', v_{-i}), \\ h(v_i, v_{-i}) - \sum_{j \neq i} v_j(x(v_i, v_{-i})) &= h(v_i', v_{-i}) - \sum_{j \neq i} v_j(x(v_i', v_{-i})). \end{aligned}$$

We are currently considering the case where $x(v_i, v_{-i}) = x(v_i', v_{-i})$. Thus we can write

$$\begin{aligned} h(v_i, v_{-i}) - \sum_{j \neq i} v_j(x(v_i, v_{-i})) &= h(v_i', v_{-i}) - \sum_{j \neq i} v_j(x(v_i, v_{-i})), \\ h(v_i, v_{-i}) &= h(v_i', v_{-i}). \end{aligned}$$

This is a contradiction.

Case 2: $x(v_i, v_{-i}) \neq x(v_i', v_{-i})$. Without loss of generality, let $h(v_i, v_{-i}) < h(v_i', v_{-i})$. Since this inequality is strict, there must exist some $\epsilon \in \mathbb{R}_+$ such that $h(v_i, v_{-i}) < h(v_i', v_{-i}) - \epsilon$.

Our mechanism must work for every v. Consider a case where i's valuation is

$$v_i''(x) = \begin{cases} -\sum_{j \neq i} v_j(x(v_i, v_{-i})) & x = x(v_i, v_{-i}); \\ -\sum_{j \neq i} v_j(x(v_i', v_{-i})) + \epsilon & x = x(v_i', v_{-i}); \\ -\sum_{j \neq i} v_j(x) - \epsilon & \text{for any other } x. \end{cases}$$

Note that agent i still declares his valuations as real numbers; they just happen to satisfy the constraints given above. Also note that the ϵ used here is the same $\epsilon \in \mathbb{R}_+$ mentioned earlier. From the fact that C is truthfully implementable in dominant strategies, an agent i whose true valuation was v_i'' would be better off declaring v_i'' than v_i:

$$v_i''(x(v_i'', v_{-i})) - \wp_i(v_i'', v_{-i}) \geq v_i''(x(v_i, v_{-i})) - \wp_i(v_i, v_{-i}). \tag{10.2}$$

Because our mechanism is efficient, it must pick the choice that solves

$$f = \max_x \left(v_i''(x) + \sum_j v_j(x) \right).$$

Picking $x = x(v_i', v_{-i})$ gives $f = \epsilon$; picking $x = x(v_i, v_{-i})$ gives $f = 0$, and any other x gives $f = -\epsilon$. Therefore, we can conclude that

$$x(v_i'', v_{-i}) = x(v_i', v_{-i}). \tag{10.3}$$

Substituting Equation (10.3) into Equation (10.2), we get

$$v_i''(x(v_i', v_{-i})) - \wp_i(v_i'', v_{-i}) \geq v_i''(x(v_i, v_{-i})) - \wp_i(v_i, v_{-i}). \tag{10.4}$$

Expand Equation (10.4):

$$\left(-\sum_{j \neq i} v_j(x(v_i', v_{-i})) + \epsilon \right) - \left(h(v_i'', v_{-i}) - \sum_{j \neq i} v_j(x(v_i'', v_{-i})) \right)$$
$$\geq \left(-\sum_{j \neq i} v_j(x(v_i, v_{-i})) \right) - \left(h(v_i, v_{-i}) - \sum_{j \neq i} v_j(x(v_i, v_{-i})) \right). \tag{10.5}$$

We can use Equation (10.3) to replace $x(v_i'', v_{-i})$ by $x(v_i', v_{-i})$ on the left-hand side of Equation (10.5). The sums then cancel out, and the inequality simplifies to

$$h(v_i, v_{-i}) \geq h(v_i'', v_{-i}) - \epsilon. \tag{10.6}$$

Since $x(v_i'', v_{-i}) = x(v_i', v_{-i})$, by the argument from Case 1 we can show that

$$h(v_i'', v_{-i}) = h(v_i', v_{-i}). \tag{10.7}$$

Substituting Equation (10.7) into Equation (10.6), we get

$$h(v_i, v_{-i}) \geq h(v_i', v_{-i}) - \epsilon.$$

This contradicts our assumption that $h(x(v_i, v_{-i})) < h(x(v_i', v_{-i})) - \epsilon$. We have thus shown that there cannot exist v_i, v_i' such that $h(v_i, v_{-i}) \neq h(v_i', v_{-i})$. ■

Although we do not give the proof here, it has also been shown that Groves mechanisms are unique among Bayes–Nash incentive compatible efficient mechanisms, in a weaker sense. Specifically, any Bayes–Nash incentive compatible efficient mechanism corresponds to a Groves mechanism in the sense that each agent makes the same *ex interim* expected payments and hence has the same *ex interim* expected utility under both mechanisms.

10.4.2 *The VCG mechanism*

So far, we have said nothing about how to set the function h_i in a Groves mechanism's payment function. Here we will discuss the most popular answer, which is called the Clarke tax. In the subsequent sections we will discuss some of its properties, but first we define it.

Clarke tax

Definition 10.4.4 (Clarke tax) *The* Clarke tax *sets the h_i term in a Groves mechanism as*

$$h_i(\hat{v}_{-i}) = \sum_{j \neq i} \hat{v}_j \left(x(\hat{v}_{-i})\right),$$

where x is the Groves mechanism allocation function.

The resulting Groves mechanism goes by many names. We will see in Chapter 11 that the Vickrey auction (invented in 1961) is a special case; thus, in resource allocation settings the mechanism is sometimes known as the *generalized Vickrey auction*. Second, the mechanism is also known as the *pivot mechanism*; we will explain the rationale behind this name in a moment. From now on, though, we will refer to it as the *Vickrey–Clarke–Groves mechanism* (VCG), naming its contributors in chronological order of their contributions. We restate the full mechanism here.

Vickrey–Clarke–Groves (VCG) mechanism

Definition 10.4.5 (Vickrey–Clarke–Groves (VCG) mechanism) *The* VCG mechanism *is a direct quasilinear mechanism $(x, \wp)$, where*

$$x(\hat{v}) = \arg\max_x \sum_i \hat{v}_i(x),$$

$$\wp_i(\hat{v}) = \sum_{j \neq i} \hat{v}_j \left(x(\hat{v}_{-i})\right) - \sum_{j \neq i} \hat{v}_j(x(\hat{v})).$$

First, note that because the Clarke tax does not depend on an agent i's own declaration $\hat{v}_i$, our previous arguments that Groves mechanisms are dominant-strategy truthful and efficient, carry over immediately to the VCG mechanism. Now, we try to provide some intuition about the VCG payment rule. Assume that

all agents follow their dominant strategies and declare their valuations truthfully. The second sum in the VCG payment rule pays each agent i the sum of every other agent $j \neq i$'s utility for the mechanism's choice. The first sum charges each agent i the sum of every other agent's utility for the choice that *would have been made* had i not participated in the mechanism. Thus, each agent is made to pay his *social cost*—the aggregate impact that his participation has on other agents' utilities.

What can we say about the amounts of different agents' payments to the mechanism? If some agent i does not change the mechanism's choice by his participation—that is, if $x(v) = x(v_{-i})$—then the two sums in the VCG payment function will cancel out. The social cost of i's participation is zero, and so he has to pay nothing. In order for an agent i to be made to pay a nonzero amount, he must be *pivotal* in the sense that the mechanism's choice $x(v)$ is different from its choice without i, $x(v_{-i})$. This is why VCG is sometimes called the pivot mechanism—only pivotal agents are made to pay. Of course, it is possible that some agents will *improve* other agents' utilities by participating; such agents will be made to pay a negative amount, or in other words will be paid by the mechanism.

Let us see an example of how the VCG mechanism works. Recall that Section 10.1.2 discussed the problem of buying a shortest path in a transportation network. We will now reconsider that example, and determine what route and what payments the VCG mechanism would select. For convenience, we reproduce Figure 10.1 as Figure 10.4, and label the nodes so that we have names to refer to the agents (the edges).

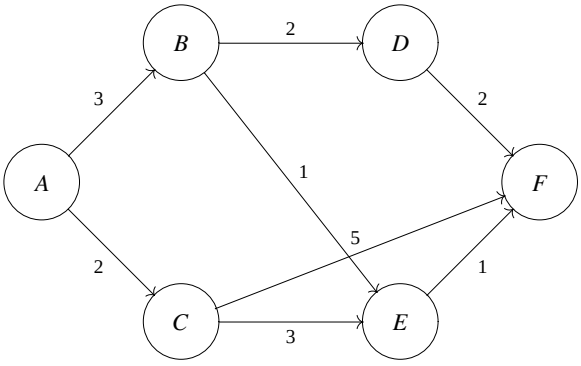


Figure 10.4 Transportation network with selfish agents.

Note that in this example, the numbers labeling the edges in the graph denote agents' costs rather than utilities; thus, an agent's utility is $-c$ if a route involving his edge (having cost c) is selected, and zero otherwise. The arg max in x will amount to cost minimization. Thus, $x(v)$ will return the shortest path in the graph, which is $ABEF$.

How much will agents have to pay? First, let us consider the agent AC. The shortest path taking his declaration into account has a length of 5 and imposes a cost of -5 on agents other than him (because it does not involve him). Likewise, the shortest path without AC's declaration also has a length of 5. Thus, his

payment is $p_{AC} = (-5) - (-5) = 0$. This is what we expect, since AC is not pivotal. Clearly, by the same argument BD, CE, CF, and DF will all be made to pay zero.

Now let us consider the pivotal agents. The shortest path taking AB's declaration into account has a length of 5, and imposes a cost of 2 on other agents. The shortest path without AB is $ACEF$, which has a cost of 6. Thus $p_{AB} = (-6) - (-2) = -4$: AB is paid 4 for his participation. Arguing similarly, you can verify that $p_{BE} = (-6) - (-4) = -2$, and $p_{EF} = (-7) - (-4) = -3$. Note that although EF had the same cost as BE, they are paid different amounts for the use of their edges. This occurs because EF has more *market power*: for the other agents, the situation without EF is worse than the situation without BE.

10.4.3 *VCG and individual rationality*

We have seen that Groves mechanisms are dominant-strategy truthful and efficient. We have also seen that no other mechanism has both of these properties in general quasilinear settings. Thus, we might be a bit worried that we have not been able to guarantee either individual rationality or budget balance, two properties that are quite important in practice. (Recall that individual rationality means that no agent would prefer not to participate in the mechanism; budget balance means that the mechanism does not lose money.) We will consider budget balance in Section 10.4.6; here we investigate individual rationality.

As it turns out, our worry is well founded: even with the freedom to set h_i, we cannot find a mechanism that guarantees us individual rationality in an unrestricted quasilinear setting. However, we are often able to guarantee the strongest variety of individual rationality when the setting satisfies certain mild restrictions.

choice-set monotonicity

Definition 10.4.6 (Choice-set monotonicity) *An environment exhibits* choice-set monotonicity *if* $\forall i,\ X_{-i} \subseteq X$ *(removing any agent weakly decreases—that is, never increases—the mechanism's set of possible choices X).*

no negative externalities

Definition 10.4.7 (No negative externalities) *An environment exhibits* no negative externalities *if* $\forall i \forall x \in X_{-i},\ v_i(x) \geq 0$ *(every agent has zero or positive utility for any choice that can be made without his participation).*

These assumptions are often quite reasonable, as we illustrate with two examples. First, consider running VCG to decide whether or not to undertake a public project such as building a road. In this case, the set of choices is independent of the number of agents, satisfying choice-set monotonicity. No agent negatively values the project, though some might value the situation in which the project is not undertaken more highly than the situation in which it is.

Second, consider a market consisting of a set of agents interested in buying a single unit of a good such as a share of stock and another set of agents interested in selling a single unit of this good. The choices in this environment are sets of buyer–seller pairings. (Prices are imposed through the payment function.) If a new agent is introduced into the market, no previously existing pairings become infeasible, but new ones become possible; thus choice-set monotonicity

is satisfied. Because agents have zero utility both for choices that involve trades between other agents and no trades at all, there are no negative externalities.

Under these restrictions, it turns out that the VCG mechanism ensures *ex post* individual rationality.

Theorem 10.4.8 *The VCG mechanism is* ex post *individually rational when the choice-set monotonicity and no negative externalities properties hold.*

Proof. All agents truthfully declare their valuations in equilibrium. Then we can write agent i's utility as

$$\begin{aligned} u_i &= v_i(x(v)) - \left(\sum_{j \neq i} v_j(x(v_{-i})) - \sum_{j \neq i} v_j(x(v)) \right) \\ &= \sum_j v_j(x(v)) - \sum_{j \neq i} v_j(x(v_{-i})). \end{aligned} \tag{10.8}$$

We know that $x(v)$ is the choice that maximizes social welfare, and that this optimization could have picked $x(v_{-i})$ instead (by choice-set monotonicity). Thus,

$$\sum_j v_j(x(v)) \geq \sum_j v_j(x(v_{-i})).$$

Furthermore, from no negative externalities,

$$v_i(x(v_{-i})) \geq 0.$$

Therefore,

$$\sum_i v_i(x(v)) \geq \sum_{j \neq i} v_j(x(v_{-i})),$$

and thus Equation (10.8) is nonnegative. ■

10.4.4 *VCG and weak budget balance*

What about weak budget balance, the requirement that the mechanism will not lose money? Our two previous conditions, choice-set monotonicity and no negative externalities, are not sufficient to guarantee weak budget balance: for example, the "buying the shortest path" example given earlier satisfied these two conditions, but we saw that the VCG mechanism paid out money and did not collect any. Thus, we will have to explore further restrictions to the quasilinear setting.

no single-agent effect

Definition 10.4.9 (No single-agent effect) *An environment exhibits* no single-agent effect *if* $\forall i, \forall v_{-i}, \forall x \in \arg\max_y \sum_j v_j(y)$ *there exists a choice* x' *that is feasible without* i *and that has* $\sum_{j \neq i} v_j(x') \geq \sum_{j \neq i} v_j(x)$.

In other words, removing any agent does not worsen the total value of the best solution to the others, regardless of their valuations. For example, this property is satisfied in a single-sided auction—dropping an agent just reduces the amount of competition in the auction, making the others better off.

Theorem 10.4.10 *The VCG mechanism is weakly budget balanced when the no single-agent effect property holds.*

Proof. As before, we start by assuming truth telling in equilibrium. We must show that the sum of transfers from agents to the center is greater than or equal to zero.

$$\sum_i \wp_i(v) = \sum_i \left(\sum_{j \neq i} v_j(x(v_{-i})) - \sum_{j \neq i} v_j(x(v)) \right)$$

From the no single-agent effect condition we have that

$$\forall i \ \sum_{j \neq i} v_j(x(v_{-i})) \geq \sum_{j \neq i} v_j(x(v)).$$

Thus the result follows directly. ■

Indeed, we can say something more about VCG's revenue properties: restricting ourselves to settings in which VCG is *ex post* individually rational as discussed earlier, and comparing to all other efficient and *ex interim* IR mechanisms, VCG turns out to collect the maximal amount of revenue from the agents. This is somewhat surprising, since this result does not require dominant strategies, and hence compares VCG to all Bayes–Nash mechanisms. A useful corollary of this result is that VCG is as budget balanced as any efficient mechanism can be: it satisfies weak budget balance in every case where *any* dominant strategy, efficient and *ex interim* IR mechanism would be able to do so.

10.4.5 *Drawbacks of VCG*

The VCG mechanism is one of the most powerful positive results in mechanism design: it gives us a general way of constructing dominant-strategy truthful mechanisms to implement social-welfare-maximizing social choice functions in quasilinear settings. We have seen that no fundamentally different mechanism could do the same job. And VCG gives us even more: under the right conditions it further guarantees *ex post* individual rationality and weak budget balance. Thus, it is not surprising that this mechanism has been enormously influential and continues to be widely studied.

However, despite these attractive properties, VCG also has some undesirable characteristics. In this section, we survey six of them. Before we go on, however, we offer a caveat: although there exist mechanisms that circumvent each of the drawbacks we discuss, none of the drawbacks are *unique* to VCG, or even to Groves mechanisms. Indeed, in some cases the problems are known to crop up in extremely broad classes of mechanisms; we cite some arguments to this effect at the end of the chapter.

1. Agents must fully disclose private information

VCG requires agents to fully reveal their private information (e.g., in the transportation network example, every agent has to tell the mechanism his costs

Agent	U (build road)	U (do not build road)	Payment
1	200	0	150
2	100	0	50
3	0	250	0

Table 10.1 Valuations for agents in the road-building referendum example.

Agent	U (build road)	U (do not build road)	Payment
1	250	0	100
2	150	0	0
3	0	250	0

Table 10.2 Agents in the road-building referendum can gain by colluding.

exactly). In some real-world domains, this private information may have value to agents that extends beyond the current interaction—for example, the agents may know that they will compete with each other again in the future. In such settings, it is often preferable to elicit only as much information from agents as is required to determine the social welfare maximizing choice and compute the VCG payments. We discuss this issue further when we come to ascending auctions in Chapter 11.

2. Susceptibility to collusion

Consider a referendum setting in which three agents use the VCG mechanism to decide between two choices. For example, this mechanism could be useful in the road-building referendum setting discussed earlier. Table 10.1 shows a set of valuations and the VCG payments that each agent would be required to make.

We know from Theorem 10.4.2 that no agent can gain by changing his declaration. However, the same cannot be said about groups of agents. It turns out that groups of colluding agents can achieve higher utility by coordinating their declarations rather than honestly reporting their valuations. For example, Table 10.2 shows that agents 1 and 2 can reduce both of their payments without changing the mechanism's decision by both increasing their declared valuations by $50.

3. VCG is not frugal

Consider again the transportation network example that we worked through in Section 10.4.2. We saw that the shortest path has a length of 5, the second shortest disjoint path has a length of 7, and VCG ends up paying 9. Can we give a bound on how much more than the agents' costs VCG can pay? Loosely speaking, mechanisms whose payments are small by such a measure are called *frugal*.

frugal mechanism

Before deciding whether VCG is frugal, we must determine what kind of bound to look for. We might want VCG to pay an amount similar to the agents'

Agent	U (build road)	U (do not build road)	Payment
1	0	90	0
2	100	0	90

Table 10.3 Valuations for agents in the road-building referendum example.

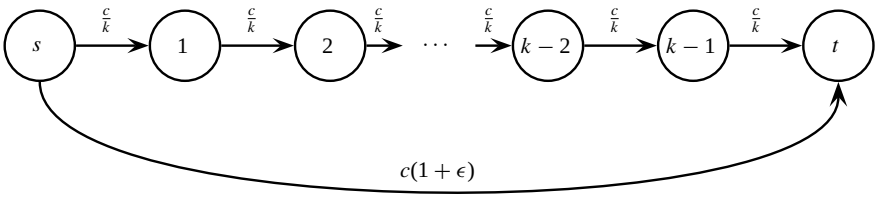


Figure 10.5 A transportation network example for which VCG's payments are not even close to the cost of the second disjoint path.

true costs. However, even in the simplest possible network it is easy to see that this is not possible. Consider a graph where there are only two paths, each owned by a single agent. In this case VCG selects the shortest path and pays the cost of the longer path, no matter how much longer it is.

It might seem more promising to hope that VCG's payments would be at least close to the cost of the *second* shortest disjoint path. Indeed, in our two-agent example this is always exactly what VCG pays. However, now consider a different graph that has two paths as illustrated in Figure 10.5. The top path involves k agents, each of whom has a cost of $\frac{c}{k}$; thus, the path has a total cost of c. The lower path involves a single agent with a cost of $c(1+\epsilon)$. VCG would select the path with cost c, and pay each of the k agents $c(1+\epsilon) - (k-1)\frac{c}{k}$. Hence VCG's total payment would be $c(1+k\epsilon)$. For fixed ϵ, this means that VCG's payment is $\Theta(k)$ times the cost of the second shortest disjoint path. Thus VCG is said not to be a frugal mechanism.

4. Dropping bidders can increase revenue

revenue monotonicity

Now we will consider *revenue monotonicity*: the property that a mechanism's revenue always weakly increases as agents are added to the mechanism. Although it may seem intuitive that having more agents should never mean less revenue, in fact VCG does not satisfy this property. To see why, let us return to the road-building example.

Consider the new valuations given in Table 10.3. Observe that the social-welfare-maximizing choice is to build the road. Agent 2 is pivotal and so would be made to pay 90, his social cost. Now see what happens when we add a third agent, as shown in Table 10.4. Again, VCG would decide that the road should be built. However, since in this second case the choice does not change when *either* winning agent is dropped, neither of them is made to pay anything, and

Agent	U (build road)	U (do not build road)	Payment
1	0	90	0
2	100	0	0
3	100	0	0

Table 10.4 Adding agent 3 causes VCG to select the same choice but to collect zero revenue.

so the mechanism collects zero revenue. Observe that the road-building referendum problem satisfies the "no single-agent effect" property; thus revenue monotonicity can fail even when the mechanism is guaranteed to be weakly budget balanced.

The fact that VCG is not revenue monotonic can also be understood as a strategic opportunity for agent 2, in the setting where agent 3 does not exist. Specifically, agent 2 can reduce his payment to zero if he is able to participate in the mechanism under multiple identities, submitting valuations both as himself and as agent 3. (This assumption might be reasonable, for example, if the mechanism is executed over the Internet.) Note that this strategy is not without its risks, however: for example, if agent 1's valuation were 150, both of agent 2's identities would be pivotal and so agent 2 would end up paying more than his true valuation.

5. Cannot return all revenue to the agents

In a setting such as this road-building example, we may want to use VCG to induce agents to report their valuations honestly, but may not want to make a profit by collecting money from the agents. In our example this might be true if the referendum was held by a government interested only in maximizing social welfare. Thus, we would want to find some way of returning the mechanism's profits back to the agents—that is, we would want a (strictly) budget-balanced mechanism rather than a weakly budget-balanced one. This turns out to be surprisingly hard to achieve, even when the "no single-agent effect" property holds, because the possibility of receiving a rebate after the mechanism has been run changes the agents' incentives. In fact, even if profits are given to a charity that the agents care about, or spent in a way that benefits the local economy and hence benefits the agents, the VCG mechanism can be undermined. This having been said, it *is* possible to return at least some of the revenues to the agents, although this must be done carefully. We give pointers to the relevant literature at the end of the chapter.

6. Computational intractability

Finally, even when there are no problems *in principle* with using VCG, there can still be practical obstacles. Perhaps the biggest such problem is that efficient mechanisms can require unreasonable amounts of computation: evaluating the arg max can require solving an NP-hard problem in many practical domains.

Thus, VCG can fail to satisfy the *tractability* property (Definition 10.3.10). This problem is not just theoretical: we present important examples in which VCG is intractable in Sections 11.2.3 and 11.3.2. In Section 10.5 below, we consider some alternatives to VCG for use in such settings.

10.4.6 *Budget balance and efficiency*

In Section 10.4.4 we identified a realistic case in which the VCG mechanism is weakly budget balanced. However, we also noted that there exist other important and practical settings in which the no single-agent effect property does not hold.

simple exchange

For example, define a *simple exchange* as an environment consisting of buyers and sellers with quasilinear utility functions, all interested in trading a single identical unit of some good. The no single-agent effect property is not satisfied in a simple exchange because dropping a seller could make some buyer worse off and vice versa. Can we find some other argument to show that VCG will remain budget balanced in this important setting?

It turns out that neither VCG nor any other Groves mechanism is budget balanced in the simple exchange setting. (Recall Theorem 10.4.3, which showed that only Groves mechanisms are both dominant-strategy incentive-compatible and efficient.)

Theorem 10.4.11 (Green–Laffont; Hurwicz) *No dominant-strategy incentive-compatible mechanism is always both efficient and weakly budget balanced, even if agents are restricted to the simple exchange setting.*

Furthermore, another seminal result showed that a similar problem arises in the broader class of Bayes–Nash incentive-compatible mechanisms (which, recall, includes the class of dominant-strategy incentive-compatible mechanisms) if we also require *ex interim* individual rationality and allow general quasilinear utility functions.

Theorem 10.4.12 (Myerson–Satterthwaite) *No Bayes–Nash incentive-compatible mechanism is always simultaneously efficient, weakly budget balanced, and* ex interim *individually rational, even if agents are restricted to quasilinear utility functions.*

On the other hand, it turns out that it *is* possible to design a Bayes–Nash incentive compatible mechanism that achieves any two of these three properties.

10.4.7 *The AGV mechanism*

Of particular interest is the AGV mechanism, which trades away *ex interim* individual rationality and dominant strategies in exchange for budget balance and *ex ante* individual rationality.

Arrow; d'Aspremont–Gérard-Varet (AGV) mechanism

Definition 10.4.13 (Arrow; d'Aspremont–Gérard-Varet (AGV) mechanism) *The* Arrow; d'Aspremont–Gérard-Varet mechanism *(AGV) is a direct quasilinear mechanism* $(\chi, \wp)$*, where*

$$\chi(\hat{v}) = \arg\max_{x} \sum_{i} \hat{v}_i(x),$$

$$\wp_i(\hat{v}) = \left(\frac{1}{n-1} \sum_{j \neq i} ESW_{-j}(\hat{v}_j) \right) - ESW_{-i}(\hat{v}_i),$$

$$ESW_{-i}(\hat{v}_i) = \mathbb{E}_{v_{-i}} \left[\sum_{j \neq i} v_j \left(\chi(\hat{v}_i, v_{-i}) \right) \right].$$

ESW (standing for "expected social welfare") is an intermediate term that is used to make the definition of $\wp$ more concise. Observe that AGV's allocation rule is the same as under Groves mechanisms. Although we will not prove this or any of the other properties we mention here, AGV is incentive compatible, from which we can conclude that it is also efficient. Again like Groves mechanisms, each agent i is given a payment reflecting the other agents' valuations for the choice selected given his declaration. While in Groves mechanisms this calculation used $-i$'s declared valuations, however, AGV computes $-i$'s *ex ante* expected social welfare given i's declaration. The rest of the payment is computed very differently than it is under VCG: each agent is charged a $\frac{1}{n-1}$ share of the payments made to each of the other agents. This guarantees that the mechanism is budget balanced (i.e., that it always collects from the agents exactly the total amount that it pays back to them). Two sacrifices are made in exchange for this property: AGV is truthful only in Bayes–Nash equilibrium rather than in dominant strategies and is only *ex ante* individually rational.

The AGV mechanism illustrates two senses in which we can discover new, useful mechanisms by relaxing properties that we had previously insisted on. First, our move from dominant-strategy incentive compatibility to Bayes–Nash incentive compatibility allowed us to circumvent Theorem 10.4.3, which told us that efficiency can be achieved under dominant strategies only by Groves mechanisms. (AGV is also efficient, but is not a Groves mechanism.) Second, moving from *ex interim* to *ex ante* individual rationality is sufficient to get around the negative result from Theorem 10.4.12, that we cannot simultaneously achieve weak budget balance, efficiency, and *ex interim* individual rationality, even under Bayes–Nash equilibrium.

10.5 Beyond efficiency

Throughout our consideration of the quasilinear setting in this chapter we have so far focused on efficient mechanisms. As we discussed in Section 10.4.5, efficient choice rules can require unreasonable amounts of computation, and hence both Groves mechanisms and AGV can fail to satisfy the *tractability* property. In this

section we consider two ways of addressing this issue. The first is to explore dominant-strategy mechanisms that implement different social choice functions. We have already seen that the quasilinear preference setting is considerably more amenable to dominant strategy implementation than the unrestricted preferences setting. However, there are still limits—what are they? Second, we will examine an alternate way of building mechanisms, by using a Groves payment rule with an alternate choice function, and leveraging agents' computational limitations in order to achieve the implementation.

10.5.1 *What else can be implemented in dominant strategies?*

Here we give some characterizations of the social choice functions that can be implemented in dominant strategies in the quasilinear setting and of how payments must be constrained in order to enable such implementations. As always, the revelation principle allows us to restrict ourselves to truthful mechanisms without loss of generality. We also restrict ourselves to *deterministic* mechanisms: this restriction does turn out to be substantive.

Let $X_i(\hat{v}_{-i}) \subseteq X$ denote the set of choices that can be selected by the choice rule x given the declaration $\hat{v}_{-i}$ by the agents other than i (i.e., the range of $x(\cdot, \hat{v}_{-i})$). Now we can state conditions that are both necessary and sufficient for dominant-strategy truthfulness that are both intuitive and useful.

Theorem 10.5.1 *A direct, deterministic mechanism is dominant-strategy incentive-compatible if and only if, for every $i \in N$ and every $\hat{v}_{-i} \in V_{-i}$:*

1. *the payment function $\wp_i(\hat{v})$ can be written as $\wp_i(\hat{v}_{-i}, x(\hat{v}))$; and*
2. *for every $\hat{v}_i \in V_i$, $x(\hat{v}_i, \hat{v}_{-i}) \in \arg\max_{x \in X_i(\hat{v}_{-i})}(\hat{v}_i(x) - \wp_i(\hat{v}_{-i}, x))$.*

The first condition says that an agent's payment can only depend on other agents' declarations and the selected choice; it *cannot* depend otherwise on the agent's own declaration. The second condition says that, taking the other agent's declarations and the payment function into account, from every player's point of view the mechanism selects the most preferable choice. This result is not very difficult to prove; the interested reader is encouraged to try.

As the above characterization suggests, there is a tight connection between the choice rules and payment rules of dominant-strategy truthful mechanisms. In fact, under reasonable assumptions about the valuation space, once a choice rule is chosen, all possible payment rules differ only in their choice of a function $h_i(\hat{v}_{-i})$ that is added to the rest of the payment. We already saw an example of this with the Groves family of mechanisms: these mechanisms share the same choice rule, and their payment rules differ only in a constant $h_i(\hat{v}_{-i})$.

Given this strong link between choice rules and payment rules, it is interesting to characterize a set of *choice rules* that can be implemented in dominant strategies, without reference to payments. Here we will consider such a characterization, though in general it turns out only to offer a *necessary* condition for dominant-strategy truthfulness.

weak monotonicity (WMON)

Definition 10.5.2 (WMON) *A social choice function C satisfies* weak monotonicity *(WMON) if for all $i \in N$ and all $v_{-i} \in V_{-i}$, $C(v_i, v_{-i}) \neq C(v_i', v_{-i})$ implies that $v_i(C(v_i, v_{-i})) - v_i(C(v_i', v_{-i})) \geq v_i'(C(v_i, v_{-i})) - v_i'(C(v_i', v_{-i}))$.*

In words, WMON says that any time the choice function's decision can be altered by a single agent changing his declaration, it must be the case that this change expressed a relative increase in preference for the new choice over the old choice.

Theorem 10.5.3 *All social choice functions implementable by deterministic dominant-strategy incentive-compatible mechanisms in quasilinear settings satisfy WMON. Furthermore, let C be an arbitrary social choice function $C : V_1 \times \cdots \times V_n \mapsto X$ satisfying WMON and having the property that $\forall i \in N$, V_i is a convex set. Then C can be implemented in dominant strategies.*

Although Theorem 10.5.3 does not provide a full characterization of those social choice functions that can be implemented in dominant strategies, it gets pretty close—the convexity restriction is often acceptable. A bigger problem is that WMON is a *local* characterization, speaking about how the mechanism treats each agent individually. It would be desirable to have a global characterization that gave the social choice function directly. This also turns out to be possible.

affine maximizer

Definition 10.5.4 (Affine maximizer) *A social choice function is an* affine maximizer *if it has the form*

$$\arg\max_{x \in X} \left(\gamma_x + \sum_{i \in N} w_i v_i(x) \right),$$

where each γ_x is an arbitrary constant (may be $-\infty$) and each $w_i \in \mathbb{R}_+$.

In the case of general quasilinear preferences (i.e., when each agent can have any valuation for each choice $x \in X$) and where the choice function selects from more than two alternatives, affine maximizers turn out to be the only social choice functions implementable in dominant strategies.

Theorem 10.5.5 (Roberts) *If there are at least three choices that a social choice function will select given some input, and if agents have general quasilinear preferences, then the set of (deterministic) social choice functions implementable in dominant strategies is precisely the set of affine maximizers.*

Note that efficiency is an affine-maximizing social choice function for which $\forall x \in X, \gamma_x = 0$ and $\forall i \in N, w_i = 1$. Indeed, affine maximizing mechanisms can be seen as weighted Groves mechanisms—they transform both the choices and the agents' valuations by applying linear weights, and then effectively run a Groves mechanism in the transformed space. Thus, Theorem 10.5.5 says that we cannot stray very far from Groves mechanisms even if we are willing to give up on efficiency.

Is this the end of the story on dominant-strategy implementation in quasilinear settings? Not quite. It turns out that the assumption that agents have general quasilinear preferences is a strong one, and does not hold in many domains of

single-parameter valuation

interest. As another extreme, we can consider *single-parameter valuations*: each agent i partitions the set of choices X into a set $X_{i,wins}$ in which i "wins" and receives some constant payoff v_i that does not depend on the choice $x \in X_{i,wins}$, and a set of choices $X_{i,loses} = X \setminus X_{i,wins}$ in which i "loses" and receives zero payoff.[10] Importantly, the sets $X_{i,wins}$ and $X_{i,loses}$ are assumed to be common knowledge, and so the agent's private information can be summarized by a single parameter, v_i. Such settings are quite practical: we will see several that satisfy these conditions in Chapter 11. Single-parameter settings are interesting because for such preferences, it *is* possible to go well beyond affine maximizers. In fact, additional characterizations exist describing the social choice functions that can be implemented in this and other restricted-preference settings. We will not describe them here, instead referring interested readers to the works cited at the end of the chapter. However, we do present a dominant-strategy incentive-compatible, non-affine-maximizing mechanism for a single-parameter setting in Section 11.3.5.

10.5.2 *Tractable Groves mechanisms*

Now we consider a general approach that attempts to implement tractable, inefficient social choice functions by sticking with Groves mechanisms, but replacing the (possibly exponential-time) computation of the arg max with some other polynomial-time algorithm. The very clever idea here is not to build mechanisms that are impossible to manipulate (indeed, in many cases it can be shown that this cannot be done), but rather to build mechanisms that agents will be unable to manipulate in practice, given their computational limitations.

First, we define the class of mechanisms being considered.

Groves-based mechanism

Definition 10.5.6 (Groves-based mechanisms) Groves-based mechanisms *are direct quasilinear mechanisms* $(x, \wp)$*, for which:*

$x(\hat{v})$ *is an arbitrary function mapping type declarations to choices; and*

$$\wp_i(\hat{v}) = h_i(\hat{v}_{-i}) - \sum_{j \neq i} \hat{v}_j(x(\hat{v})).$$

That is, a mechanism is Groves based if it uses the Groves payment function, regardless of what allocation function it uses. (Contrast Definition 10.5.6 with Definition 10.4.1, which defined a Groves mechanism.)

Most interesting Groves-based mechanisms are not dominant-strategy truthful. For example, consider a property sometimes called *reasonableness*: if there exists some good g that only one agent i values above zero, g should be allocated to i. It can be shown that the only dominant-strategy truthful Groves-based mechanisms that satisfy reasonableness are the Groves mechanisms themselves. This rules out the use of most greedy algorithms as candidates for the allocation function x in truthful Groves-based mechanisms, as most such algorithms would select "reasonable" allocations.

10. The assumption that this second payoff is zero can be understood as a normalization and does not change the set of social choice functions that can be implemented.

If tractable Groves-based mechanisms lose the all-important property of dominant-strategy truthfulness, why are they still interesting? The proof of Theorem 10.4.2 essentially argued that the Groves payment function aligns agents' utilities, making all agents prefer the optimal allocation. Groves-based mechanisms still have this property, but may not select the optimal allocation. We can conclude that the only way an agent can gain by lying to a Groves-based mechanism is to *help it* by causing it to select a more efficient allocation.

We now come to the idea of a second-chance mechanism. Intuitively, since lies by agents can only help the mechanism, the mechanism can simply ask the agents how they intend to lie and select an choice that would be picked because of such a lie if it turns out to be better than what the mechanism would have picked otherwise.

second-chance mechanism

Definition 10.5.7 (Second-chance mechanisms) *Given a Groves-based mechanism* $(x, \wp)$, *a* second-chance mechanism *works as follows:*

1. *Each agent* i *is asked to submit a valuation declaration* $\hat{v}_i \in V_i$ *and an* appeal function $l : V \mapsto V$.
2. *The mechanism computes* $x(\hat{v})$, *and also* $x(l_i(\hat{v}))$ *for all* $i \in N$. *From the set of choices thus identified, the mechanism keeps one that maximizes the sum of agents' declared valuations.*
3. *The mechanism charges each agent* i $\wp(\hat{v})$.

appeal function

Intuitively, an appeal function maps agents' valuations to valuations that they might instead have chosen to report by lying. It is important that the appeal functions be computationally bounded (e.g., their execution could be time limited). Otherwise, these functions can solve the social welfare maximization problem and then select an input that would cause x to select this choice. When appeal functions are computationally restricted, we cannot in general say that second-chance mechanisms are truthful. However, they are *feasibly truthful*, because an agent can use the appeal function to try out any lie that he believes might help him. Thus in a second-chance mechanism, a computationally limited agent can do no better than to declare his true valuation along with the best appeal function he is able to construct.

feasibly truthful

10.6 Computational applications of mechanism design

We now survey some applications of mechanism design, to give a sense of some more recent work that has drawn on the theory we have described so far. However, we must offer two caveats. First, we speak here only about *computational* applications of mechanism design, by which we mean mechanisms that contain an interesting computational ingredient and/or mechanisms applied to computational domains (e.g., computer networks). Thus we skip over some highly influential applications from economics, such as theories of taxation, government regulation, and corporate finance. Second, without a doubt the most significant application of mechanism design—computational or not—is the design and analysis of auctions. Because there is so much to say about this application, we defer its discussion to Chapter 11.

algorithmic mechanism design

Some of the mechanism design applications we discuss in this section are examples of so-called *algorithmic mechanism design*. This term describes settings in which a center wants to solve an optimization problem, but the inputs to this problem are the private information of self-interested agents. The center must thus design a mechanism that solves the optimization while inducing the agents to reveal their information truthfully. Observe that this setting does not really describe a different problem from classical mechanism design, though it does adopt a different perspective. It also tends to describe work that has a somewhat different flavor, often emphasizing approximation algorithms and worst-case analysis.

10.6.1 *Task scheduling*

One problem that has been well studied in the context of algorithmic mechanism design is that of task scheduling. Consider n agents who can perform tasks and a set T tasks that must be allocated. Each agent i's type t_i is a vector, giving the minimum amount of time $t_{i,j}$ in which i can perform each task j. The center's goal is to minimize the completion time of the last task, called the *makespan*. A choice x by the mechanism is an allocation of each task to some agent; agents must perform the tasks they are assigned. Let $x(i, j)$ equal 1 if an agent i is assigned task j, and zero otherwise. Note that some agents may be given more than one task and some may not be given a task at all. The mechanism is able to *verify* the agents' work, observing the true amount of time it took an agent to complete his tasks. We write the true amount of time i spent on task j as $\tilde{t}_{i,j}$; of course $\tilde{t}_{i,j}$ must always be greater than or equal to $t_{i,j}$. An agent i's valuation for a choice x by the mechanism is $-\sum_{j \in T} x(i, j)\tilde{t}_{i,j}$, the sum of the true amounts of time he spends on his assigned tasks. Of course, an agent i can lie about the amount of time it will take him to perform a task. We denote the tuple of all agents' declarations as $\hat{t}$.

makespan

The task scheduling problem cannot be solved with a Groves mechanism. While such a mechanism would indeed provide agents with dominant strategies for truthfully revealing their types, it would choose the wrong allocation, maximizing the sum of agents' welfare rather than minimizing the makespan. Indeed, note that makespan is like a worst-case version of social welfare: it measures the unhappiness of the unhappiest agent, and ignores the other agents completely. Another family of mechanisms does work for solving the task allocation scheduling problem. These mechanisms can be understood as generalizing Groves mechanisms to objective functions other than social welfare.

compensation and penalty mechanism

Definition 10.6.1 (Compensation and penalty mechanisms) Compensation and penalty mechanisms *are quasilinear mechanisms* $(\chi, \wp)$*, for which*

$$\chi(\hat{t}) = \arg\min_{x} \left(\max_{i \in N} \sum_{j \in T} x(i, j)\hat{t}_{i,j} \right)$$

$$\wp_i(\hat{t}) = h_i(\hat{t}_{-i}) - \sum_{j \in T} x(i, j)\tilde{t}_{i,j} + \max\left\{ \sum_{j \in T} x(i, j)\tilde{t}_{i,j}, \max_{i' \neq i \in N} \sum_{j \in T} x(i', j)\hat{t}_{i',j} \right\}.$$

Thus, the mechanism selects the choice that minimizes makespan, given the agents' declarations. What types should agents declare? Should agents solve tasks as quickly as possible, or can they increase their utilities by taking longer? An answer is given by the following theorem.

Theorem 10.6.2 *Compensation and penalty mechanisms are dominant-strategy incentive compatible: agents choose to complete their tasks as quickly as possible ($\tilde{t}_{i,j} = t_{i,j}$) and to report these completion times truthfully ($\hat{t}_{i,j} = t_{i,j}$).*

Proof. The first term in the payment function $\wp_i$, $h_i(\hat{t}_{-i})$, does not depend on i's declaration. Thus it does not affect i's incentives, and so we can disregard it.

The rest of $\wp_i$ consists of two terms. The second term is a payment to agent i equal to his true cost for his assigned tasks. This payment exactly *compensates* i for any tasks he was assigned, making him indifferent between all task assignments regardless of how long he spent completing his tasks.

The third term of $\wp_i$ is a *penalty* to i in the amount of the mechanism's objective function, except that i's actual task completion time is used instead of his declared time. The strategic problem for i is thus to choose the $\tilde{t}$ and $\hat{t}$ that will lead the mechanism to select the x that makes this penalty as small as possible. By choosing $\tilde{t}_{i,j} > t_{i,j}$, i does not influence x (this depends only on $\hat{t}_{i,j}$) and can only increase his penalty. $\tilde{t}_{i,j} < t_{i,j}$ is impossible, and so it is a dominant strategy for i to choose $\tilde{t}_{i,j} = t_{i,j}$. If i declares $\hat{t}_{i,j} > t_{i,j}$, then he can only increase the makespan and hence his penalty, by making the mechanism allocate tasks suboptimally to the other agents. If i declares $\hat{t}_{i,j} < t_{i,j}$, he can reduce the makespan; however, he cannot reduce his penalty since it depends on $\tilde{t}_{i,j}$ rather than $\hat{t}_{i,j}$. In this case he still can *increase* his penalty by causing the mechanism to allocate tasks suboptimally. Thus, i's dominant strategy is to declare $\hat{t}_{i,j} = t_{i,j}$. ■

Observe that it is important that the mechanism can verify the amount of time an agent took to complete the task. If this were not the case, the agent could under-report his completion time, driving down the makespan and hence reducing his own penalty. Note also that these mechanisms really do generalize Groves mechanisms: if we replace the mechanism's objective function (in x and the third term of $\wp$) with social welfare, we recover Groves.

While compensation and penalty mechanisms are truthful even with $h_i = 0$, they are not individually rational, as an agent's utility is always simply the negative of his penalty, and this penalty is always positive. However, we can regain individual rationality in the same way as we did in moving from Groves mechanisms to VCG. Specifically, we can set h_i to be the mechanism's objective function when i does not participate,

$$h_i(\hat{t}_{-i}) = \min_x \left(\max_{i' \neq i \in N} \sum_{j \in T} x(i', j) \hat{t}_{i',j} \right).$$

Now h_i will always be greater than or equal to i's penalty, because the makespan is guaranteed to weakly increase if we omit i. This ensures that i never loses by participating in the mechanism.

As we indicated at the beginning of the section, work on algorithmic mechanism design often focuses on the use of approximation algorithms. Such an approach is sensible for the task scheduling problem because finding the makespan-minimizing allocation ($x(\hat{t})$ in compensation and penalty mechanisms) is an NP-hard problem, whereas approximation algorithms can run in polynomial time. Although we do not go into the details here, there is a whole constellation of results about what approximation bounds are achievable by which variety of dominant-strategy approximation-algorithm-based mechanism, under what assumptions (e.g., verification possible or not; restrictions on valuations). For example, in the case without verification no deterministic mechanism based on an approximation algorithm can achieve better than a 2-approximation; this bound is tight for the 2 agent case. On the other hand, randomized mechanisms can do better, achieving a 1.75-approximation. More details are available in the paper cited at the end of the chapter.

10.6.2 *Bandwidth allocation in computer networks*

When designing a computer network, the network operator wants to ensure that the most important network traffic gets the best performance and/or that some fairness criterion is satisfied. However, optimizing traffic in this way is difficult because the network operator does not know the users' tasks or how important they are. Thus, the network operator faces a mechanism design problem. Although much more elaborate settings have been studied (see the notes at the end of the chapter), in this section we will consider the problem of allocating the capacity of a single link in a network. The reason that this problem is still tricky is that the bandwidth of a link is not allocated all-or-nothing to a single buyer, as it was in our example at the beginning of the chapter (Section 10.1.2). Instead, the link has a real-valued capacity that can be divided arbitrarily between the agents. Thus, even this simple problem considers a choice space X that is uncountably infinite, and valuation functions that can be understood as continuous "demand curves."

Formally, consider a domain with N users who want to use a network resource with capacity $C \in \mathbb{R}_+$. Each user has a valuation function $v_i : \mathbb{R}_+ \mapsto \mathbb{R}$ expressing his happiness for being allocated any nonnegative amount of capacity d_i. We will assume throughout that this function v_i is concave, strictly increasing, and continuous.[11]

proportional allocation mechanism

We will begin by considering a particular mechanism that has been widely studied: the *proportional allocation mechanism*. This is a quasilinear mechanism in which agents are invited to specify a single value $w_i \in \mathbb{R}_+$. The mechanism interprets each value w_i as the payment that user i offers to make to the network. In order to determine the amount of capacity that each user will receive, we start

11. Furthermore, it is necessary to make some differentiability assumptions about the valuation functions; for details see the references cited at the end of the chapter.

from the assumption that each user must be charged for his use of the resource at the same rate, μ. Assuming that the network operator wants to allocate all capacity, we can then calculate this rate uniquely as $\mu = \frac{\sum_i w_i}{C}$, implying that each agent i receives the allocation $d_i = \frac{w_i}{\mu}$.

Unlike most of the mechanisms discussed in this chapter, the proportional allocation mechanism is not direct. However, this is one of its attractive qualities. Even under our assumptions of concavity, continuity, and monotonicity, an agent's valuation function can be arbitrarily complex. In a real network system, it would defeat the purpose of an allocation mechanism to allow agents to communicate a great deal of information about their valuation functions—the whole idea is to allocate bandwidth efficiently. Since the proportional allocation mechanism requires each agent to declare only a single real number, its proponents have argued that it is practical and have even gone so far as to describe ways that it could be added to existing (e.g., TCP/IP) network architectures.

A more serious concern is that the proportional allocation mechanism appears strategically complex, since agents can affect their payments (rather than just their allocations) by changing their declarations. Nevertheless, there are a number of interesting things that we can say about the mechanism. First, let us set aside our usual game-theoretic assumption that agents play best responses to each other and to the rules of the mechanism. Instead, let us assume that agents are *price takers*: that they consider the rate μ to be fixed and that they select the best declarations w_i given μ. (In fact, an agent's declaration w_i is used in the calculation of μ; thus, we assume that agents disregard this connection.) Given this assumption, it is interesting to ask whether allocations chosen by our mechanism constitute a *competitive equilibrium* (Definition 2.3.4). Formally, a declaration profile w and rate μ constitute a competitive equilibrium if each w_i maximizes i's quasilinear valuation $v_i(\frac{w_i}{\mu}) - w_i$, and if $\mu = \frac{\sum_i w_i}{C}$. It is possible to prove the following result.

price taker

competitive equilibrium

Theorem 10.6.3 *Given n agents with valuation functions $(v_1, \ldots, v_n)$ and a resource with capacity $C > 0$, there exists a competitive equilibrium (w, μ) of the proportional allocation mechanism. Furthermore, the allocation is efficient: the choices $d_i = \frac{w_i}{\mu}$ maximize the social welfare $\sum_i v_i(d_i) - w_i$ subject to capacity constraints.*

Thus, given price-taking agents, full efficiency can be achieved by the proportional allocation mechanism, even though it only elicits a single scalar value from each agent.

Now, let us return to the more standard game-theoretic setting, in which agents take into account their abilities to affect μ through their own declarations. Thus, our solution concept shifts from the competitive equilibrium to the Nash equilibrium. It is possible to show that a Nash equilibrium exists[12] and that it is unique. How does this Nash equilibrium compare to the competitive equilibrium described in Theorem 10.6.3? The natural way to formalize this question is to ask what fraction of the social welfare achieved in the competitive equilibrium is also achieved in the Nash equilibrium. When we ask how small this fraction

12. In settings with continuous action spaces, the existence of Nash equilibrium is not guaranteed.

becomes in the worst case, we arrive precisely at the notion of minimizing the *price of anarchy* (see Definition 10.3.14; recall also our previous use of the price of anarchy in the context of "selfish routing" in Section 6.4.5).

price of anarchy

Theorem 10.6.4 *Let $n \geq 2$, let d^{CE} be an allocation profile achievable in competitive equilibrium and let d^{NE} be the unique allocation profile achievable in Nash equilibrium. Then any profile of valuation functions v for which $\forall i$, $v_i(0) \geq 0$ satisfies*

$$\sum_i v_i(d_i^{NE}) \geq \frac{3}{4} \sum_i v_i(d_i^{CE}).$$

In other words, the price of anarchy is $\frac{4}{3}$; in the worst case, the Nash equilibrium achieves 25% less efficiency than the competitive equilibrium. While it is always disappointing not to achieve full efficiency, this result should be understood as good news. Even in the worst case, strategic behavior by agents will only cause a small reduction in social welfare.

So far, we have analyzed a given mechanism rather than showing that this mechanism optimizes some objective function. However, the proportional allocation mechanism can indeed be justified in this way. Specifically, it achieves minimal price of anarchy, as compared to a broad family of mechanisms in which agents' declarations are a single scalar and the mechanism charges all users the same rate. We do not state this result formally here, as the precise definition of the family of mechanisms is quite technical; instead, we refer the reader to the references cited at the end of the chapter. We also note that when the setting is relaxed so that users still submit only a single scalar but the mechanism is allowed to charge different users at different rates, a VCG-like mechanism can be used to achieve full efficiency.

10.6.3 *Multicast cost sharing*

Consider the problem of streaming media (e.g., a television broadcast) over a digital network. If this information is sent naively (e.g., using the TCP/IP protocol), then each user establishes a separate connection with the server and the same information may be sent many times over the same network links. This approach can easily overwhelm a link's capacity. A more sensible alternative is *multicast routing*, in which information is sent only once across each link, and it is replicated onto multiple outgoing links where necessary. Besides saving bandwidth, this approach can also make more economic sense. For example, individual users sharing a satellite link might not be willing to pay the full cost of receiving a high-bandwidth video stream, but could be willing to split the cost among themselves. Such a system faces the problem of *multicast cost sharing*: given a set of users with different values for receiving the transmission and a network with costly links, who should receive the transmission and how much should they each pay? This is a mechanism design problem.

multicast routing

multicast cost sharing

Formally, consider an undirected graph with nodes N (a set of agents) and links L. Each link $l \in L$ has a cost $c(l) \geq 0$. One of the agents, $\alpha_0 \in N$ is the

$S \leftarrow N^*$ // assume that every agent will receive the transmission
repeat
  Find the multicast routing tree $T(S)$
  Compute payments p_i such that each agent i pays an equal share of the cost for every link in $T(\{i\})$
  foreach $i \in S$ **do**
    if $\hat{v}_i < p_i$ **then** $S \leftarrow S \setminus \{i\}$ // i is dropped from S
until *no agents were dropped from* S

Figure 10.6 An algorithm for computing the allocation and payments for the Shapley value mechanism.

source of the transmission; there is also a set of agents $N^* \subseteq N$ who are interested in receiving it. Each $i \in N^*$ values the transmission at $v_i > 0$.

Our goal is to find a cost-sharing mechanism, a direct quasilinear mechanism $(\chi, \wp)$ that receives declarations $\hat{v}_i$ of each agent i's utility and determines which agents will receive the transmission and how much they will pay. The function χ determines a set of users $S \subseteq N^*$ who will receive the transmission. In order to do so, we must find a *multicast routing tree* $T(S) \subseteq L$ rooted at α_0 that spans S. We make a monotonicity assumption about the algorithm used to find $T(S)$,

multicast routing tree

$$S_1 \subseteq S_2 \Rightarrow T(S_1) \subseteq T(S_2).$$

The mechanism also includes a payment function $\wp$ that ensures that the agents in S share the costs of the links in $T(S)$. We denote the payment collected from $i \in S$ as p_i. We assume that the mechanism is computed by trusted hardware in the network (e.g., the routers); however, we will be concerned with communication complexity, and hence will look for ways that this computation can be distributed throughout the system.

Ideally, we would like a cost-sharing mechanism to be dominant-strategy incentive compatible, budget balanced, and efficient. However, we have already seen (Theorem 10.4.11) that such mechanisms do not exist. Thus, we will consider mechanisms that achieve two of these properties and relax the third.

Truthful and budget balanced: The Shapley value mechanism

First, we will describe a dominant-strategy truthful mechanism that achieves budget balance at the expense of efficiency. Intuitively, this mechanism is built around the idea that the cost of a link should be divided equally among the agents that use it. Its name comes from the fact that this objective can be seen as a special case of the Shapley value from coalitional game theory (see Section 12.2.1). We describe a centralized version of the mechanism in Figure 10.6.

To see why this algorithm leads to a dominant-strategy truthful mechanism, observe that the payments are "cross-monotonic." This means that each agent's payment can only increase when another agent is dropped, and hence that an agent's incentives are not affected by the order in which agents are dropped by the algorithm. That is, if the payment that the mechanism would charge an agent i given a set of other agents S exceeds i's utility, then i's payment is guaranteed

to exceed his utility for all subsets of the other agents $S' \subset S$. Since we only drop agents when their proposed payments exceed their utilities, the order in which we drop them is unimportant. Because we can drop agents "greedily" (i.e., without having to consider the possibility of reinstating them) the algorithm runs in polynomial time.

This algorithm can be run in a network by having all agents send their utilities to some node (e.g., the source α_0) and then running the algorithm there. However, although the algorithm is computationally tractable, this centralized approach requires an unreasonable amount of *communication* as the network becomes large. Thus, we would prefer a distributed solution. Unfortunately, no distributed algorithm can compute the same allocation and payments using asymptotically less communication than the centralized solution.

Theorem 10.6.5 *Any (deterministic or randomized) distributed algorithm that computes the same allocation and payments as the Shapley value algorithm must send $\Omega(|N^*|)$ bits over linearly many links in the worst case.*

Truthful and efficient: The VCG mechanism

Now we consider relaxing the budget balance requirement and instead insisting on efficiency. Unsurprisingly (consider Theorem 10.4.3) we must obtain a Groves mechanism in this case; VCG is the obvious choice. VCG can be easily used as a cost-sharing mechanism in the centralized case. Like the Shapley value mechanism, it requires only polynomial computation and hence is tractable. However, it has an interesting and important advantage over the Shapley value mechanism: it can also be made to work efficiently as a distributed algorithm.

Theorem 10.6.6 *A distributed algorithm can compute the same allocation and payments as VCG by sending exactly two values across each link.*

> **Proof.** The algorithm in Figure 10.7 computes VCG payments and allocations. Let l_i be the link connecting node i to its parent. Every nonroot node i sends and receives a single real-valued message over l_i. ■

This algorithm can be understood as passing messages from one node in the tree to the next. Observe that the first for loop proceeds "bottom up" (i.e., computing m for all children of a node i before computing m for node i itself), while the second for loop proceeds "top down" (i.e., computing s for a node i before computing s for any of i's children). Thus we can see the m's as messages that are passed up the tree, starting at the leaves, and the s's as messages that are passed back down, starting at the root.

Let us consider applying this algorithm to a sample multicast spanning tree (Figure 10.8a). In the upward pass (Figure 10.8b), every node i computes m_i, the marginal value connecting i to the network, given that its parent is connected. This is the maximum amount the agents on the subtree rooted at i would be willing to pay to join the multicast. In the downward pass (Figure 10.8c), every node i computes s_j for each child node j. s_j is the actual total surplus generated by connecting j to the multicast tree. If m_j or s_j is negative, the efficient allocation

```
// Upward pass
foreach node i, bottom up do
    m_i ← v̂_i − c(l_i)
    foreach node j ∈ children of i do
        m_i ← m_i + max(m_j, 0)
// Downward pass
S ← ∅
s_root ← m_root
foreach node i, top down do
    if s_i ≥ 0 then
        S ← S ⋃ {i}
        p_i ← max(v̂_i − s_i, 0)
    foreach node j ∈ children of i do
        s_j ← min(s_i, m_j)
```

Figure 10.7 A distributed algorithm for computing the efficient allocation and VCG payments for multicast cost sharing.

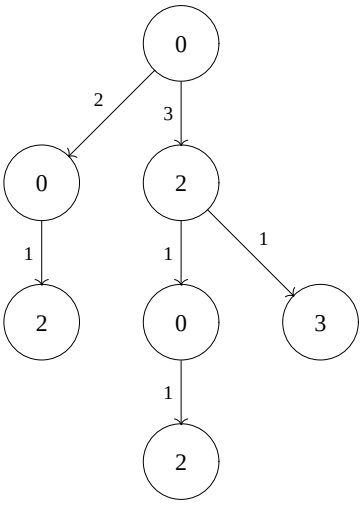
does not connect j. Thus, s_j can also be seen as the maximum amount by which agents in the subtree rooted at s_j could reduce their joint value declaration while remaining connected. Each connected node j is charged $\max(\hat{v}_j - s_j, 0)$, meaning that his surplus is equal to the amount he could have under-reported his value without being disconnected. These payments are illustrated in Figure 10.8d.

10.6.4 *Two-sided matching*

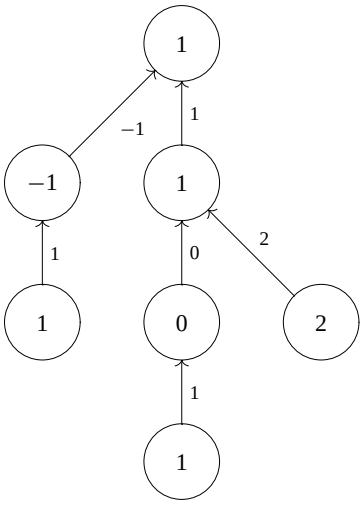
two-sided matching

So far in this chapter we have concentrated on mechanism design in quasilinear settings, meaning that we have assumed that money can be transferred between agents and the mechanism. However, there exist many interesting settings where such transfers are impossible, for example, because of legal restrictions. Examples of such problems include kidney exchanges, college admissions, and the assignment of medical interns to hospitals. *Two-sided matching* is a widely studied model that can be used to describe such domains. Under this model, each agent belongs to one of two groups. Members of each group are matched up, based on their declared preferences over their candidate partners. The mechanism design problem is to induce agents to disclose these preferences in a way that allows a desirable matching to be chosen, despite the restriction that payments cannot be imposed.

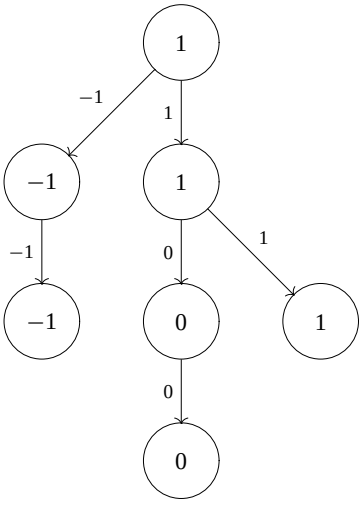
We will use the running example of a cohort of graduate students who must align with thesis advisors. Each student has a preference ordering over advisors (depending on their research interests, personalities, etc.), and likewise each potential advisor has a preference ordering over students. In this setting a social choice function is a decision about which students should be assigned to which advisors, given their preferences; as always, the mechanism design concern is how to implement a desired social choice function.



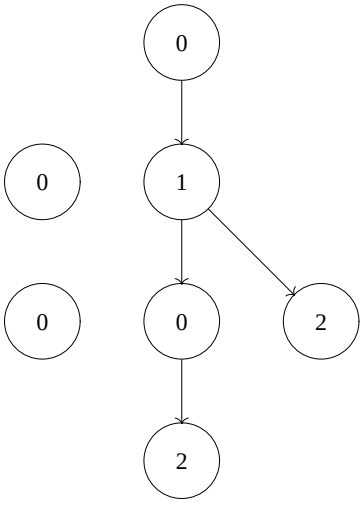
(a) Multicast spanning tree: nodes are labeled with values, edges are labeled with costs.



(b) Upward pass: each node i computes m_i and passes it to its parent.



(c) Downward pass: each node i computes s_j for child j and passes it down.



(d) Final allocation: only connected edges are shown; nodes are labeled with payments.

Figure 10.8 An example run of the algorithm from Figure 10.7.

We now define the setting more formally. Let A be a set of advisors and let S be a set of graduate students. We do not assume that $|A| = |S|$; thus, some students and/or advisors may remain unpaired. We assume that each student can have at most one advisor and each advisor will take at most one new student.[13] Each student i has a preference ordering $\succ_i$ over the advisors, and each advisor j has a preference ordering $\succ_j$ over the students. We write $a \succ_s a'$ to mean that student

13. Many but not all of the results in this section can also be extended to the case where advisors can take multiple students.

s prefers advisor a to advisor a', and $\varnothing \succ_s a$ to mean that s prefers not finding a supervisor to aligning with advisor a. In the latter case we say that advisor a is *unacceptable* to student s. Similarly, we write $s \succ_a s'$ and $\varnothing \succ_a s$. Note that we have assumed that all preferences are strict,[14] but that each agent can identify a set of partners with whom he would prefer not to be matched, effectively expressing a tie among unacceptable partners. In what follows, we adopt the convention that all advisors are female and all students are male. The resulting problem of finding good male–female pairings pays homage to the problem introduced in the two-sided matching literature a half-century ago, so-called *stable marriage*.

unacceptable matching

stable marriage

matching

Definition 10.6.7 (Matching) *A* matching $\mu : A \cup S \to A \cup S \cup \{\varnothing\}$ *is an assignment of advisors to students such that each advisor is assigned to at most one student and vice versa. More formally,* $\mu(s) = a$ *if and only if* $\mu(a) = s$*. Furthermore,* $\forall s \in S$*, either* $\exists a \in A, \mu(s) = a$ *or* $\mu(s) = \varnothing$ *(the student is unpaired), and likewise* $\forall a \in A$*, either* $\exists s \in S, \mu(a) = s$ *or* $\mu(a) = \varnothing$*.*

Note that it is always possible that some student s has the same match under two different matchings μ and μ', that is $\mu(s) = \mu'(s)$. In this case, s must be indifferent between matchings μ and μ'. A similar argument is true for advisors. Therefore, we use the operator $\succeq$ as well as $\succ$ when describing an agent's preference relation over matchings. More formally, $\mu(s) \succeq_s \mu'(s)$ means that either $\mu(s) \succ_s \mu'(s)$ or $\mu(s) = \mu'(s)$. Similarly, $\mu(a) \succeq_a \mu'(a)$ means that either $\mu(a) \succ_a \mu'(a)$ or $\mu(a) = \mu'(a)$.

Clearly, there are many possible matchings. The key question is which matching should be chosen, given students' and advisors' preference orderings. In other words, what properties does a desirable matching have? We identify two.

individually rational matching

Definition 10.6.8 (Individual rationality) *A matching* μ *is* individually rational *if no agent* i *prefers to remain unmatched than to be matched to* $\mu(i)$*.*

unblocked matching

Definition 10.6.9 (Unblocked) *A matching* μ *is* unblocked *if there exists no pair* (s, a) *such that* $\mu(s) \neq a$*, but* $a \succ_s \mu(s)$ *and* $s \succ_a \mu(a)$*.*

Intuitively, a matching is individually rational if no agent is matched with an unacceptable partner; a matching is unblocked if there exists no pair that would prefer to be matched with each other than with their respective partners. Putting these two definitions together, we obtain the concept of a stable matching.

stable matching

Definition 10.6.10 (Stable matching) *A matching* μ *is* stable *if and only if it is individually rational and unblocked.*

It turns out that in the setting we have defined above, no matter how many students and advisors there are and what preferences they have, there always exists at least one stable matching.

14. Our assumption that preferences are strict is restrictive; some of the results presented in this section no longer hold if it is relaxed.

Step 1: *each student applies to his most preferred advisor.*
repeat
  Step 2: *each advisor keeps her most preferred acceptable application (if any) and rejects the rest (if any).*
  Step 3: *each student who was rejected at the previous step applies to his next acceptable choice.*
until *no student applied in the last step*

Figure 10.9 Deferred acceptance algorithm, student-application version.

Theorem 10.6.11 (Gale and Shapley, 1962) *A stable matching always exists.*

Proof. The proof is obtained by giving a procedure that produces a stable matching given any set of student and advisor preferences. Here, we describe the so-called "student-application" version of the algorithm (Figure 10.9). There is an analogous algorithm in which advisors apply to students. This algorithm must stop in at most a quadratic number of steps, since no student ever applies more than once to any advisor. The outcome is a matching, since at any step each student is paired with at most one advisor and vice versa. The matching is individually rational, since no student or advisor is ever matched to an unacceptable agent.

It only remains to show that the matching is unblocked. Let μ be the matching produced by the algorithm. Assume for contradiction that μ is blocked by some student s and advisor a. Since s prefers a to his own match at μ, a must be acceptable to s, and so he must have applied to her before having applied to his match. Since s is not matched to a in μ, he must have been rejected by her in favor of someone she liked better. Therefore, (s, a) does not block μ, a contradiction. ■

Thus, there always exists at least one stable matching. However, these matchings are not necessarily unique—given a set of student and advisor preferences, there may exist many stable matchings. Let us now consider how different matchings can be compared.

student-optimal matching

Definition 10.6.12 *A stable matching μ is* student optimal *if every student likes it at least as well as any other stable matching; that is, $\forall s \in S$ and for every other stable matching μ', $\mu(s) \succeq_s \mu'(s)$.*

advisor-optimal matching

Along the same lines, we can define *advisor-optimal* matching. Now we can draw the following conclusions about stable matchings.

Theorem 10.6.13 *There exists exactly one student-optimal stable matching and one advisor-optimal stable matching. The matching produced by the student-application version of the deferred application algorithm is the student-optimal stable matching, and the matching produced by the advisor-application version of the deferred application algorithm is the advisor-optimal stable matching.*

Next, it turns out that any stable matching that is better for all the students is worse for all the advisors and vice versa.

Theorem 10.6.14 *If μ and μ' are stable matchings, $\forall s \in S$, $\mu(s) \succeq_s \mu'(s)$ if and only if $\forall a \in A$, $\mu'(a) \succeq_a \mu(a)$.*

achievable match

Say that an advisor a is *achievable* for student s, and vice versa, if there is a stable matching μ that matches a to s. Then we can state one implication of the above theorem: that the student-optimal stable matching is the worst stable matching from each advisor's point of view, and vice versa.

Corollary 10.6.15 *The student-optimal stable matching matches each advisor with her least preferred achievable student, and the advisor-optimal stable matching matches each student with her least preferred achievable advisor.*

Now let us move to the mechanism design question. If agents' preferences are private information, can we find a mechanism that ensures that a stable matching will be achieved? As is common in the matching literature, we restrict our attention to settings in which neither the agents' equilibrium strategies nor the mechanism itself are allowed to depend on the distribution over agents' preferences. Thus, we must rely on either dominant-strategy or *ex post* equilibrium implementation. Unfortunately, it turns out that stable matchings cannot be implemented under either equilibrium concept.

Theorem 10.6.16 *No mechanism implements stable matching in dominant strategies.*

Proof. By the revelation principle, if such a mechanism exists, then there also exists a direct truthful mechanism that selects matchings that are stable with respect to the declared preference orderings. Consider a setting with two students, s_1 and s_2, and two advisors, a_1, and a_2. Imagine that s_1, s_2 and a_1 declare the following preference orderings: $a_1 \widehat{\succ}_{s_1} a_2$, $a_2 \widehat{\succ}_{s_2} a_1$, and $s_2 \widehat{\succ}_{a_1} s_1$. Assume that a_2's true preference ordering is the following: $s_1 \succ_{a_2} s_2$. If a_2 declares the truth, then (1) the setting will have two stable matchings, μ and μ', given by $\mu(s_i) = a_i$ for $i \in \{1, 2\}$, and $\mu'(s_i) = a_j$ for $i, j \in \{1, 2\}$, $j \neq i$, and (2) any stable matching mechanism must choose one of μ or μ'. Suppose the mechanism chooses μ. Observe that if a_2 declares that her only acceptable student is s_1, then μ' is the only stable matching with respect to the stated preferences and the mechanism must select μ'—which a_2 prefers to μ. Similarly, we can show that if the mechanism chooses μ' when the above preference orderings are stated, then in a setting where $a_2 \succ_{s_2} a_1$ is s_2's true preference ordering, s_2 benefits by misreporting his preference ordering. Therefore, declaring the truth is not a dominant strategy for every agent. ■

Furthermore, it does not help to move to the *ex post* equilibrium concept, as can be proved along the same lines as Theorem 10.6.16.

Theorem 10.6.17 *No mechanism implements stable matching in* ex post *equilibrium.*

All is not hopeless, however—it turns out that we can obtain a positive mechanism design result for stable two-sided matching. The key is to relax our assumption that *all* agents are strategic. In our setting we will assume that advisors can

be compelled to behave honestly. Under this assumption, it is enough to prove the following result.

Theorem 10.6.18 *Under the direct mechanism associated with the student-application version of the deferred acceptance algorithm, it is a dominant strategy for each student to declare his true preferences.*

Proof. This proof proceeds by contradiction. Suppose that the claim is not true and, without loss of generality, say that it is not a dominant strategy for student s_1 to state his true preference ordering. Then, there is a preference profile $[\succsim] = (\succ_{s_1}, \succsim_{s_2}, \ldots, \succsim_{s_{|S|}}, \succsim_{a_1}, \ldots, \succsim_{a_{|A|}})$ such that s_1 benefits from reporting $\succ'_{s_1} \neq \succ_{s_1}$. Let μ be the stable matching obtained by applying the student application version of the deferred acceptance algorithm to $[\succsim]$. By Theorem 10.6.13, μ is student optimal with respect to $[\succsim]$. Let μ' be the stable matching obtained by applying the same algorithm to $[\succsim'] = (\succ'_{s_1}, \succsim_{s_2}, \ldots, \succsim_{s_{|S|}}, \succsim_{a_1}, \ldots, \succsim_{a_{|A|}})$. Note that except for s_1, all the other students and advisors declare the same preference ordering under $[\succsim]$ and $[\succsim']$.

Let $R = \{s_1\} \cup \{s : \mu'(s) \succsim_s \mu(s)\}$ denote the set of students who strictly prefer μ' to μ (with respect to their declared preferences $[\succsim]$). Note that we have included s_1 in R because, by assumption, $\mu'(s_1) \succ_{s_1} \mu(s_1)$. Let $T = \{a : \mu'(a) \in R\}$ denote the set of advisors who are matched with some student from R under μ'. In what follows we first show (Part 1) that any advisor $a \in T$ is matched with an (always different) student from R under μ; that is, $\{a : \mu'(a) \in R\} = \{a : \mu(a) \in R\} = T$. Then (Part 2) we show that there exist some $a_\ell \in T$ and $s_r \notin R$ such that (s_r, a_ℓ) blocks μ' at $[\succsim']$ and therefore μ' is not stable with respect to $[\succsim']$. This contradicts our assumption that μ' is a stable matching with respect to $[\succsim']$.

Part 1: For any $s \in R$, let $a = \mu'(s)$. Stability of μ with respect to $[\succsim]$ requires that advisor a be matched to some student under μ (rather than being unpaired), as otherwise (s, a) would block μ at $[\succsim]$; let $s' = \mu(a)$. If $s' = s_1$, then since s_1 prefers his match under μ' to his match under μ, $s' \in R$. Otherwise, since (with respect to his preferences declared in $[\succsim]$) s strictly prefers $\mu'(s)$ to $\mu(s)$, stability of μ with respect to $[\succsim]$ implies that $s' \succsim_a s$. Since we defined $s = \mu'(a)$, thus $s' \succsim_a \mu'(a)$. Then, stability of μ' with respect to $[\succsim']$ implies that $\mu'(s') \succsim_{s'} a$. Since we defined $a = \mu(s')$, thus $\mu'(s') \succsim_{s'} \mu(s')$ and therefore $s' \in R$. As a result, we can write $T = \{a : \mu'(a) \in R\} = \{a : \mu(a) \in R\}$.

Part 2: Since every student $s \in R$ prefers $\mu'(s)$ to $\mu(s)$, stability of μ with respect to $[\succsim]$ implies that $\forall a \in T,\ \mu(a) \succsim_a \mu'(a)$. Therefore, during the execution of the student-application algorithm on $[\succsim]$, each student $s \in R$ will apply to $\mu'(s)$ and will get rejected by $\mu'(s)$ at some iteration. In other words, each $a \in T$ rejects $\mu'(a) \in R$ at some iteration. Let s_ℓ be (weakly) the last student in R who applies to an advisor during the execution of the student-application algorithm. This application is sent to $\mu(s_\ell) \in T$; let $\mu(s_\ell) = a_\ell$. By construction, a_ℓ must have rejected $\mu'(a_\ell)$ at some strictly earlier iteration of the algorithm. Thus, when s_ℓ applies to a_ℓ, a_ℓ must reject an application from some $s_r \notin R$ such that $s_r \succsim_{a_\ell} \mu'(a_\ell)$ (fact 1). Note that $s_r \neq s_1$, since

$s_r \notin R$ and $s_1 \in R$. Since s_r applies to a_ℓ before he finally gets matched to $\mu(s_r)$, we have that $a_\ell \widehat{\succ}_{s_r} \mu(s_r)$. Furthermore, since $s_r \notin R$, we also have that $\mu(s_r) \widehat{\succeq}_{s_r} \mu'(s_r)$. Therefore $a_\ell \widehat{\succ}_{s_r} \mu'(s_r)$ (fact 2). Thus, from (fact 1) and (fact 2), (s_r, a_ℓ) blocks μ' at $[\widehat{\succ}']$ and μ' is not stable with respect to $[\widehat{\succ}']$, yielding our contradiction. ■

Of course, it is similarly possible to achieve a direct mechanism under which truth telling is a dominant strategy for advisors by using the advisor-application version of the deferred acceptance algorithm.

10.7 Constrained mechanism design

So far we have assumed that the mechanism designer is free to design any mechanism, but this assumption is violated in many applications—the ones discussed in this section, and many others. In particular, often one starts with given strategy spaces for each of the agents, with limited or no ability to change those. Examples abound:

- A city official who wishes to improve the traffic flow in the city cannot redesign cars or build new roads;
- A UN mediator who wishes to incent two countries fighting over a scarce resource to cease hostilities cannot change their military capabilities or the amount or value of the resource;
- A computer network operator who wishes to route traffic a certain way cannot change the network topology or the underlying routing algorithm.

Many other examples exist, and in fact such constraints can be thought of as the norm rather than the exception. How can such would-be mechanism designers intervene to influence the course of events?

social law

In Chapter 2 we already encountered this problem. Specifically, in Section 2.4 we saw how imposing *social laws*—that is, restricting the options available to agents—can be beneficial to all agents. Social laws played an important coordinating role (as in "drive on the right side of the road") and, furthermore, in some cases prevented the narrow self interests of the agents from hurting them (e.g., allowing cooperation in the Prisoners' Dilemma game). However, in that discussion we made the important assumption that once a social law was imposed (or agreed upon, depending on the interpretation), the agents could be assumed to follow it.

Here we relax this assumption, and we do so in three ways. First, we view the players as having the option of entering into a contract among themselves. Once they do—and only then—the center can impose arbitrary fines on law breakers, if he is aware of such deviations. The question in this case is which contracts the agents can be expected to enter, and how the work of the center can be minimized or even eliminated. Second, we consider the case in which the center can simply bribe the players to play a certain way (or, in more neutral language, offer positive incentives for certain actions). The question in this case is how the center can bias the outcome toward the desired one while minimizing his cost. Finally, we

consider a center who offers to play on behalf of the agents, who in turn are free to accept or reject the offer. We look at each setting in turn.

10.7.1 *Contracts*

Consider any given game G, and a center who can do the following.

1. Propose a contract before G is played. This contract specifies a particular outcome, that is, a unique action for each agent,[15] and a penalty for deviating from it.
2. Collect signatures on the contract and make it common knowledge who signed.
3. Monitor the players' actions during the execution of G.
4. If the contract was signed by all agents, fine anyone who deviated from it as specified by the contract.

Here we assume that players still have the freedom to choose whether or not to honor the agreement; the challenge is to design a mechanism such that, in equilibrium, they will do so.

The technical results in this line of work will refer to games of complete information, but for intuition consider the example of an online marketplace such as eBay. (We discuss auctions in detail in Chapter 11, but those details are not needed here.) Consider the entire game being played, including the decision after the close of the auction by the seller of whether to deliver the good and by the buyer of whether to send payment. Straightforward analysis shows that that the equilibrium is for neither to keep his promise, and the experience with fraud in online auctions demonstrates that the problem is not merely theoretical. It would be in an online auction site's interest to find a way to bind its customers to their promises.

The first question one may ask is what the achievable outcomes are. What outcomes may the center suggest, with associated penalties, that the agents will accept? However, once the problem is couched in a formal setting, it is not hard to show a folk theorem of sorts: any outcome will be accepted when accompanied by appropriate fines, so long as the payoffs of each agent in that outcome are better than that player's payoffs in *some* equilibrium of the original game.

Although the center can achieve almost any outcome, it would seem to require great effort: suggesting an outcome, collecting signatures, observing the game, and enforcing the contracts. If this procedure happens not just for one game, but for hundreds or thousands per day, the center may wish to find a way to avoid this burden while still achieving the same effect.

However, one can often achieve the same effects with much less effort on the part of the center. We continue to assume that the center still needs to propose a contract. We also simply assume that it does not monitor the game. Nor does it participate in the signing phase; the agents do that among themselves

15. In the parlance of Section 2.4, a *convention*.

using a broadcast channel. While we might imagine that the players could simply broadcast their signatures, this protocol allows a single player to learn the others' signatures and threaten them with fines. Nonetheless, one can construct a more complicated protocol—using a second stage of contracts—that does not require the center's participation. The only phase in which the center's protocol requires it to get involved under some conditions is the enforcement stage. However, here too one can minimize the effort required in actuality. This is done by devising contracts that, *in equilibrium*, at this stage too the center sits idle. Among other things, one can show that if the game play is *verifiable* (if the center can discover after the fact whether players obeyed the contract), then anything achievable by a fully engaged center is also achievable by a center that in equilibrium always sits idle.

10.7.2 *Bribes*

Consider the following simple congestion setting, similar to the one discussed in Section 10.1.2. Assume that there are two agents, 1 and 2, who must select among two service providers. One of the service providers, f, is a fast one, while the other, s, is a slower one. We capture this by having an agent obtain a payoff of 6 when he is the only one who uses f, and a payoff of 4 when he is the only one who uses s. If both agents select the same service provider then the speeds they each obtain decrease by a factor of 2, leading to half the payoff. Thus, if both agents use f then each of them obtains a payoff of 3, while if both use s then each obtains 2. Written in normal form, this game is described as follows.

$M =$

	f	s
f	3, 3	6, 4
s	4, 6	2, 2

Assume that the mechanism designer wishes to prevent the agents from using the same service provider (leading to low payoffs for both) and further wants to obtain a mechanism in which each agent has a dominant strategy. Then it can do as follows: it can promise to pay agent 1 a value of 10 if both agents will use f, and promise to pay agent 2 a value of 10 if both agents will use s. These promises transform M to the following game.

$M' =$

	f	s
f	13, 3	6, 4
s	4, 6	2, 12

Notice that in M', strategy f is dominant for agent 1, and strategy s is dominant for agent 2. As a result the only rational strategy profile is the one in which agent 1 chooses f and agent 2 chooses s. Hence, the mechanism designer implements one of the desired outcomes. Moreover, given that the strategy profile (f, s) is selected, the mechanism will have to pay nothing. It has just implemented, *in dominant strategies*, a desired behavior (which had previously been obtained in one of the game's Nash equilibria) at zero cost, relying only on its creditability, without modifying the rules of interactions or enforcing any type of behavior! In this case we say that the desired behavior has a 0-implementation. More generally, an outcome has a *k-implementation* if it can be implemented in dominant strategies using such payments with a cost in equilibrium of at most k. This definition can be used to prove the following result.

k-implementation

Theorem 10.7.1 *An outcome is* 0*-implementable iff it is a Nash equilibrium.*

10.7.3 *Mediators*

We have so far considered a center who can enforce contracts, and one who can offer monetary incentives. We now consider a more active center, one who can play on behalf of agents.

Consider the ever-recurring example of the Prisoners' Dilemma game.

	C	D
C	4, 4	0, 6
D	6, 0	1, 1

As you know, the strategy profile (D, D) is a Nash equilibrium, and even an equilibrium in weakly dominant strategies. However, it is not what is called a *strong equilibrium*, that is, a strategy profile that is stable against group deviations. If *both* players deviate to (C, C), the payoff of each one of them will increase.

strong equilibrium

Now consider a reliable mediator who offers the agents the following protocol. If both agents agree to use the mediator's services then he will perform the action C (cooperate) on behalf of both agents. However, if only one agent agrees to use his services then he will perform the action D (defect) on behalf of that agent. We assume that when accepting the mediator's offer the agent is committed to using the mediator and forgoes the option of acting on his own; however, he is free to reject the offer, in which case he is free to use any strategy. This induces the following game between the agents.

	Mediator	C	D
Mediator	4, 4	6, 0	1, 1
C	0, 6	4, 4	0, 6
D	1, 1	6, 0	1, 1

The mediated game has a most desirable property: it is a strong equilibrium for the two agents to use the mediator's services, guaranteeing each a payoff of 4. No coalition (i.e., either of the two agents alone, or the pair) can deviate and achieve for all coalition members a payoff greater than 4.

This example turns out to be more than a happy coincidence. While strong equilibria are rare in general, adding mediators make them less rare. For example, adding a mediator to any *balanced symmetric game* yields a strong equilibrium with optimal surplus.[16] Also, if we consider only deviations by coalitions of size at most k (a so-called k-strong equilibrium), we have the following. For any symmetric game with n agents, if $k!$ divides n then there exists a k-strong mediated equilibrium, leading to optimal surplus.[17] However, if $k!$ does not divide n, then it can be shown that the game may or may not possess a k-strong equilibrium.

balanced game

10.8 History and references

Mechanism design is covered to varying degrees in modern game theory textbooks, but even better are the microeconomic textbook of Mas-Colell et al. [1995] and books on auction theory such as Krishna [2002]. Good overviews from a computer science perspective are given in the introductory chapters of Parkes [2001] and in Nisan [2007]. Specific publications that underlie some of the results covered in this chapter are as follows.

The foundational idea of mechanisms as communication systems that select outcomes based on messages from agents is due to Hurwicz [1960], who also elaborated the theory to include the idea that mechanisms should be "incentive compatible" [1972]. The revelation principle was first articulated by Gibbard [1973] and was developed in the greatest generality by [Myerson, 1979, 1982, 1986]. In 2007, Hurwicz and Myerson shared a Nobel Prize (along with Maskin, whose work we do not discuss in this book), "for having laid the foundations

16. Full discussion of balanced games is beyond the scope of this discussion. However, we remark that a game in strategic form is called balanced if its associated core is nonempty. The core of a game is defined in the context of coalitional games in Chapter 12.

17. As an anecdote, we note that the Israeli parliament consists of $120 = 5!$ members. Hence, every anonymous game played by this parliament possesses an optimal surplus symmetric 5-strong equilibrium. While no Parliament member is able to give the right of voting to a mediator, this right of voting could be replaced in real life by a commitment to follow the mediator's algorithm.

of mechanism design theory." Theorem 10.2.6 is due to both Satterthwaite and Gibbard, in two separate publications [Gibbard, 1973; Satterthwaite, 1975].

The VCG mechanism was anticipated by Vickrey [1961], who outlined an extension of the second-price auction to multiple identical goods. Groves [1973] explicitly considered the general family of truthful mechanisms applying to multiple distinct goods (though the result had appeared already in his 1969 Ph.D. dissertation). Clarke [1971] proposed his tax for use with public goods (i.e., goods such as roads and national defense that are paid for by all regardless of personal use). Theorem 10.4.3 is due to Green and Laffont [1977]; Theorem 10.4.11 is due to that paper as well as to the earlier Hurwicz [1975]. The fact that Groves mechanisms are payoff equivalent to all other Bayes–Nash incentive-compatible efficient mechanisms was shown by Krishna and Perry [1998] and Williams [1999]; the former reference also gave the results that VCG is *ex interim* individually rational and that VCG collects the maximal amount of revenue among all *ex interim* individually-rational Groves mechanisms. Recent work shows that some "VCG drawbacks" are also problems with broad classes of mechanisms; for example, this has been shown for nonfrugality [Archer and Tardos, 2002; Elkind et al., 2004] and for revenue monotonicity [Rastegari et al., 2007]. The problem of participating in Groves mechanisms under multiple identities (specifically in the case of combinatorial auctions, which are described in Section 11.3) was investigated by Yokoo [2006]. Although it is not generally possible to return *all* VCG revenue to the agents, recent research has investigated VCG-like mechanisms that collect as little revenue from the agents as possible and thus minimize the extent to which they violate (strong) budget balance [Porter et al., 2004b; Cavallo, 2006; Guo and Conitzer, 2007]. Interestingly, the first of these papers came to the problem through a desire to achieve *fair* outcomes. The Myerson–Satterthwaite theorem (10.4.12) appears in Myerson and Satterthwaite [1983]. The AGV mechanism is due (independently) to Arrow [1977] and d'Aspremont and Gérard-Varet [1979].

The section on implementation in dominant strategies follows Nisan [2007]; Theorem 10.5.5 is due to Roberts [1979]. Second-chance mechanisms are due to Nisan and Ronen [2007]. (One difference: we preferred the term *Groves-based mechanisms* to Nisan and Ronen's *VCG-based mechanisms*.)

Our section on task scheduling reports results due to Nisan and Ronen [2001]; this work also introduced the term *algorithmic mechanism design*. Our section on bandwidth allocation in computer networks follows Johari [2007], which in turn draws on Johari and Tsitsiklis [2004]; the proportional allocation mechanism is due to Kelly [1997], and the VCG-like mechanism is described in Johari and Tsitsiklis [2005]. Our section on multicast cost sharing follows Feigenbaum et al. [2007], which draws especially on Feigenbaum et al. [2001; 2003]. Our discussion of mechanisms for two-sided matching draws on Roth and Sotomayor [1990], Schummer and Vohra [2007] and Gale and Shapley [1962]. The first algorithm for finding stable matchings was developed by Stalnaker [1953], and was used to match medical interns to hospitals. The stable matching problem was formalized by Gale and Shapley [1962], who also introduced the deferred acceptance algorithm. Theorems 10.6.13 and 10.6.14 follow Knuth [1976];

Theorems 10.6.16 and 10.6.17 are due to Roth [1984]; and Theorem 10.6.18 draws partly on Schummer and Vohra [2007] and subsequent unpublished correspondence between Baharak Rastegari and Rakesh Vohra. A more general version of Theorem 10.6.18 appeared in Roth and Sotomayor [1990] and Dubins and Freedman [1981].

The notion of social laws and conventions are introduced in Shoham and Tennenholtz [1995]. The use of contracts to influence the outcome of a game is discussed in McGrew and Shoham [2004]. The use of monetary incentives to influence the outcome of a game, or k-implementation, is introduced in Monderer and Tennenholtz [2003]. Mediators and their connections to strong equilibria are discussed in Monderer and Tennenholtz [2006].