CS 405/6001: Game Theory and Algorithmic Mechanism Design **Jul–Nov 2026**

Lecture 13: Introduction to Mechanism Design

Lecturer: Swaprava Nath *Scribe(s): Claude Scribe*

Disclaimer: *These notes aggregate content from several texts and have not been subjected to the usual scrutiny deserved by formal publications. If you find errors, please bring to the notice of the Instructor.*

13.1 Recap and Motivation

For twelve lectures we studied game theory in one direction: given the rules of a game – players, actions, payoffs – predict what rational, intelligent players will do. **Mechanism design** runs this in reverse, which is why it is often called *inverse game theory*. We start with a desired outcome and design the rules of a game so that self-interested play by rational agents naturally produces that outcome.

Example 13.1 (Where this shows up) *A ride-hailing platform wants trips matched to the drivers who value them most, without ever directly asking drivers "how much do you value this trip?" and trusting the answer. A recommendation platform wants advertisers to bid their true value for an ad slot. A blockchain protocol wants validators to report the block they actually observed rather than the one that benefits them most to claim. In each case the designer chooses the rules of the game; the participants, each holding private information, play it strategically.*

The central difficulty in all of these examples is the same: the designer does not know what the agents privately know. If the designer already had this information, there would be no design problem at all – simply compute the best outcome and impose it. Because agents' private information is exactly what determines the best outcome, the designer must construct a game that induces agents to *reveal* that information, honestly, through their own strategic choices.

13.2 The Formal Framework

13.2.1 Players, Outcomes, and Types

Definition 13.2 (Environment) *A mechanism design environment consists of*

- *a finite set of players* $N = \{1, 2, \ldots, n\}$*;*
- *a set of possible* ***outcomes*** X *– e.g., the winner in an election, or a full description of who gets which resource and at what price;*
- *for each player* i*, a set of possible* ***types*** Θ_i*, where a type* $\theta_i \in \Theta_i$ *is everything private to* i *that is relevant to their preferences over* X *but unknown to the designer and to the other players.*

A full type profile is $\theta = (\theta_1, \ldots, \theta_n) \in \Theta = \Theta_1 \times \cdots \times \Theta_n$.

A type contains, for instance, a bidder's true valuation for an object, or a voter's sincere ranking of candidates. Since the designer cannot observe θ_i directly, the entire enterprise of mechanism design is about coping with this asymmetry of information.

13.2.2 How Types Determine Preferences

A player's type shapes their preferences over outcomes in one of two ways.

Definition 13.3 (Ordinal preferences) *θ_i simply defines a ranking over X, with no cardinal meaning. If $X = \{a, b, c\}$, a type might be the ordering $a \succ_i b \succ_i c$.*

Definition 13.4 (Cardinal preferences) *θ_i determines a utility function u_i assigning a real number to each outcome. Two sub-cases matter:*

- ***Private value model**: $u_i : X \times \Theta_i \to \mathbb{R}$ – agent i's utility depends only on the outcome and i's own type. A cloud-compute team bidding for GPU-cluster time cares about its own training deadline, not another team's.*
- ***Interdependent value model**: $u_i : X \times \Theta \to \mathbb{R}$ – i's utility can depend on* everyone's *type. In a spectrum auction, a bidder's own geological or engineering survey is private, but their ultimate profit depends on the total usable capacity of the band, which is correlated with every bidder's signal.*

13.3 Two Running Examples

Example 13.5 (Voting) *A group of agents – for instance, a DAO's token holders, or a steering committee – must choose one option from a set of candidates. Outcome space X is the set of candidates, e.g., $X = \{Alice, Bob, Carol\}$. Agent i's type θ_i is a ranking over candidates, e.g., $\theta_i = (Alice \succ Bob \succ Carol)$.*

Example 13.6 (Single-object allocation) *There are n agents competing for one indivisible object – concretely, a cloud provider auctioning a single high-demand GPU node among n ML teams. An outcome is $x = (a, p)$, where $a = (a_1, \ldots, a_n)$, $a_i \in \{0, 1\}$, $\sum_i a_i \leqslant 1$ describes the allocation, and $p = (p_1, \ldots, p_n)$ describes the payments. Each agent's type θ_i is their private value for the object – their maximum willingness to pay. The utility of agent i is*

$$u_i(x, \theta_i) = a_i \theta_i - p_i.$$

If i wins ($a_i = 1$), their net utility is value minus price paid; if i loses ($a_i = 0$), utility is 0 (a losing team pays nothing).

13.4 Social Choice Functions

The designer's objective is formalized through a **Social Choice Function**.

Definition 13.7 (Social Choice Function) *A social choice function (SCF) is a map $f : \Theta \to X$ that specifies the outcome the designer wants for every possible profile of agents' types.*

Informally: "if I, the designer, knew the true type profile were θ, I would want to implement outcome $f(\theta)$."

Example 13.8 (SCFs for our two settings) *In the voting setting (Example 13.5), a natural SCF picks a candidate who would defeat every other candidate in a pairwise contest, when such a candidate exists. In a public-project choice setting, where each agent i has a value $\theta_i(a)$ for each candidate project $a \in X$, a utilitarian designer maximizing total value uses*

$$f(\theta) \in \arg\max_{a \in X} \sum_{i \in N} \theta_i(a).$$

This raises the central question of the course: *how can we design a game whose equilibrium outcome always coincides with $f(\theta)$, even though the designer never observes θ?*

13.5 Mechanisms

Definition 13.9 (Mechanism) *A mechanism is a pair $\langle (M_1, \dots, M_n), g \rangle$, consisting of a **message space** M_i for each agent – the set of signals i can send – and an **outcome function** $g : M_1 \times \cdots \times M_n \to X$ mapping every profile of messages to a final outcome.*

Once the mechanism is fixed, a genuine game is created: agent i, knowing only θ_i, strategically chooses $m_i \in M_i$ to maximize $u_i(g(m_1, \dots, m_n), \theta_i)$.

13.5.1 Direct versus Indirect Mechanisms

Message spaces can be arbitrarily rich – multi-round bidding ladders, sequential proposals, anything at all. We call this general class **indirect mechanisms**. A crucial and much simpler subclass restricts the message space to be the type space itself.

Definition 13.10 (Direct mechanism) *A direct mechanism sets $M_i = \Theta_i$ for every i, so the outcome function is exactly the SCF: $g = f$. Agents are simply asked to report their type, and the outcome is computed by applying f to the reported profile.*

13.6 Incentive Compatibility and the Revelation Principle

We want a strong, robust notion of implementation – one that does not require agents to form complex beliefs about each other's play. This leads to dominant-strategy implementation.

Definition 13.11 (Weakly dominant strategy) *In a mechanism $\langle M_1, \dots, M_n, g \rangle$, message m_i is weakly dominant for player i at θ_i if*

$$u_i(g(m_i, \tilde{m}_{-i}), \theta_i) \geqslant u_i(g(m_i', \tilde{m}_{-i}), \theta_i), \quad \forall \tilde{m}_{-i}, \forall m_i'.$$

Definition 13.12 (Dominant Strategy Implementable (DSI)) *An SCF $f : \Theta \to X$ is implemented in dominant strategies by $\langle M_1, \dots, M_n, g \rangle$ if there exist message mappings $s_i : \Theta_i \to M_i$ such that $s_i(\theta_i)$ is a dominant strategy for agent i at every θ_i, and $g(s_1(\theta_1), \dots, s_n(\theta_n)) = f(\theta)$ for every $\theta \in \Theta$.*

Definition 13.13 (Dominant Strategy Incentive Compatible (DSIC)) *A direct mechanism $\langle \Theta, f \rangle$ is DSIC if*

$$u_i(f(\theta_i, \tilde{\theta}_{-i}), \theta_i) \geqslant u_i(f(\theta_i', \tilde{\theta}_{-i}), \theta_i), \quad \forall \tilde{\theta}_{-i}, \forall \theta_i', \forall \theta_i, \forall i \in N.$$

A DSIC mechanism is truthful no matter what others report – such mechanisms are called **strategy-proof**. This immediately raises the question: to check whether a goal f is DSI, must we search over the enormous space of all possible indirect mechanisms? The following theorem says no.

Theorem 13.14 (The Revelation Principle) *A social choice function f is dominant strategy implementable if and only if its direct mechanism is dominant strategy incentive compatible.*

Proof: ($\Leftarrow$) If the direct mechanism for f is DSIC, truth-telling is itself a dominant strategy for every player, so the direct mechanism implements f in dominant strategies; hence f is DSI.

($\Rightarrow$) Suppose f is DSI via some mechanism $\mathcal{M} = \langle M, g\rangle$. By Definition 13.12, each player i has a dominant strategy $s_i(\theta_i) \in M_i$ at every true type θ_i, and $g(s_1(\theta_1), \dots, s_n(\theta_n)) = f(\theta)$.

We must show the direct mechanism for f is DSIC. Fix player i with true type θ_i, and compare truth-telling (reporting θ_i) against a lie (reporting some $\theta_i' \neq \theta_i$) in the direct mechanism. Reporting θ_i or θ_i' directly corresponds, from the perspective of $\mathcal{M}$, to player i playing $s_i(\theta_i)$ or $s_i(\theta_i')$ respectively, against the other players' equilibrium messages $s_{-i}(\theta_{-i})$.

Since $s_i(\theta_i)$ is dominant for i in $\mathcal{M}$, it yields weakly higher utility than $s_i(\theta_i')$ against *any* play of the others, in particular against $s_{-i}(\theta_{-i})$:

$$u_i\big(g(s_i(\theta_i), s_{-i}(\theta_{-i})), \theta_i\big) \geqslant u_i\big(g(s_i(\theta_i'), s_{-i}(\theta_{-i})), \theta_i\big).$$

Substituting $f(\cdot) = g(s(\cdot))$ gives

$$u_i(f(\theta_i, \theta_{-i}), \theta_i) \geqslant u_i(f(\theta_i', \theta_{-i}), \theta_i),$$

exactly the DSIC condition. Since θ_i, θ_i', and i were arbitrary, the direct mechanism is DSIC. ∎

Remark 13.15 *The practical force of Theorem 13.14 is that we never need to invent clever indirect mechanisms to check dominant-strategy implementability. One test suffices: is "report your type honestly" itself a dominant strategy? If not, no cleverer mechanism – however many rounds of bidding or signaling it uses – can implement f in dominant strategies either.*

13.7 Bayesian Implementation

Dominant-strategy implementation is the strongest and most robust notion, but many desirable SCFs are simply not DSI. We relax to a weaker solution concept: implementation in **Bayesian Nash Equilibrium**.

In the Bayesian model, a player's strategy need only be optimal against their *beliefs* about others, not against every conceivable action. All players share a common prior P over type profiles $\theta = (\theta_1, \dots, \theta_n)$; Nature draws θ from P, and each player i privately learns only θ_i, forming posterior beliefs about θ_{-i} via Bayes' rule.

Example 13.16 (Where this fits) *A recommendation platform running an ad auction does not know an individual advertiser's exact budget, but from historical auction data it knows the* distribution *of advertiser budgets. That distribution is the common prior P.*

Definition 13.17 (Bayesian implementable SCF) *An SCF f is implementable in Bayesian strategies if there exists a mechanism $\langle M, g\rangle$ and a strategy profile $s = (s_1, \dots, s_n)$ forming a Bayesian Nash Equilibrium, i.e., for every i and $\theta_i \in \Theta_i$,*

$$\mathbb{E}_{\theta_{-i}|\theta_i}\big[u_i(g(s_i(\theta_i), s_{-i}(\theta_{-i})), \theta)\big] \geqslant \mathbb{E}_{\theta_{-i}|\theta_i}\big[u_i(g(m_i', s_{-i}(\theta_{-i})), \theta)\big], \quad \forall m_i' \in M_i,$$

and $g(s_1(\theta_1), \ldots, s_n(\theta_n)) = f(\theta)$ *for every* $\theta \in \Theta$.

Since a strategy optimal against *every* opponent action is in particular optimal *on average*, every DSI mechanism is also Bayesian implementable – dominant-strategy implementation is strictly stronger.

Definition 13.18 (Bayesian Incentive Compatible (BIC)) *A direct mechanism* $\langle \Theta, f \rangle$ *is BIC if truth-telling forms a Bayesian Nash Equilibrium: for every* i*,*

$$\mathbb{E}_{\theta_{-i}|\theta_i}\big[u_i(f(\theta_i, \theta_{-i}), \theta)\big] \geqslant \mathbb{E}_{\theta_{-i}|\theta_i}\big[u_i(f(\theta_i', \theta_{-i}), \theta)\big], \quad \forall \theta_i, \theta_i' \in \Theta_i.$$

Unlike DSIC, which demands truthfulness against every possible report from others, BIC only requires truthfulness *in expectation*, weighted by the player's belief.

Theorem 13.19 (Revelation Principle for Bayesian Implementation) *If an SCF* f *is implementable in a Bayesian equilibrium, then its direct mechanism is Bayesian Incentive Compatible.*

The proof follows the same logic as Theorem 13.14, with every utility comparison replaced by an expected-utility comparison over the common prior. As before, this collapses the search for Bayesian implementability to a single check: is the direct mechanism truthful on average?

13.8 Summary and Preview

We introduced mechanism design as the inverse of game theory: fix a Social Choice Function f, then design a mechanism $\langle M, g \rangle$ whose equilibrium outcome coincides with $f(\theta)$ for every type profile θ, despite the designer never observing θ directly. Direct mechanisms, which simply ask agents to report their types, are of special importance because of the Revelation Principle (Theorem 13.14): checking dominant-strategy implementability of f reduces to checking whether truth-telling is itself a dominant strategy in the direct mechanism. The same logic extends to the weaker Bayesian setting via BIC. Next lecture, we set mechanism design aside for a moment and ask a more basic question: even if all agents report their preferences honestly, can we always aggregate those preferences fairly into a single social ranking? The answer, due to Kenneth Arrow, is a striking impossibility result.

References

[MSZ13] M. Maschler, E. Solan and S. Zamir, *Game Theory*, Chapter 10, Cambridge University Press, 2013.

[Nis07] N. Nisan, "Introduction to Mechanism Design (for Computer Scientists)," in *Algorithmic Game Theory*, Cambridge University Press, 2007.