Implementation Theory

Game Theory · Module 6 — Mechanism Design

Node 6.5

Position in Knowledge Graph

System: Game Theory **Structural Domain:** Mechanism Design **Node:** 6.5

Implementation theory asks a harder question than the revelation principle. The revelation principle (6.2) guarantees **partial implementation**: some equilibrium of the direct mechanism yields the desired outcome. But what if the mechanism has multiple equilibria, and some of them yield the *wrong* outcome? The revelation principle is silent on this. Implementation theory asks for **full implementation**: design a mechanism such that *every* equilibrium — not just some — yields the desired outcome for every type profile. This is a much more demanding requirement, and it requires fundamentally different tools. The central result is **Maskin's theorem** (1977, published 1999): a social choice correspondence is Nash-implementable if and only if it satisfies a monotonicity condition (now called **Maskin monotonicity**) and a no-veto-power condition. The theory extends to subgame-perfect implementation (Moore–Repullo 1988, Abreu–Sen 1990), which uses sequential mechanisms to refine away bad equilibria — a natural connection to the extensive-form tools of Module 3.

Formal Definition

Let $N = \{1, \dots, n\}$, let Θ be the type space, and let X be the outcome set. A **social choice correspondence** (SCC) is a set-valued map $F : \Theta \rightrightarrows X$ — for each type profile, $F(\theta)$ is the set of acceptable outcomes.

A mechanism $\Gamma = (M, g)$ **fully Nash-implements** F if for every $\theta \in \Theta$:

$$\mathrm{NE}(\Gamma, \theta) = F(\theta),$$

where $\mathrm{NE}(\Gamma, \theta) = \{g(m^*) : m^* \text{ is a Nash equilibrium of the complete-information game induced by } \Gamma \text{ at } \theta\}$.

The crucial difference from partial implementation: we require equality, not just $F(\theta) \subseteq \mathrm{NE}(\Gamma, \theta)$. Every equilibrium outcome must be in $F(\theta)$ (no unwanted equilibria), and every outcome in $F(\theta)$ must be supported by some equilibrium (no missing outcomes).

Maskin monotonicity. An SCC F satisfies Maskin monotonicity if: whenever $x \in F(\theta)$ and, for all i, every alternative that i weakly prefers to x at θ is also weakly preferred to x at θ', then $x \in F(\theta')$. Formally: if for all i and all $y \in X$,

$$u_i(x, \theta_i) \geq u_i(y, \theta_i) \implies u_i(x, \theta_i') \geq u_i(y, \theta_i'),$$

then $x \in F(\theta')$.

Equivalently: if $x \in F(\theta)$ and the **lower contour set** of x for each agent shrinks (weakly) from θ to θ', then $x \in F(\theta')$. The SCC is "monotone" in the sense that improving an outcome's ranking in everyone's preferences does not remove it from the social choice.

No veto power (NVP). If $n \geq 3$ and at least $n - 1$ agents rank x as their most-preferred alternative at θ, then $x \in F(\theta)$.

Maskin's Theorem. An SCC F is fully Nash-implementable if and only if: 1. F satisfies Maskin monotonicity (necessary for $n \geq 2$). 2. F satisfies NVP (sufficient, with monotonicity, for $n \geq 3$).

For $n = 2$, the characterization is more complex (Dutta–Sen 1991, Moore–Repullo 1990).

Full implementation vs partial implementation. To fix terminology: a mechanism Γ **partially implements** F if some equilibrium of Γ at each θ yields an outcome in $F(\theta)$. It **fully implements** F if every equilibrium at each θ yields an outcome in $F(\theta)$, and every outcome in $F(\theta)$ is supported by some equilibrium. The revelation principle guarantees partial implementation via direct mechanisms. Full implementation requires the Maskin conditions and generally requires indirect mechanisms (the direct mechanism typically has extra, undesirable equilibria).

The distinction matters whenever agents might coordinate on a "wrong" equilibrium. In a one-shot mechanism with no external enforcement, there is no guarantee that agents will play the "right" equilibrium. Full implementation eliminates this concern by ensuring that all equilibria are right.

Intuition

The revelation principle says: there exists a mechanism where truth-telling is *an* equilibrium yielding the desired outcome. Implementation theory says: there exists a mechanism where *every* equilibrium yields the desired outcome. The gap is the set of "bad equilibria" — strategy profiles that are Nash equilibria of the mechanism but produce the wrong outcome.

Bad equilibria are not a theoretical curiosity. In a direct revelation mechanism, "everyone reports the same false type" can be a Nash equilibrium if no single agent can profitably deviate by reporting truthfully. For example, in a voting mechanism, if all voters coordinate on a false report, no single voter's deviation changes the outcome (because the mechanism sees $n - 1$ identical false reports and one true one, and may still implement the false outcome). The direct mechanism is manipulable not by individuals but by accidental coordination on a bad equilibrium.

Maskin monotonicity captures a necessary structural condition: if preferences change in a way that only *improves* an outcome's ranking for every agent, then the SCC should still select that outcome. If this fails — if an outcome is selected at θ but not at θ' despite being ranked at least as well by everyone — then no mechanism can separate the two type profiles, because any Nash equilibrium supporting x at θ remains a Nash equilibrium at θ' (since each agent's lower contour set has only shrunk).

No veto power is a mild condition: if $n - 1$ agents all top-rank the same outcome, it should be in the social choice set. This excludes pathological SCCs where a consensus is overruled.

Subgame-perfect implementation relaxes the problem by using extensive-form mechanisms. The idea: design a sequential game where the first mover makes a claim, the second mover can challenge, the third mover can counter-challenge, and so on. The sequential structure refines away bad Nash equilibria because some are not subgame-perfect. Moore and Repullo (1988) showed that virtually any SCC satisfying a weak "condition α" (a weakening of Maskin monotonicity) is subgame-perfect implementable with $n \geq 2$ agents.

Structural Role

6.5 completes the mechanism-design circle by addressing the robustness of implementation. The sequence is:

- **6.1–6.2:** What can be implemented if we only require *some* equilibrium to yield $f(\theta)$? Answer: any BIC direct mechanism (revelation principle).
- **6.3–6.4:** What is the *optimal* such mechanism? Answer: Myerson's virtual-valuation auction.
- **6.5:** What can be implemented if we require *every* equilibrium to yield $f(\theta)$? Answer: Maskin-monotone SCCs (for Nash), broader classes for SPE implementation.

This node is the structural link between mechanism design (Module 6) and extensive-form game theory (Module 3): subgame-perfect implementation explicitly uses game trees, information sets, and backward induction to refine equilibria. It is also the link to the theory of robust mechanism design: the concern with multiple equilibria motivates Wilson's (1987) critique and the subsequent literature on "detail-free" mechanisms (Bergemann–Morris 2005).

Mathematical Development

Necessity of Maskin monotonicity

Theorem. If F is Nash-implementable, then F is Maskin-monotone.

Proof. Suppose $\Gamma = (M, g)$ implements F in Nash. Let $x \in F(\theta)$. Then there exists a Nash equilibrium m^* of Γ at θ with $g(m^*) = x$. That is, for all i and all $m_i' \in M_i$:

$$u_i(g(m^*), \theta_i) \;\geq\; u_i(g(m_i', m_{-i}^*), \theta_i).$$

Now suppose preferences change to θ' with the monotonicity property: for all i and all $y \in X$, $u_i(x, \theta_i) \geq u_i(y, \theta_i) \Rightarrow u_i(x, \theta_i') \geq u_i(y, \theta_i')$. Then for any deviation m_i', the outcome $g(m_i', m_{-i}^*) = y$ satisfies $u_i(x, \theta_i) \geq u_i(y, \theta_i)$ (since m^* was NE at θ), hence $u_i(x, \theta_i') \geq u_i(y, \theta_i')$. So m^* is still a NE at θ', and $x = g(m^*) \in \text{NE}(\Gamma, \theta') = F(\theta')$. ■

Sufficiency: Maskin's mechanism (sketch for $n \geq 3$)

Construct a mechanism where each agent reports (θ_i, x_i, k_i) — a type profile, an outcome, and an integer. The outcome rule:

1. If all agents report the same (θ, x) with $x \in F(\theta)$ and all $k_i = 0$: outcome is x.
2. If all but agent j agree on (θ, x) and agent j deviates: if j "challenges" by naming y with $u_j(y, \theta_j) \leq u_j(x, \theta_j)$ (the outcome is still x); otherwise, the outcome is determined by a tie-breaking rule that favors agent j's challenge iff the challenge reveals a preference reversal inconsistent with the claimed θ.
3. If there is no consensus, use an integer game: the agent who named the highest k_i dictates the outcome.

The integer game ensures that no "bad" equilibrium can survive: if agents coordinate on the wrong (θ, x), some agent can deviate (raise their integer) and dictate a preferred outcome. The NVP condition ensures unanimity is respected. The details are intricate but the structural idea is clean: the mechanism is designed so that the only stable coordination points are the correct ones.

The integer game is often criticized as "unrealistic" — no real-world mechanism includes a pure integer-naming component. This criticism misses the structural point: the integer game is a proof device, not a design recommendation. It demonstrates *existence* of an implementing mechanism. Practical mechanism design replaces the integer game with natural economic structures (sequential bidding, challenge rounds, appeals processes) that serve the same equilibrium-pruning function without the artificial flavor.

Bayesian implementation

Full Bayesian implementation (Jackson 1991, Postlewaite and Schmeidler 1986) extends the analysis to incomplete information. The requirement: every BNE of the mechanism yields an outcome in $F(\theta)$ for each θ. The necessary conditions are more complex than Maskin monotonicity, involving **Bayesian monotonicity** — a condition on how the SCF responds to changes in beliefs, not just preferences. Full Bayesian implementation is harder to achieve than full Nash implementation because the designer must contend with agents' prior beliefs as an additional degree of freedom.

Maskin monotonicity and common solution concepts

Solution concept	Necessary condition	Sufficient conditions
Nash (complete info)	Maskin monotonicity	Monotonicity + NVP ($n \geq 3$)
Subgame-perfect (complete info)	Weaker: "condition α"	Condition $\alpha + n \geq 2$ (Moore–Repullo)
Bayesian Nash	BIC (revelation principle suffices for partial)	Full Bayesian implementation: more complex (Jackson 1991)

Subgame-perfect implementation (Moore–Repullo)

For $n \geq 2$, the mechanism is a sequential game: 1. Agent 1 announces a type profile $\hat{\theta}$ and an outcome $x \in F(\hat{\theta})$. 2. Agent 2 can either agree (outcome is x) or challenge by naming an alternative type profile $\hat{\theta}'$ and outcome y. 3. If challenged, the mechanism tests the claim by asking agents to reveal preferences in a specific way (e.g., choosing between carefully constructed lotteries). 4. The sequential structure ensures that the only SPE involves truthful announcements.

The key insight: challenges are credible because the mechanism rewards challengers who correctly identify a false claim. The extensive form provides the "commitment" structure that the normal form lacks.

Examples of SCCs that fail Maskin monotonicity

Pareto correspondence with $n \geq 3$ and $|X| \geq 3$: the set of Pareto-efficient alternatives. This *is* Maskin-monotone and NVP, hence Nash-implementable.

Walrasian equilibrium correspondence (competitive equilibrium in exchange economies): generally *not* Maskin-monotone, hence not Nash-implementable. The failure comes from the price-dependence of budgets: improving an agent's preferences for a good can change the equilibrium price, which changes what other agents can afford, potentially removing the original allocation from the equilibrium set.

Majority rule with $|X| = 3$: can fail Maskin monotonicity due to Condorcet cycles. If the social choice at θ is a (majority winner), a preference change that only improves a can disrupt the majority ranking of other alternatives, creating a new cycle where a is no longer the winner.

The gap between partial and full implementation

The difference between partial and full implementation is quantitatively significant. In many environments, the set of BIC-implementable (partial) SCFs is large — the revelation principle imposes only local IC conditions. The set of Nash-implementable (full) SCFs is much smaller, because Maskin monotonicity is a *global* condition on how the SCF responds to preference changes. The gap is the set of SCFs that can be partially but not fully implemented — mechanisms where truth-telling is *an* equilibrium but bad equilibria also exist and cannot be eliminated.

This gap motivates the move to subgame-perfect implementation, which closes much of it. The Moore–Repullo result shows that with extensive-form mechanisms, almost any SCC satisfying a weak necessary condition can be fully implemented in SPE. The cost is mechanism complexity: SPE-implementing mechanisms often involve elaborate multi-stage games with challenges, counter-challenges, and tie-breaking rules. In practice, the simplicity of direct mechanisms (partial implementation) often wins over the robustness of Maskin-style mechanisms (full implementation).

Virtual implementation

Matsushima (1988) and Abreu–Sen (1991) showed that if the designer can use lotteries over outcomes, the requirements for full implementation weaken dramatically. **Virtual implementation** asks only that the mechanism produce the correct outcome with probability arbitrarily close to 1 (not exactly 1). Under virtual implementation, any SCF satisfying a condition called "measurability" is implementable in iteratively undominated strategies. This is a much weaker requirement than Maskin monotonicity, and it essentially resolves the impossibility at the cost of accepting a small probability of the wrong outcome.

Worked Example

Nash implementation of the Pareto correspondence

Two agents, three outcomes $X = \{a, b, c\}$. Preferences:

Type profile θ: Agent 1 ranks $a \succ b \succ c$, Agent 2 ranks $b \succ a \succ c$. Pareto set: $F(\theta) = \{a, b\}$ (both Pareto-dominate c; neither dominates the other).

Type profile θ': Agent 1 ranks $a \succ c \succ b$, Agent 2 ranks $b \succ a \succ c$. Pareto set: $F(\theta') = \{a, b\}$.

Check Maskin monotonicity for outcome a. At θ, $a \in F(\theta)$. Agent 1's lower contour set of a at θ: $\{a, b, c\}$ (everything, since a is top-ranked). At θ': still $\{a, b, c\}$ (still top-ranked). Agent 2's lower contour set of a at θ: $\{a, c\}$. At θ': $\{a, c\}$ (same). Lower contour sets have not shrunk, so monotonicity requires $a \in F(\theta')$. Confirmed: $a \in \{a, b\}$.

Mechanism for implementation. Using Maskin's construction: each agent reports a type profile and an outcome. If they agree on (θ, x) with $x \in F(\theta)$, outcome is x. If they disagree, a modular game determines the outcome based on which agent's claim is consistent with the preferences revealed by the disagreement pattern. For $n = 2$, the mechanism must be carefully tailored (the Dutta–Sen extension applies).

Failure of Maskin monotonicity

Two agents, three outcomes. Preferences:

Type θ: Agent 1: $a \succ b \succ c$. Agent 2: $b \succ a \succ c$. SCC: $F(\theta) = \{a\}$ (agent 1 dictates, say).

Type θ': Agent 1: $a \succ b \succ c$ (unchanged). Agent 2: $a \succ b \succ c$ (now a is improved). $F(\theta') = \{b\}$ (some other rule picks b).

Maskin monotonicity fails: $a \in F(\theta)$, lower contour sets of a have not shrunk (agent 2 now ranks a even higher), yet $a \notin F(\theta')$. This SCC is not Nash-implementable.

Poker Anchor

Concept mapping

Formal object	Poker instantiation
Full implementation	Every possible "equilibrium" of the poker format produces the intended outcome (fair competition)
Bad equilibria	Soft-play agreements, chip-dumping, or implicit collusion that are self-enforcing but undesired
Maskin monotonicity	If a rule works when players have certain preferences, it still works when those preferences only strengthen
No veto power	If all but one player agree on the outcome, the mechanism respects this
Subgame-perfect implementation	Using the sequential structure of poker (betting rounds, challenges) to refine away bad outcomes

Concrete situation: chip-dumping as a bad equilibrium

In a tournament, two friends can agree that one will "dump" chips to the other (intentionally losing pots) to give one a dominant stack. This is a Nash equilibrium of the tournament mechanism: neither player profits

by deviating unilaterally (the dumper would have to play normally, losing the coordination gain; the receiver would lose the free chips). But it is not the intended outcome of the mechanism.

The tournament rules attempt *full implementation*: the mechanism should produce fair competition regardless of which equilibrium the players coordinate on. Anti-collusion rules, table changes, hand reviews by floor staff — these are the poker analogue of Maskin's mechanism, designed to eliminate bad equilibria. The sequential structure of the tournament (repeated hands, changing table assignments, floor interventions) is the analogue of subgame-perfect implementation: the extensive form provides the "challenge" structure that disrupts collusion.

Concrete situation: format design as implementation

Consider the choice between a standard tournament and a shootout (single-table until one winner, then table winners face off). Both formats implement "the best player wins" as a partial equilibrium, but they have different *sets* of equilibria. In a standard tournament, bubble dynamics create bad equilibria (coordinated tightening). In a shootout, each table is independent, eliminating cross-table coordination but potentially introducing table-draw luck. The format designer choosing between these is solving an implementation problem: which mechanism has the smallest set of bad equilibria?

What this understanding buys you

Recognizing that "the rules should prevent this" is an implementation theory problem. When you see soft-play at a final table and think "the rules should be designed to prevent this," you are asking for full implementation rather than partial implementation. The revelation principle (partial implementation) says a truthful equilibrium exists; implementation theory asks whether the mechanism can *force* all equilibria to be fair.

Sequential poker structure is a design asset. The multi-round, multi-hand structure of poker is not just a feature of the game — it is a tool for full implementation. Each betting round is a "challenge" opportunity: if someone is playing suspiciously (chip-dumping), the observable betting patterns allow other players and the floor to detect and punish. This is Moore–Repullo in practice: the extensive form refines away bad equilibria that the normal form cannot eliminate.

The monotonicity check is a format-robustness test. When evaluating a new poker format, ask: if players' preferences for winning strengthen (they care more about winning), does the format still produce the right outcome? If a format breaks when players try harder (Maskin monotonicity fails), the format is poorly designed.

Bad equilibria are the structural explanation for "angle shooting." An angle shoot is a strategy that exploits ambiguity in the rules — it is a Nash equilibrium of the mechanism (no single rule violation) but produces an outcome the designer did not intend. The angle shooter is playing a "bad equilibrium." Full implementation would design the rules so that no angle shoot is an equilibrium. In practice, poker rules are supplemented by floor rulings (the "challenge" step in Moore–Repullo), which is sequential implementation in action: the floor observes the behavior, evaluates the claim, and rules on whether it was within the intended equilibrium set.

The integer game is the "any player can call the floor" rule. In Maskin's mechanism, the integer game prevents coordination on bad outcomes: any agent can unilaterally break a bad equilibrium by "raising their integer." In poker, the analogue is the rule that any player can call the floor at any time. This unilateral challenge right disrupts collusive equilibria by giving each player the power to invoke an external authority. The floor's ruling is the mechanism's tie-breaking rule. Without this challenge right, collusion (a bad equilibrium) would be much harder to dislodge.

Cross-Links

- **6.1 Social choice and IC** — the IC framework that partial implementation works with.

- **6.2 Revelation principle** — the partial-implementation baseline that this node extends.
- **3.4 Subgame-perfect equilibrium** — the refinement used in SPE implementation.
- **2.1 Nash equilibrium** — the solution concept for Nash implementation.
- **5.3 Trigger strategies** — repeated-game mechanisms that achieve full implementation over time.
- **10.4 Commitment and credibility** — institutional structures that support implementation.

Structural Compression

Invariant: Full Nash implementation requires Maskin monotonicity (necessary) and no veto power (sufficient with monotonicity for $n \geq 3$). Subgame-perfect implementation relaxes monotonicity and is achievable for a much broader class of SCCs by exploiting the sequential structure of extensive-form mechanisms.

Mechanism: The Maskin mechanism uses consensus-and-challenge: if all agents agree, implement the agreed outcome; if one deviates, test the deviation against the SCC's structure; if no consensus, use an integer game to prevent coordination on wrong outcomes. SPE mechanisms use sequential challenges and backward induction to refine away bad Nash equilibria.

Cross-System Echo: Full implementation is the game-theoretic analogue of **robust control** in systems engineering. Partial implementation guarantees the system works for *some* disturbance (equilibrium); full implementation guarantees it works for *all* disturbances. Maskin monotonicity is the analogue of a **monotone Lyapunov condition** — a structural property of the objective function that ensures stability is preserved under perturbation. The Moore–Repullo sequential mechanism is the analogue of a cascade controller that refines behavior at each stage.

Exercises

1. Verify that the Pareto correspondence (the set of Pareto-efficient alternatives) satisfies Maskin monotonicity. Then verify NVP for $n \geq 3$. Conclude that it is Nash-implementable.
2. Show that the social choice function "always pick alternative a" is trivially Maskin-monotone. Is it Nash-implementable? If so, construct a simple implementing mechanism.
3. Construct a two-agent, three-outcome example where the SCC satisfies Maskin monotonicity but not NVP (note: with $n = 2$, NVP is not sufficient anyway). Discuss what additional conditions are needed for two-agent implementation.
4. Consider a two-agent mechanism where Agent 1 proposes an outcome, Agent 2 can accept or challenge, and challenges are resolved by a coin flip between the proposed outcome and Agent 2's preferred outcome. Characterize the set of SPE outcomes and determine which SCCs this mechanism can implement.
5. Show that the Walrasian equilibrium correspondence in a two-good exchange economy can fail Maskin monotonicity. (Hint: construct two preference profiles where Agent 1's preferences for good 1 improve, but the equilibrium price changes enough to shift Agent 2's budget constraint and alter the equilibrium allocation.)
6. In a three-player voting game, show that simple majority rule over three alternatives $\{a, b, c\}$ can produce Condorcet cycles, and that the majority-rule SCC therefore fails Maskin monotonicity at specific preference profiles. Identify the profile.
7. **Argue, structurally,** why subgame-perfect implementation is strictly more permissive than Nash implementation — that is, why the class of SPE-implementable SCCs strictly contains the class of Nash-implementable SCCs. Identify the structural feature of extensive-form mechanisms that eliminates bad equilibria that survive in normal-form mechanisms.

Game Theory · Module 6 — Mechanism Design · Node 6.5
Concepts: mechanism-design, nash-equilibrium, social-choice
Prerequisites: 6.1, 6.2
Enables: 6.6

hbar.university — own.your.cognition