



**HAL**  
open science

# A Foundational Framework for the Specification and Verification of Mechanism Design

Pierre Jouvelot, Emilio Jesús Gallego Arias

► **To cite this version:**

Pierre Jouvelot, Emilio Jesús Gallego Arias. A Foundational Framework for the Specification and Verification of Mechanism Design. EC'22 - Twenty-Third ACM Conference on Economics and Computation, Jul 2022, Colorado, United States. . hal-03715847

**HAL Id: hal-03715847**

**<https://minesparis-psl.hal.science/hal-03715847v1>**

Submitted on 6 Jul 2022

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



# A Foundational Framework for the Specification and Verification of Mechanism Design

P. Jouvelot<sup>1</sup>, E. J. Gallego Arias<sup>2</sup>

<sup>1</sup>Mines Paris, PSL University, Paris, France

<sup>2</sup>Université Paris Cité, CNRS, Inria, IRIF, F-75013, Paris, France



ACM EC 2022, July 11-15, Poster Session

## Abstract

We introduce **mech.v**, an 6000-line open-source library of formally defined core mechanisms and concepts of **mechanism design** [1, 2]. It builds upon the widely-used **Coq proof assistant** and the Mathematical Components (MC) library to provides a strong infrastructure from which sound, formally **machine-verified** existing and new mechanisms can be developed, helping increase the confidence the services they offer are well-founded.

## Coq, a type-based proof assistant

Coq [3] is a mathematical proof checker based on a powerful **type theory** in which expressions  $e$  have types  $t$  that denote logical properties, written as type properties “ $e : t$ ”. For instance, one can express the infinitude of primes using the following type property, expressed as a lemma:

**Lemma** `prime_above` : forall m, exists p, m < p ∧ prime p.

To help proof developers construct expressions of a sought-after type, i.e., property, Coq uses an **interactive approach** based on typing rules and tactic languages. A large number of existing libraries such as Mathematical Components enable a **high level of abstraction** to be used for mathematical reasoning (e.g., the definition of *prime*, in MC).

## Bibliography

1. Jouvelot, P., and Gallego Arias, E. J. *A Foundational Framework for the Specification and Verification of Mechanism Design*. Mines Paris Tech. Rep., PSL University, 2022 (<https://www.cri.ensmp.fr/classement/doc/E-458.pdf>)
2. <https://github.com/jouvelot/mech.v>
3. Bertot, Y., and Castéran, P. Springer Science & Business Media, 2013. *Interactive theorem proving and program development: Coq'Art: the calculus of inductive constructions*
4. Gallego Arias, E. J., Pin, B., and Jouvelot, P. *jsCoq: Towards Hybrid Theorem Proving Interfaces*. In Proceedings of the 12th Workshop on User Interfaces for Theorem Provers, Coimbra, Portugal, July 2016.
5. Barthe, G., Gaboardi, M., Arias, E.J.G., Hsu, J., Roth, A., Strub, P.Y. (2016). *Computer-Aided Verification for Mechanism Design*. In: Cai, Y., Vetta, A. (eds) Web and Internet Economics. WINE 2016. Lecture Notes in Computer Science, vol 10123. Springer, Berlin, Heidelberg. [https://doi.org/10.1007/978-3-662-54110-4\\_20](https://doi.org/10.1007/978-3-662-54110-4_20)

## Contributions

1. **Reference.** The `mech.v` library includes the specifications and correctness theorems (e.g., truthfulness, rationality, or dominance) of well-known mechanisms and auctions such as the Vickrey-Clarke-Groves (VCG) mechanism, the VCG auction for online ad or First Price and Second Price auctions.
2. **Assurance.** To benefit from the additional assurance that a mechanized proof brings compared to a pen-and-paper proof outline (an issue of increasing importance, given the exponential growth in the number of mathematical papers reviewed and published each year), we borrow a technique from the computer science literature, **proof by refinement**, to prove that the implementation of VCG for online ads auctions is a refinement of the VCG mechanism, thus transferring its general truthfulness property for (almost) free.
3. **Extension;** Since Coq proofs are simple syntactic objects, the agents participating in a mechanism can opt to trust a **reference proof checking implementation different** from our's to check for themselves that a particular strategy is, for instance, dominant, increasing their trust in the mechanism. In turn, this added confidence may allow the mechanism designer to use more complex design methods that remain truthful while providing better welfare for all.

## Example: VCG in mech.v

A direct Coq specification of the General VCG mechanism involving  $n$  agents, where  $O$  denotes the type of outcomes  $o$ ,  $A$ , the type of agents  $i$ , and  $\{ffun O \rightarrow nat\}$ , the type of functions from  $O$  to natural numbers (the bidding action of one agent), can be written as follows.

**Variable** ( $O : finType$ ) ( $o0 : O$ ) ( $i : A$ ).

**Definition** `bidding` := {ffun  $O \rightarrow nat$ }. (\* from outcomes to bids \*)

**Definition** `biddings` := n.-tuple `bidding`.

**Variable** ( $bs$ : biddings).

**Local Notation** "`bidding_j`" := (tnth  $bs$   $j$ ) (at level 10).

**Implicit Types** ( $o : O$ ) ( $bs : biddings$ ).

**Definition** `bidSum`  $o$  := \sum\_( $j < n$ ) ' `bidding_j`  $o$ .

**Definition** `bidSum_i`  $o$  := \sum\_( $j < n \mid j \neq i$ ) ' `bidding_j`  $o$ .

**Definition** `oStar` := [argmax\_( $o > o0$ ) (bidSum  $o$ )].

**Definition** `welfare_with_i` := bidSum\_i `oStar`.

**Definition** `welfare_without_i` := \max\_o bidSum\_i  $o$ .

**Definition** `price` := welfare\_without\_i – welfare\_with\_i.

This specification, close to its mathematical definition, can be used to see VCG as an instance,  $m$ , of a general mechanism, of type `mech.type`, as introduced in `mech.v` (see [1] for details).

**Definition** `m` : mech.type  $n$  :=  
mech.new (fun  $bs \Rightarrow$   
(map\_tuple (fun  $i \Rightarrow$  price  $o0$   $i$   $bs$ ) (agent.agents  $n$ ),  
oStar  $o0$   $bs$ )).

VCG's truthfulness theorem can then be specified and easily proven in Coq via the general definition, *truthful*, of truthfulness provided in `mech.v` (see [1]).

**Lemma** `truthful_General_VCG` : truthful  $p$ .

where  $p$ , the preferences, or profile, of each agent, of type `prefs.type`, specifies the true value strategy, current strategy and utility function for  $m$  (see [1]).

## Future Work

Improvements to the current setting may include:

- adding other notions related to deterministic mechanisms;
- tackling voting or probabilistic mechanisms [5], or
- making it an “executable” textbook for economy-education purposes, following the “literate programming” movement [4].