

# Reasoning about Auctions\*

Marco Caminati<sup>1</sup>

Manfred Kerber<sup>1</sup>

Christoph Lange<sup>1,2</sup>

Colin Rowat<sup>3</sup>

<sup>1</sup> Computer Science, University of Birmingham, UK

<sup>2</sup> Fraunhofer IAIS and University of Bonn, Germany

<sup>3</sup> Economics, University of Birmingham, UK

Project homepage: <http://cs.bham.ac.uk/research/projects/formare/>

**Abstract:** In the ForMaRE project formal mathematical reasoning is applied to economics. After an initial exploratory phase, it focused on auction theory and has produced, as its first results, formalized theorems and certified executable code.

## 1 Introduction

An auction mechanism is mathematically represented through a pair of functions  $(a, p)$ : the first describes how some given goods at stake are allocated among the bidders (also called participants or agents), while the second specifies how much each bidder pays following this allocation. Each possible output of this pair of functions is referred to as an *outcome* of the auction. Both functions take the same argument, which is another function, commonly called a *bid vector*  $\mathbf{b}$ ; it describes how much each bidder values the possible outcomes of the auction. This valuation is usually expressed through money.

In this setting, some common questions are the study of the quantitative and qualitative properties of a given auction mechanism (e.g., whether it maximizes some relevant quantity, such as revenue, or whether it is efficient, that is, whether it allocates the item to the bidder who values it most), and the study of the algorithms running it (in particular, their correctness).

In the following three sections we will see three important cases of auctions for which we have proved theorems and extracted verified code using the Isabelle/HOL proof assistant.

## 2 Single-good static auctions

In the simplest case there is exactly one indivisible good auctioned in a single round of bidding. As a consequence, the formal details are simple: the allocation function  $a$  takes only two possible values, 1 and 0, depending on whether a participant receives the good or not, respectively. Its argument, the bid vector, is a map from the set of participants to the non-negative reals (or naturals, according to the kind of numbers chosen to represent money). The payoff for a participant is given by a simple algebraic expression (where  $v$  is the actual valuation of the good according to that participant):  $v * a(\mathbf{b}) - p(\mathbf{b})$ . An important auction is Vickrey's auction in which the good is allocated to the highest bidder, who pays the second-highest price. In this situation Vickrey's theorem holds, one of the most important theorems in auction theory: no participant can be better off upon bidding anything different from her actual valuation of the

good; this holds independently of how the other participants behave. We formalized Vickrey's theorem in several proof assistants to get an idea of their suitability for auctions [2].

The subsequent result of ForMaRE was to extend this theorem, using Isabelle/HOL, in two ways: proving the inverse of this result, and characterizing the class of all the mechanisms (called generalized Vickrey) enjoying this truthful bidding properties [3, Theorem 2].

Finally, a further important result was formalized in Isabelle: that it is impossible to achieve budget balancing for the generalized Vickrey mechanism, [3, Theorem 3]. This means that there will always be some acceptable bid vector giving an outcome for which the sum of the total payments will be non-zero. Such a result is another standard of auction theory; however, our proof is original and has the distinctive feature of not relying on the specific form of the generalized Vickrey mechanism, thereby establishing an algebraic property of a wide class of mechanisms.

## 3 Multiple-good static auctions

An important generalization of the single-good case (§2) is that of a set of several goods at stake, but with the important proviso that participants do not bid on each item independently, but rather bid on *subsets* of items. That is, they bid on combinations of items, e.g., allowing them to express interest for multiple items which are worth to them only when they are together (a left shoe *and* its right shoe); or allowing participants to express the same preference between distinct subsets of goods of which they need only one.

In this setting, there is a mechanism enjoying properties similar to those enjoyed by the Vickrey mechanism (§2). It is called nVCG ( $n$  goods following the mechanism of Vickrey, Clarke, and Groves). Determining the outcome  $(a, p)$  of such a second price, combinatorial auction is much more complex (indeed, NP-complete) [1, Chapter 12], since for each possible allocation, the total revenue has to be computed in order to find the maximum. This gives the value of  $a$ ; then, a similar computation has to be performed to determine  $p$ .

In ForMaRE we have extracted executable Scala code for this from the Isabelle formalization. This allows to provide certified soundness properties for the code. In particular, the prices are non-negative, the outcome of the auction al-

\*This work has been supported by EPSRC grant EP/J007498/1.

locates each good exactly once, and for each possible bid vector there is exactly one outcome (up to tie-breaking).

Producing the formalization of theorems and automatically generated Scala code has been the fundamental accomplishment of the project. One of the main goals is now to better integrate these two efforts. Indeed, at the moment, some formalized theorems only apply to the simplest among the settings for which we can generate Scala code. Notably, while we can extract code to run any combinatorial VCG auction, the proofs for the Vickrey characterization and budget imbalance theorems are currently restricted to the special case of single-good auctions.

Generalizing those results from the single-good case to the combinatorial case is not trivial, because the allocation function gets to yield values more complex than the numbers 0,1. Correspondingly, the definition of payoff has to be modified into  $v(a(b)) - p(b)$ , and the simple second price rule has to be reformulated.

#### 4 Dynamic auctions

A possible goal in designing a mechanism is to allow participants to refine their valuations about the goods at stake and in general the information they have about the possible outcomes. One common way to achieve this is to run dynamic auctions: the goods are not allocated in a single round, but multiple bidding rounds are run until some condition is met (e.g., nobody raises her bid any longer). This inevitably increases the risk of setting up an ill-specified auction (e.g., by specifying one that can never stop). Hence, formal methods get even more useful.

There are some problems for the adoption of formal methods in this setting:

1. A dynamic auction inherently requires repeating an input/output phase from participants, which typically happens through hardware and software the designer has no control on (e.g., on a remote machine).
2. The functional Isabelle/HOL formalization we used to generate Scala code has no notion of time. Hence, to actually run an auction, we wrap the Isabelle-generated code in a manually written Scala snippet providing access to the hardware clock of the machine.

For the last issue, the idea is to make the manually written wrapper as thin as possible. A `while` loop is all one needs to get access to the clock: both the termination condition and the code executed in each round can and should then be generated by Isabelle. Besides the skeleton of the `while` loop, the remaining manually written code should only concern input/output (point 1. above): we tested such a minimal wrapper loop around the Isabelle-generated code. It is eight lines of code, in the file `Dyna.scala` available in the GitHub repository linked from our homepage. After this loop, the last bid vector is passed to a second stage determining the outcome according to  $a$  and  $p$ , for which we can use the existing code of static auctions (see § 3).

#### 5 Conclusion

The ForMaRE project has been working on three themes: theorems for single-good static auctions, verified code for multiple-good combinatorial static auctions, and verified code for dynamic auctions. These themes all present potential for further work, suggesting three dimensions along which to expand the project, building on the formalization already created, and using it as a guidance. As we mentioned in section 3, combining the first two themes, we are studying the extensions of theorems from section 2 to the combinatorial auctions of section 3. But this will only be a first step, because there are much more complex auctions of theoretical and practical relevance: there are interesting properties of a mechanism which depend on the information and beliefs of the single participant, and not merely on the specification of the mechanism itself. For example, a possible desirable goal in designing an auction would be that each participant submits a bid close to her perception of the value of the good, rather than a strategic lie to influence the outcome; or even to allow each participant to form or refine such a perception. To do that, the theory must be enriched to more expressively model the participants themselves, rather than only the mechanism. This has been accomplished in a subfield of game theory, mechanism design (also known as reverse game theory), by introducing the so-called type profile  $t^i$  of a given participant  $i$ . It models the participant's information, beliefs and preferences, so that in general the *payoff*  $u^i$  of the participant is a function of both the outcome *and* the profile. Our current machinery lacks the notion of profile, and introducing it will move our whole approach beyond the generalization to combinatorial auctions, granting us the possibility to formalize and investigate the vast range of results of reverse game theory.

The theme of dynamic auctions is of practical relevance and presents challenging possible evolutions: for example, how to regulate and implement the possibility for any participant (including the current winner) to withdraw her bid, and how to handle the feedback given to each participant after each bidding round. This latter point will need more complex interfacing between the dynamic stage and the existing winner determination code.

#### References

- [1] Peter Cramton, Yoav Shoham, and Richard Steinberg, editors. *Combinatorial auctions*. MIT Press, 2006.
- [2] Christoph Lange et al. A qualitative comparison of the suitability of four theorem provers for basic auction theory. In *Intelligent Computer Mathematics: MKM, Calculemus, DML, and Systems and Projects 2013*, pages 200–215. Springer-Verlag, 2013.
- [3] Eric Maskin. The unity of auction theory: Milgrom's master class. *Journal of Economic Literature*, 42(4):1102–1115, December 2004.