A Qualitative Comparison of the Suitability of Four Theorem Provers for Basic Auction Theory

MKM@CICM 2013

Christoph Lange[1], Marco B. Caminati[2], Manfred Kerber[1], Till Mossakowski[3], Colin Rowat[1], Makarius Wenzel[4], Wolfgang Windsteiger[5]

[1]University of Birmingham, UK
[2]http://caminati.net.tf, Italy (**Mizar**)
[3]University of Bremen and DFKI GmbH Bremen, Germany (**Hets/CASL**)
[4]Univ. Paris-Sud, Laboratoire LRI, Orsay, France (**Isabelle**)
[5]RISC, Johannes Kepler University Linz (JKU), Austria (**Theorema**)
http://cs.bham.ac.uk/research/projects/formare/code/auction-theory/

Motivation

- **auction theory** as a **representative** for formalising **economics**
  - **mechanism design** close to social choice theory
    (where mechanised reasoning has been applied successfully)
  - auction theory $\subseteq$ mechanism design
  - practically relevant ($\rightarrow$ next slides)
- Vickrey's theorem as a canonical representative
- Question: **which systems are suitable** for auction theory?
  Our approach: approximate the answer
  by formalising Vickrey's theorem

The Ideal Mechanised Reasoner for Auctions

- **library** as versatile as in Isabelle or Mizar
- **prover** as efficient as Isabelle or Mizar
- **error messages** as informative as in Isabelle's jEdit GUI
- **proof language** as close to textbook style as Isabelle or Mizar (for fully automated systems: proof **exploration interface** as informative as Theorema's)
- **textbook-like term syntax** like Theorema
- integration of diverse **tools** like Isabelle or Hets
- **community** as lively as Isabelle's

Auctions

- **mechanism** for
  - **allocating** electromagnetic spectrum, airplane landing slots, bus routes, oil fields, bankrupt firms, works of art, eBay items
  - establishing exchange rates, treasury bill yields
  - determining opening prices in stock exchanges
- challenges:
  - **finding right auction** form for an allocation goal
  - maximising revenue (3G spectrum: governments earned between €20 and €600 per capita)
  - **efficient allocation**, prevent monopolies
  - Is my auction **well-defined**?

Enabling Auction Designers to Formalise

- mechanised reasoning in economics:
  so far only done by computer scientists
- enable auction designers to verify their own designs
  by building an **Auction Theory Toolbox (ATT)**
  `http://cs.bham.ac.uk/research/projects/formare/code/auction-theory/`
- goals of our ForMaRE research project beyond auctions:
  1. **increase confidence** in economics' theoretical results
  2. aid in the **discovery of new results** (also in matching, finance: see our S&P paper)
  3. **foster interest in formal methods** within economics
  4. collect **user experience feedback** from new audiences
  5. contribute **challenge problems** to computer science

Auction Designers' Requirements

1. provide **ready-to-use** formalisations of **basic auction concepts**
2. allow for **extension** and application to **custom-designed auctions**
3. provide **easy access** to mechanised reasoning systems

Computer Scientist's Requirements

1. Identify right **language to formalise** auction theory:
   1. **expressiveness** vs. **efficiency**
   2. use familiar **textbook notation**
   3. provide **libraries** of relevant mathematical foundations.
2. Identify a **mechanised reasoning system**
   1. that **assists** users with developing formalisations,
   2. that facilitates **reuse** of formalisations existing in toolbox,
   3. that creates **comprehensible output**, and
   4. whose **community is supportive** towards new users.

Note the **conflicts of interest**!

Approach to Building the Toolbox

- avoid chicken-and-egg problem
  $\Rightarrow$ build ATT while identifying suitable languages/systems
- identifying languages/systems requires having domain problems
- we take problems from Maskin's review paper of Milgrom's canonical auction theory textbook [Mas04]

Vickrey's Theorem

Static second-price auction: everyone submits one sealed bid, highest bidder wins, pays highest *remaining* bid.

Theorem (Vickrey 1961)

In a second-price auction, "truth-telling" (i.e. submitting a bid equal to one's actual valuation of the good) is a weakly dominant strategy. Furthermore, the auction is efficient.

- earliest result in modern auction theory
- simple environment for gaining intuition

Vickrey's Theorem (Elaborated towards Formalisation)

Definition (Weakly Dominant Strategy)

Given some auction, a strategy profile $\boldsymbol{b}$ supports an *equilibrium in weakly dominant strategies* if, for each $i \in N$ and any $\widehat{\boldsymbol{b}} \in \mathbb{R}^n$ with $\widehat{b}_i \neq b_i$, $u_i\left(\widehat{b}_1, \ldots, \widehat{b}_{i-1}, b_i, \widehat{b}_{i+1}, \ldots, \widehat{b}_n\right) \geq u_i\left(\widehat{\boldsymbol{b}}\right)$. I.e., whatever others do, i will not be better off by deviating from the original bid b_i.

Theorem (Vickrey 1961; Milgrom 2.1)

In a second-price auction, the strategy profile $\boldsymbol{b} = \boldsymbol{v}$ supports an equilibrium in weakly dominant strategies. Furthermore, the auction is efficient.

Vickrey's Theorem (Proof Sketch)

Suppose participant i bids truthfully, i.e. $\left(\widehat{b}_1, \ldots, \widehat{b}_{i-1}, v_i, \widehat{b}_{i+1}, \ldots, \widehat{b}_n\right) =: \widehat{\boldsymbol{b}}^{i \leftarrow v}$.

1. i **wins** . . .
   Now consider i submitting an arbitrary bid $\widehat{b}_i \neq b_i$, i.e. assume an overall bid vector $\widehat{\boldsymbol{b}}$.
   1. i **wins with the new bid** . . .
   2. i **loses with the new bid** . . .
2. i **loses** . . .
   1. i **wins with the new bid** . . .
   2. i **loses with the new bid** . . .

In each case, we obtain $u_i(\widehat{\boldsymbol{b}}) \leq u_i(\widehat{\boldsymbol{b}}^{i \leftarrow v})$.

Choosing a Mechanised Reasoning System

Systems differ in:

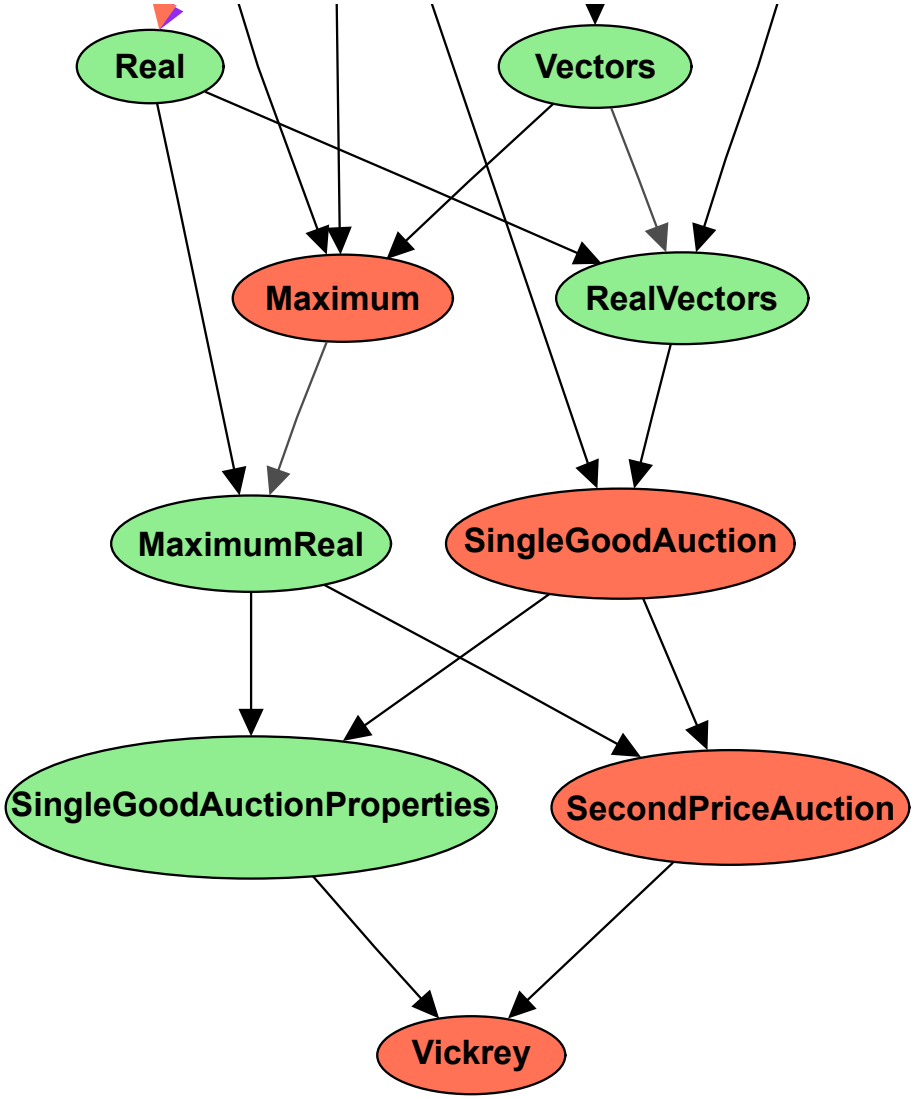
- **logic**: maximum of n bids $b_i \in \mathbb{R}$
  but proof structure is simple; no induction.
- **syntax**: some like textbook mathematics,
  others like programming language
- **user experience**: fully automated proving
  vs. proof checking
  vs. interactive proving

Mechanised Reasoning Systems we Used

Systems and state of our formalisations:

- **Mizar**: FOL + set theory, text editor, proof checker ✓
- **Isabelle/HOL**: higher-order logic (typed), interactive theorem proving environment, document-oriented IDE ✓
- **Hets/CASL/TPTP**: sorted FOL, text editor, proof management GUI, frontend to local or remote automated provers ✓
- **Theorema 2.0**: FOL + set theory, textbook-style documents (Mathematica notebooks), built-in automated provers, proof management GUI (✓)

Theory Structure



Level of Detail and Explicitness Required

- paper elaboration was detailed and explicit
  but systems need even more ($\geq$ 1.5 times as much code)
- benefits of explicitness: It becomes obvious that . . .
  - exactly one participant wins
  - a second-price auction requires at least 2 participants
    second-highest bid undefined otherwise
    alternative: define max $\varnothing := 0$

Expressiveness vs. Efficiency (Mizar)

Our Mizar formalisation uses low-level set theory:

- bid vector represented as a relation R
- natural numbers represented as sets: $0 := \varnothing$, $n + 1 := \{0, \ldots, n\}$

Advantages:

- basic set theory **well supported** in library
- certain operations are **elegant** and **concise**, e.g. max := $\bigcup$, and hence arg max as inverse image (") through R:

```
winnerof R equals the Element of R"{union rng R}
```

Disadvantages:

- **hard to read** for domain experts
  However, formalisation becomes clearer when proving lemmas with explicit assumptions such as "R is a function with range $\mathbb{N}$".
- Mizar does not support numbers and arithmetics natively
  (i.e. you *need* to represent numbers as sets)

Proof Development and Management

"Automated vs. interactive" difference blurs:

- Isabelle and Mizar give access to automated provers **workflow practically similar** to Theorema and Hets:
  1. specifying the knowledge to be used
  2. configuring the prover
- when **fully automated provers need guidance** (e.g. Theorema or Hets can't do $A \Rightarrow C$, but can do $A \Rightarrow_P B$ and $B \Rightarrow_{P'} C$):
  - these additional "proof steps" have to be emulated by lemmas
  - prover configuration has to be maintained separately from the formalisation

Term Input Syntax (Isabelle)

```
definition second_price_auction_winner ::
    "participants ⇒ bids ⇒ allocation ⇒ payments ⇒ participant ⇒ bool"
  where
    "second_price_auction_winner N b x p i ⟷
      i ∈ N ∧ i ∈ arg_max_set N b ∧ x i = 1 ∧ p i = maximum (N - {i}) b"

definition second_price_auction_loser ::
    "participants ⇒ allocation ⇒ payments ⇒ participant ⇒ bool"
  where "second_price_auction_loser N x p i ⟷ i ∈ N ∧
     x i = 0 ∧
     p i = 0"

definition spa_pred :: "participants ⇒ bids ⇒ allocation ⇒ payments ⇒ bool"
  where
    "spa_pred N b x p ⟷
      bids N b ∧
      (∃i ∈ N. second_price_auction_winner N b x p i ∧
        (∀j ∈ N . j ≠ i ⟶ second_price_auction_loser N x p j))"
```

Term Input Syntax (CASL)

```
forall n: Participants;
       x: Allocation;
       p: Payments;
       v: Valuations;
       b: Bids;
       winner, loser, i, maxBidder: Participant
. secondPriceAuctionWinner(n, b, x, p, i) <=>
  inRange(n, i) /\
  inArgMaxSet(b, i) /\
  allocated(x, b, i) /\
  payment(p, b, i) = secondPriceAuctionWinnersPayment(b, i) %(second_price_auction_winner_def)%
. secondPriceAuctionLoser(n, b, x, p, i) <=>
  inRange(n, i) /\
  not allocated(x, b, i) /\
  payment(p, b, i) = 0                                    %(second_price_auction_loser_def)%
. secondPriceAuction(n, x, p) <=>
  %[ (x, p) is the outcome of a second price auction with n participants iff … ]%
  n > 1 /\
  (forall b: Bids .
    length(b) = n
    =>
    allocation(b, x) /\
    payments(b, p) /\
    (exists i: Participant .
      inRange(n, i) /\
      secondPriceAuctionWinner(n, b, x, p, i) /\
      (forall j: Participant .
        inRange(n, j) /\
        not j = i
        =>
        secondPriceAuctionLoser(n, b, x, p, j))))          %(second_price_auction_def)%
```

Term Input Syntax (Theorema)

DEFINITION (SECOND-PRICE AUCTION)

$$\text{secondPriceAuction}[b, x, p] :\Longleftrightarrow \exists_{i=1,\ldots,n} \text{secondPriceAuctionWinner}[b, x, p, i] \bigwedge \forall_{\substack{j=1,\ldots,n \\ j\neq i}} \text{secondPriceAuctionLoser}[x, p, j] \qquad (\textit{secondPriceAuction})$$

$$\forall_{i=1,\ldots,n} \text{secondPriceAuctionWinner}[b, x, p, i] :\Longleftrightarrow b_i = \max_{j=1,\ldots,n} b_j \bigwedge x_i = 1 \bigwedge p_i = \max_{\substack{j=1,\ldots,n \\ j\neq i}} b_j \qquad (\textit{secondPriceAuctionWinner})$$

$$\text{secondPriceAuctionLoser}[x, p, i] :\Longleftrightarrow x_i = 0 \wedge p_i = 0 \qquad (\textit{secondPriceAuctionLoser})$$

Term Input Syntax (Mizar)

```
definition let R be Relation;

:: bids are cartesian products participants × offers
func topbiddersof R → Subset of dom R
  equals R"{union rng R};
  :: R"Y is the preimage of Y under R
func winnerof R equals the Element of topbiddersof R;
func losersof R equals dom R \ {winnerof R};
func priceof R equals union rng (R | losersof R);

:: allocation and payments for each participant
func R-allocat equals [:dom R,{0}:]+*[:{winnerof R},{1}:];
func R-pay
  equals [:dom R,{0}:]+*[:{winnerof R},{priceof R}:];
```

Other Comparison Criteria

- **library** coverage and searchability
  - *n*-argument maximum built in?
  - how to find reusable material?
- **comprehensibility** and trustability of the **output**
  - Why did a proof fail?
  - What was used in proving a non-trivial goal automatically?
  - A proof "succeeded" trivially; did we accidentally state a tautology?
- online **community support** and documentation

System Comparison

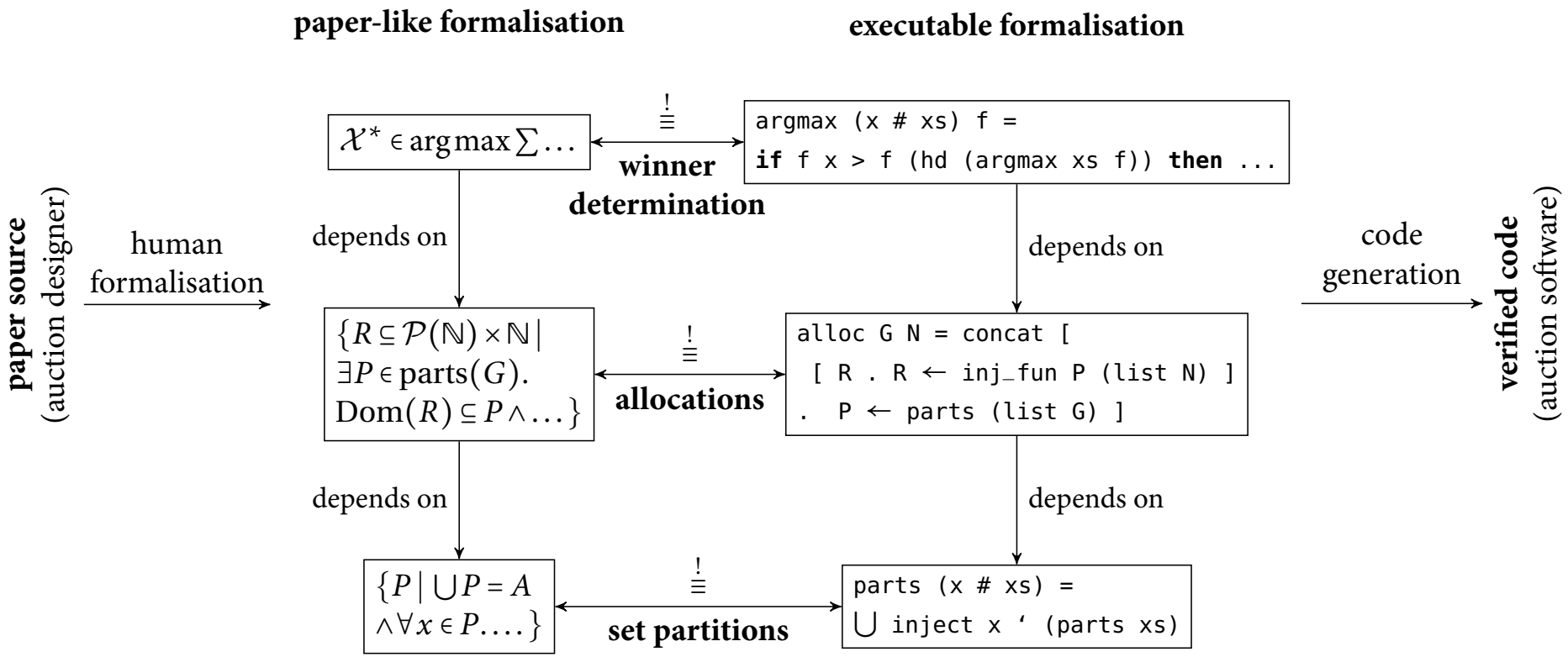
Our contribution so far:

- **recommending to auction designers what system to use**
  next: providing them with a growing library to build their formalisations on
- **giving systems' developers user experience feedback** from the field (new user group!)
  next: compile challenge problems

Formalising Auction Theory

- Our source [Mas04] contains 13 propositions and their proofs (overview on ATT homepage: `http://cs.bham.ac.uk/research/projects/formare/code/auction-theory/`)
- Bidding typically requires **forming conjectures of others' beliefs** $\Rightarrow$ integration over conditional density functions; calculation of second derivatives
- Maskin's review limited to single good auctions
  - **combinatorial auctions** are more economically critical (spectrum, monetary policy)
  - but few general results exist

Checking Well-Definedness of Combinatorial Auctions



Comparison result

System/ Language	Proof speed	Textbook closeness		Top-down proofs	Library		Output			Community	Documen-tation	de Bruijn factor
		PI[a]	TI[a]		LC[a]	LS[a]	PO[a]	CE[a]	WF[a]			
Isabelle/HOL	++[b]	++	+	++	++	++	○	++	++	++	++	1.3
Theorema	?	n/a[c]	++	++	+	––	++	n/a	–	––	–	n/a
Mizar	++	++	–	++	++	+	○	n/a	++	+	○	1.7
CASL/TPTP	○[d]	–	+	++	+	–	○	+	+	○	+	1.5

[a] PI/TI = proof/term input; LC/LS = library coverage/search; PO = proof output; CE = counterexamples (incl. consistency checks); WF = well-formedness check. [b] scores from very bad (––) to very good (++) [c] fully GUI-based [d] automated provers

Result specific to auctions? –
No, but the application orientation prioritised "soft" criteria!

etc.

- Robert Leese, who worked on the UK's spectrum auctions, has called for auction software to be added to the Verified Software Repository [Woo+09].
- other work in 'verifying' auction properties can be seen in our case checking paper - q.v. [Arc+05] and [Den+12], both described there

References I

Josep Ll. Arcos et al. "Engineering open environments with electronic institutions." In: *Engineering Applications of Artificial Intelligence* 18 (2005), pp. 191–204.

Louise A. Dennis et al. "Model checking agent programming languages." In: *Automated software engineering* 19.1 (2012), pp. 5–63.

Eric Maskin. "The unity of auction theory: Milgrom's master class." In: *Journal of Economic Literature* 42.4 (Dec. 2004), pp. 1102–1115. url: http://scholar.harvard.edu/files/maskin/files/unity_of_auction_theory.pdf.

References II

Jim Woodcock et al. “Formal method: practice and experience.” In: *ACM Computing Surveys* 41.4 (Oct. 2009), pp. 1–40.