Applying Mechanised Reasoning in Economics — Making Reasoners Applicable for Domain Experts

Manfred Kerber    Christoph Lange    Colin Rowat
Computer Science    Economics
University of Birmingham, UK

http://cs.bham.ac.uk/research/projects/formare/

Informatik 2013, Koblenz, Germany – 17 September 2013
supported by EPSRC grant EP/J007498/1
Overview

Motivation
Related work
Pillage games
Value at Risk
Matching markets
Auctions: importance, types, formal properties, toolbox, code generation
— coffee break —
Auctions: DEMO, soundness
Lessons for computer scientists
Problems not addressed
Future work
Summary
1996: Ariane 5 launch failure costs $370 million.

2012: Knight Capital’s high frequency trading software repeatedly sells shares below purchase price, loses $440 million within < 1 hr.

Economics software is mission critical, so it should be as reliable as possible!
Some Related Work
Social Choice Theory

How to aggregate individual preferences into a group preference in a fair way?

Example (Planning a Family Trip near Koblenz)

Father: Eltz Castle > Lorelei rock > Maria Laach Abbey
Mother: Maria Laach > Lorelei > Eltz
Daughter: Lorelei > Eltz > Maria Laach
Son: Eltz > Maria Laach > Lorelei

Is there a “fair” aggregation?

Source: Wikimedia Commons (see links)
Arrow’s impossibility theorem

A constitution respects **unanimity** (UN) if society puts alternative \( a \) strictly above \( b \) whenever every individual puts \( a \) strictly above \( b \). The constitution respects **independence of irrelevant alternatives** (IIA) if the social relative ranking (higher, lower, or indifferent) of two alternatives \( a \) and \( b \) depends only on their relative ranking by every individual. The constitution is a **dictatorship** (D) by individual \( n \) if for every pair \( a \) and \( b \), society strictly prefers \( a \) to \( b \) whenever \( n \) strictly prefers \( a \) to \( b \). [Gea05]

**Theorem (Arrow – 3 Proofs by Geanakoplos 2005)**

(For two or more agents, and three or more alternatives,) any constitution that respects transitivity, IIA, and UN is a D.
Arrow’s impossibility theorem (Cont’d)

- “Social choice theory turns out to be perfectly suitable for mechanical theorem proving. . . . However, it is unclear if this will lead to new insights into either social choice theory or theorem proving.” [Nip09]

- “we form an interesting conjecture and then prove it using the same [mechanized] techniques as in the previous proofs. . . . the newly proved theorem . . . subsumes both Arrow’s and Wilson’s theorems.” [TL09]

- “When applied to a space of 20 principles for preference extension familiar from the literature, this method yields a total of 84 impossibility theorems, including both known and nontrivial new results.” [GE11]

All of these are computer scientists!
Pillage Games
Pillage Games [Jor06]

Given a resource allocation $\mathcal{X} \equiv \{\{x_i\}_{i \in I} | x_i \geq 0, \sum_{i \in I} x_i = 1\}$, the following axioms can be defined. A power function $\pi$ satisfies

**WC (weak coalition monotonicity)**
if $C \subset C' \subseteq I$ then $\pi(C, x) \leq \pi(C', x) \forall x \in \mathcal{X}$;

**WR (weak resource monotonicity)**
if $y_i \geq x_i \forall i \in C \subseteq I$ then $\pi(C, y) \geq \pi(C, x)$; and

**SR (strong resource monotonicity)**
if $\emptyset \neq C \subseteq I$ and $y_i > x_i \forall i \in C$ then $\pi(C, y) > \pi(C, x)$. 
The Same in Theorema (WC) [KRW11]

WC (weak coalition monotonicity)
if $C \subset C' \subseteq I$ then $\pi(C, x) \leq \pi(C', x) \forall x \in \mathcal{X}$

Definition[“WC”, any[$\pi$, $n$], bound[allocation$_n[x]$],
$WC[\pi, n] : \iff n \in$
$\land \forall \forall \pi[C_2, x] \geq \pi[C_1, x] ]$
$C_1 \subset C_2 \land C_2 \subseteq I[n]$
Wealth Is Power

\[ \text{WIP}_\pi[C, x] := \sum_{i \in C} x_i \]
Wealth Is Power

\[ WIP_\pi[C, x] := \sum_{i \in C} x_i \]

Stable Set: 

\[ S = \begin{cases} 
(0, 0, 1), (0, 1, 0), (1, 0, 0), \\
(0, \frac{1}{2}, \frac{1}{2}), (\frac{1}{2}, 0, \frac{1}{2}), (\frac{1}{2}, \frac{1}{2}, 0), \\
(\frac{1}{4}, \frac{1}{4}, \frac{1}{2}), (\frac{1}{4}, \frac{1}{2}, \frac{1}{4}), (\frac{1}{2}, \frac{1}{4}, \frac{1}{4}), 
\end{cases} \]
Some Results [KRW11]

Formalization: Theorema 1. Represent the main definitions and results [KR12]

Proofs: Prove some theorems in Theorema

Pseudo Algorithm: Summarize the results in a Theorema algorithm with oracle, where the oracle is given by lemmas which can be proved in Theorema.

Presentation at ICE 2012 (Initiative for Computational Economics, http://ice.uchicago.edu/) ↞ look into other areas.

We organized a symposium at this year’s AISB convention on Do-Form: Enabling Domain Experts to use Formalised Reasoning http://cs.bham.ac.uk/research/projects/formare/events/aisb2013 [LRK13]
Value at Risk

[Picture from http://www.flickr.com/photos/cau_napoli/4554437754/]
Value at Risk

[The following slides are adapted and abbreviated from the talk of and discussions with Neels Vosloo from the Bank of England 4/5 April 2013 at the Do-Form symposium [LRK13].]

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Unilever plc 17 Sep 2013</td>
<td>2,536.00p</td>
</tr>
<tr>
<td>3 month libor 17 Sep 2013</td>
<td>0.51%</td>
</tr>
<tr>
<td>contract size</td>
<td>1000</td>
</tr>
<tr>
<td>sell/buy</td>
<td>sell</td>
</tr>
<tr>
<td>strike price</td>
<td>3,200.00p</td>
</tr>
<tr>
<td>maturity</td>
<td>9 Oct 2013</td>
</tr>
</tbody>
</table>

The VaR model of the bank computes from many of such assets an overall risk, taking into account 5,000 to 10,000 different risk factors, based on statistical models.

FSA’s task (team of 8): test the VaR models of 18 banks.
Value at Risk – Regulator’s Rules

From the UK financial regulation authority’s handbook [Pra]:

relevant to . . . prudent . . . must be sound, implemented with integrity . . . unless the assumption of zero correlation between these categories is empirically justified . . . sufficient number of risk factors . . . must show a good track record . . . captures the variations of volatility of rates . . . must incorporate risk factors corresponding to the individual foreign currencies . . . take account of market characteristics . . . adequately ensured . . . [must] explain the historical price variation . . . be robust to an adverse environment . . . must conservatively assess the risk arising . . . under realistic market scenarios . . . must be appropriately conservative and may only be used where available data are insufficient or is not reflective of the true volatility . . . a firm must avail itself of these advances . . . [must have] capital resources adequate to cover that risk . . . soundness standards comparable to . . . adjusted, where appropriate, to . . .
Possible Task for Computer Science

Challenge

Identify patterns that are consistent with certain deficiencies in models or their implementation

Question: Can we help to provide good tests? (e.g., [Rei87]?)
Matching Problems

[Picture from http://commons.wikimedia.org/wiki/File:Gerrit_van_Honthorst_-_De_koppelaarster.jpg]
Matching Problems

[The following slides are adapted and abbreviated from Utku Ünver’s talks on 3 April 2013 at the Do-Form symposium [LRK13].]

Examples:

- **House allocation problem** (agents to houses)
- House market problem (match pairs (agent, house) to each other)
- **match living kidney donor-receiver pairs**
- Students and schools
- Marriage matching
House allocation problem

\((A, H, >)\) with \(A = \{a_1, \ldots, a_n\}\) agents, \(H = \{h_1, \ldots, h_n\}\) houses, and preferences \(>_a\)

\[\mu : A \rightarrow H\] is a solution

1. \(\mu\) is **Pareto efficient** if there is no better \(\nu\) so that \(\nu_a(a) \geq \mu_a(a)\) for all \(a\) and \(\nu_a(a) > \mu_a(a)\) for some \(a\).

2. Mechanism is **incentive compatible** if it is best for each agent to tell the truth.

3. A mechanism is **non-bossy** if an agent can influence the allocation of houses for other agents only by getting a different house.

4. A mechanism is **neutral** if it is invariant under permutations.

1 and 2 may be incompatible.
Randomized serial dictatorship (RSD) means that one agent can choose their preferred choice, then a second theirs and so.
A characterization of serial dictatorship

**Randomized serial dictatorship (RSD)** means that one agent can choose their preferred choice, then a second theirs and so. “RSD” is incentive compatible and Pareto efficient.
A characterization of serial dictatorship

**Randomized serial dictatorship** (RSD) means that one agent can choose their preferred choice, then a second theirs and so. “RSD” is incentive compatible and Pareto efficient.

**Theorem (Svensson 1998)**

A mechanism is incentive compatible, non-bossy, and neutral iff it is serial dictatorship.

**Possible contribution:** Find counter-examples of other characterizations.
Randomized serial dictatorship (RSD) means that one agent can choose their preferred choice, then a second theirs and so. “RSD” is incentive compatible and Pareto efficient.

**Theorem (Svensson 1998)**

A mechanism is incentive compatible, non-bossy, and neutral iff it is serial dictatorship.

**Possible contribution:** Find counter-examples of other characterizations.

Question: Can other characterizations be found?
Live organ donation

Donor 1 → Patient 1
 compatibility

Donor 2 → Patient 2
 compatibility

Algorithm of Roth, Sönmez, Ünver from 2005 for 2 way exchange.

Question: Can the correctness proof be made formal?
Auctions
Importance of Auctions

Auctions are a mechanism to distribute resources.

Applications: eBay items, mobile spectrum, internet domains, possibly regulating High-Frequency Trading [Peter Cramton], …

Importance: $268.5 billion in 2008 in the US

Given: a set of individual bids for a good (not necessarily the same as the value an individual ascribes to the good!)

Goals:
- give the good to the bidder who values it most (“efficiency”)
- determine prices
- maximize revenue

Auctions are designed and some properties are proved.
Types of Auctions

- Single good vs. combinatorial
- Open-outcry: ascending vs. descending
- Sealed-bid: first-price, second-price
- Static vs. dynamic (single vs. multiple rounds)
- Private value vs. common value
- Variants and combinations
  (e.g. ascending auction, converting to a sealed-bid auction when the number of remaining bidders equals the number of items)

Overview: [Ber+04, lecture 16 “Auctions and Bidding”]
Formal Properties of Auctions

- soundly specified?
- efficient?
- successful bidding strategies (e.g. Vickrey’s theorem)
- revenue equivalence of two auctions

Some of these have been proved on paper.
Vickrey’s Theorem

Second-price auction: a 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. The auction is efficient.*

- earliest result in modern auction theory
- simple environment in which to gain intuitions
Weakly Dominant Strategy

A definition

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

- provide auction designers with a toolbox of basic formalisations, ...
- ... on top of which they can formalise and verify their own auction designs
- building guided by canonical textbooks [Mas04; CSS06]

Homepage  http://www.cs.bham.ac.uk/research/projects/formare/code/auction-theory/
Source  https://github.com/formare/auctions
Systems Used to Prove Vickrey’s Theorem

- **Isabelle/HOL (CL with Makarius Wenzel):** higher-order logic (typed), interactive theorem proving environment, document-oriented IDE
- **Theorema 2.0 (Wolfgang Windsteiger):** FOL + set theory, textbook-style documents (Mathematica notebooks), proof management GUI
- **Mizar (Marco Caminati):** FOL + set theory, text editor, proof checker
- **Hets/CASL/TPTP (CL with Till Mossakowski):** sorted FOL, text editor, proof management GUI, front-end to local or remote automated provers

Manfred Kerber, Christoph Lange, Colin Rowat
Mechanised Reasoning in Economics
Informatik 2013 Koblenz – 17 September 2013
### Comparison result [Lan+13]

<table>
<thead>
<tr>
<th>System/Language</th>
<th>Proof speed</th>
<th>Textbook closeness</th>
<th>Top-down proofs</th>
<th>Library</th>
<th>Output</th>
<th>Community</th>
<th>Documentation</th>
<th>de Bruijn factor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Isabelle/HOL</td>
<td>++&lt;sup&gt;b&lt;/sup&gt;</td>
<td>++</td>
<td>+</td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>++</td>
</tr>
<tr>
<td>Theorema</td>
<td>?</td>
<td>n/a&lt;sup&gt;c&lt;/sup&gt;</td>
<td>++</td>
<td>+</td>
<td>—</td>
<td>++</td>
<td>n/a</td>
<td>—</td>
</tr>
<tr>
<td>Mizar</td>
<td>++</td>
<td>++</td>
<td>—</td>
<td>++</td>
<td>++</td>
<td>+</td>
<td>n/a</td>
<td>++</td>
</tr>
<tr>
<td>CASL/TPTP</td>
<td>○&lt;sup&gt;d&lt;/sup&gt;</td>
<td>−</td>
<td>+</td>
<td>++</td>
<td>+</td>
<td>−</td>
<td>○</td>
<td>+</td>
</tr>
</tbody>
</table>

<sup>a</sup> PI/TI = proof/term input; LC/LS = library coverage/search; PO = proof output; CE = counterexamples (incl. consistency checks); WF = well-formedness check.

<sup>b</sup> scores from very bad (—) to very good (++).

<sup>c</sup> fully GUI-based.

<sup>d</sup> automated provers.

Result specific to auctions? –

No, but the application orientation prioritised “soft” criteria!
Generating Verified Software: General Approach [Cam+13]

1. Definitions
   - formal specification (written by Isabelle user, needs review by auction designer)

2. Theorems
   - state soundness and other properties of

3. Proof
   - (4. checked by Isabelle)
   - known to implement (by proof and by trusting code generator)

5. code generation (Isabelle)

Code
   (executable Scala)
Combinatorial Auctions [CSS06]

\[ \sum_{m+n} b_m(x^*) \]

\[ \sum_{m+n} b(m, g^{-1}(\{m\})) \]

\[ (Y, f) \]
Combinatorial Vickrey Auction [AM06; Cam+13]

Bids are submitted on any subset of the set of available goods \( X \).

Winning allocation:

\[
X^* \in \arg \max_{X_1,\ldots,X_N} \sum_{n=1}^{N} b_n (X_n) \quad \text{s.t.} \quad \bigcup_{n=1}^{N} X_n \subseteq X_0 \quad \text{and} \quad n \neq n' \iff X_n \cap X_{n'} = \emptyset
\]

Prices:

\[
p_n \equiv \alpha_n - \sum_{m \neq n} b_m (X_m^*)
\]

where

\[
\alpha_n \equiv \max_{X_m} \left\{ \sum_{m \neq n} b_m (X_m) \left| \bigcup_{m \neq n} X_m \subseteq X_0 \right. \right\}
\]

Bidder \( n \) pays the maximum sum of bids if the auction had been run without \( n \) \((= \alpha_n)\), minus the winning bids on the items \( n \) did not get.
Generating Verified Software: Combinatorial Vickrey Auction [Cam+13]

paper-like formalisation

\[ \mathcal{X}^* \in \arg\max \sum \ldots \]

\{ R \subseteq P(\mathbb{N}) \times \mathbb{N} | \exists P \in \text{parts}(G). \text{Dom}(R) \subseteq P \land \ldots \} \]

\{ P | \bigcup P = A \land \forall x \in P. \ldots \} \]

winning determination

\[ \text{alloc } G \ N = \text{concat } [ \text{[ } R \ . \ R \leftarrow \text{inj_fun } P \ (\text{list } N) \text{] } \ . \ P \leftarrow \text{parts } (\text{list } G) \text{]} \]

allocations

\[ \text{parts } (x \ # \ xs) = \bigcup \text{inject } x \ ' (\text{parts } xs) \]

set partitions

\[ \text{argmax } (x \ # \ xs) f = \begin{cases} \text{if } f \ x > f (\text{hd } (\text{argmax } xs f)) \text{ then } \ldots \end{cases} \]

executable formalisation

\[ \text{depends on} \]

\[ \text{human formalisation} \]

\[ \text{paper source (auction designer)} \]

\[ \text{code generation} \]

\[ \text{verified code (auction software)} \]
The following links take to the Auction Theory Toolbox homepage and source repository and were valid at the time of writing.

- **Vickrey’s theorem [Lan+13]**
  - **Theorema**: look at formalisation in Mathematica (show input syntax), show proof management dialog. There is more ([Win12]), but Theorema can not yet prove Vickrey’s theorem.
  - **Mizar**: look at formalisation (using Emacs Mizar mode), check a proof (C\(\rightarrow\) c)
  - **Hets/CASL**: look at formalisation (using Emacs CASL mode), show theory graph (C\(\rightarrow\) c C\(\rightarrow\) c), prove `SecondPriceAuction#only_max_bidder_wins` (right-click on `SecondPriceAuction`), explain “Prove” dialog, show proof output
Demo Script II

- **Isabelle**: look at *Vickrey.vickreyA*, Ctrl+click on some dependencies, show some aspects of the proof (*automated steps vs. simple rules vs. Sledgehammer*).

- **Combinatorial Vickrey auction (Isabelle) [Cam+13]**:
  - **Isabelle**:
    - demonstrate try and Nitpick (*screenshot*)
    - compare `CombinatorialAuction.possible_allocations_rel` and `possible_allocations_comp`, show “injective functions” equivalence proof
    - explain example auction in `CombinatorialAuctionTest`, evaluate some expressions: `winning_allocations`, `payments`
  - **Scala**:
    - show Scala code generated from `CombinatorialVickreyAuctionCode`, compare `possible_allocations_alg Scala<->Isabelle` (strip explicit types)
Demo Script III

- show how to invoke this code
  (CombinatorialVickreyAuctionHardCoded)
- demo of how to run (ask volunteers to bid on some items, enter bids into CATS-like CAB input file, run CombinatorialVickreyAuctionCAB)
Nitpick example

```
subsection {* Left-totality *}

notepad
begin
  fix admissible::"'a ⇒ bool"
  fix A::"('a × 'b) set"
  have "(∀ x . admissible x → (∃ y . (x, y) ∈ A)) → Domain A ⊆ { x . admissible x }" try
end
```

Trying "solve_direct", "quickcheck", "try0", "sledgehammer", and "nitpick"...

Nitpick found a counterexample for card 'a = 1 and card 'b = 1:

Free variables:

```plaintext
  A = {(a₁, b₁)}
  admissible = (λx. ⊥)(a₁ := False)
```
Soundness

We proved soundness – modulo . . .

- Definitions and theorems’ statements – require human inspection
- Prover errors – very low probability
- Code generator errors – low probability
- Programming language errors – very low probability
- Execution environment (virtual machine, compiler, hardware) – very low probability
- User interface
- Cheating auctioneer
Some lessons learned
Lessons for Formalisers

- **Representation is non-trivial**, since . . .
  - it is partly not easy to understand the theorems,
  - it is partly **easy to make mistakes**.
- Find mistakes by use and proof.
- Notice **hidden assumptions**
- Often proofs that look **simple, are still non-trivial** for theorem provers.
- First **rationalize proofs** (e.g., we got Vickrey’s theorem down to 4 cases rather than 9 from a straightforward translation of the paper source).
- **HOL vs FOL, automated vs interactive ATPs differences** are not that relevant after all (but the complexity of the argument).
- **Formalising a textbook source** is hard enough already; writing a formalisation that yields executable code is harder.
Problems with Encoding

- On paper, a lot of contextual information is implicit: e.g. types and scopes of variables.
  
  **Auctions:** \( N = \{1, \ldots, n\} \) is a set of participants, often indexed by \( i \) \( \Rightarrow n \in \mathbb{N}, N_i \in \mathbb{N} \).

- Unfamiliarity with the writing style, e.g. function types:
  
  **CASL (algebraic spec.):** \( f: A \times B \rightarrow C \)
  
  **Isabelle (func. prog., currying):** \( f :: A \Rightarrow B \Rightarrow C \)

- Irrelevant Information: Not all information given on paper is relevant for getting a formal proof of one specific problem done.

- Difficult information: “Let it be common knowledge that each \( v_i \) [valuation] is an independent realizations of a random variable \( \tilde{v} \), whose distribution is described by density function \( f \). Then a strategy for bidder \( i \) is a mapping \( g_i \) such that \( b_i = g_i(v_i, f) \geq 0 \), where \( b_i \) is known as \( i \)'s bid.”
Problems with Encoding (Cont’d)

Misleading illustrations:

- Geanakoplos’ 1st proof of Arrow: “by changing his [preference order] at some profile [set including his preference and the other voters’ preference] he can move [alternative] b from the very bottom of the social ranking to the very top” – but there are no state transitions, just two different preferences profiles!
- Computers hate creative (ab)use of notation and sloppy typing e.g. the loser of an auction was originally characterized as

\[ x_j = p_j = 0 \quad (x_j \in \{0, 1\} \text{ but } p_j \in \mathbb{R}) \]

- Excessive usage of sets (supported by Isabelle, but not always efficiently)
- Sums involving zeros “An auction is efficient if it maximizes \( \sum_{i \in N} v_i \cdot x_i \)” – but for a single good, \( x_i \in \{0, 1\} \) (like an indicator function), i.e. “efficiency states \( v_i \neq \bar{v} \Rightarrow x_i = 0 \)."
Lessons for Tool Developers

- provide “one-click” DWIM interfaces
e.g. Isabelle’s `try` frontend command to automated provers
  and counter-example finders

- make reusable **theorems easy to find** in the library

- provide comprehensible **error messages**
  (Sometimes that’s actually easy for developers! – Example from Hets)

- give better guidance through **documentation**
e.g. Isabelle with a lot of **historic documentation**; some
  functionality only documented in the “outdated” manuals

- care for your **community**
  (e.g. Isabelle with an **active mailing list and on StackOverflow**)
Problems not Addressed

- Multiple items per good (can be emulated, inefficiently)
- Real-valued divisions of goods (require LP-based algorithm)
- Valuation of goods: not always known in practice
  ⇒ model as probability distribution instead
- Relevant issue beyond auction theory: How to maximise participation?

Generally: idealised assumptions
Future Work

- GUI (desktop or web form) for Scala program
  (convince target audience that we are ready for business)
- Breaking ties among equally valued allocations in a fair way
  (or with a well-defined bias, e.g. preferring newcomers)
  Bidders should know in advance how ties are broken
- Dynamic auctions (i.e. multiple rounds)
Summary

- Computer Science in general, Theorem Proving in special can support economists. Our method allows for generating verified implementations of mission-critical auction software.
- We have to work towards adjusting our methods to economics problems. Lots of foundations still need to be provided in a “toolbox”.
- Specialist knowledge is required, the systems in its current form are still difficult to use by non-experts. Parallel “paper-style” and “algorithmic” formalisation needed as a bridge to auction designers.


References VIII
