Journal of Mathematical Economics 66 (2016) 26–39

Contents lists available at ScienceDirect

Journal of Mathematical Economics

journal homepage: www.elsevier.com/locate/jmateco

An introduction to mechanized reasoning☆

Manfred Kerber[a], Christoph Lange[b], Colin Rowat[c,*]

[a] School of Computer Science, University of Birmingham, UK

[b] Fraunhofer IAIS and University of Bonn, Germany

[c] Department of Economics, University of Birmingham, Edgbaston B15 2TT, UK

ARTICLE INFO

Article history:
Received 12 January 2015
Received in revised form
22 June 2016
Accepted 23 June 2016
Available online 18 July 2016

Keywords:
Mechanized reasoning
Formal methods
Social choice theory
Auction theory

ABSTRACT

Mechanized reasoning uses computers to verify proofs and to help discover new theorems. Computer scientists have applied mechanized reasoning to economic problems but – to date – this work has not yet been properly presented in economics journals. We introduce mechanized reasoning to economists in three ways. First, we introduce mechanized reasoning in general, describing both the techniques and their successful applications. Second, we explain how mechanized reasoning has been applied to economic problems, concentrating on the two domains that have attracted the most attention: social choice theory and auction theory. Finally, we present a detailed example of mechanized reasoning in practice by means of a proof of Vickrey's familiar theorem on second-price auctions.

1. Introduction

Mechanized reasoners automate logical operations, extending the scope of mechanical support for human reasoning beyond numerical computations (such as those carried out by a calculator) and symbolic calculations (such as those carried out by a computer algebra system). Such reasoners may be used to formulate new conjectures, check existing proofs, formally encode knowledge, or even prove new results. The idea of mechanizing reasoning dates back at least to Leibniz (1686), who envisaged a machine which could compute the validity of arguments and the truth of mathematical statements. The development of formal logic from 1850 to 1930, the advent of the computer, and the inception of *artificial intelligence* (AI) as a research field at the Dartmouth Workshop in 1956 all paved the way for the first mechanized reasoners in the 1950s and 1960s.[1]

Since then, mechanized reasoning has been both less and more successful than anticipated. In pure maths, mechanized reasoning has helped prove only a few high-profile theorems. Perhaps surprisingly – although consistent with the greater success of applied AI over 'pure' AI – mechanized reasoning and formal methods[2] have enjoyed greater success in industrial applications, as applied to both hardware and software design. In the past

☆ We are grateful to Makarius Wenzel for help refining our code, to Marco Caminati for research assistance, to Peter Cramton, Paul Klemperer, Peter Postl, Indra Ray, Rajiv Sarin, Arunava Sen and Ron Smith for comments, and to the EPSRC for funding (EP/J007498/1). Rowat thanks Birkbeck for its hospitality. The presentation of the formal proof of Vickrey's theorem is based on Kerber et al. (2014). Finally, we are grateful to two anonymous referees and the co-editor for working with us to improve this paper.

* Corresponding author.
E-mail address: c.rowat@bham.ac.uk (C. Rowat).

[1] Perhaps unsurprisingly, Gardner was ahead of his time in mechanized reasoning as well: four years before his regular columns with *Scientific American* began, his first article for them included a template allowing readers to make their own mechanized reasoners—out of paper.

[2] The term *formal methods* is used here to denote approaches to establishing the correctness of mathematical statements to a precision that they can be meticulously checked by a computer. Rather than being seen as distinct from other mathematical methods, researchers in the area see them as the next step in mathematics' march towards greater precision and rigour (Wiedijk, 2008). Consider: "A Mathematical proof is rigorous when it is (or could be) written out in the first-order predicate language $L(\in)$ as a sequence of inferences from the axioms ZFC" (MacLane, 1986). The advantages of taking this next step with computers include: a computer system is never tired or intimidated by authority, it does not make hidden assumptions, and can easily be rerun. A pioneer of mechanized reasoning – who saw himself building on Bourbaki's formalism – referred to computers as "slaves which are such persistent plodders" (Wang, 1960).

http://dx.doi.org/10.1016/j.jmateco.2016.06.005

decade or so, computer scientists have also begun to apply formal methods to economics.

A central inspiration for this recent work are Geanakoplos' three brief proofs of Arrow's impossibility theorem (Geanakoplos, 2005).[3] Initially, Nipkow (2009), Wiedijk (2007), and Wiedijk (2009) used theorem provers to encode and verify two of Geanakoplos' proofs. A subsequent generation of work, drawing on the inductive proof of Arrow's theorem in Suzumura (2000), used formal methods to discover new theorems. Tang and Lin (2009) introduced a hybrid technique, using computational exhaustion to show that Arrow holds on a small base case of two agents and three alternatives, and then manual induction to extend that to the full theorem. By inspecting the results of the computational step, they were able to discover a new theorem subsuming Arrow's. Tang and Lin (2011a) used this approach – exhaustively generating and evaluating base cases, and then using a manual induction proof to generalize the results – to establish uniqueness conditions for pure strategy Nash equilibrium payoffs in two player static games; they published manual proofs of two of the most significant theorems discovered this way in Tang and Lin (2011b). Geist and Endriss (2011) used the approach to generate 84 impossibility theorems in the 'ranking sets of objects' problem (Barberà et al., 2004).

To date, the economics literature remains almost untouched by research applying mechanized reasoning to economic problems.[4] The one exception that we are aware of is Tang and Lin (2011b), whose two theorems were discovered computationally, but proved manually.[5] As it is our view that these tools will become increasingly capable, this paper aims to introduce economists to mechanized reasoning.[6] It does so by means of three analytical lenses, each with narrower scope but greater magnification than its predecessor.

First, Section 2 presents an overview of mechanized reasoning in general. We do so by setting out a classificatory scheme, with the caveat that it should not be seen as implying a partition on the field: interesting research will straddle boundaries, perhaps even forcing them to be redefined.[7]

Second, Section 3 surveys the emerging literature applying mechanized reasoning to economics. We structure this survey primarily according to the problem domain within economics, referring only secondarily to our classificatory scheme. We do this to focus on the economic insights – primarily within social choice and auction theory – made possible by these techniques, rather than on the techniques *per se*.

Finally, to make this introduction more concrete, Section 4 provides an example of what mechanized reasoning looks like in practice, presenting a blueprint of a mechanized proof of Vickrey's theorem on second-price auctions. We present such an established theorem to focus attention on its implementation.

Section 5 concludes, and suggests some possible next steps for mechanized reasoning in economics.

2. Mechanized reasoning

Our overview of mechanized reasoning distinguishes between deductive and inductive systems. While the distinction has been recognized at least since Aristotle, deductive reasoning – which allows reliable inference of unknown facts from established facts – has been in the focus of the mechanized reasoning community. Inductive reasoning also generalizes from individual cases, but does not restrict itself to reliable inferences; the cost of this additional freedom is that its conjectures must then be tested.

2.1. Deductive reasoning

Historically, deductive reasoning systems were among the first AI systems, dating back to the 1950s. While the origins of deductive reasoning date to at least Aristotle, modern advances in this area built on the work of logicians in the second half of the 19th century and the start of the 20th (e.g. Whitehead and Russell, 1910). At the Dartmouth Workshop in 1956, Newell and Simon introduced the Logic Theorist, an automated reasoner which re-proved 38 of the 52 theorems in chapter of Whitehead and Russell's *Principia Mathematica* (Whitehead and Russell, 1910).[8]

Abstractly, a deductive reasoner implements a *logic* – which is comprised of a *syntax* defining well-formed formulae and a *semantics* assigning meaning to formulae – and a *calculus* for deriving formulae (called theorems) from formulae (called premises or axioms). Historically, subfields of mechanized reasoning have been defined by choice of logic, calculus and problem domain. This section provides a classificatory scheme based, first, on the choice of calculus. Following the choice of calculus, a logic is chosen to balance expressiveness and tractability. Finally, the problem domain itself will dictate some of the specialized features of a mechanized reasoner.

When a mechanized reasoner applies the calculus' permissible operations to the axioms to obtain new, syntactically-correct formulae it does not make use of the semantics: the semantics, or ascribed meanings, yield models that may assist human intuition, but which are not necessary to the formal process of reasoning itself.[9] Crucially, mechanized reasoning involves manipulating symbols.[10]

Thus, mechanized deductive reasoning since the Logic Theorist has seen reasoning as a search task for a syntactically well-defined goal.[11] Further, as the spaces through which search occurred was potentially large, successful reasoning would use *heuristics* to avoid unprofitable sequences of operations. From this point of view,

[3] All three use Barberà's replacement of Arrow's *decisive voter* with a *pivotal voter* (Barberà, 1980). Barberà (1983) also used this approach to find a direct proof of the Gibbard–Sattherthwaite theorem.

[4] A recent symposium on economics and computer science, involving central figures at the interface between the disciplines, made no mention of mechanized reasoning (q.v. Blume et al., 2015).

[5] The process by which the theorems were discovered is described in Tang and Lin (2011a,b) itself is all but silent on its mechanized origins.

[6] For more general introductions, see Wiedijk (2008) and Avigad and Harrison (2014). Harrison (2007) introduces mechanized reasoning alongside computer algebra, presenting something of a unified view.

[7] For example, we shall see that mechanized theorem discovery is usually associated with inductive reasoning. However – in economic examples – the most fruitful examples of theorem discovery (Tang and Lin, 2009, 2011a,b; Geist and Endriss, 2011) have combined very simple deductive reasoning systems with human intelligence.

[8] According to McCorduck (2004), Russell himself "responded with delight" when shown the Logic Theorist's proof of the isosceles triangle theorem, whose proof was more elegant than their manual one.

[9] Beginning with Euclid's efforts to axiomatize geometry, logicians have produced syntactical descriptions that make semantic references obsolete: Hilbert allegedly said that we would still have an axiomatization of geometry if we replaced the words 'point', 'line', and 'plane' by 'beer mug', 'bench', and 'table' (Hoffmann, 2013, p. 6).

[10] That this was an insight at one point may be inferred from Turing's famous explanation that, "computing is normally done by writing certain symbols on paper" (Turing, 1936).

[11] As noted by Harrison (2007), specialist provers have also been developed for particular problems for which more structured approaches than general search are appropriate.

mechanized reasoning operates as chess computers do.[12] For a chess computer, the premises' intended semantic interpretations are the board, its pieces and their positions; the calculus specified permissible moves. A chess computer could then test manually discovered solutions to chess puzzles by verifying that each move satisfies the requirements of its calculus, with the final operation yielding the goal-formula. More ambitiously, and interestingly, chess programs discover solutions (e.g. sequences of winning moves) by searching through permissible operations, with the benefit of heuristics (e.g. regarding relative values of pieces).

A set of premises and a formula may be related in two different ways. First, the *semantic consequence relation* describes situations in which the formula *follows from* the premises: if the symbols in the premises are interpreted in such a way that the formulae in the premises are all true, then the formula is also true when the symbols in it are interpreted in the same way. Second, the *syntactic derivability relation* describes situations in which the formula *can be derived from* the premises: it is possible to generate the formula from the premises by applying a fixed set of so-called calculus rules. (An example of such a rule is *modus ponens*: From A and $A \rightarrow B$ it is possible to derive B, where A and B may match any formal expression). A proof that applies such rules, without any appeals to intuition or to the reader filling in steps on her own, is called a *formal proof* of the formula using the premises.

A calculus is called *sound* if only formulae can be derived from the premises that actually follow from them. Deductive reasoning is sound; inductive reasoning, considered below, is not.

A calculus is *complete* if it allows derivation of any formula that follows from the set of premises. A calculus is *decidable* if, for any set of premises and any formula, there is a procedure that either derives the formula from the premises or proves that no such derivation exists; a calculus is *semi-decidable* if a procedure exists that derives the formula from premises, whenever the formula follows from them (but may not terminate if it does not).

Decidability typically depends on the expressiveness of the logic used: more expressive logics model a richer set of concepts, but are generally harder to manipulate. While ambitious exercises in mechanized reasoning often begin by specifying a suitably tailored logic,[13] we largely restrict our attention to some of the best known *classical logics*.[14]

Propositional (Boolean) logic: *Propositional* or *Boolean logic*, the simplest classical logic, only uses propositional variables – which are either true or false – and *connectives* such as $\wedge$ (and), $\vee$ (or), $\neg$ (not), and $\rightarrow$ (implies). An example of a propositional formula is

$$first\_bidder\_bids\_highest \wedge second\_bidder\_bids\_lowest.$$

Propositional logic can only make concrete, finite statements, but has a sound, complete and decidable calculus.

An advantage of this decidability is that it may allow *push-button* technology, which does not require specialist knowledge in order to use. Once a problem is adequately represented a corresponding system solves the problem fully automatically.

First-order logic: *First-order logic* (FOL) is more expressive. First, it can speak about objects (e.g. "bidder b_1") and their properties (e.g. "bidder b_1 wins auction", $bidder(b_1) \wedge wins\,(b_1)$). Second, $\exists$ and $\forall$ allow quantification over objects. For example, "every losing bidder pays nothing" may be expressed as

$$\forall i\,.\,bidder(i) \rightarrow (\neg wins\,(i) \rightarrow pay\,(i) = 0). \tag{1}$$

Expressions like *wins* are called *predicates*, Boolean functions which – when applied to their arguments – evaluate to either true or false. Gödel's completeness theorem proves that FOL has a sound and complete calculus, but FOL has only semi-decidable calculi. Furthermore, FOL is not expressive enough to express the finitude or (per negation) infinity of the non-empty sets of objects.[15]

Many-sorted FOL uses *sorts* to extend first-order logic, not to add to its expressiveness, but to allow more concise representations, and – therefore – more efficient proving. Sorts restrict the instantiation of variables to expressions of a certain sort. For instance, sorts allow us to specify that variable i is a bidder, and variable x a good. Formula (1) is then more precisely stated as:

$$\forall i_{bidder}\,.\,\neg wins\,(i) \rightarrow pay\,(i) = 0. \tag{2}$$

i (with the sort *bidder* mentioned only at the first occurrence) can be instantiated now by terms of sort bidder, but not by those of sort good, thus reducing the search space for a proof. Sorted formulae can be translated to unsorted formulae by converting the sorts to unary predicates (which take a single argument).

Higher-order logic: *Higher-order logic* (HOL) enriches the expressiveness of FOL by extending quantification to predicates and functions. It also allows predicates and functions to take certain[16] other predicates and functions as arguments. For example, bids, b, are both a function from bidders to prices and an argument (along with N, v and A) in the predicate

$$equilibrium\_weakly\_dominant\_strategy\ N\ v\ b\ A.$$

Against this, HOL's calculi are not decidable, and are – by Gödel's incompleteness theorem – incomplete.

Two common ways in which the classical logics (in particular, FOL) are augmented are, first, by the addition of set theoretical axioms and, second, by the addition of modal operators. The first allows the approximation of higher order logic while maintaining advantages of first order logic; the second allows logic to be applied to modalities, such as knowledge, belief, or time.

Set theoretical axioms allow the definition of new symbols and operations on both predicates (e.g. $\in$ and $\subseteq$) and functions (e.g. $\cup$, $\cap$ and $\emptyset$).[17] They also allow the specification of properties of sets (e.g. $a \notin X$). Adding set theoretical axioms to FOL allows it to weakly simulate HOL: functions can be expressed as relations over $X \times X$ that are left-total and right-unique; predicates are expressed as sets. While HOL is still more expressive than FOL augmented by set theory (e.g., FOL cannot express inductive arguments), HOL's incompleteness means that there are true statements that can be expressed in HOL but which may not have finite proofs. As FOL augmented by set theory uses FOL, it remains complete by using FOL's complete calculus.

Modal operators – such as 'next' and 'until' – allow the consideration of *modes* (or *states* in economic parlance). *Linear temporal logic* (LTL) is a popular simple modal logic, modelling states in a linear fashion, thus excluding the consideration of multiple possible future states. Kamp's theorem established the equivalence of LTL with a first-order logic. Another first-order approach to modelling states is the *situation calculus* (McCarthy and Hayes, 1969),

[12] Indeed, Newell's collaboration with Simon began after the latter became aware of the former's work on a chess machine.

[13] See, for example, the *judgement aggregation logic* (JAL) of Ågotnes et al. (2011).

[14] The 17 volumes in the second edition of Gabbay and Guenthner (2001/2014) make clear that the classical logics are a small subset of all logics.

[15] Thus, FOL could not express that only finitely many bidders participate in an auction.

[16] Unrestricted formula building leads to antinomies as discovered by Russell. The introduction of types imposes a hierarchy on logical objects, including predicates. This disables circular constructs such as $X(Y) := \neg Y(Y)$, which – when Y is instantiated with X – produces the set of all X for which $X \notin X$, Russell's famous antinomy.

[17] Constants such as $\emptyset$ are considered as a special case of functions, nullary functions–functions that do not take any argument.

Table 1
Mechanized reasoning using deductive logics.

	Decidable	Undecidable
Logic	SAT, CSP; description logic	ITP, ATP
Computer system	Model checking	Program verification

which allows expression of states and the temporal development of systems in first-order logic by representing the state as an extra argument of the formulae (e.g., that agent i has £10 in state s_0 can be expressed as $has(i, 10, s_0)$). By referring to the state absolutely, rather than in relation to other states, the problem can be expressed in standard FOL without recourse to specialized modal relations.

Our final level of distinction is the domain of the problem; this level will allow us to present concrete examples of the preceding. Table 1 depicts these dimensions within deductive reasoning systems.

Decidable logic: In Table 1, the *decidable logic* cell refers to decidable calculi as applied to logical problems.

Boolean satisfiability problems (SAT) are among the simplest canonical problems in propositional logic. They specify a (finite) set of statements about a (finite) set of propositional variables, and ask whether there exists an assignment of values (i.e. true and false) to each of those variables that simultaneously satisfies all of the statements.

In SAT problems, clauses of Boolean variables are typically expressed in *conjunctive normal form*, conjunctions ($\wedge$) of disjunctions ($\vee$) such as

$$(\neg p \vee q) \wedge (p \vee \neg q)\,; \tag{3}$$

where p and q are Boolean variables, evaluating either to true or false.[18] Revisiting the example that in auctions the non-winning player pays nothing, Eq. (1) can be translated for a finite number of bidders (here, three) to a propositional logic formula,

$$(wins1 \vee \neg pays1) \wedge (wins2 \vee \neg pays2) \wedge (wins3 \vee \neg pays3); \tag{4}$$

stating for each of the three players separately that they win or pay nothing.

Any formula in propositional logic can be expressed in this form, as can any formula in first-order logic when the domain is restricted to a concrete finite domain (such as three bidders in an auction). A SAT solver is used to try to assign the variables such that all of the clauses are true. For instance, assigning *wins1* and *pays1* to *true* and the other predicates to *false* shows that the single formula (4) is satisfiable.

SAT problems are $\mathcal{NP}$-hard (Karp, 1972), requiring – in the worst case – trial of every possible input. Thus, while the logic and calculi involved are simple, SAT problems may not be computable in practice except in small cases. However, techniques have been developed so that SAT solvers are able to solve typical cases very quickly. One application area of SAT solvers are model checkers, as described below.

Constraint satisfaction problems (CSP) are triples, $\langle V, D, C\rangle$, where V is a set of variables, D their domain, and C the constraint set. In CSPs, the variables may take on more values than in Boolean satisfiability's binary assignments. For example, an *hours* variable might take one of twelve values. While apparently richer, CSPs can be reduced to SATs by suitable definition of additional auxiliary variables.[19]

The third example of decidable calculi applied to logical problems that we consider are *description logics*. These are central to automated reasoning about concept hierarchies in classification (or ontological) tasks. One of their most important applications is to the *semantic web*, which allows computers to extract semantic information from web pages. As a simple example, semantically enabled web searches could recognize that $x^2 + y^2 = z^2$ and $a = \sqrt{c^2 - b^2}$ were both statements of Pythagoras' theorem.[20]

Model checking: Model checking (Clarke et al., 1986, 1994) builds finite models to describe computer hardware systems or simple software systems and then tests their properties. Typical questions include whether certain states of the system can be reached, or whether information is flowing properly through a circuit design.

Such models are typically expressed as *finite automata*. A finite automaton can model either a finite system or an infinite system if abstraction allows the infinite state space to be simplified to a finite one.[21] Then the model is systematically checked for desired properties, e.g. by using SAT solvers. Viewing digital computer chips as a set of Boolean statements allows them to be modelled as *decidable computer systems* allowing, in turn, SAT solvers to automatically verify their properties. Since the mid-1990s, Intel has used formal methods to formally prove properties like 'this chip implements the IEEE division standard' following an embarrassing and costly recall of a Pentium chip that was discovered not to properly implement IEEE floating point division (Harrison, 2006). No further such problems have been reported since then.[22]

Undecidable logic: The upper right cell in Table 1 refers to the application of undecidable calculi to logical problems. The two types of mechanized reasoning mentioned here, *interactive theorem proving* (*ITP*) and *automated theorem proving* (*ATP*) have traditionally been equated with theorem proving, but seen as distinct, with the former involving more steering from a human user than the latter. Stereotypically, an ITP system could check an existing proof, while an ATP system could suggest steps in a proof or, in some cases, a whole proof. In practice, the distinction between the two has decreased, with ITP systems implementing ATP procedures.[23]

The traditional identification of theorem proving with work in these areas owes partly to some high profile successes in pure mathematics, the focus of the most hope in mechanized reasoning's early days. The earliest major success was – as might be expected in an emerging field – not even a clear example of mechanized reasoning: in the 1970s, computers were used to carry out the exhaustive computations required to prove the four-colour map theorem (q.v. Appel and Haken, 1977; Appel et al., 1977). Here, the computers were used to perform simple (algebraic) calculations, rather than to (logically) 'reason'. More recently, mechanized proof checkers have confirmed these results formally (q.v. Gonthier, 2008).[24]

[18] The sentence given here is logically equivalent to $p \equiv q$, an equivalence exploited by Tang and Lin (2011a) in their search for uniqueness conditions in bimatrix games.

[19] See Bordeaux et al. (2006) for a comparison of SAT and constraint programming.

[20] See Lange (2013) for a more in-depth discussion of applications of semantic web technology to mathematics.

[21] For example, in proofs involving real numbers, it may suffice to reduce an infinite number of possible values – which cannot be handled by a decidable calculus – to a trinary partition defined by $>$, $<$ and $=$. See Burch et al. (1990) for an application to large, complex microprocessor circuits.

[22] With chip design becoming more and more sophisticated, the reasoning in the verification needed to become also more sophisticated. Thus, HOL theorem provers such as HOL-Light are now also used for hardware verification.

[23] Harrison (2007) noted that ITP may be preferred to ATP, as – in working more closely alongside human reasoning – it may be better at developing human understanding.

[24] Gonthier's team has now also formally checked the Feit–Thompson Odd Order Theorem (Gonthier et al., 2013).

The first major mathematical result to be established by mechanized reasoning – rather than 'mere' calculation – was Robbins' conjecture that two bases for Boolean algebras are equivalent. While appearing to be a beguilingly simple problem, it remained unresolved for 60 years, becoming a favourite of Tarski, who set it as an open problem (q.v. Henkin et al., 1971, p. 245). One of the complicating factors of the conjecture was that the only known example of a Robbins algebra was also a Boolean algebra, reducing the evidence base that mathematicians could use to form intuitions about the problem. Nonetheless, in the late 1990s, McCune (1997) was able to pose the problem in a way that allowed EQP, an automated theorem prover related to his well-known Otter prover, to generate – not just check – a 17-step proof, later reduced to eight steps (McCune, 1997).[25]

Perhaps the highest profile success of mechanized reasoning in pure mathematics is the solution to Kepler's conjecture that there is no denser packing of spheres in $\mathbb{R}^3$ than the face-centred cubic. Hales' original proof was 120 pages long (excluding computer code that exceeded 500 MB), requiring a team of 12 referees five years to become "99% certain" that it was correct. Unsatisfied with this standard, Hales founded *Project Flyspeck* to establish a fully formal proof of the conjecture (Hales, 2012). In August 2014, the project was completed (Hales et al., 2015), close to Hales's original estimate of 20 person-years (Avigad and Harrison, 2014).

More mundanely, ITP has been used to translate existing human proofs into formal proofs that are sufficiently detailed that a computer can mechanically verify them: as of January 2016, 91 of the 'top 100' mathematical theorems on a list maintained by Wiedijk (2014) had been formalized.[26] While most of these are considerably less spectacular than the examples cited above – in which theorem provers have been used to help convince mathematicians as to the validity of major, new results – the gradual accretion of small proof libraries builds a foundation for applying ATPs more widely.

The distinction between high-profile, major theorems and lower-profile bodies of theory has been suggested as a reason that ATP has yet to fulfil its early hopes: Buchberger (2006) noted that human mathematicians typically do not try to prove isolated theorems but explore a whole theory, thereby building up valuable intuition which helps them in proving related theorems. Additionally, Newell (1981) stated that standard theorem proving techniques – while often highly efficient – do not make use of advanced human approaches (as described in Pólya's books) such as simplifying a problem to one they can solve; applying the simplified solution to the original problem may still be very hard, but the intuition gained by solving the simplified problem may help solve the original problem.[27]

Program verification Table 1's lower right cell corresponds to software engineering's *program verification*, reasoning about software systems. This can be highly complex in the case of complex programs. Within program verification, traditional proof approaches have sought to prove that the software correctly implements properties specified in the design brief. As such proofs are very costly, full correctness proofs that seek to verify all desired properties of the code, are done only for 'mission critical' systems (Vijay D'Silva Daniel Kroening, 2008).

Some well known examples of program verification have come from transport and finance: in code controlling automated commuter rail systems, theorems that no two trains occupy the same location at the same time have been proved; within financial transactions software, theorems that transactions do not create or destroy value, but merely transfer it, have also been proved (Woodcock et al., 2009). More recently, a compiler for the C programming language has been formally verified (Boldo et al., 2013). These techniques are becoming more mainstream: in 2013, Facebook acquired Monoidics, a start-up firm applying theorem proving to software code analysis; in 2015, another start-up, Aesthetic Integration beat 600 competitors to win first prize in UBS' Future of Finance Challenge for its ability to automatically prove failure or compliance in financial algorithms.[28]

Historically, program verification has been conducted as a *post mortem*: given existing code, program verification determines whether or not it is correct. More recently, *code extraction* techniques have been developed to generate code that provably implements the desired properties.

2.2. Inductive reasoning

As noted above, both inductive and deductive reasoning date back at least to Aristotle, but the former is not sound, while the latter has been the focus of the mechanized reasoning community. The distinction between the two – as well as the utility of each – was expressed by Pólya (1954, p. vi), who referred to deductive reasoning as *demonstrative reasoning*, and inductive reasoning as *plausible reasoning*:

> We secure our mathematical knowledge by *demonstrative reasoning*, but we support our conjectures by *plausible reasoning* . . . Demonstrative reasoning is safe, beyond controversy, and final. Plausible reasoning is hazardous, controversial, and provisional. . . .
> In strict reasoning the principal thing is to distinguish a proof from a guess, a valid demonstration from an invalid attempt. In plausible reasoning the principal thing is to distinguish a guess from a guess, a more reasonable guess form a less reasonable guess. . . . [plausible reasoning] is the kind of reasoning on which [a mathematician's] creative work will depend.

Inductive systems seek to derive general statements based on a finite number of statements (e.g. if A_1 is true, and A_2 is true, and so on up to A_N for some finite N, then A_n is true for all natural numbers n).[29] This sort of reasoning is immediately familiar to us when we reflect on how we form conjectures: we expect the sun to rise tomorrow without any understanding of astrophysics; this expectation, though, may lead to the formation of conjectures about astrophysics. However compelling the weight of evidence, inductive reasoning is not sound—as may be demonstrated by single counterexamples. In number theory, Euler's attempted generalization of Fermat's last theorem remained open for two

[25] Dahn (1998) manually reworked EQP's proof to provide a more human-readable proof.

[26] Exceptions include Fermat's last theorem.

[27] Conversely, Dick (2011) observed that the 'resolution' inference rule (Robinson, 1965), central to mechanized reasoning, "was not based on any known human practice and was in fact difficult and counterintuitive for humans to understand". Indeed, reviewing mechanized reasoning since resolution, Robinson lamented that it may have harmed mechanized reasoning by contributing to a parting of ways between human mathematicians and mechanized reasoners (Dick, 2015).

[28] Their entry formally defined a UBS 'dark pool' and a set of SEC regulations which the SEC had found the dark pool in breach of. Aesthetic Integration was able not only to verify the dark pool failure found by the SEC, but discovered that its order prioritization failed to satisfy transitivity (Ignatovich and Passmore, 2015).

[29] Inductive reasoning is distinct from mathematical induction, which involves proving A_0 and that A_{n+1} is true given A_n. Mathematical induction is a sound *deductive* method.

Table 2
Some applications of mechanized reasoning to economic problems.

	Decidable	Undecidable
Logic	Geist and Endriss (2011), Brandt and Geist (2016): SAT Tang and Lin (2009): SAT, CSP Bai et al. (2014): description logic	Nipkow (2009), Wiedijk (2007), Wiedijk (2009), Lange et al. (2013): ITP Grandi and Endriss (2012): ATP
Computer system	Xu and Cheng (2007), Arcos et al. (2005), Tadjouddine et al. (2009): model checking	Caminati et al. (2015): code extraction

centuries until a computer found a counterexample.[30] In game theory, Neumann and Morgenstern conjectured that stable sets ('solutions' in their parlance) always existed; it took almost a quarter-century for counterexamples to be found (Lucas, 1968).

Inductive reasoning may be used for *theorem discovery*, whereby regularities in observed data are used to form conjectures to test.[31]

Mechanized inductive reasoning dates back to two systems built in the 1970s and 1980s to discover new conjectures, AM (Automated Mathematician) (Lenat, 1976) and Eurisko (Lenat, 1983). These were able to detect conjectures such as the unique prime factorization theorem and Goldbach's conjecture.[32] The systems use certain measures of interestingness for concepts. For instance, concepts that are always true or always false are not interesting. However, if a concept is true for a significant proportion of examples (such as divisibility by only 1 and the number itself) then this is considered as an interesting concept ('primality' for divisibility by only 1 and the number itself).[33]

Lenat's work was continued by Colton in the HR (Hardy–Ramanujan) system (Colton et al., 1999), where more advanced measures for interestingness were developed. For instance,

> The novelty measure of a concept calculates how many times the categorisation produced by the concept has been seen. For example, square numbers categorise integers into two sets: $\{1, 4, 9, \ldots\}$ and $\{2, 3, 5, \ldots\}$. If this categorisation had been seen often, square numbers would score poorly for novelty, and vice-versa (Colton et al., 2000).

Another important advance in Colton's work is that the HR system weeds out simple conjectures, namely those that can be easily verified or falsified by automated theorem provers.[34] One of the successes of HR was that it invented the concept of 'integers with a square number of divisors' which was added to Sloane's Encyclopedia of Integer Sequences.[35]

3. Mechanized reasoning for economic problems

Over the past decade, computer scientists have become interested in economic problems—often publishing economically novel and interesting results, but almost entirely within the computer science literature. This section reviews that literature, focusing on the applications to social choice and auction theory. We structure this survey primarily according to the problem domain within economics, and only secondarily according to our classificatory scheme, in order to focus on the insights into economic problems made possible by these techniques, rather than the techniques themselves.

Table 2 places the papers reviewed in this section into our original classificatory scheme. This classification is imperfect. For example, Tang and Lin (2009) and Geist and Endriss (2011) both used propositional logic solvers (and, therefore, deductive reasoning), but used them to discover new results—which we have associated, above, with inductive reasoning. Papers like this therefore span historical distinctions.

Social choice has been mechanized reasoning's main point of contact with economics, making it a convenient lens for illustrating mechanized reasoning. Auction theory is, we feel, promising as a new point of contact between mechanized reasoning and economics, due both to the technical parallels between social choice (where mechanized reasoning has proved fruitful) and mechanism design (q.v. Reny (2001)), and to auctions' importance as allocation mechanisms.

3.1. Social choice

Geanakoplos' three brief and distinct proofs of Arrow's impossibility theorem – that, for three or more alternatives and a finite set of agents, there is no social choice rule satisfying unanimity (*UA*), independence of irrelevant alternatives (*IIA*) and non-dictatorship (*ND*) – served as the mechanized reasoning community's entrée to economic problems: social choice was novel to this community, yet used familiar structures – particularly linear orders – and the three proofs by Geanakoplos (2005) gave the mechanized reasoning community an opportunity to attempt to compare the relative difficulty of encoding those proofs for computers.

One primitive measure of the relative difficulty of formal proofs is to compare their size to that of human proofs.[36] Table 3 reports on the relative sizes of Nipkow's proofs in Isabelle – a higher-order logic theorem prover – and Wiedijk's proof[37] in Mizar – a set theoretic proof checker, which augments first-order logic by

[30] Euler's conjecture states: let n and k be integers greater than one, and let $a_1, \ldots, a_n$ and b be non-zero integers; then $\left(\sum_{i=1}^{n} a_i^k = b^k\right) \Rightarrow (n \geq k)$. The first known counterexample, found by computer, is $27^5 + 84^5 + 110^5 + 133^5 = 144^5$ (Lander and Parkin, 1966).

[31] One of the most dynamic subfields of AI currently is *machine learning*. Some definitions are agnostic as to how the machines learn – e.g. whether deductively or inductively – while, perhaps more typically, others link machine learning more closely to inductive reasoning. Some of the highest profile applications of machine learning are statistical, positing rules that fit the existing data well, rather than perfectly.

[32] The prime factorization theorem states that any positive integer has a unique decomposition as the product of primes. Goldbach's conjecture states that every even integer beyond two can be expressed as the sum of two primes.

[33] Dick's case study of the Argonne National Laboratory's AURA system noted that, while "the capacity to identify what was 'promising' or 'interesting' was precisely one of those unautomatable human abilities . . . the Argonne practitioners decided what was important on the basis of extensive experimenting with AURA".

[34] See also the introduction of Tang and Lin (2011a) for a brief review of the history of mechanized theorem discovery; a lengthier review is available in Tang (2010).

[35] https://oeis.org/.

[36] The easiest way of determining the size of a formal proof is by counting lines of source code. In Section 4 we discuss a less biased measure, the de Bruijn factor.

[37] Wiedijk justified his decision to formalize only Geanakoplos' first proof by noting that they became successively more abstract, making the first the most challenging as, generally "abstract mathematics is easier to formalize than concrete mathematics" (Wiedijk, 2009).

Table 3
Relative lengths of human and machine proofs of Arrow's theorem.

	1st proof	3rd proof
Paper (Geanakoplos, 2005)	1 page	1 page
Isabelle (Nipkow, 2009)	350 lines (6 pages)	300 lines
Mizar (Wiedijk, 2007, 2009)	1100 lines	

the axioms of Tarski–Grothendieck set theory.[38] Nipkow (2009) attributed the greater length of the Mizar proofs to Isabelle's "higher level of automation"—something to which we return in our Isabelle proof of Vickrey's theorem.

Nipkow's formalization attempts began with Geanakoplos (2001), a working paper that preceded the published version (Geanakoplos, 2005). In seeking to formalize the first proof, he discovered a statement in one of the lemmas that required a 20 line auxiliary proof to properly establish. Further, a relationship between a pivotal voter and a dictator only "hinted at" in the original text required elaboration. Nipkow did not discover any errors in this first proof. Similarly, Wiedijk (2009) reported on missing cases, but no "real errors".

As to the third proof, Nipkow found two instances of omitted material in its central lemma, preventing him from formalizing the proof. Nipkow presented these concerns to Geanakoplos by e-mail; both concerns were resolved in Geanakoplos (2005).[39]

Both Nipkow and Wiedijk's proofs were written by the authors themselves, and are therefore examples of ITP. By contrast, Grandi and Endriss (2012) sought to, first, restate Arrow's theory in FOL and, then, to automatically generate a proof for it.[40] Expressing Arrow's theory in FOL presented the challenge that quantifying over all possible linear orders of agents' preference profiles appears to be a second-order quantification as it involves quantifying over agents, alternatives, and the agents' preference profiles. Grandi and Endriss addressed this by adopting the approach taken in Tang and Lin (2009), namely to apply the situation calculus (mentioned in Section 2.1) for the representation. Thus, they could present a first-order formalization of the requisite axioms, T_{ARROW}, allowing them to restate Arrow's theorem as:

Theorem 1 (*Arrow à la Grandi and Endriss, 2012*). *T_{ARROW} has no finite models.*

A model in this sense is an instantiation (or example) of the variables used in the theory. For Arrow's theorem, the variables include N (the set of agents), A (the set of alternatives), the set of the agents' preference profiles, and the set of social welfare functions (SWFs) mapping from such profiles to a social preference. In the two-agent, three-alternative case, that T_{ARROW} "has no finite models" means that none of the 6^{36} possible SWFs satisfy the theory's axioms.[41] The theorem claims this property for any finite number of agents, and any finite number of alternatives in excess of three.

FOL's completeness allows any property of the system to be explicitly derived. However, the second problem with FOL encountered by Grandi and Endriss is that FOL is unable to express finitude, for the same reason that it cannot express induction: intuitively, HOL defines finitude by considering the complement of the infinite, which it can define by induction on the natural numbers. Thus, formulating Arrow's Theorem in FOL requires a separate formulation for each $|N|$. Similarly, proofs of Arrow's theorem in FOL may differ for each $|N|$. Thus, Grandi and Endriss' attempts to use a first-order theorem prover to automatically generate proofs of Arrow's theorem failed outside of minimal cases.[42]

Independently of Geanakoplos' proofs, Suzumura (2000) had presented an induction proof of Arrow's impossibility theorem for a base case of two agents and $|A|$ alternatives; an induction result then demonstrated its truth in general. This motivated Tang and Lin (2009) to manually derive a second induction result in the number of agents. Proving the impossibility in a two-agent, three-alternative base case, would – by their two induction lemmas – cause it to hold in general. They computationally exhausted this base case in two different ways.

First, they expressed the problem as a Boolean SAT problem. Tang and Lin then used the situation calculus, which allows many of the problem's symmetries to be efficiently dealt with by the action of swapping arguments, to reduce the number of variables needed in the base case to 35,973 in 106,354 clauses. These are too many cases to check manually. However, using the SAT solver Chaff2 they could show the inconsistency between the three basic axioms in less than a second on a desktop computer.

Second, Tang and Lin expressed the problem as a CSP, in which V, the set of variables, consists – in their base case – of 36 preference profiles; D, their domain, of six linear orderings for each profile; and C, their constraint set, of the *UN* and *IIA* axioms. As the base case implies $6^{36} \approx 10^{28}$ possible SWFs – far too many to be feasibly generated – the authors used the (first-order) logical programming language Prolog to generate all SWF satisfying the constraints of *UN* and *IIA*. Running in less than a second on a desktop computer, their Prolog code generated two SWFs, both of which were also dictatorial.

A similar approach yielded the Muller–Satterthwaite theorem, and Sen's Paretian liberal result, among others.[43]

When implementing the CSP, the authors noticed that imposing even just the *IIA* constraint reduced the set of SWFs from 6^{36} to 94. By inspecting these manually, Tang and Lin (2009) posited a new theorem that implies both Arrow's and Wilson's. Before stating it, note that a social order is *inversely dictatorial* if it ranks elements in the opposite way to at least one agent; the *Kendall tau* distance between two orderings is the number of pairs on which they disagree. Then:

Theorem 2 (*Tang and Lin, 2009*). *If a social welfare function W on (N, A) satisfies IIA, then for every subset Y of A such that $|Y| = 3$,*

1. *W_Y is dictatorial, or*
2. *W_Y is inversely dictatorial, or*
3. *The range of W_Y has at most 2 elements, whose [Kendall tau] distance is at most 1.*

As an example of an SWF accepted under condition 3 of theorem, consider the function that always prefers the first alternative to the second, always prefers the first to the third, and prefers the second to the third alternative unless both agents prefer

[38] The advantage of Tarski–Grothendieck set theory over Zermelo–Fraenkel is that the former only requires finitely many axioms to axiomatize sets.

[39] Mechanized reasoning can identify omissions by forcing close scrutiny. This, of course, is also possible without mechanical support. For example, in the matching literature, Aygün and Sönmez (2013) identified a hidden assumption in Hatfield and Milgrom (2005) – which they view as "widely considered to be one of the most important advances of the last two decades in matching theory" – without which many of their results fail to hold. The oversight arose from "an ambiguity in setting the primitives of the model". This ambiguity would likely have been detected by a mechanized reasoner as well.

[40] Grandi and Endriss (2012) is also a good guide to related work on formalizing results in social choice.

[41] There are a total of 36 preference profiles in the domain, and six orders in the range, yielding a total of $\prod_{i=1}^{36} 6$.

[42] They used Prover9, a successor to Otter, and – therefore – a close relative of the system that found the proof of Robbins' conjecture (McCune, 1997).

[43] See Geist (2010) for a more complete list.

the third to the second. This is neither dictatorial nor inversely dictatorial: the agents' preferences for the first item are ignored; there are only two elements in its range (e.g. $a \succ b \succ c$ and $a \succ c \succ b$), the distance between which is one.[44] As Tang and Lin noted, the third case of their result violates Arrow's original non-imposition axiom, which requires that the SWF be surjective, mapping to every possible value in its range.

Of the 94 SWFs satisfying *IIA*, there are 84 of the sort described above, 6 constant SWFs (one for each ordering), two dictatorial functions, and two inversely dictatorial functions.

As before, the theorem is established by exhaustive computation on the two-agent, three-alternative base case, and then extended to arbitrary finite domains by the manually-derived induction lemmas. Chatterjee and Sen (2014) observed that, as far as they were aware, this is the "only Arrow-type result in the literature that does not use an axiom other than *IIA*", an achievement that they believe "could not have been conjectured without computational aid".[45]

Social choice is replete with characterization and impossibility results. Geist and Endriss (2011) applied the Tang and Lin (2009) approach to the problem of ranking sets of objects (Kannai and Peleg, 1984), for which Barberà et al. (2004) supplied almost 50 possibly desirable axioms.[46]

Rather than deriving an induction lemma for every base case of interest, they derived a broadly applicable induction theorem based on model theory's Łoś–Tarski preservation theorem, which describes when properties (φ, below) are retained in substructures, namely essentially when the theory can be expressed using universal quantifiers in the form $\forall x \,.\, \varphi$.[47]

Furthermore, as they wished to distinguish between individual alternatives, sets of preferences, and preference orders the authors used a many-sorted FOL. Many-sorted FOL also allows relations (including set inclusion or union) to be defined on one domain that do not hold on the other.

Geist and Endriss then encoded 20 axioms drawn from Barberà et al. (2004) in their many-sorted FOL. As their induction result translated impossibilities generated on small, finite domains to full-blown impossibility results, they took advantage of these concrete, finite base cases to re-write the axioms in propositional logic (using the kind of rewriting that transformed formula (1) to formula (4) in Section 2.1). This, in turn, allowed them to use SAT solvers to search for subsets of axioms which generate impossibility results in these base cases; once found, the induction theorem generalized them to full impossibility results. Doing so for all base cases up to sets of eight items yielded 84 impossibility theorems from about one million combinations.[48]

Their results included known results (e.g. those of Kannai and Peleg (1984) and Barberà and Pattanaik (1984)); variations on known results, typically formed by strengthening axioms to reduce the impossibility's minimal domain; direct consequences of other results (as they did not prune implications of existing impossibilities); a trivial contradiction between the axioms of uncertainty aversion and uncertainty appeal; and – perhaps most interestingly – new theorems. These last resolved an open question in the literature, which we now describe.

Letting $\succ$ (resp. $\succsim$) denote strict (resp. weak) preference on individual choice objects (denoted by lower case letters), and $\rhd$ (resp. $\unrhd$) strict (resp. weak) preference on sets of objects (denoted by capital letters), Bossert et al. (2000) presented a theorem characterizing the min–max ordering in terms of four axioms. The min–max ordering is defined as

$$\begin{aligned} A \unrhd_{mnx} B \Leftrightarrow\ & [\min\{A\} \succ \min\{B\} \vee (\min\{A\} = \min\{B\} \\ & \wedge \max\{A\} \succsim \max\{B\})]\,; \end{aligned}$$

where $\min\{A\}$ is the minimal element of A with respect to $\succsim$ and $\max\{A\}$ the maximal element. Thus, a set A is weakly preferred under the min–max ordering to set B iff either the worst element of A is strictly preferred to that or B, or (when the worst elements are equally preferred) the best element of A is weakly preferred to that of B.

The four axioms were:

1. *Simple dominance*,

   $$x \succ y \Rightarrow (\{x\} \rhd \{x, y\} \wedge \{x, y\} \rhd \{y\})$$

   for all x and y, so that a set consisting of a strictly preferred object is preferred to a set containing it as well as a strictly less preferred object, which – in turn – is preferred to a set consisting only of that less preferred object.
2. *Independence*,

   $$A \rhd B \Rightarrow A \cup \{x\} \unrhd B \cup \{x\}$$

   for all A and B and x not contained in A or B. Thus, adding a single object to two sets ranked by strict preference does not reverse that ranking (but it may weaken it).
3. *Uncertainty aversion*,

   $$(x \succ y \succ z) \Rightarrow \{y\} \rhd \{x, z\}$$

   for all x, y and z, so that a set consisting only of an intermediately preferred object is strictly preferred to a set consisting of a strictly more favourable and a strictly less favourable object.
4. *Simple top monotonicity*,

   $$x \succ y \Rightarrow \{x, z\} \rhd \{y, z\}$$

   for all x, y and z such that $x \succ z$ and $y \succ z$, so that – if an object is strictly preferred to another – a set containing it and a third object is strictly preferred to a set containing the less preferred object and the third object.

Arlegi (2003) showed that the min–max ordering was, in fact, inconsistent with the independence axiom, and presented an alternative axiomatic basis for it. Geist and Endriss (2011) presented a complementary result to Arlegi's, finding a contradiction between the four original axioms at even four choice objects, thus establishing that the original four axioms are inconsistent, so cannot form the basis of any transitive binary relationship.

Geist and Endriss (2011) also presented the first impossibility result in this literature not to use any dominance axiom.

In cases of interest, the authors were able to quickly derive manual proofs for the computationally discovered results.[49]

[44] Represent preferences over three objects as a three-digit binary character, the first indicating whether $a \succ b$, the second whether $a \succ c$ and the third whether $b \succ c$. There are six permissible three digit numbers, 000, 001, 011, 100, 110 and 111, after eliminating the two cyclical ones. *IIA* then requires that each digit in the social preference is a function of the corresponding digits in the individual preferences alone. The 1-distance condition then allows only one of those digits to vary.

[45] In private correspondence, Sen has conjectured that the result of Malawski and Zhou (1994) linking Wilson's and Arrow's theorems may be an immediate consequence of Tang and Lin's.

[46] Geist (2010) had initially attempted an approach more akin to Grandi and Endriss (2012), seeking to derive an automated proof of the Kannai and Peleg theorem using three different first-order theorem provers; none of them was able to derive a proof after 120 h of CPU time on 2.26 GHz machines with 24 GB RAM.

[47] As a trivial example, the property that a structure contains three distinct elements cannot be preserved in substructures with fewer than three elements.

[48] Resource constraints limited them to eight items and 20 axioms. They derived their results in about one day.

[49] For the min–max ordering inconsistency, the manual proof is about a half-page long.

Finally, the large set of impossibility results allowed the authors to statistically consider the role of the various axioms. For example, the linear order axiom appeared in all theorems; the 'even-numbered extension of equivalence' and reflexivity occurred in none; 'intermediate independence' occurred in all results for seven or eight choice items, but never for fewer than five choice items.

Brandt and Geist (2016) extended the methodology of Geist and Endriss (2011) by performing an initial encoding in HOL, and then deriving implications capable of expression in propositional logic for small base cases. This allowed expression of more properties than was possible in the many-sorted FOL of Geist and Endriss (2011). Thus, Brandt and Geist (2016) could encode a neutrality axiom that Geist and Endriss (2011) could not, but at the cost of generating exponentially many new variables, restricting the size of cases that could be computed.

3.2. Auctions

Applications of mechanized reasoning to auction design and implementation are less sophisticated than those to social choice. Nevertheless, given auctions' practical importance, we expect that these will ultimately become more widespread. This section surveys work in two separate areas—applying mechanized reasoning to checking results in auction theory, and checking implementations of auction designs.

On the former, Vickrey's theorem has provided a basic testbed result. Section 4 illustrates in detail our Isabelle implementation. It therefore complements Lange et al. (2013), which compared implementations of Vickrey's theorem in four different mechanized reasoners.

Conceptually, as higher-order logic is sufficient to express all concepts in auction theory, it is not challenging to represent basic results in auction theory using a higher-order logic theorem prover like Isabelle. Doing so in more basic logics is both more conceptually challenging, and may offer more promise of automation.

In simpler logics, model checking can automatically establish properties of systems by exhaustively inspecting the system's state space. Tadjouddine et al. (2009) used SPIN, a widely-used commercial model checker based on a linear temporal logic (LTL), to verify Vickrey auctions' strategy-proofness property that bidders cannot do better than to bid their valuations.[50] They implemented two techniques to reduce the search space while verifying strategy-proofness for arbitrary bid ranges and numbers of agents: *program slicing* removed variables irrelevant to the property; *abstraction* discretized the domain of bids into a three-element domain, depending on whether a bid exceeded, equalled, or was less than an agent's valuation. A manual proof was required to establish the abstraction's soundness. Together, the two simplifications allowed strategy-proofness to be verified for any number of agents in a Vickrey auction in a quarter of a second.

The second branch of applications of mechanized reasoning to auctions has sought to establish properties of auction designs as implemented. This is of interest for at least two reasons: first, even if theoretical properties of an auction are known, errors may be introduced when translating the auction from a design to an operational auction. Second, and more commonly for modern auctions, practice may simply outstrip theory. In both cases, mechanized reasoning can be used to reduce the likelihood that an auction will fail when run.

Caminati et al. (2015) used Isabelle to prove that a combinatorial Vickrey auction is soundly specified, in the sense of guaranteeing that – whatever the bids received as input – the output allocated only the available goods, at non-negative prices, and assigned a unique output to each input. Furthermore, it implemented two parallel specifications of the auction, the first close to its standard paper specification, and the second a constructive one. Constructive definitions are essentially algorithmic descriptions. By contrast, definitions in classical logics need only state properties of the defined object. For instance, a classical definition of the maximum of a (non-empty) list of bids identifies an element of the list that is greater than or equal to every other element in the list. A constructive definition would begin by noting that – for a one-element list – the maximum is the single element of the list; it would then proceed recursively by computing the maximum of the remainder of the list. It would then return the larger of the two: the initial element, or the maximum of the remaining elements.

Isabelle was used to formally prove the equivalence of the two specifications. While the constructive specification is less intuitive, its algorithmic nature allows Isabelle to automatically generate verified executable code from it.

Model checking has also been used to examine auctions for evidence of shill bidding. Xu and Cheng (2007) used SPIN to define predicates corresponding to suspicious behaviour, including pushing prices to a reserve price before dropping out, and bidding on the higher priced of two identical goods. The model checker was then used to see whether the predicates were present in a finite dataset of actual bidding behaviour.

Arcos et al. (2005) developed a toolkit to verify properties of multi-agent environments, with a traditional open outcry auction as their leading example. Their toolkit implemented liveness checks to ensure that agents are not blocked (i.e. can bid in every round), that each bidding round can be reached, and that the final bidding round is reachable from any other, as well as correctness of the bidding language (that is, that by following the rules, the system always remains in a defined state). Their toolkit also includes a simulation tool that conducts a 'what-if' analysis by performing a complete check of all cases. While the authors themselves do not refer to what they do as model checking, that is what it most closely resembles.

Finally, Bai et al. (2014) consider the question of how potential users of online auctions can trust the auctions' protocols. They develop a protocol for specifying auction designs that can be read by Coq, a mechanized reasoner. Future work building on this should eventually allow Coq to verify properties claimed for the auction.

4. Blueprint of a formal proof of Vickrey's theorem

The preceding has provided an overview of mechanized reasoning, both in general, and as applied to economic problems. This section provides a detailed description of how a mechanized reasoner is used in practice, in this case to verify a formal proof of Vickrey's theorem. We use Vickrey's familiar theorem to focus attention on the formal proof's implementation, rather than the details of the result or proof.

We begin with a standard statement of Vickrey's theorem and proof, in this case from Maskin (2004):

Theorem 3 (*Vickrey 1961*)**.** *In a second-price auction, it is (weakly) dominant for each buyer i to bid its valuation v_i. Furthermore, the auction is efficient.*

[50] Tadjouddine et al. (2009) did not seem to use the modal capabilities of SPIN; instead, the authors seemed to adopt SPIN as they wished – in future work – to be able to accept C code as input, and to reason about it; reasoning about computer programs in which variables can be set does require modal capability.

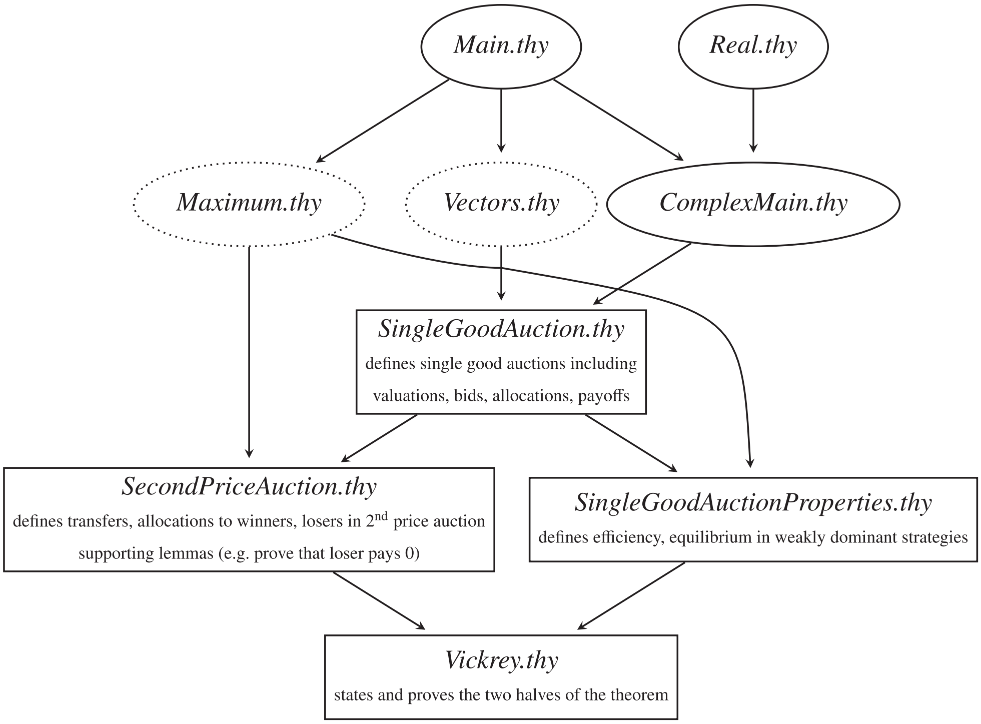
Main.thy
Real.thy
Maximum.thy
Vectors.thy
ComplexMain.thy
SingleGoodAuction.thy
defines single good auctions including valuations, bids, allocations, payoffs
SecondPriceAuction.thy
defines transfers, allocations to winners, losers in 2nd price auction supporting lemmas (e.g. prove that loser pays 0)
SingleGoodAuctionProperties.thy
defines efficiency, equilibrium in weakly dominant strategies
Vickrey.thy
states and proves the two halves of the theorem

Fig. 1. High level theory graph for the formal proof of Vickrey's theorem.

Proof #1. Suppose that buyer i bids $b_i < v_i$. The only circumstance in which the outcome for i is changed by its bidding b_i rather than v_i is when the highest bid b by other bidders satisfies $v_i > b > b_i$. In that event, buyer i loses by bidding b_i (for which its net payoff is 0) but wins by bidding v_i (for which its net payoff is $v_i - b$). Thus, it is *worse* off bidding $b_i < v_i$. By symmetric argument, it can only be worse off bidding $b_i > v_i$. We conclude that bidding its valuation (truthful bidding) is weakly dominant. Because it is optimal for buyers to bid truthfully and the high bidder wins, the second-price auction is efficient. □

However intelligible to humans, Maskin's proof is too stylized for computers: that there is only one circumstance in which changing bids changes the outcome is merely asserted; the "symmetric argument" is not explicitly elaborated. Before formalizing it, we therefore elaborated the paper proof, and restructured it to four cases, rather than the original nine:

Proof #2. Let N be the set of bidders, and suppose bidder i bids $b_i = v_i$, whatever b_j each other bidder $j \neq i$ bids. There are two cases:

1. i wins. This implies $b_i = v_i = \max_{j\in N}\{b_j\}$, $p_i = \max_{j\in N\setminus\{i\}}\{b_j\}$, and $u_i(\boldsymbol{b}) = v_i - p_i \geq 0$. Now consider i submitting an arbitrary bid $\hat{b}_i \neq b_i$ so that the bid vector is $\left(b_1, \ldots, b_{i-1}, \hat{b}_i, b_{i+1}, \ldots, b_n\right)$. This has two sub-cases:
   (a) i wins with $\hat{b}_i$, so that $u_i\left(b_1, \ldots, b_{i-1}, \hat{b}_i, b_{i+1}, \ldots, b_n\right) = u_i(\boldsymbol{b})$: i receives the same utility from winning the item, and pays the same price as the second highest bid has not changed.
   (b) i loses with $\hat{b}_i$, so that $u_i\left(b_1, \ldots, b_{i-1}, \hat{b}_i, b_{i+1}, \ldots, b_n\right) = 0 \leq u_i(\boldsymbol{b})$.
2. i loses. This implies $p_i = 0$, $u_i(\boldsymbol{b}) = 0$, and $b_i \leq \max_{j\in N\setminus\{i\}}\{b_j\}$ as, otherwise, i would have won. This yields again two cases for i's alternative bid $\hat{b}_i$:
   (a) i wins, so that $u_i\left(b_1, \ldots, b_{i-1}, \hat{b}_i, b_{i+1}, \ldots, b_n\right) = v_i - \max_{j\in N\setminus\{i\}}\{b_j\} = b_i - \max_{j\in N\setminus\{i\}}\{b_j\} \leq 0 = u_i(\boldsymbol{b})$.
   (b) i loses, so that $u_i\left(b_1, \ldots, b_{i-1}, \hat{b}_i, b_{i+1}, \ldots, b_n\right) = 0 = u_i(\boldsymbol{b})$.

By analogy for all i, $\boldsymbol{b} = \boldsymbol{v}$ supports an equilibrium in weakly dominant strategies. Efficiency is immediate: the highest bidder has the highest valuation. □

To formally prove Vickrey's theorem, we used Isabelle, whose higher-order logic allows our formalization to remain close to paper mathematics.

Our proof, *Vickrey.thy*, is a 9 KB, 185 line file that draws on five ancillary files written for this project.[51] All six files amount to 17 KB and 404 lines—much longer than their paper counterparts. A more reliable estimate of the additional effort involved in formal proofs, the *de Bruijn factor* (Wiedijk, 2012), cleans and compresses files before dividing the size of the code by the size of an informal TEXsource. It thus avoids bias by semantically irrelevant differences in the syntaxes of formalizations such as languages or code styles using different lengths of lines or of identifiers. The de Bruijn factor relating Proof #2 and its definitions (including max) to our Isabelle code is 1.1; as our TEX source is more elaborate than usual, this is lower than the typically observed factors of around four.

Fig. 1 depicts the files used in the proof. Those already in Isabelle's library are marked by ellipses. Dotted ellipses denote files containing general definitions and lemmas that we have added to Isabelle's library. Rectangles denote this paper's auction-specific files. Directed edges denote dependence, with the source code being imported into the target code.

Vickrey.thy begins with *vickreyA*, which proves that truth telling is weakly dominant in Vickrey auctions:

[51] See https://github.com/formare/auctions/tree/master/isabelle/Auction for the code.

```
theorem vickreyA :
  fixes N :: “participant set” and v :: valuations and A :: single_good_auction
  assumes val : “valuations N v”
  defines “b ≡ v”
  assumes spa : “second_price_auction A” and card_N : “card N > 1”
  shows “equilibrium_weakly_dominant_strategy N v b A”
```

The **fixes** keyword applies the theorem to any N, v and A of the given types. The type *single_good_auction* is defined as an *input* × *output* relation, with the bidders and their bids as input, and a Boolean allocation vector and a vector of transfers as outcome.[52] The *valuations* type is defined elsewhere to be a vector of real numbers. The **assumes** keyword on the next line states that the theorem holds under an assumption labelled *val*, namely that in the vector v of N real numbers, all numbers are non-negative (this defined at another place as the definition of ‘valuations’).

Next, the **defines** declaration equates bids and valuations. The following **assumes** keyword introduces and labels further assumptions (e.g. A is a second-price auction; N contains more than one bidder). The **shows** keyword states the theorem: N agents participating in auction A, with valuations v and bids b (equated to valuations) yields an equilibrium in weakly dominant strategies.

SingleGoodAuctionProperties.thy defines the equilibrium concept:

```
definition equilibrium_weakly_dominant_strategy ::
  “participant set ⇒ valuations ⇒ bids ⇒ single_good_auction ⇒ bool” where
  “equilibrium_weakly_dominant_strategy N v b A ⟷
    valuations N v ∧ bids N b ∧ single_good_auction A ∧
    (∀i ∈ N .
      (∀whatever_bid . bids N whatever_bid ⟶
        (let b′ = whatever_bid(i := b i)
          in (∀x p x′ p′ . ((N, whatever_bid), (x, p)) ∈ A ∧ ((N, b′), (x′, p′)) ∈ A
            ⟶ payoff (v i) (x′ i) (p′ i) ≥ payoff (v i) (x i) (p i)))))”
```

The definition’s second line declares the type of the *equilibrium_weakly_dominant_strategy* to be a (Boolean) predicate whose arguments are a set of participants, a valuation vector, a bid vector, and an auction.[53] The definition’s body states that the predicate, given arguments N, v, b and A, evaluates to true if and only if the remaining expression does. The expressions in the subsequent line ensure that all arguments have admissible values. Similarly, our first step when introducing *whatever_bid* is to ensure that it is an admissible bid vector. The *whatever_bid*$(i := b\ i)$ notation then takes an arbitrary vector and replaces its ith component with i’s bid $b\ i$ (which the theorem equates to i’s valuation).[54]

We denote the outcome of an arbitrary bid (*whatever_bid*) by $(\boldsymbol{x}, \boldsymbol{p})$, while $\left(\boldsymbol{x}', \boldsymbol{p}'\right)$ denotes that of i’s original bid and arbitrary bids by agents $j \neq i$. To satisfy the definition of an equilibrium in weakly dominant strategies, the outcome $\left(\boldsymbol{x}', \boldsymbol{p}'\right)$ of i’s truthful bid must yield a payoff no less than that resulting from an arbitrary bid. The **let** $\cdots$ **in** $\cdots$ notation[55] introduces local abbreviations, which can only be accessed within the **in** block; here, this makes the expression $((N, b'), (x', p')) \in A$ more readable.

The code snippet below formalizes case 2b of Proof #2. It is *declarative*, resembling a textbook proof. *Procedural* proofs, by contrast, prescribe *tactics* to apply, thus more resembling the *process* humans use to find proofs. In either case, each theorem creates a *proof obligation*, or a *goal*; these may be broken into *subgoals* (e.g. by case distinction); the set of local proof obligations implied by these subgoals are stored on a *goal stack*.

Proof #3.

```
 1  proof –
 2    (∗ · · · ∗)
 3    {
 4      fix i :: participant
 5      assume i_range : “i ∈ N”
 6      (∗ · · · ∗)
 7      let ?b = “whatever_bid(i := b i)”
 8      (∗ · · · ∗)
 9      have weak_dominance : “payoff (v i) (x′ i) (p′ i) ≥ payoff (v i) (x i) (p i)”
10      proof cases
11        assume non_alloc : “x′ i ≠ 1”
12        with spa_pred′ i_range have “x′ i = 0” using spa_allocates_binary by blast
13        with spa_pred′ i_range have loser_payoff : “payoff (v i) (x′ i) (p′ i) = 0”
14          by (rule second_price_auction_loser_payoff)
```

[52] This can be seen from expressions such as $((N, b'), (x', p')) \in A$.

[53] The $A \Rightarrow B \Rightarrow C$ notation, referred to as *currying*, is equivalent to $A \times B \rightarrow C$, but is conceptually simpler as it does not require definition of a × operation.

[54] The code snippet contains various instances of “.”: these are separators that improve readability.

[55] We use “$\cdots$” to distinguish the standard use of ellipses from Isabelle’s “...” notation, whose meaning we introduce when explaining line 30 of the following code snippet.

```
15       have i_bid_at_most_second : “?b i ≤ ?b_max′”
16       proof (rule ccontr)
17         assume “¬?thesis”
18         then have “?b i > ?b_max′” by simp
19         with defined spa_pred′ i_range have “second_price_auction_winner N ?b x′ p′ i”
20           by (simp add : only_max_bidder_wins)
21         with non_alloc show False
22           unfolding second_price_auction_winner_def
23             second_price_auction_winner_outcome_def by blast
24       qed
25       show ?thesis
26       proof cases
27         assume “x i ≠ 1”
28         then have “x i = 0” by (rule spa_allocates_binary′)
29         with spa_pred i_range have “payoff (v i) (x i) (p i) = 0”
30           by (rule second_price_auction_loser_payoff)
31         also have “… = payoff (v i) (x′ i) (p′ i)” using loser_payoff ..
32         finally show ?thesis by (rule eq_refl)
33       next
34         (∗ · · · ∗)
35       qed
36     next
37       (∗ · · · ∗)
38     qed
39   }
40   (∗ · · · ∗)
41 qed
```

□

The **proof** keyword starts the proof. Invoked alone, Isabelle would automatically select inference rules to apply. **proof**— performs manual inference. Alternatively, one can specify existing inference rules:

- **proof** *cases* (lines 10 and 26) makes a case distinction; analysis of each case concludes by **show**ing that the desired thesis holds; **qed** clears the goal stack; **next** begins the next case.
- **proof** *(rule ccontr)* (line 16) undertakes proof by contradiction, culminating in **show** *False*.

The proof considers an arbitrary but fixed participant i, which is introduced locally with the **fix** keyword, and assumed to be in the admissible range N for bidders.[56]

The **have** statements establish local facts, generating local proof obligations, which have to be discharged by corresponding **proof**s. Here, the *cases* proof establishes that $u_i(\ldots, v_i, \ldots) \geq u_i(\ldots, b_i, \ldots)$. This proof makes use of further facts, omitted to keep the snippet readable: *spa_pred* and *spa_pred′* state that $((N, \mathit{whatever\_bid}), (x, p))$ and $((N, ?b), (x', p'))$ respectively are in an $(\mathit{input}, \mathit{outcome})$ relationship of a second price auction with each other.[57] *defined* states that a vector with one component per element of the (finite) set N has a well-defined maximum component.

Both **from** and **using** introduce facts to discharge the **have** obligations. The **by** keyword invokes an automated proof method, instead of discharging proof obligations by explicit declarative means. Isabelle thus combines ATP and ITP methods.

1. *simp* (lines 18 and 20) simplifies (e.g. $x \wedge x = x$) the statement to be proved. Line 20 supplies a simplification rule of our own, *only_max_bidder_wins* .
2. *blast* (lines 12 and 13) “is (in principle) a complete proof procedure for first-order formulas” (Nipkow, 2015). In practice, *blast* either succeeds, fails, or – giving a practical example of semi-decidability – runs until the user cancels it.
3. *rule* (lines 14, 16, 28, 30 and 32) applies the given lemma as an inference rule. In line 31, “..” abbreviates **by** *rule*, which automatically applies a matching inference rule.

While interactively developing the proof, we employed the **try** and **try0** commands, which apply a range of automated methods, to find the most appropriate proof methods. Automated calls can always be replaced by explicit declarative steps; Isabelle's Sledgehammer tool (Blanchette and Paulson, 2015) can sometimes provide them automatically.

The **assume** · · · **then have** constructions (lines 17 and 18, and 27 and 28) list assumptions **then** state the proof obligations. Line 17's identifier *?thesis* refers to the proof obligation at the proof's current level of reasoning.

Lines 22–23's **unfolding** also performs substitutions, replacing stated concepts' names with the bodies of their definitions. Unlike abbreviations with *?*, the latter are semantic definitions, of which the reasoner make use (e.g. *second_price_auction_winner_def* is restated in terms of $i \in N, i \in \arg\max \boldsymbol{b}, \ldots$).

Lines 29–32's **have** · · · **also have** · · · **finally show** construction allows chains of reasoning with equality before discharging a proof obligation: the “…” following the **also have** are replaced by

[56] In Isabelle, the descriptive form of a verb (e.g. **fixes**, **assumes** or **shows**) are often used when stating theorems, while their imperative counterparts (e.g. **fix**, **assume** or **show**) are used locally in proofs.

[57] Isabelle syntactically substitutes identifiers starting with *?* by other, usually more complex expressions before checking a proof step. Syntactic substitution is performed, for example, by the preprocessor of many programming languages, allowing the programmer to use shorthand designations rather than writing complicated expressions in full. It is distinct from the semantic equation of two variables, as in “$b \equiv v$”.

the right hand side of the previous **have** statement. In line 31, this establishes that i receives zero given valuation v_i and either $(\boldsymbol{x}, \boldsymbol{p})$, or $(\boldsymbol{x}', \boldsymbol{p}')$.

5. Discussion

The decade since the mechanized reasoning community became interested in economic applications has seen rapid progress. When Nipkow reported on his formalization of Arrow's theorem, he agreed that "[s]ocial choice theory turns out to be perfectly suitable for mechanical theorem proving", but felt that it was "unclear if [it] will lead to new insights into either social choice theory or theorem proving" (Nipkow, 2009). However, that very year Tang and Lin (2009) used mechanized reasoning to discover a new theorem that subsumes Arrow's, which Chatterjee and Sen (2014) believed to be novel, and unlikely to have been found with traditional methods. Shortly thereafter, Geist and Endriss (2011) contributed their 84 impossibility theorems.

If mechanized reasoning is to make further inroads into economics it must be sensitive to a number of concerns. First, economics has no proofs of comparable complexity or length to significant results in modern mathematics. Thus, the question of whether a proof will exceed the capability of human theorists to verify is less of a concern than in mathematics. Further, it is unclear that there have been any disastrous cases of mistaken proofs within economics; instead, our greater errors likely result from poor modelling in the first place, and coding or data errors in econometrics.

Second, even when mechanized reasoners have helped identify new results, economic theorists may dismiss them as unmotivated, non-transparent or lacking insight.[58] Even, however, in the worst case, we believe that a stock of poorly-motivated, non-transparent theorems generated blindly by computer provide cases for us to think about and reason with: the presence of the intermediate independence axiom in all of the larger impossibility theorems found by Geist and Endriss (2011) should provide precisely the sort of hunch that sets us sharpening our pencils.

We close by suggesting some further possible applications of mechanized reasoning to economic problems.

First, there are open problems in auction theory that seem amenable to solution by computation (rather than 'reasoning'). For example, the simplest formulation of optimal multi-object auctions (q.v. Armstrong, 2000) defines a linear programming problem that quickly becomes too large to solve manually as the number of items increases.[59] As efficient algorithms exist for solving linear programming problems, *automated mechanism design* (q.v. Conitzer and Sandholm, 2003) has already begun to address the purely computational aspects of optimal mechanism design. As formal methods can be used to verify the results of computations (q.v. Gonthier, 2008; Hales et al., 2015), proofs in automated mechanism design could also be verified by formal methods.

Second, we believe that the exhaust-then-induct technique pioneered by Tang and Lin (2009), and developed by Geist and Endriss (2011), offers the promise of automating search for theorems in other areas of economic theory. The formal similarities between social choice and matching theory – including a reliance on discrete objects – suggests that this technique could be applied directly to the latter. Although auction theory appears richer in its use of continuous objects (prices), there is a small literature establishing results by induction (Chew and Serizawa, 2007; Morimoto and Serizawa, 2015; Adachi, 2014; Kato et al., 2015); the possibility of coupling their induction steps with computational exhaustion has not been explored.

However these tools are applied within economics, it is hard to imagine them not becoming more important, as the tools themselves become faster and easier to use, as they gain acceptance within the pure mathematics community, and as the mechanized reasoning community seeks more applications for them.

References

Adachi, T., 2014. Equity and the Vickrey allocation rule on general preference domains. Soc. Choice Welf. 42 (4), 813–830.

Appel, A., Haken, W., 1977. Every planar map is four colorable part I: Discharging. Illinois J. Math. 21 (3), 429–490.

Appel, K., Haken, W., Koch, J., 1977. Every planar map is four colorable part II: Reducibility. Illinois J. Math. 21 (3), 491–567.

Arcos, JL., Esteva, M., Noreiga, P., Rodríguez-Aguilar, JA., Sierra, C., 2005. Engineering open environments with electronic institutions. Eng. Appl. Artif. Intell. 18, 191–204.

Arlegi, R., 2003. A note on Bossert, Pattanaik and Xu's "Choice under complete uncertainty: axiomatic characterization of some decision rules". Econom. Theory 22 (1), 219–225.

Armstrong, M., 2000. Optimal multi-object auctions. Rev. Econom. Stud. 67 (3), 455–481.

Armstrong, M., Rochet, J.-C., 1999. Multi-dimensional screening: a user's guide. Eur. Econ. Rev. 43 (4–6), 959–979.

Avigad, J., Harrison, J., 2014. Formally verified mathematics. Commun. ACM 57 (4), 66–75.

Aygün, O., Sönmez, T., 2013. Matching with contracts: comment. Amer. Econ. Rev. 103 (5), 2050–2051.

Bai, W., Tadjouddine, E.M., Guo, Y., 2014. Enabling automatic certification of online auctions. In: Buhnova, J.K.B., Happe, L. (Eds.), Proceedings 11th International Workshop on Formal Engineering Approaches to Software Components and Architectures. pp. 123–132. http://dx.doi.org/10.4204/EPTCS.147.9.

Barberà, S., 1980. Pivotal voters: A new proof of Arrow's theorem. Econom. Lett. 6 (1), 13–16.

Barberà, S., 1983. Strategy-proofness and pivotal voters: A direct proof of the Gibbard-Satterthwaite theorem. Internat. Econom. Rev. 24 (2), 413–417.

Barberà, S., Bossert, W., Pattanaik, P.K., 2004. Ranking sets of objects. In: Barberà, S., Hammond, P.J., Seidl, C. (Eds.), Handbook of Utility Theory, vol. II. Kluwer Academic Publishers, Dordrecht, pp. 893–977.

Barberà, S., Pattanaik, P.K., 1984. Extending an order on the set to the power set: some remarks on Kannai and Peleg's approach. J. Econom. Theory 32 (1), 185–191.

Blanchette, J.C., Paulson, L.C., 2015. Hammering Away. A User's Guide to Sledgehammer for Isabelle/HOL. URL http://isabelle.in.tum.de/dist/doc/sledgehammer.pdf.

Blume, L., Easley, D., Kleinberg, J., Kleinberg, R., Tardos, Éva, 2015. Introduction to computer science and economic theory. J. Econom. Theory 156, 1–13.

Boldo, S., Jourdan, J.-H., Leroy, X., Melquiond, G., 2013. A Formally-Verified C compiler supporting floating-point arithmetic. In: Nannarelli, A., Seidel, P.-M., Tang, P.T.P. (Eds.), Arith - 21st IEEE Symposium on Computer Arithmetic. IEEE, Austin, United States, pp. 107–115. URL https://hal.inria.fr/hal-00743090.

Bordeaux, L., Hamadi, Y., Zhang, L., 2006. Propositional satisfiability and constraint programming: a comparative survey. ACM Comput. Surv. 38 (4).

Bossert, W., Pattanaik, P., Xu, Y., 2000. Choice under complete uncertainty: axiomatic characterizations of some decision rules. Econom. Theory 16 (2), 295–312.

Brandt, F., Geist, C., 2016. Finding strategy proof social choice functions via SAT solving. J. Artificial Intelligence Res.

Buchberger, B., 2006. Mathematical theory exploration. In: Automated Reasoning, Third International Joint Conference, IJCAR 2006, Seattle, WA, USA, August 17–20, 2006, Proceedings. pp. 1–2. http://dx.doi.org/10.1007/11814771_1.

Burch, J.R., Clarke, E.M., McMillan, K.L., Dill, D.L., Hwang, J., 1990. Symbolic model checking: 10^{20} states and beyond. In: Proceedings of the 5th Annual Symposium on Logic in Computer Science. IEEE Computer Society Press.

Caminati, M.B., Kerber, M., Lange, C., Rowat, C., M. Feldman, M. Schwarz, and T. Roughgarden (Eds.), 2015. Sound auction specification and implementation. In: Economics and Computation. 16th ACM Conference, EC'15. (Portland, Oregon, USA, June 15–19, 2015). Ed. by M. Feldman, M. Schwarz, and T. Roughgarden.

Chatterjee, S., Sen, A., 2014. Automated reasoning in social choice theory—some remarks. Math. Comput. Sci. 8 (1), 5–10.

Chew, S.H., Serizawa, S., 2007. Characterizing the Vickrey combinatorial auction by induction. Econom. Theory 33 (2), 393–406.

Clarke, E.M., Emerson, E.A., Sistla, A.P, 1986. Automatic verification of finite-state concurrent systems using temporal logic specifications. ACM Trans. Program. Lang. Syst. 8 (2), 244–263. http://dx.doi.org/10.1145/5397.5399.

[58] See Avigad and Harrison (2014, p. 73) for a discussion of the tension between rigour and insight in pure mathematics.

[59] See Armstrong and Rochet (1999) for the equivalent multi-dimensional screening problem for a monopolist.

Clarke, E.M., Grumberg, O., Long, D.E., 1994. Model checking and abstraction. ACM Trans. Program. Lang. Syst. 16 (5), 1512–1542. http://dx.doi.org/10.1145/186025.186051.

Colton, S., Bundy, A., Walsh, T., 1999. Automatic concept formation in pure mathematics. In: Proceedings of the 16th International Joint Conference on Artificial Intelligence - IJCAI'99. Morgan Kaufmann Pub Inc., pp. 786–791.

Colton, S., Bundy, A., Walsh, T., 2000. Automatic invention of integer sequences, http://www.doc.ic.ac.uk/~sgc/html_papers/colton_aaai00.html.

Conitzer, V., Sandholm, T., 2003. Applications of automated mechanism design. In: UAI-03 workshop on Bayesian Modeling Applications. Acapulco, Mexico.

Dahn, B.I., 1998. Robbins algebras are Boolean: a revision of McCune's computer-generated solution of Robbins' problem. J. Algebra 208.

Dick, S., 2011. AfterMath: The work of proof in the age of human-machine collaboration. Isis 102 (3), 494–505.

Dick, S., 2015. After math: Following mathematics into the digital. Presentation to Microsoft Research New England.

Gabbay, D.M., Guenthner, F. (Eds.), 2001/2014. Handbook of Philosophical Logic, second ed., vol. 1–17. Springer-Verlag.

Gardner, M., 1952. Logic machines. Sci. Am. 186 (3), 68–73.

Geanakoplos, J.D., 2005. Three brief proofs of arrow's impossibility theorem. Econom. Theory 26 (1), 211–215.

Geanakoplos, J.D., 2001. Three brief proofs of arrow's impossibility theorem, Discussion Paper 1123RRR. Cowles Foundation, New Haven.

Geist, C., 2010. Automated Search for Impossibility Theorems in Choice Theory: Ranking Sets of Objects (M.Sc. thesis), Institute for Logic, Language and Computation: Universiteit van Amsterdam.

Geist, C., Endriss, U., 2011. Automated search for impossibility theorems in social choice theory: ranking sets of objects. J. Artificial Intelligence Res. 40, 143–174.

Gonthier, G., 2008. Formal proof—the four color theorem. Notices Amer. Math. Soc. 55 (11), 1382–1393.

Gonthier, G., Asperti, A., Avigad, J., Bertot, Y., Cohen, C., Garillot, F., Le Roux, S., Mahboubi, A., O'Connor, R., Ould Biha, S., Pasca, I., Rideau, L., Solovyev, A., Tassi, E., Théry, L., 2013. A machine-checked proof of the odd order theorem. In: Blazy, S., Paulin, C., Pichardie, D. (Eds.), ITP 2013, 4th Conference on Interactive Theorem Proving. In: LNCS, vol. 7998. Springer, Rennes, France, pp. 163–179. http://dx.doi.org/10.1007/978-3-642-39634-2_14, URL http://hal.inria.fr/hal-00816699.

Ågotnes, T., van der Hoek, W., Wooldridge, M., 2011. On the logic of preference and judgment aggregation. Auton. Agents Multi Agent Syst. 22 (1), 4–30.

Grandi, U., Endriss, U., 2012. First-order logic formalisation of impossibility theorems in preference aggregation. J. Philos. Logic 1–24. http://dx.doi.org/10.1007/s10992-012-9240-8, English.

Hales, T.C., 2005. A proof of the Kepler conjecture. Ann. of Math. 162 (3), 1063–1185.

Hales, T.C., 2012. Dense Sphere Packings. A Blueprint for Formal Proofs. In: London Mathematical Society Lecture Note Series, Cambridge University Press.

Hales, T., Adams, M., Bauer, G., Dat, D.T., Harrison, J., Truong, H.L., Kaliszyk, C., Magron, V., McLaughlin, S., Thang, N.T., Truong, N.Q., Nipkow, T., Obua, S., Pleso, J., Rute, J., Alexey Solovyev, T.T. H.A., Trung, T.N., Diep, T.T., Urban, J., Ky, V.K., Zumkeller, R., 2015. A formal proof of the Kepler conjecture, arXiv preprint arXiv:1501.02155.

Harrison, J., 2007. A short survey of automated reasoning. In: Anai, H., Horimoto, K., Kutsia, T. (Eds.), Proceedings of the Second International Conference on Algebraic Biology, AB 2007. In: Lecture Notes in Computer Science, vol. 4545. Castle of Hagenberg, Austria, pp. 334–349.

Harrison, J., 2006. Floating-Point verification using theorem proving. In: Bernardo, M., Cimatti, A. (Eds.), Formal Methods for Hardware Verification 6th International School on Formal Methods for the Design of Computer, Communication, and Software Systems, SFM 2006. In: Lecture Notes in Computer Science, vol. 3965. Springer Verlag, Bertinoro, Italy, pp. 211–242.

Hatfield, J.W., Milgrom, P.R., 2005. Matching with contracts. American Economic Review 95 (4), 913–935.

Henkin, L., Monk, J.D., Tarski, A., 1971. Cylindric Algebras, Part I. In: Studies in Logic, vol. 64. North Holland.

Hoffmann, D.W., 2013. Die Grenzen der Mathematik—Die Gödel'schen Unvollständigkeitssätze. Springer-Verlag.

Ignatovich, D.A., Passmore, G.O., 2015. Case Study: 2015 SEC Fine Against UBS ATS, London, Aesthetic Integration.

Kannai, Y., Peleg, B., 1984. A note on the extension of an order on a set to the power set. J. Econom. Theory 32 (1), 172–175.

Karp, R.M., 1972. Reducibility among combinatorial problems. In: Miller, R.E., Thatcher, J.W. (Eds.), Complexity of Computer Computations. Plenum, New York, pp. 85–103.

Kato, M., Ohseto, S., Tamura, S., 2015. Strategy-proofness versus symmetry in economies with an indivisible good and money. Internat. J. Game Theory 44 (1), 195–207.

Kerber, M., Lange, C., Rowat, C., 2014. A Formal Proof of Vickrey's Theorem by Blast, Simp, and Rule. Working Paper 14-01. University of Birmingham, Department of Economics. URL http://ssrn.com/abstract=2376205.

Lander, L.J., Parkin, T.R., 1966. Counterexample to Euler's conjecture on sums of like powers. Bull. Amer. Math. Soc. 72 (6), 1079.

Lange, C., 2013. Ontologies and languages for representing mathematical knowledge on the semantic web. Semant. Web J. 4 (2), 119–158. http://dx.doi.org/10.3233/SW-2012-0059, URL http://www.semantic-web-journal.net/content/ontologies-and-languages-repres%enting-mathematical-knowledge-semantic-web.

Lange, C., Caminati, M.B., Kerber, M., Mossakowski, T., Rowat, C., Wenzel, M., Windsteiger, W., 2013. A qualitative comparison of the suitability of four theorem provers for basic auction theory. In: Carette, J., Aspinall, D., Lange, C., Sojka, P., Windsteiger, W. (Eds.), Lecture Notes in Computer Science. Springer, Bath, UK, pp. 200–215. http://dx.doi.org/10.1007/978-3-642-39320-4, arXiv:1303.4193 [cs.LO].

Leibniz, G.W., 1960. Projet et essais pour arriver à quelque certitude pour finir une bonne partie des disputes et pour avancer l'art d'inventer. In: Berka, K., Kreisler, L. (Eds.), Logik-Texte: Kommentierte Auswahl zur Geschichte der modernen Logik chapterI.1. Akademie-Verlag, Berlin, Deutschland, pp. 15–17. Deutsche Übersetzung aus G. W. Leibniz, Fragmente zur Logik.

Lenat, D.B., 1976. AM: An Artificial Intelligence Approach to Discovery in Mathematics as Heuristic Search (Ph.D. thesis), In: AIM-286, STAN-CS-76-570, and Heuristic Programming Project Report HPP-76-8, AI Lab, Stanford University, Stanford, California, USA.

Lenat, D.B., 1983. EURISKO: A program that learns new heuristics and domain concepts. Artifical Intelligence 21, 61–98.

Lucas, W.F., 1968. A game with no solution. Bull. Amer. Math. Soc. 74 (2), 237–239.

MacLane, S., 1986. Mathematics: Form and Function. Springer-Verlag.

Malawski, M., Zhou, L., 1994. A note on social choice theory without the Pareto principle. Soc. Choice Welf. 11 (2), 103–107.

Maskin, E., 2004. The unity of auction theory: Milgrom's master class. J. Econ. Lit. 42 (4), 1102–1115.

McCarthy, J., Hayes, P., 1969. Some philosophical problems from the standpoint of artificial intelligence. Machine Intelligence 4, 463–502.

McCorduck, P., 2004. Machines Who Think, second ed. AK Peters.

McCune, W., 1997. Solution of the Robbins problem. J. Automat. Reason. 19 (3), 263–276.

Morimoto, S., Serizawa, S., 2015. Strategy-proofness and efficiency with non-quasilinear preferences: a characterization of minimum price Walrasian rule. Theor. Econ. 10 (2), 445–487.

Neumann, J. von, Morgenstern, O., 1953. Theory of Games and Economic Behavior, second ed. Princeton University Press.

Newell, A., 1981. In: Groner, Rudolf, Groner, Marina, Bishoof, Walter F. (Eds.), The Heuristic of George Polya and its Relation to Artificial Intelligence. Tech. Rep. CMU-CS-81-133. In: Methods of Heuristics, Lawrence Erlbaum, Hillsdale, New Jersey, USA, p. 195–243, Department of Computer Science, Carnegie-Mellon University, Pittsburgh, Pennsylvania, USA.

Newell, A., Simon, H.A., 1956. The Logic Theory Machine: A Complex Information Processing System, Tech. Rep. The RAND Corporation.

Nipkow, T., 2009. Social choice theory in HOL: Arrow and Gibbard-Satterthwaite. J. Automat. Reason. 43 (3), 289–304.

Nipkow, T., 2015. Programming and proving in Isabelle/HOL, URL http://www.cl.cam.ac.uk/research/hvg/Isabelle/dist/Isabelle/doc/prog-prove.pdf.

Pólya, G., 1945. How to Solve It. Princeton, New Jersey, USA.

Pólya, G., 1954. Mathematics and Plausible Reasoning—Induction and Analogy in Mathematics. Princeton University Press, Princeton, New Jersey, USA.

Reny, P.J., 2001. Arrow's theorem and the Gibbard-Satterthwaite theorem: a unified approach. Econom. Lett. 70 (1), 99–105.

Robinson, J.A., 1965. A machine-oriented logic based on the resolution principle. J. Assoc. Comput. Mach. 12 (1), 23–41.

Suzumura, K., 2000. Welfare economics beyond welfarist-consequentialism. Japan. Econ. Rev. 51 (1), 1–32.

Tadjouddine, E.M., Guerin, F., Vasconcelos, W., 2009. Abstracting and verifying strategy-proofness for auction mechanisms. In: Baldoni, M., Son, T.C., Riemsdijk, M.B., Winikoff, M. (Eds.), Declarative Agent Languages and Technologies VI, vol. 5397. Springer Verlag, pp. 197–214.

Tang, P., 2010. Computer-aided Theorem Discovery—A New Adventure and its Application to Economic Theory (Ph.D. dissertation), Hong Kong University of Science Technology.

Tang, P., Lin, F., 2009. Computer-aided proofs of arrow's and other impossibility theorems. Artificial Intelligence 173 (11), 1041–1053.

Tang, P., Lin, F., 2011a. Discovering theorems in game theory: two-person games with unique pure nash equilibrium payoffs. Artificial Intelligence 175 (14–15), 2010–2020.

Tang, P., Lin, F., 2011b. Two equivalence results for two-person strict games. Games Econom. Behav. 71 (2), 479–486.

Turing, A.M., 1936. On computable numbers, with an application to the entscheidungsproblem. Proc. Lond. Math. Soc. Second Ser. 42, 230–265.

Vijay D'Silva Daniel Kroening, G.W., 2008. A survey of automated techniques for formal software verification. IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. 27 (7), 1165–1178.

Wang, H., 1960. Toward mechanical mathematics. IBM J. Res. Dev. 4 (1), 2–22.

Whitehead, A.N., Russell, B., 1910. Principia Mathematica, Vol. I. Cambridge University Press, Cambridge, UK.

Wiedijk, F., 2008. Formal proof: getting started. Notices Amer. Math. Soc. 55 (11), 1408–1414.

Wiedijk, F., 2007. Arrow's impossibility theorem. J. Formaliz. Math. 15 (4), 171–174.

Wiedijk, F., 2009. Formalizing arrow's theorem. Sādhanā 34 (1), 193–220.

Wiedijk, F., 2014. Formalizing 100 theorems, URL http://www.cs.ru.nl/~freek/100/.

Wiedijk, F., 2012. The "de Bruijn factor". URL http://www.cs.ru.nl/~freek/factor/.

Woodcock, J., Larsen, P.G., Bicarregui, J., Fitzgerald, J., 2009. Formal method: practice and experience. ACM Comput. Surv. 41 (4), 1–40.

Xu, H., Cheng, Y.-T., 2007. Model checking bidding behaviors in internet concurrent auctions. Int. J. Comput. Syst. Sci. Eng. 22 (4), 179–191.