The Knowledge Engineering Review, Vol. 26:1, 61–69. © Cambridge University Press, 2011
doi:10.1017/S0269888910000421

Logical mechanism design

IYAD RAHWAN[1, 2] and KATE LARSON[3]

[1]*Computing & Information Science Program, Masdar Institute of Science & Technology, Abu Dhabi, UAE;*
[2]*The Media Lab, Massachusetts Institute of Technology, Cambridge, 02139 MA, USA;*
e-mail: irahwan@acm.org;
[3]*Cheriton School of Computer Science, University of Waterloo, Waterloo, ON, N2L 3G1, Canada;*
e-mail: klarson@cs.uwaterloo.ca

Abstract

Game theory is becoming central to the design and analysis of computational mechanisms in which multiple entities interact strategically. The tools of *mechanism design* are used extensively to engineer incentives for truth revelation into resource allocation (e.g. combinatorial auctions) and preference aggregation protocols (e.g. voting). We argue that mechanism design can also be useful in the design of logical inference procedures. In particular, it can help us understand and engineer inference procedures when knowledge is distributed among self-interested agents. We set a research agenda for this emerging area, and point to some early research efforts.

1 Introduction

Game theory is becoming increasingly important in the design and analysis of computational mechanisms in which multiple entities interact strategically. In particular, the tools of *mechanism design* are now used extensively to engineer incentives for truth revelation in multi-agent interaction, such as auctions and preference aggregation.

However, the design of logical inference procedures has traditionally focused on pure computational and semantic properties, such as inference efficiency, or soundness and completeness with respect to different semantics. A common assumption is that all logical formulas are available *a priori*. This is analogous to assuming that all bids are truthfully known before determining the winners in an auction, or assuming that true agent preferences are available before applying a voting rule. It is somewhat surprising that, to date, no systematic investigation of incentives for truth revelation has been applied in the context of logical inference, which can be seen as a mechanism for aggregating logical formulas.

To illustrate the importance of incentives in logical reasoning, consider a trial in which multiple parties have different pieces of information relevant to the final judgment. On the one hand, the judgment must be logical, taking into account how various pieces of information support or contradict one another. On the other hand, the parties with whom this information resides often have conflicting preferences about the final outcomes: the judge wants to make the most informed judgment possible, the plaintiff wants to win the case, and the defendant wants to be acquitted. The crucial aspect here is that the choice of what information each party reveals (e.g. which witnesses to summon, which documents to present) is influenced by strategic considerations, such as one's own preferences, one's expectations of what information others have, and one's expectations of how the judge will reason with the information presented. As such, lawmakers face the challenge of designing rules of judgment that balance two requirements: (i) the requirement of deriving *logical* conclusions from the information available; (ii) the requirement of minimizing potential *strategic* manipulation.

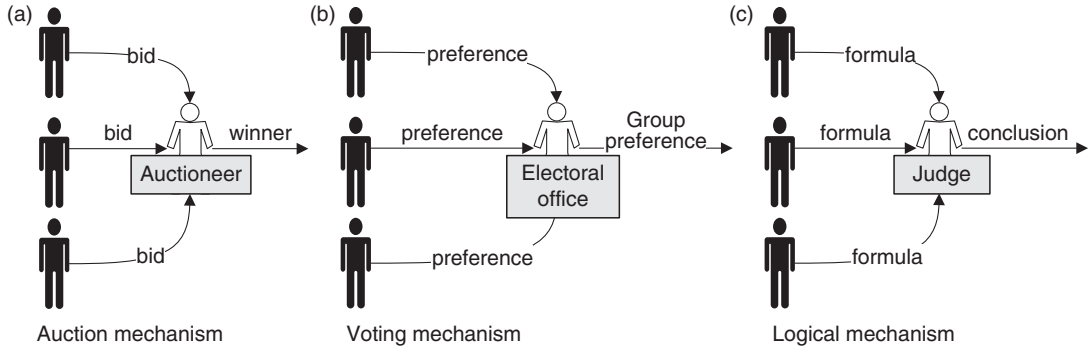

Figure 1 Logical mechanism analogous to auction or voting mechanism

The need to balance logical and strategic considerations goes beyond the courtroom. In the grand visions of the Semantic Web (Berners-Lee *et al.*, 2001) and multi-agent systems (Shoham and Leyton-Brown, 2008) communities, computerized agents will be able to exchange high-level declarative knowledge, in the form of logical statements. This high-level communication potentially enables sophisticated forms of coordination, negotiation and knowledge exchange. However, it also raises questions about the strategic behavior of the agents involved.

Against this background, we argue that the design of logical inference procedures can greatly benefit from a game-theoretic perspective, which complements the perspective of correctness in inference. Just as an auction (or a voting rule) is a rule that maps the revealed bids (or preferences) of different agents into a social outcome by allocating resources, an inference procedure can map logical formulas revealed by different agents into logical conclusions (see Figure 1). The question then becomes: *how do we engineer inference procedures that achieve soundness and completeness even when knowledge is distributed among self-interested agents?*

A general answer to the above question requires a new paradigm that marries logic and game theory in a new way. Our aim here is to motivate such a paradigm by exploring the main issues it must address. We introduce some key concepts which can be useful in its development. Finally, we point to some recent efforts.

2 Overview of mechanism design

We give a brief overview of mechanism design. See (Nisan *et al.*, 2007: Ch. 9) for more details.

Given a set of self-interested agents, denoted by I, let $\theta_i \in \Theta_i$ denote the *type* of agent i, drawn from some set of possible types Θ_i. The type represents the private information (e.g. preferences) of the agent. Agent i's preferences over *outcomes* $\mathcal{O}$ are expressed by a utility function $u_i : \mathcal{O} \times \Theta_i \rightarrow \mathbb{R}$, such that i prefers outcome o_1 to o_2 when $u_i(o_1, \theta_i) > u_i(o_2, \theta_i)$.

When agents interact, we say that they are playing *strategies*. A strategy for agent i, $s_i(\theta_i)$, is a plan that describes what actions the agent will take for every situation. We let Σ_i denote the set of all possible strategies for agent i. When it is clear from the context, we will drop the θ_i in order to simplify the notation. A *strategy profile* $s = (s_1(\theta_1), \ldots, s_I(\theta_I))$ denotes the outcome that results when each agent i is playing strategy $s_i(\theta_i)$. As a notational convenience, let: $s_{-i}(\theta_{-i}) = (s_1(\theta_i), \ldots, s_{i-1}(\theta_{i-1}), s_{i+1}(\theta_{i+1}), \ldots, s_I(\theta_I))$ and thus $s = (s_i, s_{-i})$. We then interpret $u_i((s_i, s_{-i}), \theta_i)$ to be the utility of agent i with type θ_i when all agents play strategies specified by strategy profile $(s_i(\theta_i), s_{-i}(\theta_{-i}))$.

Since the agents are all self-interested, they will try to choose strategies that maximize their own utility, taking into account the possible strategies of other agents. The *solution concepts* in game theory determine the outcomes that will arise if all agents are rational and strategic. The most well-known solution concept is the *Nash equilibrium*. A strategy profile $s^* = (s_1^*, \ldots, s_I^*)$ is a Nash equilibrium if no agent has an incentive to change their strategy, given that no other agent changes. Formally, $\forall i, \forall s_i', u_i(s_i^*, s_{-i}^*, \theta_i) \geq u_i(s_i', s_{-i}^*, \theta_i)$.

A stronger solution concept in game theory is the *dominant-strategy equilibrium*. A strategy s_i is said to be *dominant* if, by playing it, the utility of agent i is maximized no matter what strategies the other agents play formally: $\forall s_{-i}, \forall s'_i, u_i(s^*_i, s_{-i}, \theta_i) \geq u_i(s'_i, s_{-i}, \theta_i)$. A dominant-strategy equilibrium is a strategy profile where each agent is playing a dominant strategy. This is a very robust solution concept since it makes no assumptions about what information the agents have available to them, and nor does it assume that agents know that all other agents are rational.

The problem that mechanism design studies face is how to ensure a desirable system-wide outcome by designing the right kind of game. The desirable outcome is captured by a *social choice function*, which depends on the *true* agent types. A social choice function is a rule $f : \Theta_1 \times \ldots \times \Theta_I \rightarrow \mathcal{O}$, which selects some outcome $f(\theta) \in \mathcal{O}$, given the agent types $\theta = (\theta_1, \ldots, \theta_I)$.

Recall that the types of agents (the $\theta'_i s$) are private. If we ask agents to reveal their type, they may find that they are better off if they do not reveal their type truthfully, since by lying they may be able to cause the social choice function to choose an outcome that they prefer. Instead of trusting the agents to be truthful, we use a *mechanism* to try to reach the correct outcome.

A mechanism $\mathcal{M} = (\Sigma, g(\cdot))$ defines the set of allowable strategies that agents can choose, with $\Sigma = \Sigma_1 \times \ldots \times \Sigma_I$ where Σ_i is the strategy set for agent i, and an outcome function $g(s)$ that specifies an outcome o for each possible strategy profile $s = (s_1, \ldots, s_I) \in \Sigma$. This defines a game in which agent i is free to select any strategy in Σ_i, and, in particular, will try to select a strategy which will lead to an outcome that maximizes its own utility. We say that a mechanism *implements* social choice function f if the outcome induced by the mechanism is the same outcome that the social choice function would have returned if the true types of agents were known.

Formally, we say that a mechanism $\mathcal{M} = (\Sigma, g(\cdot))$ *implements* social choice function f if there exists an equilibrium s^* such that $\forall \theta \in \Theta$, $g(s^*(\theta)) = f(\theta)$. Although the definition of a mechanism puts no restrictions on the strategy spaces of the agents, an important class of mechanisms is *direct-revelation mechanisms* (or simply *direct mechanisms*), in which $\Sigma_i = \Theta_i$ for all i, and $g(\theta) = f(\theta)$ for all $\theta \in \Theta$. In other words, a direct mechanism is one where the strategies of the agents are to announce a type, θ'_i, to the mechanism. Although it is not necessary that $\theta'_i = \theta_i$, the important *Revelation Principle* states that if a social choice function, $f(\cdot)$ can be implemented, then it can be implemented by a direct mechanism where every agent reveals their true type. In such a situation, we say that the social choice function is *incentive compatible* (or *truthfully implementable*); that is, if the direct mechanism $\mathcal{M} = (\Theta, g(\cdot))$ has an equilibrium $(\theta_1, \ldots, \theta_n)$. If the equilibrium concept is the dominant-strategy equilibrium, then the social choice function is *strategy-proof*.

3 Logic as (game-theoretic) mechanism

Modern logic is the study of formal systems of inference. *Syntax* specifies what configurations of symbols are allowed in the logical language (call it $\mathcal{L}$). *Semantics* maps syntactic expressions to structures representing their meaning (e.g. possible truth value assignments in propositional logic, relational structures in predicate logic, possible world structures in modal logic, etc.). In addition to syntax and semantics, a logic also often comes with an inference calculus (also known as inference procedure, or proof theory).

Abstractly, any logic can be described in terms of a language $\mathcal{L}$ defining a set of legal formulas $2^{\mathcal{L}}$, a semantic entailment relation $\models$: $2^{\mathcal{L}} \times 2^{\mathcal{L}}$ defining logical consequence between formulas, and (if needed) an inference procedure $\vdash$ that corresponds to $\models$. For more details on this generalization of logic, the reader may refer to the discussion of 'logical systems' by Gabbay (1995).

When we design a procedure $\vdash$, we traditionally want to (at least) achieve two things. First, we want to prove that $\vdash$ is *sound* with respect to $\models$, meaning that $S \vdash \alpha$ implies $S \models \alpha$. Second, we want to prove that $\vdash$ is *complete*, meaning that $S \models \alpha$ implies $S \vdash \alpha$. To simplify the notation, we can overload the operator $\models$ (respectively $\vdash$) such that we can express $S_1 \models S_1$ (respectively $S_1 \vdash S_1$) if and only if $\forall \alpha \in S_2$ we have $S_1 \models \alpha$ (respectively $S_1 \vdash \alpha$). In this way, operators $\models$ and $\vdash$ can be seen as functions that return the *set* of all entailed (respectively inferred) formulas.

Suppose a knowledge base is distributed among a set of agents. An entailment relation $\models$ over $\mathcal{L}$ can be seen as a social choice function that we wish to implement in equilibrium. Inference procedure $\vdash$ becomes a mechanism that maps *revealed* formulas to a set of entailed formulas. These inferred formulas constitute the *outcome* of mechanism $\vdash$. In this way, we can talk of a (direct-revelation) *logic-based mechanism* as follows.

DEFINITION 1. (direct logic-based mechanism) *Given a language* $\mathcal{L}$, *inference procedure* $\vdash$, *and set of agents* I, *we can define a* direct logic-based mechanism $\vdash_{\mathcal{M}} = (\Sigma_1, \ldots, \Sigma_I, \vdash)$ *where*:

1. $\Sigma_i \subseteq 2^{\mathcal{L}}$ *are the alternative sets of formulas agent* I *can reveal* (i.e. *their strategy space*); *and*
2. $\vdash: 2^{\mathcal{L}} \rightarrow 2^{\mathcal{L}}$ *is an outcome rule making conclusions from the formulas revealed by agents.*

This enables us to propose a new criterion for an inference procedure to satisfy: that it implements the desired semantics in equilibrium (this subsumes soundness and completeness).

DEFINITION 2. (implementation of semantics) *Let* $\mathcal{L}$ *be a logical language and* $\models$ *be a semantic entailment relation over* $\mathcal{L}$, *and let* I *be a set of agents with knowledge bases* $\mathcal{K}_1, \ldots, \mathcal{K}_I$ *specified as sets of formulas in* $\mathcal{L}$. *A logical inference procedure* $\vdash$ *implements semantics* $\models$ *if using mechanism* $\vdash_{\mathcal{M}}$, $\forall \theta \in \Theta$ and *there exists an equilibrium* $s^* = (\mathcal{K}_1^\circ, \ldots, \mathcal{K}_I^\circ)$ *such that*:

$$(\mathcal{K}_1^\circ \cup \ldots \cup \mathcal{K}_I^\circ) \vdash \psi \text{ if and only if } (\mathcal{K}_1 \cup \ldots \cup \mathcal{K}_I) \models \psi$$

for any arbitrary formula $\psi \in \mathcal{L}$, *where* Θ *is the type space of all agents.*

The above definition leads to a corresponding property of the semantics itself, namely a logical version of implementable social choice functions.

DEFINITION 3. (Proof-Theoretic Implementability) *Given the language* $\mathcal{L}$, *semantics* $\models$ *is* proof-theoretically implementable *if there exists an inference procedure* $\vdash$ *whose corresponding logic-based mechanism* $\vdash_{\mathcal{M}}$ *implements* $\models$ *in equilibrium.*

To summarize, traditionally, when logicians design semantic entailment relations, they mainly focus on purely logical properties (e.g. compactness, consistency, maximality of models, intuitiveness, existence of efficient proof theories). Proof-theoretic implementability provides a new criterion for coming up with semantics in the first place, since semantics that is not implementable may not be desirable in the first place[1]. We refer to the problem of achieving proof-theoretic implementability as the *logical mechanism design* (LMD) problem.

4 Key challenges

We outline open questions that we believe are crucial to develop a comprehensive theory of LMD.

4.1 Specifying agent preferences

The first step in LMD is to identify what drives agent decision-making about what formulas to reveal, that is, their preferences as captured by a space of possible types Θ. Work on auctions often represents preferences using the notion (borrowed from economics) of a utility function that maps outcomes to real numbers. Analysis then proceeds assuming preference relations with some interesting structure – for example, diminishing marginal returns captured by utility function convexity. While such classical utility functions make sense in the context of consumer decision-making, it is not immediately clear if they are appropriate for an LMD setting.

[1] Note that one can easily define a qualified version of this condition, requiring $\models$ to be proof-theoretic implementable *for* a class of agent preferences or a restricted strategy space.

In LMD, preferences could indeed be driven by economic gains. For example, suppose agent B sues agent A for damages. B would clearly be happier if the judge concluded predicate *Compensate*(A,B, \$1000) than, say, $\neg$*Compensate*(A, B). However, in other domains, such as politics, agents may be driven by very different non-monetary incentives. An agent may simply be interested in convincing the judge (i.e. mechanism) that a particular formula must be concluded (e.g. *Innocent*(A)), regardless of how this is achieved, or whether it is actually true.

Alternatively, an agent may wish to appear knowledgeable, by ensuring that everything they says gets accepted. Such agent may withhold formulas if they believe that they may not end up being concluded by the logical mechanism. In the context of argumentation, this has been dubbed 'acceptability maximizing preferences' (Rahwan and Larson, 2008). Another familiar political example is when an agent mainly aims to make the opposition look bad, that is, in ensuring that the formulas revealed by others do not end up in the conclusion set.

In general, there is a need to systematically map the space of preference relations in the context of LMD, and to identify the various realistic structures these preferences may have. This is crucial for driving any useful subsequent analysis in LMD.

4.2 Translating classical concepts and properties

Social choice theory and mechanism theory have seen significant advancement in the last few decades, especially as they intersect with computer science (Nisan *et al*., 2007). The literature contains many well-known positive and negative results, which hold under a variety of conditions. These conditions make restrictions on the structure of agent preferences (e.g. single-peaked preferences) and the structure of the domain (e.g. feasible combinations of bids in an auction). There is an opportunity to translate many of these results to LMD.

But before one can translate existing theoretical *results*, we first need to translate the various *conditions* under which such results hold into the problem of logical mechanism design. These include both *structural* conditions as well as *desirable* properties. In terms of structural conditions, what does it mean to have *single-peaked* or *quasi-linear* preferences in a logical mechanism? Does it even make sense to talk of *budget balance* when treating logical formulas? In terms of desirable properties, what is the precise meaning of properties like *independence*, *Pareto efficiency*, *anonymity* or *systematicity*, in the context of LMD? Does *independence of irrelevant alternatives* make sense in the context of logical mechanism design, and if so, how is it defined?

4.3 Impossibility and possibility results

There is an urgent need to identify precisely which social choice functions *cannot* be implemented. Such impossibility results not only help us avoid the fruitless pursuit of mechanisms, but they also motivate the need for specialized mechanisms that work under more restrictive conditions.

One important question is whether well-known *classical* (propositional and first-order) and *non-classical* (e.g. non-monotonic or argumentation-based) semantic entailment relations are truthfully implementable by *any* mechanism.

If a given semantic entailment relation is not implementable, follow-up questions are: what restrictions are needed for it to be implementable? Can these restrictions be characterized in terms of structural properties of the underlying logical theory (e.g. the possible models/worlds of the theory)? Such restrictions may be attributed to the properties of the domain in which agents operate. A related question is whether restrictions that guarantee truth revelation can be characterized by the way in which the theory's formulas are distributed among the agents, or by the structure of the agents' preferences.

4.4 The role of computational complexity

The role of computational complexity in preventing strategic manipulation in voting has been highlighted recently (Conitzer *et al*., 2007). This important insight shows that while manipulation

(through dishonest information revelation) may sometimes be possible in theory, it may be computationally very difficult to achieve.

An important open question is to investigate whether computational hardness can also be effective in making logical mechanisms truthful. Answering this question could benefit from the wealth of computational complexity results in the underlying logical reasoning (associated with the logical inference procedure) as well as the complexity of the strategic game-theoretic reasoning imposed by the multi-agent encounter.

5 Early examples of logical mechanism design

5.1 Glazer and Rubinstein's argumentation rules

Although based in economics rather than logic, Glazer and Rubinstein were among the first to allude to the problem of LMD (Glazer and Rubinstein, 2001). In particular, they explored the mechanism design problem of constructing rules of debate that maximize the probability that a listener reaches the right conclusion given the arguments presented by two debaters.

Glazer and Rubinstein studied a very restricted setting, in which the world state is described by a vector $\omega = (w_1, \ldots, w_5)$, where each 'aspect' w_i has two possible values: 1 and 2. If $w_i = j$ for $j \in \{1, 2\}$, we say that aspect w_i supports outcome O_j. Presenting an argument amounts to revealing the value of some w_i. The setting is modeled as an extensive form game and analyzed. In particular, the authors investigate various combinations of *procedural rules* (stating in which order and what sorts of arguments each debater is allowed to state) and *persuasion rules* (stating how the outcome is chosen by the listener). In terms of procedural rules, the authors explore: (1) *one-speaker debate* in which one debater chooses two arguments to reveal; (2) *simultaneous debate* in which the two debaters simultaneously reveal one argument each; and (3) *sequential debate* in which one debater reveals one argument followed by one argument by the other. Glazer and Rubinstein investigate a variety of persuasion rules. For example, in one-speaker debate, one rule analyzed by the authors states that 'a speaker wins if and only if he presents two arguments from $\{a_1, a_2, a_3\}$ or $\{a_4, a_5\}$'. In a sequential debate, one persuasion rule states that 'if debater D_1 argues for aspect a_3, then debater D_2 wins if and only if he counter-argues with aspect a_4'.

The kinds of rules studied by Glazer and Rubinstein are clearly arbitrary, and do not follow the well-known principle of logical inference. Having said that, the work is a valuable demonstrator of how LMD could work, and a testimony to the importance and difficulty of the LMD problem.

5.2 Strategy-proofness of belief merging

A perfect early example of LMD research is work on strategic properties of *belief merging operators*. Examples include work by Everaere *et al.* (2007) and by Chopra *et al.* (2006). Here, we will focus on the former as a representative.

Essentially, a belief merging operator Δ takes as input a *belief profile* $E = \{K_1, \ldots, K_n\}$ consisting of one set of formulas K_i per agent, and a set of integrity constraints μ. Agents' beliefs may contradict one another. The operator outputs a *merged knowledge base* $\Delta_\mu(E)$, which is a set of consistent formulas that also satisfy the integrity constraints. Hence, Δ can be seen as an entailment relation $\models_\Delta$ that makes conclusions on the basis of given information. Thus, $(\bigwedge_{K_1}) \wedge \ldots \wedge (\bigwedge_{K_n}) \wedge (\bigwedge_{\mu}) \models_\Delta \psi$ denotes that formula ψ should be concluded from all information available. We can also think of an analogous proof procedure $\vdash_\Delta$. One can then ask whether an agent i has the incentive to mis-report its knowledge base K_i to cause the procedure $\vdash_\Delta$ to generate conclusions preferred by i. This is clearly an LMD problem.

Everaere *et al.* (2007) investigated the strategy-proofness of many operators from the literature on merging multiple knowledge bases expressed in propositional logic. These operators include different distance-based (e.g. based on the Hamming distance between interpretations of the propositions) and syntax-based operators (e.g. selecting the maximum number of formulas while maintaining consistency). The preferences of an agent are represented using a 'satisfaction index',

which captures the utility of a particular merging outcome. Three preference criteria are explored: the *weak drastic index* gives 1 if the result of the merging process is consistent with the agent's own base, and 0 otherwise; the *strong drastic index* gives 1 if the agent's base is a logical consequence of the result of the merging process, and 0 otherwise; and the *probabilistic index* is based on the compatibility degree between the merged base and the agent's own base measured using a ratio involving the number of models (assuming uniformly distributed outcomes).

The authors show that none of the merging operators they explore is strategy-proof in general. Then they explore various restrictions that achieve strategy-proofness: (1) restricting the number of agents; (2) restricting the number of models satisfying each agent's base; (3) adding additional integrity constraints that must be satisfied by the output; (4) restricting the strategy space (e.g. not allowing lying).

5.3 Argumentation mechanism design

Another recent example of LMD is work on argumentation mechanism design (ArgMD; Rahwan and Larson, 2008; Rahwan *et al.*, 2009). This work defined an LMD problem in the context of Dung's theory of abstract argumentation (Dung, 1995), in which a set of (potentially conflicting) formulas is seen as a set of defeasible inferences, or *arguments*. An *argumentation framework* is simply a pair $AF = \langle \mathcal{A}, \rightharpoonup \rangle$ where $\mathcal{A}$ is a set of arguments and $\rightharpoonup \subseteq \mathcal{A} \times \mathcal{A}$ is a defeat relation between arguments. This abstraction is quite powerful, since it has been shown to generalize a variety of non-monotonic logics, including extension-based approaches such as default logic and many varieties of logic programming semantics.

Various semantics have attempted to characterize 'correct' argumentation-based reasoning within an abstract argumentation framework. Given an argumentation framework $\langle \mathcal{A}, \rightharpoonup \rangle$, semantics determines whether or not the argument can be accepted. Hence, for some $\alpha \in \mathcal{A}$, we can write $\langle \mathcal{A}, \rightharpoonup \rangle \models \alpha$ to denote that argument α should be accepted. Once a desirable semantics is defined, the aim is then to find an algorithmic inference procedure $\vdash$ that corresponds to it.

Rahwan *et al.* defined the LMD problem for argumentation frameworks, given that $\mathcal{A}$ is distributed among multiple agents. They identified conditions under which the well-known *grounded* semantics can be truthfully implemented, in the sense of Definitions 2 and 3 (Rahwan and Larson, 2008; Rahwan *et al.*, 2009).

6 Logical mechanism design vs. logic games and judgment aggregation

Here, we clarify how the LMD agenda differs from related research agendas.

6.1 Early work on logic and games

The history of logic and game theory can be traced at least to the work on *game semantics*, which was pioneered by logicians such as Lorenzen (1961). Although many specific instantiations of this notion have been presented in the literature, the general idea is as follows. Given some specific logic, the truth value of a formula is determined through a special-purpose, multi-stage dialog game between two players, the *verifier* and *falsifier*. The formula is considered true precisely when the verifier has a winning strategy, while it will be false whenever the falsifier has the winning strategy. Similar ideas have been used to implement dialectical proof theories for defeasible reasoning (e.g. Prakken & Sartor, 1997).

There is a fundamental difference between the aims of game semantics and the mechanism design approach we advocate here. In game semantics, the goal is to interpret (i.e. characterize the truth value of) a specific formula by appealing to a notion of a winning strategy. As such, each player is carefully endowed with a specific set of formulas to enable the game to characterize semantics correctly (e.g. the verifier may own all the disjunctions in the formula, while the falsifier is given all the conjunctions).

In contrast, logical mechanism design is about designing the rule of inference itself to achieve correct reasoning when the theory is distributed among self-interested players who may have incentives to manipulate the outcome to their individual advantage. Game semantics has no similar notion of strategic manipulation by hiding formulas or lying.

6.2 Judgment aggregation

Recent years have seen interest in the field of *judgment aggregation* (List and Puppe, 2009): the problem of aggregating a set of individual judgments on interconnected logical sentences, to produce a collective judgment. Judgment aggregation is illustrated by the *discursive dilemma*, in which three agents have different judgments about propositions P, Q, $(P \wedge Q) \leftrightarrow R$ and R. Suppose individual judgments of the agents are as shown in the table.

	P	Q	$(P \wedge Q) \leftrightarrow R$	R
Ag_1	1	1	1	1
Ag_2	1	0	1	0
Ag_3	0	1	1	0
Majority	1	1	1	0

If we apply majority voting over each proposition, a majority accepts P, Q and $(P \wedge Q) \leftrightarrow R$, yet a majority rejects R. This example exemplifies more fundamental problems and many impossibility results (List and Puppe, 2009).

The agenda of LMD we advocate here is distinct from the agenda of judgment aggregation. In judgment aggregation, judgments are given over *given* formulas (e.g. propositional sentences (List and Puppe, 2009) or arguments (Rahwan and Tohmé, 2010)). In LMD, however, the formulas themselves are contributed by the agents. Hence, the challenge is not of extracting agents' preferences over how to evaluate existing structured logical theories, but rather the extraction of the theory itself from the agents.

7 Conclusion

We have set out a new research agenda that we dub 'LMD'. This agenda complements the existing fertile area of research at the intersection between game theory and logic. We formalized the LMD problem by defining the mechanism design problem in terms of logical semantics and inference procedures. We highlighted the emerging LMD theme in some recent work in the economics, belief revision and argumentation communities. We also outlined some key open challenges in this emerging field.

We believe that LMD will become increasingly important, particularly in the context of open knowledge-based systems, such as *strategic* knowledge sharing on the Semantic Web. LMD will play an important role in designing truthful logical mechanisms for symbolic knowledge sharing on the Web, akin to what auction mechanisms have done for electronic commerce.

References

Berners-Lee, T., Hendler, J. & Lassila, O. 2001. The semantic web. *Scientific American* 29–37.

Chopra, S., Ghose, A. & Meyer, T. 2006. Social choice theory, belief merging, and strategy-proofness. *Information Fusion* **7**, 61–79.

Conitzer, V., Sandholm, T. & Lang, J. 2007. When are elections with few candidates hard to manipulate? *Journal of the ACM* **54**(3), 14.

Dung, P. M. 1995. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artificial Intelligence* **77**(2), 321–358.

Everaere, P., Konieczny, S. & Marquis, P. 2007. The strategy-proofness landscape of merging. *Journal of Artificial Intelligence Research* **28**, 49–105.

Gabbay, D. M. 1995. What is a logical system?. In *What is a Logical System?* Gabbay, D. M. (ed.). Clarendon Press, 179–216.

Glazer, J. & Rubinstein, A. 2001. Debates and decisions: on a rationale of argumentation rules. *Games and Economic Behavior* **36**, 158–173.

List, C. & Puppe, C. 2009. Judgment aggregation: a survey. In *The Oxford Handbook of Rational and Social Choice*, Anand, P., Pattanaik, P. & Puppe, C. (eds). Oxford University Press.

Lorenzen, P. 1961. Ein dialogisches konstruktivitätskriterium. In *Infinitistic Methods*. Pergamon Press, 193–200.

Nisan, N., Roughgarden, T., Tardos, E. & Vazirani, V. V. (eds). 2007. *Algorithmic Game Theory*. Cambridge University Press.

Prakken, H. & Sartor, G. 1997. Argument-based logic programming with defeasible priorities. *Journal of Applied Non-classical Logics* **7**, 25–75.

Rahwan, I. & Larson, K. 2008. Mechanism design for abstract argumentation, In *7th International Joint Conference on Autonomous Agents & Multi Agent Systems*, *AAMAS'2008*, Padgham, L., Parkes, D., Mueller, J. & Parsons, S. (eds). Estoril, Portugal, 1031–1038.

Rahwan, I. & Tohmé, F. 2010. Collective argument evaluation as judgement aggregation. In *9th International Joint Conference on Autonomous Agents & Multi Agent Systems, AAMAS'2010*, Toronto, Canada.

Rahwan, I., Larson, K. & Tohmé, F. 2009. A characterisation of strategy-proofness for grounded argumentation semantics In *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI)*, Pasadena CA, USA, 251–256.

Shoham, Y. & Leyton-Brown, K. 2008. *Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations*. Cambridge University Press.