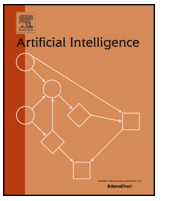
Artificial Intelligence 339 (2025) 104272

Contents lists available at ScienceDirect

Artificial Intelligence

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



Formal verification and synthesis of mechanisms for social choice

Munyque Mittelmann [a,*], Bastien Maubert [a], Aniello Murano [a], Laurent Perrussel [b]

[a] University of Naples Federico II, Italy
[b] IRIT - University Toulouse Capitole, France

ARTICLE INFO

Keywords:
Automated mechanism design
Logics for multi-agent systems
Strategic reasoning
Strategy logic

ABSTRACT

Mechanism Design (MD) aims at defining resources allocation protocols that satisfy a predefined set of properties, and Auction Mechanisms are of foremost importance. Core properties of mechanisms, such as strategy-proofness or budget balance, involve: (i) complex strategic concepts such as Nash equilibria, (ii) quantitative aspects such as utilities, and often (iii) imperfect information, with agents' private valuations. We demonstrate that Strategy Logic provides a formal framework fit to model mechanisms and express such properties, and we show that it can be used either to automatically check that a given mechanism satisfies some property (verification), or automatically produce a mechanism that does (synthesis). To do so, we consider a quantitative and variant of Strategy Logic. We first show how to express the implementation of social choice functions. Second, we show how fundamental mechanism properties can be expressed as logical formulas, and thus evaluated by model checking. We then prove that model checking for this particular variant of Strategy Logic can be done in polynomial space. Next, we show how MD can be rephrased as a synthesis problem, where mechanisms are automatically synthesized from a partial or complete logical specification. We solve the automated synthesis of mechanisms in two cases: when the number of actions is bounded, and when agents play in turns. Finally, we provide examples of auction design based for each of these two cases. The benefit of our approach in relation to classical MD is to provide a general framework for addressing a large spectrum of MD problems, which is not tailored to a particular setting or problem.

1. Introduction

Mechanism Design (MD) is a central problem in economics which consists of designing new games (a.k.a. *mechanisms*) for aggregating preferences in multi-agent settings [1]. The key challenge is to design a mechanism that chooses a *good outcome*, with respect to the designing criteria, despite the fact that agents may be self-interested and may lie about their preferences [2]. The designing criteria may specify a preferable behavior of the players (e.g. truthfulness) as well as desirable features of the outcome (e.g. social welfare maximization, budget-balance) [3]. In principle, almost any kind of market institution or economic organization can be viewed as a mechanism [4]. Some classical types of mechanisms are voting systems, auctions, and fair division protocols. Moreover,

* Corresponding author.
E-mail addresses: munyque.mittelmann@unina.it (M. Mittelmann), bastien.maubert@gmail.com (B. Maubert), aniello.murano@unina.it (A. Murano), laurent.perrussel@irit.fr (L. Perrussel).
URLs: https://www.munyque.com (M. Mittelmann), https://www.bastien-maubert.fr/ (B. Maubert), http://people.na.infn.it/~murano/ (A. Murano), https://www.irit.fr/~Laurent.Perrussel/ (L. Perrussel).

https://doi.org/10.1016/j.artint.2024.104272
Received 21 November 2022; Received in revised form 3 December 2024; Accepted 4 December 2024

Available online 10 December 2024

in recent years, there has been a growing effort at designing novel mechanisms for a wide variety of problems and settings. Examples of this include the research on school choice programs with quotas [5], matching problems with individual and regional quotas [6], peer selection to review proposals for funding [7], hedonic coalition formation games [8], sponsored search auctions [9], diffusion auctions [10], procurement auctions with budget [11], and negotiation [12].

Although logic-based languages have been widely used for verification [13] and synthesis [14] of Multi-Agent Systems (MAS), the use of formal methods for reasoning about strategic behavior in preference aggregation problems and mechanism design has not been much explored yet. An advantage in adopting such a perspective lies in the high expressivity and generality of logics for strategic reasoning [15]. Moreover, by relying on precise semantics, formal methods provide tools for rigorously analyzing the correctness of systems, which is important to improve trust in mechanisms generated by machines. The problem of formally reasoning about mechanisms is, however, nontrivial: it requires considering quantitative information (e.g., utilities and payments), private information about the participant's preferences, and complex solution concepts (such as strategy dominance and equilibria). Pauly and Wooldridge [15] first argued that strategic logics developed for the formal verification of MAS could be good candidates as formal frameworks to reason about mechanisms.

Following this idea, in this paper, we propose a logic-based approach for mechanism design. More specifically:

(i) the automated verification of mechanisms in relation to target properties expressed in a high-level logical language, and
(ii) the synthesis of mechanisms from a partial or complete logical specification.

Pauly and Wooldridge [15] considered Alternating-time Temporal Logic (ATL) [16] and showed with two case studies based on voting systems that some relevant properties for the verification of such systems can be expressed in this logic. They conclude with a research agenda in which they detail features that are missing in ATL to make it really fit for mechanism design verification:

"*We need to incorporate more game-theoretic notions in the logics we use. While the logics discussed are capable of capturing some game theoretic notions, they are still too close to their computer science origins. For example, players' preferences, strategies, equilibrium notions, are all notions which so far are inadequately represented both in the underlying semantic models and in the logical languages used. It is also still an open question whether we will eventually end up with one general-purpose logic which functions as a standard, much the way first-order logic or modal logic do in computer science.*"

In recent years much progress has been made in the field of logics for strategic reasoning, including the formulation of Strategy Logic (SL) [17,18]. Unlike ATL, which it subsumes, SL allows for explicit manipulation of strategies, and it can express important concepts in non-zero-sum games, such as Nash equilibria. Its recent quantitative extension, SL[$\mathcal{F}$] [19], introduces values in models and functions in the language, enabling the reasoning about key game-theoretic concepts such as utilities and preferences. Several works have also considered extensions of SL with imperfect information, which is also an important feature in many preference aggregation problems, such as voting and auction systems.

As pointed out by Conitzer and Sandholm [20], the downside of the classical approach for automating the process of designing mechanisms (a.k.a. Automated Mechanism Design, AMD) is that the problem needs to be solved anew each time, for new problems and settings. In this work, we propose a novel and general approach to AMD based on formal methods in the hope of building an important bridge between logics for strategic reasoning in MAS and economic theory. By enhancing the connection between different research communities, such a bridge should facilitate the transfer of results and techniques from strategic reasoning in mechanism design (for instance, the automatic verification or synthesis of games).

We argue that Strategy Logic is a good candidate for a general-purpose logic in Automated Mechanism Design. We provide examples based on auctions which, as testified in [15], "*provide a good example of mechanisms that are (a) sufficiently complex to demonstrate the usefulness of formal verification, and (b) not too complex to make formal verification infeasible*".

We first address the verification of mechanisms under imperfect information. To do so, we investigate the problem with a variant of Strategy Logic with quantitative features under imperfect information and uniform strategies. We first show how mechanisms can be cast as concurrent-game structures. We then show how SL[$\mathcal{F}$] can express that a mechanism implements a social choice function, a fundamental concept in mechanism design. This then allows us to express in SL[$\mathcal{F}$] whether a mechanism satisfies a target property. We illustrate this with several important properties often required in mechanism design, which characterize the desired behavior of the participants and the quality of the outcome: strategyproofness, individual rationality, efficiency, budget balance, and Pareto optimality [3]. Verifying that a mechanism satisfies a property then consists of model checking an SL[$\mathcal{F}$] formula. This class of strategies does not depend on the past but only on the current state. The more general *perfect-recall* strategies, which have access to the whole history of the mechanism's run, make verification undecidable in the context of imperfect information. Memoryless strategies, however, are enough for many preference aggregation scenarios.

In a second phase, we propose a novel perspective on the design of new mechanisms, by rephrasing the AMD problem in terms of optimal mechanism synthesis from SL[$\mathcal{F}$]-specifications. Such specifications may include not only game rules but also requirements regarding the strategic behavior of participants and the quality of the outcome. Here we consider the general case of perfect-recall strategies: indeed, satisfiability (and thus synthesis) is known to be undecidable already for SL, but if we consider perfect-recall strategies it is known to be decidable in two cases: when the number of actions is bounded, and when agents play in turns. The problem for memoryless strategies however remains undecidable even in these two cases. On the other hand, we drop the imperfect information aspect and focus on perfect information. Indeed it is unknown whether there are interesting cases where synthesis for SL[$\mathcal{F}$] is decidable, and SL[$\mathcal{F}$] already allows specifying in great detail a mechanism to be synthesized. To solve this synthesis problem we investigate the related satisfiability problem for SL[$\mathcal{F}$], which has not been studied so far. We show that, as for SL, it is undecidable in general, but decidable when actions are bounded or agents play in turns. We also propose an algorithm to synthesize

mechanisms that achieve the best possible satisfaction value. Since the quantitative semantics of SL[$\mathcal{F}$] reflects how well a model satisfies a formula, this effectively allows us to synthesize mechanisms that approximate properties as closely as possible even when such properties are not satisfiable, which is not possible with standard SL. We illustrate the relevance of mechanism synthesis in each of these cases with examples based on Auction Design.

We also provide complexity results for the model checking and satisfiability of SL[$\mathcal{F}$]. First, we show that model checking an SL[$\mathcal{F}$] formula can be done in PSPACE for uniform *memoryless* strategies. Second, we show that, for both models with a bounded number of actions and turned-based models, the satisfiability of SL[$\mathcal{F}$] is in time $(k+2)$-exponential, where k is the block nesting depth of the formula. Similarly to [19,21], these complexity results hold as long as the functions in $\mathcal{F}$ can be computed in the complexity class considered. Otherwise, they become the computational bottleneck.

1.1. Contribution

The main contribution of this paper is to provide a general machinery for formal reasoning about mechanisms, which can be applied to a wide range of problems and applications, including voting, fair division, peer selection, coalition formation games, and sponsored search auctions. In particular, we demonstrate how SL[$\mathcal{F}$] can express fundamental concepts and properties from MD and show that verifying a mechanism boils down to model checking a SL[$\mathcal{F}$] formula, which we prove can be done in PSPACE for memoryless strategies. The verification of mechanisms using the model-checking technique automates the process of analyzing a mechanism with respect to target properties. We then offer a novel perspective on the design of mechanisms by rephrasing the AMD problem in terms of synthesis from SL[$\mathcal{F}$] specifications. We solve the synthesis problem for SL[$\mathcal{F}$] with perfect-recall strategies by investigating the related satisfiability problem in two cases that fit many mechanisms of interest: when the number of actions is bounded, and when agents play in turns. The key advantage of synthesizing mechanisms from logical specifications is that it is a declarative approach. The designer is not required to construct a complete solution for the problem of interest; instead, she can describe the desired mechanism in terms of its rules, the existence of strategic equilibrium, as well as desirable economic properties. Furthermore, as the approach is based on formal methods, the mechanism is correct by construction, which removes the need to verify synthesized mechanisms with respect to the design criteria.

Differently from the classic AMD, we investigate the complexity of *general design* of mechanisms. That is, the solutions proposed here are not tailored to a specific scenario, but can be applied to different types of preference aggregation problems. In particular, the synthesis of mechanisms is an original approach to AMD, which allows us to compute optimal mechanisms with respect to some specifications.

Another important characteristic is that our framework allows modeling agents' knowledge via accessibility relations representing the possible worlds an agent considers possible (as for Kripke structures, see [22] for an overview). In other words, the partial observability of agents' is not captured via meta-reasoning, but it is built into the models we consider. We are able to represent that agents may not know other agents' valuations but also may have a partial observation of the mechanism itself. We also show how to enhance our framework with epistemic operators and briefly illustrate how they can be used to analyze a mechanism.

1.2. Previous version of the article

Some of the ideas and results discussed here have been already presented in a preliminary form in the conference papers [23,24]. Here, we combine and extend the results of these papers. We generalize the framework and the results from allocative mechanisms presented in [23] to the general setting of MD. We also pay special attention and expand our motivating examples, explain our approach and assumptions made in more detail, and give intuitions and illustrations. We make a proper case for the applicability of the mechanism synthesis introduced in [24] to approximate AMD and show how well-known economics impossibility results can be cast in our framework. We include new results for the model-checking and synthesis of direct mechanisms which are of first importance in Mechanism Design. We fix an error in [24], where we claimed the complexity for the satisfiability of SL[$\mathcal{F}$] was claimed to be $(k+1)$-EXPTIME instead of $(k+2)$-EXPTIME. Finally, we expanded and revised a number of proofs in the part on reasoning about mechanisms and we included the proof of satisfiability for BQCTL*, which was only sketched in [24].

1.3. Organization

We first discuss relevant related work in Section 2. In Section 3.1 we define basic notation and recall Strategy Logic. In Section 3 we define the semantics of Quantitative Strategy Logic over imperfect information. Next, in Section 5, we show how mechanisms can be represented as a weighted concurrent game structure and how classical concepts from MD are expressed as SL[$\mathcal{F}$]-formulas. Section 6 establishes the complexity of verifying mechanisms in relation to SL[$\mathcal{F}$]-properties. In Section 7 we address the satisfiability and synthesis for SL[$\mathcal{F}$]. In Section 8, we address the problem of synthesizing mechanisms using SL[$\mathcal{F}$]. In Section 9, we discuss natural extensions of the framework. Finally, Section 10 concludes the paper and provides directions for future work.

2. Related work

In this section, we present the related work on computer-aided and fully-automatic approaches for mechanism design and we recall logic-based languages for reasoning about strategic abilities in MAS.

2.1. Automated mechanism design

Mechanism Design (MD) is a field of economics that investigates how decisions are determined from individuals' information and preferences [4]. Traditionally, mechanisms have been formulated by specialists, who use their knowledge and experience to determine the game rules. Conitzer and Sandholm [25] introduced AMD, whose goal is to automatically create mechanisms for solving a specific preference aggregation problem. As pointed out by Balcan et al. [26], AMD emerges as one of the most practical and promising approaches for designing preference aggregation mechanisms and, in particular, high-revenue combinatorial auctions. Conitzer and Sandholm [20] considered AMD in problems where the designer is self-interested, that is, when there is a central authority that only cares about which outcome is chosen while incentivizing agents to participate (e.g. auctions). Sandholm et al. [27] were the first to investigate general-purpose techniques for AMD of multistage (or *iterative*) mechanisms, using greedy and dynamic programming algorithms. Balcan et al. [26] provide an analysis for the standard hierarchy of deterministic combinatorial auction classes used in AMD. Albert et al. [28] investigate the automated design of robust mechanisms, which are a class of mechanisms that provide guarantees that the game will perform at least as well as ex-post mechanisms by incorporating information on the distribution over bidders' valuations. Similarly, Zohar and Rosenschein [29] consider a class of mechanisms in which an agent attempts to elicit the private information of other participants using a payment scheme based on scoring rules. Zhang et al. [30] propose AMD for mechanisms in which each agent can misreport only a restricted set of types (i.e., preferences) with limited verification power.

AMD is usually tackled from an optimization and/or data-driven point of view. For instance, neural networks have been used to learn mechanisms that optimize a given parameter, such as revenue [31,32]. Statistical machine learning techniques have also been considered in domains without money [33]. Vorobeychik et al. [34,35] propose a black-box optimization algorithm for evaluating candidate mechanisms. Evolutionary search methods have also been used by Niu et al. [36] to optimize double auctions, while Asselin et al. [2] address AMD through linear programming and optimization. A recent work formalizes MD in the form of multiagent environments whose transition and reward functions are programs in a Domain Specific Language [37]. They propose an algorithm that combines stochastic search over programs and Bayesian optimization to maximize target properties (revenue and social welfare). By treating AMD as an engineering problem, Phelps et al. [38] focus on evolutionary and iterative approaches to mechanism design. Hajiaghayi et al. [39] develop algorithms to compute optimal, or approximately optimal, online auction mechanisms given distributional knowledge of the bid values.

Algorithmic Mechanism Design [40,41] is an MD approach to design and formally study *algorithms* for settings where the participants act according to their own self-interest, with a focus on the computational aspects. While a large number of problems have been considered in Algorithmic Mechanism Design, they are usually characterized by the multitude of input information and/or outcome possibilities. As mentioned by Nisan [42], a survey of this field is nearly impossible due to its broadness, its ill-defined borders, and its rapid pace of change. The research in the field focuses on developing and analyzing algorithms for specific application scenarios. Recent research has considered numerous applications, such as for addressing the facility location problem [43], Sponsored Search Auctions [44], obviously strategyproof mechanisms [45], and airport slot allocation [46]. Algorithmic Mechanism Design requires that the designer *craft the algorithm* and *analyse it*. Differently, as we will see, the logic-based synthesis of mechanisms relies on a declarative approach using formal methods. That is, the designer does not explicitly delineate how the algorithm aggregates preferences into social decisions. Instead, the designer will specify key rules of the intended mechanism, the existence of strategic equilibrium, and desirable economic properties. Furthermore, the foundation of formal methods ensures that the mechanism obtained by synthesis is correct by construction. This means the analysis of the solution with respect to target properties is not necessary.

AMD and formal verification techniques Some works from computer-aided verification aim at automating the analysis of mechanisms [47–50]. These works express mechanisms in high-level specification languages, which can express rich features including probabilistic aspects. The drawback of this high expressivity is that, in contrast with model checking, verification is then not fully automatic, but only assisted by a reasoner such as Isabelle or Coq. For instance, Jouvelot and Arias [49] propose a framework for the specification and verification of mechanisms based on the Coq theorem prover.

AMD and logic-based approaches Troquard et al. [51] show how to reason about voting rules properties such as strategyproofness in a formalism that allows fully automatic verification. However, the logic they use does not capture iterative protocols (such as Dutch and English auctions). Other logic-based approaches have been considered for representing and reasoning about some types of mechanisms. The Auction Description Language [52,53] and its extension with epistemic operators [54] were proposed for representing the rules governing an auction-based market. The languages are general enough to represent many kinds of mechanisms, but lack a strategic dimension and cannot be used to reason about equilibria. Okada et al. [55] use Boolean Satisfiability to model mechanisms for false-name-proof facility location. Their approach is restricted to one type of mechanism and does not handle strategic, temporal, or quantitative specifications. The works closest to ours are [15,56] which, as discussed above, advocate the use of strategic logics to reason about (possibly iterative) mechanisms, but argue that ATL is not expressive enough for this purpose.

The present paper extends our previous work [23,24]. We first proposed the use of $\mathsf{SL}[\mathcal{F}]$ as a framework for the automated verification of auction mechanisms with imperfect information [23]. Later, we rephrased the AMD problem in terms of synthesis of weighted concurrent game structures from $\mathsf{SL}[\mathcal{F}]$-specifications and studied the related satisfiability problem for the logic [24]. Our automated approach for verification of mechanisms was applied to reasoning about repeated keyword auctions with natural strategies in [57]. Since $\mathsf{SL}[\mathcal{F}]$ semantics is deterministic, the logic is unable to express probabilistic features, which are essential when considering Bayesian and randomized (or stochastic) mechanisms. Nonetheless, our work provides the basis to investigate logic-based methods apply to mechanism design. Recently, we have shown how to use Probabilistic Strategy Logic for the verification of

Bayesian and stochastic mechanisms alongside probabilistic strategies [58]. Although the probabilistic setting brings new challenges (e.g., the timeline of information disclosure), the formalization of mechanism properties is analogous to the ones proposed in this paper.

2.2. Logics for strategic reasoning

Our work is rooted in a rich body of work on logics for strategic reasoning, starting with Coalition Logic [59] and the aforementioned ATL [16], the foundational languages for strategic reasoning in MAS. Both have been extended in many directions. In [60], a first-order extension of ATL allows one to capture some quantitative aspects of relevance in mechanism design. The authors demonstrate how an English auction may be represented, and strategic properties such as manipulation and collusion verified. However, key strategic concepts such as dominance still cannot be expressed in the logic. Strategy Logic (SL) [17,18], instead, was proposed precisely to tackle this lack of expressiveness inherent to ATL-based languages. By treating strategies as first-order variables, it can express complex game-theoretic concepts.

SL was extended in several directions, including to handle imperfect information and quantitative aspects. In [61–63], SL was extended to account for imperfect information and allow reasoning about knowledge of the agents involved (see [61] for more related work on games with imperfect information, and [63] for reasoning about knowledge in strategic contexts). SL[$\mathcal{F}$] [19] was recently introduced as a quantitative extension of SL. This logic extends LTL[$\mathcal{F}$] [21], a multi-valued logic that augments the Linear Temporal Logic (LTL) with quality operators. SL[$\mathcal{F}$] subsumes both SL and LTL[$\mathcal{F}$] and is expressive enough to express complex solution concepts such as Nash equilibrium and properties about quantities.

2.3. Other related works

Concerning logics for reasoning about Social Choice, Ciná and Endriss [64] show how to formalize impossibility results (namely, Arrow's Theorem, Sen's Theorem, and the Muller–Satterthwaite Theorem) by using a modal logic and providing syntactic proofs of these theorems. Troquard et al. [65] propose a Coalition Logic that captures strategic ability and agent preferences and demonstrate that every social choice function can be characterized as a formula of this logic. At last, Ågotnes et al. [66] study a logic for reasoning about judgment aggregation scenarios, and show that it can express aggregation rules such as majority voting, rule properties such as independence, and results such as the discursive paradox, Arrow's theorem, and Condorcet's paradox. These approaches are focused on mechanisms in which agents directly report their preferences and the implementation of social choice function by indirect mechanisms and the setting of imperfect information are not addressed.

A topic related to this work is normative systems [67], which define constraints (in terms of *obligations* and *permissions*) on the behavior of agents. Bulling and Dastani [68] investigate how concepts from mechanism design can be used to analyze the enforcement of norms with preferences modeled using Linear-time Temporal Logic (LTL). Alechina et al. [69] propose an automated approach for synthesizing norms in MAS with objectives expressed in ATL. Wu et al. [70] explore the synthesis of social laws that achieve desirable economical properties via the elimination of system transitions. Their approach uses Integer-Linear Programming with constraints in Computation Tree Logic for computing an approximate mechanism. The authors highlight the need to consider a more powerful language in order to model the concurrent actions and agents with multiple private parameters, which our approach achieves to some extent.

Another related approach is rational verification, in which verification is restricted to the outcomes of the system in which agents are rational, typically in the sense of acting towards maximizing their utility (see, for instance, [71,72]). Gutierrez et al. [73] consider system specifications given in LTL, and study the implementation of a mechanism to ensure temporal properties in equilibrium. Similarly, Gutierrez et al. [74] use Reactive Modules to model check properties in equilibrium for games in which players have goals specified in LTL and CTL. Gutierrez et al. [75] propose a technique that relies on a reduction of the rational verification problem to the solution of a collection of parity games. Finally, Gutierrez et al. [76] investigate rational verification for both non-cooperative games and cooperative games in the qualitative probabilistic setting. Some recent work in this direction has focused on the problem of finding taxation schemes to incentivize the agents' behavior. This problem has been considered with combined lexicographic LTL and mean-payoff objectives [77,78], and also with LTL[$\mathcal{F}$] goals [79].

Concerning quantitative aspects in games, weighted games have been studied in the literature in relation to various kinds of objectives, including parity [80], mean-payoff [81], energy [82,83], and combining qualitative and quantitative objectives in equilibrium [84]. Other quantitative extensions of LTL and SL have been explored in the context of discounting [85,86] and averaging [87]. Quantitative extensions of ATL have also been investigated, such as timed ATL [88,89], multi-valued ATL [90], and weighted versions of ATL [91–94].

3. Preliminaries

For the remainder of the paper, we fix a finite set of atomic propositions AP, a nonempty finite set of agents Ag, and a finite set of strategy variables Var, except when stated otherwise. For each agent $a \in \mathrm{Ag}$, we assume that the set Var_a of the variables associated with a is included in Var. We will use the symbol s_a to represent the generic elements of Var_a. We let n be the number of agents in Ag. We also let $\mathcal{F} \subseteq \{f : [-1,1]^m \to [-1,1] \mid m \in \mathbb{N}\}$ be a set of functions over $[-1,1]$ of possibly different arities, that will parameterize the logics we consider.

3.1. Strategy logic

In this section, we recall Strategy Logic (SL) on which is based Quantitative SL, the logic we will consider as the specification language for mechanism design. SL [17,18] is a logic defined to express and reason about strategic aspects in multi-agent systems. It extends classic temporal logics LTL and CTL by introducing quantification on agent's strategies, and a way to assign strategies to agents and evaluate temporal properties on the behavior of the system obtained when the agents follow these strategies. This logic is very expressive, and as a result, its model-checking and satisfiability problems are often hard or even undecidable. However, the complexity comes from the possibility to freely alternate existential and universal strategy quantifiers, while in practice most properties of interest are expressed with formulas with at most one or two alternations. As we will see in this work, this is the case also when considering mechanism design. Strategy Logic has been the topic of much research, for instance on fragments with lower complexity, or extensions that incorporate aspects of multi-agent systems that were absent from its initial formulation.

Definition 1. The syntax of SL is defined by the following grammar:

$$\varphi ::= p \mid \varphi \vee \varphi \mid \neg\varphi \mid \exists s\varphi \mid (a,s)\varphi \mid \mathbf{X}\varphi \mid \varphi \mathbf{U} \varphi$$

where $p \in \mathrm{AP}$, $s \in \mathrm{Var}$, $a \in \mathrm{Ag}$,

The intuitive reading of the operators is as follows: $\exists s\varphi$ means that there exists a strategy such that φ holds; $(a,s)\varphi$ means that when strategy s is assigned to agent a, φ holds; $\mathbf{X}$ and $\mathbf{U}$ are the usual temporal operators "next" and "until".

A variable is *free* in the formula φ if it is bound to an agent without being quantified upon. An agent a is free in φ if φ contains a temporal operator ($\mathbf{X}$ or $\mathbf{U}$) not in the scope of any binding for a. The set of free variables and agents in φ is written $\mathit{free}(\varphi)$, and a formula φ is a *sentence* if $\mathit{free}(\varphi) = \emptyset$. The strategy quantifier $\exists s_a.\varphi$ quantifies on strategies *for agent* a. Except in its original formulation [17], variants of SL do not specify for which agent a strategy is at the level of strategy quantification, and this allows assigning the same strategy to different agents. However, in the imperfect information setting, we need to know concerning which observation relation a strategy should be uniform. In [61] this is done by parameterizing strategy quantifiers with observation relations. Here we adopt a slightly less general but more intuitive notation, by parameterizing directly with the agent who will use the strategy. This is enough for our purposes, because we will not need to share the same strategy between different agents, and we consider that the observation relation for each agent is fixed, as reflected by the following definition. For the sake of simplicity, in the rest of the paper, when we refer to the usual SL and its variants, we also consider this simplification.

We define the semantics of the logic based on concurrent game structures with imperfect information and explain how to capture the setting of perfect information, which was originally considered for SL. Imperfect information captures the situation in which agents have partial observability of the state. This is the case, for instance, when agents do not know each other preferences.

Definition 2. A *concurrent game structure with imperfect information* (CGSii) is a tuple $\mathcal{G} = (\{\mathrm{Ac}_a\}_{a\in\mathrm{Ag}}, V, \delta, \ell, V_\iota, \{\sim_a\}_{a\in\mathrm{Ag}})$ where

(i) Ac_a is a finite set of *actions* available for agent a;
(ii) V is a finite set of *states*;
(iii) $V_\iota \subseteq V$ is a set of *initial states*;
(iv) $\delta : V \times \prod_{a\in\mathrm{Ag}} \mathrm{Ac}_a \to V$ is a *transition function*;
(v) $\ell : V \times 2^{\mathrm{AP}}$ is a *labeling function*;
(vi) $\sim_a \subseteq V \times V$ is an equivalence relation called the *observation relation* of agent a.

As in epistemic modal logics [95], imperfect information is represented by observation relations. These relations define the states that are indistinguishable from each agent's perspective, meaning that an agent's knowledge corresponds to what is true in all states that she considers possible. Concurrent game structures with perfect information (CGS, for short) are a specific case of CGSii in which agents can distinguish every state of the game, that is, $\sim_a = \{(v,v) : v \in V\}$ for each $a \in \mathrm{Ag}$.

Remark 1. Since the class of finite mechanisms is large enough to be meaningful, we consider mechanisms that eventually output a decision. Similarly, we restrict attention to finite models.

Action profile In a state $v \in V$, each player a chooses an action $c_a \in \mathrm{Ac}_a$, and the game proceeds to state $\delta(v, \boldsymbol{c})$ where $\boldsymbol{c}$ is an *action profile* $(c_a)_{a\in\mathrm{Ag}}$.

We write $\boldsymbol{o}$ for a tuple of objects $(o_a)_{a\in\mathrm{Ag}}$, one for each agent, and such tuples are called *profiles*. Given a profile $\boldsymbol{o}$ and $a \in \mathrm{Ag}$, we let o_a be agent a's component, and $\boldsymbol{o}_{-a}$ is $(o_b)_{b\neq a}$. Similarly, we let $\mathrm{Ag}_{-a} = \mathrm{Ag} \setminus \{a\}$.

Play A *play* $\pi = v_1 v_2 \ldots$ is an infinite sequence of states such that for every $i \geq 1$ there exists an action profile $\boldsymbol{c}$ such that $\delta(v_i, \boldsymbol{c}) = v_{i+1}$. We write $\pi_i = v_i$ for the state at index i in play π. A *history* h is a finite prefix of a play, $\mathrm{last}(h)$ is the last state of history h, $|h|$ is the length of h and Hist is the set of histories. Two histories $h = v_1, \ldots, v_k$ and $h' = v'_1, \ldots, v'_k$ are *indistinguishable*, if $v_i \sim_a v'_i$ for each $1 \leq i \leq k$. With a slight abuse of notation, we write $h \sim_a h'$ if h and h' are indistinguishable.

Strategies A perfect recall *strategy* for agent a is a function $\sigma : \text{Hist} \to \text{Ac}_a$ that maps each history to an action. A perfect recall strategy σ for agent a is *uniform* if, for pair of histories h and h' such that $h \sim_a h'$, we have $\sigma(h) = \sigma(h')$. A memoryless *strategy* for agent a is a function $\sigma : V \to \text{Ac}_a$ that maps each state to an action. Uniform memoryless strategies are defined analogously to uniform perfect recall strategies. This means that agents with uniform strategies play the same action in different histories (or states) whenever they observe the same situation, that is, whenever they cannot distinguish the histories (resp. states). So considering perfect-recall strategies means that we assume that agents' actions can depend on the past, while considering memoryless ones means that their actions can only depend on the current state.

We let Str^R_a (similarly Str^r_a) be the set of perfect recall uniform strategies (resp. memoryless uniform strategies) for agent a, $Str^R = \cup_{a\in\text{Ag}} Str^R_a$, and $Str^r = \cup_{a\in\text{Ag}} Str^r_a$. Because there are finitely many states, Str^r is finite. For the remainder of the paper, we use r and R to denote memoryless and perfect recall, respectively, and we let $\rho = \{r, R\}$.

Assignment An *assignment* $\mathcal{A} : \text{Ag} \cup \text{Var} \to Str^\rho_a$ is a function from players and variables to strategies. For an assignment $\mathcal{A}$, an agent a and a strategy σ for a, $\mathcal{A}[a \mapsto \sigma]$ is the assignment that maps a to σ and is otherwise equal to $\mathcal{A}$, and $\mathcal{A}[s \mapsto \sigma]$ is defined similarly, where s is a variable.

Outcome For an assignment $\mathcal{A}$ and an history h, we let $\text{Out}(\mathcal{A}, h)$ be the unique play that continues h following the strategies assigned by $\mathcal{A}$. Formally, $\text{Out}(\mathcal{A}, h)$ is the play $hv_0v_1\ldots$ such that for all $i \geq 0$, $v_i = \delta(v_{i-1}, \boldsymbol{c})$ where for all $a \in \text{Ag}$, $\boldsymbol{c}_a = \mathcal{A}(a)(hv_0\ldots v_{i-1})$, and $v_{-1} = \text{last}(h)$.

Definition 3 *(SL semantics).* Let $\mathcal{G} = (\text{Ac}, V, \delta, \ell, v_\iota)$ be a CGS, $\mathcal{A}$ be an assignment, and $\rho \in \{R, r\}$, and φ be an SL formula. The satisfaction of φ in a history h, denoted $\mathcal{G}, \mathcal{A}, h \vDash_\rho \varphi$, is defined as follows, where π denotes $\text{Out}(h, \mathcal{A})$:

$$
\begin{array}{lll}
\mathcal{G}, \mathcal{A}, h \vDash_\rho p & \text{iff} & p \in \ell(\text{last}(h)) \\
\mathcal{G}, \mathcal{A}, h \vDash_\rho \varphi \vee \psi & \text{iff} & \mathcal{G}, \mathcal{A}, h \vDash_\rho \varphi \text{ or } \mathcal{G}, \mathcal{A}, h \vDash_\rho \psi \\
\mathcal{G}, \mathcal{A}, h \vDash_\rho \neg\varphi & \text{iff} & \mathcal{G}, \mathcal{A}, h \nvDash_\rho \varphi \\
\mathcal{G}, \mathcal{A}, h \vDash_\rho \exists s\varphi & \text{iff} & \exists \sigma \in Str \text{ s.t. } \mathcal{G}, \mathcal{A}[s \mapsto \sigma], h \vDash_\rho \varphi \\
\mathcal{G}, \mathcal{A}, h \vDash_\rho (a, s)\varphi & \text{iff} & \mathcal{G}, \mathcal{A}[a \mapsto \mathcal{A}(s)], h \vDash_\rho \varphi \\
\mathcal{G}, \mathcal{A}, h \vDash_\rho \mathbf{X}\varphi & \text{iff} & \mathcal{G}, \mathcal{A}, \pi_1 \vDash_\rho \varphi \\
\mathcal{G}, \mathcal{A}, h \vDash_\rho \varphi_1 \mathbf{U} \varphi_2 & \text{iff} & \exists k \geq i \text{ such that } \mathcal{G}, \mathcal{A}, \pi_k \vDash_\rho \psi_2 \\
 & & \text{and } \forall j \in [i, k).\ \mathcal{G}, \mathcal{A}, \pi_j \vDash_\rho \psi_1
\end{array}
$$

If φ is a sentence, its satisfaction value does not depend on the assignment, and we write $\mathcal{G}, h \vDash_\rho \varphi$ for $\mathcal{G}, \mathcal{A}, h \vDash_\rho \varphi$ where $\mathcal{A}$ is any assignment. We also let $\mathcal{G} \vDash_\rho \varphi$ iff $\mathcal{G}, v_\iota \vDash_\rho \varphi$.

We can define the following classic abbreviations: $\top := p \vee \neg p$ (for an arbitrary atomic proposition $p \in \text{AP}$), $\bot := \neg\top$, $\varphi \wedge \varphi' := \neg(\neg\varphi \vee \neg\varphi')$, $\varphi \to \varphi' := \neg\varphi \vee \varphi'$, $\mathbf{F}\psi := \top\mathbf{U}\psi$, $\mathbf{G}\psi := \neg\mathbf{F}\neg\psi$, and $\forall s.\, \varphi := \neg\exists s.\, \neg\varphi$.

4. Quantitative strategy logic

Let us now present the logical framework considered in this paper, the Quantitative Strategy Logic [19,96]. This logic is a quantitative extension of SL introduced in Bouyer et al. [19], which allows us to treat naturally the quantitative aspects inherent to many mechanism design problems. In Bouyer et al. [19], SL[$\mathcal{F}$] semantics is defined over models with perfect information. To also allow the modeling of problems where agents have partial observability of the system (e.g., agents may ignore other agents' preferences), we here define its semantics over models with imperfect information and uniform strategies. Action profiles, plays, strategies, assignments, and outcomes are defined analogously to SL (see Section 3.1).

One notable difference is that while the original formulation considers all values to be in [0,1], we slightly generalize the setting to allow for negative values in [-1,0] as well. This allows us to naturally capture the situation in which agents have negative utilities. This is the case, for instance, when agents buy an item at a higher price than what they believe it is worth, or when considering the allocation of chores [97]. Another difference with [19] is that the logic we present here does not have the outcome quantifier. This however does not restrict expressivity, as this outcome quantifier is essentially syntactic sugar that makes it more convenient to consider situations where only some of the agents have been assigned a strategy [98].

Definition 4. Let $\mathcal{F} \subseteq \{f : [-1, 1]^m \to [-1, 1] \mid m \in \mathbb{N}\}$ be a set of functions over $[-1, 1]$ of possibly different arities. The syntax of SL[$\mathcal{F}$] is defined by the following grammar:

$$\varphi ::= p \mid \exists s_a.\, \varphi \mid (a, s_a)\varphi \mid f(\varphi, \ldots, \varphi) \mid \mathbf{X}\varphi \mid \varphi\mathbf{U}\varphi$$

where $p \in \text{AP}$, $s_a \in \text{Var}_a$, $a \in \text{Ag}$, and $f \in \mathcal{F}$.

The intuitive reading of the operators $\exists s_a.\varphi$, $(a,s)\varphi$, $\mathbf{X}$, and $\mathbf{U}$ is the same as for SL. The meaning of $f(\varphi_1,...,\varphi_n)$ depends on the function f. We use $\top$, $\vee$, and $\neg$ to denote, respectively, function 1, function $x,y \mapsto \max(x,y)$ and function $x \mapsto -x$.

For a set of agents $A=\{a_1,...,a_n\}$ and variable profile $S=(s_{a_1},...,s_{a_n})$, we write $(A,S)\varphi$ for $(a_1,s_{a_1})...(a_n,s_{a_n})\varphi$. Similarly, we write $\forall S\varphi$ for $\forall s_{a_1}...\forall s_{a_n}\varphi$.

Remark 2. In [19], values are meant to represent degrees of truth value in $[0,1]$ where 0 corresponds to "false" and 1 corresponds to "true", as in Fuzzy Logics. In this setting, negation $\neg$ is the function $x \mapsto 1-x$. Here we consider instead values in $[-1,1]$. This does not affect the semantics of the logic, nor the model-checking problem, and it allows us to consider negative quantities. For instance, a positive value may denote that an agent is receiving something, while a negative value represents that she is giving something. The main difference is that "false" now corresponds to -1, and negation is the function $x \mapsto -x$.

Differently from SL, which has Boolean semantics, the semantics of SL[$\mathcal{F}$] are based on Weighted Concurrent Game Structures (wCGS). In a wCGS, atomic propositions describe features of the game and are assigned a weight in each state.

Definition 5. A *weighted concurrent game structure with imperfect information* (wCGSii) is a tuple $\mathcal{G}=(\{\mathrm{Ac}_a\}_{a\in\mathrm{Ag}}, V, \delta, \ell, V_\iota, \{\sim_a\}_{a\in\mathrm{Ag}})$ where

(i) Ac_a is a finite set of *actions* available for agent a;
(ii) V is a finite set of *states*;
(iii) $V_\iota \subseteq V$ is a set of *initial states*;
(iv) $\delta : V \times \prod_{a\in\mathrm{Ag}} \mathrm{Ac}_a \to V$ is a *transition function*;
(v) $\ell : V \times \mathrm{AP} \to [-1,1]$ is a *weight function*;
(vi) $\sim_a \subseteq V \times V$ is an equivalence relation called the *observation relation* of agent a.

Similar to $\mathcal{G}$, weighted concurrent game structures with perfect information (wCGS, for short) are a specific case of wCGSii in which agents can distinguish every state of the game, that is, $\sim_a = \{(v,v) : v \in V\}$ for each $a \in \mathrm{Ag}$. For simplicity, we omit the relations $\{\sim_a\}_{a\in\mathrm{Ag}}$ for wCGS.

Definition 6 *(SL[$\mathcal{F}$] semantics).* Let $\mathcal{G}=(\{\mathrm{Ac}_a\}_{a\in\mathrm{Ag}}, V, \delta, \ell, V_\iota, \{\sim_a\}_{a\in\mathrm{Ag}})$ be a wCGSii, $\mathcal{A}$ an assignment, and $\rho \in \{R, r\}$. The satisfaction value $[\![\varphi]\!]^{\mathcal{G},\rho}_{\mathcal{A}}(h) \in [-1,1]$ of an SL[$\mathcal{F}$] formula φ in a history h is defined as follows, where π denotes $\mathrm{Out}(\mathcal{A},h)$:

$$
\begin{aligned}
[\![p]\!]^{\mathcal{G},\rho}_{\mathcal{A}}(h) &= \ell(\mathrm{last}(h),p)\\
[\![\exists s_a.\varphi]\!]^{\mathcal{G},\rho}_{\mathcal{A}}(h) &= \max_{\sigma\in Str^{\rho}_{a}} [\![\varphi]\!]^{\mathcal{G},\rho}_{\mathcal{A}[s\mapsto\sigma]}(h)\\
[\![(a,s_a)\varphi]\!]^{\mathcal{G},\rho}_{\mathcal{A}}(h) &= [\![\varphi]\!]^{\mathcal{G},\rho}_{\mathcal{A}[a\mapsto\mathcal{A}(s)]}(h)\\
[\![f(\varphi_1,...,\varphi_m)]\!]^{\mathcal{G},\rho}_{\mathcal{A}}(h) &= f([\![\varphi_1]\!]^{\mathcal{G},\rho}_{\mathcal{A}}(h),...,[\![\varphi_m]\!]^{\mathcal{G},\rho}_{\mathcal{A}}(h))\\
[\![\mathbf{X}\varphi]\!]^{\mathcal{G},\rho}_{\mathcal{A}}(h) &= [\![\varphi]\!]^{\mathcal{G},\rho}_{\mathcal{A}}(\pi_{|h|+1})\\
[\![\varphi_1\mathbf{U}\varphi_2]\!]^{\mathcal{G},\rho}_{\mathcal{A}}(h) &= \sup_{i\geq 0} \min\big([\![\varphi_2]\!]^{\mathcal{G},\rho}_{\mathcal{A}}(\pi_{|h|+i}), \min_{0\leq j<i} [\![\varphi_1]\!]^{\mathcal{G},\rho}_{\mathcal{A}}(\pi_{|h|+j})\big)
\end{aligned}
$$

If φ is a sentence, its satisfaction value does not depend on the assignment, and we write $[\![\varphi]\!]^{\mathcal{G},\rho}(h)$ for $[\![\varphi]\!]^{\mathcal{G},\rho}_{\mathcal{A}}(h)$ where $\mathcal{A}$ is any assignment. We also let $[\![\varphi]\!]^{\mathcal{G},\rho} = \max_{v\in V_\iota}([\![\varphi]\!]^{\mathcal{G},\rho}(v))$.

The strategy quantification $\exists s_a.\varphi$ computes the maximal value a choice of strategy for variable s_a can give to formula φ. The binding $(a,s_a)\varphi$ just assigns strategy given by s_a to agent a. The temporal modality "next" $\mathbf{X}\varphi$ takes the value of φ at the next step of a given outcome, while the "until" $\varphi_1\mathbf{U}\varphi_2$ maximizes, over all states along the play, the minimum between the value of φ_2 at that state and the minimal value of φ_1 before this state.

We define the following abbreviations: $\top := 1$, $\bot := \neg\top$, $\varphi \wedge \varphi' := \neg(\neg\varphi \vee \neg\varphi')$, $\varphi \to \varphi' := \neg\varphi \vee \varphi'$, $\mathbf{F}\varphi := \top\mathbf{U}\varphi$, $\mathbf{G}\varphi := \neg\mathbf{F}\neg\varphi$ and $\forall s_a.\varphi := \neg\exists s_a.\neg\varphi$. We also use $\mathbf{A}\varphi$ as a shorthand for a universal quantification on strategies and bindings for all agents, followed by φ; this simulates the universal path quantifier of CTL*.

Remark 3. In the next sections, we often make use of the abbreviation $\mathbf{F}\varphi$. In Boolean temporal logics, it represents that φ holds at some point in the future. Notice that, given the preamble of Definition 6, the satisfaction value of $[\![\mathbf{F}\varphi]\!]^{\mathcal{G},\rho}_{\mathcal{A}}(h) = \sup_{i\geq 0}([\![\varphi]\!]^{\mathcal{G},\rho}_{\mathcal{A}}(\pi_{|h|+i}))$, that is, the highest satisfaction value of φ in the path.

Remark 4. In the particular case where atomic propositions only take values in $\{-1,1\}$ and $\mathcal{F}$ consists of the functions $x \mapsto -x$ (negation) and $x,y \mapsto \max(x,y)$ (disjunction), SL[$\mathcal{F}$] corresponds to the usual Boolean-valued SL.

5. Strategic reasoning in mechanism design

We now show how SL[$\mathcal{F}$] can be used to express all important concepts and properties from mechanism design. We first recall the classical definitions from Economic Theory. We proceed by showing how to express these notions using SL[$\mathcal{F}$] and wCGSii, and then prove the correctness of such encoding.

5.1. Social choice functions

We first recall social choice functions, used to formalize how to choose one outcome among several alternatives, based on individual preferences of the agents. Let *Alt* be a finite set of *alternatives*. Since many preference aggregation problems involve monetary transfers, it is useful to characterize the alternatives in terms of both a collective choice and the agents' payments. Thus, without loss of generality, we assume that each alternative in *Alt* is of the form $\alpha = (x, \boldsymbol{p})$ where $x \in \mathcal{X}$ is a *social choice* (the non-monetary part of the alternative) from a finite set of possible choices $\mathcal{X} \subset [-1,1]^{\mathsf{m}}$ (for some $\mathsf{m} \geq 1$), and $p_a \in [-1,1]$ is the payment for agent a. The value m represents the number of *issues* for which the agents need to make decisions (aside from payments) and we assume $\mathsf{m} \geq 1$, i.e., there is at least one issue in $\mathcal{X}$. Issues represent aspects of the decision (such as the winner, or the allocation of a specific agent).

For each agent $a \in \mathrm{Ag}$, let also $\Theta_a \subset [-1,1]$ be a finite set of possible *types* for a. We let $\boldsymbol{\Theta} = \prod_{a \in \mathrm{Ag}} \Theta_a$, and we note $\boldsymbol{\theta} = (\theta_a)_{a \in \mathrm{Ag}} \in \boldsymbol{\Theta}$ for a type profile, which assigns a type θ_a to each agent a. The type θ_a of an agent a represents her preference and determines how she values each social choice $x \in \mathcal{X}$; this is represented by a *valuation function* $\mathsf{v}_a : \mathcal{X} \times \Theta_a \to [-1,1]$.

Example 1. For instance, in a one-sided auction with one item, the collective choice has a single issue: decide who wins the item. This can be modeled by letting $\mathcal{X} = \{\mathit{wins}_a : a \in \mathrm{Ag}\} \cup \{-1\}$, where $\mathit{wins}_a \in (-1,1]$ is a constant value denoting the social choice in which a wins and -1 specifies the case where there is no winner at the end of the auction. The type θ_a of agent a reflects how much the agent desires the item. One could define the valuation of agent a as $\mathsf{v}_a(\mathit{wins}_a, \theta_a) = \theta_a$ and $\mathsf{v}_a(\mathit{wins}_b, \theta_a) = 0$ for $b \neq a$. With this definition, a type θ_a close to 1 models an agent very interested in the item, who will have a high valuation if she receives it, the item, and a low one if she does not. A type close to -1 represents an agent who strongly does not want this item and who has a low valuation if she does. This could be the case, for instance, when considering items that represent "chores", such as task assignments. Finally, type 0 represents an indifferent agent, who receives valuation 0 in all possible choices. Similarly, an agent with such a valuation function is indifferent when she is not the winner

In a two-sided auction (i.e., with multiple buyers and sellers) with $n \geq 1$ copies of a good, a social choice describes how many items each agent sells or buys. This auction has n issues: for each agent, decide the number of goods she will sell or buy. As usual in double-sided markets (see, for instance, [99]), we use negative numbers to represent that the agent is selling that amount of items, and positives represent that they are buying. Thus, the possible trade for an agent is defined over $[-n, n]$.

Since our framework considers values between $[-1,1]$, we start by normalizing the possible trades. To rescale a number x of items being traded (with $-n \leq x \leq n$), we let $\mathit{norm}_x = -1 + \frac{(x+n)}{n}$ denote its normalized value to the interval $[-1,1]$. For instance, if the total number n of goods is 10, then $\mathit{norm}_4 = 0.4$ and $\mathit{norm}_{-4} = -0.4$. We then let $\mathcal{X} = \{(\mathit{norm}_{x_a})_{a \in a} : -n \leq x_a \leq n, \sum_{a \in \mathrm{Ag}} x_a g = 0, a \in \mathrm{Ag}\}$, where norm_{x_a} denotes the normalized value that agent a is buying (if $x_a \geq 0$) or selling (if $x_a \leq 0$). The constraint $\sum_{a \in \mathrm{Ag}} x_a = 0$ imposes a strict balance in the supply and demand of goods. Assuming two agents (and thus, two issues), a choice in $\mathcal{X}$ could be $(-0.4, 0.4)$, representing that the first agent has sold 0.4 copies of the item, while the second agent bought them.

The valuation of agent a in this setting could be defined as $\mathsf{v}_a((\mathit{norm}_{x_b})_{b \in \mathrm{Ag}}, \theta_a) = (\mathit{norm}_{x_a} \cdot \theta_a)$ if a is a buyer and $\mathsf{v}_a((\mathit{norm}_{x_b})_{b \in \mathrm{Ag}}, \theta_a) = -(\mathit{norm}_{x_a} \cdot \theta_a)$ if a is a seller. With these valuations, the agents care only about the items that they buy or sell, and not the trades of others.

The (quasi-linear) *utility* of agent a with type θ_a for an alternative $\alpha = (x, \boldsymbol{p})$ is defined as

$$u_a(\alpha, \theta_a) = \mathsf{v}_a(x, \theta_a) - p_a$$

That is, the utility for agent a is the difference between how much she evaluates the social choice x and her payment p_a.

Definition 7. A *social choice function* (SCF) $\mathsf{f} : \boldsymbol{\Theta} \to \mathit{Alt}$ is a function that, given a type profile $\boldsymbol{\theta}$, chooses an alternative $\mathsf{f}(\boldsymbol{\theta}) \in \mathit{Alt}$. We can split a social choice function as follows: $\mathsf{f} = (\mathsf{x}, \{\mathsf{p}_a\})$, where $\mathsf{x} : \boldsymbol{\Theta} \to \mathcal{X}$ is a *choice function* and for each a, $\mathsf{p}_a : \boldsymbol{\Theta} \to [-1,1]$ is a *payment function* for agent a.

Example 2. Consider the choice set for one-sided auctions introduced in Example 1 and a predefined and fixed total order, denoted $\prec_{\mathrm{Ag}}$, between the agents. The first-price social choice function $\mathsf{f}_{\mathrm{fp}} = (\mathsf{x}, \{\mathsf{p}_a\})$ is defined as follows. The choice function is $\mathsf{x}(\boldsymbol{\theta}) = x$, where $x = \mathit{choice}_a$ if θ_a is the highest type in $\boldsymbol{\theta}$. In case two agents $a \neq b$ have the highest type, $x = \mathit{choice}_a$ iff $a \prec_{\mathrm{Ag}} b$, that is, ties are broken according to $\prec_{\mathrm{Ag}}$.[1] The payment function for agent a is defined as $\mathsf{p}_a(\boldsymbol{\theta}) = \theta_a$ if $x = \mathit{choice}_a$ and $\mathsf{p}_a(\boldsymbol{\theta}) = 0$ otherwise.

In the next sections, we describe how to represent mechanisms as wCGS and how to determinate whether a wCGS implements an SCF, in the sense that they assign the same alternative in strategic equilibrium.

[1] This is an usual approach for tie-breaking in computational social choice [100].

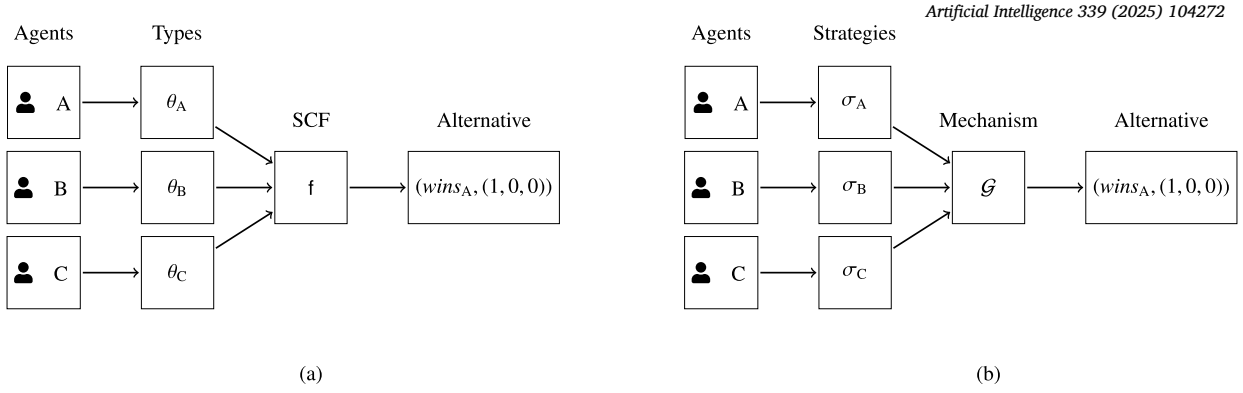


Fig. 1. (a) A Social Choice Function for aggregating the preferences of agents A, B and C. The agents' types are used for choosing an alternative. In this case, the alternative chosen selects A as the winner and she pays the price 1. (a) A mechanism for aggregating the preferences of agents A, B and C. Agents' play strategies, which may involve reporting their types.

5.2. Mechanisms as wCGSii

While social choice functions assign an alternative based on agents' preferences (types), a mechanism describes agents' strategies and their outcome, which is also in terms of a choice of alternative (see Fig. 1). Some mechanisms are "one-shot", meaning that the final alternative is reached after each agent has chosen one action, while others are iterative and may contain multiple stages. A one-shot mechanism is called *direct* when the actions available consist of reporting types. Also, the setting considered may involve agents holding some private information (for instance, in relation to their private preferences). Weighted concurrent game structures can very naturally model complex one-shot or multi-stage mechanisms with imperfect information, and we provide a general definition of mechanisms as a class of concurrent game structures with special atomic propositions to represent types, social choices, payments, etc.

Definition 8 *(Mechanism as wCGSii)*. Let AP $\supseteq \{choice_k, payment_a, terminal, type_a : a \in \mathrm{Ag}, k \leq \mathsf{m}\}$ be a set of atomic propositions, where $choice_k$, $type_a$, $payment_a$ denote, respectively, the issue k (i.e., a part of the social choice) at the current state, the type of a, and her payment. The proposition *terminal* specifies *terminal* states. We write $choice := choice_1, ..., choice_{\mathsf{m}}$ for the list combining the propositions referring to issues being decided by the agents. A *mechanism* is a wCGSii over the atomic propositions AP that satisfies the following:

(i) there is one initial state $v_\iota^{\boldsymbol{\theta}}$ for each possible type profile $\boldsymbol{\theta} \in \boldsymbol{\Theta}$;
(ii) types remain unchanged through transitions, i.e. if $\delta(v, \boldsymbol{c}) = v'$ then $\ell(v, type_a) = \ell(v', type_a)$ for each a;
(iii) each agent *knows* her own type: if $v \sim_a v'$, then $\ell(v, type_a) = \ell(v', type_a)$, that is, in each state she considers possible, her type is the same;
(iv) every play eventually reaches a *terminal state*, i.e., a sink[2] where Proposition *terminal* has value 1;
(v) in all non-terminal states, *terminal* has value -1.

A type profile $\boldsymbol{\theta}$ together with a strategy profile $\boldsymbol{\sigma}$ determines a unique terminal state $v(\boldsymbol{\theta}, \boldsymbol{\sigma})$, which is the terminal state reached from $v_\iota^{\boldsymbol{\theta}}$ via $\boldsymbol{\sigma}$. The values of propositions $choice_1, ..., choice_{\mathsf{m}}$ and $payment_a$ in terminal state $v(\boldsymbol{\theta}, \boldsymbol{\sigma})$ encode an alternative that we write $\mathcal{G}[\boldsymbol{\theta}, \boldsymbol{\sigma}]$.

Notice that each issue k in the mechanisms is captured by an atomic proposition $choice_k$ and $choice_1, ..., choice_{\mathsf{m}}$ represents a choice in $\mathcal{X}$. Going back to the two-sided auction in Example 1, $\ell(v, choice_1) = -0.4$ and $\ell(v, choice_2) = 0.4$ represent the choice $(-0.4, 0.4)$ in the state v.

We now illustrate with an example how this formal definition of mechanisms captures complex iterative mechanisms with quantitative aspects and imperfect information.

Example 3 *(Dutch auction)*. A Dutch auction is an iterative protocol with decreasing price; hereafter we assume the single good and single unit case. Initially, the auctioneer proposes a high asking price. This price is gradually lowered until some bidder accepts to purchase the good. The auction then ends and the object is sold to this bidder at the given price [101]. In the case of a draw, the winner is determined with respect to an arbitrary order $\prec_{\mathrm{Ag}}$ among the agents.

[2] A sink is a state that loops for all action profiles.

We consider the (single issue) choice set $\mathcal{X} = \{wins_a : a \in \mathrm{Ag}\} \cup \{-1\}$ as introduced in Example 1. Let us fix a price decrement $\mathsf{dec} \in (0, 1]$ and, for each agent $a \in \mathrm{Ag}$, (i) a finite set of possible types $\Theta_a \subset [0, 1]$, and (ii) her real type $\theta_a \in \Theta_a$. Agent a's valuation is $\mathsf{v}_a(wins_a, \theta_a) = \theta_a$, and $\mathsf{v}_a(wins_b, \theta_a) = 0$ for $b \neq a$.

Define the mechanism $\mathcal{G}_{dut} = (\{\mathrm{Ac}_a\}, V, \delta, \ell, V_\iota, \{\sim_a\})$ over $\mathrm{AP} = \{price, choice, payment_a, terminal, type_a : a \in \mathrm{Ag}\}$, where:

- $\mathrm{Ac}_a = \{bid, wait\}$ for each $a \in \mathrm{Ag}$,
- V consists of states of the form $\langle p, x, tr, \{\theta_a\}\rangle$ with $p \in \{1 - x \cdot \mathsf{dec} : 0 \leq x \leq \frac{1}{\mathsf{dec}}\}$ denoting the current price, $tr \in \{-1, 1\}$ denoting whether the state is terminal, $x \in \mathcal{X}$ specifying the winner, and $\theta_a \in \Theta_a$ specifying a's type.

In an initial state, the price starts at 1 and there is no winner. That is, the set of initial states is $V_\iota = \{\langle 1, -1, -1, \theta_1, ..., \theta_n\rangle \in V\}$. In non-terminal states, the transition function keeps decreasing the price p as long as it is above zero and every agent performs the action wait. If an agent a bids, the good is assigned to her ($x = 1$). Since there is only one unit of the good, ties are decided according to the order $\prec_{\mathrm{Ag}}$. If the price remains unchanged in the transition, the state is marked as terminal ($tr = 1$). The transition function defines a loop for terminal states, ensuring no change occurs in the auction afterward (see Fig. 2 for a partial illustration). Formally, for each state $v = \langle p, x, tr, \{\theta_a\}\rangle$ and joint action $\boldsymbol{c} = (c_a)_{a \in \mathrm{Ag}}$, transition $\delta(v, \boldsymbol{c})$ is defined as follows:

- If $tr = -1$, $\delta(v, \boldsymbol{c}) = \langle p', x', tr', \{\theta_a\}\rangle$ where:

$$p' = \begin{cases} p - \mathsf{dec} & \text{if } p - \mathsf{dec} \geq 0 \text{ and} \\ & c_a = wait \text{ for all } a \in \mathrm{Ag} \\ p & \text{otherwise} \end{cases}$$

$$x' = \begin{cases} wins_a & \text{if } c_a = bid \text{ for } a \in \mathrm{Ag}, \text{ such that for all } b \neq a \\ & \text{either } c_b = wait \text{ or } a \prec_{\mathrm{Ag}} b \\ -1 & \text{otherwise} \end{cases}$$

$$tr' = \begin{cases} 1 & \text{if } p' = p, \\ -1 & \text{otherwise} \end{cases}$$

- Otherwise, $\delta(v, \boldsymbol{c}) = v$.

For each $v = \langle p, x, tr, \{\theta_a\}\rangle$ and each $a \in \mathrm{Ag}$, the weight function is defined as follows: $\ell(v, price) = p$, $\ell(v, choice) = x$, $\ell(v, payment_a) = p$ if $x = wins_a$ and $\ell(v, payment_a) = 0$ otherwise, $\ell(v, terminal) = tr$, and $\ell(v, type_a) = \theta_a$.

Finally, for each agent $a \in \mathrm{Ag}$ and for any two states $v = \langle p, x, tr, \{\theta_a\}\rangle$ and $v' = \langle p', x', tr', \{\theta'_a\}_{a \in \mathrm{Ag}}\rangle$ in V, the observation relation $\sim_a$ is defined as follows: $v \sim_a v'$ if (i) $p = p'$; (ii) $x = x'$; (iii) $\theta_a = \theta'_a$; and (iv) $tr = tr'$.

Observation relations $\sim_a$ capture the fact that agents do not know other agents' preferences, and thus their actions cannot depend on them. This is reflected in $\mathsf{SL}[\mathcal{F}]$ by the notion of uniform strategy. It is out of the scope of this work to consider probabilistic beliefs about agents' preferences.

Definition 9 *(Direct mechanism as wCGSii).* A mechanism $\mathcal{G}$ is direct if

1. $\mathrm{Ac}_a = \Theta_a$ for each $a \in \mathrm{Ag}$;
2. For each $v_\iota \in V_\iota$ and $\boldsymbol{c} \in \prod_{a \in \mathrm{Ag}} \mathrm{Ac}_a$, $\ell(\delta(v_\iota, \boldsymbol{c}), terminal) = 1$.

Example 4. Notice that Example 2 can be encoded as a wCGSii representing a direct mechanism $\mathcal{G}_{fp} = (\{\Theta_a\}, V, \delta, \ell, V_\iota, \{\sim_a\})$, where: V consists of states of the form $\langle p, x, tr, \{\theta_a\}\rangle$ with $p \in \bigcup_{a \in \mathrm{Ag}} \Theta_a$ denoting the current price, $tr \in \{-1, 1\}$ denoting whether the state is a decision state, $x \in \mathcal{X}$ specifying the winner, and $\theta_a \in \Theta_a$ specifying a's type. The transition $\delta(v, \boldsymbol{c})$ is defined as follows, for $\boldsymbol{c} \in \boldsymbol{\Theta}$ and $v = \langle p, x, tr, \{\theta_a\}\rangle$:

- If $tr = -1$, $\delta(v, \boldsymbol{c}) = \langle p', x', tr', \{\theta_a\}\rangle$ where: (i) $p' = \max_{a \in \mathrm{Ag}}(\theta_a)$, (ii) $x' = choice_a$ such that $\theta_a = \max_{a' \in \mathrm{Ag}}(\theta_{a'})$ for some $a \in \mathrm{Ag}$, and for all $b \neq a$ either $\theta_b < \theta_a$ or $a \prec_{\mathrm{Ag}} b$, and (iii) $tr' = 1$
- Otherwise, $\delta(v, \boldsymbol{c}) = v$.

The other components of $\mathcal{G}_{fp}$ are defined analogously to Example 3.

Functions in $\mathcal{F}$ In this paper, we assume that $\mathcal{F}$ contains the function

$$- : (x, y) \mapsto \min(1, \max(-1, x - y))$$

as well as the valuation function v_a for each agent a, and for readability, we use the infix notation $x - y$ in the formula. We also assume that $\mathcal{F}$ contains the comparison function

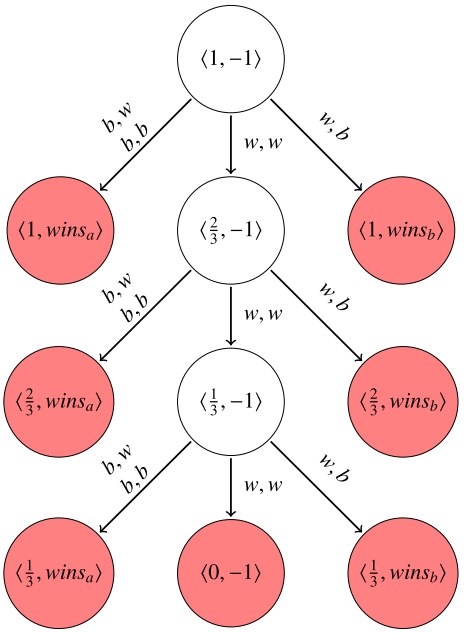


Fig. 2. Part of the mechanism for the Dutch auction with two agents (namely a and b) and decrement $\mathsf{dec} = \frac{1}{3}$. We assume $a \prec_{\mathrm{Ag}} b$, that is a wins ties. Terminal states are in red. We only represent one initial state and thus we omit types, which are the same in all states. Action *bid* is written b and *wait* is w. Finally, we did not represent loops. (For interpretation of the colors in the figure, the reader is referred to the web version of this article.)

$$\leq : (x, y) \mapsto \begin{cases} 1 & \text{if } x \leq y, \\ -1 & \text{otherwise,} \end{cases}$$

the comparison function $<$ (defined similarly, with $<$ instead of $\leq$), the equality function

$$= : (x, y) \mapsto \begin{cases} 1 & \text{if } x = y, \\ -1 & \text{otherwise,} \end{cases}$$

and the n-ary sum function

$$\sum : x_1, ..., x_{\mathsf{n}} \mapsto \min(1, \max(-1, \sum_k x_k))$$

Finally we assume that types, allocations, payments, and valuations are normalized so that all values remain in $[-1, 1]$.

Utility in $\mathsf{SL}[\mathcal{F}]$ Fix a mechanism $\mathcal{G}$. The goal of an agent is to maximize her utility, which is equal to the value of the $\mathsf{SL}[\mathcal{F}]$ formula

$$util_a := \mathsf{v}_a(choice, type_a) - payment_a$$

in the terminal situation.

We now show how important concepts of mechanism design can be expressed in $\mathsf{SL}[\mathcal{F}]$.

5.3. Implementation of social choice functions

Before defining what it means for a mechanism to implement a social choice function, we recall two classical concepts of equilibria, Nash equilibria and dominant strategy equilibria, classically used to define implementation.

Nash equilibria A strategy profile is a *Nash equilibrium* (*NE*) if no agent can increase her utility with a unilateral change of strategy [102]:

Definition 10. Let $\mathcal{G}$ be a mechanism, $\boldsymbol{\theta} = (\theta_a)_{a \in \mathrm{Ag}}$ be a type profile, and $v_\iota^{\boldsymbol{\theta}}$ be the initial state associated to $\boldsymbol{\theta}$. Let $\boldsymbol{\sigma} = (\sigma_a)_{a \in \mathrm{Ag}}$ be a strategy profile and $u_a(\alpha, \theta_a)$ be the utility associated to alternative α (corresponding to $\mathcal{G}[\boldsymbol{\theta}, \boldsymbol{\sigma}]$), and agent a's type θ_a. Strategy profile $\boldsymbol{\sigma}$ is a Nash Equilibrium (NE) from $v_\iota^{\boldsymbol{\theta}}$ iff for all agent a, for all strategy profile $\boldsymbol{\sigma'}$, it holds $u_a(\mathcal{G}[\boldsymbol{\theta}, (\boldsymbol{\sigma}_{-a}, \sigma'_a)], \theta_a) \leq u_a(\mathcal{G}[\boldsymbol{\theta}, \boldsymbol{\sigma}], \theta_a)$.

The above definition assumes that only plays starting from the initial state associated to θ are considered. Just as Strategy Logic can express Nash equilibria for Boolean objectives, $\mathsf{SL}[\mathcal{F}]$ can express Nash equilibria with quantitative objectives. Let us define the formula

$$NE(\boldsymbol{s}) := \bigwedge_{a\in \mathrm{Ag}} \forall t.\, \big[(\mathrm{Ag}_{-a}, \boldsymbol{s}_{-a})(a,t)\mathbf{F}(terminal \wedge util_a) \leq (\mathrm{Ag}, \boldsymbol{s})\mathbf{F}(terminal \wedge util_a)\big]$$

where $\boldsymbol{s} = (s_a)_{a\in \mathrm{Ag}}$ is a profile of strategy variables. Intuitively, the strategy profile assigned to $\boldsymbol{s} = (s_a)_{a\in \mathrm{Ag}}$ is a Nash equilibrium if, for each agent a and each alternative strategy t, agent a's utility when she follows the strategy t while others follow their strategy in $\boldsymbol{s}$ is *not greater* than her utility if she had also followed her strategy in $\boldsymbol{s}$. The lemma below follows directly from the definitions of Nash equilibrium in a wCGSii and the formula $NE(\boldsymbol{s})$.

Lemma 1. *Let $\theta = (\theta_a)_{a\in \mathrm{Ag}}$ be a type profile, and v_ι^θ be the initial state associated to θ. For every assignment $\mathcal{A}$, $[\![NE(\boldsymbol{s})]\!]_{\mathcal{A}}^{\mathcal{G},\rho}(v_\iota^\theta) = 1$ iff $(\mathcal{A}(s_a))_{a\in \mathrm{Ag}}$ is a NE from v_ι^θ in $\mathcal{G}$.*

Proof. (Sketch) Suppose that $[\![NE(\boldsymbol{s})]\!]_{\mathcal{A}}^{\mathcal{G},\rho}(v_\iota^\theta) = 1$; then $[\![\forall t.\, \big[(\mathrm{Ag}_{-a}, \boldsymbol{s}_{-a})(a,t)\mathbf{F}(terminal \wedge util_a) \leq (\mathrm{Ag}, \boldsymbol{s})\mathbf{F}(terminal \wedge util_a)\big]\!]_{\mathcal{A}}^{\mathcal{G},\rho}(v_\iota^\theta) = 1$ for any agent a. Thus, $[\![(\mathrm{Ag}_{-a}, \boldsymbol{s}_{-a})(a,t)\mathbf{F}(terminal \wedge util_a) \leq (\mathrm{Ag}, \boldsymbol{s})\mathbf{F}(terminal \wedge util_a)]\!]_{\mathcal{A}[s\mapsto\sigma]}^{\mathcal{G},\rho}(v_\iota^\theta) = 1$ for any agent a and strategy t, i.e. $[\![(\mathrm{Ag}_{-a}, \boldsymbol{s}_{-a})(a,t)\mathbf{F}(terminal \wedge util_a)]\!]_{\mathcal{A}[s\mapsto\sigma, t\mapsto\sigma']}^{\mathcal{G},\rho}(v_\iota^\theta) \leq [\![(\mathrm{Ag}, \boldsymbol{s})\mathbf{F}(terminal \wedge util_a)]\!]_{\mathcal{A}[s\mapsto\sigma]}^{\mathcal{G},\rho}(v_\iota^\theta)$. According to the definitions of $\mathcal{G}$ as a mechanism (Definition 8) and $util_a$, in the state where *terminal* holds, we have the outcomes $\mathcal{G}[\theta, (\boldsymbol{\sigma}_{-a}, \sigma'_a)]$ and $\mathcal{G}[\theta, \boldsymbol{\sigma}]$ and it must be the case that $u_a(\mathcal{G}[\theta, (\boldsymbol{\sigma}_{-a}, \sigma'_a)], \theta_a) \leq u_a(\mathcal{G}[\theta, \boldsymbol{\sigma}], \theta_a)$ due to the definition of wCGSii. Consequently $(\mathcal{A}(s_a))_{a\in \mathrm{Ag}}$ is a NE from v_ι^θ in $\mathcal{G}$. The other direction can be proved in a similar way. □

Remark 5. For verifying the uniqueness of a Nash equilibrium, we can check whether there exist two assignments $\mathcal{A} \neq \mathcal{A}'$ such that $[\![NE(\boldsymbol{s})]\!]_{\mathcal{A}}^{\mathcal{G},\rho}(v_\iota^\theta) = [\![NE(\boldsymbol{s})]\!]_{\mathcal{A}'}^{\mathcal{G},\rho}(v_\iota^\theta) = 1$.

Dominant strategy equilibria A strategy σ_a is a *dominant strategy* (*DS*) for agent a if it weakly maximizes her utility, for all possible strategies of other agents [3].

Definition 11. Let $\mathcal{G}$ be a mechanism, $\theta = (\theta_a)_{a\in \mathrm{Ag}}$ be a type profile, and v_ι^θ be the initial state associated to θ. Let $\boldsymbol{\sigma} = (\sigma_a)_{a\in \mathrm{Ag}}$ be a strategy profile and $u_a(\alpha, \theta_a)$ be the utility associated to alternative α (corresponding to $\mathcal{G}[\theta, \boldsymbol{\sigma}]$), and agent a's type θ_a. Strategy profile $\boldsymbol{\sigma}$ is a dominant strategy from v_ι^θ iff for agent a, for all strategy profile $\boldsymbol{\sigma}'$, it holds $u_a(\mathcal{G}[\theta, \boldsymbol{\sigma}'], \theta_a) \leq u_a(\mathcal{G}[\theta, (\boldsymbol{\sigma}'_{-a}, \sigma_a)]], \theta_a)$.

Let us rephrase this in $\mathsf{SL}[\mathcal{F}]$ by first defining the following formula

$$DS(s_a, a) := \forall \boldsymbol{t}.\big[(\mathrm{Ag}, \boldsymbol{t})\mathbf{F}(terminal \wedge util_a) \leq (a, s_a)(\mathrm{Ag}_{-a}, \boldsymbol{t}_{-a})\mathbf{F}(terminal \wedge util_a)\big]$$

Intuitively, strategy s_a is dominant iff the utility for agent a is maximized: switching to any other strategy leads to a lower or equal utility whatever the other agents do.

Proposition 1. *Let $\theta = (\theta_a)_{a\in \mathrm{Ag}}$ be a type profile, and v_ι^θ be the initial state associated to θ. For an assignment $\mathcal{A}$, it holds that $[\![DS(s_a, a)]\!]_{\mathcal{A}}^{\mathcal{G},\rho}(v_\iota^\theta) = 1$ iff $\mathcal{A}(s_a)$ is a dominant strategy from v_ι^θ for a in $\mathcal{G}$.*

Proof. (Sketch) From left to right. For any $\boldsymbol{t}$, we have $[\![(\mathrm{Ag}, \boldsymbol{t})\mathbf{F}(terminal \wedge util_a) \leq (a, s_a)(\mathrm{Ag}_{-a}, \boldsymbol{t}_{-a})\mathbf{F}(terminal \wedge util_a)]\!]_{\mathcal{A}[s\mapsto\sigma, t\mapsto\sigma']}^{\mathcal{G},\rho}(v_\iota^\theta) = 1$. Consequently, $[\![(\mathrm{Ag}, \boldsymbol{t})\mathbf{F}(terminal \wedge util_a)]\!]_{\mathcal{A}[t\mapsto\sigma']}^{\mathcal{G},\rho}(v_\iota^\theta) \leq [\![(a, s_a)(\mathrm{Ag}_{-a}, \boldsymbol{t}_{-a})\mathbf{F}(terminal \wedge util_a)]\!]_{\mathcal{A}[t\to\sigma', a\mapsto\sigma]}^{\mathcal{G},\rho}(v_\iota^\theta)$. According to the definitions of $\mathcal{G}$ and $util_a$, in the state where *terminal* holds, we have the outcomes $\mathcal{G}[\theta, \boldsymbol{\sigma}']$ and $\mathcal{G}[\theta, (\boldsymbol{\sigma}'_{-a}, \sigma_a)]$ and it must be the case that $u_a(\mathcal{G}[\theta, \boldsymbol{\sigma}'], \theta_a) \leq u_a(\mathcal{G}[\theta, (\boldsymbol{\sigma}'_{-a}, \sigma_a)], \theta_a)$ due to the definition of wCGSii as a mechanism (Definition 8). Consequently $(\mathcal{A}(s_a))_{a\in \mathrm{Ag}}$ is a DS from v_ι^θ in $\mathcal{G}$. The other direction can be proved in a similar way. □

A strategy profile $\sigma = (\sigma_a)_{a\in \mathrm{Ag}}$ is a *dominant strategy equilibrium* (*DSE*) if each σ_a is a dominant strategy for agent a which is represented as follows:

$$DSE(\boldsymbol{s}) := \bigwedge_{a\in \mathrm{Ag}} DS(s_a, a)$$

In other words, strategy profile $\boldsymbol{s}$ is a dominant strategy equilibrium if every individual strategy s_a is a dominant one Similarly to Nash equilibria, the following holds.

Lemma 2. *Let $\theta = (\theta_a)_{a\in \mathrm{Ag}}$ be a type profile, and v_ι^θ be the initial state associated to θ. For every assignment $\mathcal{A}$, we have that $[\![DSE(\boldsymbol{s})]\!]_{\mathcal{A}}^{\mathcal{G},\rho}(v_\iota^\theta) = 1$ iff $(\mathcal{A}(s_a))_{a\in \mathrm{Ag}}$ is a DSE from v_ι^θ in $\mathcal{G}$.*

Proof. This is a direct consequence of the previous proposition and Definition 11. □

Implementation of a SCF Informally, a mechanism implements a social choice function if the alternative chosen in *equilibrium* strategies is the same as the one chosen by the SCF, for all possible agent preferences; in case of multiple equilibria it is required that there exists one equilibrium that agrees with the SCF [102]. The definition of implementation is modular, so that different equilibrium concepts may be used, including Nash and dominant strategy equilibria.

Definition 12. Let $E \in \{NE, DSE\}$ be a solution concept and f a social choice function. A mechanism $\mathcal{G}$ *E-implements* f if for all type profiles $\boldsymbol{\theta} \in \boldsymbol{\Theta}$ there exists an E-equilibrium $\boldsymbol{\sigma}$ in $\mathcal{G}$ from $v_\iota^{\boldsymbol{\theta}}$ such that $\mathcal{G}[\boldsymbol{\theta}, \boldsymbol{\sigma}] = \mathsf{f}(\boldsymbol{\theta})$.

Let us again consider Dutch auctions but from the perspective of implementation of an SCF.

Example 5. Under the assumption that a Nash equilibrium exists, the Dutch auction (see Example 3) is known to implement the first-price social choice function [101] introduced in Example 2.

For a social choice function $\mathsf{f} = (\mathsf{x}, \{\mathsf{p}_a\})$ and a type profile $\boldsymbol{\theta}$, define the $\mathsf{SL}[\mathcal{F}]$ formula

$$\varphi_{\mathsf{f}(\boldsymbol{\theta})} := \mathit{choice} = x \wedge \bigwedge_{a \in \mathrm{Ag}} \mathit{payment}_a = p_a$$

where $\mathsf{x}(\boldsymbol{\theta}) = x$, $\mathsf{p}_a(\boldsymbol{\theta}) = p_a$, and values p_a, x are constants (0-ary functions) in $\mathcal{F}$. The formula $\varphi_{\mathsf{f}(\boldsymbol{\theta})}$ expresses the condition in which the social choice function assigns the same alternative as the one encoded in a state.

Define also, for $E \in \{NE, DSE\}$,

$$\varphi_{\mathit{impl}}(\mathsf{f}, E, \boldsymbol{\theta}) := \exists \boldsymbol{s}.\, E(\boldsymbol{s}) \wedge \mathbf{F}(\mathit{terminal} \wedge \varphi_{\mathsf{f}(\boldsymbol{\theta})})$$

that is,

$$\varphi_{\mathit{impl}}(\mathsf{f}, E, \boldsymbol{\theta}) := \exists \boldsymbol{s}.\, E(\boldsymbol{s}) \wedge \mathbf{F}(\mathit{terminal} \wedge \mathit{choice} = x \wedge \bigwedge_{a \in \mathrm{Ag}} \mathit{payment}_a = p_a)$$

This formula says that there exists an E-equilibrium that leads to choice $\mathsf{f}(\boldsymbol{\theta})$. It can thus be used to express that a mechanism implements a given social choice function. The following theorem states what implementation means when it is represented as wCGSii and $\mathsf{SL}[\mathcal{F}]$ formulas:

Theorem 1. *A mechanism $\mathcal{G}$ E-implements an SCF f iff for every type profile $\boldsymbol{\theta} \in \boldsymbol{\Theta}$, $[\![\varphi_{\mathit{impl}}(\mathsf{f}, E, \boldsymbol{\theta})]\!]^{\mathcal{G},\rho}(v_\iota^{\boldsymbol{\theta}}) = 1$.*

Proof. Fix a solution concept $E \in \{NE, DSE\}$, a social choice function $\mathsf{f} = (\mathsf{x}, \{\mathsf{p}_a\})$, a mechanism $\mathcal{G}$, a assignment $\mathcal{A}$ and a type profile $\boldsymbol{\theta} \in \boldsymbol{\Theta}$. For each agent $a \in \mathrm{Ag}$ let x and p_a be constants in [-1, 1] denoting the alternative chosen by f, that is $\mathsf{x}(\boldsymbol{\theta}) = x$, and $\mathsf{p}_a(\boldsymbol{\theta}) = p_a$.

Assume $\mathcal{G}$ E-implements f. By definition there exists a strategy profile $\boldsymbol{\sigma}$ that is an E-equilibrium solution in $\mathcal{G}$ from $v_\iota^{\boldsymbol{\theta}}$ and such that $\mathcal{G}[\boldsymbol{\theta}, \boldsymbol{\sigma}] = \mathsf{f}(\boldsymbol{\theta})$. It follows that:

First, because the $\mathsf{SL}[\mathcal{F}]$ formula $\mathbf{E}(\boldsymbol{s})$ correctly characterizes $\mathbf{E}$-equilibria (Lemma 1 and 2), letting $\mathcal{A}_{\boldsymbol{\sigma}} : s_a \mapsto \sigma_a$ we have that $[\![E(\boldsymbol{s})]\!]^{\mathcal{G},\rho}_{\mathcal{A}_{\boldsymbol{\sigma}}}(v_\iota^{\boldsymbol{\theta}}) = 1$.

Second, the fact that $\mathcal{G}[\boldsymbol{\theta}, \boldsymbol{\sigma}] = \mathsf{f}(\boldsymbol{\theta})$ implies that in the terminal state $v^{\mathit{terminal}} = v(\boldsymbol{\theta}, \boldsymbol{\sigma})$ (which is reached from $v_\iota^{\boldsymbol{\theta}}$ via $\boldsymbol{\sigma}$), we have $\ell(v^{\mathit{terminal}}, \mathit{choice}) = x$ and $\ell(v^{\mathit{terminal}}, \mathit{payment}_a) = p_a$, for each $a \in \mathrm{Ag}$. Therefore, we have $[\![\mathit{choice} = x \wedge \bigwedge_{a \in \mathrm{Ag}} \mathit{payment}_a = p_a]\!]^{\mathcal{G},\rho}_{\mathcal{A}_{\boldsymbol{\sigma}}}(v^{\mathit{terminal}}) = 1$ or simply $[\![\varphi_{\mathsf{f}(\boldsymbol{\theta})}]\!]^{\mathcal{G},\rho}_{\mathcal{A}_{\boldsymbol{\sigma}}}(v^{\mathit{terminal}}) = 1$. By the semantics of $\mathbf{F}$, it follows that $[\![\mathbf{F}(\mathit{terminal} \wedge \varphi_{\mathsf{f}(\boldsymbol{\theta})})]\!]^{\mathcal{G},\rho}_{\mathcal{A}_{\boldsymbol{\sigma}}}(v_\iota^{\boldsymbol{\theta}}) = 1$.

Therefore, $[\![\exists \boldsymbol{s}.\, E(\boldsymbol{s}) \wedge \mathbf{F}(\mathit{terminal} \wedge \varphi_{\mathsf{f}(\boldsymbol{\theta})})]\!]^{\mathcal{G},\rho}_{\mathcal{A}}(v_\iota^{\boldsymbol{\theta}}) = 1$ (the maximal value 1 is attained for strategy profile $\boldsymbol{\sigma}$) and $[\![\varphi_{\mathit{impl}}(\mathsf{f}, E, \boldsymbol{\theta})]\!]^{\mathcal{G},\rho}_{\mathcal{A}}(v_\iota^{\boldsymbol{\theta}}) = 1$.

Conversely, assume $[\![\exists \boldsymbol{s}.\, E(\boldsymbol{s}) \wedge \mathbf{F}(\mathit{terminal} \wedge \varphi_{\mathsf{f}(\boldsymbol{\theta})})]\!]^{\mathcal{G},\rho}_{\mathcal{A}}(v_\iota^{\boldsymbol{\theta}}) = 1$. By the semantics of the strategy quantifier, there exists a strategy profile $\boldsymbol{\sigma}$ such that $[\![E(\boldsymbol{s}) \wedge \mathbf{F}(\mathit{terminal} \wedge \varphi_{\mathsf{f}(\boldsymbol{\theta})})]\!]^{\mathcal{G},\rho}_{\mathcal{A}_{\boldsymbol{\sigma}}}(v_\iota^{\boldsymbol{\theta}}) = 1$, where $\mathcal{A}_{\boldsymbol{\sigma}} : s_a \mapsto \sigma_a$. It follows that $[\![E(\boldsymbol{s})]\!]^{\mathcal{G},\rho}_{\mathcal{A}_{\boldsymbol{\sigma}}}(v_\iota^{\boldsymbol{\theta}}) = 1$ and $[\![\mathbf{F}(\mathit{terminal} \wedge \varphi_{\mathsf{f}(\boldsymbol{\theta})})]\!]^{\mathcal{G},\rho}_{\mathcal{A}_{\boldsymbol{\sigma}}}(v_\iota^{\boldsymbol{\theta}}) = 1$. The former implies that $\boldsymbol{\sigma}$ is an E-equilibrium, and since *terminal* has value -1 in all non-terminal states, the latter implies that in the terminal state $v^{\mathit{terminal}} = v(\boldsymbol{\theta}, \boldsymbol{\sigma})$ we have $[\![\varphi_{\mathsf{f}(\boldsymbol{\theta})}]\!]^{\mathcal{G},\rho}_{\mathcal{A}_{\boldsymbol{\sigma}}}(v^{\mathit{terminal}}) = 1$ This in turn means that $\mathcal{G}[\boldsymbol{\theta}, \boldsymbol{\sigma}] = \mathsf{f}(\boldsymbol{\theta})$, hence $\mathcal{G}$ E-implements f.

Notice that if there is no $\boldsymbol{\sigma}$ such that $\boldsymbol{\sigma}$ is an E-equilibrium solution in $\mathcal{G}$, we have that $\mathcal{G}$ does not E-implement f, and $[\![\varphi_{\mathit{impl}}(\mathsf{f}, E, \boldsymbol{\theta})]\!]^{\mathcal{G},\rho}_{\mathcal{A}}(v_\iota^{\boldsymbol{\theta}}) = -1$. □

In the next section, we show how to express and verify properties of social choice functions by evaluating $\mathsf{SL}[\mathcal{F}]$ formulas on mechanisms that implement them. We will use the following parameterized formula to capture in a mechanism some equilibrium that implements the social function, and check a property of the resulting alternative. For a social choice function f, a type profile $\boldsymbol{\theta} \in \boldsymbol{\Theta}$, an equilibrium type $E \in \{NE, DSE\}$ and a formula φ expressing a property of the final alternative, define

$$\mathit{Capture\text{-}alt}(\mathsf{f}, E, \boldsymbol{\theta}, \varphi) := \exists \boldsymbol{s}.\, E(\boldsymbol{s}) \wedge \mathbf{F}(\mathit{terminal} \wedge \varphi_{\mathsf{f}(\boldsymbol{\theta})} \wedge \varphi)$$

5.4. Mechanism properties

We show how SL[$\mathcal{F}$] can express a variety of important notions in mechanism design. Let us first distinguish two types of mechanisms: *direct* and *indirect*. When participating in a mechanism, in general, agents may adjust their strategies based on the information feedback received through the game, that is, the observation of the new situation. For instance, in a Dutch auction, agents' observe the price at each state for deciding whether to bid. Mechanisms of this form are also called *indirect*.

A *direct* mechanism, such as Vickrey auction [103], is a type of mechanism that is non-iterative and in which the agents' available actions are to report any of their possible types. That is, a mechanism $\mathcal{G}$ is direct if initial states lead directly to terminal states, and $\mathrm{Ac}_a = \Theta_a$ for each a. Equivalently, it is a social choice function, as it maps type profiles to alternatives, and every social choice function can be seen as a direct mechanism [104].

Strategyproofness One of the core challenges in mechanism design is to ensure that an agent would prefer "telling the truth" by reporting her real type rather than trying to manipulate the game by reporting some other type [3]. Direct mechanisms that ensure this property are called *strategyproof* (*SP*) or *dominant strategy incentive-compatible*.

In a direct mechanism $\mathcal{G}$, we let $\hat{\theta}_a$ be the truth-revealing strategy for a, defined as $\hat{\theta}_a(v_\iota^{\boldsymbol{\theta}}) = \theta_a$.

Definition 13. A direct mechanism $\mathcal{G}$ is *strategyproof* if $(\hat{\theta}_a)_{a\in\mathrm{Ag}}$ is a dominant strategy equilibrium from $v_\iota^{\boldsymbol{\theta}}$, for all $\boldsymbol{\theta} \in \boldsymbol{\Theta}$.

Strategyproofness of a direct mechanism $\mathcal{G}$ can be expressed in SL[$\mathcal{F}$] by verifying whether the SL[$\mathcal{F}$]-formula $\mathit{DSE}(\boldsymbol{s})$ characterizing dominant strategy equilibrium has satisfaction value 1 on $\mathcal{G}$, where $\boldsymbol{s}$ denotes the joint strategy in which each agent truthfully reports her type. The following holds:

Proposition 2. *A direct mechanism $\mathcal{G}$ is strategy proof iff $[\![\mathit{DSE}(\boldsymbol{s})]\!]^{\mathcal{G},\rho}_{\mathcal{A}}(v_\iota^{\boldsymbol{\theta}}) = 1$ for all $\boldsymbol{\theta} \in \boldsymbol{\Theta}$, where $\mathcal{A}(s_a) = \hat{\theta}_a$ for each a.*

Proof. Fix a direct mechanism $\mathcal{G}$. We have that $\mathrm{Ac}_a = \Theta_a$ for each a, and $\mathcal{G}$ is strategyproof iff, for each initial state $v_\iota^{\boldsymbol{\theta}}$, the truth revealing strategy $\hat{\theta}_a$ is a dominant strategy for each $a \in \mathrm{Ag}$. By the semantics of formula $\mathit{DS}(s)$, for any type profile $\boldsymbol{\theta}$, each strategy $\hat{\theta}_a$ is dominant from $v_\iota^{\boldsymbol{\theta}}$ iff $[\![\mathit{DS}(s)]\!]^{\mathcal{G},\rho}_{\mathcal{A}_{\hat{\theta}_a}}(v_\iota^{\boldsymbol{\theta}}) = 1$, where $\mathcal{A}_{\hat{\theta}_a} : s \mapsto \hat{\theta}_a$. So for a type profile $\boldsymbol{\theta}$, all strategies $\hat{\theta}_a$ are dominant from $v_\iota^{\boldsymbol{\theta}}$ iff $[\![\mathit{DSE}(\boldsymbol{s})]\!]^{\mathcal{G},\rho}_{\mathcal{A}}(v_\iota^{\boldsymbol{\theta}}) = 1$, where $\mathcal{A} : s_a \mapsto \hat{\theta}_a$ for all a. □

Individual rationality Individual rationality (IR) expresses the idea that an agent has an incentive to participate [102], that is, she can ensure to always get nonnegative utility. Hereafter we express in SL[$\mathcal{F}$] the notion of (ex-post) individual rationality [3].

Definition 14. An SCF $\mathsf{f} = (\mathsf{x}, \{\mathsf{p}_a\})$ is *individually rational* if for every $\boldsymbol{\theta} \in \boldsymbol{\Theta}$, $\mathsf{v}_a(\mathsf{x}(\boldsymbol{\theta})) - \mathsf{p}_a(\boldsymbol{\theta}) \geq 0$ for each agent a.

Let us define the following formula which states that utility is nonnegative:

$$\mathit{IR} := \bigwedge_{a\in\mathrm{Ag}} 0 \leq \mathit{util}_a$$

Given a mechanism that E-implements an SCF f, checking that f satisfies IR amounts to checking that formula IR has satisfaction value one in the E-equilibrium that implements f, for every possible type profile $\boldsymbol{\theta}$.

Proposition 3. *Let f be an SCF, $E \in \{\mathit{NE}, \mathit{DSE}\}$, and $\mathcal{G}$ a mechanism that E-implements f. f is individually rational iff $[\![\mathit{Capture\text{-}alt}(\mathsf{f}, E, \boldsymbol{\theta}, \mathit{IR})]\!]^{\mathcal{G},\rho}(v_\iota^{\boldsymbol{\theta}}) = 1$ for all $\boldsymbol{\theta} \in \boldsymbol{\Theta}$.*

Proof. Fix a solution concept $E \in \{\mathit{NE}, \mathit{DSE}\}$, a social choice function f, a mechanism $\mathcal{G}$ that E-implements f, any assignment $\mathcal{A}$ and any type profile $\boldsymbol{\theta} \in \boldsymbol{\Theta}$.

Assume f is individually rational. Because $\mathcal{G}$ implements f there exists a strategy profile $\boldsymbol{\sigma}$ that is an E equilibrium from $v_\iota^{\boldsymbol{\theta}}$ and such that $\mathcal{G}[v_\iota^{\boldsymbol{\theta}}, \boldsymbol{\sigma}] = \mathsf{f}(\boldsymbol{\theta})$. Let $v^{\mathit{terminal}} = v(\boldsymbol{\theta}, \boldsymbol{\sigma})$. Since f is individually rational, we have that $[\![\mathit{IR}]\!]^{\mathcal{G},\rho}_{\mathcal{A}}(v^{\mathit{terminal}}) = 1$. Therefore, using $\boldsymbol{\sigma}$ as a witness, we obtain that $[\![\exists \boldsymbol{s}.\, E(\boldsymbol{s}) \wedge \mathbf{F}(\mathit{terminal} \wedge \varphi_{\mathsf{f}(\boldsymbol{\theta})} \wedge \mathit{IR})]\!]^{\mathcal{G},\rho}_{\mathcal{A}}(v_\iota^{\boldsymbol{\theta}}) = 1$.

The converse is proved in a similar way. □

Efficiency A social choice function is efficient (*EF*) if it assigns a choice that maximizes the social welfare, i.e., the total value over all agents [102].

Definition 15. A social choice function $\mathsf{f} = (\mathsf{x}, \{\mathsf{p}_a\})$ is *allocatively efficient* if for all $\theta \in \mathbf{\Theta}$,

$$\sum_{a \in \mathrm{Ag}} \mathsf{v}_a(\mathsf{x}(\theta), \theta_a) = \max_{x \in \mathcal{X}} \sum_{a \in \mathrm{Ag}} \mathsf{v}_a(x, \theta_a)$$

Define formula

$$\mathit{Eff} := \sum_{a \in \mathrm{Ag}} \mathsf{v}_a(\mathit{choice}, \mathit{type}_a) = \mathit{maxv}_\theta$$

where, for each θ, $\mathit{maxv}_\theta = \max_{x \in \mathcal{X}} \sum_{a \in \mathrm{Ag}} \mathsf{v}_a(x, \theta_a)$ is a constant in $\mathcal{F}$. In a terminal state, it means that the social welfare is the maximal from the possible social choices in $\mathcal{X}$.

The following proposition shows how one can determine whether an SCF f is efficient by verifying the satisfaction value of the formula *Eff* in a mechanism that implements f.

Proposition 4. *Let* f *be an SCF,* $E \in \{NE, DSE\}$*, and* $\mathcal{G}$ *a mechanism that E-implements* f*.* f *is allocatively efficient iff* $[\![\mathit{Capture\text{-}alt}(\mathsf{f}, E, \theta, \mathit{Eff})]\!]^{\mathcal{G}, \rho}(v_\iota^\theta) = 1$ *for all* $\theta \in \mathbf{\Theta}$*.*

Proof. Analogous to the proof of Proposition 3. □

Budget-balance Budget-balance focuses on the monetary transfer between buyers and sellers. Strong budget-balance (SBB) requires strict balance in this transfer. The no-deficit condition, or weak budget-balance (WBB), characterizes no monetary loss. We recall the notions of strong and weak budget balance [102]:

Definition 16. A social choice function $\mathsf{f} = (\mathsf{x}, \{\mathsf{p}_a\})$ is *strongly budget-balanced* (resp., *weakly budget-balanced*) if $\sum_{a \in \mathrm{Ag}} p_a(\theta) = 0$ (resp., $\sum_{a \in \mathrm{Ag}} p_a(\theta) \geq 0$) for all $\theta \in \mathbf{\Theta}$.

Define formula *SBB* requiring a strict balance, i.e. equal to zero:

$$\mathit{SBB} := 0 = \sum_{a \in \mathrm{Ag}} \mathit{payment}_a$$

and define *WBB* similarly, with $\leq$ instead of $=$. Again, we can prove that

Proposition 5. *Let* f *be an SCF,* $E \in \{NE, DSE\}$*, and* $\mathcal{G}$ *a mechanism that E-implements* f*. It holds that* f *is SBB iff* $[\![\mathit{Capture\text{-}alt}(\mathsf{f}, E, \theta, \mathit{SBB})]\!]^{\mathcal{G}, \rho}(v_\iota^\theta) = 1$ *for all* $\theta \in \mathbf{\Theta}$*, and similarly for WBB.*

Proof. Analogous to the proof for Proposition 3. □

Example 6. A Dutch auction is WBB, because the bid price is non-negative, and it is IR because the strategy of waiting in every state leads to zero utility, and thus any equilibrium would not have a negative utility. One can check that in the mechanism $\mathcal{G}_{\mathrm{dut}}$ from Example 3, with the first-price social choice function f_{fp}, for any type profile θ we have $[\![\mathit{Capture\text{-}alt}(\mathsf{f}_{\mathrm{fp}}, NE, \theta, IR)]\!]^{\mathcal{G}_{\mathit{dut}}, \rho}(v_\iota^\theta) = 1$ and $[\![\mathit{Capture\text{-}alt}(\mathsf{f}_{\mathrm{fp}}, NE, \theta, \mathit{WBB})]\!]^{\mathcal{G}_{\mathit{dut}}, \rho}(v_\iota^\theta) = 1$.

Pareto optimality A social choice function is *Pareto optimal* (*PO*) if it chooses an alternative for which no other alternative is strongly preferred by at least one agent, and weakly preferred by all others [102]. Formally:

Definition 17. A social choice function $\mathsf{f} = (\mathsf{x}, \{\mathsf{p}_a\})$ is *Pareto optimal* if, for all $\theta \in \mathbf{\Theta}$, for all $a \in \mathrm{Ag}$ and for all $\alpha \neq \mathsf{f}(\theta)$, if $u_a(\alpha, \theta_a) > u_a(\mathsf{f}(\theta), \theta_a)$ then there exists an agent $b \in Ag$ such that $u_b(\alpha, \theta_b) < u_b(\mathsf{f}(\theta), \theta_b)$.

For every alternative $\alpha \in \mathit{Alt}$, every type profile θ and agent a, let $\mathit{utilalt}_{a,\alpha} : \theta_a \mapsto u_a(\alpha, \theta_a)$ be a function in $\mathcal{F}$. Define formula (recall formula util_a, defined in Section 5.2):

$$\mathit{PO} := \bigwedge_{a \in \mathrm{Ag}, \alpha \in \mathit{Alt}} \left(\mathit{util}_a < \mathit{utilalt}_{\alpha,a}(\mathit{type}_a) \rightarrow \left(\bigvee_{b \in \mathrm{Ag}} \mathit{utilalt}_{\alpha,b}(\mathit{type}_a) < \mathit{util}_b\right)\right)$$

As for Proposition 3, we can prove:

Proposition 6. *Let* f *be an SCF,* $E \in \{NE, DSE\}$*, and* $\mathcal{G}$ *a mechanism that E-implements* f*. It holds that* f *is Pareto Optimal iff* $[\![\mathit{Capture\text{-}alt}(\mathsf{f}, E, \theta, \mathit{PO})]\!]^{\mathcal{G}, \rho}(v_\iota^\theta) = 1$ *for all* $\theta \in \mathbf{\Theta}$*.*

Proof. Analogous to the proof of Proposition 3. □

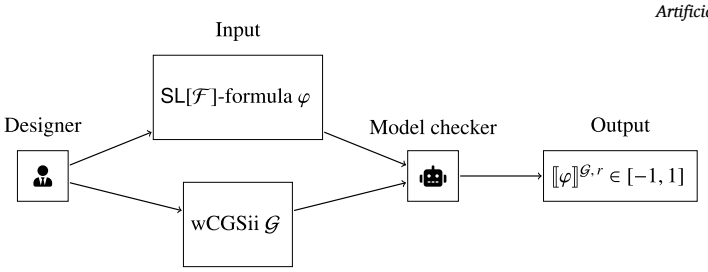


Fig. 3. Verification of a mechanism represented as a wCGSii in relation to a property expressed in SL[$\mathcal{F}$] using model checking with memoryless semantics. The output is the global satisfaction value of the formula in the wCGSii.

6. Verification of mechanisms with model checking

When the set of possible type profiles is finite, as it is the case here, our results from the previous section show that verifying the key MD properties *SP*, *IR*, *EF*, *BB* and *PO* on mechanisms amounts to model checking SL[$\mathcal{F}$] formulas for all type profiles of interest (see Fig. 3).

In this section, we show that model checking SL[$\mathcal{F}$] with imperfect information and memoryless agents is no harder than model checking LTL or classical SL with memoryless agents. Let us first define formally the quantitative model-checking problem for SL[$\mathcal{F}$].

Definition 18. The *model-checking problem* for SL[$\mathcal{F}$] consists in deciding, given a sentence φ, wCGS $\mathcal{G}$, state v in $\mathcal{G}$ and predicate $P \subseteq [-1,1]$, whether $[\![\varphi]\!]^{\mathcal{G},\rho}(v) \in P$.

ATL under perfect recall strategies and models with imperfect information has an undecidable model-checking problem [105]. As SL[$\mathcal{F}$] subsumes ATL, it is also undecidable.

Corollary 1. *Model checking* SL[$\mathcal{F}$] *with imperfect information and perfect recall agents is undecidable.*

Fortunately, for most preference aggregation settings, the information agents need to make a decision is encoded in the current game state (for instance, single executions of an auction protocol, direct mechanisms, etc). An exception is repeated game, in which agents can use their observation of the previous opponents' actions and/or the chosen outcomes to decide how to play. Even in this setting, however, the classical strategies considered in the literature are often memoryless. This is the case, for instance, in greedy strategies for repeated keyword auctions [106]. Another example includes strategies for iterative voting protocols in which agents alternate in changing their voting ballot based on their own preferences and the very last election outcome [107]. The case of memoryless strategies is therefore relevant for mechanism design, and we now show that in this setting the results are, as expected, much better.

For LTL[$\mathcal{F}$] model checking is in PSPACE [108], and so is model checking SL with memoryless agents [109]. We show that it is also the case for SL[$\mathcal{F}$]. We recall that, similarly to [19,21], our complexity results hold as long as the functions in $\mathcal{F}$ can be computed in the complexity class considered. In the case of Theorem 2, this means that the results hold whenever $\mathcal{F}$ can be computed in polynomial space. Notice that this assumption over the functions in $\mathcal{F}$ is enough to capture classic functions, such as disjunction, negation, average, minimum, etc.

Theorem 2. *Model checking* SL[$\mathcal{F}$] *with imperfect information and memoryless agents is* PSPACE*-complete.*

Proof. We first show that each recursive call only needs at most polynomial space. Let $\mathrm{Ac} = \cup_{a\in\mathrm{Ag}}\mathrm{Ac}_a$. First, observe that each assignment $\mathcal{A}$ can be stored in space $O((|\mathit{free}(\varphi)| + |\mathrm{Ag}|)\cdot|V|\cdot\log|\mathrm{Ac}|)$. Next, for the base case it is clear that $[\![p]\!]^{\mathcal{G},r}_{\mathcal{A}}(v)$ can be computed in constant space. For strategy quantification $[\![\exists s_a.\,\varphi]\!]^{\mathcal{G},r}_{\mathcal{A}}(v)$, besides the recursive call to $[\![\varphi]\!]^{\mathcal{G},r}_{\mathcal{A}[s\mapsto\sigma]}(v)$ we need space $O(|V|\cdot\log|\mathrm{Ac}|)$ to store the current strategy and the current maximum value computed. For $[\![f(\varphi_1,\ldots,\varphi_m)]\!]^{\mathcal{G},r}_{\mathcal{A}}(v)$, by assumption f is computed in polynomial space. For $[\![\mathbf{X}\varphi]\!]^{\mathcal{G},r}_{\mathcal{A}}(v)$, we only need to observe that the next state in $\mathrm{Out}(\mathcal{A},v)$ is computed in constant space.

Finally we detail how $[\![\varphi_1\mathbf{U}\varphi_2]\!]^{\mathcal{G},r}_{\mathcal{A}}(v)$ is computed. Let $\pi = \mathrm{Out}(\mathcal{A},v)$. Since $\mathcal{G}$ has finitely many states, there exist two indices $k<l$ such that $\pi_k = \pi_l$, and since strategies depend only on the current state, the suffix of π starting at index l is equal to the suffix starting at index k. So there exist $\rho_1 = v_0\ldots v_{k-1}$ and $\rho_2 = v_k\ldots v_{l-1}$ such that $\pi = \rho_1\cdot\rho_2^\omega$. It follows that

$$
\begin{aligned}
[\![\varphi_1\mathbf{U}\varphi_2]\!]^{\mathcal{G},r}_{\mathcal{A}}(v) &= \sup_{i\geq 0}\min\left([\![\varphi_2]\!]^{\mathcal{G},r}_{\mathcal{A}}(\pi_i),\ \min_{0\leq j<i}[\![\varphi_1]\!]^{\mathcal{G},r}_{\mathcal{A}}(\pi_j)\right)\\
&= \max_{0\leq i<l}\min\left([\![\varphi_2]\!]^{\mathcal{G},r}_{\mathcal{A}}(\pi_i),\ \min_{0\leq j<i}[\![\varphi_1]\!]^{\mathcal{G},r}_{\mathcal{A}}(\pi_j)\right)
\end{aligned}
$$

This can be computed by a while loop that increments i, computes $[\![\varphi_2]\!]^{\mathcal{G},r}_{\mathcal{A}}(\pi_i)$, $\min_{0\leq j<i}[\![\varphi_1]\!]^{\mathcal{G},r}_{\mathcal{A}}(\pi_j)$ and their minimum, records the result if it is bigger than the previous maximum, and stops upon reaching a state that has already been visited. This requires to store the current value of $\min_{0\leq j<i}[\![\varphi_1]\!]^{\mathcal{G},r}_{\mathcal{A}}(\pi_j)$, the current maximum, and the list of states already visited, which are at most $|V|$.

Next, the number of nested recursive calls is at most $|\varphi|$, so the total space needed is bounded by $|\varphi|$ times a polynomial in the size of the input, and is thus polynomial. □

We now focus on direct mechanisms, as defined in Definition 9, and show that in this case, we achieve a better complexity. Indeed, evaluating $\mathsf{SL}[\mathcal{F}]$ formulas in this context is no harder than evaluating QBF formulas whose quantifiers' structure follows that of strategy quantifiers in the $\mathsf{SL}[\mathcal{F}]$ formula.

Theorem 3. *Assuming that functions in $\mathcal{F}$ can be computed in polynomial time, the complexity of model checking $\mathsf{SL}[\mathcal{F}]$ with imperfect information and memoryless agents over direct mechanisms populates the polynomial hierarchy.*

Proof. Since plays in direct mechanisms consist of a single step, $\varphi_1 \mathbf{U} \varphi_2$ is equivalent to $\varphi_2 \vee \varphi_1 \wedge \mathbf{X}\varphi_2$. This gives a polynomial reduction to the model-checking problem for the until-free fragment of the logic.

On this fragment, observing the procedure in the previous proof, we see that we can solve the problem with an alternating Turing machine that runs in polynomial time and whose alternations follow the structure of strategy quantifiers in the formula. Every complexity class in the polynomial hierarchy can thus be obtained by imposing constraints on the formulas' quantifier structure: the problem is in $\Sigma^P_0 = P$ for formulas without strategy quantifiers, in Σ^P_{k+1} for formulas of alternation depth k that start with an existential quantifier, and in Π^P_{k+1} for formulas of alternation depth k that start with a universal quantifier. □

The good news is that the number of alternations between strategy quantifiers is typically very low. For instance, in all the formulas that we use in this paper, strategy quantifiers occur in formulas of the form $\exists \boldsymbol{s}.E(\boldsymbol{s})$, where $E(\boldsymbol{s})$ expresses some type of equilibrium and contains only universal strategy quantifiers. As a result, for all the formulas we consider, evaluating them on direct mechanisms is at the second level of the polynomial hierarchy, and more precisely in Σ^P_2.

7. Satisfiability and synthesis of $\mathsf{SL}[\mathcal{F}]$

We now focus on a logic-based approach for automatically creating new mechanisms. To do so, we rephrase the problem of AMD in terms of a synthesis problem where mechanisms are synthesized from a partial or complete specification in a high-level logical language. We solve the problem in two cases: when the number of actions is bounded, and when agents play in turn.

To achieve this goal, we start by investigating the satisfiability problem for $\mathsf{SL}[\mathcal{F}]$, which is defined as follows:

Definition 19. The *satisfiability problem* for $\mathsf{SL}[\mathcal{F}]$ takes a sentence $\varphi \in \mathsf{SL}[\mathcal{F}]$ and a threshold $\vartheta > -1$, and decides whether there exists a wCGS $\mathcal{G}$ s.t. $[\![\varphi]\!]^{\mathcal{G},R} \geq \vartheta$.

The satisfiability problem for SL (with perfect recall strategies) was proved undecidable in [110], but the proof there considers models with infinitely many actions. However, it is also known to be undecidable when considering finite models, i.e., models with both finitely many states and finitely many actions, as we do. Indeed, it is shown in [111,112] that satisfiability of ATL with strategy context (ATL_{sc}) is undecidable for finite models. Since ATL_{sc} can be expressed in SL [112] and SL in $\mathsf{SL}[\mathcal{F}]$, by taking $\mathcal{F} = \{\top, \vee, \neg\}$ [19], we obtain the following result.

Proposition 7. *The satisfiability problem for $\mathsf{SL}[\mathcal{F}]$ is undecidable as soon as $\mathcal{F}$ contains $\top$, $\vee$ and $\neg$.*

However satisfiability of SL is known to be decidable when restricted to turn-based systems or systems with a bounded number of actions [112]. We show that in these cases satisfiability is decidable for $\mathsf{SL}[\mathcal{F}]$ as well. To do so we first recall Booleanly-quantified CTL* ($\mathsf{BQCTL}^*[\mathcal{F}]$) and solve its satisfiability problem, and we then show that for bounded actions or turn-based systems satisfiability of $\mathsf{SL}[\mathcal{F}]$ reduces to that of $\mathsf{BQCTL}^*[\mathcal{F}]$.

Remark 6. In this section we focus on strategies with perfect recall. The main reason is that for memoryless strategies, satisfiability, and thus synthesis, is undecidable for SL even when restricted to the case of bounded actions or turn-based systems [112]. We also restrict attention to perfect information. Indeed, as we shall see, this setting is already undecidable and thus so would be the more general case of imperfect information. However for perfect information and perfect recall, we manage to obtain two decidable cases of practical interest: when the number of possible actions in the class of mechanisms considered is bounded, which is the case for instance when the only possible actions are to raise the hand or wait, as in many auction systems, and when agents play in turn, which is often the case in allocation mechanisms. From now on we will also assume, for convenience and without loss of generality, that all agents have the same set of actions and strategies Ac and *Str*.

7.1. Booleanly-quantified CTL*[F]

The logic BQCTL*[F] [19], a quantitative extension of QCTL* [113], is itself an extension of CTL* with quantifiers on atomic proposition. In BQCTL*[F] the semantics is quantitative, but the quantifiers on propositions consider only Boolean values. To be consistent with SL[F] we consider [-1,1] as range of values instead of [0,1] as in [19]. This changes nothing to the results presented there.

The syntax of BQCTL*[F] is defined by:

$$\varphi ::= p \mid \exists p.\, \varphi \mid \mathbf{E}\psi \mid f(\varphi, ..., \varphi)$$

$$\psi ::= \varphi \mid \mathbf{X}\psi \mid \psi \mathbf{U} \psi \mid f(\psi, ..., \psi)$$

where p ranges over AP and f over $\mathcal{F}$.

$\mathbf{E}\psi$ is the quantitative counterpart to the path quantifier of CTL*, and it maximizes the value of ψ over all branches. Similarly, $\exists p.\, \varphi$ maximizes the value of φ over all valuations for proposition p. Formulas of type φ are called *state formulas*, those of type ψ are called *path formulas*, and BQCTL*[F] consists of all the state formulas defined by the grammar. We again use $\top$, $\vee$, and $\neg$ to denote functions 1, max and $-x$, as well as classic abbreviations already introduced for SL[F].

Definition 20. A *weighted Kripke structure* (wKS) is a tuple $\mathcal{S} = (S, s_\iota, R, w)$ where S is a set of states, $s_\iota \in S$ is an initial state, $R \subseteq S \times S$ is a left-total (also called *serial*)[3] transition relation, and $w : S \times \mathrm{AP} \to [-1, 1]$ is a weight function.

A *path* in $\mathcal{S}$ is an infinite word $\lambda = s_0 s_1 ...$ over S such that $s_0 = s_\iota$ and $(s_i, s_{i+1}) \in R$ for all i. Finite prefixes of paths are *histories*, and we let $\mathrm{Hist}_\mathcal{S}$ be the set of all histories in $\mathcal{S}$. We also let $\mathrm{Val}_\mathcal{S} = \{w(s,p) \mid s \in S \text{ and } p \in \mathrm{AP}\}$ be the finite set of values appearing in $\mathcal{S}$.

Given finite nonempty sets X whose elements are called *directions* and $\mathcal{V} \subseteq [-1, 1]$ of possible values, a $\mathcal{V}^{\mathrm{AP}}$-labeled X-tree, (or $(\mathcal{V}^{\mathrm{AP}}, X)$-tree for short, or $\mathcal{V}^{\mathrm{AP}}$-tree when directions are understood), is a pair $t = (\tau, \ell)$ where $\tau \subseteq X^+$ is closed under non-empty prefixes, all *nodes* $u \in \tau$ start with the same direction r, called the *root*, and have at least one *child* $u \cdot d \in \tau$, and $\ell : \tau \to \mathcal{V}^{\mathrm{AP}}$ is a *weight function*. We let $\mathrm{Val}_t \subseteq \mathcal{V}$ be the image of ℓ on τ. A *branch* $\lambda = u_0 u_1 ...$ is an infinite sequence of nodes such that for all $i \geq 0$, u_{i+1} is a child of u_i. For $i \geq 0$, $\lambda_{\geq i}$ denotes the suffix of λ that starts at node u_i, and we let $Br(u)$ be the set of branches that start in node u.

Let $p \in \mathrm{AP}$. A *p-labeling* for a $\mathcal{V}$-tree $t = (\tau, \ell)$ is a mapping $\ell_p : \tau \to \{-1, 1\}$. The composition of t with ℓ_p is the $(\mathcal{V} \cup \{-1, 1\})^{\mathrm{AP}}$-tree defined as $t \otimes \ell_p := (\tau, \ell')$, where $\ell'(u)(p) = \ell_p(u)$ and $\ell'(u)(q) = \ell(u)(q)$ for $q \neq p$.

The *tree unfolding of a weighted Kripke structure* $\mathcal{S}$ with state set S is the $\mathrm{Val}_\mathcal{S}^{\mathrm{AP}}$-labeled S-tree $t_\mathcal{S} = (\mathrm{Hist}_\mathcal{S}, \ell')$, where $\ell'(p, u) = w(p, \mathrm{last}(u))$ for every $u \in \mathrm{Hist}_\mathcal{S}$ and $p \in \mathrm{AP}$.

A *binary tree* is a X-tree where $|X| = 2$. A tree is *regular* if it is the unfolding of some finite Kripke structure. A tree $t = (\tau, \ell)$ is *complete* if for all node $u \in \tau$ and direction $d \in X$, we have $u \cdot d \in \tau$. Given a $\mathcal{V}^{\mathrm{AP}}$-labeled X-tree $t = (\tau, \ell)$, we let $\overline{t} = (\overline{\tau}, \overline{\ell})$ be the only complete $\mathcal{V}^{\mathrm{AP}} \cup \{\bullet\}$-labeled X-tree such that for all $u \in \tau$, $\overline{\ell}(u) = \ell(u)$, and for all $u \in \overline{\tau} \setminus \tau$, $\overline{\ell}(u) = \{\bullet\}$, where $\bullet$ is a fresh symbol that labels artificial nodes added to make the tree complete. Similarly, every $\mathcal{V}^{\mathrm{AP}} \cup \{\bullet\}$-tree $\overline{t}$ induces a unique $\mathcal{V}^{\mathrm{AP}}$-tree t obtained by removing each subtree rooted in a node labeled with $\bullet$.

Different semantics exist for QCTL*, which differ on how proposition quantification is interpreted (see [113] for more on the different semantics for QCTL*). In this paper we focus on the tree semantics, which allows capturing perfect-recall strategies.

Definition 21 *(BQCTL*[F] semantics).* Consider a finite set $\mathcal{V} \subseteq [-1, 1]$ of possible values. The satisfaction value $[\![\varphi]\!]^t(u)$ of a BQCTL*[F] state formula φ in a node u of a $\mathcal{V}^{\mathrm{AP}}$-tree $t = (\tau, \ell)$, and the satisfaction value $[\![\psi]\!]^t(\lambda)$ of a path formula ψ along some branch λ of t, are defined inductively as follows:

$$[\![p]\!]^t(u) = \ell(u)(p)$$

$$[\![\exists p.\, \varphi]\!]^t(u) = \sup_{\ell_p : \tau \to \{-1,1\}} [\![\varphi]\!]^{t \otimes \ell_p}(u)$$

$$[\![\mathbf{E}\psi]\!]^t(u) = \sup_{\lambda \in Br(u)} [\![\psi]\!]^t(\lambda)$$

$$[\![f(\varphi_1, ..., \varphi_n)]\!]^t(u) = f([\![\varphi_1]\!]^t(u), ..., [\![\varphi_n]\!]^t(u))$$

$$[\![\varphi]\!]^t(\lambda) = [\![\varphi]\!]^t(\lambda_0)$$

$$[\![\mathbf{X}\psi]\!]^t(\lambda) = [\![\psi]\!]^t(\lambda_{\geq 1})$$

$$[\![\psi_1 \mathbf{U} \psi_2]\!]^t(\lambda) = \sup_{i \geq 0} \min([\![\psi_2]\!]^t(\lambda_{\geq i}), \min_{0 \leq j < i} [\![\psi_1]\!]^t(\lambda_{\geq j}))$$

$$[\![f(\psi_1, ..., \psi_n)]\!]^t(\lambda) = f([\![\psi_1]\!]^t(\lambda), ..., [\![\psi_n]\!]^t(\lambda))$$

[3] *i.e.*, for all $s \in S$, there exists s' such that $(s, s') \in R$.

For a tree t with root r we write $[\![\varphi]\!]^t$ for $[\![\varphi]\!]^t(r)$, for a weighted Kripke structure S we write $[\![\varphi]\!]^S$ for $[\![\varphi]\!]^{t_S}$.

7.2. Automata

We briefly recall some classic notions about alternating parity tree automata, and refer the reader to [114, Ch.8] for complete definitions (in particular that of acceptance of a tree by an automaton) and standard results on tree automata.

For a set Z, $\mathbb{B}^+(Z)$ is the set of formulas built from the elements of Z as atomic propositions using the connectives $\vee$ and $\wedge$, and with $\top, \bot \in \mathbb{B}^+(Z)$.

Fix a set $\mathcal{V} \subseteq [0,1]$. An *alternating parity tree automaton (APT)* on $(\mathcal{V}^{\mathrm{AP}}, X)$-trees is a tuple $\mathcal{A} = (Q, \delta, q_\iota, C)$ where Q is a finite set of states, $q_\iota \in Q$ is an initial state, $\delta : Q \times \mathcal{V}^{\mathrm{AP}} \to \mathbb{B}^+(X \times Q)$ is a transition function, and $C : Q \to \mathbb{N}$ is a coloring function.

The *size* $|\mathcal{A}|$ of an APT $\mathcal{A}$ is its number of states plus the sum of the sizes of all formulas appearing in the transition function. We also call *index* of an APT the number of priorities in its parity condition.

7.3. Satisfiability of $\mathsf{BQCTL}^*[\mathcal{F}]$

As for $\mathsf{SL}[\mathcal{F}]$, we define the quantitative satisfiability problem for $\mathsf{BQCTL}^*[\mathcal{F}]$ as follows.

Definition 22. The *satisfiability problem* for $\mathsf{BQCTL}^*[\mathcal{F}]$ takes a formula $\varphi \in \mathsf{BQCTL}^*[\mathcal{F}]$ and a threshold $\vartheta > -1$, and decides whether there exists a wKS S s.t. $[\![\varphi]\!]^{t_S} \geq \vartheta$.

Satisfiability for QCTL^* with structure semantics is undecidable [115], but decidable for the tree semantics [113] which we consider in this paper. Relying on the automata construction developed in [19] to model check $\mathsf{BQCTL}^*[\mathcal{F}]$ formulas, we show that satisfiability is also decidable for $\mathsf{BQCTL}^*[\mathcal{F}]$ (Theorem 4).

The lower bounds are inherited from the satisfiability problem for QCTL^* [113]. The reduction is direct with threshold $\vartheta = 1$. For the upper bounds, the first step is to show that one can restrict attention to structures with binary branching, i.e., where each state has at most two successor states.

7.3.1. Reduction to binary branching

We first show that the satisfiability problem for $\mathsf{BQCTL}^*[\mathcal{F}]$ in finite structures can be reduced to the same problem restricted to structures with binary branching. The proof is a simple adaptation to the quantitative setting of the one in [113] for QCTL^*. We sketch the proof and refer the reader to [113] for details.

For every wKS S we can define a wKS $\widetilde{S}$ with binary branching that simulates wKS: for each state of S with k successors, $\widetilde{S}$ contains a binary tree with k leaves, one for each successor. Internal nodes are dummy nodes, labeled with a fresh atomic proposition p_{int} which has value 1 in internal nodes, and -1 in normal states of S.

Now for every formula $\varphi \in \mathsf{BQCTL}^*[\mathcal{F}]$ we can define inductively a formula $\widetilde{\varphi}$ such that $[\![\varphi]\!]^S = [\![\widetilde{\varphi}]\!]^{\widetilde{S}}$. We only give the inductive case for temporal operators, all other cases distribute over the operators:

$$\widetilde{\mathbf{X}\varphi} = \mathbf{X}\left[p_{int}\mathbf{U}(\neg p_{int} \wedge \widetilde{\varphi})\right]$$
$$\widetilde{\varphi \mathbf{U} \varphi'} = (p_{int} \vee \widetilde{\varphi})\mathbf{U}(\neg p_{int} \wedge \widetilde{\varphi'})$$

We have the following:

Lemma 3. *For every* $\mathsf{BQCTL}^*[\mathcal{F}]$ *formula* φ *and every finite wKS* S, $[\![\varphi]\!]^S = [\![\widetilde{\varphi}]\!]^{\widetilde{S}}$.

7.3.2. Deciding satisfiability of $\mathsf{BQCTL}^*[\mathcal{F}]$

We first define a notion often used to characterize the complexity of modal logics (see [19] for a formal definition). Unlike the nesting depth of a formula, which is the maximal number of nested quantifiers it contains, the block nesting depth counts as one blocks of consecutive quantifiers of the same polarity (existential or universal).

Definition 23. The *block nesting depth* $\mathrm{bnd}(\varphi)$ *of a* $\mathsf{BQCTL}^*[\mathcal{F}]$ *formula is defined inductively as follows*:

$$\mathrm{bnd}(p) = 0$$
$$\mathrm{bnd}(\exists p.\, \varphi) = \begin{cases} \mathrm{bnd}(\varphi) & \text{if } \varphi = \exists q.\, \varphi' \text{ for some } q \in \mathrm{AP} \\ 1 + \mathrm{bnd}(\varphi) & \text{otherwise} \end{cases}$$
$$\mathrm{bnd}(f(\varphi_1, \cdots, \varphi_k)) = \max\{\mathrm{bnd}(\varphi_i) \mid 1 \leq i \leq k\}$$
$$\mathrm{bnd}(\mathbf{E}\psi) = \mathrm{bnd}(\psi)$$
$$\mathrm{bnd}(\mathbf{X}\psi) = \mathrm{bnd}(\psi)$$
$$\mathrm{bnd}(\psi \mathbf{U} \psi') = \max\{\mathrm{bnd}(\psi), \mathrm{bnd}(\psi')\}$$

This notion resembles the usual notion of alternation depth, but with block nesting depth, nested quantifiers of the same polarity are all counted if they are not contiguous, as seen in the last example below:

Example 7. $\mathrm{bnd}(p \vee q) = 0$, $\mathrm{bnd}(\exists p.\, p \vee q) = 1$, $\mathrm{bnd}(\exists p.\, \exists q.\, p \vee q) = 1$, $\mathrm{bnd}(\exists p.\, \exists q.\, \forall r.\, p \vee q \vee r) = 2$, and $\mathrm{bnd}(\exists p.\, p \wedge \mathbf{EF}\exists q.\, q) = 2$.

The following result was established in [19], for the tree semantics of $\mathsf{BQCTL}^*[\mathcal{F}]$:

Proposition 8 *([19]). Let $\mathcal{V} \subset [-1,1]$ be a finite set of values such that $\{-1,1\} \subseteq \mathcal{V}$, and let D be a finite set of directions. For every $\mathsf{BQCTL}^*[\mathcal{F}]$ state formula φ and predicate $P \subseteq (-1,1]$, one can construct an alternating parity tree automaton (APT) $\mathcal{A}_\varphi^{\mathcal{V},P}$ over $(\mathcal{V}^{\mathrm{AP}} \cup \{\bullet\})$-trees such that for every $\mathcal{V}^{\mathrm{AP}}$-labeled D-tree t,*

$\mathcal{A}_\varphi^{\mathcal{V},P}$ *accepts* $\bar{t}$ *if and only if* $[\![\varphi]\!]^t \in P$.

The APT $\mathcal{A}_\varphi^{\mathcal{V},P}$ is of size at most $(\mathrm{bnd}(\varphi)+1)$*-exponential in* $|\varphi|$*, and its index is at most* $\mathrm{bnd}(\varphi)$*-exponential in* $|\varphi|$.

We recall that $\bar{t}$ is the complete tree corresponding to t (see Section 7.1).
From this result, we obtain that satisfiability of $\mathsf{BQCTL}^*[\mathcal{F}]$ formulas is decidable.

Theorem 4. *The satisfiability problem for $\mathsf{BQCTL}^*[\mathcal{F}]$ is nonelementary decidable. For formulas of block nesting depth at most k, the problem is $(k+2)$-*EXPTIME*-complete.*

Proof. The lower bounds are inherited from the satisfiability problem for QCTL^* [113]. The reduction is direct with threshold $\vartheta = 1$.
For membership, let Φ be a $\mathsf{BQCTL}^*[\mathcal{F}]$ formula and $\vartheta > -1$ a threshold. Let $\widetilde{\Phi}$ be the corresponding formula on structures with binary branching, and let $\varphi_2 = \neg p_{int} \wedge \mathbf{AGF}\neg p_{int}$.

Lemma 4. *There exists a (finite) wKS $\mathcal{S}$ such that $[\![\Phi]\!]^{\mathcal{S}} \geq \vartheta$ if and only if there exists a regular binary tree t such that $[\![\mathbf{AG}\mathit{Bool}(p_{int}) \wedge \varphi_2 \wedge \widetilde{\Phi}]\!]^t \geq \vartheta$.*

Proof. If there exists $\mathcal{S}$ such that $[\![\Phi]\!]^{\mathcal{S}} \geq \vartheta$, then by Lemma 3 $[\![\widetilde{\Phi}]\!]^{\widetilde{\mathcal{S}}} \geq \vartheta$. Also by construction we have $[\![\mathbf{AG}\mathit{Bool}(p_{int})]\!]^{\widetilde{\mathcal{S}}} = 1$ and $[\![\varphi_2]\!]^{\widetilde{\mathcal{S}}} = 1$, so that $[\![\mathbf{AG}\mathit{Bool}(p_{int}) \wedge \varphi_2 \wedge \widetilde{\Phi}]\!]^t \geq \vartheta$, where $t = t_{\mathcal{S}}$ is the regular tree obtained by unfolding $\widetilde{\mathcal{S}}$.
For the other direction assume there exists a regular binary tree t such that $[\![\mathbf{AG}\mathit{Bool}(p_{int}) \wedge \varphi_2 \wedge \widetilde{\Phi}]\!]^t \geq \vartheta$. There exists a wKS $\mathcal{S}'$ with binary branching of which t is the unfolding. Since $\vartheta > -1$ and $\mathbf{AG}\mathit{Bool}(p_{int}) \wedge \varphi_2$ can only take values 1 or -1, we have that $[\![\mathbf{AG}\mathit{Bool}(p_{int}) \wedge \varphi_2]\!]^{\mathcal{S}'} = 1$. It follows that $[\![\widetilde{\Phi}]\!]^{\mathcal{S}'} \geq \vartheta$ and that $\mathcal{S}' = \widetilde{\mathcal{S}}$ for some $\mathcal{S}$. By Lemma 3 we have that $[\![\Phi]\!]^{\mathcal{S}} = [\![\widetilde{\Phi}]\!]^{\widetilde{\mathcal{S}}} \geq \vartheta$. □

Now from Proposition 8 with $P = [\vartheta, 1]$ and $\varphi = \mathbf{AG}\mathit{Bool}(p_{int}) \wedge \varphi_2 \wedge \widetilde{\Phi}$ we can build an automaton $\mathcal{A}$ over $(\mathcal{V}^{\mathrm{AP}} \cup \{\bullet\})$-labeled binary trees such that for every $\mathcal{V}^{\mathrm{AP}}$-labeled binary tree t, $\mathcal{A}$ accepts $\bar{t}$ if and only if $[\![\Phi]\!]^t \geq \vartheta$, with a number of states at most $(\mathrm{bnd}(\varphi)+1)$-exponential in $|\varphi|$ and index at most $\mathrm{bnd}(\varphi)$-exponential in $|\varphi|$. From Lemma 4 and the fact that every regular tree language contains a regular tree it follows that $\mathcal{A}$ is nonempty if and only if there exists a finite wKS $\mathcal{S}$ such that $[\![\Phi]\!]^t \geq \vartheta$. The emptiness of $\mathcal{A}$ can be tested in time exponential in the index and the number of states [116]. Since for formulas of block nesting depth at most k the number of states is at most $(k+1)$-exponential and the index at most k-exponential in $|\varphi|$, the overall complexity is $(k+2)$-exponential in time in the size of φ. We finish by observing that $|\varphi| = O(|\widetilde{\Phi}|) = O(|\Phi|)$. □

7.4. Decidable cases for $\mathsf{SL}[\mathcal{F}]$ satisfiability

In the qualitative setting, satisfiability of ATL with strategy context and SL are known to be decidable in two cases: when the number of possible actions is bounded, and when agents play in turns. With respect to bounded actions, we call Ac- wCGS a wCGS whose set of actions is Ac. Concerning turn-based systems, intuitively a wCGS is turn-based when in every state there is a unique agent who can determine the next state by the choice of its action. Formally, a wCGS $\mathcal{G} = (\mathrm{Ac}, V, \delta, \ell, V_i)$ is *turn-based* if for every state $v \in V$ there exists a unique agent a such that for every successor $v' \in \{\delta(v, \boldsymbol{c}) \mid \boldsymbol{c} \in \mathrm{Ac}^{\mathrm{Ag}}\}$ there is an action c such that $\delta(v, \boldsymbol{c}) = v'$ for all joint actions $\boldsymbol{c}$ where $\boldsymbol{c}_a = c$. We call *owner* of a state the agent that can choose the successor.
We show that, as for SL, the satisfiability problem for $\mathsf{SL}[\mathcal{F}]$ is decidable when the number of actions is bounded (*a priori*) or systems are turn-based, if in addition we restrict to models in which atomic propositions take values in a given finite set of possible values. Given a set $\mathcal{V} \subseteq [-1,1]$ of possible values with $\{-1,1\} \subseteq \mathcal{V}$, we call *$\mathcal{V}$-weighted* wCGS a wCGS whose weight function takes values in $\mathcal{V}$. The notion of block nesting depth for $\mathsf{SL}[\mathcal{F}]$ is defined similarly to $\mathsf{BQCTL}^*[\mathcal{F}]$, counting blocks of strategy quantifiers instead of blocks of quantifiers on propositions.

Definition 24. Given a finite set $\mathcal{V} \subset [-1,1]$ such that $\{-1,1\} \subseteq \mathcal{V}$, the *$\mathcal{V}$-satisfiability problem* for $\mathsf{SL}[\mathcal{F}]$ is the restriction of the satisfiability problem to $\mathcal{V}$-weighted wCGS.

We get the following key result characterizing the complexity for synthesizing mechanisms.[4]

Theorem 5. *Let $\mathcal{V}$ be a finite set of values and* Ac *a finite set of actions. Then $\mathcal{V}$-satisfiability of* SL[$\mathcal{F}$] *over* Ac- *wCGS is decidable in time $(k+2)$-exponential in the size of the formula, where k is the block nesting depth of the formula.*

Proof. We use a slight modification of the proof in [112] which shows how, when the number of actions is bounded, one can reduce the satisfiability problem for SL to that for QCTL*.

Let $\mathrm{Ac} = \{c_1, ..., c_n\}$ and $\mathrm{Ag} = \{a_1, ..., a_m\}$. First, consider the formula

$$\Phi_{\mathrm{Bool}} = \mathbf{AG} \bigwedge_{a \in \mathrm{Ag}, c \in \mathrm{Ac}} Bool(mov_a^c)$$

This formula has a value of 1 in a tree if propositions mov_a^c only take Boolean values in all nodes of the tree, otherwise, it has a value of -1. Define also

$$\Phi_{edge} = \mathbf{AG}\left[\left(\bigwedge_{c \in \mathrm{Ac}^{\mathrm{Ag}}} \mathbf{E}_1 \mathbf{X} mov_c \right) \wedge \mathbf{AX}\left(\bigvee_{c \in \mathrm{Ac}^{\mathrm{Ag}}} mov_c \right) \right]$$

where mov_c stands for $\bigwedge_{a \in \mathrm{Ag}} (mov_a^{c_a} \wedge \bigwedge_{c' \neq c_a} \neg mov_a^{c'})$, and

$$\mathbf{E}_1 \mathbf{X} \varphi = \mathbf{EX}\varphi \wedge \forall p. (\mathbf{EX}(\varphi \wedge p) \rightarrow \mathbf{AX}(\varphi \rightarrow p))$$

expresses the existence of a unique successor that satisfies φ (where φ is a formula that only takes Boolean values). When all propositions mov_a^c have Boolean value, Φ_{edge} can only take value 1 or -1 in a tree, and this value is 1 if and only if the tree is the unfolding of some wCGS where in every state, the proposition mov_a^c has value 1 if agent a played action c in the last move, and -1 otherwise.

Consider now the following translation from SL[$\mathcal{F}$] to BQCTL*[$\mathcal{F} \cup \{Bool\}$]. For every SL[$\mathcal{F}$] formula φ and partial function $V : \mathrm{Ag} \rightarrow \mathrm{Var}$, we inductively define the BQCTL*[$\mathcal{F}$] formula $\widehat{\varphi}^V$. The translation is identical to the one in [112], but for completeness we give the cases for strategy quantification and binding, as well as temporal operators.

$$\widehat{\exists s. \varphi}^V = \exists p_s^{c_1} ... \exists p_s^{c_n} . \mathbf{AG}\left(\bigvee_{c \in \mathrm{Ac}} p_s^c \wedge \bigwedge_{c' \neq c} \neg p_s^{c'} \right) \wedge \widehat{\varphi}^V$$

$$\widehat{(a,s)\varphi}^V = \widehat{\varphi}^{V[a \mapsto s]}$$

$$\widehat{\mathbf{X}\varphi}^V = \mathbf{A}\left[\psi_{out}^V \rightarrow (\mathbf{X}\widehat{\varphi}^V) \right]$$

$$\widehat{\varphi \mathbf{U} \psi}^V = \mathbf{A}\left[\psi_{out}^V \rightarrow (\widehat{\varphi}^V \mathbf{U} \widehat{\psi}^V) \right]$$

where $\psi_{out}^V = \mathbf{G} \bigwedge_{a \in \mathrm{Ag}} \bigwedge_{c \in \mathrm{Ac}} (p_{V(a)}^c \rightarrow \mathbf{X} mov_a^c)$.

Finally we let $\widehat{\Phi} = \Phi_{\mathrm{Bool}} \wedge \Phi_{edge} \wedge \widehat{\Phi}^{\emptyset}$. We have that for every finite wCGS $\mathcal{G}$ there exists a finite wKS S such that $[\![\Phi]\!]^{\mathcal{G}} = [\![\widehat{\Phi}]\!]^S$, where S is obtained by partially unfolding $\mathcal{G}$ to place propositions mov_a^c appropriately. Conversely, for every finite S such that $[\![\widehat{\Phi}]\!]^S > -1$ we have that $[\![\Phi_{\mathrm{Bool}} \wedge \Phi_{edge}]\!]^S = 1$ and thus S encodes a finite wCGS $\mathcal{G}$ such that $[\![\Phi]\!]^{\mathcal{G}} = [\![\widehat{\Phi}]\!]^S$. Observe that $\widehat{\Phi}$ is of size $O(|\Phi|)$, so that we obtain the announced complexity. □

Theorem 6. *Let $\mathcal{V}$ be a finite set of values. Then $\mathcal{V}$-satisfiability of* SL[$\mathcal{F}$] *over turn-based wCGS is decidable in time $(k+2)$-exponential in the size of the formula, where k is the block nesting depth of the formula.*

Proof. Again, we adapt the proof from [112]. We assume that in a turn-based wCGS, each state is labeled with a proposition $turn_a$ where a is the owner of the state. We use formula

$$\varphi_{tb} = \mathbf{AG}\left(\bigwedge_{a \in \mathrm{Ag}} Bool(turn_a) \wedge \bigvee_{a \in \mathrm{Ag}} (turn_a \wedge \bigwedge_{a' \neq a} \neg turn_{a'}) \right)$$

that has Boolean value, and has value 1 if and only if propositions $turn_a$ only take Boolean values (-1 or 1), and exactly one of them has value 1 in each node.

We now define the following translation from SL[$\mathcal{F}$] to BQCTL*[$\mathcal{F}$], where $V : \mathrm{Ag} \rightarrow \mathrm{Var}$ is a partial function.

$$\widehat{\exists s. \varphi}^V = \exists mov_s . \mathbf{AG}(\mathbf{E}_1 \mathbf{X} mov_s \wedge \widehat{\varphi}^V)$$

$$\widehat{(a,s)\varphi}^V = \widehat{\varphi}^{V[a \mapsto s]}$$

$$\widehat{\mathbf{X}\varphi}^V = \mathbf{A}\left[\psi_{out}^V \rightarrow (\mathbf{X}\widehat{\varphi}^V) \right]$$

[4] We point out that there was an error in the conference version of this paper, where we claimed the complexity to be $(k+1)$-ExpTime, instead of $(k+2)$-ExpTime.

$$\widehat{\varphi \mathbf{U} \psi}^{V} = \mathbf{A}\left[\psi_{out}^{V} \rightarrow (\widehat{\varphi}^{V} \mathbf{U} \widehat{\psi}^{V})\right]$$

where $\psi_{out}^{V} = \mathbf{G}(\bigwedge_{a \in \mathrm{Ag}}(turn_a \rightarrow \mathbf{X} mov_a))$. Finally, formula Φ has value greater than ϑ in some turn-based wCGS if and only if formula $\varphi_{tb} \wedge \widehat{\Phi}^{\vartheta}$ has value greater than ϑ in some tree. Again the latter formula is of size $O(|\Phi|)$, and the result on complexity follows. □

7.5. The case of direct mechanisms

We now establish a last decidable case and show that, as for model checking, the satisfiability problem for $\mathsf{SL}[\mathcal{F}]$ is much easier on direct mechanisms.

Indeed, since a wCGS that models a direct mechanism reaches a terminal state in one step from the initial state (see Definition 9), we can restrict attention to models of exponential size, containing one initial state plus one state for each possible valuation over atomic propositions in the formula φ for which a mechanism is to be synthesized. As a result, synthesis can be solved by a nondeterministic Turing machine that first guesses a wCGS of exponential size, and then checks whether it satisfies the specification φ. As we saw this can be done in PSPACE (Theorem 2), but because of the exponential size of the model that we have guessed, we obtain an overall procedure that is in nondeterministic exponential space. Since, by Savitch's theorem, NEXPSPACE = EXPSPACE, we obtain the following result:

Theorem 7. *The satisfiability problem for* $\mathsf{SL}[\mathcal{F}]$ *on direct mechanisms is decidable in exponential space.*

7.6. Automated synthesis of optimal mechanism

We describe how we can use our algorithm for $\mathsf{SL}[\mathcal{F}]$ satisfiability to synthesize mechanisms that optimally satisfy the specification, in the sense that they achieve the best possible satisfaction value for the specification. First, we observe that the algorithms developed in the previous section for the satisfiability problem of $\mathsf{SL}[\mathcal{F}]$ in the case of bounded actions or turn-based systems can be tweaked to actually return a satisfying wCGS when one exists. Indeed classic algorithms to solve emptiness of parity tree automata can produce a witness regular tree (in the form of a finite graph) accepted by the automaton (see [117] for the procedure), and from such a tree in our setting we can infer a witness wCGS.

Second, it is proved in [19] that given a finite set $\mathcal{V}$ of possible values for atomic propositions and a formula $\varphi \in \mathsf{SL}[\mathcal{F}]$ there is only a finite number of possible satisfaction values φ can take in any wCGS, and we can compute an over-approximation of this set.

Lemma 5 *([19]). Let* $\mathcal{V} \subset [-1, 1]$ *be a finite set of values with* $\{-1, 1\} \subseteq \mathcal{V}$ *and let* φ *be an* $\mathsf{SL}[\mathcal{F}]$ *sentence. The set* $\mathrm{Val}_{\varphi,\mathcal{V}} = \{[\![\varphi]\!]^{\mathcal{G}} \mid \mathcal{G}$ *is a* $\mathcal{V}$*-weighted wCGS*$\}$ *is finite, and one can compute a set* $\widetilde{\mathrm{Val}}_{\varphi,\mathcal{V}}$ *of size at most* $|\mathcal{V}|^{|\varphi|}$ *such that* $\mathrm{Val}_{\varphi,\mathcal{V}} \subseteq \widetilde{\mathrm{Val}}_{\varphi,\mathcal{V}}$.

Algorithm 1: Optimal mechanism synthesis.

```
Data: A SL[F] specification Φ and a set of possible values for atomic propositions V
Result: A wCGS G such that [[Φ]]^G is maximal
Compute Val~_{Φ,V};
Let ν_1, ..., ν_n be a decreasing enumeration of Val~_{Φ,V};
for i = 1...n do
    Solve V-satisfiability for Φ and ϑ = ν_i;
    if there exists G such that [[Φ]]^G ≥ ν_i then
        return G;
    end
end
```

Consider Algorithm 1. This algorithm synthesizes a wCGS that maximizes the satisfaction value of the given $\mathsf{SL}[\mathcal{F}]$ specification, in all cases where the satisfiability problem for $\mathsf{SL}[\mathcal{F}]$ can be solved and a witness produced. In particular, it works in the case of bounded actions and the case of turn-based systems. To see that the algorithm is correct, first consider that since the set $\mathrm{Val}_{\varphi,\mathcal{V}}$ of possible values for Φ is finite (Lemma 5) it has a maximal value $\nu_{\max}$ which is attained on some wCGS. Still by Lemma 5, this value $\nu_{\max}$ belongs to the over-approximation $\widetilde{\mathrm{Val}}_{\varphi,\mathcal{V}}$. Let k be such that $\nu_k = \nu_{\max}$, with $\nu_1, ..., \nu_n$ a decreasing enumeration of $\widetilde{\mathrm{Val}}_{\Phi,\mathcal{V}}$, and consider iteration i of the for loop with $i < k$. Since $\nu_1, ..., \nu_n$ is decreasing we have that $\nu_i > \nu_k = \nu_{\max}$, hence ν_i does not belong to $\mathrm{Val}_{\varphi,\mathcal{V}}$. As a result, the satisfiability test for Φ with $\vartheta = \nu_i$ is negative, and we move to the next iteration. When we reach iteration $i = k$, we have $\nu_i = \nu_k = \nu_{\max}$ so by definition of $\nu_{\max}$ the satisfiability problem for Φ with threshold ν_i has a positive answer. The procedure to test for satisfiability returns some finite wCGS $\mathcal{G}_{\max}$ such that $[\![\Phi]\!]^{\mathcal{G}_{\max}} \geq \nu_{\max}$, which is the result returned by Algorithm 1. Since no wCGS achieves a satisfaction value higher than $\nu_{\max}$, we have that $[\![\Phi]\!]^{\mathcal{G}_{\max}} = \nu_{\max}$.

We now show how this can be used to solve automated mechanism design.

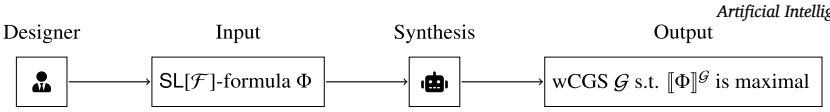


Fig. 4. Synthesis of a mechanism from a specification expressed as a SL[$\mathcal{F}$]-formula. The output is a mechanism represented as a wCGS that maximizes the satisfaction value of the specification.

8. Synthesis for mechanism design

This section shows how to use SL[$\mathcal{F}$] synthesis for mechanism design. As shown in Section 5.4, SL[$\mathcal{F}$] can express a variety of important notions in mechanism design, such as strategyproofness, individual rationality, efficiency, budget-balance, Pareto optimality, and different kinds of game-theoretic equilibria. Thus, these mechanisms' properties can be included in the specification for synthesizing an optimal wCGS with Algorithm 1. The overview of the synthesis approach is illustrated in Fig. 4.

Similar to Section 5.2, we start by representing mechanisms with wCGS. The definition here is simplified since we restrict our attention to wCGS, which does not encode imperfect information. This is because our goal is not to synthesize agents' knowledge, but rather to find a mechanism that satisfies the specification for any possible type profiles. Differently from the previous definition, the real type of the agents is not encoded in the states. In the former case, multiple initial states were defined to handle the possible worlds the agents consider possible (based on the indistinguishable relations). This is no longer necessary when not modeling the agents' knowledge.

Definition 25. [Mechanisms as a wCGS] Let AP $\supseteq \{choice, payment_a, terminal : a \in \mathrm{Ag}\}$, where *choice* and $payment_a$ denote respectively the choice elected by the mechanism, and the payment of agent a. The proposition *terminal* specifies whether a state is terminal. A *mechanism* $\mathcal{G}$ is a wCGS over the atomic propositions AP that satisfies the following:

(i) every play eventually reaches a *terminal state*, i.e., a sink;
(ii) in all non-terminal states, *terminal* has value -1.

We now use these formulas to illustrate our approach to mechanism synthesis. Again, we assume an order over the agents $\prec_{\mathrm{Ag}}$ to resolve ties, and we let $order_a \in [-1, 1]$ be a normalized constant for each agent a representing her priority in $\prec_{\mathrm{Ag}}$. More precisely, it is a constant value such that $order_a > order_b$ if $a \prec_{\mathrm{Ag}} b$, for any two distinct agents a and b. Given an agent a, we let $wins_a \in (-1, 1]$ be a constant value denoting the choice in which a is the winner, with $wins_a \neq wins_b$ for any $b \neq a$. We consider the choice set $\mathcal{X} = \{wins_a : a \in \mathrm{Ag}\} \cup \{-1\}$, where -1 specifies the case where there is no winner at the end of the game. In the examples we let each valuation function v_a be defined as $\mathsf{v}_a(\theta_a, x) = \theta_a$ if $x = wins_a$, and $\mathsf{v}_a(\theta_a, x) = 0$ otherwise. That is, an agent's valuation depends only on her type and whether she won. We assume $wins_a$ is a 0-ary function in $\mathcal{F}$.

8.1. Action-bounded mechanisms

Action-bounded mechanisms are of great interest since the amount of resources available is often limited. For instance, the actions in a market could consist of bids representing discrete monetary values bounded by the participants' budget. Another example is traditional elections, where the action set is usually formed by abstention and the preference for a specific candidate. Notice that this is also the case for direct mechanisms represented as weighted concurrent game structures (see Definition 9).

In this section, we illustrate the synthesis problem with the action-bounded restriction by considering rules based on the Japanese auction.

Example 8. The Japanese auction is an ascending protocol in which the price is repeatedly raised by the auctioneer until only one bidder remains. The remaining bidder wins the item at the final price [118]. Let us fix a price increment $\mathsf{inc} > 0$. There are only two possible actions, accept (*acc*) or decline (*dec*), so that the set $\mathrm{Ac} = \{acc, dec\}$ is indeed bounded. Furthermore, we let $\mathrm{AP} = \{price, sold, initial, choice, bid_a, payment_a, terminal : a \in \mathrm{Ag}\}$, where *price* denotes the current price, *initial* denotes whether the state is the initial one, *sold* specifies whether the item was sold, and bid_a specifies whether a is an active bidder. Let $\mathcal{V}_{jap} \subset [-1, 1]$ be a finite set of values containing the possible values for the atomic propositions, that is, it contains the possible types for the agents (that is, the elements of Θ_a for each agent a), the constants $wins_a$ for each $a \in \mathrm{Ag}$, the possible bid values, as well as the values -1 and 1.

The following SL[$\mathcal{F}$]-formulas are a description of the key rules of a Japanese auction. Algorithm 1 computes a mechanism that agrees with this specification. The meaning of Rules J1-J8 is intuitive. Similar rules for encoding auctions through a logic-based language can be seen in [52,53]. Rule J9 states that, in non-terminal states, each agent a always has a strategy to make the proposition bid_a true in the next state and a strategy to make it false. Rule J10 specifies that for all type profiles there should exist a *NE* whose outcome is *IR* and *EF*.

J1. $\mathbf{AG}\Big((initial \rightarrow price = 0 \wedge \neg terminal) \wedge (\mathbf{XG}\neg initial \wedge \mathbf{F}\ terminal)\Big)$

J2. $\mathbf{AG}\Big(sold \leftrightarrow choice \neq -1\Big)$

J3. $\mathbf{AG}\Big((\neg sold \wedge price + \mathsf{inc} \leq 1) \rightarrow (price + \mathsf{inc} = \mathbf{X}price \wedge \neg \mathbf{X}terminal)\Big)$

J4. $\mathbf{AG}\Big((sold \vee price + \mathsf{inc} > 1) \rightarrow (price = \mathbf{X}price \wedge \mathbf{X}terminal)\Big)$

J5. $\mathbf{AG}\Big(choice = wins_a \leftrightarrow bid_a \wedge \bigwedge_{b \neq a}(\neg bid_b \vee order_a > order_b)\Big)$

J6. $\mathbf{AG}\Big(choice = -1 \leftrightarrow \neg\big(\bigvee_{a \in \mathrm{Ag}}(bid_a \wedge \bigwedge_{b \neq a}(\neg bid_a \vee order_a > order_b))\big)\Big)$

J7. $\mathbf{AG}\Big(\bigwedge_{a \in \mathrm{Ag}}(choice = wins_a \rightarrow payment_a = price)\Big)$

J8. $\mathbf{AG}\Big(\bigwedge_{a \in \mathrm{Ag}}(choice \neq wins_a \rightarrow payment_a = 0)\Big)$

J9. $\mathbf{AG}\Big(\neg terminal \rightarrow \bigwedge_{a \in \mathrm{Ag}}\big(\exists s_a \forall \boldsymbol{s}_{-a}(a, s_a)(\mathrm{Ag}_{-a}, \boldsymbol{s}_{-a})\, \mathbf{X}\, bid_a \wedge \exists s_a \forall \boldsymbol{s}_{-a}(a, s_a)(\mathrm{Ag}_{-a}, \boldsymbol{s}_{-a})\, \mathbf{X}\neg bid_a\big)\Big)$

J10. $\bigwedge_{\theta \in \boldsymbol{\Theta}} \exists \boldsymbol{s}.\, NE(\boldsymbol{s}, \boldsymbol{\theta}) \wedge \mathbf{F}\big(terminal \wedge IR(\boldsymbol{\theta}) \wedge EF(\boldsymbol{\theta})\big)$

We denote by Σ_{jpn} the conjunction of Rules J1-J10. Algorithm 1 constructs a wCGS that maximizes the satisfaction value of Σ_{jpn}. We show that this value is 1, meaning that there exists a mechanism that is individually rational and efficient for some Nash equilibrium, for all type profiles.

Notice that the possible types are used as constants in the formulas, but the actual type of an agent is not a part of the mechanism description.

Proposition 9. *There exists a wCGS-mechanism $\mathcal{G}_{jpn}$ such that $[\![\Sigma_{jpn}]\!]^{\mathcal{G}_{jpn}, R} = 1$.*

Proof. We construct a wCGS-mechanism $\mathcal{G}_{jpn}$ in which Σ_{jpn} has the satisfaction value equal to one, for any state and assignment.

Let $\mathcal{G}_{jpn} = (\mathrm{Ac}, V, \delta, \ell, V_\iota)$ over $\mathrm{AP} = \{price, sold, initial, wins_a, bid_a, payment_a, terminal : a \in \mathrm{Ag}\}$, where:

- V consists of states of the form $\langle p, winner, (b_a)_{a \in \mathrm{Ag}}, s\rangle$ with $p \in \{0 + x \cdot \mathsf{inc} : 0 \leq x \leq \frac{1}{\mathsf{inc}}\}$ denoting the current price, $winner \in \mathrm{Ag} \cup \{none\}$ denoting the current winner, $b_a \in \{-1, 1\}$ specifying whether a is an active bidder, and $s \in \{-1, 1\}$ specifying whether the item was sold.
- In an initial state, the price starts at 0, the item is unsold and there is no winner or active bidder. That is, the initial states are $V_\iota = \{\langle 0, none, (-1, \ldots, -1), -1\rangle\}$.
- If the item is unsold, the transition function increases the price p as long as it is under 1 and there is at least one active bidder. If there is only one bidder, she is assigned as the winner. When the item is sold or the price reaches 1, the atomic propositions remain unchanged after the transition. Formally, for each state $v = \langle p, winner, (b_a)_{a \in \mathrm{Ag}}, s\rangle$ and joint action $\boldsymbol{c} = (c_a)_{a \in \mathrm{Ag}}$, transition $\delta(v, \boldsymbol{c})$ is defined as follows:
  - If $s = -1$ and $p + \mathsf{inc} \leq 1$, $\delta(v, \boldsymbol{c}) = \langle p', winner', (b'_a)_{a \in \mathrm{Ag}}, s'\rangle$ where:

$$p' = p + \mathsf{inc}$$

$$winner' = \begin{cases} a & \text{if } c_a = acc \text{ and } c_b = dec \\ & \text{for all } b \neq a \\ none & \text{otherwise} \end{cases}$$

$$b'_a = \begin{cases} 1 & \text{if } c_a = acc \\ -1 & \text{otherwise} \end{cases}$$

$$s' = \begin{cases} 1 & \text{if } winner' \neq none \\ -1 & \text{otherwise} \end{cases}$$

  - Otherwise, $\delta(v, \boldsymbol{c}) = v$.
- For each $v = \langle p, winner, (b_a)_{a \in \mathrm{Ag}}, s\rangle$ and each $a \in \mathrm{Ag}$, the weight function is defined as follows:

(i) $\ell(v, price) = p$;
(ii) $\ell(v, choice) = wins_{winner}$ iff $winner \neq none$ and $\ell(v, wins_a) = -1$ otherwise;
(iii) $\ell(v, bid_a) = b_a$;
(iv) $\ell(v, sold) = s$;
(v) $\ell(v, initial_a) = 1$ iff $v \in V_\iota$ and $\ell(v, initial_a) = -1$ otherwise;
(vi) $\ell(v, payment_a) = p$ iff $wins_a = winner$ and $\ell(v, payment_a) = 0$ otherwise;
(vii) $\ell(v, terminal_a) = 1$ iff $s = -1$ and $p + \mathsf{inc} \leq 1$ and $\ell(v, terminal_a) = -1$ otherwise.

It is straightforward to see that Rules J1-J9 are true in any state $v \in V$ and assignment $\mathcal{A}$. For Rule J10, let $\pi \in \text{Hist}$ be a history of length i. We show that there exists a strategy profile $\boldsymbol{\sigma}$ such that $[\![\exists \boldsymbol{s}.\, NE(\boldsymbol{s}, \boldsymbol{\theta}) \wedge \mathbf{F}(terminal \wedge (IR(\boldsymbol{\theta}) \wedge EF(\boldsymbol{\theta})))]\!]^{\mathcal{G}_{jpn}}(\pi) = 1$. For each agent a, let σ_a be a strategy defined as follows: $\sigma_a(\text{last}(\pi)) = acc$ iff $\mathsf{v}_a(wins_a, \theta_a) \geq \ell(\text{last}(\pi), price) + \text{inc}$; and $\sigma_a(v) = dec$ otherwise.

To see that this strategy is a Nash equilibrium, let $\mathcal{A}$ be an assignment such that $\mathcal{A}(a) = \sigma_a$ and $\mathcal{A}'$ be an assignment equal to $\mathcal{A}$, except by the strategy associated with agent a (that is, $\mathcal{A}'(a) \neq \mathcal{A}(a)$).

The definition of $\mathcal{G}_{jpn}$ ensures the agents' utilities can be different from zero only in terminal states. Furthermore, as this value depends only on the immediately preceding state and action, we consider the agent's utility after each transition. Let $\boldsymbol{\theta} = (\theta_a)_{a \in \text{Ag}}$ be a type profile in $\boldsymbol{\Theta}$. If $\mathcal{A}'(a)(\text{last}(\pi)) = acc$ when $\mathsf{v}_a(wins_a, \theta_a) < \ell(\text{last}(\pi), price) + \text{inc}$, then $[\![\mathbf{X} util_a(\theta_a)]\!]^{\mathcal{G}_{jpn}}_{\mathcal{A}'}(\pi) \leq 0 \leq [\![\mathbf{X} util_a(\theta_a)]\!]^{\mathcal{G}_{jpn}}_{\mathcal{A}}(\pi)$, since by doing the action acc the agent would be risking to get the item at a higher price than her valuation. Assume $\mathcal{A}'(a)(\pi) = acc$ and $\mathsf{v}_a(wins_a, \theta_a) < \ell(\text{last}(\pi), price) + \text{inc}$. By the definition of σ_a, $\mathcal{A}(a)(\text{last}(\pi)) = dec$ then $[\![\mathbf{X} util_a(\theta_a)]\!]^{\mathcal{G}_{jpn}}_{\mathcal{A}'}(\pi) \leq 0$ and $[\![\mathbf{X} util_a(\theta_a)]\!]^{\mathcal{G}_{jpn}}_{\mathcal{A}}(\pi) = 0$, since by doing the action acc the agent would be risking to get the item at a higher price than her valuation. Now consider the case in which $\mathcal{A}'(a)(\text{last}(\pi)) = dec$ and $\mathsf{v}_a(wins_a, \theta_a) \geq \ell(\text{last}(\pi), price) + \text{inc}$. By definition, $\mathcal{A}(a)(\text{last}(\pi)) = acc$. Thus, $[\![\mathbf{X} util_a(\theta_a)]\!]^{\mathcal{G}_{jpn}}_{\mathcal{A}'}(\pi) = 0$, while $[\![\mathbf{X} util_a(\theta_a)]\!]^{\mathcal{G}_{jpn}}_{\mathcal{A}}(\pi) \geq 0$, since by doing the action dec the agent would not win the item, whereas doing the action acc could lead to winning at a price lower than her valuation. The other cases are proven similarly. Since no agent can improve her utility through a unilateral change of strategy, the strategy profile $\boldsymbol{\sigma} = (\sigma_a)_{a \in \text{Ag}}$ is a Nash equilibrium. Furthermore, no agent a following the strategy σ_a can have a negative utility, since she chooses the action dec whenever the price becomes higher than her valuation. From Rule J1, we have $[\![\mathbf{F} terminal]\!]^{\mathcal{G}_{jpn}}(\pi)$. Finally, since the agents have distinct valuations, following $\boldsymbol{\sigma}$ would lead to a state in which the item is assigned to the agent who values it the most. If no agent had a valuation for winning the item greater than 0, no winner would be assigned and the social welfare would be 0. Thus, $[\![\exists \boldsymbol{s}.\, NE(\boldsymbol{s}, \boldsymbol{\theta}) \wedge \mathbf{F}(terminal \wedge (IR(\boldsymbol{\theta}) \wedge EF(\boldsymbol{\theta})))]\!]^{\mathcal{G}_{jpn}}(\pi) = 1$. □

Algorithm 1 produces an optimal mechanism, which may be different from the one above.

Corollary 2. *Given $\mathcal{G}_{jpn}$, and $\mathcal{V}_{jap} \subset [-1, 1]$, Algorithm 1 returns a wCGS-mechanism $\mathcal{G}_{jpn}$ such that $[\![\Sigma_{jpn}]\!]^{\mathcal{G}_{jpn}, R} = 1$.*

8.2. Turn-based mechanisms

In a number of mechanisms for preference aggregation, agents play in turns. For instance, picking sequences are a well-known mechanism for fair division in which agents alternate into choosing items [119]. Hereby, we exemplify the synthesis of a turned-based mechanism. In this example, we also show how Algorithm 1 can be used to maximize social welfare.

Example 9. The English auction is an ascending auction in which the participants are allowed to outbid the last bidder by proposing the highest price. The auction ends when no agent is willing to raise the last bid. The highest bidder wins the item at her proposed price [3].

In this specification, let us consider two agents, with $\text{Ag} = \{a_1, a_2\}$. We let $-a$ denote the opponent of $a \in \text{Ag}$. Furthermore, we let $\text{AP} = \{price, initial, choice_a, bid_a, turn_a, payment_a, terminal : a \in \text{Ag}\}$, where *price* denotes the current price, *initial* denotes whether the state is the initial one, $turn_a$ specifies whether it is a's turn and bid_a specifies the value of a's bid. Let $\mathcal{V}_{eng} \subset [-1, 1]$ be a finite set of values containing the possible values for the atomic propositions, defined analogously to $\mathcal{V}_{eng}$. We let *bound* be a constant denoting the second highest value in $\mathcal{V}_{eng}$.

The following $\mathsf{SL}[\mathcal{F}]$-formulae are a description of the rules of a two-player turned-based variant of the English auction.

E1. $\mathbf{AG}\Big(initial \rightarrow turn_{a_1} \wedge \neg turn_{a_2} \wedge \neg terminal \wedge \bigwedge_{a \in \text{Ag}} (bid_a = 0)\Big)$

E2. $\mathbf{AG}\Big(price = \max(bid_{a_1}, bid_{a_2})\Big)$

E3. $\mathbf{AG}\Big(\neg terminal \rightarrow \bigwedge_{a \in \text{Ag}} (turn_a \rightarrow \neg \mathbf{X} turn_a \wedge \neg turn_a \rightarrow \mathbf{X} turn_a)\Big)$

E4. $\mathbf{AG}\Big(\neg terminal \rightarrow \bigwedge_{a \in \text{Ag}} \big(bid_{-a} < \mathbf{X} bid_a \wedge turn_a \rightarrow \mathbf{X} choice = wins_a\big)\Big)$

E5. $\mathbf{AG}\Big(\neg terminal \rightarrow \bigwedge_{a \in \text{Ag}} \big(bid_{-a} \geq \mathbf{X} bid_a \wedge turn_a \rightarrow \mathbf{X}(choice = wins_{-a} \wedge terminal)\big)\Big)$

E6. $\mathbf{AG}\Big(\bigwedge_{a \in \text{Ag}} \big(bid_a = 1 \wedge turn_a \wedge \mathbf{X}(choice = wins_a \wedge terminal)\big)\Big)$

E7. $\mathbf{AG}\Big(\bigwedge_{a \in \text{Ag}} \big((\neg terminal \wedge bid_a \leq bound \wedge turn_a) \rightarrow (\exists s_a \forall \boldsymbol{s}_{-a}(a, s_a)(\text{Ag}_{-a}, \boldsymbol{s}_{-a}) bid_a = \mathbf{X} bid_a \wedge \exists s_a \forall \boldsymbol{s}_{-a}(a, s_a)(\text{Ag}_{-a}, \boldsymbol{s}_{-a}) bid_a < \mathbf{X} bid_a)\big)\Big)$

E8. $\exists \boldsymbol{s}.\, NE(\boldsymbol{s}, \boldsymbol{\theta}) \wedge \mathbf{F}\big(terminal \wedge \sum_{a \in \text{Ag}} \mathsf{v}_a(choice, \theta_a)\big)$

The value of formula E8 is the social welfare in the best *NE* for type profile $\boldsymbol{\theta}$. Rule E7 connects agents' actions to propositions, by stating that the agent can control the proposition representing her bid value (either by raising it or keeping it unchanged). Letting $\varphi_{eng}(\boldsymbol{\theta})$ be the conjunction of Rules E1-E8 for each $\boldsymbol{\theta} \in \boldsymbol{\Theta}$ and Σ_{eng} be the conjunction of $\bigwedge_{\boldsymbol{\theta} \in \boldsymbol{\Theta}} \varphi_{eng}(\boldsymbol{\theta})$ alongside with the payment rules in Σ_{jpn} (Rules J7-J8). We then have:

Proposition 10. *There exists a wCGS-mechanism $\mathcal{G}_{eng}$ such that $[\![\Sigma_{eng}(\boldsymbol{\theta})]\!]^{\mathcal{G}_{eng},R} = \max_{x\in\mathcal{X}} \sum_{a\in\mathrm{Ag}} \mathsf{v}_a(x,\theta_a)$ for each type profile $\boldsymbol{\theta}\in\boldsymbol{\Theta}$.*

Proof. We construct a wCGS-mechanism $\mathcal{G}_{eng}$ in which Σ_{eng} has the satisfaction value equal to one, for any history and assignment. Let $0 > \mathsf{inc} > 1$ be a constant value and the action set be defined as $\mathrm{Ac} = \{1 - x\cdot\mathsf{inc} : 0 \le x \le \frac{1}{\mathsf{inc}}\cup\{0\}$, where each action denotes the value the agent is willing to pay for the item. Let $\mathcal{G}_{eng} = (\mathrm{Ac}, V, \delta, \ell, V_\iota)$ over $\mathrm{AP} = \{price, initial, choice_a, bid_a, turn_a, payment_a, terminal : a\in\mathrm{Ag}\}$, where:

- V consists of states of the form $\langle (b_a)_{a\in\mathrm{Ag}}, own, winner, tr\rangle$ with $b_a\in\mathrm{Ac}$ denoting the value of a's bid and $own\in\mathrm{Ag}$ specifying the owner of the state and $tr\in\{-1,1\}$ denoting whether the state is terminal;
- In an initial state, the bids are 0, a_1 has the turn and it is not a terminal state. That is, the set of initial states is $V_\iota = \{\langle (0,0), a_1, a_1, -1\rangle\}$.
- For each state $v = \langle (b_a)_{a\in\mathrm{Ag}}, own, winner, tr\rangle$ and joint action $\boldsymbol{c} = (c_a)_{a\in\mathrm{Ag}}$, transition $\delta(v,\boldsymbol{c})$ is defined as follows:
  - If $tr = -1$, $\delta(v,\boldsymbol{c}) = \langle (b'_a)_{a\in\mathrm{Ag}}, own', winner, tr'\rangle$ where:

$$b'_a = \begin{cases} c_a & \text{if } own = a \\ b_a & \text{otherwise} \end{cases}$$

$$own' = \begin{cases} a & \text{if } own \neq ae \\ -a & \text{otherwise} \end{cases}$$

$$winner' = \begin{cases} a & \text{if } c_a > b_{-a} \\ -a & \text{otherwise} \end{cases}$$

$$tr' = \begin{cases} 1 & \text{if } \max(c_{own}, b_{-own}) = 1 \text{ or} \\ & c_{own} \le b_{-own} \\ -1 & \text{otherwise} \end{cases}$$

  - Otherwise, $\delta(v,\boldsymbol{c}) = v$.
- For each $v = \langle (b_a)_{a\in\mathrm{Ag}}, own, winner, tr\rangle$ and each $a\in\mathrm{Ag}$, the weight function is defined as follows:

(i) $\ell(v, price) = \max(b_{a_1}, b_{a_2})$;
(ii) $\ell(v, initial_a) = 1$ iff $v\in V_\iota$ and $\ell(v, initial_a) = -1$ otherwise;
(iii) $\ell(v, choice) = wins_{winner}$;
(iv) $\ell(v, bid_a) = b_a$;
(v) $\ell(v, turn_a) = 1$ iff $own = a$ and $\ell(v, turn_a) = -1$ otherwise;
(vi) $\ell(v, payment_a) = \ell(v, price)$ iff $wins_a = winner$ and $\ell(v, payment_a) = 0$ otherwise;
(vii) $\ell(v, terminal_a) = tr$.

It is routine to see that Rules E1-E7 and Rules J7-J8 have a satisfaction value 1 for any history in Hist and assignment $\mathcal{A}$. Let $\boldsymbol{\theta}\in\boldsymbol{\Theta}$. The proof for Rule E7 proceeds similarly to the proof for Rule J10 (Proposition 9), by noticing that the strategy profile in which the agents raise their bid the minimum possible amount (up to their valuation) in each turn is a Nash equilibrium. Furthermore, since this equilibrium is efficient, Rule E7 will have a satisfaction value equal to the maximum social welfare, that is $[\![\Sigma_{eng}(\boldsymbol{\theta})]\!]^{\mathcal{G}_{eng},R} = \max_{x\in\mathcal{X}} \sum_{a\in\mathrm{Ag}} \mathsf{v}_a(x,\theta_a)$. □

As a result, Algorithm 1 applied to $\Sigma_{eng}(\boldsymbol{\theta})$ returns a mechanism that satisfies all the rules E1-E8 (for each type profile) and J7-J8.

Corollary 3. *Given Σ_{eng}, and $\mathcal{V}_{eng}\subset[-1,1]$, Algorithm 1 returns a wCGS-mechanism $\mathcal{G}_{eng}$ such that $[\![\Sigma_{eng}]\!]^{\mathcal{G}_{eng},R} = 1$.*

Complexity The synthesis problem in these examples can be solved in 4-ExpTime (see Theorem 6). The complexity is dominated by Rules J9, J10, E7, and E8, which have block nesting depth 2. Without them, the complexity would be in 2-ExpTime. Most importantly, the complexity is only in the size of the formula, which is typically rather small.

9. Extensions

In Section 5.2, we show how to capture standard definitions and properties from mechanism design. Clearly, by a slight change in the formulas, the framework can be used to reason about less standard mechanisms, without computational cost (in terms of complexity for model-checking and satisfiability). For instance, the setting in which agents have dynamic preferences, that is, their types change over time, can be captured by dropping the requirement that types remain unchanged through transitions (Item 2 on Definition 25).

9.1. Infinite mechanisms

We now show how to consider infinite mechanisms, that is, mechanisms without terminal states. In this case, since there is no final decision, the utility of the agents and economical properties are considered throughout the execution of the game.

Definition 26 *(Infinite Mechanism as a wCGSii).* Let $\mathrm{AP} \supseteq \{choice, payment_a, type_a : a \in \mathrm{Ag}\}$ be a set of atomic propositions, where $choice$, $type_a$, $payment_a$ denote, respectively, the choice at the current state, the type of a, and her payment. A *infinite mechanism* is a wCGS over the atomic propositions AP that satisfies the following:

1. there is one initial state v_ι^θ for each possible type profile $\theta \in \mathbf{\Theta}$;
2. types remain unchanged through transitions, i.e. if $\delta(v, c) = v'$ then $\ell(v, type_a) = \ell(v', type_a)$ for each a;
3. each agent knows her own type: if $v \sim_a v'$, then $\ell(v, type_a) = \ell(v', type_a)$.

The formulas introduced in section 5.2 can be easily adapted to this case by removing the condition over the proposition *terminal*. For instance, for Nash equilibrium, we write:

$$NE_{inf}(\boldsymbol{s}) := \bigwedge_{a \in \mathrm{Ag}} \forall t. \left[(\mathrm{Ag}_{-a}, \boldsymbol{s}_{-a})(a, t)\mathbf{G}(util_a) \leq (\mathrm{Ag}, \boldsymbol{s})\mathbf{G}(util_a)\right]$$

Example 10. Let us illustrate how to represent repeated keyword auctions as wCGSii. This type of mechanism is used by search engines for selling advertising slots after a keyword search [106]. For a keyword of interest, the advertisers submit the maximum amount they are willing to pay for a click on her sponsored link. In each stage, a new allocation of sponsored links is determined. An allocation mechanism that is often considered for this problem is the Generalized Second Price (GSP) [106], in which the agents are allocated slots in decreasing order of bids and the payment for the slot s is the bid of the agent allocated to the slot $s + 1$.

Let us fix a price increment $\mathsf{inc} \in (0, 1]$, a set of slots $S = \{1, ..., \mathsf{l}\}$, where $\mathsf{l} \in \mathbb{N} \setminus \{0\}$. We let the constant $allocation_{a,s} \in [-1, 1]$ denote..... The agents in Ag are the advertisers, each one having a private valuation $\theta_a \in \Theta_a$ for a click, where $\Theta_a \subset [0, 1]$ is a finite set of possible valuations. We assume the valuations are distinct, that is, if $a \neq a'$, then $\theta_a \neq \theta_{a'}$. We denote by $\prec_{\mathrm{Ag}}$ an arbitrary order among the agents in Ag, used in the case of ties. For convenience, we use multiple atomic propositions to denote the social choice. The atomic propositional set is $\mathrm{AP} = \{allocation_{a,s}, payment_a, price_s, type_a : a \in \mathrm{Ag}, s \in S\}$, where $allocation_{a,s}$ represents whether agent a is allocated to slot s, $payment_a$ denotes the payment of agent a, $price_s$ denotes the price of slot s and $type_a$ denotes a's valuation. Define $\mathcal{G}_{GSP} = (\mathrm{Ac}, V, \delta, \ell, V_\iota, \{\sim_a\}_{a \in \mathrm{Ag}})$, where:

- $\mathrm{Ac} = \{0 + x \times \mathsf{inc} : 0 \leq x \leq \frac{1}{\mathsf{inc}}\}$, where $b \in \mathrm{Ac}$ denotes a bid with price b for a click; given $\boldsymbol{c} = (c_a)_{a \in \mathrm{Ag}}$, let $rank_{\boldsymbol{c}} = (a_1, ..., a_\mathsf{n})$ be the sequence of distinct agents in Ag ordered by their bid, that is, $i < j$ if $c_{a_i} > c_{a_j}$ or $c_{a_i} = c_{a_j}$ and $a_i \prec_{\mathrm{Ag}} a_j$ for $i, j \in \{1, ..., \mathsf{n}\}$ with $i \neq j$. In case of draws, the sequence is determined with respect to $\prec_{\mathrm{Ag}}$. We let $rank_{\boldsymbol{c}}(i)$ denote the agent in the i-th position of the sequence $rank_{\boldsymbol{c}}$.
- $V = \{\langle al_1, ..., al_\mathsf{l}, pr_1, ..., pr_\mathsf{l}, (vl_a)_{a \in \mathrm{Ag}}\rangle : al_s \in \mathrm{Ag} \cup \{none\}\ \&\ pr_s \in \mathrm{Ac}\ \&\ vl_a \in \Theta_a\ \&\ a \in \mathrm{Ag}\ \&\ 1 \leq s \leq \mathsf{l}\}$, where each state represents the current slot allocation and prices, with al_s, pr_s, and vl_a denoting the winner of slot s, the price per click of s and a's valuation, resp.;
- For each $v \in V$ and $\boldsymbol{c} = (c_a)_{a \in \mathrm{Ag}}$, the transition function uses the agent's bids to choose the next allocations and prices and is defined as follows: $\delta(v, (c_a)_{a \in \mathrm{Ag}}) = \langle al'_1, ..., al'_\mathsf{l}, pr'_1, ..., pr'_\mathsf{l}, (vl_a)_{a \in \mathrm{Ag}}\rangle$, where for each agent a and slot s, (i) $al_s = rank_{\boldsymbol{c}}(s)$ if $s \leq \mathsf{n}$, and $al_s = none$ otherwise; (ii) $pr_s = c_{rank_{\boldsymbol{c}}(s+1)}$ if $s + 1 \leq \mathsf{n}$, and $pr_s = 0$ otherwise.
- For each agent a, slot $s \in S$ and state $v = \langle al_1, ..., al_\mathsf{l}, pr_1, ..., pr_\mathsf{l}, (vl_a)_{a \in \mathrm{Ag}}\rangle$, the weight function is defined as follows: (i) $\ell(v, allocation_{a,s}) = 1$ if $al_s = a$, and $\ell(v, allocation_{a,s}) = 0$ otherwise; (ii) $\ell(v, price_s) = pr_s$; (iii) $\ell(v, payment_a) = pr_s$ if $allocation_{a,s} = 1$ for some $s \in S$ and $\ell(v, payment_a) = 0$, otherwise; and (iv) $\ell(v, type_a) = vl_a$.
- In an initial state, the prices are 0 and the slots are allocated to *none*, that is, $V_\iota = \{\langle none, ..., none, 0, ..., 0, vl_1, ..., vl_\mathsf{n}\rangle \in V\}$;
- For each agent a and two states $v = \langle al_1, ..., al_\mathsf{l}, pr_1, ..., pr_\mathsf{l}, (vl_a)_{a \in \mathrm{Ag}}\rangle$ and $v' = \langle al'_1, ..., al'_\mathsf{l}, pr'_1, ..., pr'_\mathsf{l}, (vl'_a)_{a \in \mathrm{Ag}}\rangle$ in V, the observation relation $\sim_a$ is such that if $v \sim_a v'$ then (i) $al_s = al_{s'}$, for each $1 \leq s \leq \mathsf{l}$; (ii) $p_s = p_{s'}$, for each $1 \leq s \leq \mathsf{l}$; (iii) $vl_a = vl'_a$.

An important concept when considering infinite mechanisms is the concept of convergence, in the sense that, eventually, the alternative chosen will stabilize and not change in the following states. In the case of keyword auctions, for instance, the convergence of prices is an important aspect of their evaluation [106].

This notion can be easily encoded in $\mathsf{SL}[\mathcal{F}]$: we say a wCGS $\mathcal{G}$ converge to a property φ if the initial states lead to φ being eventually always the case. Formally, a wCGS converge to a condition φ if $[\![\mathbf{FG}(\varphi)]\!]^{\mathcal{G}}_{A}(v_\iota) = 1$ for each initial state $v_\iota \in V_\iota$.

The condition expressed by the formula $NE_{inf}(\boldsymbol{s})$ requires that no agent, by a unilateral change of strategy, can improve their utility in *any* state. To relax this requirement for the convergence to Nash, we can use the formula:

$$NE_{conv}(\boldsymbol{s}) := \mathbf{FG}\Big(\bigwedge_{a \in \mathrm{Ag}} \forall t. \left[(\mathrm{Ag}_{-a}, \boldsymbol{s}_{-a})(a, t)util_a \leq (\mathrm{Ag}, \boldsymbol{s})util_a\right]\Big)$$

9.2. Knowledge-based properties

The framework introduced in this paper captures the classic properties of mechanism design, including the situation in which agents do not know all aspects of the mechanisms (for instance, each others' valuations). A natural extension is to consider properties that take into account the agents' knowledge. This would allow one to model-check, for instance, whether the agents' know the winner of an election or the vote of other agents'.

Here, we introduce a variant of SL[$\mathcal{F}$], called Epistemic SL[$\mathcal{F}$] (or simply, SLK[$\mathcal{F}$]) which augments the logic with epistemic operators.

Definition 27. The syntax of SLK[$\mathcal{F}$] is defined as follows

$$\varphi ::= \mathsf{SL}[\mathcal{F}] \mid K_a \varphi$$

where $a \in \mathrm{Ag}$.

Definition 28 *(SLK[$\mathcal{F}$] semantics).* Let $\mathcal{G} = (\{\mathrm{Ac}_a\}_{a\in\mathrm{Ag}}, V, \delta, \ell, V_\iota, \{\sim_a\}_{a\in\mathrm{Ag}})$ be a wCGSii, $\mathcal{A}$ an assignment, and $\rho \in \{R, r\}$. The satisfaction value $[\![\varphi]\!]^{\mathcal{G},\rho}_{\mathcal{A}}(h) \in [-1,1]$ of an SL[$\mathcal{F}$] formula φ in a history h is defined as follows:

$$[\![K_a\varphi]\!]^{\mathcal{G},\rho}_{\mathcal{A}}(h) = \min_{h\sim_a h'} [\![\varphi]\!]^{\mathcal{G},\rho}_{\mathcal{A}}(h')$$

The remaining formulas of the logic are defined as for SL[$\mathcal{F}$] (Definition 6).

Remark 7. Assuming that functions in $\mathcal{F}$ can be computed in polynomial space, the problem of model-checking SLK[$\mathcal{F}$] with memoryless strategies and imperfect information remains the same for SL[$\mathcal{F}$] under the same assumptions.

In the next section, we illustrate how this operator can be used for mechanism design.

9.2.1. Revenue benchmarks with knowledge

Let us now consider the interplay between the agent's epistemic state and mechanism properties. Hereafter, we focus on the *auctioneer's revenue*, i.e., the total payment among the agents. Guaranteeing revenue is an important MD issue [101]. To address this problem, Chen and Micali (2015,2016) [120,121] investigate the design of auction mechanisms based on "possibilistic beliefs", i.e., beliefs an agent may hold about other agents' types. The mechanism then sets a clear link between the revenue and the agents' epistemic state. We show that this can be represented in a natural way in SLK[$\mathcal{F}$].

Second-belief benchmark Let us consider the *second-belief benchmark* [120] for single good mechanisms. Given a set of possible type profiles $\boldsymbol{\Theta}$, a set $\mathcal{B}_a \subset \boldsymbol{\Theta}$ denotes a belief for agent a about all agents' types.

Given a tuple $S \in [-1,1]^{\mathrm{n}}$, let 2nd-max($S$) be the second maximum value in S, and assume that 2nd-max $\in \mathcal{F}$. Given a correct belief profile $\boldsymbol{\mathcal{B}}$ (i.e., a profile in which the true type is considered possible), the second-belief benchmark (for single-good auctions) is defined as follows:

$$2^{\mathrm{nd}}(\boldsymbol{\mathcal{B}}) := \text{2nd-max}(smv_{a_1}(\boldsymbol{\mathcal{B}}), ..., smv_{a_{\mathrm{n}}}(\boldsymbol{\mathcal{B}}))$$

where $smv_a(\boldsymbol{\mathcal{B}}) := \min_{\theta\in B_a}(\max_{b\in\mathrm{Ag}}(\theta_b))$ denotes the *sure maximum value* according to a.

Let $\mathcal{G}$ be a mechanism. To each state $v \in V$ we can associate a correct belief $\mathcal{B}_a(v)$ for each agent a as follows: $\mathcal{B}_a(v) := (\{\ell(v', type_b) : v' \sim_a v\})_{b\in\mathrm{Ag}}$. We then let $\boldsymbol{\mathcal{B}}(v) = (\mathcal{B}_a(v))_{a\in\mathrm{Ag}}$. The sure maximum value for an agent and the second-belief benchmark in a state correspond to the semantics of the following epistemic SLK[$\mathcal{F}$]-formulas:

$$\varphi^{smv}_a := K_a \max_{a'\in\mathrm{Ag}}(type_{a'})$$

$$\varphi_{2nd} := \text{2nd-max}(\varphi^{smv}_{a_1}, ..., \varphi^{smv}_{a_{\mathrm{n}}})$$

It follows directly that:

Proposition 11. *Given a mechanism $\mathcal{G}$, a state v and a belief profile $\boldsymbol{\mathcal{B}}(v)$, it holds that $[\![\varphi_{2nd}]\!]^{\mathcal{G},\rho}(v) = 2^{\mathrm{nd}}(\boldsymbol{\mathcal{B}}(v))$.*

Proof. Fix an assignment $\mathcal{A}$. We have that $[\![\varphi_{2nd}]\!]^{\mathcal{G},\rho}_{\mathcal{A}}(v) = [\![\text{2nd-max}(\varphi^{smv}_{a_1}, ..., \varphi^{smv}_{a_{\mathrm{n}}})]\!]^{\mathcal{G},\rho}_{\mathcal{A}}(v)$. By the semantics of K_a, φ^{smv}_a denotes the minimum value of $[\![\max_{a\in\mathrm{Ag}}(type_a)]\!]^{\mathcal{G},\rho}_{\mathcal{A}}(v')$, for all $v' \sim_a v$. Since $\mathcal{B}_a(v) = (\{\ell(v', type_b) : v' \sim_a v\})_{b\in\mathrm{Ag}}$, it holds that $[\![\varphi^{smv}_a]\!]^{\mathcal{G},\rho}_{\mathcal{A}}(v) = \min_{\theta\in B_a}(\max_{b\in\mathrm{Ag}}(\theta_b))$. Therefore, $[\![\varphi_{2nd}]\!]^{\mathcal{G},\rho}_{\mathcal{A}}(v) = \text{2nd-max}(smv_{a_1}(\boldsymbol{\mathcal{B}}), ..., smv_{a_{\mathrm{n}}}(\boldsymbol{\mathcal{B}}))$ or simply $[\![\varphi_{2nd}]\!]^{\mathcal{G},\rho}_{\mathcal{A}}(v) = 2^{\mathrm{nd}}(\boldsymbol{\mathcal{B}}(v))$. □

Chen and Micali (2015) design a reward-based single-stage mechanism that ensures that, in equilibrium, the revenue is greater than the second belief minus ϵ, where $\epsilon > 0$ is a reward factor associated to the mechanism. In this one-stage mechanism, agents'

beliefs are constant. But should we devise a multi-stage mechanism to achieve a similar result, one may ask the question whether the revenue in equilibrium is (modulo ϵ) greater than the *initial* second belief, or the *last* second belief (before termination) for instance. Such properties can be expressed in $\mathsf{SLK}[\mathcal{F}]$, as we show for the latter one (the former one is easier). Define formulas

$$\varphi_{revenue} := \mathbf{F}(terminal \wedge \sum_{a \in \mathrm{Ag}} (\mathsf{p}_a))$$

$$\varphi_{last-2nd} := \mathbf{F}(\mathbf{X} terminal \wedge \varphi_{2nd})$$

which compute the final revenue and the second belief before the last round, respectively. Now to check whether a given mechanism satisfies the property in all equilibria of a given kind $E \in \{NE, DSE\}$, one can check whether the following formula has value 1 on this mechanism:

$$\varphi_{2nd,\epsilon} := \forall \boldsymbol{s}.\, E(\boldsymbol{s}) \rightarrow (\varphi_{last-2nd} - \epsilon \leq \varphi_{revenue})$$

Best-belief benchmark for combinatorial auctions Chen and Micali (2016) propose the *best-belief* benchmark for combinatorial auctions. This benchmark maximizes, over all agents, the maximum revenue each one would be sure to obtain if she were to sell all her currently allocated goods to her opponents, based on her beliefs about their preferences over bundles of goods.

Similar to the second-belief benchmark, one could express the best-belief benchmark using $\mathsf{SLK}[\mathcal{F}]$-formulas. The main difference is that the formula would consider the agents beliefs' about each other's valuations over possible choices. Notice that the allocation of bundles can be easily encoded as choices.

10. Conclusion and discussion

Mechanism Design is the problem of creating preference aggregation games that satisfy target properties when played by agents with individual preferences and strategic behavior. In this paper, we propose a novel approach for Automated Mechanism Design (AMD) in which mechanisms can be automatically (i) verified with respect to epistemic and quantitative properties and (ii) generated (or *synthesized*) from partial or complete specifications in a rich logical language. We demonstrate how Strategy Logic provides a formal framework expressive enough to reason about core concepts from AMD intuitively. The main benefit of our approach consists of having a general machinery, which enables us to fully automate the reasoning process. Unlike classical AMD, our definitions and properties are general and not tailored to specific problems and settings. The applications of such an approach are many-fold and include problems extensively considered in the literature, such as voting, fair division, peer selection, coalition formation games, and sponsored search auctions. We provided new complexity results for the model-checking and satisfiability problems for $\mathsf{SL}[\mathcal{F}]$. Similarly to [19,21], these complexity results hold as long as the functions in $\mathcal{F}$ can be computed in the complexity class considered.

The ability of SL to naturally express key strategic concepts such as Nash Equilibria, and the possibility to extend it with quantitative aspects, make it a perfect candidate to become a standard logic for verifying mechanisms, as called for in [15]. Different from standard Social Choice Theory, our approach does not abstract the details of mechanisms. Each state of the game, actions, and transitions are explicitly defined with the wCGS, which enables more practical reasoning about the mechanism. We demonstrate the usefulness of $\mathsf{SL}[\mathcal{F}]$ with examples based on auctions because they "provide a good example of mechanisms which are sufficiently complex to demonstrate the usefulness of formal verification" [15]. In addition, we showed how our setting allows capturing properties that are central in the design of many types of mechanisms other than auctions, including efficiency and Pareto optimality.

The great expressiveness of the language $\mathsf{SL}[\mathcal{F}]$ makes the synthesis very general, unlike previous proposals. While mechanism synthesis from $\mathsf{SL}[\mathcal{F}]$ specifications is undecidable, we solve it in two cases: when the number of actions is bounded, and when agents play in turns. We achieve this thanks to reductions to the satisfiability problem for $\mathsf{BQCTL}^*[\mathcal{F}]$, which we prove to be decidable. These two restrictions still preserve enough expressiveness to model relevant scenarios of mechanism synthesis, which we illustrate with examples based on auctions. The high complexity of the synthesis problem is only in the size of the formula, which is typically rather small as we saw in the examples.

We also notice that some aspects of AMD could be expressed in SL with standard Boolean-valuated semantics. However, the quantitative semantics of $\mathsf{SL}[\mathcal{F}]$ makes it possible to synthesize mechanisms that approximate a specification (satisfy it as much as possible) or maximize some value (such as social welfare in an equilibrium), which is not possible with SL.

The logical formalization of mechanism design may provide a view of social choice problems inherently different from the classical characterizations. In particular, the quantitative semantics of $\mathsf{SL}[\mathcal{F}]$ can help to circumvent well-established economic impossibility results (such as Green and Laffont [122] theorem) by approximating the satisfaction value of target properties and, therefore, achieving good results despite the theoretical impossibility.

Approximate mechanisms The impossibility result of Green and Laffont [122] shows there is no dominant strategy mechanism that is strategyproof, efficient, and budget-balanced. Thus, there is no wCGS under which the truth-revealing strategy profile is a DSE and leads to an efficient and budget-balanced outcome. When we focus on pure strategies, there is no wCGS-mechanism that is both EF and SBB in NE.

Corollary 4 *([122]). For any $v \in V_i$, there exists no wCGS-mechanism $\mathcal{G}$ such that:*

$$[\![DSE(\boldsymbol{s}) \wedge \mathbf{F}(terminal \wedge EF(\boldsymbol{\theta}) \wedge SBB)]\!]_{\mathcal{A}}^{\mathcal{G},\rho}(v) = 1$$

where $\mathcal{A}(s_a) = \hat{\theta}_a$ *for each agent* a.

This impossibility result motivates the design of mechanisms that attempt to circumvent this problem by approximating or relaxing target properties. One approach is to relax the budget-balance constraint to no monetary loss [123]. We can express such a condition through an $\mathsf{SL}[\mathcal{F}]$-formula, similarly to *SBB*. Another solution is to look for mechanisms that approximate the requirements. For instance, Mishra and Sharma (2018) investigate a mechanism that is *SP*, *SBB*, and *nearly* efficient.

The quantitative semantics of $\mathsf{SL}[\mathcal{F}]$ enables to synthesize mechanisms that satisfy a specification above a given threshold $\epsilon \geq 0$, which is not possible with Boolean-valuated SL. On the other hand, Algorithm 1 enables synthesizing mechanisms that approximate efficiency by maximizing social welfare.

Future work There are several directions for future work. First, in relation to the verification, the present setting is enough to capture many kinds of mechanisms where memoryless strategies are sufficient to represent the agents' behavior, such as one-shot or English auctions. However, when participating in sequential auctions, agents could gather information from other agents' behavior and act based on what happened in previous steps of the game [125]. For such situations, we plan to study further the model-checking problem for $\mathsf{SL}[\mathcal{F}]$ with memoryful strategies. In the qualitative setting already, imperfect information yields undecidability, but known decidable cases exist [61,62]. We will investigate them in the quantitative case.

Algorithm 1 returns an arbitrary mechanism that maximizes the satisfaction value of the specification, which does not need to be unique. Choosing the *best* mechanism (e.g. in terms of compactness of the wCGS) when the solution is not unique is an interesting open problem.

We believe the automated synthesis of mechanisms is a promising and powerful tool for AMD. However, the high expressiveness of $\mathsf{SL}[\mathcal{F}]$ may not always be needed for simple classes of mechanisms, and one may consider fragments of it to achieve better complexity. Therefore, an interesting direction for future work is to study the complexity of synthesizing from $\mathsf{SL}[\mathcal{F}]$-fragments, inspired by the SL-fragments One-Goal SL [110,126] and Simple-Goal SL [127], for instance. These fragments are usually computationally easier than full SL, and we can hope that similar results can be established in the quantitative setting.

Finally, experimental results are particularly interesting for generating mechanisms that approximate the satisfaction value of target properties and achieving good results despite the theoretical impossibility. Experimental research will also serve to assess the practical relevance of our proposed approaches, especially in relation to mechanism synthesis from $\mathsf{SL}[\mathcal{F}]$ specification due to the high theoretical complexity of the problem.

CRediT authorship contribution statement

Munyque Mittelmann: Writing – review & editing, Writing – original draft, Methodology, Investigation, Formal analysis, Conceptualization. **Bastien Maubert:** Writing – review & editing, Writing – original draft, Investigation, Formal analysis. **Aniello Murano:** Writing – review & editing, Writing – original draft, Investigation, Supervision. **Laurent Perrussel:** Writing – review & editing, Writing – original draft, Investigation, Supervision, Project administration.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Munyque Mittelmann and Laurent Perrussel report financial support was provided by French National Research Agency (project AGAPE ANR-18-CE23-0013). Aniello Murano, Bastien Maubert, and Munyque Mittelmann report financial support was provided by Ministero dell'Università e della Ricerca (projects PE0000013-FAIR, ECS00000037-MUSA-INFANT, and RIPER E63C22000400001). Aniello Murano reports financial support was provided by JP Morgan Chase Bank (Research Award "Resilience-based Generalized Planning and Strategic Reasoning"). Laurent Perrussel and Aniello Murano report financial support was provided by EU Framework Programme for Research and Innovation ICT Leadership in Enabling and Industrial Technologies (project TAILOR No. 952215). Munyque Mittelmann reports financial support was provided by European Union's Horizon 2020 (Marie Skłodowska-Curie project with grant agreement No 101105549).

Acknowledgements

This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 101105549. This research is partially supported by the ANR project AGAPE ANR-18-CE23-0013, the PNRR MUR projects PE0000013-FAIR and ECS00000037-MUSA-INFANT, and the PRIN 2020 project RIPER - CUP E63C22000400001. The authors would like to thank the anonymous reviewers for their careful reading of the manuscript and their insightful comments and suggestions.

Data availability

No data was used for the research described in the article.

References

[1] V. Conitzer, T. Sandholm, Complexity of mechanism design, in: Proceedings of the 18th Conference in Uncertainty in Artificial Intelligence, UAI 2002, University of Alberta, Morgan Kaufmann, Edmonton, 2002, pp. 103–110.
[2] F. Asselin, B. Jaumard, A. Nongaillard, A technique for large automated mechanism design problems, in: Proceedings of the International Conference on Intelligent Agent Technology, IAT 2006, 2006.
[3] N. Nisan, T. Roughgarden, É. Tardos, V. Vazirani, Algorithmic Game Theory, Cambridge University Press, 2007.
[4] R.B. Myerson, Mechanism Design, Palgrave Macmillan UK, London, 2018, pp. 8632–8644.
[5] N. Hamada, C.-L. Hsu, R. Kurata, T. Suzuki, S. Ueda, M. Yokoo, Strategy-proof school choice mechanisms with minimum quotas and initial endowments, Artif. Intell. 249 (2017) 47–71, https://doi.org/10.1016/j.artint.2017.04.006.
[6] M. Goto, A. Iwasaki, Y. Kawasaki, R. Kurata, Y. Yasuda, M. Yokoo, Strategyproof matching with regional minimum and maximum quotas, Artif. Intell. 235 (2016) 40–57, https://doi.org/10.1016/j.artint.2016.02.002.
[7] H. Aziz, O. Lev, N. Mattei, J.S. Rosenschein, T. Walsh, Strategyproof peer selection using randomization, partitioning, and apportionment, Artif. Intell. 275 (2019) 295–309, https://doi.org/10.1016/j.artint.2019.06.004.
[8] M. Flammini, B. Kodric, G. Varricchio, Strategyproof mechanisms for friends and enemies games, Artif. Intell. 302 (2022) 103610, https://doi.org/10.1016/j.artint.2021.103610.
[9] N. Gatti, A. Lazaric, M. Rocco, F. Trovò, Truthful learning mechanisms for multi-slot sponsored search auctions with externalities, Artif. Intell. 227 (2015) 93–139, https://doi.org/10.1016/j.artint.2015.05.012.
[10] B. Li, D. Hao, H. Gao, D. Zhao, Diffusion auction design, Artif. Intell. 303 (2022) 103631, https://doi.org/10.1016/j.artint.2021.103631.
[11] Y. Luo, N.R. Jennings, A budget-limited mechanism for category-aware crowdsourcing of multiple-choice tasks, Artif. Intell. 299 (2021) 103538, https://doi.org/10.1016/j.artint.2021.103538.
[12] G. Zlotkin, J.S. Rosenschein, Mechanism design for automated negotiation, and its application to task oriented domains, Artif. Intell. 86 (1996) 195–244, https://doi.org/10.1016/0004-3702(95)00104-2.
[13] E. Clarke, O. Grumberg, D. Kroening, D. Peled, H. Veith, Model Checking, MIT Press, 2018.
[14] C. David, D. Kroening, Program synthesis: challenges and opportunities, Philos. Trans. R. Soc. A 375 (2017) 20150403.
[15] M. Pauly, M. Wooldridge, Logic for mechanism design–a manifesto, in: Workshop on Game Theory and Decision Theory in Agent Systems (GTDT), 2003.
[16] R. Alur, T.A. Henzinger, O. Kupferman, Alternating-time temporal logic, J. ACM 49 (2002) 672–713.
[17] K. Chatterjee, T.A. Henzinger, N. Piterman, Strategy logic, Inf. Comput. 208 (2010) 677–693.
[18] F. Mogavero, A. Murano, G. Perelli, M.Y. Vardi, Reasoning about strategies: on the model-checking problem, ACM Trans. Comput. Log. 15 (2014) 1–47.
[19] P. Bouyer, O. Kupferman, N. Markey, B. Maubert, A. Murano, G. Perelli, Reasoning about quality and fuzziness of strategic behaviours, in: Proceedings of the International Joint Conference on Artificial Intelligence, IJCAI 2019, 2019.
[20] V. Conitzer, T. Sandholm, Self-interested automated mechanism design and implications for optimal combinatorial auctions, in: Proceedings of the 5th ACM Conference on Electronic Commerce, EC'04, Association for Computing Machinery, 2004, pp. 132–141.
[21] S. Almagor, U. Boker, O. Kupferman, Formally reasoning about quality, J. ACM 63 (2016) 1–56.
[22] E.M. Clarke, The birth of model checking, in: 25 Years of Model Checking: History, Achievements, Perspectives, Springer, 2008, pp. 1–26.
[23] B. Maubert, M. Mittelmann, A. Murano, L. Perrussel, Strategic reasoning in automated mechanism design, in: Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning, KR 2021, 2021, pp. 487–496.
[24] M. Mittelmann, B. Maubert, A. Murano, L. Perrussel, Automated synthesis of mechanisms, in: Proceedings of the International Joint Conference on Artificial Intelligence, IJCAI 2022, 2022, pp. 426–432.
[25] T. Sandholm, Automated mechanism design: a new application area for search algorithms, in: Principles and Practice of Constraint Programming, CP 2003, Springer Berlin Heidelberg, Berlin, Heidelberg, 2003, pp. 19–36.
[26] M.-F.F. Balcan, T. Sandholm, E. Vitercik, Sample complexity of automated mechanism design, in: D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, R. Garnett (Eds.), Advances in Neural Information Processing Systems, vol. 29, Curran Associates, Inc., 2016, https://proceedings.neurips.cc/paper/2016/file/c667d53acd899a97a85de0c201ba99be-Paper.pdf.
[27] T. Sandholm, V. Conitzer, C. Boutilier, Automated design of multistage mechanisms, in: IJCAI, vol. 7, 2007, pp. 1500–1506.
[28] M. Albert, V. Conitzer, P. Stone, Automated design of robust mechanisms, in: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 31, 2017.
[29] A. Zohar, J.S. Rosenschein, Robust mechanisms for information elicitation, in: Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems, 2006, pp. 1202–1204.
[30] H. Zhang, Y. Cheng, V. Conitzer, Automated mechanism design for classification with partial verification, in: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 35, 2021, pp. 5789–5796.
[31] W. Shen, P. Tang, S. Zuo, Automated mechanism design via neural networks, in: Proceedings of the International Conference on Autonomous Agents and Multi-Agent Systems, AAMAS 2019, 2019.
[32] P. Dütting, Z. Feng, H. Narasimhan, D. Parkes, S.S. Ravindranath, Optimal auctions through deep learning, in: Proceedings of the International Conference on Machine Learning, ICML 2019, 2019.
[33] H. Narasimhan, S.B. Agarwal, D.C. Parkes, Automated mechanism design without money via machine learning, in: Proceedings of the International Joint Conference on Artificial Intelligence, IJCAI 2016, 2016.
[34] Y. Vorobeychik, D.M. Reeves, M.P. Wellman, Constrained automated mechanism design for infinite games of incomplete information, in: Proceedings of the Conference on Uncertainty in Artificial Intelligence, UAI 2007, 2007.
[35] Y. Vorobeychik, D.M. Reeves, M.P. Wellman, Constrained automated mechanism design for infinite games of incomplete information, Auton. Agents Multi-Agent Syst. 25 (2012) 313–351.
[36] J. Niu, K. Cai, S. Parsons, M. Fasli, X. Yao, A grey-box approach to automated mechanism design, Electron. Commer. Res. Appl. 11 (2012) 24–35.
[37] S.K. Narayanaswami, S. Chaudhuri, M. Vardi, P. Stone, Automating mechanism design with program synthesis, in: Adaptive and Learning Agents Workshop, ALA 2022, 2022.
[38] S. Phelps, P. McBurney, S. Parsons, Evolutionary mechanism design: a review, Auton. Agents Multi-Agent Syst. 21 (2010) 237–264.
[39] M.T. Hajiaghayi, R. Kleinberg, T. Sandholm, Automated online mechanism design and prophet inequalities, in: AAAI, vol. 7, 2007, pp. 58–65.
[40] N. Nisan, Introduction to Mechanism Design (for Computer Scientists), Cambridge University Press, 2007, pp. 209–242.
[41] N. Nisan, A. Ronen, Algorithmic mechanism design, Games Econ. Behav. 35 (2001) 166–196.
[42] N. Nisan, Algorithmic Mechanism Design: Through the Lens of Multiunit Auctions, Handbook of Game Theory with Economic Applications, vol. 4, Elsevier, 2015, pp. 477–515.
[43] H. Aziz, H. Chan, B. Lee, B. Li, T. Walsh, Facility location problem with capacity constraints: algorithmic and mechanism design perspectives, in: AAAI, AAAI Press, 2020, pp. 1806–1813.
[44] G. Farina, N. Gatti, Adopting the cascade model in ad auctions: efficiency bounds and truthful algorithmic mechanisms, J. Artif. Intell. Res. 59 (2017) 265–310.
[45] D. Ferraioli, C. Ventre, Obvious strategyproofness needs monitoring for good approximations, in: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 31, 2017.

[46] A.K. Dixit, G. Shakya, S.K. Jakhar, S. Nath, Algorithmic mechanism design for egalitarian and congestion-aware airport slot allocation, Transp. Res., Part E, Logist. Transp. Rev. 169 (2023) 102971.
[47] M. Caminati, M. Kerber, C. Lange, C. Rowat, Sound auction specification and implementation, in: ACM Conference on Economics and Computation, 2015.
[48] G. Barthe, M. Gaboardi, E. Arias, J. Hsu, A. Roth, P.-Y. Strub, Computer-aided verification for mechanism design, in: Conference on Web and Internet Economics, 2016.
[49] P. Jouvelot, E.J.G. Arias, A foundational framework for the specification and verification of mechanism design, in: EC'22-Twenty-Third ACM Conference on Economics and Computation, 2022.
[50] M. Kerber, C. Lange, C. Rowat, An introduction to mechanized reasoning, J. Math. Econ. 66 (2016) 26–39.
[51] N. Troquard, W. van der Hoek, M. Wooldridge, Reasoning about social choice functions, J. Philos. Log. 40 (2011) 473–498.
[52] M. Mittelmann, L. Perrussel, Auction Description Language (ADL): General Framework for Representing Auction-Based Markets, Proceedings of the European Conference on Artificial Intelligence (ECAI 2020), vol. 325, IOS Press, 2020, pp. 825–832.
[53] M. Mittelmann, S. Bouveret, L. Perrussel, Representing and reasoning about auctions, Auton. Agents Multi-Agent Syst. 36 (2022) 20, https://doi.org/10.1007/s10458-022-09547-9.
[54] M. Mittelmann, L. Perrussel, An epistemic logic for reasoning about strategies in general auctions, in: Proceedings of the Workshops of the International Conference on Logic Programming, vol. 2678, 2020.
[55] N. Okada, T. Todo, M. Yokoo, Sat-based automated mechanism design for false-name-proof facility location, in: Proceedings of the International Conference on Principles and Practice of Multi-Agent Systems, PRIMA 2019, 2019.
[56] M. Wooldridge, T. Ågotnes, P. Dunne, W. Van der Hoek, Logic for automated mechanism design-a progress report, in: Proceedings of AAAI Conference on Artificial Intelligence, AAAI 2007, 2007.
[57] F. Belardinelli, W. Jamroga, V. Malvone, M. Mittelmann, A. Murano, L. Perrussel, Reasoning about human-friendly strategies in repeated keyword auctions, in: Proceedings of the International Conference on Autonomous Agents and Multi-Agent Systems, AAMAS 2022, ACM, 2022, pp. 1602–1604.
[58] M. Mittelmann, B. Maubert, A. Murano, L. Perrussel, Formal verification of Bayesian mechanisms, in: AAAI, AAAI Press, 2023, pp. 11621–11629.
[59] M. Pauly, A modal logic for coalitional power in games, J. Log. Comput. 12 (2002) 149–166.
[60] F. Belardinelli, A. Lomuscio, Abstraction-based verification of infinite-state reactive modules, in: Proceedings of the European Conference on Artificial Intelligence, ECAI 2016, 2016.
[61] R. Berthon, B. Maubert, A. Murano, S. Rubin, M. Vardi, Strategy logic with imperfect information, ACM Trans. Comput. Log. 22 (2021).
[62] F. Belardinelli, A. Lomuscio, A. Murano, S. Rubin, Verification of multi-agent systems with public actions against strategy logic, Artif. Intell. 285 (2020).
[63] B. Maubert, A. Murano, Reasoning about knowledge and strategies under hierarchical information, in: Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning, KR 2018, AAAI Press, 2018, pp. 530–540.
[64] G. Ciná, U. Endriss, Proving classical theorems of social choice theory in modal logic, Auton. Agents Multi-Agent Syst. 30 (2016) 963–989.
[65] N. Troquard, W. van der Hoek, M.J. Wooldridge, Reasoning about social choice functions, J. Philos. Log. 40 (2011) 473–498.
[66] T. Ågotnes, W. van der Hoek, M.J. Wooldridge, On the logic of preference and judgment aggregation, Auton. Agents Multi-Agent Syst. 22 (2011) 4–30.
[67] T. Ågotnes, W. Van Der Hoek, J.A. Rodríguez-Aguilar, C. Sierra, M.J. Wooldridge, On the logic of normative systems, in: Proceedings of the International Joint Conference on Artificial Intelligence, IJCAI 2007, vol. 7, 2007, pp. 1175–1180.
[68] N. Bulling, M. Dastani, Norm-based mechanism design, Artif. Intell. 239 (2016) 97–142.
[69] N. Alechina, G. De Giacomo, B. Logan, G. Perelli, Automatic synthesis of dynamic norms for multi-agent systems, in: Proc. of the 19th International Conference on Principles of Knowledge Representation and Reasoning, 2022, pp. 12–21.
[70] J. Wu, J. Cao, H. Sun, C. Wang, A Bayesian optimal social law synthesizing mechanism for strategical agents, Auton. Agents Multi-Agent Syst. 36 (2022) 48.
[71] L. Brice, J. Raskin, M. van den Bogaard, Rational verification for Nash and subgame-perfect equilibria in graph games, in: J. Leroux, S. Lombardy, D. Peleg (Eds.), Proc. of the Int. Symposium on Mathematical Foundations of Computer Science, MFCS 2023, 2023.
[72] S. Almagor, O. Kupferman, G. Perelli, Synthesis of controllable Nash equilibria in quantitative objective game, in: Proc. of the Int. Joint Conference on Artificial Intelligence, IJCAI 2018, ijcai.org, 2018, pp. 35–41.
[73] J. Gutierrez, M. Najib, G. Perelli, M.J. Wooldridge, Equilibrium design for concurrent games, in: 30th International Conference on Concurrency Theory, CONCUR 2019, 2019.
[74] J. Gutierrez, P. Harrenstein, M.J. Wooldridge, From model checking to equilibrium checking: reactive modules for rational verification, Artif. Intell. 248 (2017) 123–157.
[75] J. Gutierrez, M. Najib, G. Perelli, M.J. Wooldridge, Automated temporal equilibrium analysis: verification and synthesis of multi-player games, Artif. Intell. 287 (2020) 103353.
[76] J. Gutierrez, L. Hammond, A.W. Lin, M. Najib, M. Wooldridge, Rational verification for probabilistic systems, in: Proc. of KR-21, 2021, pp. 312–322.
[77] D. Hyland, J. Gutierrez, M.J. Wooldridge, Incentive engineering for concurrent games, in: Proc. of the Conference on Theoretical Aspects of Rationality and Knowledge, TARK 2023, in: EPTCS, vol. 379, 2023, pp. 344–358.
[78] D. Hyland, J. Gutierrez, M. Wooldridge, Principal-agent Boolean games, in: Proc. of the Int. Joint Conference on Artificial Intelligence, IJCAI 2023, ijcai.org, 2023, pp. 144–152.
[79] D. Hyland, M. Mittelmann, A. Murano, G. Perelli, M. Wooldridge, Incentive design for rational agents, in: Proceedings of the 21st International Conference on Principles of Knowledge Representation and Reasoning, 2024, pp. 464–474.
[80] E. Emerson, C. Jutla, Tree automata, mu-calculus and determinacy, in: Proceedings 32nd Annual Symposium of Foundations of Computer Science, 1991, pp. 368–377.
[81] A. Ehrenfeucht, J. Mycielski, Positional strategies for mean payoff games, Int. J. Game Theory 8 (1979) 109–113.
[82] A. Chakrabarti, L. de Alfaro, T.A. Henzinger, M. Stoelinga, Resource interfaces, in: R. Alur, I. Lee (Eds.), Embedded Software, Springer Berlin Heidelberg, 2003, pp. 117–133.
[83] P. Bouyer, U. Fahrenberg, K.G. Larsen, N. Markey, J. Srba, Infinite runs in weighted timed automata with energy constraints, in: F. Cassez, C. Jard (Eds.), Formal Modeling and Analysis of Timed Systems, 6th International Conference, FORMATS 2008, Saint Malo, France, September 15-17, 2008. Proceedings, in: Lecture Notes in Computer Science, vol. 5215, Springer, 2008, pp. 33–47.
[84] J. Gutierrez, A. Murano, G. Perelli, S. Rubin, T. Steeples, M.J. Wooldridge, Equilibria for games with combined qualitative and quantitative objectives, Acta Inform. 58 (2021) 585–610, https://doi.org/10.1007/s00236-020-00385-4.
[85] S. Almagor, U. Boker, O. Kupferman, Discounting in LTL, in: E. Ábrahám, K. Havelund (Eds.), Tools and Algorithms for the Construction and Analysis of Systems - 20th International Conference, TACAS 2014, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2014, Grenoble, France, April 5-13, 2014. Proceedings, in: Lecture Notes in Computer Science, vol. 8413, Springer, 2014, pp. 424–439.
[86] M. Mittelmann, A. Murano, L. Perrussel, Discounting in strategy logic, in: IJCAI, ijcai.org, 2023, pp. 225–233.
[87] P. Bouyer, N. Markey, R.M. Matteplackel, Averaging in LTL, in: P. Baldan, D. Gorla (Eds.), CONCUR 2014 - Concurrency Theory - 25th International Conference, CONCUR 2014, Rome, Italy, September 2-5, 2014. Proceedings, in: Lecture Notes in Computer Science, vol. 8704, Springer, 2014, pp. 266–280.
[88] T.A. Henzinger, V.S. Prabhu, Timed alternating-time temporal logic, in: E. Asarin, P. Bouyer (Eds.), Formal Modeling and Analysis of Timed Systems, 4th International Conference, FORMATS 2006, Paris, France, September 25-27, 2006, Proceedings, in: Lecture Notes in Computer Science, vol. 4202, Springer, 2006, pp. 1–17.

[89] T. Brihaye, F. Laroussinie, N. Markey, G. Oreiby, Timed concurrent game structures, in: L. Caires, V.T. Vasconcelos (Eds.), CONCUR 2007 - Concurrency Theory, 18th International Conference, CONCUR 2007, Lisbon, Portugal, September 3-8, 2007, Proceedings, in: Lecture Notes in Computer Science, vol. 4703, Springer, 2007, pp. 445–459.
[90] W. Jamroga, B. Konikowska, D. Kurpiewski, W. Penczek, Multi-valued verification of strategic ability, Fundam. Inform. 175 (2020) 207–251, https://doi.org/10.3233/FI-2020-1955.
[91] F. Laroussinie, N. Markey, G. Oreiby, Model-checking timed, in: E. Asarin, P. Bouyer (Eds.), Formal Modeling and Analysis of Timed Systems, 4th International Conference, FORMATS 2006, Paris, France, September 25-27, 2006, Proceedings, in: Lecture Notes in Computer Science, vol. 4202, Springer, 2006, pp. 245–259.
[92] N. Bulling, V. Goranko, Combining quantitative and qualitative reasoning in concurrent multi-player games, Auton. Agents Multi-Agent Syst. 36 (2022) 2, https://doi.org/10.1007/s10458-021-09531-9.
[93] S. Vester, On the complexity of model-checking branching and alternating-time temporal logics in one-counter systems, in: B. Finkbeiner, G. Pu, L. Zhang (Eds.), Automated Technology for Verification and Analysis, Springer International Publishing, Cham, 2015, pp. 361–377.
[94] W. Jamroga, M. Mittelmann, A. Murano, G. Perelli, Playing quantitative games against an authority: on the module checking problem, in: AAMAS, International Foundation for Autonomous Agents and Multiagent Systems/ACM, 2024, pp. 926–934.
[95] R. Fagin, J.Y. Halpern, Y. Moses, M. Vardi, Reasoning About Knowledge, MIT Press, 2004.
[96] P. Bouyer, O. Kupferman, N. Markey, B. Maubert, A. Murano, G. Perelli, Reasoning about quality and fuzziness of strategic behaviors, ACM Trans. Comput. Log. 24 (2023) 1–38.
[97] H. Aziz, I. Caragiannis, A. Igarashi, T. Walsh, Fair allocation of indivisible goods and chores, Auton. Agents Multi-Agent Syst. 36 (2022) 1–21.
[98] S. Knight, B. Maubert, Dealing with imperfect information in strategy logic, CoRR, arXiv:1908.02488, http://arxiv.org/abs/1908.02488.
[99] B. Lubin, A.I. Juda, R. Cavallo, S. Lahaie, J. Shneidman, D.C. Parkes, Ice: an expressive iterative combinatorial exchange, J. Artif. Intell. Res. 33 (2008) 33–77.
[100] R. Freeman, M. Brill, V. Conitzer, General tiebreaking schemes for computational social choice, in: G. Weiss, P. Yolum, R.H. Bordini, E. Elkind (Eds.), Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2015, Istanbul, Turkey, May 4-8, 2015, ACM, 2015, pp. 1401–1409.
[101] V. Krishna, Auction Theory, Academic Press, 2009.
[102] D.C. Parkes, L.H. Ungar, Iterative Combinatorial Auctions: Achieving Economic and Computational Efficiency, University of Pennsylvania, Philadelphia, PA, 2001.
[103] W. Vickrey, Counterspeculation, auctions, and competitive sealed tenders, J. Finance 16 (1961) 8–37.
[104] M.O. Jackson, Optimization and Operations Research -Volume III, EOLSS Publications, 2009.
[105] P.-Y. Schobbens, Alternating-time logic with imperfect recall, Electron. Notes Theor. Comput. Sci. 85 (2004) 82–93.
[106] M. Cary, A. Das, B. Edelman, I. Giotis, K. Heimerl, A.R. Karlin, C. Mathieu, M. Schwarz, Greedy bidding strategies for keyword auctions, in: Proc. of the 8th ACM Conf. on Electronic Commerce, EC 2007, Association for Computing Machinery, New York, 2007, pp. 262–271.
[107] U. Grandi, A. Loreggia, F. Rossi, K.B. Venable, T. Walsh, Restricted manipulation in iterative voting: Condorcet efficiency and Borda score, in: P. Perny, M. Pirlot, A. Tsoukiàs (Eds.), Algorithmic Decision Theory, Springer, Berlin, Heidelberg, 2013, pp. 181–192.
[108] S. Almagor, U. Boker, O. Kupferman, Formally reasoning about quality, J. ACM 63 (2016) 24, https://doi.org/10.1145/2875421.
[109] P. Cermák, A. Lomuscio, F. Mogavero, A. Murano, Practical verification of multi-agent systems against SLK specifications, Inf. Comput. 261 (2018) 588–614.
[110] F. Mogavero, A. Murano, G. Perelli, M.Y. Vardi, Reasoning about strategies: on the satisfiability problem, Log. Methods Comput. Sci. 13 (2017), https://doi.org/10.23638/LMCS-13(1:9)2017.
[111] N. Troquard, D. Walther, On satisfiability in ATL with strategy contexts, in: Proceedings of the European Conference on Logics in Artificial Intelligence, JELIA 2012, 2012.
[112] F. Laroussinie, N. Markey, Augmenting ATL with strategy contexts, Inf. Comput. 245 (2015) 98–123, https://doi.org/10.1016/j.ic.2014.12.020.
[113] F. Laroussinie, N. Markey, Quantified CTL: expressiveness and complexity, Log. Methods Comput. Sci. 10 (2014), https://doi.org/10.2168/LMCS-10(4:17)2014.
[114] J.-É. Pin, Handbook of Automata Theory, European Mathematical Society Publishing House, Zürich, 2021.
[115] T. French, Quantified propositional temporal logic with repeating states, in: Proceedings of the 10th International Symposium on Temporal Representation and Reasoning, 2003 and Fourth International Conference on Temporal Logic, TIME 2003, 2003.
[116] C. Löding, Automata on infinite trees, in: J. Pin (Ed.), Handbook of Automata Theory, European Mathematical Society Publishing House, Zürich, Switzerland, 2021, pp. 265–302.
[117] A. Pnueli, R. Rosner, On the synthesis of a reactive module, in: Symposium on the Principles of Programming Languages, POPL 1989, ACM, New York, 1989, pp. 179–190.
[118] P. Klemperer, Auction theory: a guide to the literature, J. Econ. Surv. 13 (1999) 227–286.
[119] S. Bouveret, J. Lang, Manipulating picking sequences, in: Proc. of the European Conference on Artificial Intelligence, ECAI 2014, 2014.
[120] J. Chen, S. Micali, Mechanism design with possibilistic beliefs, J. Econ. Theory 156 (2015) 77–102.
[121] J. Chen, S. Micali, Leveraging possibilistic beliefs in unrestricted combinatorial auctions, Games 7 (2016).
[122] J. Green, J.-J. Laffont, Incentives in Public Decision-Making, Elsevier North-Holland, 1979.
[123] R. Cavallo, Optimal decision-making with minimal waste: strategyproof redistribution of vcg payments, in: AAMAS, 2006, pp. 882–889.
[124] D. Mishra, T. Sharma, A simple budget-balanced mechanism, Soc. Choice Welf. 50 (2018) 147–170.
[125] T.D. Jeitschko, Learning in sequential auctions, South. Econ. J. 65 (1998) 98–112.
[126] P. Cermák, A. Lomuscio, A. Murano, Verifying and synthesising multi-agent systems against one-goal strategy logic specifications, in: Proceedings of the AAAI Conference on Artificial Intelligence, AAAI 2015, 2015.
[127] F. Belardinelli, W. Jamroga, D. Kurpiewski, V. Malvone, A. Murano, Strategy logic with simple goals: tractable reasoning about strategies, in: Proceedings of the International Joint Conference on Artificial Intelligence, IJCAI 2019, 2019.