Automated Synthesis of Mechanisms

Munyque Mittelmann[1*], **Bastien Maubert**[2], **Aniello Murano**[2] and **Laurent Perrussel**[1]
[1]IRIT - Université Toulouse 1 Capitole, France
[2]Università degli Studi di Napoli "Federico II", Italy
{munyque.mittelmann,laurent.perrussel}@ut-capitole.fr, {bastien.maubert,nello.murano}@gmail.com

Abstract

Mechanism Design aims to design a game so that a desirable outcome is reached regardless of agents' self-interests. In this paper, we show how this problem can be rephrased as a synthesis problem, where mechanisms are automatically synthesized from a partial or complete specification in a high-level logical language. We show that Quantitative Strategy Logic is a perfect candidate for specifying mechanisms as it can express complex strategic and quantitative properties. We solve automated mechanism design in two cases: when the number of actions is bounded, and when agents play in turn.

1 Introduction

Mechanism Design is focused on the designing of games, that is, *mechanisms*, that aggregate agents' preferences towards a single joint decision. When participants act rationally (in the game theoretical sense), such games should ensure a preferable behavior of the players (e.g. truthfullness) as well as desirable features of the decision (e.g. social welfare maximization) [Nisan *et al.*, 2007]. Traditionally, mechanisms have been formulated by human specialists, who use their knowledge and experience for defining the game rules. Conitzer and Sandholm (2003) introduced Automated Mechanism Design (AMD), whose goal is to automatically create mechanisms for solving a specific preference aggregation problem.

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 [Shen *et al.*, 2019; Dütting *et al.*, 2019]. Statistical machine learning techniques have also been considered in domains without money [Narasimhan *et al.*, 2016]. [Vorobeychik *et al.*, 2007] proposes a black-box optimization algorithm for evaluating candidate mechanisms. Evolutionary search methods have also been used to optimize double auctions [Niu *et al.*, 2012], and [Asselin *et al.*, 2006] addresses AMD through linear programming and optimization.

In the spirit of the long-established logical approach to systems verification [Clarke *et al.*, 2018] and synthesis [David and Kroening, 2017], in this work we propose a new approach to Automated Mechanism Design, which consists in automatically synthesizing mechanisms from a partial or complete specification expressed in a high-level logical language.

A related topic is normative systems [Ågotnes *et al.*, 2007], which define constraints (or *obligations*) on the behaviour of agents. [Bulling and Dastani, 2016] investigates how concepts from mechanism design can be used to analyse the enforcement of norms with preferences modelled using Linear-time Temporal Logic (LTL).

Concerning the choice of specification language, Auction Description Language [Mittelmann and Perrussel, 2020] is general enough to represent many kinds of protocols, but lacks a strategic dimension and cannot be used to reason about equilibria. [Okada *et al.*, 2019] uses Boolean Satisfiability, but their approach is restricted to one type of mechanism and does not handle strategic, temporal and quantitative specifications. [Gutierrez *et al.*, 2019] considers system specifications given in LTL, and studies the implementation of a mechanism to ensure temporal properties in equilibrium. [Pauly and Wooldridge, 2003; Wooldridge *et al.*, 2007] first advocated the use of strategic logics to reason about mechanisms. They consider Alternating-time Temporal Logic (ATL) [Alur *et al.*, 2002], but observe that it lacks the ability to reason about quantitative aspects such as preferences, as well as game-theoretic concepts such as equilibria. Such features are key for modelling and evaluating mechanisms, specially the ones with monetary transfers.

We consider Quantitative Strategy Logic (SL[$\mathcal{F}$]) [Bouyer *et al.*, 2019], a logic that has recently been shown to be a great candidate to formally reason about mechanisms [Maubert *et al.*, 2021]. Indeed SL[$\mathcal{F}$], which subsumes ATL, is expressive enough to express complex solution concepts such as Nash equilibrium and properties about quantities. This language thus allows for specifications that contain constraints on mechanism properties (for instance, in Auction Design, the efficiency and budget-balance). Its quantitative semantics, with satisfaction values that reflect how well a model satisfies a formula, also allows us to investigate the constructions of mechanisms that approximate such properties, which is not possible with standard Strategy Logic (SL) [Chatterjee *et al.*, 2010; Mogavero *et al.*, 2014].

Contribution In this work we propose a novel perspective on the design of mechanisms. Our approach enables automatically generating optimal mechanisms from a quantitative

*Contact Author

logical specification, which may include not only game rules but also requirements over the strategic behaviour of participants and quality of the outcome. We rephrase the AMD problem in terms of synthesis from $\mathsf{SL}[\mathcal{F}]$ specifications. To solve this synthesis problem we investigate the related satisfiability problem for $\mathsf{SL}[\mathcal{F}]$, which had not been studied so far. Finally, we illustrate the relevance of mechanism synthesis with examples based on Auction Design.

Outline In Section 2 we recall the logic $\mathsf{SL}[\mathcal{F}]$. In Section 3 we address the satisfiability and synthesis for this logic. In Section 4, we address the problem of AMD using $\mathsf{SL}[\mathcal{F}]$. Finally, Section 5 concludes the paper.

2 Quantitative Strategy Logic

For the remainder of the paper, we fix a set of atomic propositions AP, a set of agents Ag and a set of strategy variables Var, except when stated otherwise. We let n be the number of agents in Ag. We also let $\mathcal{F} \subseteq \{f\colon [-1,1]^m \to [-1,1] \mid m \in \mathbb{N}\}$ be a set of functions over $[-1,1]$ of possibly different arities, that will parameterise the logics we consider.

Definition 1. The syntax of $\mathsf{SL}[\mathcal{F}]$ is defined by the grammar

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

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

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". 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 $x \mapsto -x$.

Remark 1. While [Bouyer *et al.*, 2019] considers values in [0,1], we follow [Maubert *et al.*, 2021] and use interval [-1,1] instead, which is more adapted to model some mechanisms of interest, such as double-sided auctions.

A variable is *free* in formula φ if it is bound to an agent without being quantified upon, and 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 free(φ), and a formula φ is a *sentence* if free(φ) $= \emptyset$.

Definition 2. A *weighted concurrent game structure* (wCGS) is a tuple $\mathcal{G} = (\text{Ac}, V, v_\iota, \delta, \ell)$ where (i) Ac is a finite set of *actions*; (ii) V is a finite set of *positions*; (iii) $v_\iota \subseteq V$ is an *initial position*; (iv) $\delta : V \times \text{Ac}^{\text{Ag}} \to V$ is a *transition function*; (v) $\ell : V \times \text{AP} \to [-1,1]$ is a *weight function*.

Remark 2. Because we are interested in synthesizing mechanisms of finite size, we restrict attention to finite models.

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

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

A *play* $\pi = v_1 v_2 ...$ is an infinite sequence of positions 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 position at index i in play π. A *history* h is a finite prefix of a play, last(h) is the last position of history h, $|h|$ is the length of h and Hist is the set of histories.

A *strategy* is a function $\sigma : \text{Hist} \to \text{Ac}$ that maps each history to an action. We let *Str* be the set of strategies.

Remark 3. Unlike [Maubert *et al.*, 2021], which considers strategies without memory, we consider strategies with perfect recall, as in [Bouyer *et al.*, 2019]. The main reason is that for memoryless strategies satisfiability, and thus synthesis, is undecidable already for SL, even for bounded actions or turn-based systems [Laroussinie and Markey, 2015]. Considering perfect recall strategies means that the systems we synthesize satisfy the strategic aspects of the specification assuming that agents' actions can depend on the past.

An *assignment* $\mathcal{A} : \text{Ag} \cup \text{Var} \to \textit{Str}$ 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.

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

Definition 3. Let $\mathcal{G} = (\text{Ac}, V, \delta, \ell, V_\iota)$ be a wCGS, and $\mathcal{A}$ an assignment. The satisfaction value $[\![\varphi]\!]^{\mathcal{G}}_{\mathcal{A}}(h) \in [-1,1]$ of an $\mathsf{SL}[\mathcal{F}]$ formula φ in a history h is defined as follows, where π denotes Out$(\mathcal{A}, h)$:

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

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

We can define the following classic abbreviations: $\bot := \neg\top$, $\varphi \wedge \varphi' := \neg(\neg\varphi \vee \neg\varphi')$, $\varphi \to \varphi' := \neg\varphi \vee \varphi'$, $\mathbf{F}\psi := \top \mathbf{U} \psi$, $\mathbf{G}\psi := \neg\mathbf{F}\neg\psi$ and $\forall s.\, \varphi := \neg\exists s.\, \neg\varphi$. 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^*.

3 Satisfiability and Synthesis of $\mathsf{SL}[\mathcal{F}]$

We investigate the following satisfiability problem for $\mathsf{SL}[\mathcal{F}]$.

Definition 4. 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. $[\![\varphi]\!]^{\mathcal{G}} \geq \vartheta$.

The satisfiability problem for SL was proved undecidable in [Mogavero *et al.*, 2017], 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 [Troquard and Walther, 2012; Laroussinie and Markey, 2015] that satisfiability of ATL with strategy context ($\mathsf{ATL}_{\mathrm{sc}}$) is undecidable for finite models. Since $\mathsf{ATL}_{\mathrm{sc}}$ can be expressed in SL [Laroussinie and Markey, 2015] and SL in $\mathsf{SL}[\mathcal{F}]$, by taking $\mathcal{F} = \{\top, \vee, \neg\}$ [Bouyer *et al.*, 2019], we obtain the following result.

Proposition 1. *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 [Laroussinie and Markey, 2015]. 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 $\mathsf{BQCTL}^*[\mathcal{F}]$.

3.1 Booleanly-quantified $\mathsf{CTL}^*[\mathcal{F}]$

The logic $\mathsf{BQCTL}^*[\mathcal{F}]$ [Bouyer *et al.*, 2019], a quantitative extension of QCTL^* [Laroussinie and Markey, 2014], itself an extension of CTL* with quantifiers on atomic proposition. In $\mathsf{BQCTL}^*[\mathcal{F}]$ the semantics is quantitative, but the quantifiers on propositions consider only Boolean values. To be consistent with $\mathsf{SL}[\mathcal{F}]$ we consider [-1,1] as range of values instead of [0,1] as in [Bouyer *et al.*, 2019]. This changes nothing to the results presented there.

The syntax of $\mathsf{BQCTL}^*[\mathcal{F}]$ 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. Formulas of type φ are called *state formulas*, those of type ψ are called *path formulas*, and $\mathsf{BQCTL}^*[\mathcal{F}]$ 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 $\mathsf{SL}[\mathcal{F}]$.

Definition 5. A *weighted Kripke structure* (wKS) is a tuple $\mathcal{S} = (S, s_\iota, R, \ell)$ where S is a set of states, $s_\iota \in S$ is an initial state, $R \subseteq S \times S$ is a left-total[1] transition relation, and $\ell \colon S \to [-1,1]^{\mathrm{AP}}$ 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 $\mathrm{Val}_{\mathcal{S}} = \{\ell(s)(p) \mid s \in S \text{ and } p \in \mathrm{AP}\}$ be the finite set of values appearing in $\mathcal{S}$.

Given finite nonempty sets X of 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 $\mathrm{Br}(u)$ be the set of branches that start in node u.

A *binary tree* is a X-tree where $|X| = 2$. A tree is *regular* if it is the unfolding of some finite Kripke structure.

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 $q \neq p$.

Finally, 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 $\ell'(u) = \ell(\mathrm{last}(u))$ for every $u \in \mathrm{Hist}_{\mathcal{S}}$.

Different semantics exist for QCTL^*, which differ on how proposition quantification is interpreted (see [Laroussinie and Markey, 2014] for more on the different semantics for QCTL^*). In this paper we focus on the tree semantics, which allows capturing perfect-recall strategies.

Definition 6 (Semantics)**.** Consider a finite set $\mathcal{V} \subseteq [-1,1]$ of possible values. The satisfaction value $[\![\varphi]\!]^t(u)$ of a $\mathsf{BQCTL}^*[\mathcal{F}]$ 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:

$$[\![p]\!]^t(u) = \ell(u)(p)$$
$$[\![\exists p.\, \varphi]\!]^t(u) = \sup_{\ell_p : \tau \to \{-1,1\}} [\![\varphi]\!]^{t \otimes \ell_p}(u)$$
$$[\![\mathbf{E}\psi]\!]^t(u) = \sup_{\lambda \in \mathrm{Br}(u)} [\![\psi]\!]^t(\lambda)$$
$$[\![f(\varphi_1, ..., \varphi_n)]\!]^t(u) = f([\![\varphi_1]\!]^t(u), ..., [\![\varphi_n]\!]^t(u))$$
$$[\![\varphi]\!]^t(\lambda) = [\![\varphi]\!]^t(\lambda_0)$$
$$[\![\mathbf{X}\psi]\!]^t(\lambda) = [\![\psi]\!]^t(\lambda_{\geq 1})$$
$$[\![\psi_1 \mathbf{U} \psi_2]\!]^t(\lambda) = \sup_{i \geq 0} \min([\![\psi_2]\!]^t(\lambda_{\geq i}), \min_{0 \leq j < i} [\![\psi_1]\!]^t(\lambda_{\geq j}))$$
$$[\![f(\psi_1, ..., \psi_n)]\!]^t(\lambda) = f([\![\psi_1]\!]^t(\lambda), ..., [\![\psi_n]\!]^t(\lambda))$$

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

3.2 Deciding $\mathsf{BQCTL}^*[\mathcal{F}]$ satisfiability

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

Definition 7. 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 $\mathcal{S}$ s.t. $[\![\varphi]\!]^{t_{\mathcal{S}}} \geq \vartheta$.

Satisfiability for QCTL^* with structure semantics is undecidable [French, 2003], but decidable for the tree semantics [Laroussinie and Markey, 2014] which we consider in this paper. Relying on the automata construction developed in [Bouyer *et al.*, 2019] to model check $\mathsf{BQCTL}^*[\mathcal{F}]$, we show that satisfiability is also decidable for $\mathsf{BQCTL}^*[\mathcal{F}]$.

Theorem 1. *Satisfiability of* $\mathsf{BQCTL}^*[\mathcal{F}]$ *is decidable.*

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

The lower bounds are inherited from the satisfiability problem for QCTL^* [Laroussinie and Markey, 2014]. 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.

Given a finite wKS $\mathcal{S}$, by adding dummy nodes labeled with proposition p_{int} one can build a wKS $\widetilde{\mathcal{S}}$ with binary branching that simulates $\mathcal{S}$; given a $\mathsf{BQCTL}^*[\mathcal{F}]$ formula φ, one can then define a formula $\widetilde{\varphi}$ that ignores these dummy nodes and has same satisfaction value on $\widetilde{\mathcal{S}}$ as φ on $\mathcal{S}$.

Lemma 1. *For every* $\mathsf{BQCTL}^*[\mathcal{F}]$ *formula* φ *and every finite wKS* $\mathcal{S}$, $[\![\varphi]\!]^{\mathcal{S}} = [\![\widetilde{\varphi}]\!]^{\widetilde{\mathcal{S}}}$.

Let Φ be a $\mathsf{BQCTL}^*[\mathcal{F}]$ formula and $\vartheta > -1$ a threshold. Let also $\widetilde{\Phi}$ be the corresponding formula on structures with binary branching, and $\varphi_2 = \neg p_{\text{int}} \wedge \mathbf{AGF}\neg p_{\text{int}}$. Consider also function Bool such that $\text{Bool}(x) = 1$ if $x \in \{-1, 1\}$, and -1 otherwise, which we use to check that propositions take only Boolean values -1 or 1. We can prove the following:

Lemma 2. *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* $[\![\mathbf{AG}\mathit{Bool}(p_{\mathit{int}}) \wedge \varphi_2 \wedge \widetilde{\Phi}]\!]^t \geq \vartheta$.

It is shown in [Bouyer *et al.*, 2019] that we can build an automaton $\mathcal{A}$ that accepts binary trees t on which the satisfaction value of $\mathbf{AG}\text{Bool}(p_{\text{int}}) \wedge \varphi_2 \wedge \widetilde{\Phi}$ belongs to $[\vartheta, 1]$ (we refer to [Pin, 2021, Ch. 8] for standard definitions and results on tree automata). The number of states of the automaton is $(k+1)$-exponential in the nesting depth of the formula, and its index k-exponential. From Lemma 2 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 polynomial in the number of states and exponential in the index. The overall complexity is thus $(k+1)$-exponential in time.

3.3 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 position there is a unique agent who can determine the next position by the choice of its action. Formally, a wCGS $\mathcal{G} = (\text{Ac}, V, v_\iota, \delta, \ell)$ is *turn-based* if for every position $v \in V$ there exists a unique agent a such that for every successor $v' \in \{\delta(v, \boldsymbol{c}) \mid \boldsymbol{c} \in \text{Ac}^{\text{Ag}}\}$ 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 position 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 $\mathcal{V}$.

Definition 8. 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 establish the following results (proofs are in appendix).

Theorem 2. *Let* $\mathcal{V}$ *be a finite set of values and* Ac *a finite set of actions. Then* $\mathcal{V}$*-satisfiability of* $\mathsf{SL}[\mathcal{F}]$ *over* Ac*-wCGS is decidable.*

Theorem 3. *Let* $\mathcal{V}$ *be a finite set of values. Then* $\mathcal{V}$*-satisfiability of* $\mathsf{SL}[\mathcal{F}]$ *over turn-based wCGS is decidable.*

Concerning complexity, for Theorems 1, 2 and 3 the problems are $(k+1)$-EXPTIME-complete where k is the maximal number of nested quantifiers on propositions (for $\mathsf{BQCTL}^*[\mathcal{F}]$) or strategies (for $\mathsf{SL}[\mathcal{F}]$). Blocks of successive quantifiers can be counted as one if they are all existential or all universal, so that an existential quantification on a strategy profile for all agents just adds one exponential for instance.

3.4 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 accepted by the automaton (see [Pnueli and Rosner, 1989]), and from such a tree in our setting we can infer a witness wCGS.

Second, it is proved in [Bouyer *et al.*, 2019] 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 3 ([Bouyer *et al.*, 2019])**.** *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* $\text{Val}_{\varphi,\mathcal{V}} = \{[\![\varphi]\!]^{\mathcal{G}} \mid \mathcal{G}$ *is a* $\mathcal{V}$*-weighted wCGS*$\}$ *is finite, and one can compute a set* $\widetilde{\text{Val}}_{\varphi,\mathcal{V}}$ *of size at most* $|\mathcal{V}|^{|\varphi|}$ *such that* $\text{Val}_{\varphi,\mathcal{V}} \subseteq \widetilde{\text{Val}}_{\varphi,\mathcal{V}}$.

Algorithm 1: Optimal mechanism synthesis

Data: A $\mathsf{SL}[\mathcal{F}]$ specification Φ and a set of possible values for atomic propositions $\mathcal{V}$
Result: A wCGS $\mathcal{G}$ such that $[\![\Phi]\!]^{\mathcal{G}}$ is maximal
Compute $\widetilde{\text{Val}}_{\Phi,\mathcal{V}}$;
Let $\nu_1, ..., \nu_n$ be a decreasing enumeration of $\widetilde{\text{Val}}_{\Phi,\mathcal{V}}$;
for *i=1...n* **do**
  Solve $\mathcal{V}$-satisfiability for Φ and $\vartheta = \nu_i$;
  if *there exists* $\mathcal{G}$ *such that* $[\![\Phi]\!]^{\mathcal{G}} \geq \nu_i$ **then**
    return $\mathcal{G}$;
  end
end

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

4 Automated Mechanism Design

This section aims to motivate the use of synthesis of $\mathsf{SL}[\mathcal{F}]$ for mechanism design. We first recall basic concepts used to formalize mechanisms, which determine how to choose one option among several alternatives, based on agents' strategies. Since many mechanisms describe monetary transfers, we assume that each alternative is a tuple $(x, \boldsymbol{p})$ where $x \in \mathcal{X}$ is a *choice* from a finite set of choices $\mathcal{X} \subset [-1, 1]$, and $p_a \in [-1, 1]$ is the payment for agent a. 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 determines how she values each choice $x \in \mathcal{X}$; this is represented by a *valuation function* $\mathsf{v}_a : \mathcal{X} \times \Theta_a \to [-1, 1]$.

In this section we assume that $\mathcal{F}$ contains the constant θ_a and the function v_a for each agent a and type $\theta_a \in \Theta_a$. We also assume that $\mathcal{F}$ contains the difference function $-$, the n-ary sum function $\sum$, the equality function $=$ and the comparison functions $\leq$, $>$, $<$ and $\neq$. For readability we use the infix notation $x-y$. These functions are defined with the standard meaning. Finally, we assume that types, payments and valuations are normalized so that all values remain in $[-1, 1]$.

A mechanism consists of a description of the agents' possible strategies, and a description of the alternatives that result from them. As described in [Maubert *et al.*, 2021], we can represent mechanisms as wCGS and verify their equilibrium outcome in relation to a number of economic properties.

Definition 9. Let $\mathrm{AP} \supseteq \{\mathrm{choice}, \mathrm{pay}_a, \mathrm{ter} : a \in \mathrm{Ag}\}$, where choice and pay_a denote respectively the choice elected by the mechanism, and the payment of agent a. The proposition ter specifies whether a position 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 position*, i.e., a sink[2] where proposition ter has value 1; (ii) in all non-terminal positions, ter has value -1.

4.1 Characterizing properties with $\mathsf{SL}[\mathcal{F}]$

$\mathsf{SL}[\mathcal{F}]$ can express a variety of important notions in mechanism design, such as strategy proofness, individual rationality, efficiency, budget-balance, Pareto optimality, and different kinds of game-theoretic equilibria [Maubert *et al.*, 2021]. We recall the formulas for some of these notions. Let $\boldsymbol{\theta} = (\theta_a)_{a\in\mathrm{Ag}}$ be a type profile in $\boldsymbol{\Theta}$.

First define $\mathrm{util}_a(\theta_a) := \mathsf{v}_a(\mathrm{choice}, \theta_a) - \mathrm{pay}_a$. This is an $\mathsf{SL}[\mathcal{F}]$ formula, whose value in a terminal position is equal to the agent's utility, which she tries to maximize.

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

Individual rationality (IR) expresses the idea that an agent has an incentive to participate [Parkes, 2001], that is, she can ensure to always get nonnegative utility. In $\mathsf{SL}[\mathcal{F}]$ we encode (ex-post) individual rationality [Nisan *et al.*, 2007] through the formula $\mathrm{IR}(\boldsymbol{\theta}) := \bigwedge_{a\in\mathrm{Ag}} 0 \leq \mathrm{util}_a(\theta_a)$.

A mechanism is *efficient* (EF) if it chooses the alternative maximizing the social welfare, i.e.the cumulative of the agent's valuations over the assigned choice [Parkes, 2001]. In $\mathsf{SL}[\mathcal{F}]$, we can write this condition with the formula $\mathrm{EF}(\boldsymbol{\theta}) := \sum_{a\in\mathrm{Ag}} \mathsf{v}_a(\mathrm{choice}, \theta_a) = \mathrm{maxv}_{\boldsymbol{\theta}}$, where $\mathrm{maxv}_{\boldsymbol{\theta}} = \max_{x\in\mathcal{X}} \sum_{a\in\mathrm{Ag}} \mathsf{v}_a(x, \theta_a)$ is a constant in $\mathcal{F}$. In a terminal position, it means that the social welfare is maximal.

To express that a mechanism satisfies a property we need to capture its equilibria. A strategy profile $\sigma = (\sigma_a)_{a\in\mathrm{Ag}}$ is a *Nash equlibrium* (NE) if no agent can increase her utility with a unilateral change of strategy [Parkes, 2001]. The following $\mathsf{SL}[\mathcal{F}]$-formula characterizes Nash equilibria:

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

where $\boldsymbol{s} = (s_a)_{a\in\mathrm{Ag}}$ is a profile of strategy variables.

The following formula maximizes the value of φ in the terminal positions of all Nash equilibria:

$$\text{Max-Nash}(\varphi, \boldsymbol{\theta}) := \exists \boldsymbol{s}.\, \mathrm{NE}(\boldsymbol{s}, \boldsymbol{\theta}) \wedge \mathbf{F}(\mathrm{ter} \wedge \varphi)$$

We now use these formulas to illustrate our approach to mechanism synthesis. To avoid detailing tie-breaking rules, in the examples we assume agents have distinct types, that is $\theta_a \neq \theta_b$ for any $b \neq a$ and $\boldsymbol{\theta} \in \boldsymbol{\Theta}$. Given an agent a, we let $\mathrm{wins}_a \in (-1, 1]$ be a constant value denoting the choice in which a is the winner, with $\mathrm{wins}_a \neq \mathrm{wins}_b$ for any $b \neq a$. We consider the choice set $\mathcal{X} = \{\mathrm{wins}_a : a \in \mathrm{Ag}\} \cup \{-1\}$, 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 $x = \mathrm{wins}_a$, and $\mathsf{v}_a(\theta_a, x) = 0$ otherwise. That is, the valuation of an agent depends only on her type and whether she won.

4.2 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 in bids representing discrete monetary values bounded by the participants' budget. In this section, we illustrate the synthesis problem with such restriction by considering rules based on the Japanese auction.

Example 1. 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 [Klemperer, 1999]. Let us fix a price increment $\mathsf{inc} > 0$. There are only two possible actions, accept (acc) or decline (dec), so that the set $\mathrm{Ac} = \{\mathrm{acc}, \mathrm{dec}\}$ is indeed bounded. Furthermore, we let $\mathrm{AP} = \{\mathrm{price}, \mathrm{sold}, \mathrm{init}, \mathrm{choice}, \mathrm{bid}_a, \mathrm{pay}_a, \mathrm{ter} : a \in \mathrm{Ag}\}$, where price denotes the current price, init denotes whether the position is the initial one, sold specifies whether the item was sold, and bid_a specifies whether a is an active bidder.

The following $\mathsf{SL}[\mathcal{F}]$-formulae are a partial description of a mechanism, inspired by the Japanese auction. The meaning of Rules J1-J8 is intuitive. Similar rules for encoding auctions through a logic-based language can be seen in [Mittelmann and Perrussel, 2020]. Rule J9 specifies that for all type profiles there should exist a NE whose outcome is IR and EF.

J1. $\mathbf{AG}((\text{init} \rightarrow \text{price} = 0 \wedge \neg\text{ter}) \wedge (\mathbf{XG}\neg\text{init} \wedge \mathbf{F}\ \text{ter}))$

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

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

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

J5. $\mathbf{AG}(\text{choice} = \text{wins}_a \leftrightarrow \text{bid}_a \wedge \bigwedge_{b\neq a} \neg\text{bid}_a)$

J6. $\mathbf{AG}(\text{choice} = -1 \leftrightarrow \neg(\bigvee_{a\in\text{Ag}}(\text{bid}_a \wedge \bigwedge_{b\neq a} \neg\text{bid}_a)))$

J7. $\mathbf{AG}\big(\bigwedge_{a\in\text{Ag}}(\text{choice} = \text{wins}_a \rightarrow \text{pay}_a = \text{price})\big)$

J8. $\mathbf{AG}\big(\bigwedge_{a\in\text{Ag}}(\text{choice} \neq \text{wins}_a \rightarrow \text{pay}_a = 0)\big)$

J9. $\bigwedge_{\boldsymbol{\theta}\in\boldsymbol{\Theta}} \text{Max-Nash}(\text{IR}(\boldsymbol{\theta}) \wedge \text{EF}(\boldsymbol{\theta}), \boldsymbol{\theta})$

We denote by Σ_{jpn} the conjunction of Rules J1-J9. 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.

Proposition 2. *There exists a wCGS-mechanism $\mathcal{G}_{jpn}$ such that $[\![\Sigma_{jpn}]\!]^{\mathcal{G}_{jpn}} = 1$.*

Such an optimal mechanism is produced by Algorithm 1.

4.3 Turn-based mechanisms

In a number of mechanisms, such as picking sequences, agents play in turns. Hereby, we exemplify the synthesis of a turned-based mechanism. In this example we also show how Algorithm 1 can be used to maximize the social welfare.

Example 2. The English auction is an ascending auction in which the participants are allowed to outbid the last bidder by proposing a 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 [Nisan *et al.*, 2007].

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 $\text{AP} = \{\text{price}, \text{init}, \text{choice}_a, \text{bid}_a, \text{tu}_a, \text{pay}_a, \text{ter} : a \in \text{Ag}\}$, where price denotes the current price, init denotes whether the position is the initial one, tu_a specifies whether it is a's turn and bid_a specifies the value of a's bid.

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

E1. $\mathbf{AG}(\text{init} \rightarrow \text{tu}_{a_1} \wedge \neg\text{tu}_{a_2} \wedge \neg\text{ter} \wedge \bigwedge_{a\in\text{Ag}}(\text{bid}_a = 0))$

E2. $\mathbf{AG}(\text{price} = \max(\text{bid}_{a_1}, \text{bid}_{a_2}))$

E3. $\mathbf{AG}\big(\neg\text{ter} \rightarrow \bigwedge_{a\in\text{Ag}}(\text{tu}_a \rightarrow \neg\mathbf{X}\text{tu}_a \wedge \neg\text{tu}_a \rightarrow \mathbf{X}\text{tu}_a)\big)$

E4. $\mathbf{AG}\big(\neg\text{ter} \rightarrow \bigwedge_{a\in\text{Ag}}(\text{bid}_{-a} < \mathbf{X}\text{bid}_a \wedge \text{tu}_a \rightarrow \mathbf{X}\text{choice} = \text{wins}_a)\big)$

E5. $\mathbf{AG}\big(\neg\text{ter} \rightarrow \bigwedge_{a\in\text{Ag}}(\text{bid}_{-a} \geq \mathbf{X}\text{bid}_a \wedge \text{tu}_a \rightarrow \mathbf{X}(\text{choice} = \text{wins}_{-a}) \wedge \text{ter})\big)$

E6. $\mathbf{AG}\big(\bigwedge_{a\in\text{Ag}}(\text{bid}_a = 1 \wedge \text{tu}_a \wedge \mathbf{X}(\text{choice} = \text{wins}_a) \wedge \text{ter})\big)$

E7. $\text{Max-Nash}(\sum_{a\in\text{Ag}} \mathsf{v}_a(\text{choice}, \theta_a), \boldsymbol{\theta})$

The value of formula E7 is the social welfare in the best NE for type profile $\boldsymbol{\theta}$. Letting $\Sigma_{\text{eng}}(\boldsymbol{\theta})$ be the conjunction of Rules E1-E7 alongside with the payment rules in Σ_{jpn} (Rules J7-J8), we have:

Proposition 3. *There exists a wCGS-mechanism $\mathcal{G}_{eng}$ such that $[\![\Sigma_{eng}(\boldsymbol{\theta})]\!]^{\mathcal{G}_{eng}} = \max_{x\in\mathcal{X}} \sum_{a\in\text{Ag}} \mathsf{v}_a(x, \theta_a)$, for each type profile $\boldsymbol{\theta} \in \boldsymbol{\Theta}$.*

As a result, Algorithm 1 applied to $\bigwedge_{\boldsymbol{\theta}\in\boldsymbol{\Theta}} \Sigma_{\text{eng}}(\boldsymbol{\theta})$ returns a mechanism that satisfies all the rules E1-E7 and J7-J8, and in which the minimal social welfare in all possible type profiles is as high as possible.

Complexity The synthesis problem in these examples can be solved in 3-EXPTIME. The complexity is dominated by Rules J9 and E7, which express the existence of NE. 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.

Approximate mechanisms The well-known results of Green and Laffont (1979) and Myerson and Satterthwaite (1983) show the impossibility of defining mechanisms whose equilibrium is efficient while having strict balance of monetary transfers. The quantitative semantics of $\mathsf{SL}[\mathcal{F}]$ and Algorithm 1 enable synthesizing mechanisms that approximate efficiency by maximizing social welfare.

5 Conclusion

We propose a novel approach for AMD in which mechanisms can be automatically generated (or *synthesized*) from partial or complete specifications in a rich logical language. The great expressiveness of the specification language $\mathsf{SL}[\mathcal{F}]$ makes our approach of automated 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 turn. We achieve this thanks to reductions to the satisfiability problem for $\mathsf{BQCTL}^*[\mathcal{F}]$, which we prove to be decidable.

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 from the SL-fragments One-Goal SL [Mogavero *et al.*, 2017; Cermák *et al.*, 2015] and Simple-Goal SL [Belardinelli *et al.*, 2019], 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.

Acknowledgements

This research is partially supported by the ANR project AGAPE ANR-18-CE23-0013, the PRIN project RIPER (No. 20203FFYLK), the JPMorgan AI Faculty Research Award "Resilience-based Generalized Planning and Strategic Reasoning", and the EU ICT-48 2020 project TAILOR (No. 952215).

References

[Ågotnes *et al.*, 2007] T. Ågotnes, W. Van Der Hoek, J. A. Rodríguez-Aguilar, C. Sierra, and M. J. Wooldridge. On the logic of normative systems. In *IJCAI*, 2007.

[Alur *et al.*, 2002] R. Alur, T.A. Henzinger, and O. Kupferman. Alternating-time temporal logic. *J. ACM*, 49(5):672–713, 2002.

[Asselin *et al.*, 2006] F. Asselin, B. Jaumard, and A. Nongaillard. A technique for large automated mechanism design problems. In *IAT*, 2006.

[Belardinelli *et al.*, 2019] F. Belardinelli, W. Jamroga, D. Kurpiewski, V. Malvone, and A. Murano. Strategy logic with simple goals: Tractable reasoning about strategies. In *IJCAI*, 2019.

[Bouyer *et al.*, 2019] P.Bouyer, O.Kupferman, N.Markey, B. Maubert, A.Murano, and G.Perelli. Reasoning about quality and fuzziness of strategic behaviours. In *IJCAI*, 2019.

[Bulling and Dastani, 2016] N. Bulling and M. Dastani. Norm-based mechanism design. *Artificial Intelligence*, 239:97–142, 2016.

[Cermák *et al.*, 2015] P. Cermák, A. Lomuscio, and A. Murano. Verifying and synthesising multi-agent systems against one-goal strategy logic specifications. In *AAAI*, 2015.

[Chatterjee *et al.*, 2010] K. Chatterjee, T. A. Henzinger, and N. Piterman. Strategy Logic. *Inf. Comput.*, 208(6):677–693, 2010.

[Clarke *et al.*, 2018] E.M.Clarke, O.Grumberg, D.Kroening, D. Peled, and H. Veith. *Model checking*. MIT press, 2018.

[David and Kroening, 2017] C. David and D. Kroening. Program synthesis: challenges and opportunities. *Phil. Trans. Royal Society A*, 375(2104):20150403, 2017.

[Dütting *et al.*, 2019] P. Dütting, Z. Feng, H. Narasimhan, D. Parkes, and S. S. Ravindranath. Optimal auctions through deep learning. In *ICML*, 2019.

[French, 2003] T. French. Quantified propositional temporal logic with repeating states. In *TIME*, 2003.

[Green and Laffont, 1979] J. R. Green and J. J. Laffont. *Incentives in Public Decision Making*. North-Holland, Amsterdam, 1979.

[Gutierrez *et al.*, 2019] J. Gutierrez, M. Najib, G. Perelli, and M. J. Wooldridge. Equilibrium design for concurrent games. In *CONCUR*, 2019.

[Klemperer, 1999] P. Klemperer. Auction theory: A guide to the literature. *Journal of Economic Surveys*, 13(3):227–286, 1999.

[Laroussinie and Markey, 2014] F. Laroussinie and N. Markey. Quantified CTL: expressiveness and complexity. *Log. Methods Comput. Sci.*, 10(4), 2014.

[Laroussinie and Markey, 2015] F. Laroussinie and N. Markey. Augmenting ATL with strategy contexts. *Inf. Comput.*, 245:98–123, 2015.

[Maubert *et al.*, 2021] B. Maubert, M. Mittelmann, A. Murano, and L. Perrussel. Strategic reasoning in automated mechanism design. In *KR*, 2021.

[Mittelmann and Perrussel, 2020] M. Mittelmann and L. Perrussel. Auction description language (ADL): a general framework for representing auction-based markets. In *ECAI*, 2020.

[Mogavero *et al.*, 2014] F. Mogavero, A. Murano, G. Perelli, and M. Y. Vardi. Reasoning about strategies: On the model-checking problem. *ACM Trans. Comput. Log.*, 15(4), 2014.

[Mogavero *et al.*, 2017] F. Mogavero, A. Murano, G. Perelli, and M. Y. Vardi. Reasoning about strategies: on the satisfiability problem. *Log. Methods Comput. Sci.*, 13(1), 2017.

[Myerson and Satterthwaite, 1983] R. B. Myerson and M. A. Satterthwaite. Efficient mechanisms for bilateral trading. *Journal of Economic Theory*, 29(2):265–281, 1983.

[Narasimhan *et al.*, 2016] H. Narasimhan, S. B. Agarwal, and D. C. Parkes. Automated mechanism design without money via machine learning. In *IJCAI*, 2016.

[Nisan *et al.*, 2007] N. Nisan, T. Roughgarden, É. Tardos, and V. Vazirani. *Algorithmic Game Theory*. Cambridge University Press, 2007.

[Niu *et al.*, 2012] J. Niu, K. Cai, S. Parsons, M. Fasli, and X. Yao. A grey-box approach to automated mechanism design. *Elec. Com. Research and App.*, 11(1):24–35, 2012.

[Okada *et al.*, 2019] N. Okada, T. Todo, and M. Yokoo. Sat-based automated mechanism design for false-name-proof facility location. In *PRIMA*, 2019.

[Parkes, 2001] D. Parkes. *Iterative combinatorial auctions: Achieving economic and computational efficiency*. Univ. of Pennsylvania Philadelphia, 2001.

[Pauly and Wooldridge, 2003] M. Pauly and M. Wooldridge. Logic for mechanism design–a manifesto. In *Workshop on Game Theory and Decision Theory in Agent Syst.*, 2003.

[Pin, 2021] Jean-Éric Pin. *Handbook of Automata Theory*. European Math. Society Publis. House, Zuerich, 2021.

[Pnueli and Rosner, 1989] A. Pnueli and R. Rosner. On the synthesis of a reactive module. In *POPL*, 1989.

[Sandholm, 2003] T. Sandholm. Automated mechanism design: A new application area for search algorithms. In *CP*, 2003.

[Shen *et al.*, 2019] W. Shen, P. Tang, and S. Zuo. Automated mechanism design via neural networks. In *AAMAS*, 2019.

[Troquard and Walther, 2012] N. Troquard and D. Walther. On satisfiability in ATL with strategy contexts. In *JELIA*, 2012.

[Vorobeychik *et al.*, 2007] Y. Vorobeychik, D. M. Reeves, and M. P. Wellman. Constrained automated mechanism design for infinite games of incomplete information. In *UAI*, 2007.

[Wooldridge *et al.*, 2007] M. Wooldridge, T. Agotnes, P. Dunne, and W. Van der Hoek. Logic for automated mechanism design-a progress report. In *AAAI*, 2007.