Abstraction-based Verification of Infinite-state Reactive Modules

Francesco Belardinelli[1] and **Alessio Lomuscio**[2]

Abstract. We introduce the formalism of infinite-state reactive modules to reason about the strategic behaviour of autonomous agents in a setting where data are explicitly exhibited in the systems description and in the specification language. Technically, we endow reactive modules with an infinite domain of interpretation for individual variables, and introduce FO-ATL, a first-order version of alternating time temporal logic, for the specification of properties of interest. We show that their verification is decidable for classes of data types of interest. This result is proved by defining a first-order version of alternating bisimulations and finite bisimilar abstractions. We illustrate the formal machinery by applying it to English and sealed bid auctions. In particular, we show that strategic properties of agents in auctions, including manipulability and collusion, can be expressed and verified in this framework.

1 Introduction

The formalism of alternating-time temporal logic (ATL) [5] has been widely used to reason about the strategic behaviour of agents in multi-agent systems (MAS) [2, 13]. Specifications in ATL can express the ability of agents to bring about particular states of affairs (represented as temporal formulas in linear-time temporal logic) in the system. Models for ATL are traditionally given in terms of concurrent game structures, alternating transition systems, or in variants of interpreted systems. An attractive feature of some of these semantics is that models can be given via compact representations [26]. For instance, programs in the interpreted system programming language (ISPL), supported by the MCMAS model checker [30], generate interpreted systems upon which ATL formulas can be evaluated. Alternatively, *reactive modules* have been put forward as a flexible framework to model relevant behaviours of distributed systems, including pure and observable asynchronicity, atomic and non-atomic synchronicity [4]. Moreover, this compact representation constitutes the basis of the programming language for the jMOCHA model checker [3]. Both ISPL and simple reactive modules denote finite-state systems.

Yet, it is crucial to be able to reason about multi-agent systems that are intrinsically associated with infinite-state models. These arise naturally in MAS programming, when, for example, the data type of a given variable is not finite. These requirements may appear also in the modelling phase, when one is unable to assign a bound to a particular modelling concept, e.g., a queue. In this paper we focus on rational real variables [20]. Programs normally generate infinite-state models, but their verification is prone to undecidability, at least in the most general case.

[1] Imperial College London, UK, email: f.belardinelli@imperial.ac.uk
[2] Imperial College London, UK, email: a.lomuscio@imperial.ac.uk

In this paper we introduce *infinite-state reactive modules*, an extension of reactive modules to reason about the strategic behaviour of autonomous agents in a setting where data are explicitly exhibited in the systems description and in the specification language. We show that while their execution model is infinite, their model checking problem is decidable. Specifically, we endow reactive modules with an infinite domain of interpretation for individual variables, as well as relational symbols for total orders. Then, to express strategic behaviours of modules, we introduce FO-ATL, a first-order version of alternating time temporal logic, suitable for representing explicitly the data content of modules. We prove the decidability of the model checking problem for this setting by introducing a novel, first-order version of alternating bisimulation. We show that although reactive modules are infinite-state systems in general, for specific classes of data types we can construct a finite abstraction, which is bisimilar to the concrete, infinite-state system. As a result, the verification procedure can be conducted on the finite abstraction, and the result transferred to the original system. We illustrate the interest and workings of the formal machinery through an application to two auction mechanisms: English ascending bid auctions and repeated sealed bid auctions. In particular, we show that strategic properties of agents in auctions, including manipulability and collusion, can be expressed and verified in this framework.

Related Work. The area of logics for reasoning about strategies has witness a steady growth in recent years [5, 15, 31]. Here we consider only the contributions most closely related to the present setting. The inspiration for this work comes from the original paper on reactive modules [4], even though here we consider their *simple* version introduced in [33], where the *implicit* model checking problem is analysed, even though in a purely propositional setting.

This paper builds on a stream of results on the verification of *data-aware systems* [18, 11, 7, 10, 22], i.e., systems whose execution depends crucially on their data content. In these works a data model is coupled with an update mechanism that determine the system's evolution. For instance, in the line of [7, 14] the data model is represented by means of description logics, while the system evolves in response to conjunctive queries. In [22, 17] the formalism of situation calculus is exploited to represent the data model and update mechanisms. In this line of research, including [10, 23], the model checking problem is proved to be decidable by using two key features of these systems: *boundedness* (only a bounded number of individuals is active at each state in the system's execution) and *uniformity* (the system's execution is determined only by the elements that are named explicitly in the system's description). These properties are shared also by reactive modules. However, our contribution differs from previous works in several aspects. First, we adopt a modular, agent-based approach to data-aware systems, while in [18, 11, 7, 22] systems are described monolithically. This feature is key to model distributed scenario, such as auctions, where agents have only imperfect knowledge of the system's global state. This is also reflected in the specification language: here we build upon ATL, a logic for strategies of individuals and coalitions, while previous works have focused on CTL, LTL [19], and μ-calculus, that account only for the system's global behaviour. Aspects of agency and individual knowledge have been analysed in [10], but protocols and actions are given completely abstractedly therein, while here we provide a computational semantics grounded on reactive modules.

Furthermore, this paper contributes towards the formal verification of auction-based mechanisms, which is a topic of growing interest in the AI community [24, 35, 34]. However, with some notable exceptions, most of the research in this area has focused on the design of auctioning mechanisms and the analysis of their formal properties, while the automated verification of these designs has only partially been addressed, and only for specific classes of auctions, by using purpose-built formalisms [6]. Here we put forward a principled approach to the verification of infinite-state systems that can handle general classes of auctions as well.

Scheme of the Paper. In Section 2 we introduce infinite-state (agent) modules, the first-order specification language FO-ATL, define the semantics of infinite-state reactive modules systems (IRMS) and the corresponding model checking problem. We exemplify and motivate the technical notions in Section 3, where we briefly describe modules for English auctions and repeated sealed bid auctions. Section 4 is devoted to the main result of this paper: decidability through finite bisimilar abstractions, which is then applied in Section 5 to our auctioning mechanisms. We conclude in Section 6 with discussion and future work.

2 Infinite-state Reactive Modules

In this section we introduce a generalisation of reactive modules [4, 33], that admits variables with an infinite domain of interpretation, possibly totally ordered (e.g., natural, rational, and real numbers). The specifications for these systems will be given in an expressive first-order extension of alternating-time temporal logic [5], also defined here.

In the following we assume a finite set $\mathcal{T} = \{T_1, \ldots, T_k\}$ of *types* (e.g. booleans, integers, rationals, etc.), each endowed with a (possibly ordered) *interpretation domain* D_T. Also, for each type T we consider a set $V_T = \{v_0, v_1, \ldots\}$ of *variables* and a set $P_T = \{x_0, x_1, \ldots\}$ of *parameters*. We use V (resp. P) to denote $\bigcup_{T \in \mathcal{T}} V$ (resp. $\bigcup_{T \in \mathcal{T}} P$). Intuitively, variables are used to describe the data model, while parameters appear in formulas.

Further, we introduce a set Ag of *agent modules* (or simply agents), each comprising of a set L of *local states*, a set Act of *actions*, and a *protocol function* Pr, according to the formal account of agents in the literature on interpreted systems [21]. In line with reactive systems [4], we assume that each agent module $m \in Ag$ controls a *finite* set $cnt_m \subseteq V$ of variables. Specifically, $\{cnt_1, \ldots, cnt_{|Ag|}\}$ form a partition of V, that is, every variable in V is controlled by exactly one agent. Hence, the set V can be assumed to be finite as well. Next, the set cnt_m of variables controlled by module m is partitioned into the sets $priv_m$ and $intf_m$ of *private* and *interface* variables respectively: private variables are only accessible to owner m, while interface variables are readable, but not writable, by any other agent. Given private and interface variables for a set Ag of agent modules, the variables in obs_m observable by agent module m are comprised of her controlled variables and the interface variables of all agents, i.e., $obs_m = cnt_m \cup \bigcup_{j \neq m} intf_j$, or equivalently, $obs_m = priv_m \cup \bigcup_{j \in Ag} intf_j$. Observe that by considering controlled, private, and interface variables, in [4] the authors are able to model a number of different behaviours for reactive systems, including pure asynchronicity (interleaving), observable asynchronicity, atomic and non-atomic synchronicity. For our purposes, we will use private and interface variables to model partial observability and imperfect information of agents in auctions.

To provide a formal account of the local state of an agent module, we introduce *local interpretations* as functions $\theta_m : cnt_m \to D$, i.e., (finite, type-consistent) interpretations of the variables in cnt_m with values in D. For simplicity, in the following we often identify an interpretation θ_m with its range $\theta_m(cnt_m) \subseteq D$, whenever domain cnt_m is clear by the context. Then, provided interpretation $\theta_1, \ldots, \theta_{Ag}$ for all agent modules, the local states $l \in L$ of agent module m is comprised of the values for her observed variables in obs_m, i.e., $l = \theta_m \cup \bigcup_{j \neq m} \theta_j(intf_j)$ by definition. Any local state l_m is assumed to characterise the knowledge of agent module m. Notice that l is well-defined as $\{cnt_i, \ldots, cnt_{|Ag|}\}$ is a partition. Moreover, since the domain D is infinite in general, the set L of local states is infinite as well.

To define the individual actions in Act and the protocol Pr for each agent module, we introduce a typed first-order language built on variables, parameters and relational symbols $=$ and $\leq$ when appropriate.

Definition 1 (FO-formulas) *First-order formulas over types $\mathcal{T}$ are defined according to the following BNF:*

$$\phi \quad ::= \quad z = z' \mid z \leq z' \mid \neg\phi \mid \phi \to \phi \mid \forall x \phi$$

where $z, z' \in V \cup P$ have the same type, and $x \in P$.

The symbols $\neq$, $<$, $\geq$, $\top$, $\bot$, connectives $\wedge$, $\vee$, quantifier $\exists$, and free and bound variables and parameters are defined as standard [25]. Notice that quantification applies to parameters only, this is in accordance with the intuition above on the use of variables and parameters. Throughout the paper we assume that types are manipulated consistently, without explicitly mentioning this fact each time.

Following [33], we now introduce a particular notion of action.

Definition 2 (Guarded Command) *A* guarded command γ *over V and P is an expression*

$$g(x_1, \ldots, x_k) \quad \rightsquigarrow \quad v_1 := x_1; \ldots; v_k := x_k$$

where (i) *guard g is an FO-formula with free parameters among $x_1, \ldots, x_k$;* (ii) *all v_i are variables in V; and* (iii) *in each assignment $v_i := x_i$, v_i and x_i have the same type.*

As customary [33], we require that no variable v_i appears on the left-hand-side of two assignments in the same guarded command (hence no issue on the ordering of updates arises). Also, the sets of parameters appearing in the various commands are assumed to be disjoint. The intuitive meaning of a guarded command is that if guard g evaluates to true for some interpretation $\sigma : P \to D$ of parameters, then the command is enabled for execution. By executing the command we set each variable v_i to value $\sigma(x_i) \in D$. We say that $v_1, \ldots, v_k$ are the variables controlled by γ, and denote this set by $ctr(\gamma)$, while the variables in g are the *observable* variables $obs(\gamma)$. A set of guarded commands is *disjoint* if their controlled variables are mutually disjoint. In particular, the *skip* command can be represented as $\top \rightsquigarrow \epsilon$, where ϵ is the empty sequence.

We now have all preliminary notions necessary to introduce agent modules.

Definition 3 (Agent Module) *An* agent module *is a tuple* $m = \langle ctr, init, update\rangle$ *where*

- *$ctr \subseteq V$ is the (finite) set of variables controlled by m, partitioned into sets $priv$ and $intf$;*
- *$init$ is a (finite) set of initialisation guarded commands s.t. for all $\gamma \in init$, $ctr(\gamma) \subseteq ctr$ and $obs(\gamma) \subseteq obs$;*
- *$update$ is a (finite) set of update guarded commands s.t. for all $\gamma \in update$, $ctr(\gamma) \subseteq ctr$ and $obs(\gamma) \subseteq obs$.*

According to Def. 3, an agent module initialises the variables she controls according to her guarded commands in $init$, then the same variables are updated following the commands in $update$. In particular, controllability and observability of variables in guards and assignments has to be respected. Given an agent module m, we denote the initialisation and update commands of m by $init_m$ and $update_m$ respectively.

Next, we define the *global state* of a reactive system as a tuple $s = \langle\theta_1, \ldots, \theta_{|Ag|}\rangle$, where each θ_m is an interpretation for agent m. Equivalently, global states can be represented as functions $s : V \to D$, i.e., (finite, type-consistent) interpretations of the variables in V with values in D such that for every $v \in V$, $s(v) = \theta_m(v)$, where m is the agent controlling v. As anticipated above, any state s is well-defined as V is partitioned among the agents in Ag. Further, given a global state s, we denote as $l_1, \ldots, l_{|Ag|}$ the corresponding local states for all agents in Ag. Observe that $\langle\theta_1, \ldots, \theta_{|Ag|}\rangle$ and $\langle l_1, \ldots, l_{|Ag|}\rangle$ are equivalent representation of a global state s, in terms of controlled, respectively observable, variables. So, we will use the two notations interchangeably. We remarked that agent modules have only partial observability of the global state of the system. Specifically, two states s and s' are *indistinguishable* for agent module m, or $s \sim_m s'$, iff $l_m = l'_m$, that is, iff s and s' coincide on the interpretation of obs_m. We denote the set of all global states as $\mathcal{G}$.

To introduce the semantics of guarded commands formally, we define the satisfaction relation $\models$ for FO-formulas. An FO-formula ϕ is given meaning by a *finite interpretation* $\sigma : fr(\phi) \to D$ that assigns values in D to the free parameters in ϕ. A *reinterpretation* σ^x_u coincides with σ, but assigns value $u \in D$ to parameter $x \in fr(\phi)$. By Σ we denote the set of all interpretations σ for parameters. Further, given $z \in V \cup P$, $(s, \sigma)(z) = s(z)$ for $z \in V$, and $(s, \sigma)(z) = \sigma(z)$ for $z \in P$, that is, variables are interpreted according to s, while parameters according to σ.

Definition 4 (Satisfaction) *A state s* satisfies *an FO-formula ϕ for a finite interpretation σ, or $(s, \sigma) \models \phi$, iff (clauses for propositional connectives are immediate and thus omitted)*

$$\begin{array}{lll} (s,\sigma) \models z = z' & \textit{iff} & (s,\sigma)(z) = (s,\sigma)(z') \\ (s,\sigma) \models z \leq z' & \textit{iff} & (s,\sigma)(z) \leq (s,\sigma)(z') \\ (s,\sigma) \models \forall x\phi & \textit{iff} & \textit{for all } u \in s(V),\ \sigma^x_u \models \phi \end{array}$$

The interpretation of FO-formulas is completely standard, but for quantification that takes values from the finite set $s(V) = \{u \in D \mid u = s(v) \text{ for some } v \in V\}$ of images of variables in V. This is consistent with the interpretation of quantification on *active domains* in database theory [1]. Indeed, at this stage quantification can be considered syntactic sugar, as $s(V)$ is finite. However, once the temporal evolution of reactive modules is taken into account – as we shall see shortly – quantification makes the specification language strictly more expressive than its propositional counterpart.

Since the set V of variables and the set P of parameters appearing free in any guarded command are both finite and defined at design-time, we deem them fixed. Hence, hereafter we will always consider suitable states s and finite interpretations σ. Further, when evaluating guards for commands of an agent module m, it is sufficient to look at the interpretation of observable variables provided by m's local state l_m. Therefore, we can introduce the satisfaction relation $(l_m, \sigma) \models \phi$, for $fr(\phi) \subseteq obs_m$, in analogy with Def. 4.

Finally, an *infinite-state reactive module system* (IRMS) is defined as a set $M = \{m_1, \ldots, m_{|Ag|}\}$ of agent modules. Given an IRMS M, the sets Act and ACT of *individual* and *joint actions*, the set I of *initial states*, the *protocol* Pr, the *transition function* τ, and the set S of *reachable states* are defined as follows:

- for every agent module m, $Act = update$; while ACT is the set of tuples of update commands $(\gamma_1, \ldots, \gamma_{|Ag|})$, for $\gamma_m \in update_m$;
- I is the set of states s' such that for every $v \in V$, $s'(v) = \sigma(x)$ for some state s, interpretation σ, and initialisation command $\gamma = (\gamma_1, \ldots, \gamma_{|Ag|})$, such that each local state satisfies the corresponding guard, i.e., $(l_m, \sigma) \models g_m$;
- $Pr_m : L \to ((2^{Act_m} \setminus \emptyset) \times \Sigma)$ such that $Pr_m(l) = \{(\gamma, \sigma) \mid (l, \sigma) \models g_\gamma\}$;
- $\tau : \mathcal{G} \times ACT \times \Sigma \to \mathcal{G}$ such that $\tau(s, \gamma, \sigma) = s'$ iff *(i)* for all $m \in M$, $(\gamma_m, \sigma_m) \in Pr_m(l_m)$; and *(ii)* $s'(v_i) = \sigma(x_i)$. Often we write $s \xrightarrow{\gamma,\sigma} s'$ for $\tau(s, \gamma, \sigma) = s'$;
- S is the closure of I according to the transition function τ.

The definitions above provide the computational counterpart to the notions of action and protocol introduced earlier. Specifically, an agent module m can update the variables she controls by means of actions in Act, according to protocol Pr, which returns the actions whose guard is satisfied in her local state l_m for some interpretation σ of parameters. Overall, an IRMS M describes the evolution of a reactive system from an initial state $s \in I$, according to the transition function τ, which returns the successive global state provided the current state and an enabled joint action (including its data content σ). Again, since the domain D is infinite in general, IRMS are infinite-state systems, differently from [33].

Since we assumed that the sets of parameters in the various commands are disjoint, in the definition of τ the restriction σ_m of interpretation σ to the parameters in $update_m$ is well-defined. Hereafter we make use of a notion of *extension* $\sigma \subseteq \sigma'$ between interpretations, viewed as functions. Hence, above we have that for every $m \in M$, $\sigma_m \subseteq \sigma$.

To specify the behaviour of IRMS and to reason about the strategic abilities of agent modules, we introduce a first-order version of alternating-time temporal logic.

Definition 5 (FO-ATL) *Formulas in first-order ATL are defined in BNF as follows:*

$$\psi ::= \phi \mid \neg\psi \mid \psi \wedge \psi \mid \forall x\psi \mid \langle\langle C\rangle\rangle X\psi \mid \langle\langle C\rangle\rangle(\psi U \psi) \mid \langle\langle C\rangle\rangle G\psi$$

where ϕ is an FO-formula and $C \subseteq M$ is coalition of agents.

The meaning of ATL operators is standard: a formula $\langle\langle C\rangle\rangle\Phi$ says that *coalition C has a (collective) strategy to achieve* Φ. Again, quantification is defined on parameters only. Notice that in FO-ATL we can have arbitrary alternations of quantifiers and ATL operators. A consequence of this, as we shall see, is that quantification in FO-ATL is not syntactic sugar. Also, this is in contrast with previous works [7, 10, 22] that consider modalities (e.g., CTL, LTL, μ-calculus) capable of expressing only the temporal evolution of the system, but do not support naturally the specification of strategic abilities of agents.

To interpret FO-ATL formulas, we introduce a suitable notion of *local strategy*.

Definition 6 (Strategy) *An imperfect information, memoryless strategy (henceforth simply a strategy) for an agent module $m \in M$ is a function $f_m : L \to Act_m$ such that for every local state $l \in L$, $(f_m(l), \sigma) \in Pr_m(l)$ for some interpretation $\sigma \in \Sigma$.*

We can check that strategies, as introduced in Def. 6, are *uniform* in the sense of [28], as they only depend on the local state of agents. In particular, if $s \sim_m s'$, then $l_m = l'_m$ by definition, and therefore $f_m(l_m) = f_m(l'_m)$.

Given an IRMS M, a *path* λ is an infinite sequence $s_0 s_1 \ldots$ of states, in which $\lambda(i)$ denotes the $i+1$-th element s_i of λ. Further, for a set $F_C = \{f_m \mid m \in C\}$ of strategies, a path λ is F_C*-compatible* iff for every $j \geq 0$, $\lambda(j+1) = \tau(\lambda(j), \gamma, \sigma)$ for some joint action γ and interpretation σ such that *(i)* for $m \in C$, $\gamma_m = f_m(\lambda(j)_m)$; and *(ii)* for $m \notin C$, $(\gamma_m, \sigma_m) \in Pr_m(\lambda(j)_m)$. We denote the set of F_C-compatible paths from state s as $out(s, F_C)$.

In the following definition we assume that the sets of parameters appearing in commands and in formula ϕ are disjoint. This can be done without loss of generality, as both sets are finite and defined at design-time.

Definition 7 (Satisfaction) *Given an IRMS M, a state s* satisfies *an FO-ATL formula ψ for interpretation σ, or $(M, s, \sigma) \models \psi$, iff (clauses for propositional connectives are immediate and thus omitted).*

$$\begin{array}{ll}
(M,s,\sigma) \models \phi & \textit{iff } (s,\sigma) \models \phi, \textit{ where } \phi \textit{ is an FO-formula} \\
(M,s,\sigma) \models \forall x \psi & \textit{iff for every } u \in s(V),\ (M,s,\sigma^x_u) \models \psi \\
(M,s,\sigma) \models \langle\langle C \rangle\rangle X \psi & \textit{iff for some strategy } F_C, \textit{ for all } \lambda \in out(s,F_C), \\
 & \quad (M,\lambda(1),\sigma) \models \psi \\
(M,s,\sigma) \models \langle\langle C \rangle\rangle G \psi & \textit{iff for some strategy } F_C, \textit{ for all } \lambda \in out(s,F_C), \\
 & \quad \textit{for all } i \geq 0,\ (M,\lambda(i),\sigma) \models \psi \\
(M,s,\sigma) \models \langle\langle C \rangle\rangle (\psi U \psi') & \textit{iff for some strategy } F_C, \textit{ for all } \lambda \in out(s,F_C), \\
 & \quad \textit{for some } i \geq 0,\ (M,\lambda(i),\sigma) \models \psi', \textit{ and} \\
 & \quad \textit{for all } j,\ 0 \leq j < i \textit{ implies } (M,\lambda(j),\sigma) \models \psi
\end{array}$$

We remark that the semantics of ATL operators in Def. 7 is standard, while quantification ranges on the active domain $s(V)$. However, differently from Def. 4, quantification is not syntactic sugar: transitions might take us to a successor state s', in which an individual $u \in s(V)$ is no longer active, i.e., $u \notin s'(V)$. As a consequence, quantification in FO-ATL gives us a language that is strictly more expressive than propositional ATL, as it allows to refer to individuals across states. This feature of FO-ATL will become apparent in Section 3.

An FO-ATL formula ψ is *true* in state s, or $(M, s) \models \psi$, iff for all interpretations σ, $(M, s, \sigma) \models \psi$; ψ is *true* in M, or $M \models \psi$, iff for all initial states $s \in I$, $(M, s) \models \psi$. We can now state the model checking problem for infinite-state reactive module systems against FO-ATL.

Definition 8 (Model Checking) *Given an IRMS M and an FO-ATL formula ψ, the* model checking problem *concerns determining whether $M \models \psi$.*

Notice that M is an infinite-state system, and the model checking problem for infinite-state data-aware systems is normally undecidable [18]. However, in what follows we define an abstraction-based technique to obtain decidability. First we present some instances of IRMS.

3 Auctions

In this section we illustrate the formal machinery introduced in Section 2 with examples from the literature on auctions. Specifically, we model English ascending bid auctions and repeated sealed auctions as infinite-state reactive module systems, and specify the behaviour of agents participating in the corresponding IRMS by means of FO-ATL formulas. We provide an informal description of these auctioning mechanisms and refer to [20] for further details.

In English auctions several bidders bid for an item auctioned by the auctioneer. All the participating agents can be represented as modules, beginning with the auctioneer.

Definition 9 (Auctioneer) *The auctioneer module $m_a = \langle ctr_a, init_a, update_a \rangle$ is such that*

- *$ctr_a = \{base, t\_out\} = inft_a$, while $priv_a = \emptyset$. Variable t_out has type boolean, while base ranges over the rational numbers.*
- *$init_a$ contains guarded commands:*

$$\top \quad \rightsquigarrow \quad base := x_1; t\_out := \bot$$

- *$update_a$ contains guarded commands skip and*

$$\begin{array}{rcl}
t\_out = \bot & \rightsquigarrow & t\_out := \top \\
t\_out = \top & \rightsquigarrow & base := x_2; t\_out := \bot
\end{array}$$

Intuitively, the auctioneer module keeps track of the base price $base$ for the auctioned item (given as a rational), and owns a boolean variable t_out to terminate non-deterministically the bidding round (both are public). At the start of the execution the auctioneer initialises the base price $base$ to a random rational number x_1 and t_out to false ($\bot$). Then, by using the updates, she can either do nothing or terminate the bidding round, and then start a new one, with a different base price x_2 for a possibly different item.

The modules for bidders can be given as follows.

Definition 10 (Bidder) *The bidder module $m_i = \langle ctr_i, init_i, update_i \rangle$ is such that*

- *$ctr_i = \{tvalue_i, bid_i\}$ with $inft_i = \{bid_i\}$ and $priv_i = \{tvalue_i\}$. Both $tvalue_i$ and bid_i range over rational numbers.*
- *$init_i$ contains guarded commands:*

$$\top \quad \rightsquigarrow \quad bid_i := \mathrm{uu}; tvalue_i := x_3$$

- *$update_i$ contains guarded commands skip and*

$$\begin{array}{rcl}
(t\_out = \bot) \wedge \bigwedge_{j \in M} (bid_j = \mathrm{uu}) \wedge (x_4 \leq tvalue_i) & \rightsquigarrow & bid_i := x_4 \\
(t\_out = \bot) \wedge \bigvee_{j \neq i} (bid_i < bid_j) \wedge & & \\
\bigwedge_{j \neq i} (bid_j \neq \mathrm{uu} \to bid_j < x_5) \wedge (x_5 \leq tvalue_i) & \rightsquigarrow & bid_i := x_5 \\
t\_out = \top & \rightsquigarrow & bid_i := \mathrm{uu}; \\
 & & tvalue_i := x_6
\end{array}$$

By Def. 10 every bidder i has a *private* true value $tvalue_i \in \mathbb{Q}$, up to which she is happy to bid, and a *public* $bid_i \in \mathbb{Q}$. At the beginning she initialises her true value, while her bid is set to 'undefined'. Thereafter, she might choose to bid and then update it according to the other bidders' offers. At the end of the bidding round, she reinitialises her true value for a new round.

Given the auctioneer and bidder modules as defined above, an IRMS for an English auction is a set $M = \{m_a, m_1, \ldots, m_n\}$ of modules for the auctioneer a and bidders $b_1, \ldots, b_n$. Since base prices, true values, and bids all take rationals as values, M is actually an infinite-state system. Notice that IRMS M is non-terminating, as agents can skip indefinitely. We can eliminate such behaviours by introducing fairness constraints. Also, new items are put on auction, thus bidders can take part in successive auctions.

As a further example of the expressivity of IRMS we present modules for a repeated sealed auction, in which, differently from above, the winning bid is used to provide feedback on the value of the base price and true values for the next bidding round. This scenario is inspired to real-time bidding, where this feedback is provided by complex algorithms [32]. Given the limited expressivity of our specification language, here we consider a much simpler mechanism.

We start with a new module for the auctioneer.

Definition 11 (Auctioneer) *The auctioneer module* $m_a = \langle ctr_a, init_a, update_a\rangle$ *is such that*

- $ctr_a = \{base, t\_out, w\_bid\} = intf_a$*, while* $priv_a = \emptyset$*, where* w_bid *has type rational;*
- $init_a$ *contains guarded commands:*

$$\top \quad \rightsquigarrow \quad base := x_7; t\_out := \bot; w\_bid := \text{uu};$$

- *update contains guarded commands*

$$\begin{array}{lcl} (t\_out = \bot) \wedge \bigvee_{i \in M} (bid_i = x_8) \wedge & & \\ \bigwedge_{j \neq i} (bid_j \neq \text{uu} \rightarrow bid_j \leq bid_i) & \rightsquigarrow & w\_bid = x_8 \wedge t\_out := \top \\ (t\_out = \top) \wedge (x_9 \leq w\_bid) & \rightsquigarrow & base := x_9; t\_out := \bot \end{array}$$

By Def. 11 bidding rounds are only one-step long (given by toggling t_out from false to true), in line with sealed auctions. Further, the auctioneer keeps track of the highest (winning) bid w_bid, and makes use of this value as upper bound when setting the new base price for the next bidding round.

As regards bidders, the corresponding module is as follows:

Definition 12 (Bidder) *The bidder module* $m_i = \langle ctr_i, init_i, update_i\rangle$ *is such that component* $init_i$ *is given as above, while*

- $ctr_i = \{tvalue_i, bid_i\} = priv_i$ *and* $intf_i = \emptyset$*;*
- $update_i$ *contains guarded commands skip and*

$$\begin{array}{lcl} (t\_out = \bot) \wedge (x_9 \leq tvalue_i) & \rightsquigarrow & bid_i := x_9 \\ (t\_out = \top) \wedge (w\_bid \leq x_{10}) & \rightsquigarrow & bid_i := \text{uu}; tvalue_i := x_{10} \end{array}$$

Notice that, differently from Def. 10, now bids are private, all bidders submit them at the same time and cannot raise them, as it is customary in sealed bid auctions. Moreover, bidders use the winning bid as lower bound when setting the new true value for the next bidding round. We observe that the feedback provided by the winning bid is rather crude; more sophisticated mechanisms can be considered. Note that no quantification appears in the guards of commands for the auctioneer and bidders. In fact, we remarked above that quantification in guards is purely syntactic sugar. However, this is no longer the case when quantification in combined with ATL operators, as we now show.

Once we modelled auctions as IRMS, we might want to verify the behaviour of the auctioneer and bidders against properties written in FO-ATL. For instance, we might want to check that there is always one base price $base$ and each bidder i is associated with *at most* one defined true value $tvalue_i$ (possibly none). This can be expressed in FO-ATL as follows:

$$AG(\exists! x(base = x) \wedge \exists^{\leq 1} y(y \neq \text{uu} \wedge tvalue_i = y))$$

where the CTL operator AG is tantamount to $\langle\langle\emptyset\rangle\rangle G$, and quantifiers $\exists!$ and $\exists^{\leq 1}$ are defined as standard in first-order logic with identity.

Further, each bidder b_i can (has a strategy to) bid less or as much as her true value:

$$\langle\langle b_i\rangle\rangle G\ (bid_i \leq tvalue_i)$$

More interestingly, we can specify elaborate strategic abilities of agents. For instance, each bidder b_i can raise her bid unless she has already hit her true value:

$$AG\, \forall x(x = bid_i \rightarrow (x = tvalue_i \vee \langle\langle b_i\rangle\rangle F\, \exists y(y > x \wedge y = bid_i))) \quad (1)$$

Observe that in (1) the use of quantification on parameters allows us to compare values of bid_i at different moments of the system's execution. Such features are not expressible in a purely propositional language.

A crucial notion to analyse in auctions is *collusion*: does a certain coalition C of bidders have a strategy to win the auction, possibly by bidding a lower amount than 'normally necessary'? In order to express variants of this property in FO-ATL, we introduce a formula $win_i = t\_out \wedge \bigwedge_{j \neq i}(bid_j \leq bid_i)$, which intuitively says that bidder b_i is among the winners of the current auction (we may have multiple winner, but this issue is not relevant for the present discussion). The simplest form of collusion we can specify in FO-ATL states that coalition C can act so as to enforce that one of its members eventually wins:

$$\langle\langle C\rangle\rangle F \bigvee_{b_i \in C} win_i$$

Yet another key property analysed on auctions is *manipulability*: certain agents might exploit the auction design to their benefit. For instance, we might want to check that if bidder b_i has a strategy to win by bidding x, then, no matter what the other bidders do, b_i has a strategy to win in which x is strictly less than her true value:

$$AG\, \forall x(\langle\langle b_i\rangle\rangle(win_i \wedge bid_i = x) \rightarrow [\![Ag \backslash \{b_i\}]\!](win_i \wedge x < tvalue_i))$$

We conclude this section by observing that IRMS, together with FO-ATL, are a sound framework to represent various relevant types of auctions, as well as to specify interesting properties thereof. In the following section we study the problem of verifying these properties on infinite-state reactive module systems.

4 Decidability by Finite Abstraction

In this section we introduce abstractions of infinite-state reactive module systems, and show that, under specific assumptions, these abstractions are finite. Moreover, we prove a preservation result for FO-ATL specifications that allows us to verify an IRSM by model checking its finite abstraction. Here we use ideas from [10, 9], but contextualise them to ATL specifications. In the rest of the section we consider a *finite* abstract interpretation domain D^A_T for every type T. We start by considering a notion of abstract (global) state.

Definition 13 (Abstract Global State) *Given an IRMS* $M = \{m_1, \ldots, m_{|Ag|}\}$*, an* abstract global state *is a tuple* $s = \langle \theta_1, \ldots, \theta_{|Ag|}\rangle$ *of interpretations* $\theta_m : cnt_m \rightarrow D^A$*, together with a total order* $\leq_s$ *defined on each* D^A_T *(when appropriate).*

Notice that, differently from the concrete global states in Section 2, the total order $\leq_s$ depends on the particular abstract state s, rather than being defined on D_T.

Next, we introduce a notion of *isomorphism* between concrete and abstract states.

Definition 14 (Isomorphism) *A global state s and an abstract state s' are* isomorphic*, or $s \simeq s'$, iff for some type-consistent bijection $\iota : s(V) \mapsto s'(V)$, we have*

(i) *for every $m \in Ag$, $\theta'_m = \iota \circ \theta_m$;*
(ii) *ι preserves $\leq$, that is, for every $v, v' \in V$, $s(v) \leq s(v')$ iff $s'(v) \leq_{s'} s'(v')$.*

Any function ι as above is a witness *for $s \simeq s'$, or $s \overset{\iota}{\simeq} s'$ for short.*

Intuitively, isomorphic states share the same relational structure. Observe that, as regards order $\leq_{s'}$, witness ι preserves the interpretation of individuals in the active domain $s'(V)$ only. This feature of isomorphisms is key to obtain finite abstractions.

Now we show that isomorphic states satisfy the same FO-formulas ϕ. However, ϕ might contain free variables interpreted outside the active domain. This remark motivates the following definition.

Definition 15 (Equivalent Interpretations) *Given a state s, an isomorphic abstract state s', and a FO-formula ϕ, the finite interpretations $\sigma : fr(\phi) \mapsto D$ and $\sigma' : fr(\phi) \mapsto D^A$ are* equivalent for ϕ w.r.t. s and s' *iff for some bijection $\chi : s(V) \cup \sigma(fr(\phi)) \mapsto s'(V) \cup \sigma'(fr(\phi))$, we have*

(i) *the restriction $\chi|_{s(V)}$ is a witness for $s \simeq s'$;*
(ii) *$\sigma' = \chi \circ \sigma$;*
(iii) *for every $u, u' \in s(V) \cup \sigma(fr(\phi))$, $u \leq u'$ iff $\chi(u) \leq_{s'} \chi(u')$.*

Again, notice that for order $\leq_{s'}$, a witness χ preserves only the interpretation of individuals in $s'(V) \cup \sigma(fr(\phi))$, which are in finite number. As customary in first-order logic, we can prove that equivalent assignments on isomorphic states preserve FO-formulas. We report this result for our particular setting.

Lemma 1 *Given a state s, and an isomorphic abstract state s', if interpretations σ and σ' are equivalent for FO-formula ϕ w.r.t. s and s', then*

$$(s, \sigma) \models \phi \quad \textit{iff} \quad (s', \sigma') \models \phi$$

Proof. The proof is by induction on the structure of ϕ. If $\phi \equiv (z = z')$, then $(s, \sigma) \models \phi$ iff $(s, \sigma)(z) = (s, \sigma)(z')$. In particular, for some bijection $\chi : s(V) \cup \sigma(fr(\phi)) \mapsto s'(V) \cup \sigma'(fr(\phi))$, we have $s \overset{\chi}{\simeq} s'$ and $\sigma' = \chi \circ \sigma$. Hence, $(s', \sigma')(z) = (\chi \circ s, \chi \circ \sigma)(z) = \chi((s, \sigma)(z)) = \chi((s, \sigma)(z')) = (s', \sigma')(z')$, as required.

If $\phi \equiv (z \leq z')$, then $(s, \sigma) \models \phi$ iff $(s, \sigma)(z) \leq (s, \sigma)(z')$. Again, this is the case iff $(s', \sigma')(z) \leq_{s'} (s', \sigma')(z')$, as $s \overset{\chi}{\simeq} s'$ and for every $u, u' \in s(V) \cup \sigma(fr(\phi))$, $u \leq u'$ iff $\chi(u) \leq_{s'} \chi(u')$.

The cases for propositional connectives are immediate.

If $\phi \equiv \forall x \psi$, then $(s, \sigma) \models \phi$ iff for all $u \in s(V)$, $(s, \sigma^x_u) \models \psi$. Observe that interpretations σ^x_u and $\sigma'^x_{\chi(u)}$ are equivalent for ψ w.r.t. s and s'. Hence, by induction hypothesis, $(s', \sigma'^x_{\chi(u)}) \models \psi$. Since u is arbitrary and $\chi : s(V) \to s'(V)$ is a bijection, we have that $(s', \sigma') \models \phi$. □

Lemma 1 also applies to local states. Hence, if s, s' are isomorphic, and σ, σ' are equivalent for all guards g_γ of module m's commands, then 'isomorphic' actions are available to module m in s and s', that is, $(\gamma, \sigma) \in Pr_m(s_m)$ iff $(\gamma, \sigma') \in Pr'_m(s'_m)$, for the same command γ. This remark will be frequently used in the following, without explicitly mentioning it.

We now define the execution of an infinite-state reactive module system on the abstract domain D^A. Given an IRMS M defined on infinite domain D, the abstraction M^A of M is the same IRMS, executed on abstract domain D^A with a different semantics. Specifically, the abstract components Act^A_m, ACT^A, Pr^A_m, and S^A are defined as for M (in particular, $Act^A_m = Act_m$ and $ACT^A = ACT$), while components I^A and τ^A are given as follows:

- I^A is the set of abstract states s' such that for every $v \in V$, $s'(v) = \sigma(x)$ for some state s, interpretation σ, and initialisation command γ, such that $(s, \sigma) \models g_\gamma$. Moreover, $\leq_{s'}$ is any total linear extension on D^A of the partial order $\leq_s |_{s'(V)}$;
- $\tau^A : \mathcal{G}^A \times ACT^A \times \Sigma^A \to 2^{\mathcal{G}^A}$ is such that $s' \in \tau^A(s, \gamma, \sigma)$ iff *(i)* for all $m \in M$, $(\gamma_m, \sigma_m) \in Pr^A_m(l_m)$; and *(ii)* $s'(v_i) = \sigma(x_i)$. Moreover, $\leq_{s'}$ is any total linear extension of the partial order $\leq_s |_{s'(V)}$.

The main difference between the concrete and abstract execution of an IRMS is that in the latter the total orders are updated at each transition. Specifically, in a transition from a state s to s' we preserve the order $\leq_s$ on the elements in s that appear also in the active domain of s'. As to the remaining elements in $D^A \setminus (s(V) \cap s'(V))$, we extend the restriction $\leq_s |_{V(s')}$ to the whole D^A arbitrarily. Notice that this can be done in polynomial time. By doing so, the abstract transition τ^A is non-deterministic, differently from the concrete τ. In particular, all abstract states $s' \in \tau^A(s, \gamma, \sigma)$ are total completions of the partial order $\leq_s |_{s'(V)}$.

We now prove that Lemma 1 can be lifted to the full language FO-ATL. To do so, we need a few more definitions. First, an IRMS is *dense (with no end points)* iff for each type, the total linear order $\leq$ is. Also, for a type T, $par_T(\psi)$ is the set of all parameters (free and bound) appearing in ψ, while N_T denotes the set of parameters appearing in all guarded command, for all agents. For instance, in the IRMS for English auctions in Section 3, for the type of rational numbers $N = 2 + 4n$, where n is the number of bidders. Finally, for a coalition $C \subseteq Ag$, a C-action $\gamma_C \in Act_C$ is a tuple of commands for all agents in C. We say that a joint action $\gamma' \in ACT$ *extends* γ_C, or $\gamma' \sqsupseteq \gamma_C$, iff for all $i \in C$, $\gamma'_i = \gamma_i$.

We now prove the following auxiliary lemma, which states that, provided a sufficient number of abstract values, transitions in a dense IRMS M (with no endpoints) can be replicated in the abstraction M^A, and viceversa.

Lemma 2 *Consider a dense IRMS M with abstraction M^A, a state $s \in S$, an isomorphic abstract state $s' \in S'$, and an FO-ATL formula ψ. If for every type T, $|D^A_T| \geq |V_T| + |par_T(\psi)| + |N_T|$, then for every assignments σ and σ' equivalent for ψ w.r.t. s and s', we have*

1. *for every C-action $\gamma_C \in Act_C$ and finite interpretation ρ_C, if for every $m \in C$, $(\gamma_m, \rho_m) \in Pr_m(l_m)$, then there exists a finite interpretation ρ'_C such that for every $m \in C$, $(\gamma_m, \rho'_m) \in Pr^A_m(l'_m)$, and for every extension $\gamma \sqsupseteq \gamma_C$ and $\rho' \supseteq \rho'_C$, if $s' \xrightarrow{\gamma, \rho'} t'$, then for some extension $\rho \supseteq \rho_C$, $s \xrightarrow{\gamma, \rho} t$ and σ and σ' are equivalent for ψ w.r.t. t and t'.*
2. *for every C-action $\gamma'_C \in Act^A_C$ and finite interpretation ρ'_C, if for every $m \in C$, $(\gamma'_m, \rho'_m) \in Pr^A_m(l'_m)$, then there exists a finite interpretation ρ_C such that for every $m \in C$, $(\gamma'_m, \rho_m) \in Pr_m(l_m)$, and for every extension $\gamma' \sqsupseteq \gamma'_C$ and $\rho \supseteq \rho_C$, if $s \xrightarrow{\gamma', \rho} t$, then for some extension $\rho' \supseteq \rho'_C$, $s' \xrightarrow{\gamma', \rho'} t'$ and σ and σ' are equivalent for ψ w.r.t. t and t'.*

Proof. To prove (1), let $\chi : s(V) \cup \sigma(fr(\psi)) \mapsto s'(V) \cup \sigma'(fr(\psi))$ be a bijection witnessing that σ and σ' are equivalent for ψ w.r.t. s

and s', i.e., $\sigma' = \chi \circ \sigma$. Also, consider C-action γ_C and finite interpretation ρ_C such that for every $m \in C$, $(\gamma_m, \rho_m) \in Pr_m(l_m)$. Since for every type T, $|D^A_T| \geq |V_T| + |par_T(\varphi)| + |N_T|$, we can extend χ to an injective function $\overline{\chi} : s(V) \cup \sigma(fr(\psi)) \cup \rho_C(N) \to D^A$ satisfying the condition: $u \leq u'$ iff $\overline{\chi}(u) \leq_{s'} \overline{\chi}(u')$. Then define $\rho'_C = \overline{\chi} \circ \rho_C$. By construction and Lemma 1, for every $m \in C$ and action γ_m, $(l'_m, \rho'_m) \models g_m$ iff $(l_m, \rho_m) \models g_m$. Hence, action γ_C is enabled in state s' as well. Then, consider the execution of any joint action $\gamma \sqsupseteq \gamma_C$ with parameters $\rho' \supseteq \rho'_C$, thus giving $s' \xrightarrow{\gamma, \rho'} t'$ for some $t' \in S^A$. Since the IRMS M is infinite, dense, and with no end points, we can always find a further extension $\overline{\overline{\chi}} : s'(V) \cup \sigma'(fr(\psi)) \cup \rho'(N) \to D$, that agrees with the converse $\overline{\chi}^{-1}$ of $\overline{\chi}$ on $s'(V) \cup \sigma'(fr(\psi)) \cup \rho'_C(N)$ and also satisfies the condition: $u \leq_{s'} u'$ iff $\overline{\overline{\chi}}(u) \leq \overline{\overline{\chi}}(u')$. Then define $\rho = \overline{\overline{\chi}} \circ \rho'$. Again, by construction and Lemma 1, for every $m \in Ag$ and action γ_m, $(l_m, \rho_m) \models g_m$ iff $(l'_m, \rho'_m) \models g_m$. Hence, joint action γ is also enabled in state s, and the execution of γ in s with parameters ρ gives $s \xrightarrow{\gamma, \rho} t$, where in particular t is isomorphic to t' with witness $\overline{\overline{\chi}}$. Also, by the construction above, σ and σ' are equivalent for ψ w.r.t. t and t'.

The proof for (2) follows a similar line of reasoning: given a C-action $\gamma'_C \in Act^A_C$ and finite interpretation ρ'_C such that for every $m \in C$, $(\gamma'_m, \rho'_m) \in Pr^A_m(l'_m)$, by exploiting the density of M (and the lack of end points), we can construct a finite interpretation ρ_C such that for every $m \in C$, $(\gamma'_m, \rho_m) \in Pr_m(l_m)$. Moreover, for every joint action $\gamma' \sqsupseteq \gamma'_C$ and extension $\rho \supseteq \rho_C$, if $s \xrightarrow{\gamma', \rho} t$, then by using the constraint on the cardinality of D^A in M^A, we can construct an extension $\rho' \supseteq \rho'_C$ such that $s' \xrightarrow{\gamma', \rho'} t'$ and σ and σ' are equivalent for ψ w.r.t. t and t'. □

Lemma 2 states that, by the constraint on the cardinality of D^A, in abstraction M^A we have 'enough' elements to simulate the transitions in M. On the other hand, by using density (and the lack of endpoints), in M we can simulate the transitions in abstraction M^A. Actually, Lemma 2 provides a notion of alternating bisimulation for first-order ATL, which is an original contribution of the paper to our knowledge. Also, this result is applied in the proof of the following key lemma.

Lemma 3 *Consider a dense IRMS M with abstraction M^A, state $s \in S$ and isomorphic abstract state $s' \in S'$, and an FO-ATL formula ψ. If for every type T, $|D^A_T| \geq |V_T| + |par_T(\varphi)| + |N_T|$, then for every assignments σ and σ' equivalent for ψ w.r.t. s and s', we have*

1. *for every joint strategy F_C, there exists F'_C such that for every $\lambda' \in out(s', F'_C)$, there exists some $\lambda \in out(s, F_C)$ such that for all $i \geq 0$, σ and σ' are equivalent for ψ w.r.t. $\lambda(i)$ and $\lambda'(i)$.*
2. *for every joint strategy F'_C, there exists F_C such that for every $\lambda \in out(s, F_C)$, there exists some $\lambda' \in out(s', F'_C)$ such that for all $i \geq 0$, σ and σ' are equivalent for ψ w.r.t. $\lambda(i)$ and $\lambda'(i)$.*

Proof. We begin by proving (1). We build the strategy F'_C and prove the statement of the lemma by induction on length n of paths. For $n = 0$, we have that σ and σ' are equivalent for ψ w.r.t. $\lambda(0) = s$ and $\lambda'(0) = s'$. As to the inductive step, suppose that σ and σ' are equivalent for φ w.r.t. $\lambda(i)$ and $\lambda'(i)$, and consider C-action γ_C such that for every $m \in C$, $\gamma_m \in F_C(\lambda(i)_m)$ and finite interpretation ρ_C such that for every $m \in C$, $(\gamma_m, \rho_m) \in Pr_m(l_m)$. We set $F'_C(\lambda'(i)_m) = \gamma_m$ for every $m \in C$. In particular, by Lemma 2.1 there exists a finite interpretation ρ'_C such that for every $m \in C$, $(\gamma_m, \rho'_m) \in Pr^A_m(l'_m)$. Now consider the joint action $\gamma \sqsupseteq \gamma_C$ and extension $\rho' \supseteq \rho'_C$ such that $\lambda'(i) \xrightarrow{\gamma, \rho'} \lambda'(i+1)$. Again by Lemma 2.1, there exist some extension $\rho \supseteq \rho_C$ and $t \in S$ such that $\lambda(i) \xrightarrow{\gamma, \rho} t$. Then, set $\lambda(i+1) = t$. In particular, σ and σ' are equivalent for ψ w.r.t. $\lambda(i+1)$ and $\lambda'(i+1)$, and therefore the statement of the lemma is satisfied.

Item (2) is proved similarly, by using Lemma 2.2 instead. □

Intuitively, Lemma 3 states that joint strategies can be simulated between M and its abstraction M^A, provided that the relevant constraint are met. By this lemma we can prove the main result of this section.

Theorem 4 *Consider a dense IRMS M, its abstraction M^A, a state $s \in S$, an isomorphic abstract state $s' \in S'$, and an FO-ATL formula ψ. If for every type T, $|D^A_T| \geq |V_T| + |par_T(\varphi)| + |N_T|$, then for every assignments σ and σ' equivalent for ψ w.r.t. s and s', we have that*

$$(M, s, \sigma) \models \psi \quad \textit{iff} \quad (M^A, s', \sigma') \models \psi.$$

Proof. The proof is by induction on the structure of ψ. The base case for first-order formulas follows by Lemma 1; the inductive cases for propositional connectives are immediate. In particular, notice that, since for every type T, $|D^A_T| \geq |V_T| + |par_T(\neg\psi)| + |N_T| = |V_T| + |par_T(\psi)| + |N_T|$ and $|D^A_T| \geq |V_T| + |par_T(\psi_1 \wedge \psi_2)| + |N_T| \geq |V_T| + |par_T(\psi_i)| + |N_T|$, for $i = 1, 2$, the induction hypothesis holds.

For $\psi \equiv \forall x \phi$, $(M, s, \sigma) \models \psi$ iff for all $u \in s(V)$, $(M, s, \sigma^x_u) \models \phi$. If χ is a witness to the fact that σ and σ' equivalent for ψ w.r.t. s and s', then interpretations σ^x_u and $\sigma'^x_{\chi(u)}$ are equivalent for ϕ (also w.r.t. s and s'). Moreover, $|D^A_T| \geq |V_T| + |par_T(\psi)| + |N_T| \geq |V_T| + |par_T(\phi)| + |N_T|$. Hence, the induction hypothesis holds and it follows that $(M^A, s', \sigma'^x_{\chi(u)}) \models \phi$. Since χ is a bijection, we obtain that $(M^A, s', \sigma') \models \psi$.

Suppose that $\psi \equiv \langle\langle C \rangle\rangle X \phi$. As regards the $\Rightarrow$ direction, $(M, s, \sigma) \models \psi$ iff for some joint strategy F_C, for every $\lambda \in out(s, F_C)$, $(M, \lambda(1), \sigma) \models \phi$. By Lemma 3.1 there exists a strategy F'_C depending on F_C, such that for every $\lambda' \in out(s', F'_C)$, there exists some $\lambda \in out(s, F_C)$ such that σ and σ' are equivalent for φ w.r.t. $\lambda(1)$ and $\lambda'(1)$. Since $|D^A_T| \geq |V_T| + |par_T(\psi)| + |N_T| = |V_T| + |par_T(\phi)| + |N_T|$, by induction hypothesis $(M^A, \lambda'(1), \sigma') \models \psi$ for every $\lambda' \in out(s', F'_C)$, that is, $(M^A, s', \sigma') \models \psi$. The $\Leftarrow$ direction is proved similarly, by using Lemma 3.2. The proof for the other ATL operators follows an analogous line of reasoning. □

By Theorem 4 a dense IRMS M and its abstraction M^A satisfy the same formulas in FO-ATL, whenever the abstract domain D^A contains enough elements to replicate transitions in M. Most importantly, M^A can be assumed to be finite. As a consequence, we can verify an FO-ATL formula ψ on M by model checking M^A. We state this last result formally in the following corollary.

Corollary 5 *Consider a dense IRMS M, its abstraction M^A, and an FO-ATL formula φ. If for every type T, $|D^A_T| \geq |V_T| + |par_T(\varphi)| + |N_T|$, then*

$$M \models \varphi \quad \textit{iff} \quad M^A \models \varphi$$

To conclude, we have identified a significant class of infinite-state reactive module systems for which the model checking problem is decidable. Specifically, whenever, the orders on the domains of interpretations are assumed to be dense, with no end points, as it is the case for the rational number in the auction IRMS, by choosing a domain of abstract values of appropriate cardinality, we can construct a finite abstraction that preserves the interpretation of FO-ATL formulas.

5 Discussion: Complexity and Abstract Auctions

In Section 4 we provided an abstraction-based technique for the verification of IRMS. Here we elaborate more on the model checking procedure.

By previous contributions [18, 10] we know that the model checking problem for *finite* general data-aware systems, against a first-order extension of CTL, is EXPSPACE-complete in the combined size $|D| + ||\varphi||$ of data and the formula. Since ATL subsumes CTL, we immediately obtain that model checking finite reactive module systems against FO-ATL is EXPSPACE-hard. Moreover, by combining the procedures for model checking ATL under imperfect information and first-order logic, we obtain an algorithm in EXPSPACE for FO-ATL. Hence, we can state the complexity of model checking, for instance, the finite abstractions in Section 4.

Theorem 6 *The model checking problem for finite reactive module systems with respect to FO-ATL is EXPSPACE-complete in the combined size $|D| + ||\varphi||$ of data and the formula.*

It should be noted that the complexity is the same as that of model checking similar structures against first-order CTL [10]. Thus, the enhanced expressiveness of ATL comes at no extra computational cost. This is in contrast with the propositional case, where complexity jumps from PTIME to Δ_2^P under imperfect information [27]. Here the situation is different as the complexity of model checking data trumps that of the modal fragment. Moreover, we discussed above that reactive module can be thought of as compact representations of transition systems, where states and transitions are given implicitly in the form of agent programs. Hence, Def. 8 is really an instance of *implicit* model checking, whose complexity is typically higher than the explicit counterparts [12].

Additionally, we anticipate to be able to find cases of interest, whose complexity is amenable to practical model checking. For example, consider the IRMS for English auctions described in Section 3 for n bidders. The only infinite type are the rationals used to represent bids, true values, and base prices, for which we have $|V| = 1 + 2n$ variables, $|N| = 2 + 4n$ parameters, and $par(\phi) = 5n$, where ϕ is the conjunction of all specifications appearing in Section 3. Thus, to simulate rational values in the IRMS for English auctions, it is sufficient to consider a domain D of abstract elements, whose size is $|D| \geq (1 + 2n) + (2 + 4n) + 5n = 3 + 11n$, but finite and linear in the number n of bidders. As a result, to verify English auctions against the strategic behaviours formalised in Section 3, it is enough to consider the finite abstraction built on such finite domain D, and whose execution is described in Section 4. Moreover, to alleviate further the verification burden, we can exploit the symmetries of IRMS, both at the level of data and of the behaviours of agents (e.g., all bidders share the same actions and protocol). Thus, we envisage to deploy data-symmetry reductions [16] as well as further abstraction methodologies (e.g., multi-valued abstraction [8, 29]) on the finite, abstract reactive module system. We leave this for future work.

6 Conclusions

In this paper we have put forward a technique for the verification of infinite-state MAS against first-order modal specifications expressing strategic abilities of agents. The contributions of the paper are as follows. Firstly, we have introduced infinite-state reactive modules as an extension of simple reactive modules [33], which are suitable to model MAS with variables ranging over infinite domains. Secondly, we have defined a first-order ATL to specify strategic interactions of reactive modules in IRMS. However, the execution of IRMS generates infinite-state systems; these normally admit an undecidable verification problem. The third contribution consisted in observing that IRMS admit a decidable verification problem under specific conditions, namely a dense total order with no endpoints. Additional validation of the formalism here studied came from the modelling of auctions: IRMS were used to describe formally English and repeated sealed bid auctions, while FO-ATL was employed to capture specifications accounting for infinite domains. Purely in terms of modelling, we are not aware of other formalisms able to capture the strategic interaction of agents in a first-order setting. Moreover, to achieve the decidability results, specifically to show that a finite abstract model can be used to reason about an IRMS, we introduced a novel notion of alternating bisimulation at the first order. This notion is likely to be applicable in similar forms to show decidability for other first-order logics for strategic reasoning.

Several extensions of the proposed framework appear promising. Firstly, we envisage to introduce epistemic operators in FO-ATL to represent individual and group knowledge explicitly, and be able to express secrecy properties in auctioning scenarios. Secondly, IRMS can be extended with richer specification languages supporting arithmetic operations for instance [19].

Acknowledgements

This research was funded by the EPSRC under grant EP/I00520X.

REFERENCES

[1] S. Abiteboul, R. Hull, and V. Vianu, *Foundations of Databases*, Addison-Wesley, 1995.

[2] T. Ågotnes, V. Goranko, W. Jamroga, and M. Wooldridge, ‘Knowledge and ability’, in *Handbook of Logics for Knowledge and Belief*, College Publications, (2015).

[3] R. Alur, L. de Alfaro, R. Grosu, T. Henzinger, A. Thomas, M. Kang, C. Kirsch, R. Majumdar F. Mang, and B-Y. Wang, ‘jMocha: A model checking tool that exploits design structure’, in *Proceedings of the 23rd International Conference on Software Engineering (ICSE01)*, pp. 835–836. IEEE, (2001).

[4] R. Alur and T. Henzinger, ‘Reactive modules’, *Formal Methods in System Design*, **15**(1), 7–48, (1999).

[5] R. Alur, T. A. Henzinger, and O. Kupferman, ‘Alternating-time temporal logic’, *Journal of the ACM*, **49**(5), 672–713, (2002).

[6] A. Badica and C. Badica, ‘Specification and verification of an agent-based auction service’, in *Information Systems Development*, 239–248, Springer US, (2010).

[7] B. Bagheri, D. Calvanese, M. Montali, G. Giacomo, and A. Deutsch, ‘Verification of relational data-centric dynamic systems with external services’, in *Proceedings of the 32nd Symposium on Principles of Database Systems (PODS13)*, pp. 163–174. ACM, (2013).

[8] T. Ball and O. Kupferman, ‘An abstraction-refinement framework for multi-agent systems’, in *Proceedings of the 21st Annual IEEE Symposium on Logic in Computer Science (LICS06)*, pp. 379–388. IEEE, (2006).

[9] F. Belardinelli, ‘Model checking auctions as artifact systems: Decidability via finite abstraction’, in *Proceedings of the 21st European Conference on Artificial Intelligence (ECAI14)*, pp. 81–86, (2014).

[10] F. Belardinelli, A. Lomuscio, and F. Patrizi, ‘Verification of agent-based artifact systems’, *Journal of Artificial Intelligence Research*, **51**, 333–376, (2014).

[11] K. Bhattacharya, C. Gerede, R. Hull, R. Liu, and J. Su, ‘Towards formal analysis of artifact-centric business process models’, in *Business Process Management: Proceedings of the 5th International Conference (BPM07)*, pp. 288–304. Springer, (2007).

[12] N. Bulling, J. Dix, and W. Jamroga, 'Model checking logics of strategic ability: Complexity', in *Specification and Verification of Multi-agent Systems*, 125–159, Springer, (2010).

[13] N. Bulling and W. Jamroga, 'Comparing variants of strategic ability: how uncertainty and memory influence general properties of games', *Autonomous Agents and Multi-Agent Systems*, **28**(3), 474–518, (2014).

[14] D. Calvanese, G. Delzanno, and M. Montali, 'Verification of relational multiagent systems with data types', in *Proceedings of the 29th AAAI Conference on Artificial Intelligence (AAAI15)*, pp. 2031–2037. AAAI Press, (2015).

[15] K. Chatterjee, T. Henzinger, and N. Piterman, 'Strategy logic', in *Proceedings of the 18th International Conference on Concurrency Theory (CONCUR07)*, volume 4703, pp. 59–73, (2007).

[16] M. Cohen, M. Dam, A. Lomuscio, and H. Qu, 'A symmetry reduction technique for model checking temporal-epistemic logic', in *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI09)*, pp. 721–726, (2009).

[17] G. de Giacomo, Y. Lesperance, F. Patrizi, and S. Vassos, 'Progression and verification of situation calculus agents with bounded beliefs', in *Proceedings of the International conference on Autonomous Agents and Multi-Agent Systems (AAMAS14)*, pp. 141–148. IFAAMAS, (2014).

[18] A. Deutsch, R. Hull, F. Patrizi, and V. Vianu, 'Automatic verification of data-centric business processes', in *Proceedings of the 12th International Conference on Database Theory (ICDT09)*, pp. 252–267. ACM, (2009).

[19] A. Deutsch, Y. Li, and V. Vianu, 'Verification of hierarchical artifact systems'.

[20] D. Easley and J. Kleinberg, *Networks, Crowds, and Markets: Reasoning About a Highly Connected World*, Cambridge University Press, New York, NY, USA, 2010.

[21] R. Fagin, J. Y. Halpern, Y. Moses, and M. Y. Vardi, *Reasoning about Knowledge*, MIT Press, Cambridge, 1995.

[22] G. De Giacomo, Y. Lesperance, and F. Patrizi, 'Bounded epistemic situation calculus theories', in *Proceedings of the 23rd International Joint Conference on Artificial Intelligence (IJCAI13)*, (2013).

[23] P. Gonzalez, A. Griesmayer, and A. Lomuscio, 'Verification of GSM-based artifact-centric systems by predicate abstraction', in *Proceedings of the 13th International Conference on Service Oriented Computing (ICSOC15)*, volume 9435 of *Lecture Notes in Computer Science*, pp. 253–268. Springer, (2015).

[24] E. M. Tadjouddine F. Guerin and W. Vasconcelos, 'Abstractions for model-checking game-theoretic properties of auctions', in *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems (AAMAS08)*, pp. 1613–1616. IFAAMAS, (2008).

[25] A. G. Hamilton, *Logic for Mathematicians*, Cambridge University Press, 1978.

[26] W. Jamroga and T. Ågotnes, 'Modular interpreted systems', in *Proceedings of the 6th International Conference on Autonomous Agents and Multi-Agent systems (AAMAS07)*, pp. 131–138. IFAAMAS, (2007).

[27] W. Jamroga and J. Dix, 'Model checking abilities under incomplete information is indeed δ_p^2-complete', in *Proceedings of the 4th European Workshop on Multi-Agent Systems (EUMAS'06)*, pp. 14–15, (2006).

[28] W. Jamroga and W. van der Hoek, 'Agents that know how to play', *Fundamenta Informaticae*, **62**, 1–35, (2004).

[29] A. Lomuscio and J. Michaliszyn, 'Verification of multi-agent systems via predicate abstraction against ATLK specifications', in *Proceedings of the 15th International Conference on Autonomous Agents and Multiagent Systems (AAMAS16)*, (2016).

[30] A. Lomuscio, H. Qu, and F. Raimondi, 'MCMAS: A model checker for the verification of multi-agent systems', *Software Tools for Technology Transfer*, (2015). http://dx.doi.org/10.1007/s10009-015-0378-x.

[31] F. Mogavero, A. Murano, G. Perelli, and M. Y. Vardi, 'Reasoning about strategies: On the model-checking problem', *ACM Transactions in Computational Logic*, **15**(4), 34:1–34:47, (2014).

[32] N. Nisan, J. Bayer, D. Chandra, Tal Franji, R. Gardner, Y. Matias, N. Rhodes, M. Seltzer, D. Tom, Hal Varian, and D. Zigmond, 'Google's auction for TV ads', in *Automata, Languages and Programming, 36th Internatilonal Colloquium (ICALP09), Proceedings*, pp. 309–327. Springer, (2009).

[33] W. van der Hoek, A. Lomuscio, and M. Wooldridge, 'On the complexity of practical ATL model checking knowledge, strategies, and games in multi-agent systems', in *Proceedings of the 5th international joint conference on Autonomous agents and multiagent systems (AAMAS06)*, pp. 201–208. ACM Press, (2006).

[34] H. Xu, C. K. Bates, and S. M. Shatz, 'Real-time model checking for shill detection in live online auctions', in *Software Engineering Research and Practice*, pp. 134–140, (2009).

[35] H. Xu and Y. Cheng, 'Model checking bidding behaviors in internet concurrent auctions.', *Computer System Science & Engineering*, **22**(4), (2007).