Proceedings, Fifteenth International Conference on
Principles of Knowledge Representation and Reasoning (KR 2016)

Imperfect Information in Reactive Modules Games

Julian Gutierrez, Giuseppe Perelli, Michael Wooldridge
Department of Computer Science
University of Oxford

Abstract

Reactive Modules is a high-level modelling language for concurrent, distributed, and multi-agent systems, which is used in a number of practical model checking tools. Reactive Modules Games are a game-theoretic extension of Reactive Modules, in which agents in a system are assumed to act strategically in an attempt to satisfy a temporal logic formula representing their individual goal. Reactive Modules Games with perfect information have been closely studied, and the complexity of game theoretic decision problems relating to such games have been comprehensively classified. However, to date, no work has considered the imperfect information case. In this paper we address this gap, investigating Reactive Modules Games in which agents have only partial visibility of their environment.

Introduction

A common technique to design or verify computer systems is to represent their behaviour using games in which two players—sometimes called "System" and "Environment" or "Player" and "Opponent"—interact with each other, possibly, for infinitely many rounds. In these games, it is assumed that the system has a goal given in a logical form, *e.g.*, expressed as a temporal logic formula φ, which the system wishes to satisfy. Such a goal can represent either the behaviour of the computer system one wants to synthesize (an automated design problem (Pnueli and Rosner 1989)) or a particular system property which one wants to check (an automated verification problem (Clarke, Grumberg, and Peled 2000)). In this framework, it is assumed that the system plays against an adversarial environment, that is, that the goal of the environment is to prevent the system from achieving its goal. In game-theoretic terms, this means that the problem is modelled as a zero-sum game, and hence that its solution is given by the computation of a winning strategy for either the system or the environment. From a logical viewpoint, this assumption amounts to letting the goal of the environment be $\neg\varphi$, whenever the goal of the system is given by the temporal logic formula φ. A great deal of work has been based on this idea—see, *e.g.*, (Ghica 2009; Walukiewicz 2004) and the references therein for surveys.

Although this paradigm has been found to be useful in a range of settings, the zero-sum assumption is often too restrictive or inappropriate. For instance, when dealing with concurrent systems one may have to account for several computer components—each with its own temporal goal—that are not necessarily in conflict. This situation leads to the definition of a non-zero-sum n-player game (naturally modelling a multi-agent system) rather than a two-player zero-sum game.

In the non-zero-sum n-player setting it is no longer the computation of a winning strategy what provides a solution to the problem under consideration, but rather, the computation of a strategy profile (a set of strategies, one for each player in the game) which can be regarded as in equilibrium in the game-theoretic sense (Osborne and Rubinstein 1994): a situation where no player wishes to deviate from the strategy it is currently using. This problem of modelling computer systems as non-zero-sum games instead of zero-sum ones has already been identified, and some work has been done; see, for instance (Bouyer et al. 2011; 2012; Chatterjee and Henzinger 2012) for some references.

Here, we study non-zero-sum n-player games in which the choices available to players are defined using the Simple Reactive Modules Language (SRML), a subset of Reactive Modules (Alur and Henzinger 1999), a popular and expressive system modelling language that is used in several practical model checking systems (e.g., MOCHA (Alur et al. 1998) and Prism (Kwiatkowska, Norman, and Parker 2011)). Reactive Modules supports *succinct* and *high-level* modelling of concurrent and multi-agent systems. In the games we study, the preferences of system components are specified by associating with each player in the game a temporal logic (LTL) formula that the player desires to be satisfied. Reactive Modules Games with perfect information (where each player can see the entire system state) have been extensively studied (Gutierrez, Harrenstein, and Wooldridge 2015a), but in this paper we focus on *imperfect information* cases. We study the decidability and complexity of checking the existence of Nash equilibria in Reactive Modules games with imperfect information. In our framework, one can analyse the behaviour of open systems modelled as multi-player games using a specification language that is close to real-world programming and system modelling languages, and which already has a number of tool implementations. However, our results go beyond the interest in SRML itself as, more generally, we provide complexity results that apply to a wide range of imperfect information games with succinct representations.

Copyright © 2016, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

Our key results are as follows. We show that Reactive Modules Games with imperfect information are undecidable if three or more players are allowed. In contrast, if restricted to two players, the games are decidable and their solution (computing a Nash equilibrium if one exists) can be obtained in 2EXPTIME. For the latter decidability result, we provide a conceptually simple decision procedure based on synthesis techniques for CTL* under imperfect information. We also explore a number of variants of the general imperfect-information framework. For instance, we study variants of these games with respect to the class of strategies under consideration, *e.g.*, memoryless, myopic, polynomially bounded, and show that such games can be solved, respectively, in NEXPTIME, EXPSPACE, and PSPACE; we also explore the use of a solution concept where coordinated behaviour is allowed—in whose case strong Nash equilibrium is considered instead—and show that going from Nash to strong Nash equilibria can be done without paying a (worst-case) complexity cost. We then study in more detail the connection between imperfect information and the existence of Nash equilibria. Specifically, we provide conditions under which the set of Nash equilibria of an imperfect-information game can be preserved (or refined) with respect to the amount of information that players in such a game have. Note that due to lack of space most proofs are either omitted or sketched.

Preliminaries

Logic. We work with logics that extend propositional logic. These logics are based on a finite set Φ of Boolean variables. A *valuation* for propositional logic is a set $v \subseteq \Phi$, with the intended interpretation that $p \in v$ means that p is true under valuation v, while $p \notin v$ means that p is false under v. Let $V(\Phi) = 2^\Phi$ be the set of all valuations for variables Φ; where Φ is clear, we omit reference to it and write V.

Kripke Structures. We use *Kripke structures* to model the dynamics of our systems. A Kripke structure K over Φ is given by $K = \langle S, S_0, R, \pi\rangle$, where $S = \{s_0, \ldots\}$ is a finite non-empty set of *states*, $S_0 \subseteq S$ is the set of *initial states*, $R \subseteq S \times S$ is a total *transition relation* on S, and $\pi : S \to V$ is a valuation function, assigning a valuation $\pi(s)$ to every $s \in S$. Where $K = \langle S, S_0, R, \pi\rangle$ is a Kripke structure over Φ, and $\Psi \subseteq \Phi$, we denote the *restriction of K to* Ψ by $K|_\Psi$, where $K|_\Psi = \langle S, S_0, R, \pi|_\Psi\rangle$ is the same as K except that $\pi|_\Psi$ is the valuation function defined as follows: $\pi|_\Psi(s) = \pi(s) \cap \Psi$.

Runs. A *run of K* is a sequence $\rho = s_0, s_1, s_2, \ldots$ where for all $t \in \mathbb{N}$ we have $(s_t, s_{t+1}) \in R$. Using square brackets around parameters referring to time points, we let $\rho[t]$ denote the state assigned to time point t by run ρ. We say ρ is an s-run if $\rho[0] = s$. A run ρ of K where $\rho[0] \in S_0$ is referred to as an *initial* run. Let $runs(K, s)$ be the set of s-runs of K, and let $runs(K)$ be the set of initial runs of K. Notice that a run $\rho \in runs(K)$ induces an infinite sequence $\boldsymbol{\rho} \in V^\omega$ of propositional valuations, viz., $\boldsymbol{\rho} = \pi(\rho[0]), \pi(\rho[1]), \pi(\rho[2]), \ldots$. The set of these sequences, we denote by $\mathbf{runs}(K)$. Given $\Psi \subseteq \Phi$ and a run $\boldsymbol{\rho}\colon \mathbb{N} \to V(\Phi)$, we denote the restriction of $\boldsymbol{\rho}$ to Ψ by $\boldsymbol{\rho}|_\Psi$, *i.e.*, $\boldsymbol{\rho}|_\Psi[t] = \boldsymbol{\rho}[t] \cap \Psi$ for each $t \in \mathbb{N}$.

Linear Temporal Logic. In this paper, we mostly use Linear Temporal Logic (LTL), an extension of propositional logic with two modal tense operators, $\mathbf{X}$ ("next") and $\mathbf{U}$ ("until"), that can be used to express properties of runs, for instance, the runs of Kripke structures. The syntax of LTL is defined with respect to a set Φ of Boolean variables as follows:

$$\varphi ::= \top \mid p \mid \neg\varphi \mid \varphi \vee \varphi \mid \mathbf{X}\varphi \mid \varphi\,\mathbf{U}\,\varphi$$

where $p \in \Phi$. The remaining classical logic operators are defined in the standard way; we also use the following abbreviations: $\mathbf{F}\varphi = \top\ \mathbf{U}\,\varphi$ and $\mathbf{G}\varphi = \neg\mathbf{F}\neg\varphi$, for "eventually" and "always" respectively. We interpret formulae of LTL with respect to pairs (ρ, t), where ρ is a run of a Kripke structure $K = \langle S, S_0, R, \pi\rangle$ and $t \in \mathbb{N}$ is a temporal index into ρ:

$$\begin{array}{lll}
(\rho,t) \models \top & & \\
(\rho,t) \models p & \text{iff} & p \in \pi(\rho[t]) \\
(\rho,t) \models \neg\varphi & \text{iff} & \text{it is not the case that } (\rho,t) \models \varphi \\
(\rho,t) \models \varphi \vee \psi & \text{iff} & (\rho,t) \models \varphi \text{ or } (\rho,t) \models \psi \\
(\rho,t) \models \mathbf{X}\varphi & \text{iff} & (\rho,t+1) \models \varphi \\
(\rho,t) \models \varphi\,\mathbf{U}\,\psi & \text{iff} & \text{for some } t' \geq t\colon\ ((\rho,t') \models \psi \text{ and} \\
 & & \qquad \text{for all } t \leq t'' < t'\colon (\rho,t'') \models \varphi).
\end{array}$$

If $(\rho, 0) \models \varphi$, we also write $\rho \models \varphi$ and say that ρ *satisfies* φ. An LTL formula φ is *satisfiable* if there is some run satisfying φ. Moreover, a Kripke structure K satisfies φ if $\rho \models \varphi$ for all initial runs ρ of K. Finally, with $|\varphi|$ we denote the size of the LTL formula φ, given by its number of subformulae.

Reactive Modules Games

Reactive Modules. We focus on Simple Reactive Modules, the subset of the language introduced by (van der Hoek, Lomuscio, and Wooldridge 2006) to study the complexity of practical ATL model checking. Agents in Reactive Modules are known as *modules*. An SRML module with imperfect information (SMRLI) consists of:

(i) an *interface*, which defines the module's name, the set of Boolean variables under the *control* of the module, and the set of variables that are *visible* to the module; and

(ii) a number of *guarded commands*, which define the choices available to the module at every state.

Guarded commands are of two kinds: those used for *initialising* the variables under the module's control (**init** guarded commands), and those for *updating* these variables subsequently (**update** guarded commands). A guarded command has two parts: a condition part (the "guard") and an action part, which defines how to update the value of (some of) the variables under the control of a module. The intuitive reading of a guarded command $\varphi \rightsquigarrow \alpha$ is "if the condition φ is satisfied, then *one of the choices available to the module is to execute the action* α". We note that the truth of the guard φ does not mean that α *will* be executed: only that such a command is *enabled* for execution—it *may be chosen*.

Formally, a guarded command g over some set of Boolean (visible) variables Vis is an expression

$$\varphi \rightsquigarrow x'_1 := \psi_1; \cdots; x'_k := \psi_k$$

where φ (the guard) is a propositional formula over Vis, each x_i is a controlled variable, and each ψ_i is a propositional logic

formula over Vis. Let $guard(g)$ denote the guard of g. Thus, in the above rule, $guard(g) = \varphi$. We require that no variable appears on the left hand side of two assignment statements in the same guarded command. We say that $x_1, \ldots, x_k$ are the *controlled variables* of g, and denote this set by $ctr(g)$. If no guarded command of a module is enabled, the values of all variables in $ctr(g)$ are left unchanged; in SRML notation, if needed, **skip** will refer to this particular case.

Formally, an SRMLI module, m_i, is defined as a quadruple $m_i = \langle \Phi_i, \mathrm{Vis}_i, I_i, U_i \rangle$, where: $\Phi_i \subseteq \Phi$ is the (finite) set of variables controlled by m_i; Vis_i is the (finite) set of variables that are visible to m_i, with $\Phi_i \subseteq \mathrm{Vis}_i$; I_i is a (finite) set of *initialisation* guarded commands, such that for all $g \in I_i$, we have $ctr(g) \subseteq \Phi_i$; and U_i is a (finite) set of *update* guarded commands, such that for all $g \in U_i$, we have $ctr(g) \subseteq \Phi_i$. To simplify notation, since by definition $\Phi_i \subseteq \mathrm{Vis}_i$, hereafter by $\mathrm{Vis}_i = \Psi$, where $\Psi \subseteq \Phi$, we mean $\mathrm{Vis}_i = \Phi_i \cup \Psi$.

An SRMLI *arena* is defined to be an $(n+2)$-tuple

$$A = \langle N, \Phi, m_1, \ldots, m_n \rangle$$

where $N = \{1, \ldots, n\}$ is a set of agents, Φ is a set of Boolean variables, and for each $i \in N$, $m_i = \langle \Phi_i, \mathrm{Vis}_i, I_i, U_i \rangle$ is an SRMLI module over Φ that defines the choices available to agent i. We require that $\{\Phi_1, \ldots, \Phi_n\}$ forms a partition of Φ (so every variable in Φ is controlled by some module, and no variable is controlled by more than one module).

The behaviour of an SRMLI arena is obtained by executing guarded commands, one for each module, in a synchronous and concurrent way. The execution of an SMRLI arena proceeds in rounds, where in each round every module $m_i = \langle \Phi_i, \mathrm{Vis}_i, I_i, U_i \rangle$ produces a valuation v_i for the variables in Φ_i on the basis of a current valuation v. For each SRMLI arena A, the execution of guarded commands induces a unique Kripke structure, denoted by K_A, which formally defines the semantics of A. Based on K_A, one can define the sets of runs allowed in A, namely, those associated with the Kripke structure K. Finally, we sometimes will be concerned with the size of an arena. We say that the size of an arena $A = \langle N, \Phi, m_1, \ldots, m_n \rangle$, denoted by $|A|$ is $|m_1| + \ldots + |m_n|$, where the size of a module $m_i = \langle \Phi_i, \mathrm{Vis}_i, I_i, U_i \rangle$, denoted by $|m_i|$, is $|\Phi_i| + |\mathrm{Vis}_i| + |I_i| + |U_i|$. In particular, we will use LTL characterisations of the runs of arenas A and modules m. Such LTL formulae, denoted by $TH(A)$ and $TH(m)$, respectively, are polynomial in the sizes of A and m.

Games. The model of games we consider has two components. The first component is an *arena*: this defines the players, the variables they control, and the choices available to them in every game state. The arena plays a role analogous to that of a *game form* in conventional game theory (Osborne and Rubinstein 1994, p. 201): while it defines players and their choices, it does not specify the preferences of players. Preferences are specified by the second component of the game: every player i is associated with a *goal* γ_i, which will be a logical formula. The idea is that players desire to see their goal satisfied by the outcome of the game. Formally, a game is given by a structure $G = \langle A, \gamma_1, \ldots, \gamma_n \rangle$ where $A = \langle N, \Phi, m_1, \ldots, m_n \rangle$ is an arena with player set N, Boolean variable set Φ, and m_i an SRMLI module defining the choices available to each player i; moreover, for each $i \in N$, the logical formula γ_i represents the *goal* that i aims to satisfy. On this basis, the size of a game, $|G|$, is given by $|A| + |\gamma_1| + \ldots + |\gamma_n|$, where $|\gamma_i|$ is the size of γ_i.

Games are played by each player i selecting a *strategy* σ that will define how to make choices over time. Given an SRMLI arena $A = \langle N, \Phi, m_1, \ldots, m_n \rangle$, a *strategy* for module $m_i = \langle \Phi_i, \mathrm{Vis}_i, I_i, U_i \rangle$ is a structure $\sigma = (Q_i, q_i^0, \delta_i, \tau_i)$, where Q_i is a finite and non-empty set of *states*, $q_i^0 \in Q_i$ is the *initial* state, $\delta_i : Q_i \times V(\mathrm{Vis}_i) \to 2^{Q_i} \setminus \{\emptyset\}$ is a *transition function*, and $\tau_i : Q_i \to V_i$ is an *output function*. Note that not all strategies for a module may comply with that module's specification, not even in case of perfect information. For instance, if the only guarded update command of a module m_i has the form $\top \rightsquigarrow x' := \bot$, then a strategy for m_i should not prescribe m_i to set x to true under any contingency. Moreover, if a module's visibility set does not contain some variable p, then no strategy for such a module can be defined depending on the value of p. Strategies that comply with the module's specification (*i.e.*, strategies in the Kripke structure induced by the module) are called consistent. Let Σ_i be the set of consistent strategies for m_i. A strategy σ can be represented by an SRML module (of polynomial size in $|\sigma|$) with variable set $\Phi_i \cup Q_i$. We write m_σ for such a module specification.

Games are played by each player i by selecting a *strategy* σ that will define how to make choices over time. Once every player i has selected a strategy σ, a *strategy profile* $\vec{\sigma} = (\sigma_1, \ldots, \sigma_n)$ results and the game has an *outcome*, which we will denote by $[\![\vec{\sigma}]\!]$. The outcome $[\![\vec{\sigma}]\!]$ of a game with SRML arena $A = \langle N, \Phi, m_1, \ldots, m_n \rangle$ is defined to be the Kripke structure associated with the SRML arena $A_{\vec{\sigma}} = \langle N, \Phi \cup \bigcup_{i \in N} Q_i, m_{\sigma_1}, \ldots, m_{\sigma_n} \rangle$ restricted to valuations with respect to Φ, that is, the Kripke structure $K_{A_{\vec{\sigma}}}|_\Phi$. The outcome of a game will determine whether or not each player's goal is or is not satisfied. Because outcomes are Kripke structures, in general, goals can be given by any logic with a well defined Kripke structure semantics. Assuming the existence of such a satisfaction relation, which we denote by "$\models$", we can say that a goal γ_i is satisfied by an outcome $[\![\vec{\sigma}]\!]$ if and only if $[\![\vec{\sigma}]\!] \models \gamma_i$; to simplify notations, we may simply write $\vec{\sigma} \models \gamma_i$. Moreover, if we only consider deterministic strategies, that is, those where $\delta_i : Q_i \times V(\mathrm{Vis}_i) \to Q_i$, then all outcomes are single runs and we can even write $\rho(\vec{\sigma})$ for the unique run induced by $\vec{\sigma}$ in such a case. Hereafter, we will assume that goals are LTL formulae and that strategies are deterministic.

We are now in a position to define a preference relation $\succsim_i$ over outcomes for each player i with goal γ_i. For strategy profiles $\vec{\sigma}$ and $\vec{\sigma}'$, we say that

$$\vec{\sigma} \succsim_i \vec{\sigma}' \quad \text{if and only if} \quad \vec{\sigma}' \models \gamma_i \text{ implies } \vec{\sigma} \models \gamma_i.$$

On this basis, we also define the concept of Nash equilibrium (Osborne and Rubinstein 1994): given a game $G = (A, \gamma_1, \ldots, \gamma_n)$, a strategy profile $\vec{\sigma}$ is said to be a *Nash equilibrium* of G if for all players i and all strategies σ', we have

$$\vec{\sigma} \succsim_i (\vec{\sigma}_{-i}, \sigma_i'),$$

where $(\vec{\sigma}_{-i}, \sigma_i')$ denotes $(\sigma_1, \ldots, \sigma_{i-1}, \sigma_i', \sigma_{i+1}, \ldots, \sigma_n)$, the strategy profile where the strategy of player i in $\vec{\sigma}$ is replaced by σ_i'. Hereafter, let $NE(G)$ be the set of Nash equilibria of G.

Perfect vs. Imperfect Information

In this section, we describe a system which demonstrates two important facts about game-like system specifications: that imperfect information provides a more realistic framework (when compared with perfect information games); and that imperfect information can be used as a tool to eliminate undesirable rational behaviours (given by Nash equilibria).

Note that the following example is intended to illustrate the concepts introduced so far rather than to constitute a real-life specification. Then, consider a system with two agents, *Casino* and *Player*, who interact with each other at a casino in Las Vegas. The two agents are playing the following game. The game is played in two rounds, where in the first round *Casino* chooses one side of a 1 dollar coin (and keeps its choice hidden from *Player*) and in the second round *Player* tries to guess what side of the coin was chosen by *Casino*. If *Player* guesses correctly, *Player* wins; otherwise, *Casino* wins. In principle, the two agents can interact for as long as they want since there is no a priori bound on the amount of money or time they have to play the game. Moreover, the goals of the agents are to win the game infinitely often. Note that under normal circumstances, neither *Casino* nor *Player* should always win, as that outcome would be both unnatural and rather suspicious. Of course, they do not want to always lose the game either! We model this game, using the specific notation of SRMLI, with the following modules—cf. general definition given by a tuple $m_i = \langle \Phi_i, \mathrm{Vis}_i, I_i, U_i \rangle$—

module *Casino* **controls** $\{\mathsf{turn}, \mathsf{coin_c}\}$ **under** Vis_{Casino}
init
$:: \top \rightsquigarrow \mathsf{turn}' := \top;\ \mathsf{coin_c}' := \bot$
$:: \top \rightsquigarrow \mathsf{turn}' := \top;\ \mathsf{coin_c}' := \top$
update
$:: \mathsf{turn} \rightsquigarrow \mathsf{coin_c}' := \top;\ \mathsf{turn}' := \bot$
$:: \mathsf{turn} \rightsquigarrow \mathsf{coin_c}' := \bot;\ \mathsf{turn}' := \bot$
$:: \neg\mathsf{turn} \rightsquigarrow \mathsf{turn}' := \top$
module *Player* **controls** $\{\mathsf{coin_p}\}$ **under** Vis_{Player}
init
$:: \top \rightsquigarrow \mathsf{coin_p}' := \top$
$:: \top \rightsquigarrow \mathsf{coin_p}' := \bot$
update
$:: \neg\mathsf{turn} \rightsquigarrow \mathsf{coin_p}' := \top$
$:: \neg\mathsf{turn} \rightsquigarrow \mathsf{coin_p}' := \bot$

and goals: $\gamma_{Casino} = \mathbf{GF}(\neg\mathsf{turn} \rightarrow \neg(\mathsf{coin_c} \leftrightarrow \mathbf{X}\mathsf{coin_p}))$ and $\gamma_{Player} = \mathbf{GF}(\neg\mathsf{turn} \rightarrow (\mathsf{coin_c} \leftrightarrow \mathbf{X}\mathsf{coin_p}))$. If the game is with perfect information then $\mathrm{Vis}_{Casino} = \Phi = \mathrm{Vis}_{Player}$. Such a model has two kinds of Nash equilibria: one where *Player* always wins (using the strategy below), and another one where both agents satisfy their goals. Clearly, the former is an undesirable modelling scenario. But, if the game has imperfect information, *e.g.*, with $\mathrm{Vis}_{Player} = \{\mathsf{turn}\}$, then such "bad" equilibria disappear and only scenarios where both agents satisfy their goals remain as rational outcomes.

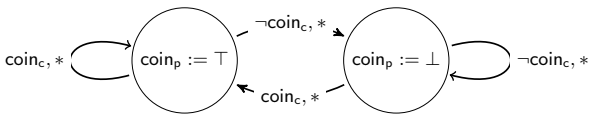

Figure 1: A winning strategy for *Player* if $\mathrm{Vis}_{Player} = \Phi$. Symbol $*$ is $\neg\mathsf{turn}$. Edges if $\mathsf{turn} = \top$ are loops (for **skip**).

Undecidability of SRMLI Games

As in many game-theoretic scenarios, the main problem related to the solution of a game is the existence of Nash equilibria. In our framework, such a problem is stated as follows:

Given: SRMLI G.
NONEMPTINESS: Is it the case that $NE(G) \neq \emptyset$?

We say that SRMLI games are undecidable if their non-emptiness problem is undecidable. In this section we will show that SRMLI games are undecidable when considering goals given by LTL formulae. In order to do so let us first provide some preliminary technical results.

We will reduce the uniform distributed synthesis problem (Finkbeiner and Schewe 2005) for LTL formulae, which is known to be undecidable, to NONEMPTINESS with three modules and goals given by LTL formulae. In order to define such a reduction we need to define some behaviour preserving transformations, in particular, one that deals with the preservation of LTL properties, which is presented next. This transformation is needed since SRML games are concurrent and the game for distributed synthesis is sequential instead.

LTL formula transformation

Let us start this subsection by giving some useful definitions and notations. For $\rho : \mathbb{N}_{\mathrm{o}} \rightarrow 2^{\Phi}$ a run and $d \geq 1$ an integer, we say that $\rho' : \mathbb{N}_{\mathrm{o}} \rightarrow 2^{\Phi}$ is a *d-fold inflation* of ρ if $\rho'[d \times t] = \rho[t]$ for every $t \geq 0$. For a set Ψ of propositional variables with $\Phi \subseteq \Psi$, also say that a run $\rho' : \mathbb{N}_{\mathrm{o}} \rightarrow 2^{\Psi}$ a *d-fold inflation* of ρ if $\rho'[d \times t] \cap \Phi = \rho[t]$ for every $t \geq 0$. Moreover, for $q \in \Psi \setminus \Phi$, we say that a d-fold inflation ρ' of ρ is *q-labelled* if for all $t \geq 0$, $q \in \rho'[t]$ if and only if t is a multiple of d, *i.e.* there is some $t' \in \mathbb{N}$ with $t = d \times t'$. Thus, in a q-labelled, d-fold inflation ρ' of ρ we have that $\rho'[t] \models q$ if and only if t is a multiple of d.

Clearly, from a run $\rho' : \mathbb{N}_{\mathrm{o}} \rightarrow 2^{\Psi}$, we can define the d-fold deflation ρ over Φ to be the run $\rho : \mathbb{N}_{\mathrm{o}} \rightarrow 2^{\Phi}$ which satisfies that $\rho[t] = \rho'[d \times t] \cap \Phi$ for every $t \geq 0$. Note that, for a given run ρ', there is a unique d-fold deflation ρ over Φ. Clearly, the d-fold inflation and deflation can be extended to partial runs. Moreover, for purposes that will be clear later in the paper, for a given partial run $h : \{0, \ldots, n\} \rightarrow \Psi$, by h^d we denote the partial run such that $|h^d| = k = \mathtt{quot}(|h|, d)$—where $\mathtt{quot}(x, y)$ denotes the quotient of the Euclidean division of x by y—and defined as $h^d[j] = h[j \times d] \cap \Psi$, for each $j < |h^d|$, and $h^d[k] = \mathsf{lst}(h) \cap \Psi$.

We now define, for each $d \geq 1$, a translation function τ_d which maps LTL formulae φ over Φ to LTL formulae $\tau_d(\varphi)$ over $\Phi \cup \{q\}$, where $q \notin \Phi$. Moreover, we omit the argument d when it is clear from the context.

- $\tau_d(p) = p$;
- $\tau_d(\neg\varphi) = \neg\tau_d(\varphi)$;
- $\tau_d(\varphi \vee \psi) = \tau_d(\varphi) \vee \tau_d(\psi)$;
- $\tau_d(\mathbf{X}\varphi) = \mathbf{X}^d\tau_d(\varphi)$;
- $\tau_d(\varphi\, \mathbf{U}\, \psi) = (q \rightarrow \tau_d(\varphi))\, \mathbf{U}\, (q \wedge \tau_d(\psi))$.

Finally, to prove the Lemma 1, we use the standard semantics of LTL formulae on infinite runs (Emerson 1990), which can be extended to Kripke structures just as defined before.

Lemma 1 (Inflation)**.** *Let Φ and Φ' be two disjoint sets of propositional variables with $q \in \Phi'$, $\rho : \mathbb{N}_0 \to 2^{\Phi}$ a run, $d \geq 1$, and $\rho' : \mathbb{N}_0 \to 2^{\Phi \cup \Phi'}$ a q-labelled, d-fold inflation of ρ. Then, for all* LTL *formulae φ over Φ, it holds that $\rho \models \varphi$ if and only if $\rho' \models \tau_d(\varphi)$.*

Architectures and synthesis

An *architecture* is a tuple $\mathcal{A} = \langle \mathrm{P}, p_0, p_{\mathtt{idle}}, E, \mathrm{O} \rangle$ where:

- P is a set of *processes*, with p_0 and $p_{\mathtt{idle}}$ being the *environment* and *idle* processes, respectively;
- (P, E) is a directed acyclic graph with p_0 having no incoming edges and $p_{\mathtt{idle}}$ being the unique vertex having no outcoming edges;
- $\mathrm{O} = \{\mathrm{O}_e : e \in E\}$ is a set of nonempty sets of output variables where $\mathrm{O}_{(p_1,p_2)} \cap \mathrm{O}_{(p_1',p_2')} \neq \emptyset$ implies $p_1 = p_1'$.

By $\mathrm{Vr} = \bigcup_{e \in \mathrm{O}} \mathrm{O}_e$ we denote the set of all variables. Moreover, by $\mathrm{I}_p = \bigcup_{p' \in \mathrm{P}} \mathrm{O}_{(p',p)}$ we denote the set of input variables for process p. Finally, $\mathrm{O}_p = \bigcup_{p' \in \mathrm{P}} \mathrm{O}_{(p,p')}$ denotes the set of output variables for player.

A *strategy* for a process p is a function $\mathtt{s}_p : (2^{\mathrm{I}_p})^* \to 2^{\mathrm{O}_p}$ mapping each history of visible truth-assignments to a truth-assignment of the output variables. A *strategy profile* $\vec{\mathtt{s}}$ is a tuple of strategies, one for each non-idle process. A strategy profile generates a run $\rho(\vec{\mathtt{s}})$ over the set of variables Vr. An *implementation* S is a set of strategies one for each process in $\mathrm{P}^- \triangleq \mathrm{P} \setminus \{p_0, p_{\mathtt{idle}}\}$. We say that a profile strategy $\vec{\mathtt{s}}$ is *consistent* with an implementation S if the strategy in S corresponds to the one associated in $\vec{\mathtt{s}}$, for each process p in P^-. Finally, for a given LTL formula φ, we say that an implementation S *realizes* φ if $\rho(\vec{\mathtt{s}}) \models \varphi$ for all strategy profiles $\vec{\mathtt{s}}$ that are consistent with S.

Definition 1. *For a given architecture $\mathcal{A}$ and an* LTL *specification φ, the synthesis problem for $\mathcal{A}$ and φ consists of finding an implementation* S *in $\mathcal{A}$ that realizes φ.*

Theorem 1 ((Finkbeiner and Schewe 2005))**.** *The uniform distributed synthesis problem for a generic architecture $\mathcal{A}$ with three players and an* LTL *formula φ is undecidable.*

Undecidability

In this section, we show that the uniform distributed synthesis problem (Finkbeiner and Schewe 2005) for LTL formulae can be reduced to the NONEMPTINESS problem for SRMLI games with LTL goals. To do this, we first need to introduce some auxiliary definitions and notations.

First of all, note that the fact that an architecture $\mathcal{A}$ is acyclic provides a partial order among processes, which can be extended to a total order $<_P$ in a consistent way. Moreover, starting from $<_P$, we can totally order the set of variables in a way that, for each $x, y \in \Phi$, if $x \in \Phi_{p_1}$, $y \in \Phi_{p_2}$, and $p_1 <_P p_2$, then $x <_\Phi y$. Thus, every variable x can be associated to a different natural number $i \in \{1, \ldots, d = |\Phi|\}$, denoting its position in the ordering $<_\Phi$, and renamed with x_i. Note that, if x_i is in a variable depending on some variable x_j in the architecture, then it holds that $j < i$ and so that $x_j <_\Phi x_i$. At this point, for a given process p, we define the corresponding module m_p as follows.

module $\mathtt{m}_p$ **controls** $\mathrm{O}(p)$ **under** $\mathrm{I}(p) \cup \{1, \ldots, d\}$
init
$:: i \rightsquigarrow x_i' := \bot$ for $x_i \in \mathrm{O}(p)$
$:: i \rightsquigarrow x_i' := \top$ for $x_i \in \mathrm{O}(p)$
update
$:: i \rightsquigarrow x_i' := \bot$ for $x_i \in \mathrm{O}(p)$
$:: i \rightsquigarrow x_i' := \top$ for $x_i \in \mathrm{O}(p)$

We also need to keep track of the moment when variables have to be updated. To simplify our presentation, we do this with the use of an additional module, `var`, defined below.

module $\mathtt{var}_d$ **controls** $\{1, \ldots, d\}$ **under** $\emptyset$
init
$:: \top \rightsquigarrow 1' := \top; 2' := \bot; \ldots; d' := \bot$
update
$:: i \rightsquigarrow i' := \bot; (i+1)' := \top$ for $i \neq d$
$:: d \rightsquigarrow d' := \bot; 1' := \top;$

We can now define the arena having all the modules defined above, one per each process plus the auxiliary module `var` to give turns to the variables:

$$\mathrm{A}_{\mathcal{A}} = \langle n+2, \Phi, \mathtt{var}, \mathtt{m}_{p_0}, \mathtt{m}_{p_1}, \ldots, \mathtt{m}_{p_n} \rangle.$$

At this point, we describe a fundamental translation Γ of strategies, making a suitable bridge between an architecture $\mathcal{A}$ and the corresponding arena $\mathrm{A}_{\mathcal{A}}$. The translation shows that a strategy for a process of an architecture, $\mathcal{A}$, can be represented in our framework, $\mathrm{A}_{\mathcal{A}}$. Let $\mathtt{s} : (2^{\mathrm{I}_p})^* \to 2^{\mathrm{O}_p}$ be a strategy for process p. Then, we define the strategy $\Gamma(\mathtt{s}) = \sigma : (2^{Vis(m_p)})^* \to 2^{\mathrm{Vr}(m_p)}$ such that, for any given variable $x_i \in \mathrm{O}(p)$ and history $h \in (2^{Vis(m_p)})^*$ we have:

$$\Gamma(\mathtt{s})(h)(x_i) = \begin{cases} \mathtt{s}(h^m)(x_i), & \text{if } |h| \equiv_d i \\ \mathbf{skip}, & \text{otherwise} \end{cases}$$

where h^d is the partial run defined from h. It is not hard to see that, for a given strategy σ, for a module m corresponding to process p, there is a unique strategy $\mathtt{s}$ with $\Gamma(\mathtt{s}) = \sigma$. So, the function Γ is bijective. Moreover, for a given implementation $\vec{\mathtt{s}}$, by overlapping of the notation, by $\Gamma(\vec{\mathtt{s}})$ we denote the profile strategy assigning $\Gamma(\mathtt{s})$ to the module m corresponding to the process p. We can now prove the following lemma, which gives a further characterisation of strategy profiles in the uniform distributed synthesis problem.

Lemma 2. *Let $\mathcal{A}$ be an architecture with $d = |\Phi|$ variables and $\mathrm{A}_{\mathcal{A}}$ be the corresponding* SRMLI *arena. Then:*

1. *For each profile $\vec{\mathtt{s}}$ it holds that $\rho(\vec{\mathtt{s}}) = \rho(\Gamma(\vec{\mathtt{s}}))^d$;*
2. *For each profile $\vec{\sigma}$ it holds that $\rho(\vec{\sigma}) = \rho(\Gamma^{-1}(\vec{\sigma}))^d$.*

We now introduce two additional modules to $\mathrm{A}_{\mathcal{A}}$, named m_A and m_B, which will be used to make an easy connection between the solution of NONEMPTINESS and the uniform distributed synthesis problem. These two additional modules, as well as `var`, can be removed in a more general construction. However, we prefer to have them here, again, to simplify our presentation. Modules m_A and m_B simply control one Boolean

variable each, namely *a* and *b* respectively, which they can set to any Boolean value they want at initialisation, and cannot modify thereafter. Call $\mathrm{A}_{\mathcal{A}}{}'$ such an SRMLI system.

Observe that whereas module `var` has only one possible strategy, modules m_A and m_B have only two possible strategies, namely, set *a* to true or to false, and similarly for *b*. Because of this, we often can reason about strategy profiles where we simply consider the other modules, $\mathtt{m}_{p_0}, \mathtt{m}_{p_1}, \ldots, \mathtt{m}_{p_n}$, and the cases given by the possible Boolean values, and therefore strategies, for *a* and *b*.

Theorem 2. *Let $\mathcal{A}$ be an architecture with $d = |\Phi|$ variables and φ an* LTL *specification. Moreover, consider the* SRMLI $G = \langle \mathrm{A}_{\mathcal{A}}{}', \psi_{\mathtt{var}}, \psi_0, \psi_1, \ldots, \psi_n, \psi_A, \psi_B \rangle$ *such that $\mathrm{A}_{\mathcal{A}}{}'$ is the arena derived by $\mathcal{A}$, $\psi_{\mathtt{var}} = \top$, $\psi_0 = \neg\tau_d(\varphi)$, $\psi_1 = \ldots = \psi_n = \tau_d(\varphi)$, and $\psi_A = \tau_d(\varphi) \vee (a \leftrightarrow b)$ and $\psi_B = \tau_d(\varphi) \vee \neg(a \leftrightarrow b)$. Then, the architecture $\mathcal{A}$ realizes the* LTL *formula φ if and only if G has a Nash equilibrium.*

Proof. We prove the theorem by double implication.

($\Rightarrow$). Assume that $\mathcal{A}$ realizes φ. Then, there exists a winning strategy $\mathtt{s}^{\rightarrow}_{-\mathrm{o}}$ for $p_1, \ldots, p_n, m_A, m_B, \mathtt{var}$ such that $\rho(\mathtt{s}_{\mathrm{o}}, \mathtt{s}^{\rightarrow}_{-\mathrm{o}}) \models \varphi$, for all possible strategies $\mathtt{s}_{\mathrm{o}}$ for the environment p_0. Consider the strategy $\Gamma(\mathtt{s}^{\rightarrow}_{-\mathrm{o}})$ given for the modules $m_1, \ldots m_n, m_A, m_B, \mathtt{var}$, and consider a strategy σ_{o} for module m_{o}. By Lemma 2, we have that $\rho(\Gamma^{-1}(\sigma_{\mathrm{o}}), \mathtt{s}^{\rightarrow}_{-\mathrm{o}}) = \rho(\sigma_{\mathrm{o}}, \Gamma(\mathtt{s}^{\rightarrow}_{-\mathrm{o}}))|_{\Phi} \models \tau_d(\varphi)$. Then, by Lemma 1, we have that $\rho(\sigma_{\mathrm{o}}, \Gamma(\mathtt{s}^{\rightarrow}_{-\mathrm{o}})) \models \tau_d(\varphi)$. Moreover, the strategy profile $(\sigma_{\mathrm{o}}, \Gamma(\mathtt{s}^{\rightarrow}_{-\mathrm{o}}))$ is a Nash equilibrium. Indeed, $m_1, \ldots, m_n, m_A, m_B, \mathtt{var}$ have their goal satisfied and so have no incentive to deviate. On the other hand, assume by contradiction that module m_{o} has a strategy σ'_{o} such that $\rho(\sigma'_{\mathrm{o}}, \Gamma(\mathtt{s}^{\rightarrow}_{-\mathrm{o}})) \models \neg\tau_d(\varphi)$. Then, due to Lemmas 1 and 2, we have $\rho(\sigma'_{\mathrm{o}}, \Gamma(\mathtt{s}^{\rightarrow}_{-\mathrm{o}})) \models \neg\tau_d(\varphi)$ and therefore $\mathtt{s}^{\rightarrow}_{-\mathrm{o}}$ is not winning, which is a contradiction.

($\Leftarrow$). Let $(\sigma_{\mathrm{o}}, \sigma^{\rightarrow}_{-\mathrm{o}}) \in NE(G)$. Because of modules m_A and m_B, it must be the case that $\rho(\sigma_{\mathrm{o}}, \sigma^{\rightarrow}_{-\mathrm{o}}) \models \tau_d(\varphi)$. Then, consider the strategy profile $\Gamma^{-1}(\sigma^{\rightarrow}_{-\mathrm{o}})$. We have that it is a winning strategy. Indeed, assume by contradiction that $\rho(\mathtt{s}_{\mathrm{o}}, \Gamma^{-1}(\sigma^{\rightarrow}_{-\mathrm{o}})) \models \neg\tau_d(\varphi)$ for some environment strategy $\mathtt{s}_{\mathrm{o}}$. Then, by Lemmas 1 and 2, we obtain that $\rho(\Gamma(\mathtt{s}_{\mathrm{o}}), \sigma^{\rightarrow}_{-\mathrm{o}}) \models \neg\tau_d(\varphi)$ and so the strategy $\Gamma(\mathtt{s}_{\mathrm{o}})$ incentives module m_{o} to deviate from the strategy profile $(\sigma_{\mathrm{o}}, \sigma^{\rightarrow}_{-\mathrm{o}})$, which is a contradiction. □

Because the uniform distributed synthesis problem is undecidable with three processes, we can restrict ourselves to that setting, where m_{o} can be extended to take care of turns (to eliminate `var`) and the behaviour of m_A and m_B can be encoded into that of m_1 and m_2 respectively (to eliminate m_A and m_B), to obtain a 3-player SRMLI.

Corollary 1. NONEMPTINESS *for* SRMLI *games with LTL goals is undecidable for games with more than two players.*

In fact, the uniform synthesis problem is undecidable for logics even weaker than full LTL. However, the main construction heavily relies on the existence of at least three players. Because of this, we now study the case where we still allow LTL goals, but restrict to systems with only 2 players.

Decidability of 2-Player Games

In this section, we show that SRMLI games with two players are decidable. More importantly, we show that this class of games can be solved using a logic-based approach: as shown next, NONEMPTINESS for games with two players and LTL goals can be reduced to a series of temporal logic synthesis problems. This approach provides a mechanical solution using already known automata-theoretic techniques originally developed for LTL and CTL* synthesis with imperfect information. Let us first, in the next two subsections, present some useful technical results and notations.

On the power of myopic strategies

Myopic strategies are strategies whose transition function do not depend on the values of the variables it reads, but only on the states where they are evaluated at. We say that a strategy $\sigma_i = (Q_i, q_i^0, \delta_i, \tau_i)$ is *myopic* if, for every $q \in Q_i$ and $\Psi, \Psi' \subseteq \mathrm{Vis}_i$, we have $\delta_i(q, \Psi) = \delta_i(q, \Psi')$. Myopic strategies are powerful enough to describe any ω-regular run—an ultimately periodic run (Sistla and Clarke 1985):

Lemma 3. *For every ω-regular run ρ, if $\rho = \rho(\vec{\sigma})$ for some profile $\vec{\sigma} = (\sigma_1, \ldots, \sigma_n)$, then for every $i \in \{1, \ldots, n\}$ there is a myopic strategy σ'_i such that $\rho = \rho(\vec{\sigma}_{-i}, \sigma'_i)$.*

What is important to observe about myopic strategies is that once they are defined for a given module, the same myopic strategies can be defined for all modules with (at least) the same guarded commands. This observation is used to show the following result about the preservation of myopic strategies in games with imperfect information.

Lemma 4. *Let $m_i = \langle \Phi_i, \mathrm{Vis}_i, I_i, U_i \rangle$ be an* SRMLI *module of a game with variable set Φ. If σ_i is a myopic strategy of module m_i then σ_i is also a strategy of $m'_i = \langle \Phi_i, \mathrm{Vis}_i{}', I_i, U_i \rangle$, for every set $\mathrm{Vis}_i \subseteq \mathrm{Vis}_i{}' \subseteq \Phi$.*

Moreover the following lemma, whose proof relies on Lemmas 3 and 4, is key to obtain the decidability result for two-player games presented later. Informally, it states that if a player has a winning strategy, then such a strategy is winning even if the other player has perfect information.

Lemma 5. *Let G be an* SRMLI *game with two modules, $m_1 = \langle \Phi_1, \mathrm{Vis}_1, I_1, U_1 \rangle$ and $m_2 = \langle \Phi_2, \mathrm{Vis}_2, I_2, U_2 \rangle$, and $i, j \in \{1, 2\}$. Then, σ_i is a winning strategy of player i for φ if and only if σ_i is a winning strategy for φ in the game G' where $m'_j = \langle \Phi_j, \Phi, I_j, U_j \rangle$, with $i \neq j$.*

From synthesis to Nash equilibria

The behaviour of reactive modules can be characterised in LTL using formulae that are polynomial in the size of the modules. Then, given a module m_i, we will write $TH(m_i)$ for such an LTL formula, which satisfies, for all runs ρ, that ρ is a run of m_i iff it is a run satisfying $TH(m_i)$. Observe that $TH(m_i)$ is a satisfiable formula and, in particular, it is satisfied by any module or Kripke structure whose runs are exactly those of m_i. Moreover, we use the following notation. For a synthesis problem with imperfect information, where φ is the formula to be synthesised, I is the set of input variables, $E \subseteq I$ is the set of visible input variables, and O is the set of output variables, we write SYN(φ, O, E, I). We will

consider synthesis problems where φ is an LTL or a CTL* formula. In particular, in case φ is a CTL* formula, we use the standard notation and semantics in the literature (Emerson 1990): informally, the CTL* formula $\mathbf{E}\,\psi$ means "there is a path where formula ψ holds" whereas the CTL* formula $\mathbf{A}\,\psi$ means "on all paths, formula ψ holds."

With the above in mind, consider the algorithm in Figure 2, which can be used to solve NONEMPTINESS in the setting we are considering, where the input SRMLI game is

$$G_2 = (\{1,2\}, \Phi_1, \Phi_2, m_1, m_2, \gamma_1, \gamma_2)$$

and the following abbreviations are used

- $coop = \gamma_1 \wedge \gamma_2 \wedge TH(m_1) \wedge TH(m_2)$
- $block_1 = TH(m_1) \rightarrow (\neg\gamma_1 \wedge TH(m_2))$
- $block_2 = TH(m_2) \rightarrow (\neg\gamma_2 \wedge TH(m_1))$
- $nodev_1 = \mathbf{A}\,TH(m_1) \rightarrow (\mathbf{E}\,\gamma_2 \wedge \mathbf{A}\,\neg\gamma_1 \wedge \mathbf{A}\,TH(m_2))$
- $nodev_2 = \mathbf{A}\,TH(m_2) \rightarrow (\mathbf{E}\,\gamma_1 \wedge \mathbf{A}\,\neg\gamma_2 \wedge \mathbf{A}\,TH(m_1))$

where each formula characterises the following situations: for *coop*, the case where both γ_1 and γ_2 are satisfied while respecting the behaviour of both modules, m_1 and m_2; for $block_1/block_2$, the case where $\neg\gamma_1/\neg\gamma_2$ is satisfied while respecting the behaviour of module m_2/m_1, provided that the behaviour of m_1/m_2 is respected too—*i.e.*, a case where module 2/1 "blocks" or prevents module 1/2 from achieving its goal; for $nodev_1/nodev_2$, the case where $\neg\gamma_1/\neg\gamma_2$ is satisfied in all possible runs, with at least one satisfying γ_2/γ_1, while respecting the behaviour of module m_2/m_1, provided that the behaviour of module m_1/m_2 is respected too—*i.e.*, a case where module 2/1 ensures that module 1/2 has no incentive to deviate from any run satisfying $nodev_1/nodev_2$ to another run that also satisfies such a formula.

Nonemptiness(G_2)
1. if *coop* is satisfiable then
   return "yes"
2. if SYN($block_1$, Φ_2, Vis_2, Φ_1) and
   SYN($block_2$, Φ_1, Vis_1, Φ_2) then
   return "yes"
3. if SYN($nodev_1$, Φ_2, Vis_2, Φ_1) or
   SYN($nodev_2$, Φ_1, Vis_1, Φ_2) then
   return "yes"
4. return "no"

Figure 2: Algorithm for NONEMPTINESS in 2-player games.

Using Nonemptiness(G_2), which in turn makes use of algorithms for LTL satisfiability (PSPACE-complete) as well as CTL* and LTL synthesis with imperfect information (both 2EXPTIME-complete), one can show that, in a two-player SRMLI game with LTL goals, there exists a Nash equilibrium if and only if one of the following three cases holds:

1. both players have their goals satisfied; or
2. both players have winning strategies for the negation of the other player's goal; or
3. some player, say *i*, has a winning strategy for the negation of the other player's goal, while at least one of the runs allowed by such a winning strategy satisfies player *i*'s goal.

Theorem 3. NONEMPTINESS *for two-player* SRMLI *games with* LTL *goals is 2EXPTIME-complete.*

Sketch of the proof. To prove the correctness (soundness and completeness) of the algorithm we first assume that there is a Nash equilibrium and check that at least one of the three possible cases that deliver a positive answer is successfully executed. In particular, for steps 2 and 3, we use the fact that, because of Lemma 5, we can assume that when checking $block_1/nodev_1$ (resp. $block_2/nodev_2$) only player 2 (resp. 1) has imperfect information—the "verifier" in the associated synthesis game—whereas the other player—the "falsifier" in the associated synthesis game—has perfect information. In addition, we also check that if a game does not have a Nash equilibrium, then steps 1–3 fail, and therefore step 4 is executed, thus delivering again the correct answer. For hardness we reduce from the LTL synthesis problem. □

From Nash to strong Nash equilibria

Even though Nash equilibrium is the best-known solution concept in non-cooperative game theory (Osborne and Rubinstein 1994), it has been criticised on various grounds—most importantly, because it is not in fact a very stable solution concept. In order to address this problem, other solution concepts have been proposed, among them being the notion of *strong* Nash equilibrium. Strong Nash equilibrium considers the possibility of players forming coalitions in order to deviate from a strategy profile. Formally, a strategy profile $\vec{\sigma} = (\sigma_1, \ldots, \sigma_n)$, with $N = \{1, \ldots, n\}$, is a *strong Nash equilibrium* if for each $C \subseteq N$ and set of strategies σ'_C of C, there is $i \in C$ such that

$$\vec{\sigma} \succsim_i (\vec{\sigma}_{-C}, \sigma'_C).$$

Then, in a strong Nash equilibrium a coalition of players C has an incentive to deviate if and only if every player i in such a coalition has an incentive to deviate. Let $sNE(G)$ be the set of strong Nash equilibria of an SRMLI game G and S-NONEMPTINESS be NONEMPTINESS, but with respect to $sNE(G)$. Then, we can prove the following.

Theorem 4. S-NONEMPTINESS *for two-player* SRMLI *games with* LTL *goals is 2EXPTIME-complete.*

Sketch of the proof. A two-player SRMLI game G has a strong Nash equilibrium iff G has a Nash equilibrium. The ($\Rightarrow$) direction is trivial. For the other direction, ($\Leftarrow$), we only need to check the case where $|C| = 2$ and neither player has its goal achieved. In such a case, if there is a beneficial deviation where both players get their goals achieved, say to a strategy profile $\sigma'_C = (\sigma'_1, \sigma'_2)$, then we obtain a profile that is, in particular, both a Nash equilibrium and a strong Nash equilibrium—as both players get their goals achieved. □

What we can learn from (the proof of) Theorem 4 is that cooperation in this setting does not help from the point of view of the existence of equilibria. Instead, it may help to obtain better equilibria. This is because if there is a profile that is not a strong Nash equilibrium but it is a Nash equilibrium, necessarily, it is one where neither player achieves its goal. However, as such a profile is not a strong Nash equilibrium, there must be another one where both achieve their goals.

Games with Memoryless Nash Equilibria

Another way of obtaining a class of SRMLI games that is decidable is by restricting the kind of strategies the players in the game are allowed to use, rather than by restricting the number of players in the game. This is the issue that we study in this section. We consider the class of games where a Nash equilibrium strategy profile can be defined only in terms of *memoryless* strategies. We do not restrict the strategies the players may use to deviate. More specifically, in this section, we show that the NONEMPTINESS problem for this class of SRMLI games is NEXPTIME-complete.

The solution to this variant of the general problem is given by the non-deterministic algorithm presented in Figure 3.

Nonemptiness(G)
1. Guess $\vec{\sigma}$
2. If $\vec{\sigma} \in NE(G)$ then return "yes"
3. return "no"

Figure 3: Algorithm to solve NONEMPTINESS in SRMLI games with Nash equilibria in memoryless strategies.

Whereas step 1 can be done in NEXPTIME, step 2 can be done in EXPTIME, leading to an NEXPTIME algorithm. Moreover, hardness in NEXPTIME follows from the fact that the satisfiability problem for Dependency Quantified Boolean Formulae (DQBF) can be reduced to NONEMPTINESS for this class of games. The non-deterministic algorithm in Figure 3 relies on the following intermediate results.

Lemma 6. *Let G be a game with memoryless Nash equilibria. If $\vec{\sigma} \in NE(G)$, for some $\vec{\sigma} = (\sigma_1, \ldots, \sigma_i, \ldots, \sigma_n)$, then σ_i is at most exponential in the size of G, for every σ_i in $\vec{\sigma}$.*

Sketch of the proof. First, construct the Kripke structure induced by G. Such a structure, denoted by K_G, is at most exponential in the size of G. Because we only consider (equilibria in) memoryless strategies, such strategies cannot be bigger than $|K_G|$, thus at most exponential in the size of G. □

Lemma 6 is used to do step 1 of the algorithm in Figure 3. In addition, the lemma below—which relies on the fact that LTL model checking, say for an instance $K \models \varphi$ where K is a model and φ an LTL formula, is polynomial in $|K|$ and exponential in $|\varphi|$—is used to do step 2 of the algorithm.

Lemma 7. *Let $G = (A, \gamma_1, \ldots, \gamma_n)$ be a game with memoryless Nash equilibria and $\vec{\sigma}$ a strategy profile in G. Checking whether $\vec{\sigma} \in NE(G)$ can be done in time exponential in $|A|$ and exponential in $|\gamma_1| + \ldots + |\gamma_n|$.*

Then, Lemmas 6 and 7 can be used to show:

Theorem 5. NONEMPTINESS *for* SRMLI *games with memoryless Nash equilibria is NEXPTIME-complete.*

Sketch of the proof. This NONEMPTINESS problem is solved using the non-deterministic algorithm in Figure 3. That the algorithm runs in NEXPTIME follows from the fact that if a Nash equilibrium exists, due to Lemma 6, such a strategy profile can be guessed in NEXPTIME and verified to be a Nash equilibrium in EXPTIME, using Lemma 7. For hardness, we reduce from the satisfiability problem for DQBF, which is known to be NEXPTIME-complete. □

Bounded Rationality

Another interesting game-theoretic setting, which is commonly found in the literature, is the one where we assume that the agents in the system have only "bounded rationality." This is modelled, for instance, by assuming that the number of rounds of the game is finite or that strategies are of some bounded size. From the computational point of view, a natural assumption is that strategies are at most polynomial in the size of the module specifications they are associated with. Under this latter assumption, we can use the algorithm in Figure 3 to show these games can be solved in PSPACE.

Since strategies are polynomially bounded, step 1 can be done in NPSPACE by guessing $\vec{\sigma}$. Furthermore, step 2 can also be done in NPSPACE: whenever a goal γ_j is not satisfied by $\vec{\sigma}$, we can check in NPSPACE if there is a polynomially bounded strategy σ'_j such that $\rho(\vec{\sigma}_{-j}, \sigma'_j) \models \gamma_j$. For hardness, we use a reduction from the LTL model checking problem for compressed words (Markey and Schnoebelen 2003).

Theorem 6. NONEMPTINESS *for* SRMLI *games G with strategies polynomially bounded by $|G|$ is PSPACE-complete.*

Local Reasoning

Another decidable, and simple, class of SRMLI games with LTL goals where the reasoning power of the players is also restricted is the class of games where only myopic strategies are allowed. Such games, which we call myopic SRMLI games, can be solved in EXPSPACE. A particular feature of this class of games is that in this setting players cannot be informed by the behaviour of other players in the game. As a consequence, all reasoning must be done in a purely local way. Indeed, these are "zero-knowledge" games with respect to the information that could be obtained from each module's environment, that is, from the other modules in the system.

Firstly, given a myopic SRMLI game $G = (A, \gamma_1, \ldots, \gamma_n)$, let $\varphi_A = \bigwedge_{i \in N} TH(m_i)$ be the LTL formula characterising the behaviour of the modules in A. Now, to check if there is a strategy profile in myopic strategies we check if the following Quantified LTL (QPTL) formula is satisfiable:

$$\bigvee_{W \subseteq N} \left(\varphi_A \wedge \exists \Phi_1, \ldots, \Phi_n . \left(\bigwedge_{i \in W} \gamma_i \wedge \bigwedge_{j \in N \setminus W} \left(\forall \Phi_j . \neg \gamma_j \right) \right) \right)$$

such that $\exists \Phi_i$ stands for $\exists p_i^1, \ldots, p_i^{|\Phi_i|}$, where Φ_i is the set of Boolean variables $\{p_i^1, \ldots, p_i^{|\Phi_i|}\}$, and similarly for $\forall \Phi_i$. Such a formula is in Σ_2^{QPTL}. Therefore, by (Sistla, Vardi, and Wolper 1987), its satisfiability problem is in EXPSPACE and has an ω-regular (ultimately periodic) model. Moreover, the formula is satisfied by all runs satisfying the modules' specifications (given by φ_A) where a set of "winners" (given by W) get their goals achieved and a set of "losers" (given by $N \setminus W$) cannot deviate. The semantics of QPTL (Sistla, Vardi, and Wolper 1987) ensures that models of such a formula have a game interpretation using the definition of myopic strategies. Finally, for hardness, we use a reduction from the satisfiability problem of Σ_2^{QPTL} formulae. Formally, we have:

Theorem 7. NONEMPTINESS *for myopic* SRMLI *games is EXPSPACE-complete.*

Refinement and Preservation of Equilibria

In this section, we show that the monotonic increase of players' knowledge may only induce an increase in the number of strategy profiles in the set of Nash equilibria of a game with imperfect information, if any. For instance, this situation is illustrated with the following example.

Example 1. *Consider the two-player* SRMLI *game* $G = (\Phi = \{p, q\}, m_1, m_2, \gamma_1 = p \leftrightarrow \mathbf{X}q, \gamma_2 = p \leftrightarrow \mathbf{X}\neg q)$ *such that module* m_1/m_2 *controls variable* p/q *and has visibility set* Φ_1/Φ_2 *and can set* p/q *to any Boolean value at all rounds in the game. Moreover, consider the game* G'*, defined just as* G *save that* m_2 *has visibility set* Φ*. It is easy to see that whereas* $NE(G) = \emptyset$*, we have that* $NE(G') \neq \emptyset$*.*

More specifically, in this section, we present a general result about the *preservation of Nash equilibria*, provided that no player's knowledge about the overall system is decreased. A key technical result to prove this is that a player's deviation is a "zero-knowledge" process, which can be implemented in our game-theoretic framework using only myopic strategies, as stated by the following lemma.

Lemma 8. *Let* G *be an* SRMLI *game and* $\vec{\sigma} \notin NE(G)$*. Then, there is a player* j *and a myopic strategy* σ_j' *for player* j *such that* $\rho(\vec{\sigma}) \not\models \gamma_j$ *and* $\rho(\vec{\sigma}_{-j}, \sigma_j') \models \gamma_j$*.*

Using Lemma 8, we can show that the power a module m_j has to beneficially deviate from a strategy profile is neither increased by giving such a module more knowledge by increasing its visibility set Vis_j nor decreased by giving such a module less knowledge by decreasing its visibility set Vis_j (as long as all guarded commands are kept). Formally, the next theorem about the preservation of Nash equilibria holds.

Theorem 8. *Let* G *and* G' *be two* SRMLI *games, with*

- $G = ((N, \Phi, m_1, \ldots, m_n), \gamma_1, \ldots, \gamma_n)$ *and*
- $G' = ((N, \Phi, m_1', \ldots, m_n'), \gamma_1, \ldots, \gamma_n)$,

such that for each $i \in N$ *we have* $m_i = \langle \Phi_i, \mathrm{Vis}_i, I_i, U_i \rangle$*, and* $m_i' = \langle \Phi_i, \mathrm{Vis}_i{}', I_i, U_i \rangle$*, and* $\mathrm{Vis}_i \subseteq \mathrm{Vis}_i{}'$*. Then,*

$$NE(G) \subseteq NE(G').$$

Proof. We prove that, under the assumptions of the theorem, if $\vec{\sigma} \in NE(G)$ then $\vec{\sigma} \in NE(G')$. First, observe that if $\vec{\sigma}$ is a strategy profile in G, then so is in G', since for each player j and strategy σ_j of j in G, such a strategy σ_j is also available to j in G'. Now, suppose, for a contradiction, that $\vec{\sigma} \in NE(G)$ and $\vec{\sigma} \notin NE(G')$. Then, because of Lemma 8, we know that there is a player j and a myopic strategy σ_j' of j such that $\rho(\vec{\sigma}) \not\models \gamma_j$ and $\rho(\vec{\sigma}_{-j}, \sigma_j') \models \gamma_j$. And, due to Lemma 4, we know that σ_j' is a myopic strategy that is available to player j in both G and G'. However, this means that player j could also beneficially deviate in G and therefore that $\vec{\sigma}$ is not a Nash equilibrium of G, which is a contradiction. □

Since we know that NONEMPTINESS with LTL goals is decidable in 2EXPTIME for perfect-information games, but undecidable for imperfect-information games, one can also show that, in general, the other direction, namely in which $NE(G') \subseteq NE(G)$, does not hold, as otherwise we would have $NE(G) = NE(G')$ under the assumptions of the theorem. Alternatively, a counter-example to such an equality can be shown, *e.g.*, as the one given by Example 1 above.

Related Work, Conclusions, and Future Work

Imperfect information in logic-based games. There is a long tradition in logic and theoretical computer science of using logic-based games to characterise complexity classes, using both perfect and imperfect information games; see, *e.g.*, (Stockmeyer and Chandra 1979; Peterson, Reif, and Azhar 2001). In this paper, we obtain similar results, in particular, in a multi-player imperfect information setting. A summary of our results from this viewpoint is in Table 1.

	General case	Memoryless	Poly. Bound.	Myopic
n-Pl	Undecidable	NEXPTIME-c	PSPACE-c	EXPSPACE-c
2-Pl	2EXPTIME-c	NEXPTIME	PSPACE	EXPSPACE

Table 1: Summary of results for n-player games (n-Pl) and 2-player games (2-Pl). Abbreviations, with X a complexity class: X-c means X-complete and X means in X.

Imperfect information in logics for strategic reasoning. Although logics for games have been studied since at least the 1980s (Parikh 1985), recent interest in the area was prompted by the development of logics to reason about strategic ability; see, *e.g.*, (Alur, Henzinger, and Kupferman 2002; Chatterjee, Henzinger, and Piterman 2010; Gutierrez, Harrenstein, and Wooldridge 2014; Mogavero et al. 2014). Some of these logics are even able to express the existence of Nash equilibria for games in which players have LTL goals. Also, some variations that can deal with imperfect information have been already studied (van der Hoek and Wooldridge 2003; Ågotnes et al. 2015). However, a comprehensive study under imperfect information is far from complete.

Solving games with imperfect information. There is much work in the multi-agent systems community on *solving* incomplete information games (*e.g.*, poker), although this work does not use logic (Sandholm 2015). The main challenge in this work is dealing with large search spaces: naive approaches fail even on the most trivial cases. It would be interesting to explore if such problems can be addressed using logic-based methods—techniques developed for verification, *e.g.*, model checking, have been usefully applied in similar settings. Other issues for future work include more work on mapping out the complexity landscape of our games (see Table 1), and on practical implementations; see, *e.g.*, (Lomuscio, Qu, and Raimondi 2009; Cermák et al. 2014; Toumi, Gutierrez, and Wooldridge 2015).

On iBG and SRMLI **games.** Our model of study, SRMLI games, is a natural extension of the iBG model as defined in (Gutierrez, Harrenstein, and Wooldridge 2015b). There are many reasons from theoretical and practical viewpoints to consider the more general model given by SRMLI games, for instance, in the verification of multi-agent systems (Wooldridge et al. 2016). In particular, while players in iBG have unrestricted powers (as used for synthesis or satisfiability), players in SRMLI games can have restricted behaviour (as needed to model real-world multi-agent systems).

Acknowledgements: We thank the support of the ERC Advanced Investigator Grant 291528 ("Race") at Oxford.

References

Ågotnes, T.; Goranko, V.; Jamroga, W.; and Wooldridge, M. 2015. Knowledge and ability. In *Handbook of Epistemic Logic*. College Publications.

Alur, R., and Henzinger, T. A. 1999. Reactive Modules. *Formal Methods in System Design* 15(1):7–48.

Alur, R.; Henzinger, T.; Mang, F.; Qadeer, S.; Rajamani, S.; and Tasiran, S. 1998. MOCHA: modularity in model checking. In *International Conference on Computer-Aided Verification (CAV)*, volume 1427 of *LNCS*, 521–525. Springer.

Alur, R.; Henzinger, T. A.; and Kupferman, O. 2002. Alternating-time temporal logic. *Journal of the ACM* 49(5):672–713.

Bouyer, P.; Brenguier, R.; Markey, N.; and Ummels, M. 2011. Nash equilibria in concurrent games with Büchi objectives. In *Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, volume 13 of *LIPIcs*, 375–386. Schloss Dagstuhl.

Bouyer, P.; Brenguier, R.; Markey, N.; and Ummels, M. 2012. Concurrent games with ordered objectives. In *Conference on Foundations of Software Science and Computation Structures (FOSSACS)*, volume 7213 of *LNCS*, 301–315. Springer.

Cermák, P.; Lomuscio, A.; Mogavero, F.; and Murano, A. 2014. MCMAS-SLK: A model checker for the verification of strategy logic specifications. In *International Conference on Computer-Aided Verification (CAV)*, volume 8559 of *LNCS*, 525–532. Springer.

Chatterjee, K., and Henzinger, T. A. 2012. A survey of stochastic ω-regular games. *Journal of Computer and System Sciences* 78(2):394–413.

Chatterjee, K.; Henzinger, T.; and Piterman, N. 2010. Strategy logic. *Information and Computation* 208(6):677–693.

Clarke, E. M.; Grumberg, O.; and Peled, D. A. 2000. *Model Checking*. The MIT Press: Cambridge, MA.

Emerson, E. A. 1990. Temporal and modal logic. In *Handbook of Theoretical Computer Science Volume B: Formal Models and Semantics*. Elsevier. 996–1072.

Finkbeiner, B., and Schewe, S. 2005. Uniform distributed synthesis. In *Annual Symposium on Logic in Computer Science (LICS)*, 321–330. IEEE Computer Society.

Ghica, D. R. 2009. Applications of game semantics: From program analysis to hardware synthesis. In *Annual Symposium on Logic in Computer Science (LICS)*, 17–26. IEEE Computer Society.

Gutierrez, J.; Harrenstein, P.; and Wooldridge, M. 2014. Reasoning about equilibria in game-like concurrent systems. In *International Conference on Principles of Knowledge Representation and Reasoning (KR)*, 408–417. AAAI Press.

Gutierrez, J.; Harrenstein, P.; and Wooldridge, M. 2015a. Verification of temporal equilibrium properties of games on Reactive Modules. Technical report, University of Oxford.

Gutierrez, J.; Harrenstein, P.; and Wooldridge, M. 2015b. Iterated Boolean games. *Information and Computation* 242:53–79.

Kwiatkowska, M. Z.; Norman, G.; and Parker, D. 2011. PRISM 4.0: Verification of probabilistic real-time systems. In *International Conference on Computer-Aided Verification (CAV)*, volume 6806 of *LNCS*, 585–591. Springer.

Lomuscio, A.; Qu, H.; and Raimondi, F. 2009. MCMAS: A model checker for the verification of multi-agent systems. In *International Conference on Computer-Aided Verification (CAV)*, volume 5643 of *LNCS*, 682–688. Springer.

Markey, N., and Schnoebelen, P. 2003. Model checking a path. In *International Conference on Concurrency Theory (CONCUR)*, volume 2761 of *LNCS*, 248–262. Springer.

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

Osborne, M. J., and Rubinstein, A. 1994. *A Course in Game Theory*. The MIT Press: Cambridge, MA.

Parikh, R. 1985. The logic of games and its applications. In *Topics in the Theory of Computation*. Elsevier.

Peterson, G. L.; Reif, J. H.; and Azhar, S. 2001. Lower bounds for multiplayer noncooperative games of incomplete information. *Computers and Mathematics with Applications* 41:957–992.

Pnueli, A., and Rosner, R. 1989. On the synthesis of an asynchronous reactive module. In *International Colloquium on Automata, Languages, and Programs (ICALP)*, volume 372 of *LNCS*, 652–671. Springer.

Sandholm, T. 2015. Solving imperfect-information games. *Science* 347(6218).

Sistla, A. P., and Clarke, E. M. 1985. The complexity of propositional linear temporal logics. *Journal of the ACM* 32(3):733–749.

Sistla, A. P.; Vardi, M. Y.; and Wolper, P. 1987. The complementation problem for Büchi automata with appplications to temporal logic. *Theoretical Computer Science* 49:217–237.

Stockmeyer, L. J., and Chandra, A. K. 1979. Provably difficult combinatorial games. *SIAM Journal of Computing* 8(2):151–174.

Toumi, A.; Gutierrez, J.; and Wooldridge, M. 2015. A tool for the automated verification of Nash equilibria in concurrent games. In *International Colloquium on Theoretical Aspects of Computing (ICTAC)*, volume 9399 of *LNCS*, 583–594. Springer.

van der Hoek, W., and Wooldridge, M. 2003. Time, knowledge, and cooperation: Alternating-time temporal epistemic logic and its applications. *Studia Logica* 75(1):125–157.

van der Hoek, W.; Lomuscio, A.; and Wooldridge, M. 2006. On the complexity of practical ATL model checking. In *International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 201–208. ACM.

Walukiewicz, I. 2004. A landscape with games in the background. In *Annual Symposium on Logic in Computer Science (LICS)*, 356–366. IEEE Computer Society.

Wooldridge, M.; Gutierrez, J.; Harrenstein, P.; Marchioni, E.; Perelli, G.; and Toumi, A. 2016. Rational verification: From model checking to equilibrium checking. In *AAAI Conference on Artificial Intelligence (AAAI)*. AAAI Press.