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 system components 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 extensively studied, and the complexity of game theoretic decision problems relating to such games (such as the existence of Nash equilibria) have been comprehensively classified. In this article, we study Reactive Modules Games in which agents have only partial visibility of their environment.

Keywords: Games, Temporal Logic, Multi-Agent Systems, Formal Verification.

1. Introduction

A common technique in the formal analysis computer systems is to model a system as a game in which two players—sometimes called “System” and “Environment” or “Player” and “Opponent”—interact with each other, possibly over infinitely many rounds. In these games, it is typically 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 may represent either the behaviour of the computer system one wants to synthesize (an automated design problem [1]) or a particular system property which one wants to check (an automated verification problem [2]). 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 done based on this idea—see, *e.g.*, [3, 4] and the references therein for surveys containing several results on this topic.

Although this approach has been found to be appropriate in a wide range of settings, the zero-sum assumption is often either too restrictive or else simply inappropriate. For example, when dealing with concurrent systems one may have several system components, each with their own temporal goal, which are not necessarily in conflict.

The appropriate model here is a non-zero-sum n-player game, 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 that 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 [5]: a situation where no player wishes to deviate from the strategy it is currently using. While the use of zero-sum games in the analysis and design of computer systems is well-established, the approach of modelling computer systems as non-zero-sum games is much less so. Nevertheless, over the past decade, an increasing number of authors have begun to investigate this approach—see, for example, [6, 7, 8] for references.

In this article, 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 [9], a popular and expressive system modelling language that is used in several practical model checking systems (e.g., MOCHA [10] and Prism [11]). 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 Linear Temporal Logic (LTL) formula, representing a goal 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 [12], but in this paper we focus on *imperfect information* cases. We interpret imperfect information to mean that system components must make choices based on a partial view of the system state—more precisely, that there are some variables in the system whose value is hidden from them. Agents may have different views of the system: thus, one agent might have only a very minimal view of the system state while another is able to perceive the values of all variables in the system.

We study the decidability and complexity of checking the existence of Nash equilibria in Reactive Modules games with imperfect information. Using our framework, one can analyse the behaviour of open systems modelled as multi-player games using a modelling language that is widely used, and which already has a number of tool implementations. However, our results go beyond simply the specific language of SRML itself as, more generally, we provide complexity results that apply to a wide range of imperfect information games with succinct representations.

There are a number of good reasons to study imperfect information in this setting. Firstly, from a modelling point of view, it may not be realistic to expect that a simple module, which is supposed to represent some local, reactive, and independent behaviour, must be aware of the entirety of the state space of the systems it is part of. A more realistic situation is, for instance, that the module is aware only of the values of the Boolean variables associated with its linear temporal logic goal. Secondly, from a practical point of view, it is desirable to have specifications that are as small as possible so that the synthesis task may be simpler. Take, for instance, a system composed of thousands of Boolean variables. In a perfect information setting, every module will have to have a strategy that considers all possible valuations for those variables. Instead, if a particular module is allowed to have imperfect information, then a specification for such a module could be given so that the set of strategies associated with such a module will only have to take into account the values of a small subset of some of those Boolean variables, leading to potentially better results in practice. Thirdly, it may also be the case, as illustrated by Example 1 presented later on, that it is simply not the case that a setting with perfect information faithfully represents the situation we want to model. Finally, from a game-theoretic point of view, because the set of Nash equilibria of a system is sensitive to the visibility of players in a game, we can use imperfect information rather easily in the context of Reactive Modules Games to modify the set of Nash equilibria of a given game. This issue is discussed in more detail in Section 9.

The main results contained in the paper 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 Reactive Modules Game have.

Technically, the undecidability results in the paper rely on the undecidability of the *uniform synthesis problem* for LTL formulae [13]. In Section 4 we show how such a problem can be described as a Reactive Modules Game with three players in which one is interested in the existence of a Nash equilibrium in the game. On the positive side, our decidability results rely on the solution of different problems, showing the diversity of techniques that one could use to solve the particular instances at hand. For instance, for two-player games with imperfect information we propose a technique, to the best of our knowledge only previously explored in [14], where checking the existence of a Nash equilibrium is reduced to a number of CTL* synthesis problems. On the other hand, for the positive results given for memoryless and polynomially bounded strategies a simple nondeterministic algorithm can be used. The lower bounds, in NEXPTIME and PSPACE, respectively, are obtained via reductions from the satisfiability problem for Dependency Quantified Boolean Formulae (DQBF [15, pp. 86–87]) and the LTL model checking problem for compressed words [16]. Finally for the case considering myopic strategies we use the satisfiability problem for Quantified LTL (QPTL [17]) formulae.

Structure of the paper. Section 2 gives some background information and Section 3 introduces the model of games we consider here, namely Reactive Modules Games, and illustrate with an example the difference between perfect and imperfect information games. In Section 4 we show that checking the existence of Nash equilibria in imperfect information games with more than two players is an undecidable problem, even for the simple case of *iterated Boolean games* [18]. Section 5, contrarily, shows that in games with at most two players, such a problem is decidable and can be solved in 2EXPTIME.

In Sections 6, 7, and 8, we investigate three special cases: games with memoryless, bounded, and myopic strategies, respectively. In Sections 9 and 10 we study conditions to ensure the preservation of Nash Equilibria, whenever they exist, and the problems given by *rational verification* as proposed in [19]. Finally, in Section 11, we discuss relevant related work, draw a number of conclusions, and outline ideas for future work.

A note on this contribution. In this document, we provide more explanations of the material contained in [20], as well as the full proof details of all the results. Moreover, we include more comprehensive examples, as well as results on *iterated Boolean Games* [18] and other questions addressed within the *rational verification* framework [19].

2. Preliminaries

Logic. We work with logics that extend classical 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 the variable 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* [2] to model the dynamics of our systems. A Kripke structure K over Φ is given by a tuple $K = \langle S, S_0, R, \pi \rangle$, 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 state $s \in S$. 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 an infinite 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 $\mathit{runs}(K, s)$ be the set of s-runs of K, and let $\mathit{runs}(K)$ be the set of initial runs of K. Notice that a run $\rho \in \mathit{runs}(K)$ 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 $\boldsymbol{\rho}|_{\Psi}$, that is, we have $\boldsymbol{\rho}|_{\Psi}[t] = \boldsymbol{\rho}[t] \cap \Psi$ for each $t \in \mathbb{N}$.

Linear Temporal Logic. We make extensive use of Linear-Temporal Logic (LTL), an extension of classical propositional logic with two modal tense operators, $\mathbf{X}$ ("next") and $\mathbf{U}$ ("until") [21]. LTL is a standard language for expressing properties of infinite runs, such as the runs of Kripke structures. The syntax of LTL is defined with respect to a set Φ of Boolean variables by the following grammar:

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

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:

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

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.

3. Reactive Modules Games

We use Reactive Modules [9] as our basic language for modelling concurrent and multi-agent systems. The main purpose of Reactive Modules is to permit the specification of a collection of individual agents ("modules" in the terminology of Reactive Modules)—the choices that are available to them over time, and how these choices affect their shared environment. We use a natural imperfect information extension of the Simple Reactive Modules language, a subset of Reactive Modules introduced by [22] to study the complexity of practical ATL model checking.

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 $m_i = \langle \Phi_i, Vis_i, I_i, U_i \rangle$, 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 $\Phi_i \subseteq Vis_i$; 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 notations, since, by definition, $\Phi_i \subseteq Vis_i$, when we hereafter write $Vis_i = \Psi$ (where $\Psi \subseteq \Phi$), we mean $Vis_i = \Phi_i \cup \Psi$.

Moreover, 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 $i \in N$, we let $m_i = \langle \Phi_i, Vis_i, I_i, U_i \rangle$ be an SRMLI module over the set of Boolean variables Φ that defines the choices available to agent i. In addition, 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 $m_i = \langle \Phi_i, Vis_i, I_i, U_i \rangle$ 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 concern 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 $m_i = \langle \Phi_i, Vis_i, I_i, U_i \rangle$, denoted by $|m_i|$, is $|\Phi_i| + |Vis_i| + |I_i| + |U_i|$.

In what follows, we will use LTL characterisations of the runs of arenas A and modules m. The basic idea is that for every arena A, we can define an LTL formula $TH(A)$ (the theory of A), such that the runs satisfying $TH(A)$ are precisely the runs of A. Similarly, for every module m we can define an LTL formula $TH(m)$ such that the satisfying runs of $TH(m)$ are the possible runs of m. (In fact, $TH(A)$ is defined in terms of $TH(m)$ for component modules m of A). Moreover—and this is essential for many constructions—$TH(A)$ and $TH(m)$ are of size polynomial in $|A|$ and $|m|$ respectively. We will not give details of the constructions here: they are provided in detail in [12].

As an aside, we note that the formalism of Reactive Modules is related to Boolean games [18], and the underlying framework of propositional control [23]. A propositional control setting is defined by a collection of agents, a collection of Boolean variables, and an allocation of the variables to the agents. The basic idea is that agents have the power to choose the values for all the variables under their control. Boolean games extend such settings by assuming that each player has a goal, expressed as a logical formula over the set of variables; players then attempt to choose values for their variables so as to satisfy the formula. In [18], formulas were given in LTL, and the resulting framework is close to the setting of Reactive Module games. In fact, it is easy to see that propositional control settings can be modelled within Reactive Modules, but Reactive Modules in fact provide greater expressive power compared to propositional control settings. This is because, in propositional control settings, no restrictions are placed on the assignments that players can make to their variables—all possible allocations are permitted. In Reactive Modules, in any given state, the only assignments to variables that can be made are those corresponding to the enabled guarded commands. This makes it possible to more naturally model settings such as (for example) multi-agent planning [12].

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 [5, p. 201]: while it defines players and their choices, it does not specify the preferences of players. Preferences associated with a *goal* γ_i, which will be a logic 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 logic formula γ_i represents the goal that i aims to satisfy. For the moment, we will not restrict ourselves to any particular language for goals. Thus, each player i desires to act so as to satisfy its goal γ_i. We define the size of a game, $|G|$, by $|A| + |\gamma_1| + \cdots + |\gamma_n|$, where $|\gamma_i|$ denotes 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. We use a finite state representation of strategies; more precisely we model strategies as Moore machines (finite state machines with output). Given an SRMLI arena $A = \langle N, \Phi, m_1, \ldots, m_n \rangle$, a strategy for $m_i = \langle \Phi_i, Vis_i, I_i, U_i \rangle$ is a structure

$$\sigma_i = \langle Q_i, q_i^0, \delta_i, \tau_i \rangle,$$

where

- Q_i is a finite and non-empty set of *states*,
- $q_i^0 \in Q_i$ is the *initial* state,
- $\delta_i : Q_i \times 2^{Vis_i} \to 2^{Q_i} \setminus \emptyset$ is a *transition function*, and
- $\tau_i : Q_i \to 2^{\Phi_i}$ is an *output function*.

Since a strategy may not comply with a module's specification, we define consistency between a module and a strategy in the following way. We say that a strategy $\sigma_i = \langle Q_i, q_i^0, \delta_i, \tau_i \rangle$ is *consistent* with module $m_i = \langle \Phi_i, Vis_i, I_i, U_i \rangle$ if the following two conditions hold (a similar definition of consistency is also given in [12]):

1. $\tau_i(q_i^0) = eval_i(g_i^0, \emptyset)$, for some $g_i^0 \in I_i$; and,
2. for all $q, q' \in Q_i$, if $q' \in \delta_i(q, v)$ then $\tau(q') = (\tau_i(q) \setminus ctr(g_i)) \cup eval_i(g_i, v)$, for some $g_i \in U_i$ that is enabled by v, *i.e.*, such that $v \models guard(g_i)$,

where $eval_i : (I_i \cup U_i) \times 2^{Vis_i} \to 2^{\Phi_i}$ is a function that determines the value of the Boolean variables at the right-hand side of a guarded command when such a guarded command is enabled by a valuation. Formally, $eval_i$ is defined, for a guarded command $g_i = \varphi \rightsquigarrow x'_1 := \psi_1; \cdots ; x'_k := \psi_k$ and a (visible) valuation v, as follows: $eval_i(g_i, v) = \{x_j \in \{x_1, \ldots, x_k\} \ : \ v \models \psi_j\}$. Based on the definition of consistency given above, note, for instance, that if the only guarded update command of a module m_i has the form $\top \rightsquigarrow x' := \bot$, then a strategy for m_i will not prescribe m_i to set x to true under any contingency. Moreover, note also that if a module's visibility set does not contain some variable p, then no strategy for such a module will depend on the value of p. Let Σ_i be the set of consistent strategies for module m_i. Moreover, hereafter, by a strategy for module m_i we mean a strategy that is consistent with module m_i.

We now mention a construction that we find useful in what follows. Any finite state machine strategy σ can be represented by an SRML module (of polynomial size in $|\sigma|$) with variable set $\Phi_i \cup Q_i$ (that is, in addition to the variables Φ_i controlled by the player i, we introduce new variables, one for each of the states Q_i of the finite state machine. We write m_σ for such a module specification. The actual definition of m_σ is straightforward, and we leave this as an exercise for the reader.

Games are played by each player i 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 denoted 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 $[\![\vec{\sigma}]\!] = K_{A_{\vec{\sigma}}}|_{\Phi}$.

The basic assumption in our model is that 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 defined using 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 notation, we may simply write $\vec{\sigma} \models \gamma_i$.

If we only consider deterministic strategies, that is, those where $\delta_i : Q_i \times 2^{Vis_i} \to Q_i$, then it is easy to see that outcomes correspond to single runs, and we 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 [5]: 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.

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

Figure 1: Modules for the Casino example.

We now 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). This second point is formally studied in Section 9.

Note that the following example is intended to illustrate the concepts introduced so far rather than to constitute a real-life specification.

Example 1. *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 modules in Figure 1 and following 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 $Vis_{Casino} = \Phi = Vis_{Player}$. This model has two kinds of Nash equilibria: one where Player always wins (using the strategy above), 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 $Vis_{Player} = \{\mathsf{turn}\}$, then such "bad" equilibria disappear and only scenarios where both agents satisfy their goals remain as rational outcomes.*

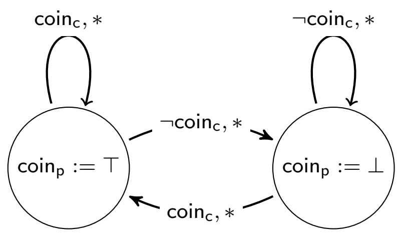

Figure 2: A winning strategy for *Player* if $Vis_{Player} = \Phi$. Symbol $*$ is $\neg\mathsf{turn}$. Edges if $\mathsf{turn} = \top$ are loops (for $\mathbf{skip}$).

4. 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 setting, the 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 results.

We will reduce the uniform distributed synthesis problem [24] 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 introduce some behaviour preserving transformations, in particular, one that deals with the preservation of LTL properties, which is presented next. More specifically, we need to deal with the fact that, in uniform distributed synthesis, the assignment of truth-values for variables is *asynchronous*, *i.e.*, the agents evaluate their variables in turn, while in SRMLI games is *synchronous*. Thus, in the next sections, we show some preliminary results that will be used to emulate the asynchrony of uniform distributed synthesis into our setting.

First, we introduce a notion of path inflation, in which an infinite sequence of variables evaluations is interleaved with intermediate states. Intuitively, such intermediate states are used to update the variables in the asynchronous process. After this, we provide a transformation of a generic LTL formula φ that is invariant under inflation, *i.e.*, the formula φ is satisfied by a run ρ if, and only if, its transformation is satisfied by the inflation of ρ. Finally, after introducing the uniform distributed synthesis problem, we show how to build an SRMLI game, whose associated strategy profiles produce exactly the inflated runs of the synthesis problem. We then derive a reduction from the uniform distributed synthesis problem to (the existence of Nash equilibria in) SRMLI games.

4.1. LTL formula transformation

Let us start this subsection by giving some useful definitions and notations. For $\rho : \mathbb{N}_o \to 2^{\Phi}$ a run and $d \geq 1$ an integer, we say that $\rho' : \mathbb{N}_o \to 2^{\Phi}$ is a *d-fold inflation* of ρ if $\rho'[d \cdot t] = \rho[t]$ for every $t \geq 0$. For a set Ψ of propositional variables with $\Phi \subseteq \Psi$, also say that a run $\rho' : \mathbb{N}_o \to 2^\Psi$ a *d-fold inflation* of ρ if $\rho'[d \cdot t] \cap \Phi = \rho[t]$ 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 $t = d \cdot t'$. 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 $\rho' : \mathbb{N}_o \to 2^\Psi$, we can define the d-fold deflation ρ over Φ to be the run $\rho : \mathbb{N}_o \to 2^\Phi$ which satisfies that $\rho[t] = \rho'[d \cdot t] \cap \Phi$ 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 $h : \{0, \ldots, n\} \to \Psi$, 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 $h^d[j] = h[j \cdot d] \cap \Psi$, 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.

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

Finally, to prove the Lemma 1, we use the standard semantics of LTL formulae on infinite runs [21], 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 $q \in \Phi'$, $\rho : \mathbb{N}_o \to 2^\Phi$ a run, $d \geq 1$, and $\rho' : \mathbb{N}_o \to 2^{\Phi \cup \Phi'}$ 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 $\rho' \models \tau_d(\varphi)$.*

Proof. We prove by structural induction on φ that, for all $t \geq 0$ and $\varphi \in \mathit{LTL}(\Phi)$, we have $\rho[t] \models \varphi$ if and only if $\rho'[d \cdot t] \models \tau_d(\varphi)$.

The basis, *i.e.*, if $\varphi = p$, is immediate by the definition of a d-fold inflation. For the induction step, the cases in which $\varphi = \neg\psi$ or $\varphi = \psi \vee \chi$, follow by an immediate application of the induction hypothesis. Then, it remains to prove the lemma for the temporal operator cases.

- Assume that $\varphi = \mathbf{X}\psi$ and so that $\rho[t] \models \mathbf{X}\psi$. Then, by the definition of semantics, we have that $\rho[t+1] \models \psi$. Now, by the induction hypothesis, we have that $\rho'[d \cdot (t+1)] \models \tau_d(\psi)$, which is equivalent to $\rho'[d \cdot t + d] \models \tau_d(\psi)$. Then, again by the definition of semantics, we have that $\rho'[d \cdot t] \models \mathbf{X}^d\tau_d(\psi)$, which, from the definition of τ_d, implies that $\rho'[d \cdot t] \models \tau_d(\mathbf{X}\psi)$
- Finally, assume that $\rho[t] \models \psi_1 \,\mathbf{U}\, \psi_2$. Then, by the definition of semantics, $\rho[t'] \models \psi_2$ and $\rho[t''] \models \psi_1$, for some $t' \geq t$ and for all $t \leq t'' < t'$. Which is equivalent, by

induction hypothesis, to the fact that $\rho'[d \cdot t'] \models \tau_d(\psi_2)$ and $\rho'[d \cdot t''] \models \tau_d(\psi_1)$. Now, remind that, by the definition of ρ', we have that $q \in \rho'[d \cdot t'']$, for all $t'' \leq t$ and $q \notin \rho'[h]$, if $h \neq d \cdot t''$, for all $t \leq t'' < t'$. This means that $\rho'[d \cdot t'] \models q \wedge \tau_d(\psi_1)$ and $\rho'[h] \models q \rightarrow \tau_d(\psi_2)$, for all $t \leq h < t'$, which, by definition of semantics, is equivalent to $\rho'[t] \models (q \rightarrow \tau_d(\psi_2)) \, \mathbf{U} \, (q \wedge \tau_d(\psi_1))$. Hence, we have that $\models \tau_d(\psi_2 \, \mathbf{U} \, \psi_1)$.

□

4.2. 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, $\mathrm{I}_p = \bigcup_{p' \in \mathrm{P}} \mathrm{O}_{(p', p)}$ denotes 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. To let the reader be familiar with architectures, we now present an example.

$$\begin{tikzcd}
 & \mathtt{p}_0 \arrow[dl, "a"'] \arrow[dr, "b"] & \\
\mathtt{p}_1 \arrow[dr, "c"'] & & \mathtt{p}_2 \arrow[dl, "d"] \\
 & \mathtt{p}_{idle} &
\end{tikzcd}$$

Figure 3: The architecture $\mathcal{A}_0$.

Example 2. *Consider the architecture represented in Figure 3, in which the process* $\mathtt{p}_0$ *controls the variables a and b, distributing them to processes* $\mathtt{p}_1$ *and* $\mathtt{p}_2$*, respectively. Moreover, processes* $\mathtt{p}_1$ *and* $\mathtt{p}_2$ *output variables c and d, respectively, to the idle process. Such an architecture is known in the literature as* $\mathcal{A}_0$ *and it has been used to show the undecidability of several problems related to that.*

A *strategy* for a process p is a function $\mathtt{s}_p : (2^{\mathrm{I}_p})^* \rightarrow 2^{\mathrm{O}_p}$ mapping each history of visible truth-assignments to a truth-assignment of the output variables. A *profile strategy* $\vec{\mathtt{s}}$ 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. The definition of the *Uniform Distributed Synthesis* problem follows.

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.*

In [24] it has been shown the undecidability of this problem. Formally, we have:

Theorem 1 ([24])**.** *The uniform distributed synthesis problem for a generic architecture $\mathcal{A}$ with three players and an LTL formula φ is undecidable.*

4.3. Undecidability

In this section, we show that the uniform distributed synthesis problem [24] 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 $x_j <_\Phi x_i$. Now, for a given process p, we define 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 have to keep track of the turn of which variable has to be set. 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 $\Gamma(\mathtt{s}) = \sigma : (2^{\mathit{Vis}(m_p)})^* \to 2^{\mathrm{Vr}(m_p)}$ such that, for any given variable $x_i \in \mathrm{O}(p)$ and history $h \in (2^{\mathit{Vis}(m_p)})^*$ 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 strategy profile 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 $\rho(\vec{\sigma}) = \rho(\Gamma^{-1}(\vec{\sigma}))^d$.*

Proof. We only show the proof for Item 1, as the proof of Item 2 is similar.

Let $\vec{\mathtt{s}}$ be a strategy profile and consider the corresponding $\vec{\sigma} = \Gamma(\vec{\mathtt{s}})$. Moreover, consider the runs $\rho_{\mathtt{s}} = \rho(\vec{\mathtt{s}})$ and $\rho_\sigma = \rho(\vec{\sigma})$. We need to prove show that, for all $t \in \mathbb{N}$, it holds that $\rho_{\mathtt{s}}[t] = \rho_\sigma[d \cdot t] \cap \Phi$. We proceed by induction on t. As base case, for $t = 0$, we have that $\rho_{\mathtt{s}}[0] = \emptyset = \{0\} \cap \varphi = \rho_\sigma[0] \cap \Phi$. As inductive case, assume that $\rho_{\mathtt{s}}[t'] = \rho_\sigma[d \cdot t'] \cap \Phi$, for all $t' \leq t$. Then, in particular, we have that $\rho_{\mathtt{s}}[0, t] = (\rho_\sigma[0, d \cdot t])^d$. At this point, let p be a process and m its corresponding module. By the construction of Γ, we know that, for all $x_i \in \mathrm{O}(p)$, we have that $x_i \in \rho_{\mathtt{s}}[t+1]$ iff $x_i \in \rho_\sigma[d \cdot t + i + 1]$ and, since m can only perform a **skip** up to $d \cdot (t+1)$, we have that $x_i \in \rho_{\mathtt{s}}[t+1]$ iff $x_i \in \rho_\sigma[d \cdot (t+1)]$. And since $x_i \in \Phi$, we obtain $x_i \in \rho_{\mathtt{s}}[t+1]$ iff $x_i \in \rho_\sigma[d \cdot (t+1)] \cap \Phi$. Finally, since this reasoning applies on all processes p and variables $x_i \in \mathrm{Vr}(p)$, we can conclude that $\rho_{\mathtt{s}}[t] = \rho_\sigma[d \cdot t] \cap \Phi$. $\square$

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 reason, 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 $G = \langle \mathrm{A}_{\mathcal{A}}', \gamma_{\mathtt{var}}, \gamma_0, \gamma_1, \ldots, \gamma_n, \gamma_A, \gamma_B \rangle$ *such that* $\mathrm{A}_{\mathcal{A}}'$ *is the arena derived by* $\mathcal{A}$*,*

- $\gamma_{\mathtt{var}} = \top$,
- $\gamma_0 = \neg \tau_d(\varphi)$,
- $\gamma_1 = \cdots = \gamma_n = \tau_d(\varphi)$,
- $\gamma_A = \tau_d(\varphi) \vee (a \leftrightarrow b)$ *and*
- $\gamma_B = \tau_d(\varphi) \vee \neg(a \leftrightarrow b)$.

Then, $\mathcal{A}$ realizes the LTL formula φ if and only if G has a Nash equilibrium.

Proof. We prove the theorem by double implication.

($\Rightarrow$) Assume $\mathcal{A}$ realizes φ. Then, there is a winning strategy $\vec{\mathtt{s}_{-0}}$ for $p_1, \ldots, p_n, m_A, m_B, \mathtt{var}$ such that $\rho(\mathtt{s}_0, \vec{\mathtt{s}_{-0}}) \models \varphi$, for all possible strategies $\mathtt{s}_0$ for the environment p_0. Consider the strategy $\Gamma(\vec{\mathtt{s}_{-0}})$ given for the modules $m_1, \ldots m_n, m_A, m_B, \mathtt{var}$, and consider a strategy σ_0 for module m_0. By Lemma 2, we have that

$$\rho(\Gamma^{-1}(\sigma_0), \vec{\mathtt{s}_{-0}}) = \rho(\sigma_0, \Gamma(\vec{\mathtt{s}_{-0}}))|_{\Phi} \models \tau_d(\varphi).$$

Then, by Lemma 1, we have that $\rho(\sigma_0, \Gamma(\vec{\mathtt{s}_{-0}})) \models \tau_d(\varphi)$. Moreover, the strategy profile $(\sigma_0, \Gamma(\vec{\mathtt{s}_{-0}}))$ 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_0 has a strategy $\sigma_0^{'}$ such that $\rho(\sigma_0^{'}, \Gamma(\vec{\mathtt{s}_{-0}})) \models \neg\tau_d(\varphi)$. Due to Lemmas 1 and 2, we have $\rho(\sigma_0^{'}, \Gamma(\vec{\mathtt{s}_{-0}})) \models \neg\tau_d(\varphi)$ and therefore $\vec{\mathtt{s}_{-0}}$ is not winning, which is a contradiction.

($\Leftarrow$) Let $(\sigma_0, \vec{\sigma_{-0}}) \in NE(G)$. Because of modules m_A and m_B, it must be the case that $\rho(\sigma_0, \vec{\sigma_{-0}}) \models \tau_d(\varphi)$. Then, consider the strategy profile $\Gamma^{-1}(\vec{\sigma_{-0}})$. We have that it is a winning strategy. Indeed, assume by contradiction that $\rho(\mathtt{s}_0, \Gamma^{-1}(\vec{\sigma_{-0}})) \models \neg\tau_d(\varphi)$ for some environment strategy $\mathtt{s}_0$. Then, by Lemmas 1 and 2, we obtain that $\rho(\Gamma(\mathtt{s}_0), \vec{\sigma_{-0}}) \models \neg\tau_d(\varphi)$ and so the strategy $\Gamma(\mathtt{s}_0)$ incentives module m_0 to deviate from the strategy profile $(\sigma_0, \vec{\sigma_{-0}})$, which is a contradiction.

□

Because the uniform distributed synthesis problem is undecidable with three processes, we can restrict ourselves to that setting, where m_0 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, in a later section, we will study the case where we still allow LTL goals, but restrict to systems with only two players. Before we do that, we will first show that our undecidability result holds for an even simpler model of games, namely, for iterated Boolean games [18].

4.4. Undecidability of Iterated Boolean Games

In this section, we show that the same undecidability result can be obtained for *Iterated Boolean Games* with imperfect information (iBGi), a special class of SRMLI games in which agents can assign values to the variable under control without any guard restriction. Formally, an iBGi is a tuple $H = (N, \Phi, \Phi_1, \dots, \Phi_n, \mathit{Vis}_1, \dots \mathit{Vis}_n, \gamma_1, \dots, \gamma_n)$ where, N is the (finite) set of players, $\Phi_i \subseteq \Phi$ is the set of variables under the control of player i, $\mathit{Vis}_i \subseteq \Phi$ is the set of variables that are visible to player i, and γ_i is the LTL objective of player i in the game. Since every player can set the truth-value of his own variables with no restrictions, at any round of the game, an iBGi can be represented as an SRMLI game in which every module $\mathtt{m}_i$ has exactly $|2^{\Phi_i}|$ guards, each of them having a tautology as a condition part and allowing one of the possible evaluations of the variables as an action part. However, the reader might note that the representation of an iBG in terms of SRMLI games involves an exponential blow-up.

As in SRMLI games, a strategy σ for a player in an iBGi is a finite state machine, with the difference that in this case it does not have to comply with any module's specification. However, this additional power of strategies can be constrained from a goal specification point of view. Indeed, given the propositional logic nature of guards, we can enforce the same level of restrictions in an iBGi by injecting an LTL encoding of a guard in players objectives. This allows us to show that the problem of finding a Nash equilibrium is undecidable also in iBGi*s*. To show this, consider the SRMLI game G defined in Theorem 2, and a corresponding iBGi $H = \langle N, \Phi, \Phi_{\mathtt{var}_d} = \{1, \dots d\}, \Phi_0, \Phi_1, \dots \Phi_2, \Phi_A = \{a\}, \Phi_B = \{b\}, , \gamma'_{\mathtt{var}_d} \gamma'_0, \gamma'_1, \dots, \gamma'_n, \gamma'_A, \gamma'_B\rangle$ having the same set of players, variables, and visibility as G. We need to adjust the LTL goals of the player to simulate the guards in G. For module $\mathtt{var}_d$, observe that the formula

$$\mathit{TH}(\mathtt{var}_d) = 1 \wedge \bigwedge_{\mathtt{h}=2}^{d} \neg \mathtt{h} \wedge \mathbf{G}((\bigwedge_{\mathtt{h}=1}^{d-1} \mathtt{h} \leftrightarrow \mathbf{X}(\mathtt{h}+1)) \wedge d \leftrightarrow \mathbf{X}1),$$

is true only if and only if player $\mathtt{var}_d$ in H follows the only strategy allowed by its module, that starts by setting the only variable 1 to be true and keeps updating the d-module counter round-by-round.

Analogously, for each module $\mathtt{m}_i$ in the game, the formula

$$\mathit{TH}(\mathtt{m}_i) = \bigwedge_{\mathtt{x} \neq 1; \mathtt{x} \in \Phi_i} \neg \mathtt{x} \wedge \mathbf{G}((\bigwedge_{\mathtt{h}=1}^{d} \mathtt{h} \rightarrow \bigwedge_{\mathtt{x}_j \neq \mathtt{x}_h; \mathtt{x}_j \in \Phi_i} \mathtt{x}_j \leftrightarrow \mathbf{X}\mathtt{x}_j),$$

is true if and only if player i in H follows a strategy that complies with the guards of $\mathtt{m}_i$.

Finally, the formula

$$TH(G) = TH(\mathtt{var}_d) \wedge \bigwedge_{i \in N} TH(\mathtt{m}_i)$$

is true if and only if every player follows a strategy that complies with the corresponding module in the SRMLI game G.

Now, we can define the goals of the players in H. We have the following:

- $\gamma'_{\mathtt{var}_d} = TH(\mathtt{var}_d)$;
- $\gamma_0 = \neg\tau_d(\varphi)$;
- $\gamma'_i = \tau_d(\varphi) \wedge TH(\mathtt{m}_i)$, for all $i \in \{1, \ldots, n\}$
- $\gamma'_A = (TH(G) \wedge \tau_d(\varphi)) \vee (a \leftrightarrow b)$;
- $\gamma'_A = (TH(G) \wedge \tau_d(\varphi)) \vee \neg(a \leftrightarrow b)$.

And the following theorem holds.

Theorem 3. *For every* SRMLI *game G defined in Theorem 2 and its corresponding iBGi H, it holds that*

$$NE(G) = \emptyset \quad \textit{if and only if} \quad NE(H) = \emptyset.$$

Proof. The proof proceed by double implication, showing that a strategy profile $\vec{\sigma}$ is a Nash equilibrium in G if and only if it is so in H and vice-versa. First, assume that $\vec{\sigma}$ is a Nash equilibrium in G. Observe that, due to the construction, it necessarily holds that $\vec{\sigma} \models \tau_d(\varphi)$. Moreover, $\vec{\sigma}$ is consistent with the guards in G. This means that $\vec{\sigma} \models TH(G)$. This implies that all the players but 0 get their goal achieved in H. Moreover, for what regards player 0, every strategy for 0 in H is also a strategy in G, meaning that a beneficial deviation in H would be so also in G, in contradiction with the fact that $\vec{\sigma}$ is a Nash equilibrium in G.

Now assume that $\vec{\sigma}$ is a Nash equilibrium in H. Then, it necessarily holds that $\vec{\sigma} \models TH(G) \wedge \tau_d(\varphi)$. Indeed, otherwise players A and B would be involved in a "matching pennies" game, which does not have any Nash equilibria. Thus, the strategy profile $\vec{\sigma}$ complies with all the module guards in G and then it is a possible strategy profile in G as well. Moreover, as $\vec{\sigma} \models \tau_d(\varphi)$, this means that every player but 0 achieves his goal in G. For what regards player 0, observe again that a beneficial deviation from $\vec{\sigma}$ in G would be so also in H, in contradiction with $\vec{\sigma}$ being a Nash equilibrium in H. $\square$

5. Decidability of Two-Player SRMLI Games

In [24], the authors identify in the notion of *information fork* the source of undecidability for the uniform distributed synthesis problem. More precisely, they show that restricting to architectures without information forks, the problem becomes decidable, and of complexity 2EXPTIME-complete. Also in logics for strategic reasoning, assuming that the information known by players is arranged in a *hierarchical* manner brings back the decidability of the model-checking problem [25, 26]. It is not clear yet if such decidability results transfer to the case of finding Nash equilibria in SRMLI 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 an intuitively simple logic-based approach: as we will show next, NONEMPTINESS for games with two players and LTL goals can be reduced to a series of temporal logic synthesis problems. In particular, this solution immediately implies the existence of 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.

It is important to observe that the solution techniques for the two-player case strongly rely on the synthesis problem, in which the agents goals are opposite. Therefore, it would not be possible to extend them to achieve results for decidable multi-agent settings, such as (we conjecture) the ones without information forks.

5.1. On the power of myopic strategies

Myopic strategies are strategies whose transition function do not depend on the values of the Boolean variables it reads, but only on the states where they are evaluated at. Formally, we have that a strategy $\sigma_i = (Q_i, q_i^0, \delta_i, \tau_i)$ is *myopic* if, for every $q \in Q_i$ and $\Psi, \Psi' \subseteq \mathit{Vis}_i$, we have $\delta_i(q, \Psi) = \delta_i(q, \Psi')$.

Intuitively, myopic strategies are "zero-knowledge" processes, as they are defined so that no dependency on the variables in the game is defined. This observation will be useful later in the paper, since myopic strategies will be used represent deviations of a player in a game. It is known that myopic strategies are powerful enough to describe any ω-regular run—an ultimately periodic run [27]. This is stated in the next lemma.

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 $\rho = \rho(\vec{\sigma}_{-i}, \sigma_i')$.*

Proof. Let $\rho(\vec{\sigma}) = \rho = \alpha\beta\beta\ldots$, such that α is a finite word of size k and $\beta^\omega = \beta\beta\ldots$ is an infinite word built from the concatenation of a finite word β of size p. Since ρ is a word over $(2^\Phi)^\omega$, it can be written as the superposition/union of n words over $(2^{\Phi_1} \times \ldots \times 2^{\Phi_n})^\omega$, that is, a word whose projection $\rho|_{\Phi_i}$ with respect to Φ_i is the ω-regular Φ_i-word

$$\begin{aligned} \rho|_{\Phi_i} \quad = \quad & \alpha_i[0], \alpha_i[1], \ldots, \alpha_i[k-1], \\ & \beta_i[0], \beta_i[1], \ldots, \beta_i[p-1], \beta_i[0], \beta_i[1], \ldots \end{aligned}$$

because of the definition of ρ, and where each α_i and β_i is, respectively, the restriction of α and β with respect to Φ_i.

We can then define the myopic machine strategy $\sigma_i = (Q_i, q_i^0, \delta_i, \tau_i)$, where

- $Q_i = \{q_i^s : 0 \leq s < k + p\}$,
- $(q_i^s, v, q_i^t) \in \delta_i$ if either
  - $s + 1 = t$ or
  - $s = k + p - 1$ and $t = k$,

- $\tau_i(q_i^s) = \rho[s]|_{\Phi_i}$.

Note that the correctness of the construction—that in fact σ_i is a myopic machine strategy—lies in the fact that there is a unique q_i^t for each q_i^s. As a consequence, the valuation in the transition function δ_i becomes irrelevant. It, then, immediately follows that $\rho = \rho(\vec{\sigma}) = \rho(\vec{\sigma}_{-i}, \sigma_i')$ for each 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 $m_i = \langle \Phi_i, Vis_i, I_i, U_i \rangle$ 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 $m_i' = \langle \Phi_i, {Vis_i}', I_i, U_i \rangle$, for every set $Vis_i \subseteq {Vis_i}' \subseteq \Phi$.*

Proof. Since by definition a myopic strategy is independent of the visibility set—that is, for every state q of the strategy and $\Psi, \Psi' \subseteq Vis_i$, we have $\delta_i(q, \Psi) = \delta_i(q, \Psi')$—it follows that σ_i is also a strategy of m_i' for every set $Vis_i \subseteq {Vis_i}' \subseteq \Phi$. We observe that since $Vis_i \subseteq {Vis_i}'$ the module m_i' is well defined. Note that the lemma may not hold for sets $\emptyset \subseteq {Vis_i}' \subset Vis_i$ as in such a case m_i may not be a well defined module. □

In particular, in this section, the following result, whose proof relies on Lemmas 3 and 4, is key to obtain the decidability result for two-player games presented later on.

Lemma 5. *Let G be an* SRMLI *game with two modules, $m_1 = \langle \Phi_1, Vis_1, I_1, U_1 \rangle$ and $m_2 = \langle \Phi_2, Vis_2, I_2, U_2 \rangle$, 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 $i \neq j$.*

Proof. The ($\Leftarrow$) direction is trivial. For the other direction, ($\Rightarrow$), suppose that σ_i is a winning strategy of player i for φ in G and, for a contradiction, that there is some strategy σ_j of player j in the game G' such that $\rho(\sigma_i, \sigma_j') \not\models \varphi$, that is, such that σ_i is not a winning strategy for φ in G'. Due to Lemma 3 we know that there is also a strategy profile (σ_i, σ_j'') such that both σ_j'' is myopic and $\rho(\sigma_i, \sigma_j'') \not\models \varphi$. And, because of Lemma 4 we also know that σ_j'' is a strategy of player j in G. But, if this was the case, then σ_i would not be a winning strategy in G; contradiction. Then, such a myopic strategy cannot exist and σ_i must be a winning strategy for φ in both G and G'. □

5.2. From synthesis to Nash equilibria

It is also known that 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

```
Nonemptiness($G_2$)
1.  if $coop$ is satisfiable then
        return “yes”
2.  if SYN($block_1$, $\Phi_2$, $Vis_2$, $\Phi_1$) and
        SYN($block_2$, $\Phi_1$, $Vis_1$, $\Phi_2$) then
        return “yes”
3.  if SYN($nodev_1$, $\Phi_2$, $Vis_2$, $\Phi_1$) or
        SYN($nodev_2$, $\Phi_1$, $Vis_1$, $\Phi_2$) then
        return “yes”
4.  return “no”
```

Figure 4: NONEMPTINESS in two-player games.

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 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 [21]: 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.”

Having the above in mind, consider the algorithm in Figure 4, 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

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

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. The following result can be shown:

Theorem 4. 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. □

Proof. Using the algorithm Nonemptiness(G_2), which in turn makes use of algorithms for LTL satisfiability as well as CTL* and LTL synthesis with imperfect information, we will show that if in a two-player SRMLI game there is a Nash equilibrium, then 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.

First, we will show that Nonemptiness(G_2) correctly checks the existence of a Nash equilibrium. We assume that G_2 has a Nash equilibrium and show that in such a case one of the three "if" cases of the algorithm must hold. We do so by analysing each case given in the algorithm, and therefore showing that the decision procedure is *sound*. After that, we will show that if a two-player SRMLI with LTL goals does not have a Nash equilibrium, then none of the three "if" cases of the algorithm holds—*i.e.*, that the decision procedure is *complete*. This part of the proof shows membership of NONEMPTINESS in 2EXPTIME. For hardness, we provide a reduction from LTL synthesis to a two-player perfect-information game with five Boolean variables.

We simplify reasoning by showing at the same time both that each case is an instance of a game with a Nash equilibrium and that if a Nash equilibrium exists, then one of the three cases will succeed, delivering the correct answer.

1. Case 1: trivial. In this case both players get their goal achieved and their behaviour is consistent with their own module's specification. Then, if *coop* is satisfiable there is a Nash equilibrium where both players achieve their goals. Because *coop* is an LTL formula, step 1 can be done in PSPACE using an algorithm for LTL satisfiability. Now, note that this case will always succeed when $[\![\gamma_1]\!] \cap [\![\gamma_2]\!] \neq \emptyset$. Therefore, if this step is not successfully executed we may assume that $[\![\gamma_1]\!] \cap [\![\gamma_2]\!] = \emptyset$ (as we do in the remaining cases).
2. Case 2: if both players have winning strategies for the negation of the other player's goal, by playing such strategies they can make sure that the other player does not get its goal achieved and, moreover, that it has no way to deviate in a beneficial way. Therefore, this case checks for the existence of a Nash equilibrium

where no player gets its goal achieved. Because both block_1 and block_2 are LTL formulae, step 2 can be done in 2EXPTIME using an algorithm for LTL synthesis with imperfect information. Observe that because of Lemma 5 we can assume that when checking $\mathit{block}_1/\mathit{block}_2$ only player 2/1 has imperfect information (the "verifier" in the associated synthesis game), whereas the other player (the "falsifier" in the synthesis game) has perfect information. Now, note that this case will not succeed if at least one of the players does not have a winning strategy for the negation of the goal of the other player. We can, therefore, make such an assumption in the remaining cases. Moreover, since we are assuming that the game has a Nash equilibrium, that at least one of the players does not have a winning strategy for the negation of the goal of the other player, and that $[\![\gamma_1]\!] \cap [\![\gamma_2]\!] = \emptyset$, then it follows that one of the players has a winning strategy for the negation of the goal of the other players, as otherwise a beneficial deviation would be possible. Thus, we also make this assumption hereafter.

3. Case 3: let player i be the player who has a winning strategy for the negation of the goal of the other player, that is, the player who wins the CTL* synthesis problem. For instance, player 1 in SYN(nodev_2, Φ_1, Vis_1, Φ_2); let player j be the other player. Since player i has a winning strategy, say σ_i, to synthesise nodev_j, player j cannot get its goal satisfied and has no beneficial deviation, provided that j plays consistently with its module's specification. However, player i may deviate if its own goal, γ_i, is not satisfied. However, since there is a run ρ, allowed by σ_i, such that $\rho \models \gamma_i \wedge \neg\gamma_j \wedge \mathit{TH}(m_i)$ while $\mathit{TH}(m_j)$ is also satisfied, then we know that there is a strategy σ_j of player j (which due to Lemma 5 can be chosen to be myopic) such that $\rho = \rho(\sigma_i, \sigma_j)$. Then, the strategy profile (σ_i, σ_j) is a Nash equilibrium where player i gets its goal achieved and player j does not. Also, because $\mathit{nodev}_1/\mathit{nodev}_2$ is a CTL* formula, step 3 can be done in 2EXPTIME using an algorithm for CTL* synthesis with imperfect information. By noting that this case also covers the case where a player has a winning strategy for its own goal, then we observe that no remaining possible cases can arise with respect to the existence of winning strategies, for each player, either for their own goals or for the negation of the goal of the other player. As a consequence, the only remaining cases will always admit deviations from at least one of the players, which contradicts the assumption of the existence of a Nash equilibrium in G_2.

Then, the algorithm Nonemptiness(G_2) is sound. To show that it is also complete, now let us show that if a game does not have a Nash equilibrium, then all of lines 1–3 in Nonemptiness(G_2) must fail, and hence that line 4 must be reached/executed.

A few observations first. Note that if a game does not have a Nash equilibrium then, necessarily, no run satisfying $\gamma_1 \wedge \gamma_2$, if any, is consistent with the specifications of the two modules. Then, line 1 will fail. Moreover, if there is no Nash equilibrium, at least one player must deviate from any strategy profile (σ_1, σ_2) and, in particular, due to Lemmas 3 and 4, it can do so using a myopic strategy. Therefore, if a player, say j, can deviate in a beneficial way is because, necessarily, the other player, say i, is not using a winning strategy for $\neg\gamma_j$—otherwise, such a beneficial deviation from j would not be possible so long its behaviour is consistent with its module's specification; hence, line 2 will fail too. Then, we are left with only one case, one where $[\![\gamma_1]\!] \cap [\![\gamma_2]\!] = \emptyset$

with at least one of the players not having its goal achieved, while the other player is not playing a winning strategy against the goal of the former player.

In other words, we have in this final case that for every strategy profile $\vec{\sigma} = (\sigma_1, \sigma_2)$, there is some $j \in \{1,2\}$ and some (myopic) σ'_j such that $\rho(\vec{\sigma}) \not\models \gamma_j$ and $\rho(\vec{\sigma}_{-j}, \sigma'_j) \models \gamma_j$, that is, such that the strategy $\sigma_i = \vec{\sigma}_{-j}$ is not winning for $\neg\gamma_j$. By contradiction, let us assume that line 3 of the algorithm does not fail, even though there is no Nash equilibrium in G_2. If line 3 does not fail there are three possibilities, namely that either both synthesis problems succeed, or only the first synthesis problem succeeds, or only the second synthesis problem succeeds. Let us analyse each case separately.

If both synthesis problems succeed then we know that each player has a winning strategy for the negation of the goal of the other player. But, in such a case, there would be a Nash equilibrium, which leads to a contradiction—in fact, in this case, line 2 would not fail. If only the first synthesis problem succeeds then we know that player 2 has a winning strategy for $\neg\gamma_1$. Let σ_2 be such a strategy. Moreover, we also know that for some strategy σ_1 of player 1, it is the case that $\rho(\sigma_1, \sigma_2) \models \gamma_2$. But, if this was possible then $\vec{\sigma} = (\sigma_1, \sigma_2)$ would be a Nash equilibrium (player 1 cannot beneficially deviate because σ_2 is a winning strategy for $\neg\gamma_1$ and player 2 will not deviate because $\rho(\vec{\sigma})$ satisfies γ_2), which leads to contradiction with the hypothesis. The remaining case is analogous to the former one. Then, if there is no Nash equilibrium, line 3 of the algorithm will necessarily fail too. Therefore, line 4 is necessarily executed, delivering the correct answer. Thus, the algorithm is complete.

Now, for hardness, one can reduce LTL synthesis with two variables, say x and y to a two-player perfect-information game with five Boolean variables. More specifically, we reduce the version of LTL synthesis/realisability where both players in the synthesis/realisability game play concurrently, as in our framework. (An explicit proof of the equivalence between these two problems is given, for instance, in [18].) The reduction takes as input an LTL synthesis game with two players, 1 and 2, and an LTL formula φ such that player 1 has a winning strategy in the synthesis game iff φ can be synthesised; otherwise, player 2 has a winning strategy to show that $\neg\varphi$ can be synthesised—because it is a zero-sum game. Based on this input game, the reduction uses the two modules below and following LTL goals:

- $\gamma_1 = (\neg\varphi \to (p \leftrightarrow q)) \wedge (\varphi \to r)$
- $\gamma_2 = (\neg\varphi \to \neg(p \leftrightarrow q)) \wedge (\varphi \to (\varphi \leftrightarrow \neg r))$

where φ is an LTL formula over Boolean variables x and y.

Given this two-player perfect-information reactive modules game, G, it is not hard to check that player 1 has a winning strategy to synthesise φ if and only if G has a Nash equilibrium, from which 2EXPTIME-hardness follows.

($\Rightarrow$) Assume that player 1 has a winning strategy to synthesise φ. Then, module m_1 can use such a winning strategy to ensure that φ holds, while setting r to $\top$. Then, γ_1 is satisfied and m_1 will not deviate. On the other hand, in this case, m_2 will not have its goal γ_2 satisfied, but cannot beneficially deviate since $(\varphi \to (\varphi \leftrightarrow \neg r))$ will still be false in every unilateral deviation of m_2.

```
module $m_1$ controls $\{x, p, r\}$ under $\Phi$
  init
  :: $\top \rightsquigarrow x := \bot;\ p := \bot;\ r := \bot$
  :: $\top \rightsquigarrow x := \bot;\ p := \bot;\ r := \top$
  :: $\top \rightsquigarrow x := \bot;\ p := \top;\ r := \bot$
  :: $\top \rightsquigarrow x := \bot;\ p := \top;\ r := \top$
  :: $\top \rightsquigarrow x := \top;\ p := \bot;\ r := \bot$
  :: ...
  update
  :: $\top \rightsquigarrow x := \bot;\ p := \bot;\ r := \bot$
  :: $\top \rightsquigarrow x := \bot;\ p := \bot;\ r := \top$
  :: $\top \rightsquigarrow x := \bot;\ p := \top;\ r := \bot$
  :: $\top \rightsquigarrow x := \bot;\ p := \top;\ r := \top$
  :: $\top \rightsquigarrow x := \bot;\ p := \bot;\ r := \bot$
  :: ...
module $m_2$ controls $\{y, q\}$ under $\Phi$
  init
  :: $\top \rightsquigarrow y := \bot;\ q := \bot$
  :: $\top \rightsquigarrow y := \bot;\ q := \top$
  :: ...
  update
  :: $\top \rightsquigarrow y := \bot;\ q := \bot$
  :: $\top \rightsquigarrow y := \bot;\ q := \top$
  :: ...
```

($\Leftarrow$) We prove the contrapositive: assume that player 1 does not have a winning strategy to synthesise φ and show that in such a case G does not have a Nash equilibrium. Since the game for LTL synthesis is determined and player 1 does not have a winning strategy to synthesise φ then we know that player 2 has a winning strategy for $\neg\varphi$. We will analyse all possible cases and show that in each instance at least one of the players has a beneficial deviation. Let $\vec{\sigma} = (\sigma_1, \sigma_2)$ be an arbitrary strategy profile and assume first that $\vec{\sigma} \models \neg\varphi$. Then, if $\vec{\sigma} \models (p \leftrightarrow q)$ player m_2 will have a beneficial deviation. If, on the other hand, $\vec{\sigma} \models \neg(p \leftrightarrow q)$ then player m_1 will have a beneficial deviation. Now, suppose that $\vec{\sigma} \models \varphi$. Then, if $\vec{\sigma} \models \neg r$ player m_1 will have a beneficial deviation. If, on the other hand, $\vec{\sigma} \models r$ then player m_2 will have a beneficial deviation: in case $\vec{\sigma} \models \neg(p \leftrightarrow q)$, player m_2 simply has to deviate to a strategy σ_2' that is winning for $\neg\varphi$ and keep the same value for q; in case $\vec{\sigma} \models (p \leftrightarrow q)$, player m_2 can deviate to a strategy σ_2' that is winning for $\neg\varphi$ and change the value of q with respect to the one given by σ_2. This analysis covers all possible cases *w.r.t.* φ, r, and p, q.

Then, the problem is 2EXPTIME-hard with two players, perfect information, and five Boolean variables. $\square$

5.3. From Nash to strong Nash equilibria

Even though Nash equilibrium is the best-known solution concept in non-cooperative game theory [5], 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 $\vec{\sigma} = (\sigma_1, \dots, \sigma_n)$, with $N = \{1, \dots, n\}$, 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. Strong Nash Equilibria are a refinement of Nash Equilibria, meaning that every sNE is also NE. Indeed, as also the singleton coalitions deviations correspond to individual deviations. The converse, instead, do not hold, as we can show by the following example.

Example 3. *Consider a perfect information iBG* [1] *given by four player* 1 *,* 2*,* 3*, and* 4*, controlling variables* p_1*,* p_2*,* p_3*, and* p_4*, respectively. Morover, the goals are* $\gamma_1 = \gamma_2 = (\mathrm{p}_1 \leftrightarrow \mathrm{p}_3) \wedge (\mathrm{p}_2 \leftrightarrow \mathrm{p}_4)$*, and* $\gamma_3 = \gamma_4 = (\neg\mathrm{p}_1 \leftrightarrow \mathrm{p}_3) \wedge (\neg\mathrm{p}_2 \leftrightarrow \mathrm{p}_4)$*. Such a game has a NE, that is, for example, the strategy profile assigning true to all the variables in the first round. Indeed, player* 1 *and* 2 *satisfy their goal, while player* 3 *cannot deviate to make the conjunct* $(\neg\mathrm{p}_2 \leftrightarrow \mathrm{p}_4)$ *to be true, and, symmetrically player* 4 *cannot deviate to make the conjunct* $(\neg\mathrm{p}_1 \leftrightarrow \mathrm{p}_3)$ *to be true. However, there is no sNE in this game, as in every strategy profile, either the coalition* $\{1, 2\}$ *or* $\{3, 4\}$ *has a beneficial deviation.*

The concept of sNE can be also used to filter the stable outcomes from undesired ones. For instance, we have the following example.

Example 4. *Consider a two-player game in which agent* 1 *controls* p*, agent* 2 *controls* q*, and they have the common objective* $\mathrm{p} \wedge \mathrm{q}$*. Clearly, the strategy profile assigning false to both* p *and* q *is a NE, as no player can beneficially deviate from it. However, this is not a sNE, as the two player can collaborate in order to achieve their goal.*

Let $sNE(G)$ be the set of strong Nash equilibria of G and let S-NONEMPTINESS be NONEMPTINESS, but with respect to $sNE(G)$. Then, we can prove the following.

Theorem 5. S-NONEMPTINESS *for two-player* SRMLI *games with LTL goals is 2EXPTIME-complete.*

Proof. A two-player SRMLI game G has a strong Nash equilibrium if and only if 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,

[1]The reader might note that perfect information games are special cases of imperfect information games, as well as iBGs are special cases of SRML games.

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. For hardness, one can use essentially the same reduction used in the Nash equilibrium case since in such a reactive modules game there is no strategy profile where both m_1 and m_2 get their goals achieved—making impossible a joint beneficial deviation, *i.e.*, all possible beneficial deviations are one-player deviations as in the Nash equilibrium case. $\square$

What we can learn from (the proof of) Theorem 5 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.

6. 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 say that a strategy $\sigma_i = (\mathrm{Q}_i, q_i^0, \delta_i, \tau_i)$ for a module $\mathtt{m}_i$ is *memoryless* if $\mathrm{Q} = 2^{Vis_i}$ and, for all $q \in \mathrm{Q}$ and $v \in Vis_i$, it holds that $\delta_i(q, v) = v$. As one might expect, since we restrict the set of strategies to the memoryless, the set of NE in the game might change. This is the case for a class of games we introduce in the following. For a given natural number $n \in \mathbb{N}$, consider the LTL formula for the *n-delayed matching pennies* game:

$$MP^n = (\mathtt{p}_2 \rightarrow (\mathbf{X}^n \mathtt{p}_1 \wedge \bigwedge_{1 \leq k < n} \mathbf{X}^k \neg \mathtt{p}_1)) \wedge (\neg \mathtt{p}_2 \rightarrow (\mathbf{X}^n \neg \mathtt{p}_1 \wedge \bigwedge_{1 \leq k < n} \mathbf{X}^k \mathtt{p}_1)).$$

Then consider a two-player game in which agent 1 controls $\mathtt{p}_1$ and player 2 controls $\mathtt{p}_2$. Moreover, $Vis_1 = \{\mathtt{p}_1\}$ and $Vis_2 = \{\mathtt{p}_2\}$ and the goals are $\gamma_1 = MP^2$ and $\gamma_2 = \neg MP^2$, respectively. It is easy to see that, in the general case, there is no NE, as in every outcome, the non satisfied player can beneficially deviate. Indeed, observe that, since player 1 cannot see the values of $\mathtt{p}_2$, he can only guess it at the first round and play accordingly. This gives player 2 the possibility to deviate from a non satisfying outcome. However, observe that, since the visibility set of players is given by the variables under their control, the only two possible memoryless strategies for player 1 are the one setting $\mathtt{p}_1$ constantly to true and the one setting $\mathtt{p}_1$ constantly to false, which can clearly never satisfy MP^2. Hence, in the memoryless strategies assumption, all the outcomes satisfy $\neg MP^2$ and there is no beneficial deviation for player 1, that is, every outcome is a NE. In the paper, however, we restrict to memoryless the strategies that players use in the strategy profile, but not the ones that a player may use to beneficially deviate.

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 5.

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

Figure 5: 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 5 relies on the following intermediate results.

Lemma 6. *For a game G 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 $|G|$, for every $\sigma_i \in \vec{\sigma}$.*

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, *i.e.*, each σ_i is also at most exponential in the size of G. $\square$

```
Memoryless Membership($G$, $\vec{\sigma}$)
1.  For all $i \in N$
2.      If $\vec{\sigma} \models \neg\gamma_i$ then
3.          For all $\sigma_i^*$
4.              If $(\vec{\sigma}_{-i}, \sigma_i^*) \models \gamma_i$ then
5.                  return "no"
6.  return "yes"
```

Figure 6: Algorithm to solve MEMBERSHIP in SRMLI games with Nash equilibria in memoryless strategies.

Lemma 6 is used to do step 1 of the algorithm in Figure 5. 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 $|\gamma_1| + \ldots + |\gamma_n|$.*

Proof. First of all, observe that checking whether $\vec{\sigma} \models \psi$, for some LTL formula ψ, can be done in time polynomial with respect to $|K_A|$, the Kripke structure induced by A, and exponential with respect to $|\psi|$. Moreover, recall that, for a fixed player i, the set of memoryless strategies is finite and every single strategy $\sigma_i : 2^{\Phi} \to 2^{\Phi_i}$ is of size exponential with respect to the set of variables in the game, *i.e.*, with respect to the size of A. Then, we can adapt the usual algorithm to check membership of a strategy profile in the set of Nash equilibria of a game as follows; see Figure 6 for the algorithm. The cycle on line 3 runs an LTL model-checking procedure for a number of times that is exponential in the size of the arena. Moreover, the cycle starting on line 1 bounds the number of executions of line 3 to be linear in the number of agents. In addition to this, the procedure on line 2 is again done using an LTL model-checking procedure, which is executed n times. Then, it follows that the overall complexity of the algorithm is exponential in both $|A|$ and $|\gamma_1| + \ldots + |\gamma_n|$. □

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

Theorem 6. NONEMPTINESS *for* SRMLI *games with memoryless Nash equilibria is NEXPTIME-complete.*

Proof. This NONEMPTINESS problem is solved using the non-deterministic algorithm in Figure 5. 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. The reduction is as follows. The DQBF satisfiability problem relates to a 3 player game, containing players B (Black) and W_1, W_2 (White 1, White 2). The white players form a team, attempting to beat the black player. The game is played on a Boolean formula φ over variables $X_1 \cup X_2 \cup Y_1 \cup Y_2$. Whereas the black player has perfect information, Player W_i has visibility only of variables $X_i \cup Y_i$. The game is played as follows:

- Player B chooses an assignment for the variables $X_1 \cup X_2$;
- Player W_1 chooses an assignment for variables Y_1;
- Player W_2 chooses an assignment for variables Y_2;
- If the overall assignment for $X_1 \cup X_2 \cup Y_1 \cup Y_2$ satisfies φ then black wins, otherwise team white wins.

The question is then whether there is a winning strategy for team white in this game. An important feature of this game, relevant to the reduction we want to produce, is that it is that if team white has a winning strategy, then it has a memoryless winning strategy which simply associates with each valuation for the variables in X_i a valuation for the variables in Y_i. Likewise, if team white does not have a winning strategy, the black player has a memoryless winning strategy which simply assigns a valuation for the variables in $X_1 \cup X_2$.

Our reduction produces an SRMLI game with 3 players, $N = \{B, W_1, W_2\}$; as might be guessed, each player corresponds to the player with the same name in the DQBF instance. We let $\Phi = X_1 \cup X_2 \cup Y_1 \cup Y_2 \cup \{p_1, p_2\} \cup Z$ where $\{p_1, p_2\}$ is a set of new Boolean variables as well as Z, which contains some variables that are used to define the behaviour of the modules in the SRMLI game. Players are modelled with the modules below, and goals are as follows:

- $\gamma_B = \mathbf{FG}\varphi$;
- $\gamma_{W_1} = (\mathbf{FG}\neg\varphi) \vee (p_1 \leftrightarrow p_2)$;
- $\gamma_{W_2} = (\mathbf{FG}\neg\varphi) \vee \neg(p_1 \leftrightarrow p_2)$.

```
module $B$ controls $X_1 \cup X_2 \cup Z_{X_1 \cup X_2} \cup \{\mathsf{done}_B\}$ under $\Phi$
  init
  $:: \top \rightsquigarrow x := \bot; \dots; z_x := \top; \dots; \mathsf{done}_B := \bot$
  update
  $:: z_x \rightsquigarrow x := \bot; z_x := \bot$
  $:: z_x \rightsquigarrow x := \top; z_x := \bot$
  $:: \dots$
  $:: \bigwedge_{x \in X_1 \cup X_2} \neg z_x \rightsquigarrow \mathsf{done}_B := \top$
```

such that $(Z_{X_1 \cup X_2} \cup \{\mathsf{done}_B\}) \subseteq Z$. Moreover, the construction for players in the white team is as follows.

```
module $W_i$ controls $Y_i \cup Z_i \cup \{p_i\}$ under $\Phi_i \cup X_i \cup \{\mathsf{done}_B\}$
  init
  $:: \top \rightsquigarrow y := \bot; \dots; z_y := \top; \dots; p_i := \bot$
  $:: \top \rightsquigarrow y := \bot; \dots; z_y := \top; \dots; p_i := \top$
  update
  $:: (\mathsf{done}_B \wedge z_y) \rightsquigarrow y := \bot; z_y := \bot$
  $:: (\mathsf{done}_B \wedge z_y) \rightsquigarrow y := \top; z_y := \bot$
  $:: \dots$
```

such that $Z_i \subseteq Z$.

Clearly, the size of the modules is linear in the size of the input DQBF game. In the above SRMLI system, module B starts, round by round, giving Boolean values to the variables it controls until a final valuation with respect to $X_1 \cup X_2$ is produced (in whose case done_B is set to true). After that, the white team can start giving values to the Boolean variables they control, under the visibility restrictions of the game. As with the black module, a final valuation with respect to the variables in $Y_1 \cup Y_2$ is reached. The overall valuation can no longer be modified by any player. Such a valuation, which either satisfies or not formula φ determines the outcome of the game. It is easy to see that any (winning) memoryless strategy for the DQBF game, which we already described, defines a memoryless strategy in our SRMLI game.

Given the above construction, it is not hard to check that the white team has a winning strategy in the input game iff the constructed SRMLI game has a Nash equilibrium.

($\Rightarrow$) Assume that team white has a winning strategy for $\neg\varphi$. Then, team white can use that information to built a winning strategy for $\mathbf{FG}\neg\varphi$, that is, a profile $(\sigma_{W_1}, \sigma_{W_2})$ such that the strategy profile $\vec{\sigma} = (\sigma_B, \sigma_{W_1}, \sigma_{W_2})$ satisfies both γ_{W_1} and γ_{W_2} for every strategy σ_B of the black player. Then, since γ_{W_1} and γ_{W_2} will be satisfied, no white player will have an incentive to deviate. On the other hand, in this case, the black player will not have its goal γ_B satisfied, but cannot beneficially deviate since team white is playing a winning strategy for $\mathbf{FG}\neg\varphi$.

($\Leftarrow$) We prove the contrapositive statement: assume that team white does not have a winning strategy for φ and show that in such a case the SRMLI game does not have a Nash equilibrium. We will analyse all possible cases and show that in each instance at least one of the players has a beneficial deviation. Let $\vec{\sigma} = (\sigma_B, \sigma_{W_1}, \sigma_{W_2})$ be an arbitrary strategy profile and assume first that $\vec{\sigma} \models \mathbf{FG}\varphi$. Then, exactly one of the two white players will not have its goal achieved and a beneficial deviation but changing the value assigned to the variable p_i that it controls. If, on the other hand, $\vec{\sigma} \models \mathbf{FG}\neg\varphi$, then the black player will not have its goal γ_B satisfied, but can beneficially deviate to a strategy σ'_B such that $(\vec{\sigma}_{-B}, \sigma'_B) \models \gamma_B$ since we know that team white does not have a winning strategy for $\mathbf{FG}\neg\varphi$. This analysis covers all possible cases *w.r.t.* φ and p, q.

Then, the game is NEXPTIME-hard with three players and memoryless strategies. Whether the game is NEXPTIME-hard even with two players is an open question. □

7. 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, *i.e.*, that the set Q_i is of size polynomial with respect to the size of the module $\mathtt{m}_i$. As in the case of memoryless strategies, also for polynomially bounded strategies, we have that the n-delayed matching pennies game can have more Nash equilibria for a sufficiently large n. Indeed, assuming $n \geq 2^{\mathtt{m}_1}$, we have that player 1 is not capable of keep track of the game from the beginning up to the n-th round, in which he has to switch the value of $\mathtt{p}_1$ accordingly. This makes every outcome generated by polynomially bounded strategies to satisfy $\neg MP^n$ and not having beneficial deviations for player 1, *i.e.*, to be a Nash equilibrium.

Under the polynomially bounded strategies assumption, we can use the algorithm in Figure 5 to show that these games can be solved in PSPACE. Indeed, we have:

Theorem 7. NONEMPTINESS *for* SRMLI *games G with strategies polynomially bounded by $|G|$ is PSPACE-complete.*

Proof. 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 NSPACE 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 [16]. We first recall that a compressed word

$$w = w_1^i \ldots w_{m-1}^j (w_m^k)^\omega$$

represents the "uncompressed" infinite word

$$w_1^1 \ldots w_1^i \ldots w_{m-1}^1 \ldots w_{m-1}^j w_m^1 \ldots w_m^k w_m^1 \ldots w_m^k \ldots$$

where w_y^x is the yth finite subword of w which is repeated x times. Then, an infinite word w is built using m finite subwords from which the mth subword is repeated infinitely often. We can assume that the "counters" $i, \ldots, j, k$ are given in binary and that the word w is over the alphabet of symbols $\{\bot, \top\}$. The LTL model checking problem for an LTL formula φ and a compressed word w asks whether $w \models \varphi$. Such a problem is PSPACE-complete. A reduction of this problem to the existence of Nash equilibria in an SRMLI game with polynomially bounded strategies is as follows.

We build a game where we define two modules, namely m_x and m_y for each w_y^x. The first module will implement a counter for x and the second module will implement the yth subword of w. We also define three additional modules, namely m_w, m_p, m_q, which will be used to build w and to ensure that a Nash equilibrium exists if, and only if, w satisfies φ in the input problem. The definition of such modules is as given below and the goals of the modules are as follows:

- $\gamma_{m_x} = \top$, for each module m_x with $x \in \{i, \ldots, j, k\}$;
- $\gamma_{m_y} = \top$, for each module m_y with $y \in \{1, \ldots, m\}$;
- $\gamma_w = \mathbf{X}\varphi$;
- $\gamma_p = \gamma_w \vee (p \leftrightarrow q)$;
- $\gamma_q = \gamma_w \vee \neg(p \leftrightarrow q)$.

```
module $m_p$ controls $\{p\}$ under $\Phi$
  init
  :: $\top \rightsquigarrow p := \bot$
  :: $\top \rightsquigarrow p := \top$
  update
  :: $\top \rightsquigarrow$ skip
module $m_q$ controls $\{q\}$ under $\Phi$
  init
  :: $\top \rightsquigarrow q := \bot$
  :: $\top \rightsquigarrow q := \top$
  update
  :: $\top \rightsquigarrow$ skip
module $m_w$ controls $\{b\}$ under $\Phi$
  init
  :: $\top \rightsquigarrow b := \bot$
  update
  :: $\mathsf{subw}_1 \rightsquigarrow b := b_1$
  $\vdots$
  :: $\mathsf{subw}_m \rightsquigarrow b := b_m$
```

```
**module** $m_x$ **controls** $\{x_{|x|}, \dots, x_1\}$ **under** $\Phi$
  **init**
  $:: \top \rightsquigarrow\ x_{|x|} := \bot;\ \dots;\ x_1 := \bot$
  **update**
  $:: \mathsf{subw}_y \wedge \neg\psi_x \wedge \neg x_t \wedge \bigwedge_{c<t} x_c \rightsquigarrow$
          $x_t := \top;\ x_{t-1} := \bot;\ x_{t-2} := \bot;\ \dots$
  $\vdots$
  $:: \mathsf{subw}_y \wedge \psi_x \rightsquigarrow\ x_{|x|} := \bot;\ \dots;\ x_1 := \bot$
```

where $t \in \{1, \dots, |x|\}$ and formula ψ_x is an abbreviation of the representation in binary of counter x using variables in $\{x_{|x|}, \dots, x_1\}$. For instance, if x is a counter to 4, then $\psi_x = x_3 \wedge \neg x_2 \wedge \neg x_1$ since the natural number 4 is written 100 in binary.

```
**module** $m_y$ **controls** $\{b_y, \mathsf{subw}_y, \mathsf{done}_y, \mathsf{bit}_{|w_y|}, \dots, \mathsf{bit}_1\}$ **under** $\Phi$
  **init**
  $:: \top \rightsquigarrow\ b_y := w_y^x[1];\ \mathsf{subw}_y = \mathbf{bool};\ \mathsf{done}_y = \bot;$
       $\mathsf{bit}_{|w_y|} := \bot;\ \dots;\ \mathsf{bit}_1 := \top$
  **update**
  $:: \mathsf{subw}_y \wedge \neg\psi_x \wedge \mathsf{bit}_1 := \top \rightsquigarrow$
          $b_y := w_y^x[1];\ \mathsf{bit}_1 := \bot;\ \mathsf{bit}_2 := \top$
  $\vdots$
  $:: \mathsf{subw}_y \wedge \neg\psi_x \wedge \mathsf{bit}_{w_y} := \top \rightsquigarrow$
          $b_y := w_y^x[|w_y|];\ \mathsf{bit}_{|w_y|} := \bot;\ \mathsf{bit}_1 := \top$
  $::\ \bigwedge_{c<y} \mathsf{done}_c \wedge \neg\mathsf{subw}_y \rightsquigarrow\ \mathsf{subw}_y = \top$
  $:: \psi_x \rightsquigarrow\ \mathsf{done}_y = \top;\ \mathsf{subw}_y = \bot$
```

where $w_y^x[l]$ is the lth symbol of the finite sequence w_y^x and $\mathbf{bool} \in \{\bot, \top\}$ is a Boolean value that depends on w. In particular, it is $\top$ for m_1 and $\bot$ for all other "subword" modules m_y. Also, the module m_y does not contain the last update rule since the subword generated by such a module must be repeated infinitely often, *i.e.*, such a module "is never done" which is modelled by the fact that once the Boolean variable subw_m becomes true, it remains true forever after.

The SRMLI system is designed so that each module m_y outputs, using b_y, the symbols, either $\bot$ or $\top$, in w_y^x and it does so x number of times, which are counted using m_x. Moreover, the module m_w outputs, using b, the symbol corresponding to the right subword at each time point, which is known to m_w via the variables subw_y. Indeed, the goal γ_w of m_w is over b. The final key observation is that in every round of the game up to one update rule of each module is enabled for execution, which can be used to define polynomially bounded strategies.

Since all components of the game are now in place, the remainder of the proof simply checks that $w \models \varphi$ if and only if the game has a Nash equilibrium. We prove the statement by checking that both directions hold.

($\Rightarrow$) We assume that $w \models \varphi$. Then, the only infinite run with respect to b, which we denote by ρ_w, induced by the game is $\rho_w = \bot w[1]w[2]\dots$. Therefore, since

$w \models \varphi$ we know that $\rho_w \models \mathbf{X}\varphi$ and that $\rho|_{\Phi \setminus \{p,q\}}$, the unique run restricted to $\Phi \setminus \{p, q\}$ and induced by the game also satisfies $\mathbf{X}\varphi$, that is, we also know that any of the four possible different runs ρ of the game satisfy that $\rho \models \gamma_w$. Then, all modules have their goals achieved and we know that every strategy profile $\vec{\sigma}$ such that $\rho = \rho(\vec{\sigma})$ is a Nash equilibrium, and more importantly must exist.

($\Leftarrow$) We prove the contrapositive, that is we assume $w \not\models \varphi$ and show that in such a case the game does not have a Nash equilibrium. Reasoning as in the ($\Rightarrow$) direction, we find that if $w \not\models \varphi$ then $\rho \not\models \gamma_w$, for every run of the game. Then, player m_w does not get its goal achieved. As a consequence, in every outcome of the game, either m_p or m_q does not get its goal achieved either and have a beneficial deviation; hence, the game does not have a Nash equilibrium.

Note that the problem is already PSPACE-hard for games with perfect information. □

8. 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:

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

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 [17], 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 [17] 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 8. NONEMPTINESS *for myopic* SRMLI *games is EXPSPACE-complete.*

Proof. Membership in EXPSPACE is proved as follows. Firstly, given a myopic SRMLI game $G = (A, \gamma_1, \dots, \gamma_n)$, 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:

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

such that $\exists \Phi_i$ stands for $\exists p_i^1, \dots, p_i^{|\Phi_i|}$, where Φ_i is the set of Boolean variables $\{p_i^1, \dots, p_i^{|\Phi_i|}\}$, and similarly for $\forall \Phi_i$. Such a formula is in Σ_2^{QPTL}. Then, by [17], 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 [17] ensures that models of such a formula have a game interpretation using myopic strategies.

Now, for hardness, we provide a reduction from the satisfiability problem of Σ_2^{QPTL} formulae. Let $\varphi = \exists x_1 \dots \exists x_n \forall y_1 \dots \forall y_m \psi$ be a Σ_2^{QPTL} formula with $\vec{x} = \{x_1, \dots, x_n\}$ being the existentially quantified variables and $\vec{y} = \{y_1, \dots, y_m\}$ being the universally quantified ones, and ψ being an LTL formula over those variables.

Then, consider the 4-player game G in which Player 1 controls $\vec{x}$, Player 2 controls $\vec{y}$, while Player 3 and Player 4 control two additional variables p and q, respectively. Moreover, the respective goals are as follows: $\gamma_1 = \psi$, $\gamma_2 = \neg\psi$, $\gamma_3 = \psi \vee (p \leftrightarrow q)$, and $\gamma_4 = \psi \vee (\neg p \leftrightarrow q)$. We now show by double implication that φ is satisfiable if and only if G admits a myopic Nash Equilibrium.

($\Rightarrow$) First, assume that φ is satisfiable. Then, there exists an infinite sequence $\rho_{\vec{x}}$ of evaluations on $\vec{x}$ satisfying $\forall y_1 \dots \forall y_m \psi$, *i.e.*, such that every induced history ρ satisfies ψ. Moreover, we can assume that $\rho_{\vec{x}}$ is generated by a Moore machine $\mathcal{A}_{\vec{x}}$. Observe that the output produced by such a machine does not depend on the evaluations of $\rho_{\vec{y}}$. Thus, $\mathcal{A}_{\vec{x}}$ is a myopic strategy for Player 1 in G. At this point, consider a myopic strategy profile $\vec{\sigma}$ in G such that $\sigma_1 = \mathcal{A}_{\vec{x}}$. It holds that $\vec{\sigma} \models \psi$. This proves that Player 1, 3, and 4 do not have any incentive to deviate from $\vec{\sigma}$. In addition to this, since Player 2 is not satisfied, assume by contradiction that a deviation σ_2^* is such that $(\vec{\sigma}_{-2}, \sigma_2^*) \models \neg\psi$. This implies that the same run ρ^* does not satisfy ψ and so $\mathcal{A}_{\vec{x}}$ does not satisfy $\forall y_1 \dots \forall y_m \psi$, which is a contradiction.

($\Leftarrow$) On the other hand, assume that G has a myopic Nash equilibrium $\vec{\sigma}$. Then, it holds that $\vec{\sigma} \models \psi$. Indeed, otherwise either Player 3 or Player 4 would deviate on the associated/internal "matching pennies" game played with variables p and q. Now, since $\vec{\sigma}$ is myopic, then in particular σ_1 is so. Then, the evaluation sequence $\rho_{\vec{x}}$ generated by σ_1 is such that $\rho_{\vec{x}} \models \forall y_1 \dots \forall y_m \psi$. Indeed, assume by contradiction that there exists $\rho_{\vec{y}}$ such that the joint run $\rho = (\rho_{\vec{x}}, \rho_{\vec{y}})$ is such that $\rho \models \neg\psi$. Then, the myopic strategy σ_2 generating $\rho_{\vec{y}}$ is a beneficial deviation from $\vec{\sigma}$ for Player 2, which contradicts the fact that $\vec{\sigma}$ is a Nash equilibrium. □

9. 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 5. *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 $\rho(\vec{\sigma}_{-j}, \sigma'_j) \models \gamma_j$.*

Proof. Since $\vec{\sigma} \notin NE(G)$, because of the definition of Nash equilibrium, we know that there is a player j which has a strategy σ^*_j such that $\rho(\vec{\sigma}_{-j}, \sigma^*_j) \models \gamma_j$, while $\rho(\vec{\sigma}) \not\models \gamma_j$. And because $\rho(\vec{\sigma}_{-j}, \sigma^*_j)$ is an ω-word, due to Lemma 3, we know that there is also a myopic strategy, say σ'_j such that $\rho(\vec{\sigma}_{-j}, \sigma^*_j) = \rho(\vec{\sigma}_{-j}, \sigma'_j)$, that is, with $\rho(\vec{\sigma}_{-j}, \sigma'_j) \models \gamma_j$. □

Lemma 8 shows that the power that a module m_j has to beneficially deviate from a strategy profile is not due to the knowledge that such a module has about the variables in the system, as such a deviation can always be turned into a myopic (zero-knowledge) strategy. On the other hand, the ability of players to correctly account for other players' deviations strongly relies on their visibility of the actions performed by deviating agents. For a deeper discussion about this latter property, we refer the interested reader to [14].

This result has an effect on the preservation of Nash Equilibria. Indeed, the lack of visibility, on one hand, does not affect the ability of a player to beneficially deviate, but, on the other hand, gives agents less power to successfully prevent beneficial deviations.

Formally, the next theorem about the preservation of Nash equilibria holds.

Theorem 9. *Let G and G' be two* SRMLI *games, with*

- $G = ((N, \Phi, m_1, \dots, m_n), \gamma_1, \dots, \gamma_n)$ *and*
- $G' = ((N, \Phi, m'_1, \dots, m'_n), \gamma_1, \dots, \gamma_n)$,

such that for each $i \in N$ we have $m_i = \langle \Phi_i, Vis_i, I_i, U_i \rangle$, and $m'_i = \langle \Phi_i, Vis_i', I_i, U_i \rangle$, and $Vis_i \subseteq Vis_i'$. 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. $\square$

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. NONEMPTINESS problem using Theorem 9 and letting each Vis_i be Φ, with $i \in N$, which is not possible. Alternatively, a counter-example to such an equality can be shown, *e.g.*, as the one given by Example 5 above.

10. Rational Verification

So far we have studied the decidability and complexity of NONEMPTINESS in various scenarios, including general SRMLI games, iterated Boolean games with imperfect information, and games with memoryless, myopic, and polynomially bounded strategies. These results give a very comprehensive picture of how hard is to solve the main problem related to *rational verification* [19], which is concerned with the game-theoretic analysis and verification of concurrent and multi-agent systems modelled as multi-player games.

More specifically, rational verification consists three problems: NONEMPTINESS, and the following two, which are formally stated for general SRMLI:

Given: SRMLI game G, LTL formula φ.
E-NASH: Does $\exists \vec{\sigma} \in NE(G).\ \boldsymbol{\rho}(\vec{\sigma}) \models \varphi$ hold?

Given: SRMLI game G, LTL formula φ.
A-NASH: Does $\forall \vec{\sigma} \in NE(G).\ \boldsymbol{\rho}(\vec{\sigma}) \models \varphi$ hold?

We now show that E-NASH can be reduced to NON-EMPTINESS, regardless of whether the question is asked for SRMLI games or iterated Boolean games with imperfect information. It should be noted that a reduction in the other direction is trivial, and already known for games with perfect information [18]: NON-EMPTINESS is E-NASH in case $\varphi = \top$. Such a reduction takes constant time, as it does the one presented next.

Theorem 10. *Let G be a game (either an iBGi or an* SRMLI *game) and φ be an LTL formula. There is a game H (an iterated Boolean game with imperfect information or an* SRMLI *game, correspondingly), of constant size in G, such that*

$$NE(H) \neq \emptyset \quad \textit{if and only if} \quad \exists \vec{\sigma} \in NE(G)\ .\ \boldsymbol{\rho}(\vec{\sigma}) \models \varphi$$

Proof. Since the statement, and proof, uniformly applies to both iterated Boolean games with imperfect information and SRMLI games, let us call G simply a game of an iterated Boolean game with imperfect information or an SRMLI game. Now, let H be the game G which in addition has two more agents, say $n+1$ and $n+2$, with goals $\gamma_{n+1} = \varphi \vee (p \leftrightarrow q)$ and $\gamma_{n+2} = \varphi \vee \neg(p \leftrightarrow q)$, where $\Phi_{n+1} = \{p\}$ and $\Phi_{n+2} = \{q\}$, for two fresh Boolean variables p and q, and who have perfect information. Also, for a given strategy profile $\vec{\sigma}$, say in G/H, let $\vec{\sigma}'$ be the strategy profile in H/G that can be constructed from $\vec{\sigma}$ as follows. Let every agent j in H/G, with $i = j$, use a strategy σ_j^* that plays consistently with $\sigma_i = (Q_i, q_i^0, \delta_i, \tau_i)$ in $\vec{\sigma}$. Formally, such a strategy $\sigma_j^* = (Q_j, q_j^0, \delta_j, \tau_j)$ is defined as follows: $Q_j = Q_i$; $q_j^0 = q_i^0$; $\tau_j = \tau_i$; and $\delta_j(q, v \cup v') = \delta_i(q, v)$, for every $v \in V(\Phi)$ and $v' \in V(\{p, q\})$. Moreover, let agents $n+1$ and $n+2$ use any available strategy, say σ_{n+1} and σ_{n+2}. Let $\vec{\sigma}_H$ be the strategy profile $(\sigma_1^*, \ldots, \sigma_n^*, \sigma_{n+1}, \sigma_{n+2})$. Informally, $\vec{\sigma}'$ adds/removes two agents to/from $\vec{\sigma}$. Now, we prove the statement by showing both implications.

For the right-to-left direction, suppose that, for some $\vec{\sigma}$ in G, we have $\vec{\sigma} \in NE(G)$ and $\boldsymbol{\rho}(\vec{\sigma}) \models \varphi$. Then,

$$\boldsymbol{\rho}(\vec{\sigma}') \models \gamma_{n+1}$$

and

$$\boldsymbol{\rho}(\vec{\sigma}') \models \gamma_{n+2}$$

for all strategies σ_{n+1} and σ_{n+2} in H. Obviously, if $\vec{\sigma} \in NE(G)$ then $\vec{\sigma}' \in NE(H)$ because no player can beneficially deviate and $n+1$ and $n+2$ will get their goals satisfied since, in particular, φ must be satisfied.

Now, for the left-to-right direction, suppose that $\vec{\sigma} \in NE(H)$, for some $\vec{\sigma}$ in H. Because of players $n+1$ and $n+2$, we know that

$$\boldsymbol{\rho}(\vec{\sigma}) \models \varphi$$

in H. Then, it follows that

$$\boldsymbol{\rho}(\vec{\sigma}') \models \varphi$$

in G. Agents who get their goal achieved in H will necessarily get their goal achieved in G. Moreover, agents who do not get their goal achieved in H do not have a beneficial deviation in G. If they did such a beneficial deviation would also be possible in H, which is impossible since $\vec{\sigma} \in NE(H)$. Therefore, $\vec{\sigma}' \in NE(G)$ and $\vec{\sigma}' \models \varphi$ in G, as required. Finally, that H is of constant size in the size of the game G immediately follows from the construction of H. $\square$

The other reduction, namely from A-NASH to NONEMPTINESS, goes via E-NASH. This reduction is simple, and relies on the fact that A-NASH and E-NASH are (logically) dual problems. Let G be a game and φ an LTL formula. To solve A-NASH with respect to G and φ, we can simply ask if E-NASH for G and $\neg\varphi$ can be answered negatively. If it does, then, for all $\vec{\sigma}$ in $NE(G)$ we have $\boldsymbol{\rho}(\vec{\sigma}) \models \varphi$, either because $NE(G)$ is empty or because every $\vec{\sigma}$ in $NE(G)$ satisfies φ, from which the claim follows.

Corollary 2. E-NASH *and* A-NASH *for* SRMLI *and iBGi are undecidable.*

11. Related Work, Conclusions, and Future Work

Imperfect information in logic-based games. There is a long tradition in theoretical computer science of using logic-based games to characterise complexity classes. For example, [28] introduced a class of games called "PEEK" which are closely related to Boolean satisfiability problems, for example showing that the existence of winning strategies in a game called PEEK-G_4, (which can be understood as a natural generalisation of Quantified Boolean Formulae), is EXPTIME-complete. Later [29] generalised some of these PEEK games to handle imperfect information, and established complexity and (un)decidability results for these games. A summary of our decidability and complexity results from this viewpoint is given 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 and Nash equilibrium in logics for strategic reasoning. Although logics for games have been studied since at least the 1980s [30], recent interest in the area was prompted by the development of Alternating-time Temporal Logic (ATL), a CTL-like language for reasoning about cooperative strategic ability [31]. Variations of ATL for imperfect information settings have subsequently been extensively studied [32, 33].

However, ATL provides no mechanism for naming strategies in the object language, which limits its usefulness for reasoning about equilibria. Strategy Logic (SL) was developed to rectify this omission: SL is closely related to ATL, but explicitly makes it possible to name strategies in the object language, and to bind these strategies to players [34, 35]. Using SL, for example, it is possible to express the existence of Nash equilibria for games in which players have LTL goals. SL with incomplete information was recently studied in [26] where a decidability result is given for games with a "hierarchical" structure of information. As expected, SL formulae expressing the existence of Nash equilibria in multi-player games do not possess such a hierarchical structure of information since, as shown in this paper, the problem is undecidable.

Another logic with enough power to reason about Nash equilibrium in the presence of imperfect information is ATL with strategy contexts [36]. The model checking problem for this logic is non-elementary for games with perfect information and becomes undecidable for games with imperfect information. However, in [36], a class of decidable games has been identified: imperfect information games where all players have a "uniform" amount of information. In other words, players in the game may have partial information about the system under consideration, but all players must be able to make the same observations on the system. Thus, as in the case of SL with imperfect information, it is not possible to reason about Nash equilibrium in all cases—as it should be the case—but only in situations where the semantic constraint above mention holds.

In terms of the specific model that we use here (where the state of the system is characterised via a set of Boolean variables, and imperfect information is defined by assuming agents have access to only some subset of these variables), the closest work we are aware of is by Van Der Hoek *et al.* [37]. They developed an extension to epistemic logic with Kripke semantics, in which epistemic accessibility relations were defined with respect to the variables that are visible to an agent. The same model was then used with a dynamic component, making it possible to characterise the effects of revealing or concealing variables [38].

Model/Module checking vs. equilibrium checking. Imperfect information is also studied with respect to the model and module checking problems. These two problems can be characterised by a two-player zero-sum game and as a consequence are decidable even with imperfect information. However, checking the existence of a Nash equilibrium is dramatically different if we consider three or more players instead of only two, as in a model/module checking problem. Extensions of ATL, and other logics to reason about strategic abilities in an imperfect information setting, with which module and model checking problems can be studied can be found in [13, 39, 40]. In these papers, a number of decidability results are provided. However, such decidability results are not about the existence of Nash equilibria nor about multi-player games with imperfect information in the general setting. For instance, the decidability result in [39], which appears to be a particular case of our decidability result, is for two-player games, namely, the case where one considers a zero-sum game and asks for the existence of a winning strategy for one of the two players taking part in a module/model checking game.

Imperfect information in multi-player games. A great deal of work has been done for two-player games with perfect information; see, *e.g.*, [8]. In most cases these are games on graphs where stochastic information is also allowed. However, results for multi-player games are more scarce. A recent study detailing the difference between two-player and multi-player games for games on graphs can be found in [41]. In this paper, the authors study the decidability of checking whether a coalition of players have a joint strategy to achieve some ω-regular goal. As in [26, 36, 13], it is shown that the problem is decidable only if a particular pattern of partial information holds between the players involved in the game, usually with some coalitions of players not having incomparable amounts of information in the game. These results give insights into how imperfect information affects the decidability of computing strategies in a multi-player game, but are some distance from giving a conclusive indications as to when checking the existence of a Nash equilibrium in a game can be decided, which is the main problem we studied in the present paper. Also, as mentioned above, some of these results hold in fact for games with infinite or randomized strategies, which are different models from the one we used here.

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 [42]. 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—we note that a great deal of techniques developed for verification, *e.g.*, model checking, have been usefully applied in similar settings. Other issues for future work on this direction include more work on mapping out the complexity landscape of our games (see Table 1), and on practical implementations; see, *e.g.*, [43, 44, 45].

Models preceding SRMLI *games.* Two similar models of games were studied before SRMLI games, namely, iterated Boolean games [18] and SRML games with perfect information [12]. In both cases, checking the existence of Nash equilibria is a 2EXPTIME-complete problem, as it is the case for two-player SRMLI games. In fact, the 2EXPTIME lower bound holds even for two-player zero-sum games with LTL goals, *i.e.*, even for LTL synthesis [46]. The work presented in this paper differs from [18] and [12] in a number of ways, not only in the fact that imperfect rather than perfect information is studied. Firstly, the (positive) results obtained in [18, 12] rely on reductions to the *rational synthesis* problem as studied in [47]. Contrarily, the (negative) results obtained in this paper rely on the *distributed synthesis* problem as studied in [24]. Secondly, the positive decidability results presented here for two-player games (with imperfect information) rely on CTL* *synthesis*, and not in the automata constructions used for SRML games with perfect information or for iterated Boolean games in [18, 12]. Thus, the results presented in [18, 12] and those reported here are technically quite different. Finally, in this paper, as we were interested in finding decidable cases when checking for the existence of Nash equilibria, we studied games with simpler models of strategies, namely, memoryless, myopic, and polynomially bounded. Instead, in [12], we focused on the model theory underlying reactive modules games and explored goals given as branching-time temporal specifications, in particular, expressing goals in CTL.

Acknowledgements. The authors gratefully acknowledge the financial support of the ERC Advanced Investigator Grant 291528 ("RACE") at Oxford.

References

[1] A. Pnueli, R. Rosner, On the Synthesis of an Asynchronous Reactive Module, in: ICALP, Vol. 372 of LNCS, Springer, 1989, pp. 652–671.

[2] E. Clarke, O. Grumberg, D. Peled, Model Checking, MIT Press, 2002.

[3] D. R. Ghica, Applications of Game Semantics: From Program Analysis to Hardware Synthesis, in: LICS, IEEE Computer Society, 2009, pp. 17–26.

[4] I. Walukiewicz, A Landscape with Games in the Background, in: LICS, IEEE Computer Society, 2004, pp. 356–366.

[5] M. Osborne, A. Rubinstein, A Course in Game Theory, MIT Press, 1994.

[6] P. Bouyer, R. Brenguier, N. Markey, M. Ummels, Nash Equilibria in Concurrent Games with Büchi Objectives, in: FSTTCS, Vol. 13 of LIPIcs, Schloss Dagstuhl, 2011, pp. 375–386.

[7] P. Bouyer, R. Brenguier, N. Markey, M. Ummels, Concurrent Games with Ordered Objectives, in: FOSSACS, Vol. 7213 of LNCS, Springer, 2012, pp. 301–315.

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

[9] R. Alur, T. A. Henzinger, Reactive Modules, Formal Methods in System Design 15 (1) (1999) 7–48.

[10] R. Alur, T. A. Henzinger, F. Y. C. Mang, S. Qadeer, S. K. Rajamani, S. Tasiran, MOCHA: modularity in model checking, in: CAV, Vol. 1427 of LNCS, Springer, 1998, pp. 521–525.

[11] M. Z. Kwiatkowska, G. Norman, D. Parker, PRISM 4.0: Verification of Probabilistic Real-Time Systems, in: CAV, Vol. 6806 of LNCS, Springer, 2011, pp. 585–591.

[12] J. Gutierrez, P. Harrenstein, M. Wooldridge, From model checking to equilibrium checking: Reactive modules for rational verification, Artificial Intelligence 248 (2017) 123–157.

[13] B. Finkbeiner, S. Schewe, Coordination logic, in: CSL, Vol. 6247 of LNCS, Springer, 2010, pp. 305–319.

[14] J. Gutierrez, P. Harrenstein, M. Wooldridge, Expresiveness and complexity results for strategic reasoning, in: CONCUR, Vol. 42 of LIPIcs, Schloss Dagstuhl, 2015, pp. 268–282.

[15] R. A. Hearn, E. D. Demaine, Games, Puzzles & Computation, A K Peters, 2009.

[16] N. Markey, P. Schnoebelen, Model Checking a Path, in: CONCUR, Vol. 2761 of LNCS, Springer, 2003, pp. 248–262.

[17] A. Sistla, M. Vardi, P. Wolper, The Complementation Problem for Büchi Automata with Applications to Temporal Logic, Theoretical Computer Science 49 (1987) 217–237.

[18] J. Gutierrez, P. Harrenstein, M. Wooldridge, Iterated Boolean Games, Information and Computation 242 (2015) 53–79.

[19] M. Wooldridge, J. Gutierrez, P. Harrenstein, E. Marchioni, G. Perelli, A. Toumi, Rational verification: From model checking to equilibrium checking, in: AAAI, AAAI Press, 2016, pp. 4184–4191.

[20] J. Gutierrez, G. Perelli, M. Wooldridge, Imperfect Information in Reactive Modules Games, in: KR, 2016, pp. 390–400.

[21] E. Emerson, Temporal and Modal Logic, in: Handbook of Theoretical Computer Science (Vol. B), MIT Press, 1990, pp. 995–1072.

[22] W. van der Hoek, A. Lomuscio, M. Wooldridge, On the complexity of practical ATL model checking, in: AAMAS, ACM, 2006, pp. 201–208.

[23] W. van der Hoek, M. Wooldridge, On the logic of cooperation and propositional control, Artificial Intelligence 64 (2005) 81–119.

[24] B. Finkbeiner, S. Schewe, Uniform Distributed Synthesis, in: LICS, IEEE Computer Society, 2005, pp. 321–330.

[25] R. Berthon, B. Maubert, A. Murano, Decidability results for ATL* with imperfect information and perfect recall, in: AAMAS, ACM, 2017, pp. 1250–1258.

[26] R. Berthon, B. Maubert, A. Murano, M. Y. Vardi, Strategy logic with imperfect information, in: LICS, IEEE, 2017, pp. 1–12, To appear.

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

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

[29] G. L. Peterson, J. H. Reif, S. Azhar, Lower bounds for multiplayer noncooperative games of incomplete information, Computers and Mathematics with Applications 41 (2001) 957–992.

[30] R. Parikh, The Logic of Games and its Applications, in: Topics in the Theory of Computation, Elsevier, 1985, pp. 111–139.

[31] R. Alur, T. Henzinger, O. Kupferman, Alternating-time Temporal Logic., Journal of the ACM 49 (5) (2002) 672–713.

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

[33] T. Ågotnes, V. Goranko, W. Jamroga, M. Wooldridge, Knowledge and Ability, in: Handbook of Epistemic Logic, College Publications, 2015, pp. 534–589.

[34] K. Chatterjee, T. Henzinger, N. Piterman, Strategy Logic., Information and Computation 208 (6) (2010) 677–693.

[35] F. Mogavero, A. Murano, G. Perelli, M. Vardi, Reasoning About Strategies: On the Model-Checking Problem., ACM Transactions on Computational Logic 15 (4) (2014) 34:1–47.

[36] F. Laroussinie, N. Markey, A. Sangnier, ATLsc with partial observation, in: GandALF, Vol. 193 of EPTCS, 2015, pp. 43–57.

[37] W. Van Der Hoek, N. Troquard, M. Wooldridge, Knowledge and control, in: AAMAS, ACM, 2011, pp. 719–726.

[38] W. van der Hoek, P. Iliev, M. Wooldridge, A logic of revelation and concealment, in: AAMAS, ACM, 2012, pp. 1115–1122.

[39] W. Jamroga, A. Murano, Module checking of strategic ability, in: AAMAS, ACM, 2015, pp. 227–235.

[40] N. Bulling, V. Goranko, W. Jamroga, Logics for reasoning about strategic abilities in multi-player games, in: Models of Strategic Reasoning - Logics, Games, and Communities, Vol. 8972 of LNCS, Springer, 2015, pp. 93–136.

[41] K. Chatterjee, L. Doyen, Games with a weak adversary, in: ICALP, Vol. 8573 of LNCS, Springer, 2014, pp. 110–121.

[42] T. Sandholm, Solving imperfect-information games, Science 347 (6218) (2015) 122–123.

[43] A. Lomuscio, H. Qu, F. Raimondi, MCMAS: an open-source model checker for the verification of multi-agent systems, STTT 19 (1) (2017) 9–30.

[44] P. Cermák, A. Lomuscio, F. Mogavero, A. Murano, MCMAS-SLK: A model checker for the verification of strategy logic specifications, in: CAV, Vol. 8559 of LNCS, Springer, 2014, pp. 525–532.

[45] A. Toumi, J. Gutierrez, M. Wooldridge, A tool for the automated verification of Nash equilibria in concurrent games, in: ICTAC, Vol. 9399 of LNCS, Springer, 2015, pp. 583–594.

[46] A. Pnueli, R. Rosner, On the Synthesis of a Reactive Module, in: POPL, ACM, 1989, pp. 179–190.

[47] D. Fisman, O. Kupferman, Y. Lustig, Rational synthesis, in: TACAS, Vol. 6015 of LNCS, Springer, 2010, pp. 190–204.