Iterated Games with LDL Goals over Finite Traces

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

ABSTRACT

Linear Dynamic Logic on finite traces ($\mathrm{LDL_F}$) is a powerful logic for reasoning about the behaviour of concurrent and multi-agent systems. In this paper, we investigate techniques for both the *characterisation* and *verification* of equilibria in multi-player games with goals/objectives expressed using logics based on $\mathrm{LDL_F}$. This study builds upon a generalisation of *Boolean games*, a logic-based game model of multi-agent systems where players have goals succinctly represented in a logical way. Because $\mathrm{LDL_F}$ goals are considered, in the setting we study—iterated Boolean games with goals over finite traces (iBG_F)—players' goals can be defined to be regular properties while achieved in a finite, but arbitrarily large, trace. In particular, using *alternating automata*, the paper investigates automata-theoretic approaches to the characterisation and verification of (pure strategy Nash) equilibria, shows that the set of Nash equilibria in games with $\mathrm{LDL_F}$ objectives is regular, and provides complexity results for the associated automata constructions.

Keywords

Boolean Games, Nash Equilibria, Automata, Temporal Logics

1. INTRODUCTION

Boolean games (BG [12]) are a logic-based model of multi-agent systems where each agent/player i is associated with a goal, represented as a *propositional logic* (PL) formula γ_i, and player i's main purpose is to ensure that γ_i is satisfied. The strategies and choices for each player i are defined with respect to a set of Boolean variables Φ_i, drawn from an overall set of variables Φ. Player i is assumed to have unique control over the variables in Φ_i, in that it can assign truth values to these variables in any way it chooses. Strategic concerns arise in Boolean games as the satisfaction of player i's goal γ_i can depend on the variables controlled by other players.

Iterated Boolean games (iBG [10]) generalise Boolean games by making players interact with each other for infinitely many rounds. As in the standard (one-shot or one-round) setting described above, there are n players each of whom uniquely controls a subset of Boolean variables and defines the achievement of a particular goal formula γ_i satisfied. In an iBG, however, players' goals γ_i are *Linear Temporal Logic* formulae (LTL [20]), rather than PL formulae, which are naturally interpreted over infinite sequences of valuations of the variables in Φ; thus, in iBGs, such infinite sequences of valuations represent the plays of these games.

Even though both iterated and conventional Boolean games are logic-based models of multi-agent systems, they capture players' goals—and therefore the desired behaviour of the underlying multi-agent systems—in two radically different ways: whereas Boolean games have PL goals (which are naturally evaluated over one-round games), iterated Boolean games have LTL goals (which are naturally evaluated over games with infinitely many rounds), encompassing the two extremes of the landscape when considering repeated games. However, there are games, systems, or situations where goals evaluated after an unbounded, but certainly finite, number of rounds should, or must, be considered.

In this paper we fill this gap and define and investigate *iterated Boolean games with goals over finite traces* (iBG_F), which are games where players' goals can be satisfied/achieved after a finite, but arbitrarily large, number of rounds. More specifically, the goals in these games are given by *Linear Dynamic Logic* formulae ($\mathrm{LDL_F}$) which are evaluated over finite sequences of valuations of the variables in Φ, that is, over *finite traces* of valuations, instead of PL or LTL formulae as in conventional or iterated Boolean games, respectively. Thus, while in a iBG_F a play still is an infinite trace of valuations, the satisfaction of a player's goal may occur after an unbounded but finite number of rounds. This sharply contrasts with the case of goals given by LTL formulae (*e.g.*, as in iBG), where it may be that a player's objective is satisfied only after considering the full infinite trace of valuations. This simple feature has significant implications, since rather complex automata constructions for the analysis of logics and games over infinite traces may become conceptually simpler under this new semantic (logic-based) framework. More importantly, this key observation allows one to define an automata model that exactly characterises the set of Nash equilibria in games with goals given by regular objectives.

There are several reasons to consider $\mathrm{LDL_F}$ goals. $\mathrm{LDL_F}$ offers great expressive power to our logic-based framework, which is indeed equivalent to monadic second-order logic (MSO). On the other hand, LTL interpreted on finite traces ($\mathrm{LTL_F}$) is as expressive as first-order logic (FOL) over finite traces [4]. This, in turn, implies that, over finite traces, while with $\mathrm{LTL_F}$ we can only describe star-free regular languages/properties, with $\mathrm{LDL_F}$ we can describe all regular languages/properties—that is, the properties and languages that can be described by regular expressions or finite state automata. Nevertheless, the automata-theoretic approach and complexity results for solving their related decision problems are equivalent, showing that the gain in expressiveness is achieved for free. In this paper, we first define Boolean games with $\mathrm{LDL_F}$ goals and then investigate its main game-theoretic properties using a new automata-theoretic approach to reason about Nash equilibria. Our technique to reason about equilibria builds on automata con-

Appears in: *Proceedings of the 16th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2017), S. Das, E. Durfee, K. Larson, M. Winikoff (eds.), May 8–12, 2017, São Paulo, Brazil.*

structions originally defined to reason about LDL_F formulae [4, 3]. Using this automata-theoretic technique we show a number of subsequent verification and characterisation results, as follows.

Firstly, we show that *checking* whether some strategy profile is a Nash equilibrium of a game is a PSPACE-complete problem, thus no harder than LDL_F satisfiability [4]. Secondly, we focus on the NE-NONEMPTINESS problem—which asks for the existence of a Nash equilibrium in a multi-player game succinctly specified by a set of Boolean variables and LDL_F formulae—and show that deciding whether an iBG_F has a Nash equilibrium can be solved in 2EXPTIME, thus no harder than solving LDL_F synthesis [3]. The automata technique we use for this problem also shows that the set of Nash equilibria in these games is ω-regular and can therefore be *characterised* using alternating automata. Thirdly, we also provide complexity results for the main decision problems related to the equilibrium analysis of these games with respect to extensions and restrictions of the initially studied iBG_F framework. In particular, we show that a small extension of the goal language, which we call *Quantified-Prefix Linear Dynamic Logic* (QPLDL_F), has the same automata-theoretic characteristics as LDL_F, and so it can be study using the same techniques. Moreover, LDL_F synthesis can be expressed in QPLDL_F, ensuring 2EXPTIME-completeness.

Regarding restrictions on the general framework, we first focus on the problem of reasoning with memoryless strategies. We show, using an automata construction, that the set of Nash equilibria for this games is also ω-regular. However, an alternative procedure for this problem, not based on automata, shows that improved complexity can be obtained when compared with the standard automata techniques to reason about LDL_F. Another restriction on strategies considered in the paper is the one of myopic strategies (which can be used to define all beneficial deviations in a game), in which players perform actions that are independent of the current state of the game execution. We show that games with such a restriction can be solved in EXPSPACE. We also consider the much more stable solution concept of *strong Nash equilibrium*, where sets of players in the game are allowed to jointly deviate, and provide an adaptation of the automata-based approach that retains the language characterisation and complexity properties of Nash equilibrium.

A key contribution of this work is that our automata-theoretic approach features two novel properties, within the same reasoning framework. Firstly, it shows that *checking* the existence of Nash equilibria can be reduced to a number of LDL_F synthesis and satisfiability problems—generalising ideas initially used to reason about LTL objectives [9]. Secondly, our automata constructions provide reductions where not only non-emptiness but also language equivalence is preserved. This additionally shows that the set of Nash equilibria in infinite games with regular goals is an ω-regular set, to the best of our knowledge, a semantic *characterisation* not previously known, and which do not immediately follows from other representations of Nash equilibria—see, *e.g.*, [2, 7, 13, 17, 14].

Motivation and Previous Work

While studying either iterated games or LDL_F is interesting in itself, from an AI perspective, our main motivation comes from applications to multi-agent systems. In particular, it has been shown that in many scenarios, for instance in the context of planning AI systems [4, 3], while logics like LTL, or even LTL over finite traces (LTL_F), can be used to reason about the behaviour of agents in such AI systems, these logics are not powerful enough to express in a satisfactory way the main features of agents in such a context. In order to illustrate the use of LDL_F, and motivate even further our work, we will present an example in the next section, where some of the goals either are not expressible in LTL or have a more intuitive specification in LDL_F than in LTL. Together with applications to planning AI systems (see [4, 3]), this is an example of another instance where one can see an advantage of iBG_F over iBGs.

Moreover, regarding previous work, while our model builds on iBGs, where goals are given by LTL formulae, there are at least two main differences with such work. Firstly, we study scenarios that consider memoryless and myopic strategies, for which results on iBGs have not been investigated. Secondly, and most importantly, the tools developed in this paper to obtain most of our complexity and characterisation results, are *technically* remarkably different from those used for iBGs, both in [9] and in [8]. To be more precise, for iBGs the main question is reduced to rational synthesis [7], whose solution goes via a parity automaton characterising formulae of an extension of Chatterjee *et al*'s Strategy Logic [2], which leads to an automata construction that can be further optimised if computing Nash equilibria is the only concern. Instead, in our case, we reduce the problem *directly* to a question of automata constructed in a different way. As a consequence, we provide a new set of automata constructions which do not rely on nor relate to those used in rational synthesis, *i.e.*, those used to solve iBGs. Our automata constructions are also different from those used by De Giacomo and Vardi in [4, 3, 5], as described next.

In [4, 3, 5], De Giacomo and Vardi study the satisfiability and synthesis problems for LDL_F, with and without imperfect information. Because of the (game-theoretic) nature of these two problems, their automata constructions deal with two-player zero-sum turn-based scenarios only. Instead, in our case, we deal with multi-player nonzero-sum concurrent scenarios. This difference leads to a completely different technical treatment/manipulation of the automata that can be initially constructed from LDL_F formulae. In fact, their automata constructions and ours are the same only up to the point where LDL_F formulae are translated into automata—that is, the very first step in a long chain of constructions. Moreover, since De Giacomo and Vardi study synthesis and satisfiability problems, whereas we study Nash equilibria, we are required to have a different technical treatment of the automata involved in the solution of the problems investigated in this paper.

2. FORMAL FRAMEWORK

In this paper, we consider Linear Dynamic Logic (LDL_F), a temporal logic introduced in [4] in order to reason about systems whose behaviour can be characterised by sets of finite traces, that is, finite sequences of valuation for the variables of the system.

The syntax of LDL_F is as follows:

$$\varphi := p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \langle\rho\rangle\varphi \mid [\rho]\varphi$$
$$\rho := \psi \mid \varphi? \mid \rho + \rho \mid \rho;\rho \mid \rho^*,$$

where p is an atomic proposition in Φ; ψ denotes a propositional formula over the atomic propositions in Φ; ρ denotes path expressions, which are regular expressions over propositional formulae ψ, with the addition of the test construct $\varphi?$ from propositional dynamic logic (PDL); and φ stands for LDL_F formulae built by applying Boolean connectives and the modal connectives. Tests are used to insert checks for satisfaction of additional LDL_F formulae.

The semantics of LDL_F formulae is as follows. LDL_F formulae are interpreted over finite traces of the form $\pi : \{0, \ldots, t\} \to 2^\Phi$ and an integer $i \in \{0, \ldots, t\}$ as follows:

- atomic propositions and Boolean connectives as usual;
- $\pi, i \models \langle\rho\rangle\varphi$ if there exists $j \in \{i, \ldots, t\}$ such that $(i, j) \in \mathcal{R}(\rho, \pi)$ and $\pi, j \models \varphi$;

- $\pi, i \models [\rho]\varphi$ if for all $j \in \{i, \ldots, t\}$, if $(i, j) \in \mathcal{R}(\rho, \pi)$ then $\pi, j \models \varphi$;

where $\mathcal{R}(\rho, \pi) \subseteq \mathbb{N} \times \mathbb{N}$ is recursively defined by

- $\mathcal{R}(\psi, \pi) = \{(i, i+1) : \pi(i) \models \psi\}$;
- $\mathcal{R}(\varphi?, \pi) = \{(i, i) : \pi, i \models \varphi\}$;
- $\mathcal{R}(\rho_1 + \rho_2, \pi) = \mathcal{R}(\rho_1, \pi) \cup \mathcal{R}(\rho_2, \pi)$;
- $\mathcal{R}(\rho_1; \rho_2, \pi) = \{(i, j) : \exists k \in \{i, \ldots, j\}.(i, k) \in \mathcal{R}(\rho_1, \pi) \wedge (k, j) \in \mathcal{R}(\rho_2, \pi)\}$;
- $\mathcal{R}(\rho^*, \pi) = \{(i, i)\} \cup \{(i, j) : \exists k \in \{i, \ldots, j\}.(i, k) \in \mathcal{R}(\rho, \pi) \wedge (k, j) \in \mathcal{R}(\rho^*, \pi)\}$.

We now introduce *iterated Boolean games with goals over finite traces* (iBG_F), which build upon the framework of iBGs [10]. In an iBG_F, players' goals are given by LDL_F formulae interpreted on infinite paths of valuations over a given set of Boolean variables.

An iBG_F is a tuple $\mathcal{G} = \langle \mathrm{N}, \Phi, \Phi_1, \ldots, \Phi_n, \gamma_1, \ldots, \gamma_n \rangle$, where $\mathrm{N} = \{1, \ldots, n\}$ is a set of players, Φ is a set of Boolean variables, partitioned into n sets $\Phi_1, \ldots, \Phi_n$, and the goals $\gamma_1, \ldots, \gamma_n$ of the game are LDL_F formulae over Φ. In an iBG_F each player i is assumed to control a set of propositional variables Φ_i, in the sense that player i has the power to set the values (true "$\top$" or false "$\bot$") of each of the variables in Φ_i. An *action* for player i is a possible valuation $\mathsf{v}_i \in 2^{\Phi_i}$. An *action vector* $\vec{\mathsf{v}} = \langle \mathsf{v}_1, \ldots \mathsf{v}_n \rangle$ is a collection of actions, one for each player in the game. Every action vector determines an overall valuation for the variables in $\Phi = \bigcup_{i=1}^{n} \Phi_i$ of the game. An iBG_F is played for an infinite number of rounds, as an iBG, but the goals of the game, which are LDL_F formulae, are interpreted on finite traces of such an infinite run. As a consequence, the satisfaction of a player's goal in the game must occur after a finite, yet arbitrarily large, number of rounds. Due to this, we need to define how an LDL_F formula is satisfied on a iBG_F. The most natural way to do so, also implicitly followed in [3], is to say that an infinite play π satisfies an LDL_F formula φ if and only if there exists $k \in \mathbb{N}$ such that the prefix up to k of π, denoted by $\pi_{<k}$, satisfies φ, *i.e.*, $\pi_{<k} \models \varphi$. We also write $\pi \models \varphi$ if there is $k \in \mathbb{N}$ such that $\pi_{<k} \models \varphi$.

Observe that this definition allows a formula and its negation to be satisfied on the same infinite play. Consider, for example, the LDL_F formula $\varphi = \langle \top^* \rangle p$, which is satisfied by all and only finite traces ending with a state labeled by p, and the infinite play $\pi = (\bar{p}p)^\omega$ which toggles the value of p infinitely often. Clearly, φ is satisfied on every prefix of π of even length, while $\neg\varphi$ is satisfied on every prefix of π of odd length. Thus, we obtain that $\pi \models \varphi$ and $\pi \models \neg\varphi$. This means that the notion of satisfaction given by $\pi \models \neg\varphi$ cannot be used in place of $\pi \not\models \varphi$, as they are not equivalent. We discuss this later in the paper, and show that a small extension of LDL_F, which allows one to quantify over finite plays in a game, can be used to write formulae ψ that equals to the non-satisfaction of φ, *i.e.*, where $\pi \models \psi$ if and only if $\pi \not\models \varphi$.

Observe that the set of models of an LDL_F formula φ is of the form $\alpha \cdot (2^\Phi)^\omega$, with α representing the set of finite traces satisfying φ. In [4] it has been proven that α can be represented by a regular expression. This implies, as shown later, that the set of infinite plays satisfying an LDL_F formula can be described in terms of an nondeterministic Büchi word automaton (NBW) built upon the nondeterministic finite word automaton (NFW) accepting α, in which accepting states are constructed so that they are sinks with a self-loop. This makes the expressive power of LDL_F be incomparable with that of LTL when considering infinite words—*i.e.*, infinite plays. Indeed, on the one hand, it is known that LTL cannot express the ω-regular expression $(p \cdot 2^\Phi)^* \cdot (2^\Phi)^\omega$ [16], and, on the other hand, LDL_F cannot express the LTL formula $\mathbf{GF}p$ ("always eventually" p), for which every NBW accepting the set of models cannot be of the form described above.

Now, in order to illustrate the concepts introduced so far, and further motivate the iBG_F framework, we present an example where the need for LDL_F goals plays an essential role.

EXAMPLE 1. *Consider a file-sharing network composed by a protocol manager and 2 clients who want to share* file_1 *of size* n_1 *packets and* file_2 *of size* n_2 *packets, respectively. The clients want to eventually download the other client's file, while the protocol manager wants this transfer of information to happen in a fair way. For instance, the manager wants client 1 to always upload in odd time-steps of the communication, while client 2 to always upload in the even time-steps of the communication protocol. Moreover, the download of a given file can be marked as completed only after the whole number of its packets has been uploaded by the other party.*

We can represent this protocol by means of a three agent game with $\mathrm{N} = \{0, 1, 2\}$ *in which* $\Phi_0 = \{\mathtt{d}_1, \mathtt{d}_2\}$, $\Phi_1 = \{\mathtt{u}_1\}$, *and* $\Phi_2 = \{\mathtt{u}_2\}$. *Variable* $\mathtt{u}_i$ *being true means that a single packet of* file_i *has been uploaded by agent i, while variable* $\mathtt{d}_i$ *being true means that the download of* file_i *has been completed. Regarding the goals of the agents, we have the following (*LDL_F*) formulae. The two clients* 1 *and* 2 *want to eventually download* file_2 *and* file_1, *respectively. Thus, we have* $\gamma_1 = \langle \top^* \rangle \mathtt{d}_2$ *and* $\gamma_2 = \langle \top^* \rangle \mathtt{d}_1$. *Regarding the goal* γ_0 *of the protocol manager, this has to include several requirements. First of all, it requires that client 1 always uploads in odd time-steps until the download of* file_1 *has been completed, while client 2 does the same on even time-steps. We can represent these properties with the following* LDL_F *formulae:* $\gamma_{upl_1} = \langle (\mathtt{u}_1; \top)^* \rangle \mathtt{d}_1$ *and* $\gamma_{upl_2} = \langle (\top; \mathtt{u}_2)^* \rangle \mathtt{d}_2$. *Note that client 1 has no requirement on the even time-steps, as neither client 2 on odd time-steps. The reader might notice that the properties* γ_{upl_1} *and* γ_{upl_2} *can be represented neither in* LTL *nor in* LTL_F, *which is the finite trace version of* LTL *[16]. In addition to this, the protocol manager is in charge of marking the files as completely downloaded at the right time of the execution. This means that variable* $\mathtt{d}_i$ *has to be set to true once the whole amount of packets of* file_i *has been uploaded and not before. We can specify this requirement with the following* LDL_F *formulae:* $\gamma_{com_1} = [((\neg \mathtt{u}_1)^*; \mathtt{u}_1)^{n_1}]\mathtt{d}_1$ *and* $\gamma_{com_2} = [((\neg \mathtt{u}_2)^*; \mathtt{u}_2)^{n_2}]\mathtt{d}_2$. *In order to avoid that the protocol manager wrongly marks the download of a file as completed, we also have the following requirements:* $\gamma_{incom_1} = \bigwedge_{n<n_1} \neg \langle ((\neg \mathtt{u}_1)^\star; \mathtt{u}_1; (\neg \mathtt{u}_1)^\star)^n \rangle \mathtt{d}_1$ *and* $\gamma_{incom_2} = \bigwedge_{n<n_2} \neg \langle ((\neg \mathtt{u}_2)^\star; \mathtt{u}_2; (\neg \mathtt{u}_2)^\star)^n \rangle \mathtt{d}_2$. *The goal of the protocol manager is therefore given by the conjunction of all these conditions:* $\gamma_0 = \gamma_{upl_1} \wedge \gamma_{upl_2} \wedge \gamma_{com_1} \wedge \gamma_{com_2} \wedge \gamma_{incom_1} \wedge \gamma_{incom_2}$. *To see that the system we have just described/designed has a stable behaviour, from a game-theoretic point of view, we need the concepts of strategies and Nash equilibria, which are introduced next. Then, we will review this example again later on.*

Strategies in iBG and iBG_F are modelled as *deterministic finite state machines*. Formally a deterministic finite state machine for player i is a tuple $\sigma_i = (\mathrm{S}_i, s_i^{\mathrm{o}}, \delta_i, \tau_i)$ where, S_i is a finite set of *internal states*, s_i^{o} is the *initial state*, $\delta_i : \mathrm{S}_i \times 2^\Phi \to \mathrm{S}_i$ is a *transition function*, and $\tau_i : \mathrm{S}_i \to 2^{\Phi_i}$ is the *action function*. By Str_i we denote the set of possible strategies for player i. A (total) *strategy profile* is a tuple $\vec{\sigma} = (\sigma_1, \ldots, \sigma_n)$ of strategies, one for each player. We also consider partial strategy profiles. For a given set of players $\mathrm{A} \subseteq \mathrm{N}$, we use the notation σ_A to denote a tuple of strategies, one for each player in A. Moreover, we use the notation

$\sigma_{-\mathrm{A}}$ to denote a tuple of strategies, one for each player in $\mathrm{N}\setminus\mathrm{A}$. We also use σ_i in place of $\sigma_{\{i\}}$ and $\vec{\sigma}_{-i}$ in place of $\vec{\sigma}_{\mathrm{N}\setminus\{i\}}$. Finally, for two strategy profiles $\vec{\sigma}$ and $\vec{\sigma}'$, by $(\vec{\sigma}_{\mathrm{A}}, \vec{\sigma}'_{-\mathrm{A}})$ we denote the strategy profile given by associating the strategies in $\vec{\sigma}$ to players in A and strategies in $\vec{\sigma}'$ to players in $\mathrm{N}\setminus\mathrm{A}$.

Since strategies are deterministic, each profile $\vec{\sigma}$ determines a unique play, denoted by $\pi(\vec{\sigma})$, which consists of an infinite sequence of valuations, one for each round of the game. Each player i has a preference relation over plays $\pi \in (2^{\Phi})^{\omega}$, which is determined by its goal γ_i. We say that π is preferred over π' by agent i, and write $\pi \succeq_i \pi'$, if and only if $\pi' \models \gamma_i$ implies that $\pi \models \gamma_i$. Using this notion of preference, one can introduce the concept of *Nash Equilibrium*. We say that $\vec{\sigma}$ is a Nash Equilibrium strategy profile if, for each agent i and a strategy $\sigma_i{}' \in \mathrm{Str}_i$, it holds that $\pi(\vec{\sigma}) \succeq_i \pi(\vec{\sigma}_{-i}, \sigma_i{}')$. In addition, by $\mathrm{NE}(\mathcal{G}) \subseteq \mathrm{Str}_1 \times \ldots \times \mathrm{Str}_n$ we denote the set of Nash Equilibria of the game $\mathcal{G}$.

EXAMPLE 2. *Consider again the system in Example 1. A possible strategy σ_1 for player 1 is a finite-state machine that sets variable $\mathtt{u}_1$ to true on odd rounds of the execution, while a strategy σ_2 for player 2 might set $\mathtt{u}_2$ to true on even rounds of the execution. In addition, a possible strategy for player 0, say σ_0, might be a finite-state machine that sets variable $\mathtt{d}_i$ to true only after $\mathtt{u}_i$ has been set to true exactly n_i times in the execution. Then, the strategy profile $\vec{\sigma} = (\sigma_0, \sigma_1, \sigma_2)$ will be such that the execution $\pi = \pi(\vec{\sigma})$ satisfies γ_0, γ_1, and γ_2, and therefore is a Nash equilibrium. Indeed, checking that a strategy profile is a Nash equilibrium of a game is one of the main concerns of this paper, as formalised next.*

Equilibrium Checking. We are interested in a number of questions related to the *equilibrium analysis* of iterated games [10, 25].

NE MEMBERSHIP. Given a game $\mathcal{G}$ and a strategy profile $\vec{\sigma}$:

Is it the case that $\vec{\sigma} \in \mathrm{NE}(\mathcal{G})$?

which asks if a strategy profile is a Nash equilibrium of a game. The second decision problem we are interested in is the following:

NE NON-EMPTINESS. Given a game $\mathcal{G}$:

Is it the case that $\mathrm{NE}(\mathcal{G}) \neq \emptyset$?

which asks if a given game has at least one Nash equilibrium.

Finally, we also consider two decision problems (which are the analogous of model checking in a game-theoretic multi-agent setting) that are formally stated as follows:

E/A-NASH. Given a game $\mathcal{G}$ and $\mathrm{LDL_F}$ formula φ:

Does $\pi(\vec{\sigma}) \models \varphi$ hold, for some/all $\vec{\sigma} \in \mathrm{NE}(\mathcal{G})$?

which asks if φ is satisfied by some/every Nash equilibrium of $\mathcal{G}$.

In the following sections, we study the above questions, in particular using an automata-theoretic approach.

3. AUTOMATA CHARACTERISATIONS

In order to address the NE MEMBERSHIP problem, we first provide some preliminary results on automata. An interested reader can find definitions and more details in [24].

Consider a NFW $\mathcal{A} = \langle \Sigma, \mathrm{S}, s_0, \varrho, \mathrm{F}\rangle$, recognizing a regular language $\mathcal{L}(\mathcal{A})$. Then consider the NBW $\mathcal{A}^{\infty} = \langle \Sigma, \mathrm{S}, s_0, \varrho', \mathrm{F}\rangle$, where, for all σ and s, we have that $\varrho'(\sigma, s) = \varrho(\sigma, s)$, if $s \notin \mathrm{F}$, and $\varrho'(\sigma, s) = \{s\}$, otherwise.

Intuitively, the automaton $\mathcal{A}^{\infty}$ accepts all and only the infinite words π^{∞} having a prefix π accepted by $\mathcal{A}$. This fact can be shown with the following theorem.

THEOREM 1. *Let $\mathcal{A}$ be a NFW. Then, for all $\pi^{\infty} \in \Sigma^{\omega}$, we have that $\pi^{\infty} \in \mathcal{L}(\mathcal{A}^{\infty})$ iff there exists $k \in \mathbb{N}$ such that $\pi = (\pi^{\infty})_{\leq k} \in \mathcal{L}(\mathcal{A})$.*

As a corollary, we obtain the following result.

COROLLARY 1. $\mathcal{L}(\mathcal{A}_{\varphi}^{\infty}) = \{\pi^{\infty} \in (2^{\Phi})^{\omega} : \pi^{\infty} \models \varphi\}$

We can now address NE MEMBERSHIP. We show that this problem is PSPACE-complete; for the membership argument, we employ an automata-based algorithm for checking membership. We first introduce, for a given (machine) strategy $\sigma_i = \langle \mathrm{S}_i, s_i^0, \delta_i, \tau_i\rangle$ for a player i, a corresponding DFW $\mathcal{A}(\sigma_i) = \langle \Sigma, \mathrm{Q}_i, q_i^0, \rho_i, \mathrm{F}_i\rangle$ where: $\Sigma = 2^{\Phi}$ is the alphabet set, $\mathrm{Q}_i = (\mathrm{S}_i \times 2^{\Phi}) \cup \{\mathsf{sink}\}$ is the state set, where $\mathsf{sink} \notin \mathrm{S}_i$ is a fresh state, $q_i^0 = (s_i^0, \emptyset)$ is the initial state, $\mathrm{F}_i = \mathrm{S}_i \times 2^{\Phi}$ is the final state set, and ρ_i is the transition relation such that, for all $(s, \mathsf{v}) \in \mathrm{S}_i \times 2^{\Phi}$ and $\mathsf{v}' \in \Sigma$,

- $\rho_i((s, \mathsf{v}), \mathsf{v}') = \begin{cases} \delta_i(s, \mathsf{v}), & \text{if } \tau_i(s) = \mathsf{v}'|_{\Phi_i} \\ \mathsf{sink}, & \text{otherwise} \end{cases}$, and
- $\rho_i(\mathsf{sink}, \mathsf{v}') = \mathsf{sink}$

Let $\mathcal{L}(\mathcal{A}(\sigma_i))$ denote the set of infinite words in $(2^{\Phi})^{\omega}$ accepted by $\mathcal{A}(\sigma_i)$. It is easy to see that such a set is exactly the same set of plays that are possible outcomes in a game where player i uses strategy σ_i. Similarly, for a given set of players $\mathrm{A} \subseteq \mathrm{N}$ and a partial strategy profile $\vec{\sigma}_{\mathrm{A}}$, we have that, for $\mathcal{A}(\vec{\sigma}_{\mathrm{A}}) \triangleq \bigotimes_{i\in\mathrm{A}} \mathcal{A}(\sigma_i)$, the product of these automata, the language $\mathcal{L}(\mathcal{A}(\vec{\sigma}_{\mathrm{A}}))$ contains exactly those infinite plays in a game where players in A play according to the strategies given in $\vec{\sigma}_{\mathrm{A}}$. Moreover, in [3] it is shown, for every $\mathrm{LDL_F}$ formula φ, how to build and check on-the-fly a NFW $\mathcal{A}_{\varphi} = \langle \mathrm{S}, 2^{\Phi}, \{s_0\}, \delta, \{s_f\}\rangle$, such that, for every finite trace $\pi \in (2^{\Phi})^{*}$, we have $\pi \models \varphi$ if and only if $\pi \in \mathcal{L}(\mathcal{A}_{\varphi})$, where by $\mathcal{L}(\mathcal{A}_{\varphi})$ we denote the language of finite words (that is, the language of finite traces over 2^{Φ}) accepted by the automaton $\mathcal{A}_{\varphi}$.

Algorithm 1: Intersection contruction.

Input: an $\mathrm{LDL_F}$ formula φ and an NBW $\mathcal{A} = \langle 2^{\Phi}, \mathrm{Q}, q_0, \varrho, \mathrm{F}\rangle$.
Output: NFW $\mathcal{A}_{\varphi}^{\infty} \times \mathcal{A} = \langle 2^{\Phi}, \mathrm{S}', \{s_0{}'\}, \mathrm{F}', \varrho'\rangle$.
$s_0{}' \leftarrow \{(\varphi, q_0, 1)\}$;
$\mathrm{F}' \leftarrow \{\emptyset\} \times \mathrm{Q} \times \{1\}$;
$\mathrm{S} \leftarrow \{s_0\} \cup \mathrm{F}'$;
$\varrho \leftarrow \{((\emptyset, q, 1), \Pi, (\emptyset, q', 2)) : \Pi \in 2^{\Phi} \wedge q' \in \varrho(q, \Pi)\}$
$\cup \{((\emptyset, q, 2), \Pi, (\emptyset, q', 2)) :$
$\Pi \in 2^{\Phi} \wedge q' \in \varrho(q, \Pi) \wedge q \in \mathrm{Q} \setminus \mathrm{F}'\}$
$\cup \{((\emptyset, q, 2), \Pi, (\emptyset, q', 1)) : \Pi \in 2^{\Phi} \wedge q' \in \varrho(q, \Pi) \wedge q \in \mathrm{F}'\}$
;
while *(S' or ϱ' change)* **do**
    for $s \in \mathrm{S}'$, $q \in \mathrm{Q}$ *and* $\Pi \in 2^{\Phi}$ **do**
        for $q' \subseteq CL(\varphi)$ *and* $q' \in \varrho(q, \Pi)$ **do**
            if $s' \models \bigwedge_{\psi\in q} \delta(\psi, \Pi)$ **then**
                if $q \in \mathrm{F}'$ **then**
                    $\mathrm{S}' \leftarrow \mathrm{S}' \cup \{(s, q, 2), (s', q', 1)\}$;
                    $\varrho' \leftarrow \varrho' \cup \{((s, q, 2), \Pi, (s', q', 1))\}$;
                else
                    $\mathrm{S}' \leftarrow \mathrm{S}' \cup \{(s', q', 1), (s', q', 2), (s, q, 2)\}$;
                    $\varrho' \leftarrow \varrho' \cup \{((s, q, 2), \Pi, (s', q', 1))\}$;

Such a construction makes use of a function δ simulating the transition relation of the corresponding alternating finite word automaton (AFW), which takes a subformula ψ of φ and a valuation of variables $\Pi \subseteq \Phi$, and recursively returns a combination of subformulas. A suitable modification of such an algorithm allows one to construct the NBW $\mathcal{A}^{\infty}_{\varphi}$. As a matter of fact, observe that the only final state s_f of the automaton $\mathcal{A}_{\varphi}$ built in [3] does not have any outgoing transition. Then, given the construction of $\mathcal{A}^{\infty}_{\varphi}$, we only need to add a loop to it, for every possible valuation.

However, it cannot be used as it is to obtain the PSPACE complexity for the NE MEMBERSHIP problem. Indeed, we need to combine the NBW $\mathcal{A}^{\infty}_{\varphi}$ with the automata $\mathcal{A}_{\vec{\sigma}}$ and $\mathcal{A}_{\vec{\sigma}_{-i}}$ provided by the NE-MEMBERSHIP problem instance. To do this, we need to adapt the construction in order to handle these products. Note that both $\mathcal{A}_{\vec{\sigma}}$ and $\mathcal{A}_{\vec{\sigma}_{-i}}$ can be considered as NBW. Thus, it is enough to deliver an algorithm building an automaton intersection between $\mathcal{A}^{\infty}_{\varphi}$ and a generic NBW $\mathcal{A}$. From the algorithm described above, and reported in Algorithm 1, we derive the following result.

THEOREM 2. *The construction and emptiness check of the automaton $\mathcal{A}^{\infty}_{\varphi} \otimes \mathcal{A}$, where $\mathcal{A}$ is a generic NBW, can be solved in polynomial space with respect to the size of the formula φ.*

With this theorem in place, one can show that Algorithm 2 runs in PSPACE and solves NE MEMBERSHIP for iBG$_F$.

Algorithm 2: Algorithm for NE Membership.

```
1 Input: a game G and a strategy profile σ⃗.
2 Output: "Yes" if σ⃗ ∈ NE(G); "No" otherwise.
3 for i ∈ N do
4     if L(A(σ⃗) ⊗ A^∞_{γ_i}) = ∅ then
5         if L(A(σ⃗_{-i}) ⊗ A^∞_{γ_i}) ≠ ∅ then
6             return "No"
7 return "Yes"
```

THEOREM 3. NE MEMBERSHIP *is PSPACE-complete.*

PROOF. To show that Algorithm 2 is correct, assume that the algorithm returns "Yes" on a given instance $(\mathcal{G}, \vec{\sigma})$. This means that it never executes line 6 in the for-cycle starting from line 3. This means that, for every agent i, either the check on line 4 or the check on line 5 is false. In case line 4 is false, then we have that $\mathcal{L}(\mathcal{A}(\vec{\sigma})) \cap \mathcal{L}(\mathcal{A}_{\gamma_i}) \neq \emptyset$, meaning that the play $\pi(\vec{\sigma})$ is such that $\pi(\vec{\sigma}) \models \gamma_i$. Thus, player i is satisfied in the context $\vec{\sigma}$ and so it does not have any incentive to deviate from it. On the other hand, if line 4 returns true but line 5 returns false, then we have that $\mathcal{L}(\mathcal{A}(\vec{\sigma}_{-i})) \cap \mathcal{L}(\mathcal{A}_{\gamma_i}) = \emptyset$, which means that the satisfaction of γ_i is incompatible with the partial strategy profile $\vec{\sigma}_{-i}$, no matter how player i behaves. This, in terms of strategies, implies that there is no beneficial deviation for player i to get its goal achieved. Hence, the strategy profile $\vec{\sigma}$ is a Nash equilibrium of the game.

On the other hand, assume $\vec{\sigma}$ is a Nash equilibrium. Then, no player i has an incentive to deviate. This can be the case for two reasons: either $\pi(\vec{\sigma}) \models \gamma_i$, or there is no strategy $\sigma_i^{'}$ such that $\pi(\vec{\sigma}_{-i}, \sigma_i^{'}) \models \gamma_i$. If the former, then we have that $\mathcal{L}(\mathcal{A}(\vec{\sigma})) \cap \mathcal{L}(\mathcal{A}_{\gamma_i}) \neq \emptyset$ and so the check on line 4 is false. If the latter, then it follows that $\mathcal{L}(\mathcal{A}(\vec{\sigma}_{-i})) \cap \mathcal{L}(\mathcal{A}_{\gamma_i}) = \emptyset$, making the condition on line 5 false. Since this reasoning holds for every player i, it can be concluded that Algorithm 2 ends without hitting line 6 and therefore returning "Yes", which concludes the proof of correctness.

Regarding the complexity of the algorithm, note that the checks in lines 4 and 5 involve a nonemptiness test of NFW built by means of Algoritm 1. This procedure is called n times, where n is the number of agents, to obtain a PSPACE upper bound. For hardness, we reduce from the satisfiability problem of LDL$_{\text{F}}$ formulae, which is known to be PSPACE-complete [4]. $\square$

Now, let us study NE NON-EMPTINESS. Again, we use an automata-theoretic approach. We show how, given a game $\mathcal{G}$, it is possible to construct an alternating automaton $\mathcal{A}_{\mathrm{NE}}(\mathcal{G})$ such that $\mathcal{A}_{\mathrm{NE}}(\mathcal{G})$ accepts precisely the set of plays that are generated by the Nash equilibria of $\mathcal{G}$. A distinguishing feature of our automata technique is that it is *language preserving*, that is, $\mathcal{A}_{\mathrm{NE}}(\mathcal{G})$ recognizes exactly the set of plays that are obtained by some Nash equilibrium in the game. Hereafter, we call *Nash runs* the elements in such a set of runs. This property of our construction is the key to show that the set of Nash runs is, in fact, ω-regular. Also, note that as we now have to find (and not simply check) a strategy profile, we cannot use the automata of the form $\mathcal{A}(\sigma_i)$ provided above, as there is no known strategy σ_i, for each player i, that can be used here.

Now, we proceed by recalling the characterisation of Nash equilibria provided in [9]. For a given game $\mathcal{G} = \langle \mathrm{N}, \Phi_1, \ldots, \Phi_n, \gamma_1, \ldots, \gamma_n \rangle$ and a designated player $j \in \mathrm{N}$, we say that $\vec{\sigma}_{-j}$ is a punishment profile against j if, for every strategy $\sigma_j^{'}$, it holds that $(\vec{\sigma}_{-j}, \sigma_j^{'}) \not\models \gamma_j$. In [9], it has been proven that $\vec{\sigma} \in \mathrm{NE}(\mathcal{G})$ if and only if there exists $\mathrm{W} \subseteq \mathrm{N}$ such that $\sigma \models \gamma_i$ for every $i \in \mathrm{W}$ and, for every $j \in \mathrm{L} = \mathrm{N} \setminus \mathrm{W}$, the profile $\vec{\sigma}_{-j}$ is a punishment strategy against j, that is, a winning strategy profile of the coalition of players N_{-j} for the negation of the goal of player j.

Thus, we can think of finding punishment strategies in terms of synthesizing a finite state machine controlling Φ_{-j}. To do this, we apply an automata-theoretic approach. First of all, we build the Rabin automaton (ARW) $\mathcal{A}_{\gamma_j}$ used to recognize the models of γ_j. Then, we dualize the automaton to obtain the ARW $\overline{\mathcal{A}_{\gamma_j}}$ recognizing the complement language, *i.e.*, the set of infinite words that do not satisfy γ_j. At this point, by means of Theorem 2 in [21], we build a nondeterministic Rabin automaton on trees (NRT) $\overline{\mathcal{A}^{'}_{\gamma_j}}$ that recognizes exactly those trees T that are obtained from an execution of a winning strategy of the coalition N_{-j} when the goal is to avoid the satisfaction of γ_j. Now, following Corollary 17 in [18], we can build a NRW $\overline{\mathcal{A}^{''}_{\gamma_j}}$ such that $\mathcal{L}(\overline{\mathcal{A}^{''}_{\gamma_j}}) = \{\pi \in (2^{\Phi})^{\omega} : \exists \mathrm{T} \in \mathcal{L}(\overline{\mathcal{A}^{'}_{\gamma_j}}).\ \pi \subseteq \mathrm{T}\}$, where by $\pi \subseteq \mathrm{T}$ we denote the following property: π is a branch of the tree T starting at its root.

Now, let us fix $\mathrm{W} \subseteq \mathrm{N}$ for a moment, and consider the product automaton $\mathcal{A}_{\overline{\mathrm{L}}} = \bigotimes_{j \in \mathrm{L}} \overline{\mathcal{A}^{''}_{\gamma_j}}$. By the semantics of the product operation we obtain that $\mathcal{A}_{\overline{\mathrm{L}}}$ accepts those paths that are generated by some punishment profile, for each $j \in \mathrm{L}$. Moreover, consider the automaton $\mathcal{A}_{\mathrm{W}} = \bigotimes_{i \in \mathrm{W}} \mathcal{A}_{\gamma_i}$, recognizing the paths that satisfy every γ_i, for $i \in \mathrm{W}$. Thus, we have that the product automaton $\mathcal{A}_{\mathrm{W}} \otimes \mathcal{A}_{\overline{\mathrm{L}}}$ accepts exactly those paths for which every γ_i, with $i \in \mathrm{W}$, is satisfied while, for each $j \in \mathrm{L}$ the coalition N_{-j} is using a punishment strategy against j. Now, in order to exploit the characterisation given in [9], we only need to quantify over $\mathrm{W} \subseteq \mathrm{N}$. This, in terms of automata, corresponds to the union operation. Then, we get the following automata characterisation:

$$\mathcal{A}_{\mathrm{NE}}(\mathcal{G}) = \bigoplus\nolimits_{\mathrm{W} \subseteq \mathrm{N}} (\mathcal{A}_{\mathrm{W}} \otimes \mathcal{A}_{\overline{\mathrm{L}}}).$$

THEOREM 4 (EXPRESSIVENESS). *For a game $\mathcal{G}$ with* LDL$_{\text{F}}$ *goals, the automaton $\mathcal{A}_{\mathrm{NE}}(\mathcal{G})$ recognizes the set of Nash runs of $\mathcal{G}$. Moreover, the set of Nash runs of $\mathcal{G}$ is ω-regular.*

PROOF. We prove both implications. From left to right, assume that $\pi \in \mathcal{L}(\mathcal{A}_{\mathrm{NE}}(\mathcal{G}))$. Then, there is $\mathrm{W} \subseteq \mathrm{N}$ such that $\pi \in \mathcal{L}(\mathcal{A}_{\mathrm{W}} \otimes \mathcal{A}_{\overline{\mathrm{L}}})$. Observe that, *w.l.o.g.* we can assume that π is an ultimately periodic play [23] and so that there exists a finite-state machine $\Delta_\pi = (\mathrm{Q}_\pi, q^{\mathrm{o}}_\pi, \delta_\pi, \tau_\pi)$, controlling all the variables in Φ, *i.e.*, $\tau_\pi : \mathrm{Q}_\pi \to 2^\Phi$, that generates π. Moreover, observe that, for each $j \in \mathrm{L}$, $\pi \in \mathcal{L}(\overline{\mathcal{A}''_{\gamma_j}})$ implies that there exists $\mathrm{T}_j \in \mathcal{L}(\overline{\mathcal{A}'_{\gamma_j}})$ such that $\pi \subseteq \mathrm{T}_j$. This implies that, for each $j \in \mathrm{L}$, there is a finite-state machine $\Delta_j = (\mathrm{Q}_j, q^{\mathrm{o}}_j, \delta_j, \tau_j)$, controlling all the variables but Φ_j, *i.e.*, $\tau_j : \mathrm{Q}_j \to 2^{\Phi_{-j}}$, that generates the branches of T_j, according to the output of variables in Φ_j, including π. Now, for each $i \in \mathrm{N}$, define the strategy $\sigma_i = (\mathrm{S}_i, s^{\mathrm{o}}_i, \delta_i, \tau_i)$ as follows:

- $\mathrm{S}_i = \mathrm{Q}_\pi \times \bigtimes_{j\in\mathrm{L}} \mathrm{Q}_j \times (\mathrm{L} \cup \{\top\})$ is the product of the state-space of Δ_π together with the state-space of each Δ_j, for each $j \in \mathrm{L}$, plus a flag component given by $\mathrm{L} \cup \{\top\}$;
- $s^{\mathrm{o}}_i = (q^{\mathrm{o}}_\pi, q^{\mathrm{o}}_{j_1}, \ldots, q^{\mathrm{o}}_{j_{|\mathrm{L}|}}, \top)$, collecting all the initial states of the finite state machines, Δ_π and Δ_j, for each $j \in \mathrm{L}$, flagged with the symbol $\top$;
- δ_i is defined as follows: for each $(q, q_{j_1}, \ldots, q_{|\mathrm{L}|}, \top)$ and $v \in 2^\Phi$, $\delta_i((q, q_{j_1}, \ldots, q_{|\mathrm{L}|}, \top), v) = (\delta_\pi(q, v), \delta_{j_1}(q_{j_1}, v), \ldots, \delta_{j_{|\mathrm{L}|}}(q_{|\mathrm{L}|}, v), \mathsf{flag})$, where $\mathsf{flat} = \top$ if $v = \tau_\pi(q)$ and $\mathsf{flat} = j$ if $v_{-j} = (\tau_\pi(q))_{-j}$ and $v_j \neq (\tau_\pi(q))_j$.
- $\tau_i((q, q_{j_1}, \ldots, q_{|\mathrm{L}|}, \top)) = (\tau_\pi(q))_i$ and $\tau_i((q, q_{j_1}, \ldots, q_{|\mathrm{L}|}, j)) = (\tau_j(q))_i$, for each $j \in \mathrm{L}$.

Intuitively, a strategy σ_i for player i runs in parallel the i-th component of the finite-state machine Δ_π together with the i-th components of the finite-state machines Δ_j that win against the deviating players in L. Note that, by construction, as long as nobody deviates, the outcome of every single Δ_j corresponds to the one of Δ_π. We have that the strategy profile $\vec{\sigma}$, given by the union of the strategies defined above, generates π, and, as soon as a unilateral deviation occurs from player $j \in \mathrm{L}$, the partial strategy profile $\vec{\sigma}_{-j}$ starts following the finite-state machine Δ_j, which is by definition winning against j. Thus, $\vec{\sigma}$ is a Nash equilibrium.

From right to left, assume that π is a Nash run and let $\vec{\sigma}$ be a Nash equilibrium such that $\pi(\vec{\sigma}) = \pi$. Moreover, let $\mathrm{W} = \{i \in \mathrm{N} : \pi \models \gamma_i\}$. We show that $\pi \in \mathcal{L}(\mathcal{A}_\mathrm{W} \otimes \mathcal{A}_{\overline{\mathrm{L}}})$. Since $\pi \models \gamma_i$, for each $i \in \mathrm{W}$, we have that $\pi \in \mathcal{L}(\mathcal{A}_\mathrm{W})$. Moreover, let $j \in \mathrm{L}$. It holds that j does not have a beneficial deviation from $\vec{\sigma}$ and so we have that $\vec{\sigma}_{-j}$ is a winning strategy against j. From the definition of $\overline{\mathcal{A}'_{\gamma_j}}$ we have that the tree-execution T_j generated by $\vec{\sigma}_{-j}$ is in $\mathcal{L}(\overline{\mathcal{A}'_{\gamma_j}})$. Now, since $\pi \subseteq \mathrm{T}_{-j}$, we have that $\pi \in \mathcal{L}(\overline{\mathcal{A}''_{\gamma_j}})$, for each $j \in \mathrm{L}$, implying that $\pi \in \mathcal{L}(\mathcal{A}_{\overline{\mathrm{L}}})$. Hence, we have that $\pi \in \mathcal{L}(\mathcal{A}_\mathrm{W}) \cap \mathcal{L}(\mathcal{A}_{\overline{\mathrm{L}}}) = \mathcal{L}(\mathcal{A}_\mathrm{W} \otimes \mathcal{A}_{\overline{\mathrm{L}}})$, as required. $\square$

Using Theorem 4 we can address the problem of deciding if a game admits a Nash equilibrium by checking $\mathcal{A}_{\mathrm{NE}}(\mathcal{G})$ for emptiness. Regarding the complexity of building $\mathcal{A}_{\mathrm{NE}}(\mathcal{G})$, observe that the construction of each automaton $\overline{\mathcal{A}'_{\gamma_j}}$, provided in [21], is of size doubly exponential with respect to $|\gamma_j|$. Moreover, all the other operations used to build $\mathcal{A}_{\mathrm{NE}}(\mathcal{G})$ involve union and intersection of Rabin automata, which can be performed in time polynomial in the size of the constituting components. This shows that $\mathcal{A}_{\mathrm{NE}}(\mathcal{G})$ is a nondeterministic Rabin automaton on words of size doubly exponential with respect to the game $\mathcal{G}$. Since checking emptiness of a NRW can be done in NLogSpace, we obtain the following result.

THEOREM 5. NE-NONEMPTINESS *with* $\mathrm{LDL_F}$ *goals can be solved in 2EXPTIME.*

Now, to show that E-NASH and A-NASH are in 2EXPTIME, we can also apply an automata-theoretic approach. Indeed, for the E-NASH case, consider a game $\mathcal{G}$ and an $\mathrm{LDL_F}$ formula φ. Then, the automaton $\mathcal{A}_\varphi \otimes \mathcal{A}_{\mathrm{NE}}(\mathcal{G})$ recognizes all the plays that both satisfy φ and are a Nash run. Thus, checking the E-NASH problem corresponds to checking the nonemptiness of such automaton. On the other hand, for the A-NASH problem, consider the automaton $\overline{\mathcal{A}_\varphi} \otimes \mathcal{A}_{\mathrm{NE}}(\mathcal{G})$. This product automaton recognizes all plays that do not satisfy the formula φ and are a Nash run. Thus, checking the A-NASH problem corresponds to checking the emptiness of such an automaton. The two constructions above show that both E-NASH and A-NASH can be solved in 2EXPTIME. Formally, combining the results above, we also obtain the following theorem:

THEOREM 6. E-NASH *and* A-NASH *with* $\mathrm{LDL_F}$ *goals can be solved in 2EXPTIME.*

4. EXTENSIONS AND RESTRICTIONS

We now study some extensions and restrictions on the problems studied in the previous section. As a first result, we show that an extension of the $\mathrm{LDL_F}$ language used to represent players' goals can be used to encode $\mathrm{LDL_F}$ synthesis, studied in [3], as a NE NON-EMPTINESS problem. Subsequently, we restrict to two classes of strategies, namely *memoryless* and *myopic* strategies. With respect to memoryless strategies, we show that our automata-based techniques can be used to show that the set of Nash runs for games of this kind is also ω-regular, as in the original problem. An EXPSPACE brute-force approach can be used to show that the induced automata are suboptimal from a complexity point of view.[1] However, the construction is still based on a simple extension of automata on finite words, making it potentially useful in practice. The case of myopic strategies, instead, is studied using a reduction to the satisfiability problem for the 1-alternation fragment of QPTL, known to be solvable in EXPSPACE [22].

Games with Quantified prefix $\mathrm{LDL_F}$ *Goals.*

The results obtained so far show that checking whether a game has a Nash equilibrium can be solved in 2EXPTIME. We now show that a small extension of the logic $\mathrm{LDL_F}$, which we call *quantified prefix* $\mathrm{LDL_F}$ ($\mathrm{QPLDL_F}$) can also be solved using the same automata-theoretic technique, with the same complexity, and can be used to represent the $\mathrm{LDL_F}$ synthesis problem, which is 2EXPTIME-complete. Then, NE NON-EMPTINESS with respect to such an extension is 2EXPTIME-complete.

Syntactically, a $\mathrm{QPLDL_F}$ formula φ is obtained from an $\mathrm{LDL_F}$ formula ψ by simply adding either an existential $\exists$ or a universal $\forall$ quantifier in front of it, *i.e.*, $\varphi = \exists\psi$ or $\varphi = \forall\psi$. Such a quantification ranges over the set of prefixes of a given infinite path of valuations. Formally, we have that, for a given $\mathrm{QPLDL_F}$ formula of the form $\exists\psi$ and an infinite path π, we have that $\pi \models \exists\psi$ if there is $k \in \mathbb{N}$ such that $\pi_{<k} \models \psi$. Analogously, for a $\mathrm{QPLDL_F}$ formula of the form $\forall\psi$, we have $\pi \models \forall\psi$ if $\pi_{<k} \models \psi$, for all $k \in \mathbb{N}$.

The reader might note that $\exists\psi$ is equivalent to ψ on infinite plays. This means that the set of models for $\exists\psi$ corresponds to the set of infinite models of ψ and so the automaton $\mathcal{A}_{\exists\psi} = \mathcal{A}_\psi$ recognizes the models of $\exists\psi$. Moreover, observe that, for every $\mathrm{LDL_F}$ formula ψ and an infinite play π, we have that $\pi \models \forall\psi$ iff $\pi \not\models \exists\neg\psi$. This means that, in order to build the automaton $\mathcal{A}_{\forall\psi}$ for a formula of the form $\forall\psi$, one can first consider the formula $\neg\psi$ and build the corresponding automaton $\mathcal{A}_{\exists\neg\psi}$. It follows that $\mathcal{L}(\mathcal{A}_{\exists\neg\psi})$ is the set of infinite plays that satisfy $\exists\neg\psi$, which is the complement of the set of plays satisfying $\forall\psi$. Thus, $\mathcal{A}_{\forall\psi} = \overline{\mathcal{A}_{\exists\neg\psi}}$. Using these constructions one can solve NE NON-EMPTINESS, E-NASH,

[1]Of course, with respect to EXPRESSIVENESS this is an irrelevant feature of the automata construction.

and A-NASH with QPLDL_{F} goals by applying the same automata-theoretic technique used for LDL_{F}. Then, we have the next result.

THEOREM 7. NE NON-EMPTINESS*,* E-NASH*, and* A-NASH *with* QPLDL_{F} *goals can be solved in 2EXPTIME. Moreover, the sets of Nash equilibria for this class of games is ω-regular.*

To obtain a matching lower bound, observe that, given the interpretation of QPLDL_{F} formulae, it is possible to encode the synthesis problem for LDL_{F} formulae as presented in [3]. Indeed, in such a case we only have to set a two-player game $\mathcal{G}$ in which, say Player 1, controls the same variable as the system for the synthesis problem, and Player 2 controls the environment variables. At this point, by setting $\gamma_1 = \exists\psi$ and $\gamma_2 = \forall\neg\psi$, one ensures that Player 1 and Player 2 have exactly the same behaviours of system and environment in the synthesis problem, respectively. In addition to Player 1 and Player 2, to ensure a reduction to NE NON-EMPTINESS one can add two players that trigger a "matching pennies" game in case ψ is not synthesised. With this reduction it follows that NE NON-EMPTINESS is 2EXPTIME-complete.

Formally, consider an LDL_{F} formula φ and the synthesis problem for it, in which the system controls a set of (output) variables X while the environment controls a set of (input) variables Y. Then, consider the four-player iBG_F $\mathcal{G}_\varphi$ with QPLDL_{F} goals such that:

- Player 1 controls X and has $\gamma_1 = \exists\varphi$ as goal;
- Player 2 controls Y and has $\gamma_2 = \forall\neg\varphi$ as goal;
- Player 3 controls a fresh Boolean variable p and has $\gamma_3 = \exists\varphi \vee (p \leftrightarrow q)$ as goal; and
- Player 4 controls a fresh Boolean variable q and has $\gamma_4 = \exists\varphi \vee \neg(p \leftrightarrow q)$ as goal.

Using the above construction, we can show that the synthesis problem for an LDL_{F} formula φ can be solved by addressing the NE NON-EMPTINESS problem for $\mathcal{G}_\varphi$, from which we derive the following theorem.

THEOREM 8. NE NON-EMPTINESS*,* E-NASH*, and* A-NASH *are 2EXPTIME-complete in games with* QPLDL_{F} *goals.*

In fact, Theorem 8 is proved using the lemma given below.

LEMMA 1. *The synthesis problem for an* LDL_{F} *formula φ over a set of Boolean variables $\mathrm{X} \cup \mathrm{Y}$, where the system controls the variables in X and the environment the variables in Y has a positive answer if and only if the game $\mathcal{G}_\varphi$ has a Nash equilibrium.*

Games with Memoryless Strategies.

In this subsection, we study games with memoryless strategies. We say that a strategy $\sigma_i = (\mathrm{S}_i, s_i^{\mathrm{o}}, \delta_i, \tau_i)$ for Player i is *memoryless* if $\mathrm{S}_i = 2^\Phi$ and δ_i is deterministic. Intuitively, a strategy is memoryless if, for each state of the game, it always chooses the same action at such state. Moreover, a play $\pi \in (2^\Phi)^\omega$ is said to be memoryless if, for all $v, w \in 2^\Phi$, if $\pi_k = v$ and $\pi_{k+1} = w$, for some $k \in \mathbb{N}$, then, for all $h \in \mathbb{N}$, if $\pi_h = v$ then $\pi_{h+1} = w$. A profile $\vec{\sigma}$ made by memoryless strategies can only generate memoryless plays and vice-versa.

Moreover, it is not hard to build a polynomial size NBW $\mathcal{A}_{mless}$ accepting all and only the memoryless plays. This turns out to be useful in addressing the case of memoryless strategies. Indeed, to solve the NE NON-EMPTINESS problem with memoryless strategies, we only need to adjust the general procedure by pairing the automaton $\mathcal{A}_{mless}$ to every single component of the automaton $\mathcal{A}_{NE}(\mathcal{G})$ This operation then adds the memoryless requirement to the goal of a player and to the punishment strategies.

Now, although this solution technique allows one to prove that the set of Nash Equilibria in memoryless games is ω-regular, this is not optimal from a computational complexity point of view, which is still 2EXPTIME. For instance, a brute-force procedure can solve the problem in EXPSPACE. Indeed, given the definition of strategies, we know that a memoryless strategy for a player in the game has (at most) 2^Φ states. Then, a memoryless strategy, as well as a strategy profile, can be guessed in time exponential in the size of Φ and saved using exponential space. In addition, using NE MEMBERSHIP we can check in PSPACE whether such a strategy profile is a Nash equilibrium of the game. Formally, we have:

THEOREM 9. NE NON-EMPTINESS *with memoryless strategies is in EXPSPACE. Moreover, the sets of Nash equilibria for this class of games is ω-regular.*

Games with Myopic Strategies.

Another important game-theoretic setting is the one given by *myopic strategies* as they can be used to define *all* beneficial deviations. A game with myopic strategies is called a *myopic* $\textit{iBG}_F$. We say that a strategy $\sigma_i = (\mathrm{S}_i, s_i^{\mathrm{o}}, \delta_i, \tau_i)$ for Player i is myopic if its transition function does not depend on the input variables, *i.e.*, such that for each $s \in \mathrm{S}_i$ and $v, v' \in 2^\Phi$, we have $\delta_i(s, v) = \delta_i(s, v')$. In a myopic iBG_F, players are only allowed to use myopic strategies. In [11] it is shown how to reduce NE NON-EMPTINESS for myopic iBG to the satisfiability of the QPTL formula

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

where the formulae γ_i are the LTL goals of the players in the myopic iBG instance and the quantifier alternation is 1 (an alternation fragment for which the complexity is known to be EXPSPACE [22]).

To apply the solution provided in [22] to check the satisfiability of φ, one first has to transform each γ_i into the NBW automata recognizing their models. In the case of iBG_F, these LTL formulae are replaced by LDL_{F} formulae. However, as shown in the previous section, the infinite models of an LDL_{F} formula γ_i can also be recognized by NBW automata that are equivalent to some ω-regular expression of the form $\alpha \cdot (2^\Phi)^\omega$. Thus, in order to solve NE NON-EMPTINESS for myopic iBG_F, we can first transform every LDL_{F} goal γ_i into the corresponding NBW $\mathcal{A}_{\gamma_i}$ and then follow the technique used in [22]. Note that the same reasoning applies also for the case of QPLDL_{F} goals. We then obtain the following result for games with myopic strategies:

THEOREM 10. *The* NE NON-EMPTINESS *problem for myopic* $\textit{iBG}_F$ *with* LDL_{F} *or* QPLDL_{F} *goals can be solved in EXPSPACE.*

At this point it is important to note that a key observation behind this result is the fact that when playing with myopic strategies the strategies that are used to construct a run that is sustained by a Nash equilibrium (a Nash run) must be oblivious to players' deviations.

Games with Strong Nash equilibria.

Despite being the most used solution concept in non-cooperative game theory [19], Nash equilibrium still has some limitations, for instance, it is not always stable and also it includes non desirable equilibria. As an example, consider a two-player game in which Player 1 controls a variable p and has the LDL_{F} goal $\gamma_1 = q$, while

Player 2 controls a variable q and has the LDL$_F$ goal $\gamma_2 = p$ [2]. It is clear that every strategy profile $\vec{\sigma}$ is a Nash equilibrium. Indeed, even in case a goal γ_i is not satisfied, the corresponding player cannot deviate from it, as the satisfaction of each player's goal is fully controlled by the other one. However, the desired outcome for both players is to satisfy both goals. Then, if we allow the two players to *collaboratively* deviate, the only stable outcomes are the ones making true both p and q at the first round of the computation.

A strong Nash equilibrium considers not only a single player's deviation, but also every possible coalition of players having a collective deviation incentive. Formally, for a given strategy profile $\vec{\sigma}$, we say that it is a *strong Nash equilibrium* if there is no subset $\mathrm{C} \subseteq \mathrm{N}$ and partial strategy profile $\vec{\sigma}'_{\mathrm{C}}$ such that, for all $i \in \mathrm{C}$, $\pi(\vec{\sigma}_{-\mathrm{C}}, \vec{\sigma}'_{\mathrm{C}}) \succ_i \pi(\vec{\sigma})$. 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. By $\mathrm{sNE}(\mathcal{G})$ we denote the set of strong Nash equilibria in $\mathcal{G}$. To check whether there exists a strong Nash equilibrium in a game using an automata-theoretic approach, we need to be able to express this notion of beneficial collective deviation with an appropriate automaton.

To do this, we just need to adjust the automaton $\overline{\mathcal{A}''_{\gamma_j}}$ given in the previous section, used to recognize all the plays that can be generated by a punishment strategy of the coalition N_{-j} against j, having goal γ_j. Indeed, the concept of punishment can be easily lifted to punishing a group of players. To do this, for a set $\mathrm{C} \subseteq \mathrm{N}$, consider the automaton $\mathcal{A}_{\mathrm{C}} = \bigotimes_{j \in \mathrm{C}} \mathcal{A}_{\gamma_j}$ recognizing all the models that satisfy every γ_j, for $j \in \mathrm{C}$. Then, as in the previous section, we can build the automaton $\overline{\mathcal{A}''_{\mathrm{C}}}$ that recognizes the plays generated by a punishment strategy for the coalition $\mathrm{N} \setminus \mathrm{C}$ against the goal being the conjunction of goals of coalition C. At this point, as in the case of Nash equilibrium, let us fix a set of "winners" $\mathrm{W} \subseteq \mathrm{N}$ in the game and then consider the product automaton $\mathcal{A}_{\overline{2^{\mathrm{L}}}} = \bigotimes_{\mathrm{C} \in 2^{\mathrm{L}}} \overline{\mathcal{A}''_{\mathrm{C}}}$. By the semantics of the product operation we obtain that the automaton $\mathcal{A}_{\overline{2^{\mathrm{L}}}}$ accepts those paths that are generated by some punishment profile, for each coalition of players $\mathrm{C} \in 2^{\mathrm{L}}$.

Thus, we have that the product automaton $\mathcal{A}_{\mathrm{W}} \otimes \mathcal{A}_{\overline{2^{\mathrm{L}}}}$ accepts exactly those paths for which every γ_i, with $i \in \mathrm{W}$, is satisfied while, for each coalition $\mathrm{C} \in 2^{\mathrm{L}}$ the coalition $\mathrm{N}_{-\mathrm{C}}$ is using a punishment strategy against C. Now, as for Nash equilibria, we need to quantify over $\mathrm{W} \subseteq \mathrm{N}$ to obtain an automata characterisation:

$$\mathcal{A}_{\mathrm{sNE}}(\mathcal{G}) = \bigoplus_{\mathrm{W} \subseteq \mathrm{N}} \mathcal{A}_{\mathrm{W}} \otimes \mathcal{A}_{\overline{2^{\mathrm{L}}}}.$$

The proof of correctness of this construction and its complexity is as for Theorem 5. Moreover, the same result can be obtained also with QPLDL$_F$ objectives. Also, observe that the reduction from LDL$_F$ synthesis provided for NE NON-EMPTINESS with QPLDL$_F$ goals can be reused with the same construction for the case of sNE NON-EMPTINESS. Formally, we have the following result.

THEOREM 11. sNE NON-EMPTINESS *is in 2EXPTIME for both* LDL$_F$ *and* QPLDL$_F$. *In particular, for games with* QPLDL$_F$ *goals, the problem is 2EXPTIME-complete. In addition, the set of strong Nash equilibria for games with either kind of goals is ω-regular.*

5. CONCLUDING REMARKS

Iterated Boolean Games Revisited. In the introduction section it was pointed out that the iBG and iBG$_F$ frameworks rely on different automata techniques, and that iBG$_F$ is better suited in certain scenarios. However, it is not the case that the iBG$_F$ framework generalises iBGs. Indeed, it should be noted that they are incomparable models: while iBG cannot be used to reason about games with goals over finite traces, iBG$_F$ cannot be used to reason about games with goals over infinite traces *only*, that is, regardless of the satisfaction of players' goals in the associated finite traces; from a logical point of view, while iBG considers LTL, iBG$_F$ can handle goals in LDL$_F$, which on finite traces is strictly more expressive than LTL [4], and also than LTL over finite traces. However, as shown here using new automata techniques, the complexities of some problems in each game model coincide in the worst case for many variants of these different kinds of iterated games.

Automata for Linear Dynamic Logic. Logic, games, and automata are intimately related; see, *e.g.*, [1, 6] and references therein for examples. We build upon the automata constructions for Linear Dynamic Logic (LDL$_F$ [4, 3]), which were introduced to solve the satisfiability and synthesis problems for LDL$_F$ over finite traces. Specifically, we have initially used such constructions to translate LDL$_F$ formulae to alternating automata on finite words (AFW) and, based on them, we have defined *new and optimal automata constructions* that characterise the existence of (strong) Nash equilibria on top of the standard Boolean games framework.

However, even though most, but not all, of the automata constructions we have presented in this paper are optimal, they still enjoy two useful properties. Firstly, that they are strongly based on automata on finite words, with only an extension to deal with infinite runs, a feature that could be used a lot further. Secondly, that such automata constructions recognise the sets of Nash runs, which, as shown in this paper, makes them extremely useful from a semantic point of view. Indeed, using other automata approaches, *e.g.*, for iBGs, our expressiveness results do not easily follow.

In addition, the automata constructions defined in this paper can be modified to reason about other game settings, making it a rather widely adaptable reasoning technique. For instance, we believe it is also possible to extend *some* of the results we have obtained to two-player games with imperfect information. This should be possible, in some cases, using recent automata constructions to reason about LDL$_F$ formulae under partial observation [5].

Imperfect Information. Following the research line delineated by the papers [4, 3, 5], one might wonder about the complexity of solving iBG$_F$ in the context of imperfect information. It is important to notice that the synthesis problems for LTL for both perfect and imperfect information is decidable [21, 15]. On the other hand, the NE-NONEMPTINESS problems for games having LTL goals is decidable for the perfect information case [10], but undecidable for the imperfect information case [11]. Similarly, regarding LDL$_F$ goals, the synthesis problem is decidable for both perfect [3] and imperfect [5] information, while we here prove that the NE-NONEMPTINESS problem is decidable. This suggests that the same problem might be undecidable under the imperfect information assumption, as it is for conventional iBGs. However, as the expressive power of LDL$_F$ is incomparable with the one of LTL, it is not clear whether the undecidability proof (which strongly relies on the expressiveness of LTL) can be retained in this case. For this reason, we plan to address this question in future work.

Acknowledgements

We acknowledge with gratitude the financial support of the ERC Advanced Investigator grant 291528 ("RACE") at Oxford.

[2]Observe that γ_1 and γ_2 are propositional logic formulae, *i.e.*, special cases of LDL$_F$.

REFERENCES

[1] L. Bozzelli, B. Maubert, and S. Pinchinat. Uniform strategies, rational relations and jumping automata. *Information and Computation*, 242:80–107, 2015.

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

[3] G. De Giacomo and M. Vardi. Synthesis for LTL and LDL on finite traces. In *IJCAI*, pages 1558–1564, 2015.

[4] G. De Giacomo and M. Y. Vardi. Linear temporal logic and linear dynamic logic on finite traces. In *IJCAI*, pages 854–860. IJCAI/AAAI, 2013.

[5] G. De Giacomo and M. Y. Vardi. Ltl_f and ldl_f synthesis under partial observability. In *IJCAI*, pages 1044–1050. IJCAI/AAAI Press, 2016.

[6] N. Fijalkow, S. Pinchinat, and O. Serre. Emptiness of alternating tree automata using games with imperfect information. In *FSTTCS*, volume 24 of *LIPIcs*, pages 299–311. Dagstuhl, 2013.

[7] D. Fisman, O. Kupferman, and Y. Lustig. Rational synthesis. In *TACAS*, volume 6015 of *LNCS*, pages 190–204. Springer, 2010.

[8] J. Gutierrez, P. Harrenstein, and M. Wooldridge. Iterated boolean games. In F. Rossi, editor, *IJCAI*, pages 932–938. IJCAI/AAAI, 2013.

[9] J. Gutierrez, P. Harrenstein, and M. Wooldridge. Expresiveness and complexity results for strategic reasoning. In *CONCUR*, volume 42 of *LIPIcs*, pages 268–282. Schloss Dagstuhl, 2015.

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

[11] J. Gutierrez, G. Perelli, and M. Wooldridge. Imperfect information in reactive modules games. In *KR*, pages 390–400. AAAI Press, 2016.

[12] P. Harrenstein, W. van der Hoek, J. Meyer, and C. Witteveen. Boolean Games. In *TARK*, pages 287–298, 2001.

[13] O. Kupferman, G. Perelli, and M. Y. Vardi. Synthesis with rational environments. In *EUMAS*, volume 8953 of *LNCS*, pages 219–235. Springer, 2014.

[14] O. Kupferman, G. Perelli, and M. Y. Vardi. Synthesis with rational environments. *Annals of Mathematics and Artificial Intelligence*, 78(1):3–20, 2016.

[15] O. Kupferman and M. Y. Vardi. *Synthesis with Incomplete Information*, pages 109–127. Springer, 2000.

[16] A. Kučera and J. Strejček. The Stuttering Principle Revisited. *Acta Informatica*, 41(7–8):415–434, 2005.

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

[18] D. Niwinski and I. Walukiewicz. Relating hierarchies of word and tree automata. In *STACS*, volume 1373 of *LNCS*, pages 320–331. Springer, 1998.

[19] M. Osborne and A. Rubinstein. *A Course in Game Theory.* MIT Press, 1994.

[20] A. Pnueli. The temporal logic of programs. In *FOCS*, pages 46–57. IEEE Computer Society, 1977.

[21] A. Pnueli and R. Rosner. On the Synthesis of a Reactive Module. In *POPL*, pages 179–190. ACM, 1989.

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

[23] A. P. Sistla and E. M. Clarke. The Complexity of Propositional Linear Temporal Logics. *Journal of the ACM*, 32(3):733–749, 1985.

[24] M. Vardi. An Automata-Theoretic Approach to Linear Temporal Logic. In *Logics for Concurrency - Structure versus Automata*, pages 238–266, 1995.

[25] M. Wooldridge, J. Gutierrez, P. Harrenstein, E. Marchioni, G. Perelli, and A. Toumi. Rational verification: From model checking to equilibrium checking. In *AAAI*. AAAI Press, 2016.