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 (LDL\textsubscript{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 characterization and verification of equilibria in multi-player games with goals/objectives expressed using logics based on LDL\textsubscript{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 LDL\textsubscript{F} goals are considered, in the setting we study—iterated Boolean games with goals over finite traces (iBG\textsubscript{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 LDL\textsubscript{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 \(\gamma_i\), and player \(i\)'s main purpose is to ensure that \(\gamma_i\) is satisfied. The strategies and choices for each player \(i\) are defined with respect to a set of Boolean variables \(\Phi_i\), drawn from an overall set of variables \(\Phi\). Player \(i\) is assumed to have unique control over the variables in \(\Phi_i\), 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 \(\gamma_i\) 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 \(\gamma_i\) satisfied. In an iBG, however, players' goals \(\gamma_i\) are Linear Temporal Logic formulae (LTL [20]), rather than PL formulae, which are naturally interpreted over infinite sequences of valuations of the variables in \(\Phi\); 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\textsubscript{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 (LDL\textsubscript{F}) which are evaluated over finite sequences of valuations of the variables in \(\Phi\), that is, over finite traces of valuations, instead of PL or LTL formulae as in conventional or iterated Boolean games, respectively. Thus, while in an iBG 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 LDL\textsubscript{F} goals. LDL\textsubscript{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 (LTL\textsubscript{F}) is as expressive as first-order logic (FOL) over finite traces [4]. This, in turn, implies that, over finite traces, while with LTL\textsubscript{F} we can only describe star-free regular languages/properties, with LDL\textsubscript{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 LDL\textsubscript{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-
structions originally defined to reason about \(\text{LDL}_F\) 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 \(\text{LDL}_F\) 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 \(\text{LDL}_F\) formulae—and show that deciding whether an \(\text{iBG}_F\) has a Nash equilibrium can be solved in 2EXPTIME, thus no harder than solving \(\text{LDL}_F\) synthesis [3]. The automata technique we use for this problem also shows that the set of Nash equilibria in these games is \(\omega\)-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 \(\text{iBG}_F\) framework. In particular, we show that a small extension of the goal language, which we call Quantified-Prefix Linear Dynamic Logic (\(\text{QPLDL}_F\)), has the same automata-theoretic characteristics as \(\text{LDL}_F\), and so it can be studied using the same techniques. Moreover, \(\text{LDL}_F\) synthesis can be expressed in \(\text{QPLDL}_F\), 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 \(\omega\)-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 \(\text{LDL}_F\). 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 \(\text{LDL}_F\) synthesis and satisfiability problems—generalising ideas initially used to reason about \(\text{LTL}\) 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 \(\omega\)-regular set, to the best of our knowledge, a semantic characterisation not previously known, and which do not immediately follow from other representations of Nash equilibria—see, e.g., [2, 7, 13, 17, 14].

### Motivation and Previous Work

While studying either iterated games or \(\text{LDL}_F\) 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 \(\text{LTL}\) or even \(\text{LTL}\) over finite traces (\(\text{LTL}_F\)), 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 \(\text{LDL}_F\), 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 \(\text{LTL}\) or have a more intuitive specification in \(\text{LDL}_F\) than in \(\text{LTL}\). Together with applications to planning AI systems (see [4, 3]), this is an example of another instance where one can see an advantage of \(\text{iBGs}\) over \(\text{iBGs}\).

Moreover, regarding previous work, while our model builds on \(\text{iBGs}\), where goals are given by \(\text{LTL}\) 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 \(\text{iBGs}\) 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 \(\text{iBGs}\), both in [9] and in [8]. To be more precise, for \(\text{iBGs}\) 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 \(\text{iBGs}\). 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 \(\text{LDL}_F\), 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 non-zero-sum concurrent scenarios. This difference leads to a completely different technical treatment/manipulation of the automata that can be initially constructed from \(\text{LDL}_F\) formulae. In fact, their automata constructions and ours are the same only up to the point where \(\text{LDL}_F\) 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 (\(\text{LDL}_F\)), 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 \(\text{LDL}_F\) is as follows:

\[
\varphi ::= p \mid \neg \varphi \mid \varphi \land \varphi \mid \varphi \lor \varphi \mid \langle \rho \rangle \varphi \mid [\rho] \varphi \\
\rho ::= \psi \mid \varphi ? \mid \rho + \rho \mid \rho; \rho \mid \rho^* ,
\]

where \(p\) is an atomic proposition in \(\Phi\); \(\psi\) denotes a propositional formula over the atomic propositions in \(\Phi\); \(\rho\) denotes path expressions, which are regular expressions over propositional formulae \(\psi\), with the addition of the test construct \(\varphi ?\) from propositional dynamic logic (\(\text{PDL}\)); and \(\varphi\) stands for \(\text{LDL}_F\) formulae built by applying Boolean connectives and the modal connectives. Tests are used to insert checks for satisfaction of additional \(\text{LDL}_F\) formulae.

The semantics of \(\text{LDL}_F\) formulae is as follows. \(\text{LDL}_F\) formulae are interpreted over finite traces of the form \(\pi : \{0, \ldots, t\} \rightarrow 2^\Phi\) 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 \((i, j) \in R(\rho, \pi)\) and \(\pi, j \models \varphi\);
• \( \pi, i \models \varphi \) if for all \( j \in \{ i, \ldots, t \} \), if \( (i, j) \in R(\rho, \pi) \) then \( \pi, j \models \varphi \);

where \( R(\rho, \pi) \subseteq N \times N \) is recursively defined by

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

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

An iBG\(F_\omega \) is a tuple \( G = (N, \Phi, \Phi_0, \Phi_1, \gamma_1, \ldots, \gamma_n) \), where \( N = \{1, \ldots, n\} \) is a set of players, \( \Phi \) is a set of Boolean variables, partitioned into sets \( \Phi_0 \), \( \Phi_1 \), \ldots, \( \Phi_n \), and the goals \( \gamma_1, \ldots, \gamma_n \) of the game are LDL\(F_\omega \) formulae over \( \Phi \). In an iBG\(F_\omega \) each player \( i \) is assumed to control a set of propositional variables \( \Phi_i \), in the sense that player \( i \) has the power to set the values (true “⊤” or false “⊥”) of each of the variables in \( \Phi_i \). An action for player \( i \) is a possible valuation \( v_i \in 2^\Phi_i \). An action vector \( \vec{v} = (v_1, \ldots, v_n) \) is a collection of actions, one for each player in the game. Every action vector determines an overall valuation for the variables in \( \Phi \).

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\(F_\omega \) formula is satisfied on an iBG\(F_\omega \). The most natural way to do so, also implicitly followed in [3], is to say that an infinite play \( \pi \) satisfies an LDL\(F_\omega \) formula if and only if there exists a finite valuation \( v \in 2^\Phi \) that equals to the non-atomic formula \( \varphi \) under \( \pi \) and \( v \), i.e., \( \pi \models \varphi \) if there is \( \vec{v} \in 2^\Phi \) such that \( \pi \models \varphi \).

In order to avoid that the protocol manager wrongly marks the download of a file as completed, we have the following requirements:

\[ \gamma_{\text{incom}} = \Lambda_{n<n_1} \land (\neg u_1 \land (u_2, u_1)^{n_1}) \]
\[ \gamma_{\text{incom}} = \Lambda_{n<n_2} \land (\neg u_1 \land (u_2, u_1)^{n_2}) \]

The goal of the protocol manager is therefore given by the conjunction of all these conditions:

\[ \gamma_0 = \gamma_{\text{incom}} \land \gamma_{\text{incom}} \land \gamma_{\text{incom}} \land \gamma_{\text{incom}} \land \gamma_{\text{incom}} \land \gamma_{\text{incom}} \land \gamma_{\text{incom}} \land \gamma_{\text{incom}} \land \gamma_{\text{incom}} \]

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\(F_\omega \) are modelled as deterministic finite state machines. Formally a deterministic finite state machine for player \( i \) is a tuple \( \sigma_i = (S_i, s_i^0, \delta_i, \tau_i) \) where, \( S_i \) is a finite set of internal states, \( s_i^0 \) is the initial state, \( \delta_i : S_i \times 2^S \rightarrow S_i \) is a transition function, and \( \tau_i : S_i \rightarrow 2^S \) is the action function. By \( S \times \sigma \) 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 \( A \subseteq N \), we use the notation \( \sigma_A \) to denote a tuple of strategies, one for each player in \( A \). Moreover, we use the notation
σ_{1A} to denote a tuple of strategies, one for each player in N \ A. We also use σ_i in place of σ_{i(1)} and \bar{σ}_{-i} in place of \bar{σ}_{N\setminus i(1)}. Finally, for two strategy profiles \bar{σ} and \bar{σ}', by (\bar{σ}_{A}, \bar{σ}'_{-A}) we denote the strategy profile given by associating the strategies in \bar{σ} to players in A and strategies in \bar{σ}' to players in N \ A.

Since strategies are deterministic, each profile \bar{σ} determines a unique play, denoted by π(\bar{σ}), which consists of an infinite sequence of valuations, one for each round of the game. Each player i has a preference relation over plays π ∈ (2^S)^\infty, which is determined by its goal γ_i. We say that π is preferred over π' by agent i, and write π \geq_i π', if and only if π' \models γ_i implies that π \models γ_i. Using this notion of preference, one can introduce the concept of Nash Equilibrium. We say that \bar{σ} is a Nash Equilibrium strategy profile if, for each agent i and a strategy σ'_i ∈ Str_i, it holds that π(\bar{σ}) \geq_i π(\bar{σ}, σ'_i). In addition, by NE(G) \subseteq Str_1 \times \ldots \times Str_n we denote the set of Nash Equilibria of the game G.

Example 2. Consider again the system in Example 1. A possible strategy σ_i for player i is a finite-state machine that sets variable u_i to true on odd rounds of the execution, while a strategy σ_2 for player 2 might set 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 d_0 to true only after u has been set to true exactly n times in the execution. Then, the strategy profile \bar{σ} = (σ_0, σ_1, σ_2) will be such that the execution π = π(\bar{σ}) 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 G and a strategy profile \bar{σ}:

Is it the case that \bar{σ} ∈ NE(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 G:

Is it the case that NE(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 analogs of modeling checking in a game-theoretic multi-agent setting) that are formally stated as follows:

E/A-Nash. Given a game G and LDL_\Sigma formula \varphi:

Does π(\bar{σ}) \models \varphi hold, for some/all \bar{σ} ∈ NE(G)?

which asks if \varphi is satisfied by some/every Nash equilibrium of 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 A = (Σ, S, s_0, q_0, F, G), recognizing a regular language L(A). Then consider the NBW A_∞ = (Σ, S, s_0, q_0, F', G'), where, for all σ and s, we have that g'(σ, s) = g(σ, s), if s \notin F, and g'(σ, s) = \{s\}, otherwise.

Intuitively, the automaton A_∞ accepts all and only the infinite words π_∞ having a prefix π accepted by A. This fact can be shown with the following theorem.

**Theorem 1.** Let A be a NFW. Then, for all π_∞ ∈ Σ^∞, we have that π_∞ ∈ L(A_∞) iff there exists k ∈ N such that π = (π_∞)_≤k ∈ L(A).

As a corollary, we obtain the following result.

**Corollary 1.** L(A_∞) = \{π_∞ ∈ (2^S)^ω : π_∞ \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 σ_i = (S_i, s'_i, δ_i, τ_i) for a player i, a corresponding DFW A(σ_i) = (Σ, Q_i, q'_i, ρ_i, F_i) where: Σ = 2^S is the alphabet set, Q_i = (S_i × 2^S) \cup \{sink\} is the state set, where sink \notin S_i is a fresh state, q'_i = (s'_i, 0) is the initial state, F_i = S_i × 2^S is the final state set, and ρ_i is the transition relation such that, for all (s, v) ∈ S_i × 2^S and v' ∈ Σ,

\begin{align*}
\rho_i((s, v), v') &= \begin{cases} 
\delta_i(s, v), & \text{if } \tau_i(s) = v' \land s'_i, \\
\text{sink}, & \text{otherwise}, 
\end{cases} \\
\rho_i(\text{sink}, v') &= \text{sink}
\end{align*}

Let L(A(σ_i)) denote the set of infinite words in (2^S)^ω accepted by A(σ_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 A ⊆ N and a partial strategy profile \bar{σ}_A, we have that, for A(\bar{σ}_A) ≜ \bigotimes_{i \in A} A(σ_i), the product of these automata, the language L(A(\bar{σ}_A)) contains exactly those infinite plays in a game where players in A play according to the strategies given in \bar{σ}_A. Moreover, in [3] it is shown, for every LDL_\Sigma formula \varphi, how to build and check on-the-fly a NFW A_φ = (Σ, 2^S, \{s_0\}, δ, \{s_f\}) such that, for every finite trace π ∈ (2^S)^*, we have π \models \varphi if and only if π ∈ L(A_φ), where by L(A_φ) we denote the language of finite words (that is, the language of finite traces over 2^S) accepted by the automaton A_φ.

**Algorithm 1:** Intersection construction.

1. **Input:** an LDL_\Sigma formula \varphi and an NBW A = (2^S, Q, q_0, G, F).
2. **Output:** NFW A_∞^x \times A = (2^S, S', \{s_0\}, F', G').
3. s'_0 \leftarrow \{\{\varphi, q_0, 1\}\};
4. F' \leftarrow \{0\} \times Q \times \{1\};
5. S' \leftarrow \{s_0\} \cup F';
6. g' \leftarrow \{((0, q, 1), (0, (q', 2)) : Π ∈ 2^S \land q' \in g(q, Π) \}
\cup \{(0, (q, 2), (0, q', 2)) : Π ∈ 2^S \land q' \in g(q, Π) \land q \in Q \setminus F'\}
\cup \{(0, (q, 2), (0, q', 1)) : Π ∈ 2^S \land q' \in g(q, Π) \land q \in F'\}
7. **while** (S' or g' change) **do**
8. **for** s ∈ S', q ∈ Q and Π ∈ 2^S **do**
9. **if** q' \subseteq CL(\varphi) and q' \in g(q, Π) **do**
10. **if** s \notin F' **then**
11. S' \leftarrow S' \cup \{(s, q, 2), (s', q', 1)\};
12. **else**
13. S' \leftarrow S' \cup \{(s', q', 1), (s, q, 2)\};
14. g' \leftarrow g' \cup \{(s, q, 2), (s', q', 1)\};
Such a construction makes use of a function $\delta$ simulating the transition relation of the corresponding alternating finite word automaton (AFW), which takes a subformula $\psi$ of $\varphi$ 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 $A^i_\gamma$ of $i$. As a matter of fact, observe that the only final state $s_f$ of the automaton $A$, built in [3] does not have any outgoing transition. Then, given the construction of $A^i_\gamma$, 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 $A^i_\gamma$ with the automata $A_\sigma$ and $A_{\sigma'}$, 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 $A_\sigma$ and $A_{\sigma'}$, can be considered as NBW. Thus, it is enough to deliver an algorithm building an automaton intersection between $A^i_\gamma$ and a generic NBW $A$. 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 $A^i_\gamma \otimes A$, where $A$ is a generic NBW, can be solved in polynomial space with respect to the size of the formula $\varphi$.

With this theorem in place, one can show that Algorithm 2 runs in PSPACE and solves NE Membership for $ibG_2$.

**Algorithm 2:** Algorithm for NE Membership.

1. **Input:** a game $G$ and a strategy profile $\sigma$.
2. **Output:** “Yes” if $\sigma \in NE(G)$; “No” otherwise.
3. for $i \in N$
4. if $L(A(\sigma_{-i}) \otimes A^i_\gamma) = \emptyset$ then
5. if $L(A(\sigma_{-i}) \otimes A^i_\gamma) \neq \emptyset$ 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 $(G, \sigma)$. 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 $L(A(\sigma_{-i}) \otimes A^i_\gamma) = \emptyset$, meaning that the play $\pi(\sigma_{-i})$ is such that $\pi(\sigma_{-i}) \models \gamma_i$. Thus, player $i$ is satisfied in the context $\sigma$ 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 $L(A(\sigma_{-i}) \otimes A^i_\gamma) \neq \emptyset$, which means that the satisfaction of $\gamma_i$ is incompatible with the partial strategy profile $\sigma_{-i}$, 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 $\sigma$ is a Nash equilibrium of the game.

On the other hand, assume $\sigma$ is a Nash equilibrium. Then, no player $i$ has an incentive to deviate. This can be the case for two reasons: either $\pi(\sigma_{-i}) \models \gamma_i$, or there is no strategy $\sigma_i$, such that $\pi(\sigma_{-i}, \sigma_i) \models \gamma_i$. If the former, then we have that $L(A(\sigma_{-i}) \otimes A^i_\gamma) = \emptyset$ and so the check on line 4 is false. If the latter, then it follows that $L(A(\sigma_{-i}) \otimes A^i_\gamma) = \emptyset$, 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 Algorithm 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 $\text{LDL}_\omega$ formulae, which is known to be PSPACE-complete [4].

Now, let us study NE Non-emptiness. Again, we use an automata-theoretic approach. We show how, given a game $G$, it is possible to construct an alternating automaton $A_{NE}(G)$ such that $A_{NE}(G)$ accepts precisely the set of plays that are generated by the Nash equilibria of $G$. A distinguishing feature of our automata technique is that it is language preserving, that is, $A_{NE}(G)$ 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, $\omega$-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 $A(\sigma_i)$ provided above, as there is no known strategy $\sigma_i$, 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 $G = (N, \Phi_i, \ldots, \Phi_n, \gamma_1, \ldots, \gamma_n)$ and a designated player $j \in N$, we say that $\sigma_{-j}$ is a punishment profile against $j$ if, for every strategy $\sigma_j$, it holds that $(\sigma_{-j}, \sigma_j) \not\models \gamma_j$. In [9], it has been proven that $\sigma \in NE(G)$ if and only if there exists $W \subseteq N$ such that $\sigma \models \gamma_i$ for every $i \in W$ and, for every $j \in L = N \setminus W$, the profile $\sigma_{-j}$ 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 $\Phi_{-j}$. To do this, we apply an automata-theoretic approach. First of all, we build the Rabin automaton (ARW) $A_{\sigma_i}$ used to recognize the models of $\gamma_i$. Then, we dualize the automaton to obtain the ARW $A_{\sigma_i}$ recognizing the complement language, i.e., the set of infinite words that do not satisfy $\gamma_j$. At this point, by means of Theorem 2 in [21], we build a nondeterministic Rabin automaton on trees (NRT) $A_{\sigma_i}$ 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 $\gamma_j$. Now, following Corollary 17 in [18], we can build a NW automaton $A_{\sigma_i}$ such that $L(A_{\sigma_i}) = \{ \pi \in (2^\omega)^\omega : \exists T \in L(A_{\sigma_i}), \pi \subseteq T \}$, where by $\pi \subseteq T$ we denote the following property: $\pi$ is a branch of the tree $T$ starting at its root.

Now, let us fix $W \subseteq N$ for a moment, and consider the product automaton $A_T = \bigotimes_{i \in W} A_{\sigma_i}$. By the semantics of the product operation we obtain that $A_T$ accepts those paths that are generated by some punishment profile, for each $j \in L$. Moreover, consider the automaton $A_W = \bigotimes_{i \notin W} A_{\sigma_i}$, recognizing the paths that satisfy every $\gamma_i$, for $i \in W$. Thus, we have that the product automaton $A_W \otimes A_T$ accepts exactly those paths for which every $\gamma_i$, with $i \in W$, is satisfied while, for each $j \in L$ 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 $W \subseteq N$. This, in terms of automata, corresponds to the union operation. Then, we get the following automata characterisation:

$$A_{NE}(G) = \bigoplus_{W \subseteq N} (A_W \otimes A_T).$$

**Theorem 4 (Expressiveness).** For a game $G$ with $\text{LDL}_\omega$ goals, the automaton $A_{NE}(G)$ recognizes the set of Nash runs of $G$. Moreover, the set of Nash runs of $G$ is $\omega$-regular.

**Proof.** We prove both implications. From left to right, assume that $\pi \in L(A_{NE}(G))$. Then, there is $W \subseteq N$ such that $\pi \in L(A_W \otimes A_T)$. Observe that, w.l.o.g, we can assume that $\pi$ is an ultimately periodic play [23] and so that there exists a finite-state
machine $\Delta_{e} = (Q_{e}, q_{0}^{e}, \delta_{e}, \tau_{e})$, controlling all the variables in $\Phi$, i.e., $\tau_{e} : Q_{e} \rightarrow 2^{\Phi}$, that generates $\pi$. Moreover, observe that, for each $j \in L$, $\pi \in L(\Delta_{e})$ implies that there exists $T_{j} \in L(\Delta_{e})$ such that $\pi \subseteq T_{j}$. This implies that, for each $j \in L$, there is a finite-state machine $\Delta_{j} = (Q_{j}, q_{0}^{j}, \delta_{j}, \tau_{j})$, controlling all the variables but $\Phi_{j}$, i.e., $\tau_{j} : Q_{j} \rightarrow 2^{\Phi_{j}}$, that generates the branches of $T_{j}$, according to the output of variables in $\Phi_{j}$, including $\pi$. Now, for each $i \in N$, define the strategy $\sigma_{i} = (S_{i}, s_{i}^{0}, \delta_{i}, \tau_{i})$ as follows:

- $S_{i} = Q_{e} \times \bigtimes_{j \in L} Q_{j} \times (L \cup \{T\})$ is the product of the state-space of $\Delta_{e}$ together with the state-space of each $\Delta_{j}$, for each $j \in L$, plus a flag component given by $L \cup \{T\}$;

- $s_{i}^{0} = (q_{0}^{e}, q_{0}^{j}, \ldots, q_{0}^{j}, T)$, collecting all the initial states of the finite state machines $\Delta_{e}$ and $\Delta_{j}$, for each $j \in L$, flagged with the symbol $T$;

- $\delta_{i}$ is defined as follows: for each $(q, q_{1}, \ldots, q_{|L|}, T)$ and $v \in 2^{\Phi}$, $\delta_{i}(q, q_{1}, \ldots, q_{|L|}, T, v) = (\delta_{e}(q, v), \delta_{1}(q_{1}, v), \ldots, \delta_{|L|}(q_{|L|}, v), \text{flag})$, where flat $= T$ if $v = \tau_{e}(q)$ and flat $= j$ if $v_{-j} = (\tau_{e}(q))_{-j}$ and $v_{j} \neq (\tau_{e}(q))_{j}$;

- $\tau_{i}((q, q_{1}, \ldots, q_{|L|}, T)) = (\tau_{e}(q)), \text{ and } \tau_{i}((q, q_{1}, \ldots, q_{|L|}, j)) = (\tau_{j}(q))$, for each $j \in L$.

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

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

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

**Theorem 5.** $\text{NE-NEMPTINESS with LDL}_e$ goals can be solved in $2\text{EXPTIME}$. 

Now, to show that E-NASH and A-NASH are in $2\text{EXPTIME}$, we can also apply an automata-theoretic approach. Indeed, for the E-NASH case, consider a game $G$ and an LDL$_e$ formula $\varphi$. Then, the automaton $A_{\varphi} \otimes A_{\text{NE}}(G)$ recognizes all the plays that both satisfy $\varphi$ 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 $A_{\varphi} \otimes A_{\text{NE}}(G)$. This product automaton recognizes all plays that do not satisfy the formula $\varphi$ 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 $2\text{EXPTIME}$. Formally, combining the results above, we also obtain the following theorem:

**Theorem 6.** $\text{E-NASH and A-NASH with LDL}_e$ goals can be solved in $2\text{EXPTIME}$. 

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 LDL$_e$ language used to represent players’ goals can be used to encode LDL$_e$ synthesis, studied in [3], as a $\text{NE NONEMPTINESS}$ 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 $\omega$-regular, as in the original problem. An EXSPACE brute-force approach can be used to show that the induced automata are suboptimal from a complexity point of view. 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 LDL$_e$ Goals.**

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

Syntactically, a QPLDL$_e$ formula $\varphi$ is obtained from an LDL$_e$ formula $\psi$ 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 QPLDL$_e$ formula of the form $\exists \psi$ and an infinite path $\pi$, we have that $\pi \models \exists \psi$ if there is $k \in N$ such that $\pi_{k} \models \psi$. Analogously, for a QPLDL$_e$ formula of the form $\forall \psi$, we have that $\pi \models \forall \psi$ if $\pi_{k} \models \psi$, for all $k \in N$.

The reader might note that $\exists \psi$ is equivalent to $\forall \neg \psi$ on infinite plays. This means that the set of models for $\exists \psi$ corresponds to the set of models of $\neg \psi$ and so the automaton $A_{\exists \psi} = A_{\neg \psi}$ recognizes the models of $\exists \psi$. Moreover, observe that, for every LDL$_e$ formula $\psi$ and an infinite play $\pi$, we have that $\pi \models \forall \psi$ if $\pi \models \neg \exists \psi$. This means that, in order to build the automaton $A_{\exists \psi}$ for a formula of the form $\forall \psi$, one can first consider the formula $\neg \pi$ and build the corresponding automaton $A_{\neg \exists \psi}$. It follows that $L(A_{\exists \psi})$ is the set of infinite plays that satisfy $\exists \neg \psi$, which is the complement of the set of plays satisfying $\forall \psi$. Thus, $A_{\exists \psi} = A_{\neg \exists \psi}$. Using these constructions one can solve NE NONEMPTINESS, E-NASH,
and A-NASH with QPLDLₜ goals by applying the same automata-theoretic technique used for LDLₓ. Then, we have the next result.

**Theorem 7.** NE Non-emptiness, E-Nash, and A-Nash with QPLDLₜ 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 QPLDLₜ formulae, it is possible to encode the synthesis problem for LDLₓ formulae as presented in [3]. Indeed, in such a case we only have to set a two-player game 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 γ₁ = ∃ψ and γ₂ = ∀¬ψ, 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 "matching pennies" game in case ψ is not synthesised. With this reduction it follows that NE Non-emptiness is 2EXPTIME-complete.

Formally, consider an LDLₓ 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 Fr G ϕ with QPLDLₜ goals such that:

- Player 1 controls X and has γ₁ = ∃X ψ as goal;
- Player 2 controls Y and has γ₂ = ∀¬ϕ as goal;
- Player 3 controls a fresh Boolean variable p and has γ₃ = ∃ϕ ∨ (p ↔ q) as goal; and
- Player 4 controls a fresh Boolean variable q and has γ₄ = ∃ϕ ∨ (¬p ↔ q) as goal.

Using the above construction, we can show that the synthesis problem for an LDLₓ formula ϕ can be solved by addressing the NE Non-emptiness problem for G ϕ, from which we derive the following theorem.

**Theorem 8.** NE Non-emptiness, E-Nash, and A-Nash are 2EXPTIME-complete in games with QPLDLₜ goals.

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

**Lemma 1.** The synthesis problem for an LDLₓ formula ϕ over a set of Boolean variables X ∪ 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 G ϕ has a Nash equilibrium.

**Games with Memoryless Strategies.**

In this subsection, we study games with memoryless strategies. We say that a strategy σᵢ = (Sᵢ, sᵢ, δᵢ, τᵢ) for Player i is memoryless if Sᵢ = 2ω and δᵢ 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 π ∈ (2ω)ω is said to be memoryless if, for all u, w ∈ 2ω, if πₖ = u and πₖ₊₁ = w, for some k ∈ N, then, for all h ∈ N, if πₕ = u then πₕ₊₁ = w. A profile σ made by memoryless strategies can only generate memoryless plays and vice-versa.

Moreover, it is not hard to build a polynomial size NBW An(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 An(mless) to every single component of the automaton An(FG) 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) 2ω 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 iBG Fr. We say that a strategy σᵢ = (Sᵢ, sᵢ, δᵢ, τᵢ) for Player i is myopic if its transition function does not depend on the input variables, i.e., such that for each s ∈ Sᵢ and v, v’ ∈ 2ψ, we have δᵢ(s, v) = δᵢ(s, v’).

In a myopic iBG Fr, 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

$$ϕ = \bigwedge_{W \in N} (\exists \Phi_1, \ldots, \Phi_n. (\bigwedge_{i \in W} \gamma_i \land \bigwedge_{j \in N \setminus W} (\forall \Phi_j. \neg \gamma_j)))$$

where the formulæ γᵢ 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 γᵢ into the NBW automata recognizing their models. In the case of iBG Fr, these LTL formulæ are replaced by LDLₓ formulae. However, as shown in the previous section, the infinite models of an LDLₓ formula γᵢ can also be recognized by NBW automata that are equivalent to some ω-regular expression of the form ω · (2ψ)ω. Thus, in order to solve NE Non-emptiness for myopic iBG Fr, we can first transform every LDLₜ goal γᵢ into the corresponding NBW An(γᵢ) and then follow the technique used [22]. Note that the same reasoning applies also for the case of QPLDLₜ goals. We then obtain the following result for games with myopic strategies:

**Theorem 10.** The NE Non-emptiness problem for myopic iBG Fr with LDLₓ or QPLDLₜ 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 LDLₜ goal γ₁ = q, while
Player 2 controls a variable $q$ and has the LDL$_F$ goal $\gamma_2 = p^\omega$. It is clear that every strategy profile $\sigma$ is a Nash equilibrium. Indeed, even in case a goal $\gamma_i$ 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 

In the introduction section, we can build the automaton $A$ given in the previous section, used to recognize all the plays that can be generated by a punishment strategy of the coalition $N \setminus j$ against $j$, having goal $\gamma_j$. Indeed, the concept of punishment can be easily lifted to punishing a group of players.

To do this, for a set $C \subseteq N$, consider the automaton $A_C = \bigotimes_{j \in C} A_{\gamma_j}$ recognizing all the models that satisfy every $\gamma_j$, for $j \in C$. Then, as in the previous section, we can build the automaton $A_C$ that recognizes the plays generated by a punishment strategy for the coalition $N \setminus C$ 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” $W \subseteq N$ in game and then consider the product automaton $A_{\mathcal{N}_C} = \bigotimes_{C \subseteq \mathcal{N}} A_C$. By the semantics of the product operation we obtain that the automaton $A_{\mathcal{N}_C}$ accepts those paths that are generated by some punishment profile, for each coalition of players $C \subseteq 2^\mathcal{N}$. Thus, we have that the product automaton $A_W \otimes A_{\mathcal{N}_C}$ accepts exactly those paths for which every $\gamma_i$, with $i \in W$, is satisfied while, for each coalition $C \subseteq 2^\mathcal{N}$ the coalition $N \setminus C$ is using a punishment strategy against $C$. Now, as for Nash equilibria, we need to quantify over $W \subseteq \mathcal{N}$ to obtain an automata characterisation:

$$A_{\text{NE}}(\mathcal{G}) = \bigoplus_{W \subseteq \mathcal{N}} A_W \otimes A_{\mathcal{N}_C}.$$ 

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 $\omega$-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-NON-EMPTINESS 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-NON-EMPTINESS 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.
REFERENCES


