# Expressiveness and Complexity Results for Strategic Reasoning

**Julian Gutierrez, Paul Harrenstein, and Michael Wooldridge**

**Department of Computer Science, University of Oxford**
**Wolfson building, Parks road, Oxford, United Kingdom**

## Abstract

This paper presents a range of expressiveness and complexity results for the specification, computation, and verification of Nash equilibria in multi-player non-zero-sum concurrent games in which players have goals expressed as temporal logic formulae. Our results are based on a novel approach to the characterisation of equilibria in such games: a semantic characterisation based on *winning strategies* and *memoryful reasoning*. This characterisation allows us to obtain a number of other results relating to the analysis of equilibrium properties in temporal logic.

We show that, up to bisimilarity, reasoning about Nash equilibria in multi-player non-zero-sum concurrent games can be done in ATL$^*$ and that constructing equilibrium strategy profiles in such games can be done in 2EXPTIME using finite-memory strategies. We also study two simpler cases, two-player games and sequential games, and show that the specification of equilibria in the latter setting can be obtained in a temporal logic that is weaker than ATL$^*$. Based on these results, we settle a few open problems, put forward new logical characterisations of equilibria, and provide improved answers and alternative solutions to a number of questions.

**1998 ACM Subject Classification** F.4.1 Mathematical logic – Temporal logic, F.3.1 Specifying and Verifying and Reasoning about Programs, I.2.11 Distributed Artificial Intelligence – Multiagent systems

**Keywords and phrases** Temporal logic, Nash equilibrium, Game models, Formal Verification

**Digital Object Identifier** 10.4230/LIPIcs.CONCUR.2015.268

## 1 Introduction

In the last decade, there has been an increasing interest in game-theoretic models of concurrent and distributed systems as well as in temporal logic formalisms to reason about the complex behaviour of such systems. These logical formalisms are used to reason about linear-time or branching-time properties of the systems, and usually extend logics such as LTL or CTL. The alternating-time temporal logic ATL$^*$ of Alur, Henzinger, and Kupferman [1] is an extension of CTL$^*$ that is intended to support, in addition, reasoning about strategic behaviour. ATL$^*$ replaces the path quantifiers of CTL$^*$ with *agent/strategy* modalities that can be used to express properties of coalitions of agents in game-like concurrent and multi-agent systems. ATL$^*$ is probably the most popular and widely used temporal logic for *strategic reasoning* in computer science, artificial intelligence, and multi-agent systems research [8, 16].

The theory of ATL$^*$ is well understood. In particular, it is known that satisfiability and model checking for ATL$^*$ are 2EXPTIME-complete [1, 15] – theoretically better than for many of its extensions, where these problems are non-elementary, or even undecidable [3, 10, 11]. ATL$^*$ is a natural logic to reason about winning strategies in game-like systems, which makes it sufficiently powerful for many computational purposes. For instance, several design and verification problems can be reduced to questions about the existence and computation of winning strategies [1]. In fact, from a game-theoretic point of view, these problems are usually reduced to a rather simple setting: two-player zero-sum sequential games [17].

However, when considering multi-player non-zero-sum concurrent games, the notion of a winning strategy does not seem to be the most appropriate analytical concept [2]. In such settings, we need richer game-theoretic solution concepts, such as Nash equilibrium [14]. However, at first sight, ATL$^*$ may not seem to be appropriate to reason about solution concepts such as Nash equilibrium, and in fact, it seems to be widely assumed that winning strategies cannot be used to study equilibria in multi-player non-zero-sum games.

Indeed, in the quest for a formalism expressive enough to capture game-theoretic equilibria, *many* temporal logics have been developed, including: Strategy Logic (SL [3, 11]); an extension of ATL with strategy contexts (ATL$_{sc}$ [10]); Quantified CTL$^*$ (QCTL$^*$ [9]); and Coordination Logic (CL [4]). All these logical formalisms, however, have undecidable satisfiability and non-elementary model checking problems in the general case. These undesirable properties present substantial challenges from a computational point of view when applying these temporal logics for synthesis or verification purposes. In fact, only very few practical model checking tools support logical specifications with formalisms more powerful than ATL$^*$.

For some of the reasons given above, a number of studies have been conducted, partly because better results can be obtained in restricted scenarios, for instance, when considering sequential (also called turn-based) games or two-player games. For example, it is known that with respect to two-player sequential games, Nash equilibria can be expressed in a fragment of SL that is as expressive as ATL$^*$; see [3] for further details. As a consequence, reasoning about Nash equilibria in this class of games can be done in 2EXPTIME. With respect to multi-player sequential games, Nash equilibria can be expressed in both ATL$^*_{sc}$ and Quantified CTL$^*$ while keeping the logics decidable [9]. Model checking is still non-elementary. In addition, until recently, the simplest fragment of SL known to be powerful enough to express Nash equilibria was the Boolean goal fragment [12], which may require, so-called, non-behavioural strategies [12] to realise a profile of strategies.[1] Such a fragment is not known to have a decidable satisfiability problem [11], but can be used to model check Nash equilibrium properties in 2EXPTIME given the characterisation of Nash equilibria in SL. These are all very interesting results, which suggest that further work should be done.

In this paper, we will show that under a suitable set of necessary and sufficient conditions, *ATL$^*$ can be used to reason about the existence of various kinds of equilibria* in multi-player non-zero-sum concurrent games. In particular, we will be interested in how to:

1. verify if a given game has a Nash equilibrium (NE model checking);
2. check if there is a game where a desired Nash equilibrium exists (NE satisfiability);
3. synthesise a profile of strategies that realises a particular Nash equilibrium;
4. express Nash equilibria in a number of types of games.

Let us now explain how ATL$^*$ can be used to address the above questions. We say that a *property* can be *expressed* in a given logic $L$ if there is a formula $\phi$ of $L$ such that $\phi$ is satisfied in all and only the models having such a property. Moreover, we say that a *problem* can be *characterised* in a given logic $L$ if every instance of the problem can be specified in $L$. Letting $P$ be the property of having a Nash equilibrium (formally defined later on), we show that using ATL$^*$ one can both express $P$ in multi-player sequential games and characterise the NE model checking and NE satisfiability problems for these games. In fact, we can do so

[1] It was recently discovered how to express Nash equilibria in a fragment of SL that is simpler than the Boolean goal fragment and which can be given an interpretation in terms of behavioural strategies.

using a logic that is strictly weaker than ATL*. Moreover, we show that using ATL* one can characterise the NE satisfiability problem for multi-player concurrent games and the NE model checking problem for a subclass of games that we call Moore with deterministic past.

An interesting aspect of our results with respect to ATL* is that the class of Moore games with deterministic past is complete for the whole family of multi-player concurrent games in the following sense: every formula $\phi$ of ATL* is satisfiable if and only if it is satisfied by some model representing a game that is Moore with deterministic past. Then, Moore games with deterministic past can be seen as canonical within the class of multi-player concurrent games and therefore be used as an underlying model with respect to which define games and synthesise profiles of strategies. Using these results we can, in the end, perform a computational analysis of Nash equilibrium properties for various types of games.

Our results build on a novel semantic characterisation of Nash equilibria which combines the concepts of *winning strategies* and *memoryful reasoning*. Informally, in this paper we use techniques from repeated game theory (more specifically, punishment strategies as used in the literature about the Folk Nash theorems for repeated games [5, 14]) and combine them with recent developments in the study of strategic reasoning in logic and computer science: a memoryful extension of ATL* called mATL* [13]. Our results also indicate why an ATL* specification that allows one to reason about the existence of Nash equilibrium may be so hard to find: it can be obtained via a non-elementary reduction from mATL*, which was recently shown to be as expressive as ATL*, a result that partly builds on quite sophisticated automata-theoretic techniques developed for a memoryful extension of CTL* [7].

Based on what is currently known about some of the computational properties of mATL*, (for instance, 2EXPTIME satisfiability, synthesis, and model checking problems as well as a non-elementary reduction to ATL*), we are able to obtain a number of subsequent results. We also studied the two important special cases of two-player games and sequential games. In the latter setting, we show that the existence of Nash equilibria can be expressed in m⁻ATL* [13], a (memoryful) temporal logic that is known to be strictly weaker than ATL*, which improves previous results in the literature [3, 9] and shows that concurrent behaviour, even in the two-player case, appears to complicate matters considerably.

Moreover, given the translation from mATL* to ATL*, we show that Nash equilibrium can be realised with respect to finite-memory strategies [15], which in turn also answers a question about the nature of Nash equilibrium strategy profiles and improves other results in the literature [12]. Finally, we extend all of our main findings to strong Nash equilibrium, a game-theoretic solution concept designed to deal with coalitions of players in a game – a very natural situation in concurrent and multi-agent scenarios. Interestingly, the results for this more complex game-theoretic setting are obtained using virtually the same techniques as in the case for Nash equilibria, where only single-player deviations are considered.

Finally, the table below summarises some of the results mentioned before, *i.e.*, the logics that can be used to study NE satisfiability and NE model checking for each type of games.

| | 2P-SG | 2P-CG | MP-SG | MP-CG |
|---|---|---|---|---|
| NE satisfiability | m⁻ATL* | ATL* | m⁻ATL* | ATL* |
| NE model checking | m⁻ATL* | SL | m⁻ATL* | SL |

**Notation:** 2P (2-player), MP (multi-player), SG (sequential game), CG (concurrent game).

**Structure of the paper.** Section 2 introduces the models, logics, and games that we will consider throughout the paper. Section 3 presents the semantic characterisation of equilibria in terms of punishment strategies, and Section 4 defines Moore games with deterministic past. Section 5 shows how to construct an mATL$^*$ specification of the semantic characterisation of equilibria given before in order to solve NE satisfiability, and Section 6 studies the special cases of sequential games and two-player games. Section 8 concludes the paper.

## 2 Preliminaries

**Sets.** Given any set $S = \{s, q, r, \ldots\}$, we use $S^*$, $S^\omega$, and $S^+$ for, respectively, the sets of finite, infinite, and non-empty finite sequences of elements in $S$. If $w_1 = s's'' \ldots s^k \in S^*$ and $w_2$ is any other (finite or infinite) sequence, we write $w_1 w_2$ for their concatenation $s's'' \ldots s^k w_2$. For $Q \subseteq S$, we write $S_{-Q}$ for $S \setminus Q$ and $S_{-i}$ if $Q = \{i\}$. We extend this notation to tuples $u = (s_1, \ldots, s_k, \ldots, s_n)$ in $S_1 \times \ldots \times S_n$, and write $u_{-k}$ for $(s_1, \ldots, s_{k-1}, s_{k+1}, \ldots, s_n)$, and similarly for sets of elements, that is, by $u_{-Q}$ we mean $u$ without each $s_k$, for $k \in Q$. Given a sequence $w$, we write $w[t]$ for the element in position $t+1$ in the sequence; for instance, $w[0]$ is the first element of $w$. We also write $w[l \ldots m]$ for the sequence $w[l] \ldots w[m]$, and $w[l \ldots m)$ for $w[l] \ldots w[m-1]$; if $m = 0$, we let $w[l \ldots m)$ be the empty sequence, denoted $\epsilon$.

**Games.** Let $\mathrm{Ag} = \{1, \ldots, n\}$ be a set of *players* and St a set of *states*. For each player $i \in \mathrm{Ag}$ we have a set of actions $\mathrm{Ac}_i$ and with every state $s$ and player $i$ we associate a set $\mathrm{Ac}_i(s) \subseteq \mathrm{Ac}_i$ of *actions* that $i$ can perform at $s$. We write Ac for $\bigcup_{i \in \mathrm{Ag}} \mathrm{Ac}_i$ and assume that $\bigcup_{i \in \mathrm{Ag}} \mathrm{Ac}_i$ is a partition of Ac. We call a profile of actions $(a_1, \ldots, a_n) \in \mathrm{Ac}_1 \times \cdots \times \mathrm{Ac}_n$ a *direction*, and denote it by $d$. We let D be the set of directions – also called *decisions* – with respect to Ac, and write $d_i$ for the $a_i$ of $d$ that is in $\mathrm{Ac}_i$. Furthermore, we have a (deterministic) transition function $\delta \colon \mathrm{St} \times \mathrm{Ac}_1 \times \cdots \times \mathrm{Ac}_n \to \mathrm{St}$, which indicates the system transitions when $d = (a_1, \ldots, a_n)$ is performed at a state $s$. A state $s'$ is *accessible* from another state $s$ whenever there is some $d = (a_1, \ldots, a_n)$ such that $\delta(s, a_1, \ldots, a_n) = s'$. A *run* is an infinite sequence $\rho = s_0 s_1 s_2 \ldots$ such that for every $t \geq 0$ we have that $s_{t+1}$ is accessible from $s_t$. The set of runs is denoted by $R$. By a *(finite) history* we mean a finite sequence $\pi = s_0 s_1 s_2 \ldots s_k$ of accessible states. By *prefix*$(\rho)$ we denote the set of finite prefixes of $\rho$, *i.e.*, *prefix*$(\rho) = \{\pi \in \mathrm{St}^* : \rho = \pi\rho' \text{ for some } \rho' \in \mathrm{St}^\omega\}$. Given $\pi \in \mathrm{St}^+$, by *last*$(\pi)$ we denote the last state in $\pi$, *i.e.*, if $\pi = \pi' s$, then *last*$(\pi) = s$. By $s^0$ we denote the *initial state*.

A *strategy* for a player $i$ is a function $f_i \colon \mathrm{St}^+ \to \mathrm{Ac}_i$ such that $f_i(\pi s) \in \mathrm{Ac}_i(s)$ for every $\pi \in \mathrm{St}^*$ and $s \in \mathrm{St}$. That is, a strategy for a player $i$ specifies for every finite history $\pi$ an action available to $i$ in *last*$(\pi)$. The set of strategies for player $i$ is denoted by $F_i$. A *strategy profile* is a tuple $\vec{f} = (f_1, \ldots, f_n)$ in $F_1 \times \cdots \times F_n$. Observe that given a state $s$ and a transition function $\delta \colon \mathrm{St} \times \mathrm{Ac}_1 \times \cdots \times \mathrm{Ac}_n \to \mathrm{St}$, each strategy profile $\vec{f}$ defines a unique run $\rho$ where $\rho[0] = s$ and $\rho[t+1] = \delta(\rho[t], f_1(\rho[0 \ldots t]), \ldots, f_n(\rho[0 \ldots t]))$, for all $t \geq 0$. We write $\rho(\vec{f}(s))$ for such a run, and simply $\rho(\vec{f})$ if $s = s^0$. Furthermore, each player $i$ is assumed to have an associated dichotomous preference relation over runs, which is modelled as a subset $\Gamma_i$ of the set of runs $R$. Intuitively, a player $i$ strictly prefers all runs in $\Gamma_i$ to those that are not in $\Gamma_i$ and is indifferent otherwise. Thus, $\Gamma_i$ represents the *goal* or *objective* of player $i$. We also write $\rho \succsim_i \rho'$ to indicate that player $i$ weakly prefers run $\rho$ to $\rho'$ and $\rho \succ_i \rho'$ for player $i$ strictly preferring $\rho$ to $\rho'$, *i.e.*, if $\rho \succsim_i \rho'$ but not $\rho' \succsim_i \rho$.

Now we are in a position to define the concept of a *Nash equilibrium strategy profile* as a strategy profile $\vec{f} = (f_1, \ldots, f_n)$ such that for all players $i$ and all strategies $g_i$ in $F_i$,

$$\rho(\vec{f}) \succsim_i \rho(\vec{f}_{-i}, g_i).$$

We say that a run $\rho$ is *sustained by a Nash equilibrium strategy profile* if there is some Nash equilibrium strategy profile $\vec{f}$ with $\rho = \rho(\vec{f})$. Then, we know that

$$\rho(\vec{f}_{-i}, g_i) \in \Gamma_i \ \text{ implies } \ \rho(\vec{f}) \in \Gamma_i.$$

A game is played by each player $i$ selecting a strategy $f_i$ with the aim that the induced run $\rho(\vec{f})$ belongs to its goal/objective set $\Gamma_i$. If $\rho(\vec{f}) \in \Gamma_i$ we say that $i$ has its goal satisfied or achieves its objective. Otherwise, we say that $i$ does not have its goal satisfied.

## Memoryful Alternating Temporal Logics

Memoryful ATL* (mATL* [13]) is an extension of ATL* that allows for memoryful reasoning. At the syntactic level, mATL* simply adds a propositional variable *present* to the standard ATL* language. More specifically, given a set of atomic propositions AP and a set of agents Ag, the language of mATL* formulae is given by the following grammar:

$$\phi \quad ::= \quad \mathit{present} \mid p \mid \neg\phi \mid \phi \vee \phi \mid \ \mathbf{X}\,\phi \mid \phi\,\mathbf{U}\,\phi \mid \ \langle\!\langle C \rangle\!\rangle\,\phi$$

such that $p \in \text{AP}$ and $C \subseteq \text{Ag}$, and the formulae *present*, $\mathbf{X}$, and $\mathbf{U}$ are in the scope of $\langle\!\langle C \rangle\!\rangle$. We use the following abbreviations: we write $\top$ for $p \vee \neg p$, $\bot$ for $\neg\top$, $\mathbf{F}\,\phi$ for $\top\,\mathbf{U}\,\phi$, $\mathbf{G}\,\phi$ for $\neg\,\mathbf{F}\,\neg\phi$, $\mathbf{E}\,\phi$ for $\langle\!\langle \text{Ag} \rangle\!\rangle\,\phi$, $\mathbf{A}\,\phi$ for $\langle\!\langle \emptyset \rangle\!\rangle\,\phi$, and $[\![C]\!]\,\phi$ for $\neg\,\langle\!\langle C \rangle\!\rangle\,\neg\phi$; we also use the conventional abbreviations for the other propositional logic operators. We write $\phi \in \mathcal{L}(\text{AP}, \text{Ag})$ if $\phi$ is an mATL* formula in this language. When either AP or Ag, or both, are known, we may omit them. With $\text{AP}' \subseteq \text{AP}$, we may write $\phi|_{\text{AP}'}$ if $\phi \in \mathcal{L}(\text{AP}', \text{Ag})$ for some set of agents Ag.

The denotation of mATL* formulae is given by *concurrent game structures.* We let a tuple $M = (\text{AP}, \text{Ag}, \text{Ac}, \text{St}, s^0, \lambda, \delta)$ be a concurrent game structure (CGS), where $\lambda : \text{St} \to 2^{\text{AP}}$ is a labelling function, and all other components of $M$ are as defined before. The *size of* $M$ is defined to be $|\text{St}| \times |\text{Ac}|^{|\text{Ag}|}$. Moreover, in particular, we write $R^*_s$ and $R^\omega_s$ for, respectively, the finite and infinite runs of $M$ that start at state $s$. We simply write $R^*$ and $R^\omega$ if $s = s^0$. Moreover, we will write $\vec{f_C}$ for $(f_i, \ldots, f_k)$, with $C = \{i, \ldots, k\} \subseteq \text{Ag}$, that is, a joint strategy for the players in $C$. Similarly, we will write $\vec{g}_{-C}$ for a joint strategy for the players in $\text{Ag}_{-C}$. For simplicity, we will assume that all strategies are defined on all finite runs of $M$, and hence at all states. We define the set of $\vec{f_C}$*-runs from state* $s$ to be $\{\rho' \in R^\omega_s : \rho' = \rho(\vec{f_C}(s), \vec{g}_{-C}(s))$ for some joint strategy $\vec{g}_{-C}$ for $\text{Ag}_{-C}\}$.

We can now define the semantics of mATL* formulae based on the rules given below. Let $M$ be a concurrent game structure, $\rho \in R^\omega$ be an infinite run, $\pi \in R^*$ be a finite run, and $t \in \mathbb{N}$ be a temporal index. The semantics of mATL* is defined by the following rules:

| | | |
|---|---|---|
| $\rho, t, \pi \models_M \mathit{present}$ | iff | $\rho[0 \ldots t] = \pi$. |
| $\rho, t, \pi \models_M p$ | iff | $p \in \lambda(\rho[t])$. |
| $\rho, t, \pi \models_M \neg\phi$ | iff | $\rho, t, \pi \models_M \phi$ does not hold. |
| $\rho, t, \pi \models_M \phi \vee \phi'$ | iff | $\rho, t, \pi \models_M \phi$ or $\rho, t, \pi \models_M \phi'$. |
| $\rho, t, \pi \models_M \mathbf{X}\,\phi$ | iff | $\rho, t+1, \pi \models_M \phi$. |
| $\rho, t, \pi \models_M \phi\,\mathbf{U}\,\phi'$ | iff | $\rho, t', \pi \models_M \phi'$ , for some $t' \geq t$, and $\rho, k, \pi \models_M \phi$ , for all $t \leq k < t'$. |
| $\rho, t, \pi \models_M \langle\!\langle C \rangle\!\rangle\,\phi$ | iff | there is some $\vec{f_C}$ such that for all $\vec{f_C}$-runs $\rho'$ from state $\rho[t]$, it is the case that $\rho[0 \ldots t)\rho', 0, \rho[0 \ldots t] \models_M \phi$ holds. |

We say that $M$ is a *model* of $\phi$ (denoted $M \models \phi$) if $\rho, 0, s^0 \models_M \phi$ for some $\rho \in R^\omega$. We also say that $\phi$ is *satisfiable* if $M \models \phi$ for some CGS $M$. Moreover, we say that $\phi$ is *equivalent* to $\phi'$ if $M \models \phi \iff M \models \phi'$ for all $M$. Finally, the *size of* $\phi$ is its number of subformulae.

The interpretation of the tense and boolean operators is, essentially, as for LTL. The difference is with respect to the *present* proposition and the *agent/strategy quantifier* $\langle\!\langle C \rangle\!\rangle$.

On the one hand, the atomic proposition *present* holds whenever the history of the current run under consideration (*i.e.*, the finite run $\rho[0 \dots t]$) is the same as the history of the run leading to the state where *present* last was true – that is, $\pi$. On the other hand, the agent quantifier does three important things: firstly, it considers the runs $\rho'$ that are consistent with $\vec{f}_C$, but adds to them the history of play seen so far – thus, obtaining the run $\rho[0 \dots t)\rho'$; secondly, it resets the evaluation of temporal formulae by letting the temporal index $t$ be 0; and, thirdly, it resets the history with respect to which the atomic proposition *present* will now hold – then, letting $\pi$ be $\rho[0 \dots t]$. A few examples are in order (also see, *e.g.*, [13]).

**Examples.** Because the quantifiers of mATL* reset the temporal index $t$, CTL* formulae, such as $\mathbf{E}\,\mathbf{F}(p \implies \mathbf{A}\,\mathbf{G}\,q)$, cannot be interpreted in the same way in mATL*: the subformula $\mathbf{G}\,q$ will be evaluated from $t = 0$ regardless of the value of $t$ when $p$ holds. There is a standard way to remedy this problem. Any subformula starting with a path quantifier can be evaluated "in the usual CTL* sense" in the following way: $M \models_{\text{CTL}^*} \mathbf{A}\,\phi$ if and only if $M \models \mathbf{A}\,\mathbf{F}(\mathit{present} \wedge \phi)$; and similarly for $\mathbf{E}$. Therefore, in order to be interpreted as a CTL* expression, the formula above can be rewritten as $\mathbf{E}\,\mathbf{F}(p \implies \mathbf{A}\,\mathbf{F}(\mathit{present} \wedge \mathbf{G}\,q))$.

But, of course, mATL* is interesting because it can express, in a relatively simple way, properties that refer both to the past and to the future within a strategic environment. For instance, suppose an agent $i$ has two objectives: satisfy LTL goal formula $\gamma_1$ and if $i$ sees proposition $p$ satisfied, then try to satisfy LTL goal formula $\gamma_2$ too. What is important is that both $\gamma_1$ and $\gamma_2$ are "goals" that were set from the beginning of the computation, not from the point where, potentially, proposition $p$ is encountered. Expressing that $i$ can ensure that its goals are satisfied can be done in the following way: $\langle\!\langle i \rangle\!\rangle(\gamma_1 \wedge \mathbf{G}(p \implies \langle\!\langle i \rangle\!\rangle\,\gamma_2))$. Then, player $i$ must be more careful when playing the game. It has to play in such a way that if $p$ is seen while trying to satisfy $\gamma_1$, then it is in a position to satisfy $\gamma_2$ too. Seen as a formula in ATL* instead, the situation is rather different: while $\gamma_1$ has to hold from the beginning of the computation, $\gamma_2$ has to hold from the moment proposition $p$ is seen.

**Logical and Computational Properties.** Using the abbreviations $\mathbf{E}$ and $\mathbf{A}$ and the atomic proposition *present* as explained in the first example, it can be shown that CTL* is a sublogic of mATL*. Thus, CTL and LTL are also sublogics of mATL*. Moreover, none of these logics is as powerful as mATL*, as might be expected. What is surprising is that, in fact, ATL* is as expressive as mATL*. However, the translation from mATL* to ATL* formulae is non-elementary [13]. An interesting case is the fragment of mATL* without the *present* proposition. Such a fragment, called m$^-$ATL* is not as expressive as ATL*, and incomparable with CTL/CTL*. The proof is simple and has to do with the fact that *present* is needed for mATL* to be able to capture "memoryless" properties such as those given by CTL* (cf. the first example above and [13]). Letting $L_1 < L_2$ mean that logic $L_1$ is strictly less expressive than logic $L_2$, and $L_1 \equiv L_2$ that $L_1$ is as expressive as $L_2$, we have the following:

$$\text{m}^-\text{ATL} < \text{mATL}^* \equiv \text{ATL}^* < \{\text{SL, CL, ATL}_{sc}, \text{QCTL}^*\}.$$

With respect to the complexity of mATL*, it is known that it has model checking, satisfiability and synthesis problems in 2EXPTIME, thus no harder than ATL*. These results sharply contrast with those for other logics such as Strategy Logic, ATL with strategy contexts, or Coordination Logic, in which satisfiability and synthesis are *undecidable* and model checking is *non-elementary* [4, 9, 11]. The complexity properties of logics for strategic reasoning present a substantial challenge for practical purposes. In fact, a very simple fragment of Strategy Logic – where Nash equilibrium can be specified – may require non-behavioural strategies as models [12], which is not the case in ATL* and mATL*.

### Games with Temporal Logic Goals

Finally, we are in a position to define a game with temporal logic goals. Given a concurrent game structure $M = (\mathrm{AP}, \mathrm{Ag}, \mathrm{Ac}, \mathrm{St}, s^0, \lambda, \delta)$ and a set of temporal logic formulae $(\gamma_i)_{i\in\mathrm{Ag}}$, a tuple $G = (M, (\gamma_i)_{i\in\mathrm{Ag}})$ is a *game with temporal logic goals*. We say that $G$ is a game with LTL goals if each $\gamma_i \in \mathcal{L}(\mathrm{AP})$ is an LTL formula, and simply call $G$ a game. The idea is that each $\gamma_i \in \mathcal{L}(\mathrm{AP})$ is a formula that succinctly represents the goal of player $i$, that is, $\gamma_i$ is a logical representation of $\Gamma_i$. Then, we have that a run $\rho$ satisfies goal $\gamma_i$ if and only if $\rho \in \Gamma_i$.

## 3 From Winning Strategies to Nash Equilibria

In order to characterise the existence of Nash equilibrium in terms of winning strategies, we will introduce the concept of *punishment* strategies. These are a type of winning strategies that a group of players may want to use against a player $j$ who wants to refrain from following a given sequence of actions – a situation that is commonly known as a deviation by $j$.

A finite history $\pi$ is a *deviation* from a run $\rho$ if $\pi = \pi' s$ for some state $s \in \mathrm{St}$ such that $\pi' \in \mathit{prefix}(\rho)$ and $\pi \notin \mathit{prefix}(\rho)$. We also use $\mathit{dev}(\rho)$ to refer to the set of deviations from $\rho$. A finite history $\pi$ is a *deviation from a set* $R$ if $\pi \in \mathit{dev}(\rho)$ for some $\rho \in R$ and $\pi \in \mathit{prefix}(\rho')$ for no $\rho' \in R$. The set of deviations from $R$ is denoted by $\mathit{dev}(R)$.

If a run $\rho$ is intended to be sustained by a Nash equilibrium strategy profile $\vec{f}$, punishment may be successful deterring agents from adopting deviating strategies – strategies that do not comply with $\rho$. However, if a deviation is observed, it is generally not necessary to punish all players but rather only those that the deviation can be attributed to in order to prevent such undesirable behaviour. We thus introduce the following formal concept of *attributability*.

▶ **Definition 1** (Attributability). Let $\rho$ be a run and $\pi \in \mathit{dev}(\rho)$ a deviation from $\rho$. Then, the deviation $\pi$ is *attributable to a set $C$ of players* if there are profiles $\vec{f} = (f_1, \dots, f_n)$ and $\vec{g} = (g_1, \dots, g_n)$ such that $\rho(\vec{f}) = \rho$ and $\pi \in \mathit{prefix}(\rho(\vec{f}_{-C}, \vec{g}_C))$. We also say that $\pi$ is attributable to $j$ for $\pi$ being attributable to $C = \{j\}$. A deviation $\pi$ is said to be *individual* if $\pi$ is attributable to some player $j$. Moreover, a deviation $\pi$ from $\rho$ is said to be *uniquely attributable to $C$* if $C$ is the only set that the deviation $\pi$ can be attributed to. A deviation that is uniquely attributable to a player $j$ (a set $C$) is called a *$j$-deviation* (a *$C$-deviation*).

The necessity of the concept of attributability is rather clear: in order to define punishable players and punishment strategies the least that one needs to know is who should be punished once a deviation arises. Some equally important information in order to define punishment strategies is the *history* of play so far, since players' goals are defined with respect to the beginning of the game. Then, at every stage of the game one must be able to "remember" the history of play so far – that is, we need some *memoryful* power within the game. In order to formalise the above notions and, in particular, to make precise how we will deal with memoryful reasoning later on, let us introduce some notations first.

Given a finite history $\pi$, each strategy profile $\vec{f}$ induces a unique run $\rho(\vec{f}, \pi) = s^0 s^1 s^2 \dots$ defined inductively as follows, with $t \geq 0$,

$$s^t = \begin{cases} s^0 & \text{if } t = 0 \text{ and } \pi = \epsilon, \\ \delta(\mathit{last}(\pi), f_1(\pi), \dots, f_n(\pi)) & \text{if } t = 0 \text{ and } \pi \neq \epsilon, \\ \delta(s^{t-1}, f_1(s^0 \dots s^{t-1}), \dots, f_n(s^0 \dots s^{t-1})) & \text{otherwise.} \end{cases}$$

We omit $\epsilon$ in $\rho(\vec{f}, \epsilon)$ and simply write $\rho(\vec{f})$. Then, $\pi$ can be seen as the history of play with which a strategy or a profile of strategies can be made to agree, so that it is taken into account when playing beyond $\pi$. We are now in a position to formulate the central concept of this section, namely that of a player, or a set of players, being *punishable*.

▶ **Definition 2** (Punishable players, punishment strategies). A set $C$ of players is *punishable at run* $\rho$ if there is a profile $\vec{f}^C = (f_1^C, \dots, f_n^C)$ – also called a profile of *punishment strategies against* $C$ – such that for every profile of strategies $\vec{g}_C$ of $C$ and every deviation $\pi$ from $\rho$ that is uniquely attributable to $C$ we have, for some $j \in C$,

$$\pi\rho((\vec{f}^C_{-C}, \vec{g}_C), \pi) \notin \Gamma_j.$$

One thing to observe in this definition is that a profile of punishment strategies may punish different players at different $C$-deviations. Moreover, as we will see, for a set $C$ of players to be punishable, it is sufficient to punish only one player in $C$ at each $C$-deviation. More importantly, Definition 2 shows the strong link between two important components: firstly, the need to consider/remember $\pi$ and, secondly, the existence of a winning strategy $\vec{f}^C_{-C}$ for the set of players not in $C$ against the goal of at least one of the players in $C$.

Thus, a careful analysis of the definition of punishability we proposed shows the following facts. That if one is interested in that every player $j$ whose goal is not satisfied by $\rho$ must be punishable, then one has to guarantee, in particular, that at *every* stage of the game those in the set of players in $\mathrm{Ag}_{-j}$ (i) must have a winning strategy against the goal of player $j$, and such a winning strategy (ii) must agree with $\rho$ up to the time point when $j$ deviates. These two conditions for a player $j$ to be punishable are key to answering the next question:

*When can a player $j$ be guaranteed to be punishable*
*so that a desired Nash equilibrium can be rationally sustained?*

Formally, punishability with respect to a desired run $\rho$ can be fully characterised in terms of a number of winning strategies against the goal of $j$, as shown by the following lemma.

▶ **Lemma 3** (Punishability – semantic characterisation). *Player $j$ is* punishable at run $\rho$ *if and only if for every deviation $\pi$ from $\rho$ that is uniquely attributable to $j$ there is a profile $\vec{f}^\pi = (f_1^\pi, \dots, f_n^\pi)$ such that for every strategy $g_j$ of $j$ we have that*

$$\pi\,\rho((\vec{f}^\pi_{-j}, g_j), \pi) \notin \Gamma_j.$$

The above lemma supports the claim that punishability can be understood as a simple combination of *winning strategies* (given by $\vec{f}^\pi_{-j}$ to achieve $\neg\gamma_j$) with *memory* (given by $\pi$). Because the concept of punishability will be central to the characterisation of the existence of Nash equilibrium strategy profiles, the above claim about winning strategies and memoryful reasoning can be naturally extended to the more complex concept of Nash equilibria.

Indeed, the manipulation of the three main concepts introduced in this section, namely *deviations*, *attributability*, and *punishment*, can be used to provide a semantic characterisation of the fundamental concept of Nash equilibrium. More importantly, they can be used to show that Nash equilibria can be characterised by the existence of punishment strategies, which, by definition, are winning strategies against players to which deviations can be attributed. Formally, the following result can be shown.

▶ **Lemma 4** (Nash equilibrium – semantic characterisation). *Let $\rho$ be a run and assume that all individual deviations from $\rho$ are uniquely attributable. Then, $\rho$ is sustained by a Nash equilibrium strategy profile if and only if all players $j$ with $\rho \notin \Gamma_j$ are punishable at $\rho$.*

Lemma 4 shows that, provided that every $j$-deviation can be uniquely attributable to player $j$, the following condition is both necessary and sufficient for the existence of a Nash equilibrium $\rho$: that *every player $j$ whose goal is not satisfied by $\rho$ must be punishable.*

The *necessary* and *sufficient* conditions for the existence of Nash equilibria are both formulated in terms of our notion of punishability (Lemma 4). To be able to punish a deviating player, however, a coalition also has to keep track of the run that leads to the deviation. This idea could informally be summarised as follows:

**Nash equilibrium = Punishability + Memory**

The concept of punishability is closely related to that of a winning strategy. Thus, the above results prepare the ground for the characterisation of Nash equilibrium in temporal logics like mATL* that can reason about both *winning strategies* and *memoryfulness.*

## 4 Model Transformations

In order to logically describe the semantic characterisation of Nash equilibria presented in Section 3, we first define two structure-preserving transformations on CGSs, which can be used to map any CGS into another CGS in a desired canonical form. The first map is used to transform a CGS into a CGS *with deterministic past.* The second map is used to transform a CGS $M$ into a CGS $M'$ where all information about players' choices in $M$ is explicitly given via labels in the states of $M'$. This transformation is similar to the well known translation used to obtain Moore from Mealy machines. For this reason, we call such a translation a *Moore transformation*, and the resulting CGS $M'$ a *Moore* CGS.

Let us now formally define the class of CGS with deterministic past. A CGS $M$ is said to be a CGS with deterministic past if whenever $\delta(s', d') = s$ and $\delta(s'', d'') = s$ then $d' = d''$. The first map above mentioned, namely $\mathbf{detp} : \mathcal{M} \to \mathcal{M}$, called a *deterministic-past map*, transforms a CGS $M$ into a CGS $M'$ which can be shown to be a CGS *with deterministic past.* The map $\mathbf{detp}$ is defined as follows. Given $M \in \mathcal{M}$, where $M = (\mathrm{AP}, \mathrm{Ag}, \mathrm{Ac}, \mathrm{St}, s^0, \lambda, \delta)$, the structure $\mathbf{detp}(M)$ is defined to be the tuple $(\mathrm{AP}, \mathrm{Ag}, \mathrm{Ac}, \mathrm{St}', q^0, \lambda', \delta')$ such that

- $\mathrm{St}' = \{s_d \ : \ s \in \mathrm{St} \ \& \ d \in \mathrm{D}\}$, and $q^0 = s^0_d$ for any $d \in \mathrm{D}$;
- $\lambda'(s_d) = \lambda(s)$, for all $d \in \mathrm{D}$; and $\delta'(s_d, d') = s'_{d'}$ if and only if $\delta(s, d') = s'$, for all $d \in \mathrm{D}$;

where $s_d$ is an abbreviation for the pair $(s, d)$ and D is the set of directions/decisions.

▶ **Lemma 5** (Deterministic past). *Let $M$ be a CGS and* $\mathbf{detp}$ *be a deterministic-past map. Then* $\mathbf{detp}(M)$ *is a CGS with deterministic past of size $\mathcal{O}(|\mathrm{D}| \times |M|)$.*

The second map, namely $\mathbf{moore} : \mathcal{M} \to \mathcal{M}$, for consistency called a *Moore* map, is defined as follows. Given $M \in \mathcal{M}$, with $M = (\mathrm{AP}, \mathrm{Ag}, \mathrm{Ac}, \mathrm{St}, s^0, \lambda, \delta)$, the structure $\mathbf{moore}(M)$, called a *Moore* CGS, is defined to be the tuple $(\mathrm{AP}', \mathrm{Ag}, \mathrm{Ac}, \mathrm{St}, s^0, \lambda', \delta)$ such that

- $\mathrm{AP}' = \mathrm{AP} \uplus \mathrm{APi} \uplus \mathrm{APo}$, where
  $\mathrm{APi} = \{\check{a}^x_i : i \in \mathrm{Ag} \ \& \ x \in \mathrm{Ac}_i\}$ and $\mathrm{APo} = \{\hat{a}^x_i : i \in \mathrm{Ag} \ \& \ x \in \mathrm{Ac}_i\}$;
- if $\delta(s, d) = s'$, with $d = (x_1, \dots, y_n)$, then there are propositions $\check{a}^x_1, \dots, \check{a}^y_n \in \mathrm{APi}$ and $\hat{a}^x_1, \dots, \hat{a}^y_n \in \mathrm{APo}$, such that $\{\check{a}^x_1, \dots, \check{a}^y_n\} \subseteq \lambda'(s') \cap \mathrm{APi}$ and $\{\hat{a}^x_1, \dots, \hat{a}^y_n\} \subseteq \lambda'(s) \cap \mathrm{APo}$.

▶ **Lemma 6** (Moore CGS). *Let $M$ be a CGS and* $\mathbf{moore}$ *be a Moore map. Then* $\mathbf{moore}(M)$ *is a Moore CGS of size $\mathcal{O}(|M|)$.*

Composing these two maps one can obtain, from a CGS $M = (\mathrm{AP}, \mathrm{Ag}, \mathrm{Ac}, \mathrm{St}, s^0, \lambda, \delta)$, a CGS $M'' = (\mathrm{AP} \uplus \mathrm{APi} \uplus \mathrm{APo}, \mathrm{Ag}, \mathrm{Ac}, \mathrm{St}', q^0, \lambda'', \delta')$ of size polynomial in $|M|$ which preserves all mATL* properties that can be defined by formulae in $\mathcal{L}(\mathrm{AP})$, provided that both $\mathrm{AP} \cap \mathrm{APi} = \emptyset$ and $\mathrm{AP} \cap \mathrm{APo} = \emptyset$ – which is, by definition, ensured by both maps.

▶ **Lemma 7.** *Let $M = (\mathrm{AP}, \mathrm{Ag}, \mathrm{Ac}, \mathrm{St}, s^0, \lambda, \delta)$ be a CGS and $M' = \mathbf{moore}(\mathbf{detp}(M))$. Then, $M'$ is a Moore CGS with deterministic past of size $\mathcal{O}(|\mathrm{D}| \times |M|)$ which is such that $M \models \phi$ if and only if $M'|_{\mathrm{AP}} \models \phi$, for all formulae $\phi \in \mathcal{L}(\mathrm{AP})$.*

Informally, Lemma 7 relies on the fact that, because of the definition of $\delta'$ in **detp**, the branching structure of every $s$ of $M$ is preserved in each $s_d \in \{s_d : d \in \mathrm{D}\}$ of $M'|_{\mathrm{AP}}$. In particular, note that, for any given $M \in \mathcal{M}$, the map $\mathbf{detp}(M)$ simply makes up to $|\mathrm{D}|$ (bisimilar) copies of each state of $M$ where all players' choices are preserved in each $s_d$. Moreover, for any given $M \in \mathcal{M}$, the structures $M$ and $\mathbf{moore}(M)|_{\mathrm{AP}}$ are isomorphic. Also, since both **detp** and **moore** are total maps on CGSs, it can easily be shown from the above lemmas the following result with respect to the properties that can be expressed in mATL$^*$.

▶ **Corollary 8** (Model completeness). *For every $\phi \in \mathcal{L}(\mathrm{AP})$, it is the case that $\phi$ is satisfiable if and only if there is a Moore CGS with deterministic past model $M$ such that $M \models \phi$.*

## 5 Nash Equilibria in Memoryful Logical Form

We now show that, up to bisimilarity, the concepts introduced in Section 3 can be given a logical characterisation in mATL$^*$. Without much further introduction, we first present an mATL$^*$ formula that can be used to reason about Nash equilibria, and then describe how each part of the specification relates to the conditions given in Section 3. Some notation first.

Given a set of players $W \subseteq \mathrm{Ag}$, a set of actions $\mathrm{Ac} = \bigcup_{i \in \mathrm{Ag}} \mathrm{Ac}_i$ for such players, and a set of atomic propositions AP, we write $L$ for $\mathrm{Ag} \setminus W$, and $\mathrm{APi} = \bigcup_{i \in \mathrm{Ag}} \mathrm{APi}_i$ and $\mathrm{APo} = \bigcup_{i \in \mathrm{Ag}} \mathrm{APo}_i$ for two sets of atomic propositions constructed based on Ac as follows:

$$x \in \mathrm{Ac}_i \iff \check{a}_i^x \in \mathrm{APi}_i \iff \hat{a}_i^x \in \mathrm{APo}_i,$$

such that $\mathrm{AP} \uplus \mathrm{APi} \uplus \mathrm{APo}$ is a set we denote by $\mathrm{AP}'$. Then, the formula $\mathsf{NormForm}$, defined below, logically describes the labelling pattern found in the class of CGSs that satisfy the conditions to be a Moore CGS with deterministic past, just as we studied them in Section 4:

$$\begin{aligned}\mathsf{NormForm} = \mathbf{A}\,\mathbf{G} \textstyle\bigwedge_{i \in \mathrm{Ag}} \quad & \Big(\big(\textstyle\bigvee_{\check{a}_i^x \in \mathrm{APi}} \check{a}_i^x\big) \wedge \textstyle\bigwedge_{\check{a}_i^x \in \mathrm{APi}_i} \big(\check{a}_i^x \implies (\textstyle\bigwedge_{\check{a}_i^y \in \mathrm{APi}_i \setminus \{\check{a}_i^x\}} \neg \check{a}_i^y)\big) \wedge \\ & \textstyle\bigwedge_{\hat{a}_i^x \in \mathrm{APo}_i} \big(\hat{a}_i^x \iff \mathbf{E}\,\mathbf{F}(\mathit{present} \wedge \mathbf{X}\,\check{a}_i^x)\big)\Big).\end{aligned}$$

Then, the mATL$^*$ formula $\phi_{\mathrm{NE}}$ below is satisfiable iff there is a Moore with deterministic past CGS $M = (\mathrm{AP}, \mathrm{Ag}, \mathrm{Ac}, \mathrm{St}, s^0, \lambda, \delta)$ such that $G = (M, (\gamma_i)_{i \in \mathrm{Ag}})$ has a Nash equilibrium:

$$\phi_{\mathrm{NE}} = \left(\mathbf{E} \bigvee_{W \subseteq \mathrm{Ag}} \Big(\mathsf{NEGoal}(W) \wedge \mathsf{AllPunishable}(L)\Big)\right) \wedge \mathsf{NormForm}$$

such that

$$\mathsf{NEGoal}(W) = \bigwedge_{i \in W} \gamma_i \wedge \bigwedge_{j \in L} \neg \gamma_j$$

and

$$\mathsf{AllPunishable}(L) = \mathbf{G} \bigwedge_{d \in \mathrm{Ac}_1 \times \ldots \times \mathrm{Ac}_n} \Big(\mathsf{InRun}(d) \implies \bigwedge_{j \in L} \mathbf{A}\,\mathsf{Punishable}(j, d)\Big)$$

where

$$\mathsf{InRun}(d) = \bigwedge_{i \in \mathrm{Ag}} \mathbf{X}\,\check{a}_i^{d_i}$$

and

$$\mathsf{Punishable}(j,d) = \mathbf{F}\left(\mathit{present} \wedge \mathbf{X}\left(\mathsf{AttDev}(j,d) \Longrightarrow \mathsf{Punish(j)}\right)\right)$$

with

$$\mathsf{AttDev}(j,d) = (\bigwedge_{i \in \mathrm{Ag}_{-j}} \check{a}_i^{d_i}) \wedge \neg \check{a}_j^{d_j} \qquad \text{and} \qquad \mathsf{Punish}(j) = \langle\langle \mathrm{Ag}_{-j} \rangle\rangle \neg \gamma_j$$

Formula $\phi_{\mathrm{NE}}$ simply describes the conditions given by Lemmas 3 and 4 to characterise the existence of Nash equilibria within the canonical class of game models given by the family of Moore CGSs with deterministic past. Let us now describe each formula in more detail.

The first line of NormForm indicates that every state must be labelled by one, and only one, profile of players' choices in APi (deterministic past); the second line, indicates that every player's choice available at some state is recorded in such a state (Moore). Note that because in mATL* path quantifiers reset the present, the CTL formula $\mathbf{E}\,\mathbf{X}\,\check{a}_i^x$ must be written as $\mathbf{E}\,\mathbf{F}(\mathit{present} \wedge \mathbf{X}\,\check{a}_i^x)$ – a standard mechanism to interpret formulae in the present.

The rest of the specification deals with Lemmas 3 and 4. It first indicates that for some set of players $W$ (the "winners") there is a run that satisfies their goals, while the players not in $W$ (the "losers") do not. That the goals are satisfied or not is checked by NEGoal. The formula also indicates that all "losers" must be punishable, which is specified by AllPunishable. This captures Lemma 4 provided that all individual deviations are uniquely attributable.

To check that all players in $L$ are punishable in the same run and at every time point, we first check the profile of players' choices currently used ("$d$"). Because NormForm guarantees deterministic past, formula InRun will be necessarily satisfied by exactly one $d$, which is used to check, for every player in $L$, if it can be punished should it deviate. Since to consider a potential deviation one has to use a universal path quantifier, then the semantics of the logic dictates that the present is reset. This is the reason why the *present* formula must be used in Punishable. Finally, we need to check two conditions. On the one hand, that an individual deviation is uniquely attributable to $j$: formula AttDev does precisely that. On the other hand, that if this was the case, then player $j$ can be effectively punished: formula Punish does exactly that. Note that memory is used in a different way for the first time; formula $\langle\langle \mathrm{Ag}_{-j} \rangle\rangle$ resets the present again, but this time we do not use the *present* proposition. Instead, we consider (the negation of) player $j$'s goal from the beginning – *i.e.*, it is evaluated from the first time point, not from the present. Then, with respect to Lemma 3, formula $\langle\langle \mathrm{Ag}_{-j} \rangle\rangle$ represents $\vec{f}^{\pi}_{-j}$, while $\pi$ is being naturally captured by the semantics of the logic.

Formally, appealing to Lemmas 3 and 4, the following result can be shown:

► **Theorem 9.** *Let $(\gamma_i)_{i \in \mathrm{Ag}}$, with each $\gamma_i \in \mathcal{L}(\mathrm{AP})$, be the goals of a set of players $\mathrm{Ag}$ with action set $\mathrm{Ac} = \biguplus_{i \in \mathrm{Ag}} \mathrm{Ac}_i$. Then, the formula $\phi_{\mathrm{NE}}$ constructed from $(\mathrm{AP}, \mathrm{Ag}, \mathrm{Ac}, (\gamma_i)_{i \in \mathrm{Ag}})$ is satisfiable iff there is a Moore CGS with deterministic past $M = (\mathrm{AP}, \mathrm{Ag}, \mathrm{Ac}, \mathrm{St}, s^0, \lambda, \delta)$ such that the game $G = (M, (\gamma_i)_{i \in \mathrm{Ag}})$ has a Nash equilibrium.*

Because ATL* is as expressive as mATL* – there is a non-elementary translation from mATL* to ATL* – the following result easily follows from Theorem 9.

► **Corollary 10.** *NE satisfiability in multi-player non-zero-sum perfect-information concurrent games can be characterised in ATL*.*

Some complexity results follow from Theorem 9. Since satisfiability and model checking for mATL* are 2EXPTIME, it follows that, given a tuple $(\mathrm{AP}, \mathrm{Ag}, \mathrm{Ac}, (\gamma_i)_{i \in \mathrm{Ag}})$, deciding whether there is a CGS $M = (\mathrm{AP}, \mathrm{Ag}, \mathrm{Ac}, \mathrm{St}, s^0, \lambda, \delta)$ such that the game $G = (M, (\gamma_i)_{i \in \mathrm{Ag}})$ has a Nash equilibrium – the problem we call *NE satisfiability* – is a 2EXPTIME problem.

The other important decision problem, namely checking the existence of a Nash equilibrium over a given CGS $M$ – the problem we call *NE model checking* – is already known to be 2EXPTIME and solved, for instance, using the Boolean goal fragment of SL. Then, we focus on NE satisfiability instead. Formally, the following complexity result can be shown.

▶ **Corollary 11.** *NE satisfiability is 2EXPTIME-complete.*

Regarding NE model checking, we would like to note that because $\phi_{\mathrm{NE}}$ is an ATL$^*$ formula, it cannot distinguish between bisimilar models. Then, in order to do NE model checking with $\phi_{\mathrm{NE}}$, one has to necessarily ensure that the game being analysed already has deterministic past – for instance, such as the games studied in [5] – so that the map **detp**, which introduces bisimilar states to the model to be checked, need not be used.

In order to improve, or simplify, the results obtained in this section, in the next section we will study special cases where the problems under consideration may be less complex. In particular, we will consider two interesting, and frequently found, scenarios in the games literature: sequential games (sometimes also called turn-based games) and two-player games.

## 6 Special Cases

### Sequential Games

The fact that in a sequential (turn-based) game only one player makes a non-trivial move greatly simplifies the way to reason about Nash equilibria using mATL$^*$. In fact, we can express the existence of a Nash equilibrium in the logic m$^-$ATL$^*$ (mATL$^*$ without *present*), which is *strictly weaker* than mATL$^*$. The formula to express Nash equilibria in m$^-$ATL$^*$ is:

$$\phi^{\mathsf{SG}}_{\mathsf{NE}} = \mathbf{E} \bigwedge_{j \in \mathrm{Ag}} (\neg\gamma_j \implies \mathbf{G} \langle\langle \mathrm{Ag}_{-j} \rangle\rangle \neg\gamma_j)$$

from which, again using Lemmas 3 and 4, the next expressivity result can be shown.

▶ **Theorem 12.** *Let $(\gamma_i)_{i\in\mathrm{Ag}}$, with each $\gamma_i \in \mathcal{L}(\mathrm{AP})$, be the goals of a set of players* Ag. *Then, the m$^-$ATL$^*$ formula $\phi^{\mathsf{SG}}_{\mathsf{NE}}$ constructed from $(\mathrm{AP}, \mathrm{Ag}, (\gamma_i)_{i\in\mathrm{Ag}})$ is satisfiable in a sequential game if and only if there is some CGS $M = (\mathrm{AP}, \mathrm{Ag}, \mathrm{Ac}, \mathrm{St}, s^0, \lambda, \delta)$ such that the multi-player non-zero-sum sequential game $G = (M, (\gamma_i)_{i\in\mathrm{Ag}})$ has a Nash equilibrium.*

In this case, one can see that the formula $\mathbf{G} \langle\langle \mathrm{Ag}_{-j} \rangle\rangle \neg\gamma_j$ is almost an explicit logical description of $\pi\, \rho((\vec{f}^{\pi}_{-j}, g_j), \pi) \notin \Gamma_j$, from Lemma 3, as the expression must hold for every $\pi$. There are two reasons why the expression for sequential games is so much simpler than the general case: firstly, only one player can deviate at each state – thus any deviation is uniquely attributable – and, secondly, winning (punishment) strategies can be defined directly on the Nash equilibrium run being followed, rather than on paths taken by deviations.

### Two-player Games

The complexity of both checking the existence (NE satisfiability) and verifying the satisfaction (NE model checking) of Nash equilibria cannot be improved in the two-player case. However, an explicit and conceptually simple algorithm for NE satisfiability can be defined in the two-player case, which can be solved using CTL$^*$ synthesis, as usual, by associating the action set $\mathrm{Ac} = \mathrm{Ac}_1 \uplus \mathrm{Ac}_2$ with the set of atomic propositions AP. We write SAT$(\psi)$ for LTL satisfiability and SYN$(\psi, \mathit{In}, \mathit{Out})$ for CTL$^*$ synthesis, where player 1 plays first trying to realise $\psi$, controls $\mathit{In}$, and with $\mathit{In}$ and $\mathit{Out}$ the sets of input and output variables; cf. [5].

```
TWO-NE-SATISFIABILITY($\mathrm{AP}_1, \mathrm{AP}_2, \gamma_1, \gamma_2$)
1.  if SAT($\gamma_1 \wedge \gamma_2$) return "yes"
2.  if SYN($\gamma_1$, $\mathrm{AP}_1$, $\mathrm{AP}_2$) or SYN($\gamma_2$, $\mathrm{AP}_2$, $\mathrm{AP}_1$) return "yes"
3.  if SYN($\neg\gamma_2$, $\mathrm{AP}_1$, $\mathrm{AP}_2$) and SYN($\neg\gamma_1$, $\mathrm{AP}_2$, $\mathrm{AP}_1$) return "yes"
4.  if SYN($\mathbf{E}\,\gamma_1 \wedge \mathbf{A}\,\neg\gamma_2$, $\mathrm{AP}_1$, $\mathrm{AP}_2$) or SYN($\mathbf{E}\,\gamma_2 \wedge \mathbf{A}\,\neg\gamma_1$, $\mathrm{AP}_2$, $\mathrm{AP}_1$) return "yes"
5.  return "no"
```

The above algorithm, inspired by the results obtained in [6], suggests a simple and intuitive mATL* characterisation of the NE satisfiability problem in two-player games:

$$\begin{aligned}\phi^{2G}_{\mathsf{NE}} \;=\;& \Big(\mathbf{E}(\gamma_1 \wedge \gamma_2)\Big) \vee \Big(\langle\!\langle 1\rangle\!\rangle\, \gamma_1 \vee \langle\!\langle 2\rangle\!\rangle\, \gamma_2\Big) \vee \Big(\langle\!\langle 1\rangle\!\rangle\, \neg\gamma_2 \wedge \langle\!\langle 2\rangle\!\rangle\, \neg\gamma_1\Big) \vee \Big(\mathsf{NormForm} \wedge \big(\\ & \mathbf{E}\,\big(\gamma_1 \wedge \neg\gamma_2 \wedge \mathbf{G} \textstyle\bigwedge_{p\in \mathrm{AP}_1} (\mathbf{X}\,\check{a}^p_1 \implies [\![2]\!]\,\mathbf{F}(\mathit{present} \wedge \mathbf{X}(\check{a}^p_1 \implies \langle\!\langle 1\rangle\!\rangle\, \neg\gamma_2)))\big) \vee \\ & \mathbf{E}\,\big(\gamma_2 \wedge \neg\gamma_1 \wedge \mathbf{G} \textstyle\bigwedge_{p\in \mathrm{AP}_2} (\mathbf{X}\,\check{a}^p_2 \implies [\![1]\!]\,\mathbf{F}(\mathit{present} \wedge \mathbf{X}(\check{a}^p_2 \implies \langle\!\langle 2\rangle\!\rangle\, \neg\gamma_1)))\big)\big)\Big).\end{aligned}$$

That NE satisfiability for two-player turn-based games can be characterised in ATL* was known [3]. Our result is instead in the concurrent setting. The specification requires *present* and we do not see how to avoid it, which suggests that *concurrency makes things harder.*

▶ **Theorem 13.** *NE satisfiability for two-player non-zero-sum concurrent games can be solved using CTL* synthesis and characterised in ATL* using a translation from formula* $\phi^{2G}_{\mathsf{NE}}$.

## 7 Further Implications

### Strong Nash Equilibrium

The solution concept we have focused so far, namely Nash equilibrium, assumes that deviations are only possible by single players. In the concurrent and multi-agent world, however, it is natural to assume that players may deviate in groups in order to achieve a common goal. Such kind of behaviour is captured by the notion of *strong Nash equilibrium* where, informally, a strategy profile is a strong Nash equilibrium whenever no coalition/group of players who do not get their goals achieved can deviate in a beneficial way for all deviating players. Formally, a *strong Nash equilibrium strategy profile* is a strategy profile $\vec{f} = (f_1, \ldots, f_n)$ such that for all groups $C$ of players and all strategies $g_C$ in $F_C$, for all $j \in C$ we have $\rho(\vec{f}) \succsim_j \rho(\vec{f}_{-C}, g_C)$. We say that a run $\rho$ can be *sustained by a strong Nash equilibrium strategy profile* if there is some strong Nash equilibrium strategy profile $\vec{f}$ such that $\rho = \rho(\vec{f})$.

Our study of punishability in Section 3 can be extended to deviations by groups of players, and therefore to strong Nash equilibrium. Similarly, the logical specification given in Section 5 can also be easily modified to capture the new setting, in particular, by letting $\bigwedge_{C\subseteq L} \mathbf{A}\,\mathsf{Punishable}(C, d)$ in $\mathsf{AllPunishable}$, and re-defining $\mathsf{AttDev}$ and $\mathsf{Punish}$ as follows:

$$\mathsf{AttDev}(C, d) = \Big(\bigwedge_{i\in \mathrm{Ag}_{-C}} \check{a}^{d_i}_i\Big) \wedge \Big(\bigwedge_{j\in C} \neg\check{a}^{d_j}_j\Big) \qquad \text{and} \qquad \mathsf{Punish}(C) = \langle\!\langle \mathrm{Ag}_{-C}\rangle\!\rangle \bigvee_{j\in C} \neg\gamma_j\ .$$

We write $\phi_{\mathrm{SNE}}$ for the characterisation in mATL* of satisfiability for strong Nash equilibrium.

▶ **Theorem 14.** *Let* $(\gamma_i)_{i\in\mathrm{Ag}}$*, with each* $\gamma_i \in \mathcal{L}(\mathrm{AP})$*, be the goals of a set of players* $\mathrm{Ag}$ *with action set* $\mathrm{Ac} = \biguplus_{i\in\mathrm{Ag}} \mathrm{Ac}_i$*. Then, the formula* $\phi_{\mathrm{SNE}}$ *constructed from* $(\mathrm{AP}, \mathrm{Ag}, \mathrm{Ac}, (\gamma_i)_{i\in\mathrm{Ag}})$ *is satisfiable iff there is a Moore CGS with deterministic past* $M = (\mathrm{AP}, \mathrm{Ag}, \mathrm{Ac}, \mathrm{St}, s^0, \lambda, \delta)$ *such that the game* $G = (M, (\gamma_i)_{i\in\mathrm{Ag}})$ *has a strong Nash equilibrium.*

We would like to note that even though $\phi_{\text{SNE}}$ is more complex than $\phi_{\text{NE}}$, the latter formula can still characterise the satisfiability problem for strong Nash equilibrium in the two simpler settings we studied in the previous section. Firstly, note that in the sequential case, trivially, the same specification works for both Nash equilibrium and strong Nash equilibrium since only one player can deviate at a time. Likewise, with respect to two-player games the same specification also works for both solution concepts since the only deviation for a group of players greater than one, is to a run that satisfies all players' goals, which, by definition, is a Nash equilibrium. Formally, we have the following expressivity result.

▶ **Theorem 15.** *Formula $\phi_{\text{NE}}$ characterises the satisfiability problems for both Nash and strong Nash equilibria in two-player concurrent games and multi-player sequential games.*

### Synthesis

Finally, we study the nature – memoryless, finite-memory, etc. – of the strategies in the equilibrium strategy profiles discussed so far, should one intend to synthesise them. This has been an open question for some time: because the logics known to be able to express equilibria in multi-player games have an undecidable synthesis problem (and a non-elementary model checking problem), strategy synthesis is not fully understood in the most general case. On the contrary, in our setting the problem can be reduced to ATL$^*$ synthesis, which is known to be solvable using (one-bit) finite-memory strategies [15]. Formally, we have:

▶ **Corollary 16.** *Every (strong) Nash equilibrium can be realised using a (strong) Nash equilibrium profile $\vec{f} = (f_1, \dots, f_n)$, where each $f_i$ is a finite-memory strategy.*

## 8 Analysis

We have presented various expressivity and complexity results for the specification, computation, and verification of (strong) Nash equilibria, as well as particular results for two special scenarios, namely two-player games and sequential games. Our results build on a novel approach: a semantic characterisation of equilibria based on winning strategies and memoryful reasoning, which is amenable to a further temporal logic characterisation.

Our results suggest that a logic for strategic reasoning where memoryful reasoning and attributability can be explicitly captured may ease the logical description of the existence of Nash equilibria. Otherwise, having a model of games where deviations can be more easily recognised should help when reasoning about Nash equilibria within a logical framework.

**Acknowledgements.** We are extremely grateful to the anonymous reviewers for their detailed comments. We thank the financial support of the ERC grant 291528 ("RACE") at Oxford.

## References

1 Rajeev Alur, Thomas A. Henzinger, and Orna Kupferman. Alternating-time temporal logic. *J. ACM*, 49(5):672–713, 2002.

2 Krishnendu Chatterjee and Thomas A. Henzinger. A survey of stochastic $\omega$-regular games. *J. Comput. Syst. Sci.*, 78(2):394–413, 2012.

3 Krishnendu Chatterjee, Thomas A. Henzinger, and Nir Piterman. Strategy logic. *Inf. Comput.*, 208(6):677–693, 2010.

4 Bernd Finkbeiner and Sven Schewe. Coordination logic. In *CSL*, volume 6247 of *LNCS*, pages 305–319. Springer, 2010.

5 Julian Gutierrez, Paul Harrenstein, and Michael Wooldridge. Iterated Boolean games. *Inf. Comput.*, 242:53–79, 2015.

6 Julian Gutierrez and Michael Wooldridge. Equilibria of concurrent games on event structures. In *CSL-LICS*. ACM, 2014.

7 Orna Kupferman and Moshe Y. Vardi. Memoryful branching-time logic. In *LICS*, pages 265–274. IEEE Computer Society, 2006.

8 François Laroussinie. Temporal logics for games. *Bulletin of the EATCS*, 100:79–98, 2010.

9 François Laroussinie and Nicolas Markey. Quantified CTL: expressiveness and complexity. *LMCS*, 10(4), 2014.

10 François Laroussinie and Nicolas Markey. Augmenting ATL with strategy contexts. *Inf. Comput.*, 2015. To appear.

11 Fabio Mogavero, Aniello Murano, Giuseppe Perelli, and Moshe Y. Vardi. Reasoning about strategies: On the model-checking problem. *ACM Trans. Comput. Log.*, 15(4):34, 2014.

12 Fabio Mogavero, Aniello Murano, and Luigi Sauro. On the boundary of behavioral strategies. In *LICS*, pages 263–272. IEEE Computer Society, 2013.

13 Fabio Mogavero, Aniello Murano, and Moshe Y. Vardi. Relentful strategic reasoning in alternating-time temporal logic. *J. Log. Comput.*, 2015. To appear.

14 Martin Osborne and Ariel Rubinstein. *A Course in Game Theory.* The MIT Press, 1994.

15 Sven Schewe. ATL* satisfiability is 2EXPTIME-complete. In *ICALP*, volume 5126 of *LNCS*, pages 373–385. Springer, 2008.

16 Wiebe van der Hoek and Michael Wooldridge. Logics for multiagent systems. *AI Magazine*, 33(3):92–105, 2012.

17 Igor Walukiewicz. A landscape with games in the background. In *LICS*, pages 356–366. IEEE Computer Society, 2004.