Reasoning about Equilibria in Game-like Concurrent Systems

Julian Gutierrez, Paul Harrenstein, Michael Wooldridge

Department of Computer Science
University of Oxford

Abstract

In this paper we study techniques for reasoning about game-like concurrent systems, where the components of the system act rationally and strategically in pursuit of logically-specified goals. Specifically, we start by presenting a computational model for such concurrent systems, and investigate its computational, mathematical, and game-theoretic properties. We then define and investigate a branching-time temporal logic for reasoning about the equilibrium properties of game-like concurrent systems. The key operator in this temporal logic is a novel path quantifier $[\mathbf{NE}]\varphi$, which asserts that φ holds on all Nash equilibrium computations of the system.

Keywords: Games and strategic reasoning, Temporal logic, Equilibria.

1. Introduction

Our goal in this paper is to develop a theory and techniques for reasoning about *game-like concurrent systems*: concurrent systems in which system components (agents) act strategically in pursuit of their own interests. Game theory is the mathematical theory of strategic interaction, and as such is an obvious candidate to provide the analytical tools for this purpose [43]. However, since the systems we are interested in modelling and reasoning about are interacting computer programs, it seems appropriate to consider how existing techniques for the analysis of computer systems might be combined with game-theoretic concepts. Temporal logics [13] and model checking [10] form the most important class of techniques for reasoning about computer programs, and in this paper we are concerned with extending such formalisms and techniques to the game-theoretic analysis of systems.

The artificial intelligence, computer science, and multi-agent systems literatures contain a great deal of work on logics intended for reasoning about game-like systems: *e.g.*, Parikh's Game Logic was an early example [44], and more recently ATL [2] and Strategy Logic [7] have received much attention. However, these formalisms are primarily intended for reasoning about the strategies/choices of players and their effects, rather than the preferences of players and the strategic choices they will make arising from them. It is, of course, possible to use ATL or Strategy Logic (or indeed LTL, CTL, ...) to define the goals of agents and their preferences; but such languages do not provide any object language constructs for reasoning about the behaviour of such agents under the assumption that they act rationally and strategically in pursuit of their goals. In this paper, we present a branching time logic that is explicitly intended for this purpose. Specifically, we provide a logic for reasoning about the *equilibrium* properties of game-like concurrent systems.

Equilibrium concepts are the best-known and most widely applied analytical tools in the game theory literature, and of these Nash equilibrium is the best-known [43]. A Nash equilibrium is an outcome that can be obtained when no player has an incentive to deviate, *i.e.*, to change its strategy. If we consider Nash equilibrium in the context of game-like concurrent systems, then it is natural to ask *which computations (runs, histories, ...) will be generated in equilibrium*? In [20], this question was investigated using the Iterated Boolean Games (iBG) model. In this model, each player is assumed to control a set of Boolean variables, and the game is played over an infinite sequence of rounds, where at each round every player chooses values for its variables. Each player has a goal, expressed as an LTL formula, and acts strategically in pursuit of this goal. Given this, some computations of a game can be identified as being the result of Nash equilibrium strategies, and [20] suggested that the key questions in the strategic analysis of the system are whether a given LTL formula holds in some or all equilibrium computations.

While the iBG model of [20] is useful for the purposes of exposition, it is not a realistic model of concurrent programs. Moreover, [20] provides no language for reasoning *about* the equilibria of systems: such reasoning must be carried out at the meta-level. This paper fills those gaps. First, we present a computational model that is more appropriate for modelling concurrent systems than the iBG model. In this model, the goals (and thus preferences) of players are given as temporal logic formulae that the respective player aspires to satisfy. After exploring some properties of this model, we introduce *Equilibrium Logic* (EL) as a formalism for reasoning about the equilibria of such systems. EL is a branching time logic that provides a new path quantifier $[\mathbf{NE}]\varphi$, which asserts that φ holds *on all Nash equilibrium computations of the system*. Thus, EL supports reasoning about equilibria directly in the object language. We then investigate some properties of this logic.

In particular in this paper we show that via a logical characterisation of equilibria in infinite games we can check useful properties of strategy profiles. We consider four logics for players' goals: LTL [45], CTL [9], the linear-time μ-calculus [51], and the modal μ-calculus [28]. Based on our logical characterisation, three problems are studied: Strategy-Checking, NE-Checking, and Equivalence-Checking, all of which are shown to be in PSPACE or in EXPTIME depending on the particular problem and temporal logic at hand. We also study the computational complexity of checking equilibrium properties, which can be expressed in the object language of EL. We show that the problem is 2EXPTIME-hard, even for LTL or CTL goals. This result shows, in turn, that checking equilibrium properties is equally hard in the linear-time and in the branching-time frameworks. We then investigate the complexity of model checking equilibrium computations with respect to dominant strategy equilibrium, a much stronger solution concept than Nash equilibrium. A summary of these complexity results is given at the end. We also present a class of games—where players are allowed to have an ordered set of (temporal logic) goals they want to see satisfied—for which all the main complexity results in the paper can be extended.

Structure of the paper. Section 2 defines the structure we use to represent games and strategies. Section 3 defines the model of games and strategies we will use in this paper. Section 4 gives a logical characterisation of equilibrium computations and investigates its main computational questions. Section 5 introduces our Equilibrium logics, and Section 6 presents a number of examples. Then, Section 7 studies the complexity of evaluating the new modal equilibrium operator, and Section 8 extends the main complexity results from Nash to dominant strategy equilibrium and from the standard model of games we use throughout the paper to a more general model where players are allowed to have an ordered set of temporal logic goals. At the end, Section 9 provides conclusions and related work, and Section 10 outlines a number of different avenues for further developments as well as some ideas underlying ongoing work. Throughout the paper, we assume familiarity with temporal logics such as LTL, CTL, the linear-time and the modal μ-calculus—see, *e.g.*, [13, 18, 28, 51] for details.

2. Models

Before giving formal definitions, let us start by describing some situations that can naturally be modelled as game-like concurrent systems.

Example 1. Consider a situation in which two agents can request a resource from a dispatch centre infinitely often; assume that the goals of both agents are to always eventually make use of the resource. For instance, in temporal logic notation, if the resource is modelled by "r" then their goals can be given by the Linear Temporal Logic (LTL [45]) formula $\mathbf{GF}r$. The behaviour of the dispatch centre is as follows:

1. if only one of the agents requests the resource, it gets it;
2. if both agents request the resource, neither agent gets it;
3. if one agent requested the resource twice in a row while the other agent did not do so, the latter agent inevitably gets the resource for ever after regardless of the behaviour of either of the players afterwards (thus, punishing undesirable greedy behaviour).

Because of 2 and 3 it may be the case that an agent (or both) fails to achieve its goal of being granted the resource infinitely often. But, of course, we can see a simple solution: if both agents request the resource alternately, then they get their goals achieved. Indeed, a game-theoretic analysis reveals that *in all equilibria of this system both agents get the resource infinitely often.*

Example 2. Another illustration of our framework is a strategic workflow design setting in which, at every point of time, employees can form coalitions and, depending on the coalitions formed, perform certain tasks. The employees are assumed to be free in the coalitions they join, which grants them some power with respect to the tasks that are being performed at each time *given the choices of the other players.* Additionally, there is a manager overseeing the employees who wants to ensure the tasks to be performed in a certain order or a particular number of times. This she tries to achieve by assigning responsibilities to the employees. Each employee is assumed to adopt the fulfilment of the responsibility assigned to him as his sole goal. For instance, an employee may be made responsible for one particular task never preceding another one. Temporal logic formulae can be used to formulate such responsibilities. Clearly, different ways of *assigning/distributing* responsibilities lead to different behavioural properties of the system. One interesting question in this setting is therefore whether there is a way in which the manager can distribute responsibilities so as to ensure that her objectives are achieved in all Nash equilibria of the system.

The approach we develop in this paper enables us to formally model and reason about systems and situations like those in the examples above (in Section 6, we show how these examples may be formally represented using our framework). The components of our framework are as follows. First, we present a mathematical model that plays a dual role in the remainder of the paper. Essentially, the model is a directed graph in which both edges and vertices are labelled. This model is used to represent both the *structure of a concurrent system* (in which case we call it an *arena*) and also the *reactive behaviour of system components* (in which case we call it a *strategy*):

- When our model is used as an **arena** the *vertices* in the graph will correspond to *system states*, which are labelled with the propositions true in those states, while *edges* in the graph will correspond to *choices* made by system components. We can think of the input language of the arena as being a sequence of choices, and the output language as being a sequence of system states, each labelled with a set of propositions. As will be clear from their formal definition (below), our basic model generalises both Kripke frames and transition systems, among others, thus allowing them to be used in very many different contexts. More importantly, compositions of these models naturally represent the behaviour of synchronous, multi-agent, and concurrent systems with interleaving semantics as well as of asynchronous systems [41].

- When our model is used to represent **strategies** the *vertices* in the graph will correspond to *strategy states*, each labelled with the *choice* made by the strategy in that state, while *edges* are labelled with information obtained from (that is true at) *system states*. Thus, we can think of the input language for a strategy as being a sequence of propositions, which are true at system states, and the output language as being a sequence of choices. As we will see, when we use our models to represent strategies, they can be understood as Moore machines.

Diagrammatically, the system architecture of the kind of multi-agent game-like concurrent systems we will define and use here is as in Figure 1. While the arena represents the (global and external) "observable behaviour" of the concurrent and reactive multi-agent system—which may admit many

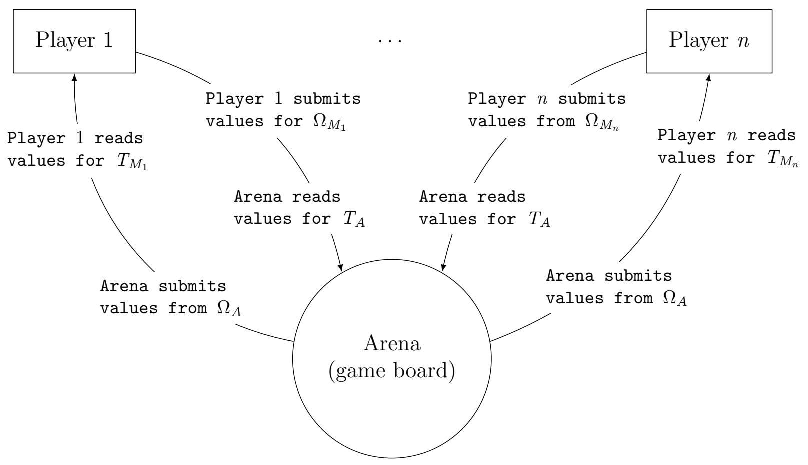
Player 1
· · ·
Player n
Player 1 submits values for Ω_{M_1}
Player n submits values from Ω_{M_n}
Player 1 reads values for T_{M_1}
Player n reads values for T_{M_n}
Arena reads values for T_A
Arena reads values for T_A
Arena submits values from Ω_A
Arena submits values from Ω_A
Arena (game board)

Figure 1: System Architecture: Multi-Agent Game-like Concurrent System.

different possible behaviours—the strategies/players will jointly determine (in a more local and internal manner) a particular choice of such behaviours.

Remark 3 (Arenas and Strategies)**.** Note that arenas and strategies contain dual information in our framework. Because we will model them using the same mathematical structure, which thus far we have called a "model" above, we will uniformly denote such structures/models by M. Later on, we will differentiate arenas from strategies using the notations A and σ, respectively.

A Graph-based Model of Concurrent and Reactive Behaviour. Let

$$M = (V, E, v^0, \Omega, \Lambda, \omega, \lambda)$$

be a Λ-labelled Ω-model M, where V is the set of vertices of M, $v^0 \in V$ is the initial vertex, $E \subseteq V \times V$ is the set of edges, and $\omega : V \to 2^\Omega$ and $\lambda : E \to 2^\Lambda$ are two functions, the former indicating the set of 'properties' of a vertex and the latter the ways to go/move from one vertex to another. Based on M, some sets can be defined. The set of *transitions*:

$$T_M = \{(v, a, v') \in V \times \Lambda \times V \mid (v, v') \in E \ \& \ a \in \lambda(v, v')\};$$

and the sets of sequences of adjacent vertices and transitions starting at v^0, which we denote by V^*_M and T^*_M.

Notation 4 (Graphs). We may write 'nodes' or 'states' when talking about vertices, and 'events' or 'actions' when referring to edges; similarly, we also write 'words' or 'strings' when talking about sequences.

A model is total if for every $v \in V$ there is a $v' \in V$ and an $a \in \Lambda$ such that $(v, a, v') \in T_M$; a model is, in addition, Λ-total if for every $v \in V$ and every $a \in \Lambda$ there is $v' \in V$ such that $(v, a, v') \in T_M$. Observe that if M is total the sets T^*_M and V^*_M contain infinite sequences only. The sets T^*_M and V^*_M induce two additional sets of sequences, one over the elements in the set Λ (the *action* names labelling the transitions) and another one over the elements in the set 2^Ω (the *properties* that hold, or can be observed, in the vertices of the model); namely the sets

$$\mathcal{A}^*_M = \{a, a', \ldots \mid (v^0, a, v'), (v', a', v'') \ldots \in T^*_M\}$$

and

$$\mathcal{P}^*_M = \{\omega(v^0), \omega(v'), \omega(v''), \ldots \mid v^0, v', v'', \ldots \in V^*_M\}.$$

Notation 5 (Sequences). Hereafter, for all sets and in all cases, we may omit their subscripts whenever clear from the context. Given a sequence ϱ (of any kind), we write $\varrho[0], \varrho[1], \ldots$ for the first, second, ... element in the sequence; if ϱ is finite, we write $last(\varrho)$ to refer to its last element. We also write $|\varrho|$ for the size of a sequence. The empty sequence is denoted by $\varrho[\,] = \epsilon$ and has size 0. Restrictions to parts of a sequence and operations on them are useful. Given $k, k' \in \mathbb{N}$, with $k \leq k'$, we write $\varrho[0 \ldots k]$ for the sequence $\varrho[0], \varrho[1], \ldots, \varrho[k]$ (an initial segment of ϱ), $\varrho[k \ldots k']$ for the sequence $\varrho[k], \ldots, \varrho[k']$, and $\varrho[k \ldots \infty]$ for the sequence $\varrho[k], \varrho[k+1], \ldots$. We also write $\varrho[k \ldots k')$ if the element $\varrho[k']$ of the sequence is not included. Given two sequences ϱ', ϱ, we write $\varrho' \in \varrho$ if $\varrho'[k] = \varrho[k]$ for all $0 \leq k < |\varrho'|$. If ϱ' is finite, we also write $\varrho'; \varrho$ for the binary operation on sequences/words—and resulting run—of concatenating a finite run ϱ' with a run ϱ. When working with sequences, we assume the standard laws over them; in particular, when expressing concatenation ";" with the empty sequence ϵ, we write $\epsilon; \varrho = \varrho$, for any ϱ, and write $\varrho'; \epsilon = \varrho'$, for any finite ϱ'.

We find it useful to associate *input and output languages* with models. The input language $\mathcal{L}_i(M)$ of a model M is defined to be $\mathcal{A}^*_M$ and the output language $\mathcal{L}_o(M)$ of M is defined to be $\mathcal{P}^*_M$. Given a finite set of models $\vec{M} = \{M_1, \ldots, M_n\}$, the input language of $\vec{M}$ is defined to be

$$\mathcal{L}_i(\vec{M}) = \bigcap_{1 \leq j \leq n} \mathcal{L}_i(M_j).$$

The set $\mathcal{L}_i(\vec{M})$ determines synchronised runs $V^*_{\vec{M}}$ for $\vec{M}$, *i.e.*, sequences

$$(v^0_1, \ldots, v^0_n), (v'_1, \ldots, v'_n), \ldots \in (V_1 \times \ldots \times V_n)^*$$

such that there are $a, a', \ldots \in \mathcal{L}_i(\vec{M})$ such that for all $1 \leq j \leq n$,

$$(v_j^0, a, v_j'), (v_j', a', v_j''), \ldots \in T^*_{M_j}.$$

The set $V^*_{\vec{M}}$, in turn, determines the output language $\mathcal{L}_o(\vec{M})$ of $\vec{M}$, defined to be all sequences

$$\bigcup_{1 \leq j \leq n} \omega_j(\varrho[0]), \bigcup_{1 \leq j \leq n} \omega_j(\varrho[1]), \ldots \in (2^{\Omega_1 \cup \cdots \cup \Omega_n})^*$$

where $\varrho \in V^*_{\vec{M}}$ and $\omega_j(\varrho[k])$, with $0 \leq k < |\varrho|$, is the application of the ω_j of M_j to the jth component of each $\varrho[k]$. Let $\mathcal{L}$ function equally for an input or output language of a model M or compound system $\vec{M}$; moreover, for finite ϱ', let $\mathcal{L}[\varrho']$ be the language of (sub)words $\{\varrho[|\varrho'| \ldots \infty] \mid \varrho' \in \varrho \in \mathcal{L}\}$. There is a tree language $\mathit{TreeL}(\mathcal{L})$ for every (word) language $\mathcal{L}$, defined as:

$$\mathit{TreeL}(\mathcal{L}) = \{\mathcal{T} \text{ is a tree} \mid \varrho \in \mathcal{L}, \text{ for each maximal path } \varrho \text{ in } \mathcal{T}\},$$

that is, the *tree language* of a *word language* $\mathcal{L}$ is the set of all trees all of whose maximal paths are in $\mathcal{L}$.

Remark 6 (Languages)**.** Note that any non-empty word language induces a tree language comprising infinitely many trees, if such trees are allowed to be non-deterministic. For instance, the (singleton) word language $\mathcal{L} = \{a\}$ induces the tree language $\mathit{TreeL}(\mathcal{L})$ containing the following trees: the empty tree, the tree with one a-labelled branch, the tree with two a-labelled branches, etc., and the infinite tree with infinitely many a-labelled branches. However, if non-determinism is not allowed (or restricted in some way) some finite word languages may always induce finite tree languages. For the sake of generality we impose no restrictions at this point.

Our models support two useful operations: *restriction* and *projection*. The former selects a subset of the output language; a restriction with respect to a subset of the input language. We denote by $\mathcal{L}_o(M)|_{\mathcal{L}}$, where $\mathcal{L} \subseteq \mathcal{L}_i(M)$, such a subset of the output language. Projection, on the other hand, takes the sequences in the output language and forgets the elements in some subset of Ω. We write $\mathcal{L}_o(M)|_{\Omega'}$ for such an operation and resulting set, which is formally given by:

$$\{\varrho[0] \cap \Omega', \varrho[1] \cap \Omega', \ldots \in (2^{\Omega'})^* \mid \varrho \in \mathcal{L}_o(M)\}.$$

Based on the model just given both games and strategies in a game can be defined, as presented in the next section.

3. Games and Strategies

Games. Using the model in Section 2, we will now define *reactive games*: a class of multi-player non-zero-sum games. In a reactive game, a finite set of players interact with each other by assigning values to variables they have control over. The game has a designated initial state and the values given to the variables at each round determine the next state of the game. The game is played for infinitely many rounds. Players in a reactive game have goals they wish to satisfy; such goals are expressed as temporal logic formulae.

Formally, a reactive game (sometimes just called a "game") is a structure:

$$G = (N, C, (C_i)_{i \in N}, (\gamma_i)_{i \in N}, X, A)$$

where $N = \{1, \ldots, n\}$ is a set of *agents* (the players), $C = \{p, q, r, \ldots\}$ is a set of *controlled variables*, $C_i \subseteq C$ is the set of variables under the *unique control* of player i, and γ_i is a formula of some logical system[1] over a set $X = \{x, y, \ldots\}$ of propositions; formula γ_i describes the *goal* that player i wants to achieve. There is a requirement on C: the sets of variables $C_1, \ldots, C_n$ form a partition of C, that is, $C_i \cap C_j = \emptyset$ for all $i \neq j \in N$, and $C = C_1 \cup \cdots \cup C_n$. A *choice* c_i for agent $i \in N$ is an assignment of values for the variables under its control. Let Ch_i be the set of choices for agent i. A *choice vector* $\vec{c} = (c_1, \ldots, c_n)$ is a collection of choices, one for each player. Let Ch be the set of all choice vectors. Finally, A—the *"arena" or "board"* where the game is played—is a Λ-total Ω-model where $\Lambda = \mathit{Ch}$ and $\Omega = X$.

Note that Λ-totality ensures in a simple and convenient manner that a reactive game is played for an infinite number of rounds without requiring further consistency or validity conditions on strategies. Moreover, it does not limit our modelling power. In particular, the inability of a player to make a choice at a particular vertex of the arena can be modelled by exploiting non-determinism and labelling all outgoing edges with *all* of its possible choices. Then, the successor state only depends on the other players' choices, which comes down to the player not being able to make a meaningful choice.

Remark 7 (Games)**.** Reactive games can be considered as a meta-model of *infinite* games of *control* since their definition does not specify the logic

[1] In this paper, we will consider several logical temporal languages for γ_i, *e.g.*, LTL [45], CTL [9], or fixpoint linear-time and branching-time modal logics [51, 28]. At this point, our definitions do not require us to fix on any one language for players' goals, although we will consider several concrete possibilities later.

each γ_i belongs to, the kinds of strategies used in the game, the types of variables the players have control over, what the outcome of a game would be given a set of players' strategies, or when the outcome of a game makes a player's goal satisfied. As we will see later in the paper, different kinds of games and results will arise from different choices with respect to these properties. All we need to know for now is that

1. the games are played for *infinitely* many rounds,
2. the players have *unique control* over some given set of variables, and
3. the players' goals are represented by formulae of some *logical language.*

Strategies. Because in a reactive game a play is infinite, it is natural to think of a strategy for a player i as a function $f_i : E^* \to Ch_i$ or $f'_i : V^* \to Ch_i$, that is, a function from what has been played so far (or at least what a player *knows so far*), which may be given for instance by a finite sequence of arena states in V^* or a finite sequence of all players' choices in E^*, to a particular choice c_i for player i. However, sometimes such representations are less concrete than one would like them to be. For instance, one would like to know their particular structure and size in order to assess whether they can be effectively constructed and, if so, how complex such a construction process would be. With this consideration in mind, we use a strategy model that is *finite*, *simple*, and *expressive* enough for most computational purposes. Our definition of strategies is based on the model in Section 2. Similar representations have been used to study, *e.g.*, the important class of 'repeated games' in game theory [43, pp. 140–143].

Formally, we define a *strategy* σ_i for player i in a reactive game $G = (N, C, (C_i)_{i\in N}, (\gamma_i)_{i\in N}, X, A)$ as a structure

$$\sigma_i = (Q_i, q_i^0, \delta_i, \tau_i)$$

modelled as a structure $M_i = (V, E, v^0, \Omega, \Lambda, \omega, \lambda)$ in which $Q_i = V$ is a finite and non-empty set of *states*, $q_i^0 = v^0$ is the *initial* state, $\delta_i : Q_i \times \Lambda \to (2^{Q_i} \backslash \emptyset)$, with $\Lambda = 2^X$, is the *transition function* given by T_{M_i}, and $\tau_i = \omega : Q_i \to Ch_i$ is a *choice function.* As one requires that a strategy for player i is able to *react* to any possible valid behaviour/strategy of the others, we only consider as *valid* the strategies that are based on structures M_i where δ_i is total. Hereafter, let Σ_i denote the class of strategies for player i.

Our strategies resemble Moore finite state machines, *i.e.*, transducers where the output function is determined only by the states of the machine.

It is easy to modify our definition of strategies to encode Mealy machines. (Recall that Moore and Mealy machines are equally powerful and, in fact, fully interchangeable.) Henceforth, given a game G with n players in N and a profile of strategies $\vec{\sigma} = (\sigma_1, \ldots, \sigma_n)$, we call $\vec{\sigma}$ or any subset of it a strategy profile. We write $\vec{\sigma}_{-S}$, with $S \subseteq N$, for $\vec{\sigma}$ without the strategies σ_i such that $i \in S$; we omit brackets if S is a singleton set. Also, we write $(\vec{\sigma}_{-i}, \sigma'_i)$, with $1 \leq i \leq n$, for the strategy profile $\vec{\sigma}$ where σ_i is replaced with σ'_i.

Example 8. Consider a game with $N = \{1, 2\}$, $C_1 = \{p\}$, and $C_2 = \{q\}$ and arena as in Figure 2. There, we have $\omega_A(v^0) = x$, $\omega_A(v') = \omega_A(v'') = \bar{x}$. The values the players can choose to assign to the variables they control are the Boolean constants $\perp$ ("false") and $\top$ ("true"). Moreover, the symbol $\bar{p}$ denotes player 1's choice to assign $\perp$ to p, that is $p := \perp$, and, similarly, p denotes the choice to assign $\top$ to p, that is, $p := \top$ (and likewise for q and x). Then, the set of choices of players 1 and 2 is the binary set of Boolean values. In this very simple game the players can guarantee $x := \top$ as long as they play the same Boolean values for the variables p and q they have control over. However, as soon as they play different Boolean values, the output of the game will be $x := \perp$ ever after. A possible strategy for player 1 would be to always play p and is depicted in Figure 3 (left). The exact outcome(s) of the game—to be defined next—can be determined only once the strategy for player 2 is given.

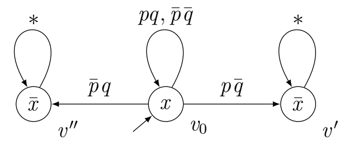

Figure 2: An arena, where $* = \{pq, \bar{p}\,q, p\,\bar{q}, \bar{p}\,\bar{q}\}$.

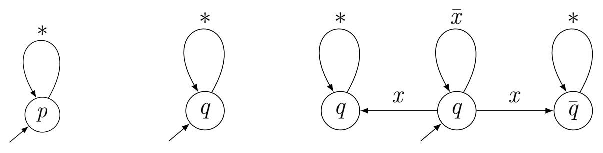

Figure 3: Strategy σ_1 for player 1 (left) and the strategies σ_2 (middle) and σ'_2 (right) for player 2. Here $* = \{x, \bar{x}\}$.

Expressivity. Our strategy model is simple but powerful; in particular, it can generate any word or tree ω-regular language. Formally, we have:

Fact 9. *If $\mathcal{T}$ is an ω-regular tree, then there is a (possibly non-deterministic) strategy σ such that $\mathcal{T} \in TreeL(\mathcal{L}_o(\sigma))$.*

It is important to note that the strategy σ may be non-deterministic. However, with respect to word languages, only deterministic strategies are needed: the linear-time languages we are interested in have ω-regular words as models. Because of this, a strategy model based on finite-state machines with output (transducers) is sufficiently powerful to generate and recognise such a kind of infinite words over any finite domain [46]. Specifically, with respect to our results, because ω-regular words are ω-regular trees that do not branch, the following statement also easily follows.

Fact 10. *If w is an ω-regular word, then there is a deterministic σ such that $w = \mathcal{L}_o(\sigma)$.*

Facts 9 and 10 are used to ensure that if some ω-regular word or tree needs to be realised, then such a word or tree can be generated by (*i.e.*, be in the output language of) some finite-state machine strategy, as defined in our framework. This property of our strategy model will be used, later on, to give a formal interpretation of logical languages with ω-regular models, for instance, of temporal logics such as LTL or CTL.[2]

Outcomes and composition of strategies. Given a finite set of strategies $(\sigma_i)_{i \in N}$, which hereafter we will denote by $\vec{\sigma}$ whenever the set N is clear from the context, the *histories* (of choices) when playing such a set of strategies in a game $G = (N, C, (C_i)_{i \in N}, (\gamma_i)_{i \in N}, X, A)$ are the sequences in the input language of A, denoted by $\mathcal{L}_i^{\vec{\sigma}}(A)$, given by

$$\mathcal{L}_i^{\vec{\sigma}}(A) = \mathcal{L}_o(\vec{\sigma})|_{\mathcal{L}_i(\vec{\sigma}) \cap \mathcal{L}_o(A)[\vec{q}^0]},$$

where $\vec{q}^0 = (\tau_1(q_1^0), \ldots, \tau_n(q_n^0))$.

Informally, since $\mathcal{L}_o(\vec{\sigma})$, and hence $\mathcal{L}_i^{\vec{\sigma}}(A)$, is restricted to $\mathcal{L}_i(\vec{\sigma}) \cap \mathcal{L}_o(A)[\vec{q}^0]$, we know that when playing a strategy profile there is an *alternation* in the

[2]In fact, the class of strategies needed to show Facts 9 and 10 is quite simple and was called "myopic" in [23], where we used them to study games with LTL goals.

interaction between strategies and the arena, with the strategies making transitions only *after* a transition in the arena has been made.

The histories of a game with respect to a strategy profile $\vec{\sigma}$ record the choices that the players make based on such a given strategy profile $\vec{\sigma}$. These choices, in turn, determine the *outcomes* of the game, denoted by $\mathcal{L}^{\vec{\sigma}}_o(A)$ and defined as

$$\mathcal{L}^{\vec{\sigma}}_o(A) = \mathcal{L}_o(A)|_{\mathcal{L}^{\vec{\sigma}}_i(A)}.$$

Then, whereas *histories* are sequences in the *input* language of A, *outcomes* are sequences in its *output* language.

As defined, the outcomes of a game form a set of words or infinite sequences over $(2^{\Omega_A})^*$. However, they naturally induce a set of trees with respect to the tree-unfolding of A. We write $\mathit{unf}(A, v)$ for the usual tree-unfolding of A—when A is seen as a graph—with respect to a given vertex $v \in V$; we simply write $\mathit{unf}(A)$ whenever $v = v^0$. The tree $\mathit{unf}(A)$ can trivially be seen as an ordered set $(W, \leq)$, with W the set of *vertices* in the tree (which correspond to *occurrences of states* in the arena A) and $\leq$ the partial order induced by the set of *edges* in the tree (which correspond to *occurrences of transitions* in the arena A). A *subtree* T of $\mathit{unf}(A)$ is a downclosed subset of $(W, \leq)$ satisfying that for each vertex w of the subtree, there is another vertex w' of the subtree, with $w \neq w'$, such that $w \leq w'$. In other words, every branch of a subtree has infinite length. Based on this definition of subtree, let us define the set of subtrees of $\mathit{unf}(A)$ as:

$$\mathit{Tree}(A) = \{T \mid T \text{ is a subtree of } \mathit{unf}(A)\}.$$

We will abuse of notation, and write $\mathit{Tree}(A)$ for the unique set of trees of valuations/properties associated with the trees of (occurrences of) states just defined. Then, using this notation and given a strategy profile $\vec{\sigma}$ in a reactive game G, the tree/branching outcomes of G are the trees in the set

$$\mathit{TreeL}(\mathcal{L}^{\vec{\sigma}}_o(A)) \cap \mathit{Tree}(A).$$

Then, the above set contains all the trees that can be generated by a (possibly non-deterministic) strategy $\vec{\sigma}$ which are consistent with the trees in the set of subtrees of $\mathit{unf}(A)$, that is, the possible tree outcomes of a game on A.

Hereafter, regardless of whether we are talking about word outcomes or tree outcomes, we will uniformly denote by $\mathit{Out}(G, \vec{\sigma})$ the outcomes of a game G when playing the set of strategies $\vec{\sigma}$. Similarly we will denote

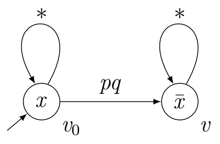

Figure 4: A non-deterministic arena where $* = \{pq, \bar{p}q, p\bar{q}, \bar{p}\bar{q}\}$.

by Out_G the set of all outcomes of the game G, that is, with respect to all valid sets of strategies, and omit the subscript G whenever which game G we are referring to is either clear or irrelevant. It is worth noting that Out is $\mathcal{L}_o(A)$ in case of word outcomes and, therefore, is $\mathit{TreeL}(\mathcal{L}_o(A)) \cap \mathit{Tree}(A)$ in case of tree outcomes. Also, note that because we allow *non-determinism*, the set $Out(G, \vec{\sigma})$ is not necessarily a singleton, as illustrated next.

Example 11. Consider again the game in Example 8 and the two strategies for player 2 depicted in Figure 3. The profile $\vec{\sigma} = (\sigma_1, \sigma_2)$ induces the unique (word) outcome $x^\omega = x; x; x; \ldots$ in Out; the strategy profile $\vec{\sigma}' = (\sigma_1, \sigma_2')$, on the other hand, induces two outcomes, namely the two infinite sequences given by the ω-regular expression $x^\omega \cup (x; \bar{x}^\omega)$. The reason why $\vec{\sigma}'$ induces more than one outcome is that σ_2' is *non-deterministic*; thus, multiple outcomes are possible even when the arena A is *deterministic*.

Also, given a set of *deterministic strategies* one can have a reactive game where multiple outcomes are possible if A is a *non-deterministic arena*, since the same players' choice can lead to different successor vertices in A. In this case the next state of the game is selected non-deterministically in A, *i.e.*, it is not under the control of any of the players.

Example 12. Take the *deterministic* strategies in the strategy profile $\vec{\sigma}$ in Example 11, and the arena in Figure 4. The outcomes of the game are the infinite sequences given by the ω-regular expression $x^\omega \cup (x^+; \bar{x}^\omega)$, but the situation here is different. Player 1 always plays $p := \top$ and player 2 always plays $q := \top$. In A this choice can lead to two different successor states from the initial one, namely the one labelled by x and the one labelled by $\bar{x}$. The choice of which the next state will be is made *non-deterministically* in A, *i.e.*, it is not under the control of any of the players.

Remark 13 (iBG model)**.** Observe that our reactive games model strictly generalises the iBG model as introduced by Gutierrez, Harrenstein, and

Wooldridge in [20], which can be represented as a simple reactive game where the arena is an *implicitly* defined clique whose nodes are the valuations of the variables the players have control over, the goals of the players are LTL formulae, and the strategies are deterministic. Moreover, the controlled variables, *i.e.*, variables in C, are precisely those in the set X. It should be noted, however, that arenas are not explicitly represented in the iBG model, and that the size of a reactive game corresponding to an iBG can in general grow exponentially with respect to the size of the iBG itself.

Algebraic properties. We finish this section by showing that the model we have defined to represent both arenas and strategies has useful mathematical and algebraic properties. Most notably, structures of this kind can be organised in categories [35]. Because such structures—which here we refer to as *models*—are used to represent both arena games and strategies, we can obtain categories of games and strategies in a uniform way.

Building a category of models is not the main purpose of this paper. However, it can be used to understand the relationships with other models of games and strategies at a more abstract mathematical level. It can also be used to relate with different models of concurrency for which categories are better known [41]. From a more practical point of view, categories have also been used to define methods for compositional reasoning, especially within the game semantics [1, 26] research communities. It then seems quite reasonable to formally define a category of our models to enable further investigations in any of the issues and questions above mentioned.

Before presenting this category of strategies and arena games, we need to define a suitable structure-preserving morphism between such structures. Such a morphism—formally a partial map—is defined as follows (we write $f(x) = \star$ when a partial function f is undefined on x, and $f(x) \neq \star$ otherwise):

Definition 14 (Morphism)**.** Let $M = (V, E, v^0, \Omega, \Lambda, \omega, \lambda)$ and let $M' = (V', E', (v^0)', \Omega', \Lambda', \omega', \lambda')$ be two models as defined in Section 2. A *morphism* $m : M \to M'$ is a triple $m = (\kappa, \alpha, \beta)$ such that $\kappa : V \to V'$, $\alpha : \Lambda \to \Lambda'$, and $\beta : 2^{\Omega} \to 2^{\Omega'}$ are partial maps satisfying the following properties:

1. $\kappa(v^0) = (v^0)'$;
2. $(v, a, r) \in T_M \wedge \alpha(a) = \star \Longrightarrow \kappa(v) = \kappa(r)$, and $(v, a, r) \in T_M \wedge \alpha(a) \neq \star \Longrightarrow (\kappa(v), \alpha(a), \kappa(r)) \in T_{M'}$;
3. $\kappa(v) \neq \star \Longrightarrow \beta(\omega(v)) = \omega'(\kappa(v))$.

With respect to this definition, the following algebraic property follows:

Proposition 15. *Our models for arena games and players' strategies, with the structure-preserving morphism given in Definition 14, form a category.*

4. Equilibria in Logical Form

Because players have goals, which they wish to satisfy, and their satisfaction depends on the outcomes—whether word or tree outcomes—of the game, the players may prefer some sets of outcomes over others. To formalise this situation we define, for each player i, a *preference* relation $\leq_i$ over 2^{Out}. Although $\leq_i$ can in principle be any binary relation over 2^{Out}, it is natural to assume that it is a preorder: that is, a reflexive and transitive relation. We write $<_i$ whenever $\leq_i$ is strict—or asymmetric, *i.e.*, $X <_i X'$ implies that $X' \leq_i X$ does not hold. Because strategy profiles induce sets of outcomes, we abuse notation by writing $\vec{\sigma} \leq_i \vec{\sigma}'$ to mean $Out(G, \vec{\sigma}) \leq_i Out(G, \vec{\sigma}')$, that is, that player i does not prefer the set of outcomes $Out(G, \vec{\sigma})$ over the set of outcomes $Out(G, \vec{\sigma}')$.

Based on players' preferences, a notion of *equilibrium* can be defined. We provide the definition of the, arguably, main concept of equilibrium—also called solution concept—in game theory, namely, *Nash equilibrium.* However, many other solution concepts can be found in the game theory and even computer science literatures, *e.g.*, dominant strategy, subgame perfect Nash, correlated, amongst others. We say that a strategy profile $\vec{\sigma}$ is a Nash equilibrium if for every player i and strategy $\sigma'_i \in \Sigma_i$ for i, we have

$$(\vec{\sigma}_{-i}, \sigma'_i) \leq_i \vec{\sigma}.$$

Intuitively, a Nash equilibrium formalises the idea that no player can be better off (have a beneficial deviation) provided that all other players do not change their strategies. Let $NE(G)$ be the set of Nash equilibria of G.

Remark 16 (Nash equilibrium)**.** Note that since strategies or arenas can be non-deterministic (implying that multiple outcomes can be induced) our definition of Nash equilibrium is in terms of *sets* of outcomes, rather than in terms of single outcomes only. Even though the definition of equilibrium is given with respect to preferences over sets of outcomes, we can think of such a definition as based on a *preference relation* over strategy profiles instead, since strategy profiles induce sets of outcomes. Thus, a preference relation allows one to define equilibria in a general way, not only for binary goals.

We can think of equilibria with respect to the goals the players of the game wish to satisfy. To make this statement precise, we need to know which logic the goals of the players belong to and when a set of outcomes satisfies such goals, that is, we need to define a semantics of players' goals with respect to 2^{Out}—*i.e.*, with respect to the outcomes of a game.

We can then abstractly think of the existence of a *satisfaction* relation "$\models$" between sets of outcomes and logical formulae, that is, a binary relation indicating whether a given goal γ_i for player i is satisfied or not by a set of outcomes $Out(G, \vec{\sigma})$ in a game G played with a strategy profile $\vec{\sigma}$. Assuming the existence of a denotation function $[\![\cdot]\!]$ from goals to sets of outcomes, we can then write

$$Out(G, \vec{\sigma}) \models \gamma_i \qquad \text{if and only if} \qquad Out(G, \vec{\sigma}) \subseteq [\![\gamma_i]\!].$$

Again, as strategy profiles induce sets of outcomes, we abuse notation and write $\vec{\sigma} \models \gamma_i$ if $Out(G, \vec{\sigma}) \models \gamma_i$. And, in order to simplify notation used in the paper, we will also write $[\![\vec{\sigma}]\!]$ for either the set of outcomes or the associated set of infinite sequences of vertices in V_A^* induced by $\vec{\sigma}$; which one we are referring to will always be clear from the context.

Based on the definitions above one can now formally state with respect to the goals $(\gamma_i)_{i \in N}$ of the game, when a strategy profile $\vec{\sigma}$ is a Nash equilibrium. We say that a strategy profile $\vec{\sigma}$ is a Nash equilibrium if, for every player i and for every strategy σ'_i, we have that

$$(\vec{\sigma}_{-i}, \sigma'_i) \models \gamma_i \qquad \text{implies} \qquad \vec{\sigma} \models \gamma_i.$$

Remark 17 (Temporal logics)**.** Since our model generalises Kripke structures and transition systems, amongst other structures, it can be used to give the standard semantics of conventional linear-time and branching-time temporal logics. In this paper, we will assume that players have *ω-regular goals.* In particular, in case of linear-time players' goals we will let each γ_i be either a linear-time μ-calculus or an LTL formula, strategies be deterministic, and outcomes be word outcomes; in case of branching-time goals, we will assume that the goals are either CTL or μ-calculus formulae, that the strategies may be non-deterministic, and that the outcomes are tree outcomes.

The details of the semantics of the temporal logics mentioned above need not be given to obtain the results in this paper. All such details can be found, *e.g.*, in [18, 28, 51]. Instead, what one needs to know is the complexities of their satisfiability and model checking problems, which are as follows: for satisfiability CTL and the μ-calculus are EXPTIME [15, 16], whereas LTL and the linear-time μ-calculus are PSPACE [47, 51]; for model checking with respect to a product of transition systems, the logics LTL, CTL, and linear-time μ-calculus are PSPACE [29, 24], while the μ-calculus is EXPTIME [29].

Because our specific models of concurrent and reactive multi-agent behaviour are strategies, the next question—which essentially corresponds to the model-checking problem in our framework—now becomes relevant:

> *Given*: Game G, strategy profile $\vec{\sigma}$, formula ψ.
> STRATEGY-CHECKING: Is it the case that $\vec{\sigma} \models \psi$?

Clearly, the answer to this question depends on the logic to which ψ belongs. The following lemma answers this question for various logics.

Lemma 18. *The* STRATEGY-CHECKING *problem for LTL, CTL, and the linear-time μ-calculus is PSPACE-complete, and it is in EXPTIME for the modal μ-calculus. The problem is PSPACE-hard even for formulae ψ of the form $\psi = \mathbf{F}\varphi$, where φ is a propositional logic formula.*

The proof of membership of Lemma 18 can be found in the appendix. The proof describes how to solve STRATEGY-CHECKING using the techniques developed in [29, 17]. The hardness proof is omitted, but can be obtained via a reduction from the finite automaton intersection problem [27], where finite automata are translated into strategies in a game in which $\psi = \mathbf{F}\varphi$ represents the situation where all automata accept the same word.

Using STRATEGY-CHECKING we can, moreover, show that the following problem, namely, membership of a strategy profile in the set of Nash equilibria of a reactive game, may be harder only in the branching-time case.

> *Given*: Game G, strategy profile $\vec{\sigma}$.
> NE-CHECKING: Is it the case that $\vec{\sigma} \in NE(G)$?

The previous problem is solved by describing the behaviour of the strategies in $\vec{\sigma}$ using an appropriate temporal logic and then solving a number of satisfiability and STRATEGY-CHECKING problems (up to $2|N|$).

Formally, we have

Lemma 19. *The* NE-CHECKING *problem for LTL and linear-time μ-calculus goals is PSPACE-complete. For CTL and μ-calculus goals the problem is EXPTIME-complete.*

The proof of membership of Lemma 19 can be found in the appendix. The proof describes how to solve NE-Checking using a variant of Algorithm 1 in [23]. It is worth noting that a solution of NE-Checking has been implemented, see [48], for the case where both arenas and strategies are described using the Reactive Modules specification language [3] and goals are given by CTL formulae. It is also worth noting that even though NE-Checking can be solved using automata-theoretic techniques, as shown in [48], our logic-based technique allows for a simple implementation using conventional temporal logic satisfiability and model checking techniques/tools. Finally, the hardness proofs easily follow from the complexities of the corresponding (temporal logic) satisfiability problems, *i.e.*, those for LTL and CTL.

Equivalences of equilibria. The characterisation of equilibrium with respect to the goals of the game given above provides a natural way of comparing strategy profiles, and hence of comparing equilibria, in a logical way. But first, we provide a notion of equivalence of strategy profiles purely based on the outcomes they induce and later on a weaker definition with a more logical flavour. Given a game G, we say that two strategy profiles $\vec{\sigma}$ and $\vec{\sigma}'$ are equivalent, and write $\vec{\sigma} \sim \vec{\sigma}'$ in such a case, if and only if they induce the same set of outcomes, that is, if and only if $[\![\vec{\sigma}]\!] = [\![\vec{\sigma}']\!]$.

Even though the definition of $\sim$ immediately provides a definition for equivalence between equilibrium strategy profiles, such a definition is rather strong. Instead, one would like a definition where only the satisfaction of goals was taken into account. Having this in mind, we propose a weaker, logically based definition of equivalence between strategy profiles. Formally, given a game G, we say that two strategy profiles $\vec{\sigma}$ and $\vec{\sigma}'$ are logically equivalent, and write $\vec{\sigma} \sim_\gamma \vec{\sigma}'$ in such a case, if and only if, they induce sets of outcomes that satisfy the same set of goals of the game, that is, if and only if for every goal in $(\gamma_i)_{i \in N}$ of G we have that

$$\vec{\sigma} \models \gamma_i \qquad \text{if and only if} \qquad \vec{\sigma}' \models \gamma_i.$$

Formally, with respect to the goals in $(\gamma_i)_{i \in N}$ of a reactive game given by $G = (N, C, (C_i)_{i \in N}, (\gamma_i)_{i \in N}, X, A)$, the problem one wants to solve is:

> *Given*: Game G, strategy profiles $\vec{\sigma}, \vec{\sigma}'$.
> Equivalence-Checking: Does $\vec{\sigma} \sim_\gamma \vec{\sigma}'$ hold in G?

An immediate consequence of Lemma 18 is:

Corollary 20. *The* EQUIVALENCE-CHECKING *problem for LTL, CTL, and linear-time μ-calculus goals is PSPACE-complete. For μ-calculus goals the problem is in EXPTIME.*

Towards a Logic for Equilibrium. In [20] we introduced a number of decision problems which, we believe, ask the main questions regarding the equilibria of a concurrent and multi-agent system. These questions are about the existence of equilibria in the system and about the satisfaction of temporal properties under the assumption that the agents (components, players, processes, etc.) in that system act independently, concurrently, and selfishly in pursuit of their own goals. Formally, such decision problems are as follows.

Let G be a reactive game, $\vec{\sigma}$ be a strategy profile, and φ be a formula of some logical formalism (LTL, CTL, etc.), whose semantics can be given in terms of the outcomes of G. Then, the following decision questions about the equilibrium properties of the system can be formulated:

> *Given*: Game G.
> NON-EMPTINESS: Is it the case that $NE(G) \neq \emptyset$?
>
> *Given*: Game G, formula φ.
> A-NASH: Is it the case that $\vec{\sigma} \models \varphi$, for all $\vec{\sigma}$ in $NE(G)$?
>
> *Given*: Game G, formula φ.
> E-NASH: Is it the case that $\vec{\sigma} \models \varphi$, for some $\vec{\sigma}$ in $NE(G)$?

These problems ask the most natural questions regarding the equilibria of a game-like concurrent and multi-agent system: namely, whether the system has *at least one* Nash equilibrium, whether there is a property that is *invariant in all* of its Nash equilibria, and whether there is a property that can be *achieved in some* Nash equilibria. A limitation of the work in [20] is that there is no specification language for expressing these questions. In this paper, we address that problem. In particular, we will define a language with which the problems above can be expressed in a transparent, logical way.

5. Equilibrium logics

We now introduce two logics for expressing equilibrium properties of game-like concurrent and multi-agent games: these two logics are closely related to the branching time logics CTL and CTL* (see, *e.g.*, [14]). We refer to our basic logical framework as *Equilibrium Logic*, (EL), and will refer to the two versions of this logic as EL (roughly corresponding to CTL) and EL* (roughly corresponding to CTL*). Since EL* will be defined as an extension of CTL*, the logic LTL will also appear as a syntactic fragment.

The basic idea of Equilibrium Logic is to extend the logic CTL* by the addition of two modal quantifiers, which we will write as "$[\mathbf{NE}]$" and "$\langle\mathbf{NE}\rangle$". In Equilibrium Logic, the modalities $[\mathbf{NE}]$ and $\langle\mathbf{NE}\rangle$ quantify over *paths that could arise as the consequence of processes (agents/players) selecting strategies in equilibrium.* For example, if we are dealing with Nash equilibrium, then an EL formula $[\mathbf{NE}]\mathbf{F}p$ (where $\mathbf{F}$ is the "eventually" LTL modality) means that on all Nash equilibrium computations—*i.e.*, on all computations (paths or trees) that correspond to runs or plays where a set of agents/players use a strategy profile in equilibrium—eventually p will hold. In this way, we can use Equilibrium Logic to directly reason about the equilibrium properties of game-like concurrent systems. Equilibrium Logic is parameterised by a solution concept, which determines the outcomes over which the "equilibrium modalities" quantify. For now, we consider Nash equilibrium as our by-default solution concept, but of course others can be considered too.

Syntax. The syntax of Equilibrium Logic is defined with respect to a set X of propositions, by the following grammars:

$$\begin{array}{lcl} \text{Path Formulae:} & \varphi & ::= \ \psi \mid \theta \mid \neg\varphi \mid \varphi \vee \varphi \mid \mathbf{X}\varphi \mid \varphi\,\mathbf{U}\,\varphi \\ \text{State Formulae:} & \psi & ::= \ \top \mid x \mid \mathbf{A}\varphi \\ \text{Equilibrium Formulae:} & \theta & ::= \ [\mathbf{NE}]\varphi \end{array}$$

where $x \in X$. Thus, as in CTL*, path formulae φ express properties of paths (*cf.* LTL), while state formulae ψ express properties of states. In addition, Nash equilibrium formulae also express properties of paths, but only if such paths are induced by strategy profiles in equilibrium.

Notation 21. All usual abbreviations for the Boolean operators not explicitly given in the grammars above are assumed to be defined. Also, we take the universal modalities $\mathbf{A}$ and $[\mathbf{NE}]$ as primitives, and define the existential modalities $\mathbf{E}$ and $\langle\mathbf{NE}\rangle$ as their duals as usual: $\mathbf{E}\varphi \equiv \neg\mathbf{A}\neg\varphi$ and $\langle\mathbf{NE}\rangle\varphi \equiv \neg[\mathbf{NE}]\neg\varphi$. We also use the following abbreviations: $\bot \equiv \neg\top$, $\mathbf{F}\varphi \equiv \top\,\mathbf{U}\,\varphi$, and $\mathbf{G} \equiv \neg\mathbf{F}\neg\varphi$. In addition, given a set of trees Θ, we write $\varrho \in \Theta$ if ϱ is a branch of a tree in Θ, that is, when using the notation "$\in$" between runs and sets of trees Θ, sets of trees will be seen as its associated set of branches.

Semantics. The semantics of EL formulae is given here with respect to a reactive game $G = (N, C, (C_i)_{i \in N}, (\gamma_i)_{i \in N}, X, A)$, where

$$A = (V_A, E_A, v_A^0, \Omega_A = X, \Lambda_A = Ch, \omega_A, \lambda_A).$$

The semantics of path formulae ("$\models_{\mathcal{P}}$") is essentially the same for LTL, and so given with respect to paths in A, with two additional rules for state and equilibrium formulae, respectively. The semantics of state formulae ("$\models_{\mathcal{S}}$") is given with respect to states/vertices $v \in V$ of A. The semantics of equilibrium formulae ("$\models_{\mathcal{E}}$") is given with respect to the set of Nash equilibria of G. Let ϱ be a run of A, *i.e.*, an infinite sequence of states over V_A^* starting at v_A^0, and $t \in \mathbb{N}$. Define

$$\begin{array}{lll}
(G, \varrho, t) \models_{\mathcal{P}} \psi & \text{iff} & (G, \varrho, t) \models_{\mathcal{S}} \psi \\
 & & \text{for state formulae } \psi. \\
(G, \varrho, t) \models_{\mathcal{P}} \theta & \text{iff} & (G, \varrho, t) \models_{\mathcal{E}} \theta \\
 & & \text{for equilibrium formulae } \theta. \\
(G, \varrho, t) \models_{\mathcal{P}} \neg\varphi & \text{iff} & (G, \varrho, t) \models_{\mathcal{P}} \varphi \\
 & & \text{does not hold.} \\
(G, \varrho, t) \models_{\mathcal{P}} \varphi \vee \varphi' & \text{iff} & (G, \varrho, t) \models_{\mathcal{P}} \varphi \text{ or } (G, \varrho, t) \models_{\mathcal{P}} \varphi' \\
(G, \varrho, t) \models_{\mathcal{P}} \mathbf{X}\varphi & \text{iff} & (G, \varrho, t+1) \models_{\mathcal{P}} \varphi \\
(G, \varrho, t) \models_{\mathcal{P}} \varphi \, \mathbf{U} \, \varphi' & \text{iff} & (G, \varrho, t') \models_{\mathcal{P}} \varphi' \\
 & & \text{for some } t' \geq t \text{ and} \\
 & & (G, \varrho, k) \models_{\mathcal{P}} \varphi \\
 & & \text{for all } t \leq k < t'.
\end{array}$$

The satisfaction relation "$\models_{\mathcal{S}}$" for state formulae is defined as follows:

$$\begin{array}{lll}
(G, \varrho, t) \models_{\mathcal{S}} \top & & \text{always} \\
(G, \varrho, t) \models_{\mathcal{S}} x & \text{iff} & x \in \omega_A(\varrho[t]) \\
(G, \varrho, t) \models_{\mathcal{S}} \mathbf{A}\varphi & \text{iff} & (G, \varrho', t) \models_{\mathcal{P}} \varphi \\
 & & \text{for all } \varrho' \text{ such that } \varrho[0 \dots t] \in \varrho'.
\end{array}$$

And the satisfaction relation "$\models_{\mathcal{E}}$" for equilibrium formulae is defined as follows:

$$\begin{array}{lll}
(G, \varrho, t) \models_{\mathcal{E}} [\mathbf{NE}]\varphi & \text{iff} & (G, \varrho', t) \models_{\mathcal{P}} \varphi \\
 & & \text{for all } \vec{\sigma} \text{ and } \varrho' \in [\![\vec{\sigma}]\!] \text{ such that} \\
 & & \vec{\sigma} \in NE(G) \text{ and } \varrho[0 \dots t] \in \varrho'
\end{array}$$

We say that G is a model of φ (in symbols $G \models \varphi$) if $(G, \varrho, 0) \models_{\mathcal{P}} \varphi$ for all ϱ of G, that is, for all paths or sequences of states of A starting at v_A^0—sequences in V_A^*. We also write $\models \varphi$ if $G \models \varphi$ for all models G.

With the definitions given above, it is easy to see that the three main decision problems described in the previous section, namely, Non-Emptiness, A-Nash, and E-Nash, can be expressed using EL in the following way:

1. Non-Emptiness corresponds to $\langle \mathbf{NE} \rangle \top$,
2. A-Nash corresponds to $[\mathbf{NE}]\varphi$, and
3. E-Nash corresponds to $\langle \mathbf{NE} \rangle \varphi$.

Remark 22. (Local reasoning) Since φ can be a player goal γ_i whenever they have LTL or CTL goals, some interesting questions can be naturally expressed. For instance, whether a player can always eventually achieve its goal in all Nash equilibrium computations of the system, expressed by $[\mathbf{NE}]\mathbf{GF}\gamma_i$. What is interesting is that using the knowledge of the goals the players have, one can reason about their particular (local) behaviour within the global context, which has to consider the complex concurrent behaviour of all the other independent, and rational, agents/processes in the system.

Expressivity. Observe that apart from $[\mathbf{NE}]$ all other operators are intended to have the same meaning as in CTL*. However, note that the semantics of $\mathbf{A}\varphi$ is not quite the same as its counter-part in CTL* because of the additional condition "such that $\varrho[0 \dots t] \in \varrho'$." In the standard semantics of CTL* it would be "for all ϱ' starting at $\varrho[t]$" instead. In other words, in CTL* the path used to reach the state $\varrho[t]$ is forgotten. This information is actually needed only for the semantics of the equilibrium modality $[\mathbf{NE}]$, but must be remembered throughout. Formally, we have

Proposition 23. *Let G be a game and φ be a $[\mathbf{NE}]$-free EL* formula. Then, for all runs ϱ and $t \in \mathbb{N}$, we have that*

$$(G, \varrho, t) \models \mathbf{A}\varphi \text{ iff } (G, \varrho', 0) \models \varphi,$$

for all ϱ' starting at $\varrho[t]$.

Thus, because of Proposition 23, we know that the semantics of EL* conservatively extends that of CTL*. To be more precise, the fact that we must "remember" how we got to a particular state when evaluating an equilibrium formula in that state means that, technically, EL is a *memoryfull* extension of CTL*; see, *e.g.*, [32].

More interesting is the question as to whether EL* can be translated into a logic for strategic reasoning already in the literature. This is largely an open question. However, a few observations can be made at this point. Recall that the existence of Nash equilibria in multi-player games and with respect to a class of deterministic strategies and LTL goals can be expressed in the Strategy Logic developed in [38].[3] Using their logical specification of the existence of a Nash equilibrium, we may be able to encode the $\langle\mathbf{NE}\rangle$ operator (and hence any $\langle\mathbf{NE}\rangle\psi$, with LTL ψ), if we were to restrict the setting to LTL goals, deterministic games where $V = \Omega$, and a class of deterministic strategies that can be represented in both logics, for instance, if we considered the arenas induced by iBG [20]. Under all of these restrictions we will obtain a 2EXPTIME upper bound for the model checking problem of this fragment of Equilibrium logic. Note that the fragment is not only syntactic since the underlying reactive games model to be considered is also being restricted. Letting EL*[LTL] be the (syntactic and semantic) fragment of EL* just described, the next complexity result immediately follows:

Proposition 24. *The model checking problem for EL*[LTL] formulae is in 2EXPTIME.*

6. Examples

We will now present some examples to illustrate our framework, and in particular the kinds of specifications and systems we are interested in.

Example 25. The first observation is that all usual properties that can be expressed in CTL*, *e.g.*, liveness, safety, fairness, etc., can be expressed with respect to "rational" behaviour simply by taking them to be either A-Nash or E-Nash properties. For instance, the strong fairness CTL* formula $\varphi = \mathbf{GF}x \rightarrow \mathbf{GF}x'$ (which expresses that if a request, denoted by x, is always eventually enabled, then it is always eventually served, denoted by x') can be checked against rational process behaviour with the E-Nash formula $\langle\mathbf{NE}\rangle\varphi$. More interestingly, expected behaviours from a game theoretic point of view find EL* specifications that are valid. For instance, the formula

$$\mathbf{E} \bigwedge_{i \in N} \gamma_i \rightarrow \langle\mathbf{NE}\rangle \bigwedge_{i \in N} \gamma_i$$

[3] Technically, strategies in Strategy Logic are different from ours: in their case, a strategy is a function from finite sequences of states in the arena to a particular player's choice.

expresses that if there is a run that satisfies all players' goals, then there is also a run supported by a Nash equilibrium that satisfies all players' goals. This contrasts sharply with the formula $\gamma_i \rightarrow \langle \mathbf{NE} \rangle \gamma_i$ which is, in general, not valid: for instance, a Nash equilibrium may exist only if a player, say i, has no way to get its goal achieved. For different game-theoretic reasons, the formula $\mathbf{E} \bigwedge_{i \in N} \gamma_i \rightarrow [\mathbf{NE}] \bigwedge_{i \in N} \gamma_i$ is not valid: a game can have multiple Nash equilibrium computations, not all of which need to satisfy all players' goals; this formula would be valid only if we focus on Pareto optimal runs, since in such a case any equilibrium run satisfying all goals would be preferred over any other equilibrium run. Finally, some formulae, such as $(\bigwedge_{i \in N} \mathbf{GE} \neg \gamma_i) \rightarrow [\mathbf{NE}]\varphi$, with satisfiable φ, that might seem valid are not: it may be the case that a player i has a (winning) strategy to achieve γ_i, without $\gamma_i \rightarrow \varphi$, and hence there may be some Nash equilibria where φ may not hold.

Example 26. The semantics of EL is such that the equilibrium operators $[\mathbf{NE}]$ and $\langle \mathbf{NE} \rangle$ quantify over runs that are sustained by a Nash equilibrium. As all of which start at time 0, it cannot happen that a sequence $\varrho[k, \ldots, k']$ is part of a run ϱ' that is sustained by a Nash equilibrium without $\varrho[0, \ldots, k']$ being a prefix of ϱ'. This fact is expressed by the following formula being satisfied in every model, at every run, and at every time:

$$\mathbf{EF}\langle \mathbf{NE} \rangle \top \leftrightarrow \langle \mathbf{NE} \rangle \top.$$

Observe that this is due to the type of quantification over equilibrium runs that is specific to the semantics of the equilibrium operators in EL. The above formula would also hold if subgame perfect Nash equilibrium had been our solution concept of choice. Moreover, as a direct consequence we can also conclude that, for state formulae φ,

$$(\langle \mathbf{NE} \rangle \top \wedge [\mathbf{NE}]\varphi) \rightarrow \varphi.$$

holds in all models, at all runs, and at all times.

We also have the general validity of

$$[\mathbf{NE}]\mathbf{GA}\varphi \rightarrow \mathbf{AG}[\mathbf{NE}]\varphi.$$

To see this, assume $(G, \varrho, t) \models [\mathbf{NE}]\mathbf{GA}\varphi$. Then, $(G, \varrho'', t') \models \varphi$ for all $\vec{\sigma} \in NE(G)$, all $\varrho' \in [\![\vec{\sigma}]\!]$ with $\varrho[0, \ldots, t] \in \varrho'$, all $t' \geq t$, and all runs ϱ'' with $\varrho'[0, \ldots, t'] \in \varrho''$. Now consider an arbitrary run ϱ''' with $\varrho[0, \ldots, t] \in \varrho'''$, an arbitrary $t'' \geq t$, as well as an arbitrary $\vec{\sigma}^* \in NE(G)$ and $\varrho^* \in [\![\vec{\sigma}']\!]$ such that $\varrho'''[0,\ldots,t''] \in \varrho^*$. Then also $\varrho[0,\ldots,t] \in \varrho^*$ and $t'' \geq t$. It follows that $(G,\varrho^*,t'') \models \varphi$ and, hence, $(G,\varrho,t) \models \mathbf{AG}[\mathbf{NE}]\varphi$ as well.

By contrast, however,

$$\mathbf{AG}[\mathbf{NE}]\varphi \to [\mathbf{NE}]\mathbf{GA}\varphi$$

does not generally hold. To appreciate this, consider a game G for which there is some player i who at every time can choose actions ensuring that variable x is set to true or false, respectively. This would be the case for instance in the iBG setting where i controls variable x (see Remark 13). Also assume that $\gamma_i = \mathbf{AFAG}x$, *i.e.*, player i wants to guarantee x to be true from some time in the future onwards, whereas $\gamma_j = \top$ for each player j distinct from i. Thus, a run is a Nash equilibrium if and only if it satisfies player i's goal. Let $\varphi = \mathbf{F}x$ and let ϱ be a run such that $x \notin \omega_A(\varrho[0])$ and $x \in \omega_A(\varrho[t])$ for all $t > 0$. Then, it is not hard to establish that $(G,\varrho,0) \models \mathbf{AG}[\mathbf{NE}]\mathbf{F}x$. It is not the case, however, that $(G,\varrho,0) \models [\mathbf{NE}]\mathbf{GAF}x$. To appreciate this, observe that ϱ satisfies all players' goals and that $\varrho \in [\![\vec{\sigma}]\!]$ for some Nash equilibrium $\vec{\sigma} \in \mathit{NE}(G)$. Let, moreover, ϱ' be a run such that $x \notin \omega_A(\varrho'[t])$ for all $t \geq 0$. In particular, let $\varrho'[0] = \varrho[0]$. Then, obviously, $(G,\varrho',0) \not\models \mathbf{GAF}x$. Still $\varrho'[0] \in \varrho$ and, thus, $(G,\varrho,0) \not\models [\mathbf{NE}]\mathbf{GAF}x$.

It holds almost trivially that in all games G, all runs ϱ, and at all times $\mathbf{AG}[\mathbf{NE}]\varphi \to [\mathbf{NE}]\varphi$ is satisfied. For this implication to hold with the **A**- and [**NE**]-operators interchanged, however, the existence of a Nash equilibrium is required. We thus find that

$$([\mathbf{NE}]\mathbf{GA}\varphi \wedge \langle\mathbf{NE}\rangle\top) \to \mathbf{A}\varphi.$$

is also generally satisfied.

It should be noted that the reflections above are not specific to the choice of Nash equilibrium as a solution concept. They also hold for, *e.g.*, subgame perfect equilibrium, dominant strategy equilibrium, secure equilibrium, etc.

We now make good on our promise to formalise the examples given at the beginning of the paper. The first example, apart from illustrating in detail the arena and strategies for the multi-agent system described there, highlights an important difference between the path quantifiers in CTL* and the equilibrium (path) quantifiers in EL*.

Example 27. The situation of Example 1 is modelled by the game $G = (N, C, C_1, C_2, \gamma_1, \gamma_2, X, A)$, where $X = \{x, y\}$, $C_1 = \{p\}$, $C_2 = \{q\}$ and A is the arena as in Figure 5. Intuitively, x and y signify player 1 and player 2 get the resource, respectively. Setting p to true corresponds to player 1 requesting the resource, while setting p to false means refraining from doing so. Similarly, for q and player 2. The goals of the players are the LTL goals $\gamma_1 = \mathbf{GF}x$ and $\gamma_2 = \mathbf{GF}y$. The structures in Figure 6 are two strategies σ_1 and σ_2 for player 1 and 2, respectively. Strategy σ_1 requests the resource by playing p until it gets it, and then refrains from doing so by playing $\bar{p}$ once. Strategy σ_2 toggles but between $\bar{q}$ and q, beginning with the former, and additionally threatens to set q to true for ever if p is true (at least) twice in a row, which can be deduced from x being set to true or $\bar{y}$ while having set q to true previously. The profile $\vec{\sigma} = (\sigma_1, \sigma_2)$ yields the following word languages: $\mathcal{L}_i^{\vec{\sigma}}(A) = \{w : w \in (p\bar{q}; \bar{p}q)^\omega\}$ and $\mathcal{L}_o^{\vec{\sigma}}(A) = \{w : w \in \bar{x}\bar{y}; (x\bar{y}; \bar{x}y)^\omega\}$. The run $\varrho = \bar{x}\bar{y}, x\bar{y}, \bar{x}y, x\bar{y}, \bar{x}y, \ldots$ in $\mathcal{L}_o^{\vec{\sigma}}(A)$ satisfies both players' goals and as such $\vec{\sigma}$ is a Nash equilibrium. Thus, we have $G \models \langle \mathbf{NE} \rangle(\gamma_1 \wedge \gamma_2)$. This is no coincidence, as we have for this game that $G \models [\mathbf{NE}](\gamma_1 \wedge \gamma_2)$: in all Nash equilibria both players' goals are satisfied. This contrasts sharply with the branching-time formula $\mathbf{A}(\gamma_1 \wedge \gamma_2)$, which does not hold in the game G.

This example shows that even in small concurrent systems it is not obvious that a Nash equilibrium exists—let alone that a temporal property holds in some or all equilibrium computations of the system. The example also shows an important feature of game-like concurrent systems: that even though a desirable property may not hold in general (*cf.*, $\mathbf{A}(\gamma_1 \wedge \gamma_2)$), it may well be the case that one can design or automatically synthesise a communication protocol or a synchronisation mechanism—directly from a logical specification—so that the desirable property holds when restricted to agents acting rationally (*cf.*, $[\mathbf{NE}](\gamma_1 \wedge \gamma_2)$).

Indeed, Equilibrium logics are designed to *reason about what can be achieved in equilibrium and what cannot*, while abstracting away from the particular strategies that the agents/players of the system/game may use.

Let us now, using our model for concurrent interactions, give a concrete representation of the multi-agent system initially described in Example 2.

Example 28. Consider the following strategic workflow design setting with three employees, players 1, 2, and 3, and four tasks, T_1, T_2, T_3, and T_4, none of which can be performed by a single player alone:

- At every point of time, each employee goes to one of two locations, L_1 and L_2. If two or three players meet at a location, they form a coalition;

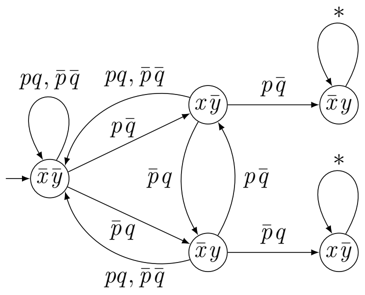

Figure 5: Formal model of the system in Example 1.

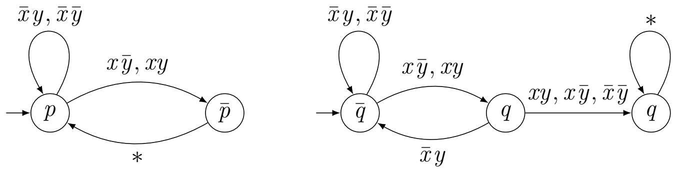

Figure 6: Strategies σ_1 (left) and σ_2 (right).

- When formed, coalitions $\{1, 2\}$, $\{1, 3\}$, $\{2, 3\}$, and $\{1, 2, 3\}$ will perform tasks T_1, T_2, T_3, and T_4, respectively;
- There is a manager who wants to see T_4 performed infinitely often.

We find that by burdening all employees with the responsibility of having T_4 performed infinitely often does not result in the manager's objective being achieved in all equilibria. Interestingly, however, we find below that there are other ways of assigning responsibilities which do ensure that the task T_4 is performed infinitely often in all equilibria.

The setting described above can be formalised as a game-like concurrent system $G = (N, C, C_1, C_2, C_3, \gamma_1, \gamma_2, \gamma_3, X, A)$. We assume the players 1, 2, and 3 each control one variable, *i.e.*, $C_1 = \{p\}$, $C_2 = \{q\}$, and $C_3 = \{r\}$. Intuitively, if a player sets his variable to true, he will go to location L_1 and if he sets his variable to false, to location L_2. We have $X = \{x, y\}$ and introduce the following abbreviations:

$$T_1 = x \wedge y \qquad T_2 = x \wedge \neg y \qquad T_3 = \neg x \wedge y \qquad T_4 = \neg x \wedge \neg y.$$

The arena A is depicted in Figure 7. Observe, in particular, that the manager in the workflow system is not a player in the model.

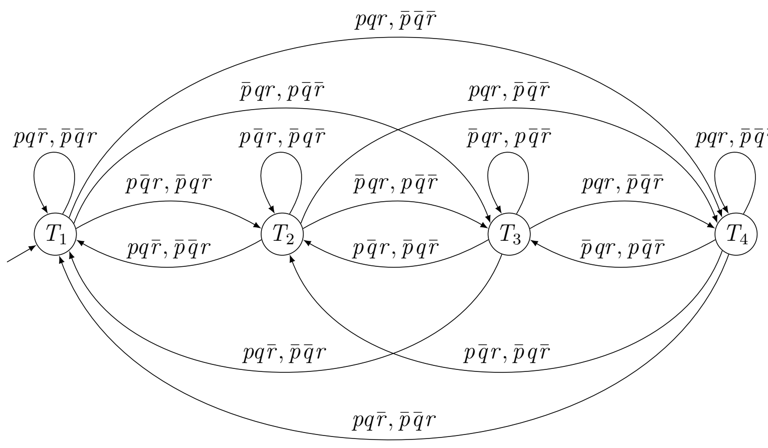
$pqr, \bar{p}\bar{q}\bar{r}$
$\bar{p}qr, p\bar{q}\bar{r}$
$pqr, \bar{p}\bar{q}\bar{r}$
$pq\bar{r}, \bar{p}\bar{q}r$
$p\bar{q}r, \bar{p}q\bar{r}$
$\bar{p}qr, p\bar{q}\bar{r}$
$pqr, \bar{p}\bar{q}\bar{r}$
$p\bar{q}r, \bar{p}q\bar{r}$
$\bar{p}qr, p\bar{q}\bar{r}$
$pqr, \bar{p}\bar{q}\bar{r}$
T_1
T_2
T_3
T_4
$pq\bar{r}, \bar{p}\bar{q}r$
$p\bar{q}r, \bar{p}q\bar{r}$
$\bar{p}qr, p\bar{q}\bar{r}$
$pq\bar{r}, \bar{p}\bar{q}r$
$p\bar{q}r, \bar{p}q\bar{r}$
$pq\bar{r}, \bar{p}\bar{q}r$

Figure 7: The arena representing the system in Example 28.

First, consider the setting in which the players are all assigned responsibility for T_4 being performed infinitely often, *i.e.*, let the preferences of the players be given by the following goals:

$$\gamma_1 = \gamma_2 = \gamma_3 = \mathbf{GF}\, T_4.$$

For better readability we let γ^* abbreviate $\mathbf{GF}\, T_4$. As there are runs in which T_4 is performed infinitely often, which then also satisfy all players' goals, we obviously have $G \models \langle \mathbf{NE} \rangle \gamma^*$. Nonetheless, $G \not\models [\mathbf{NE}]\gamma^*$ and, a fortiori, $G \not\models \mathbf{A}\gamma^*$. To see this, consider the strategy profile $\vec{\sigma} = (\sigma_1, \sigma_2, \sigma_3)$ in Figure 8, which gives rise to T_1 being performed at every point of time. Moreover, it can easily be seen that $\vec{\sigma}$ is a Nash equilibrium of the system: if player 1 deviates the outcomes of the game are runs that will lie within $(T_1 \cup (T_1; T_2))^\omega$; if player 2 deviates, within $T_1; (T_1 \cup T_3)^\omega$; and if player 3 deviates, within $T_1^\omega \cup (T_1^+; T_4; (T_2 \cup T_3)^\omega)$. Since none of these runs satisfies $\mathbf{GF}\, T_4$, the strategy profile $\vec{\sigma}$ is indeed a Nash equilibrium of the game.

Now, assume that the players have the following, more complex, goals:

$$\begin{aligned}
\gamma_1' &= \mathbf{G}(T_4 \to \neg \mathbf{X}\, T_3) \wedge (\mathbf{GF}\, T_1 \vee \mathbf{GF}\, T_4)\\
\gamma_2' &= \mathbf{G}((T_3 \to \neg \mathbf{X}(T_1 \vee T_2)) \wedge (T_4 \to \neg \mathbf{X}\, T_2)) \wedge (\mathbf{GF}\, T_3 \vee \mathbf{GF}\, T_4)\\
\gamma_3' &= \mathbf{G}(T_1 \to \neg \mathbf{X}\, T_4)) \wedge (\mathbf{GF}\, T_2 \vee \mathbf{GF}\, T_4)
\end{aligned}$$

The idea behind these goals is that there is an intended order in which the tasks T_1 through T_4 are to be performed: T_4 has to be preceded by T_3,

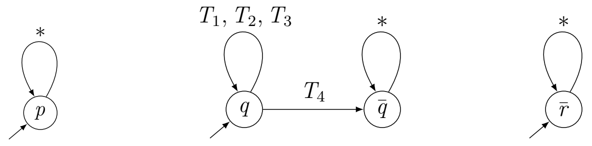

Figure 8: The strategies σ_1 (left), σ_2 (middle), and σ_3 (right) as in Example 28

and T_3 in turn has to be preceded by either T_1 or T_2. This order is then to be repeated over time and each player is responsible for part of this sequence. Player 1 is responsible for T_3 never happening immediately after T_4, and for either T_4 or T_1 being performed infinitely often. Player 2 should see to it that neither T_1 nor T_2 immediately follows T_3, and that T_4 is not followed by T_2. Moreover, player 2 has to ensure that either T_3 or T_4 are preformed infinitely often. Finally, player 3's goal is that T_4 never occurs immediately after T_1, and that either T_2 or T_4 is performed infinitely often. We find that in the corresponding game $G' = (N, C, C_1, C_2, C_3, \gamma'_1, \gamma'_2, \gamma'_3, X, A)$ we have

$$G' \models [\mathbf{NE}]\gamma^*.$$

The key to this insight is the observation that, *given any strategies σ_j and σ_k for the other two players*, for each player i, there is a strategy σ^*_i such that $(\sigma^*_i, \sigma_j, \sigma_k) \models \gamma'_i$, i.e., player i can deviate unilaterally and achieve his goal γ_i. For instance, fix strategies σ_2 and σ_3 of players 2 and 3, respectively. Then, for player 1 a strategy σ^*_1 can be designed in such a way that it mimics the behaviour of the arena A and the strategies σ_2 and σ_3, while keeping track of its own choices. In this way, σ^*_1 can predict how players 2 and 3 will set their variables at every time and, on basis of this information, output a suitable value for r. More concretely, whenever, at some time t, players 2 and 3 are going to set both q and r to true, σ^*_1 outputs p. Analogously, if players 2 and 3 are going to set both q and r to false. Thus, T_4 will be true at t. On the other hand, if at t players 2 and 3 will choose opposite values for q and r, then σ^*_1 sets p to the same value as player 2 to q. Thus, T_1 will hold at t.

Observe that in the run induced by $(\sigma^*_1, \sigma_2, \sigma_3)$ either T_1 or T_4 or both will be output by the arena infinitely often, *i.e*, $(\sigma^*_1, \sigma_2, \sigma_3) \models \mathbf{GF}\,T_1 \vee \mathbf{GF}\,T_4$. Moreover, neither T_2 nor T_3 will ever be output by the arena. Accordingly, trivially, we have that $(\sigma^*_1, \sigma_2, \sigma_3) \models \mathbf{G}(T_4 \to \neg\mathbf{X}\,T_3)$ holds. Then, we can conclude that $(\sigma^*_1, \sigma_2, \sigma_3) \models \gamma'_1$. As similar arguments hold for players 2 and 3, it follows that in every equilibrium of G' the goals of all three players will be satisfied. Moreover, it can easily be established that the runs that satisfy all three players' goals are characterised by the regular expression $((T_1 \cup T_2)^+; T_3^+; T_4^+)^\omega$, and as such also satisfy the manager's objective γ^*.

7. Complexity Lower Bounds

In this section we show that the model checking problem for Equilibrium logics is 2EXPTIME-hard, even for LTL or CTL players' goals. More precisely, the model checking problem for Equilibrium logics is stated as follows:

> *Given*: Game G, EL* formula φ.
> EL* MODEL-CHECKING: Is it the case that $G \models \varphi$?

In fact, we will show a very strong claim: that the EL* MODEL-CHECKING problem is 2EXPTIME-hard, even for the EL* formula $\varphi = \langle \mathbf{NE} \rangle \top$ (the simplest equilibrium logic formula one can write) and for either LTL or CTL goals (two of the simplest temporal logics in the literature).

To show 2EXPTIME-hardness of checking equilibrium properties given that the players of the game have linear-time goals, we use a reduction from the two-player perfect-information LTL games in [5].

Formally, we have the following lemma:

Lemma 29. *The* EL* MODEL-CHECKING *problem where players' goals are LTL formulae is 2EXPTIME-hard.*

Now, in order to show the 2EXPTIME-hardness of checking equilibrium properties given that the players of the game have *branching-time goals*, we use a reduction from the control-synthesis problem with respect to reactive environments [31]. The control-synthesis problem for reactive environments is similar to the LTL games in [5], except that one has to consider CTL goals instead of LTL goals, and non-deterministic strategies instead of deterministic ones—just as defined in our EL framework.

Formally, we have the following lemma:

Lemma 30. *The* EL* MODEL-CHECKING *problem where players' goals are CTL formulae is 2EXPTIME-hard.*

Since LTL and CTL are syntactic fragments of the linear-time μ-calculus and the modal μ-calculus, respectively, the hardness results transfer.

Corollary 31. *The* EL* MODEL-CHECKING *problem, where players' goals are linear-time μ-calculus or modal μ-calculus formulae, is 2EXPTIME-hard.*

Notice that model checking the equilibrium operators of EL* formulae may require the solution of a number of "internal" synthesis problems for $(\gamma_i)_{i \in N}$ so that $\vec{\sigma} \in NE(G)$ can be checked and a run $\varrho \in [\![\vec{\sigma}]\!]$ can be determined before an EL* formula can be checked. This problem, known as reactive synthesis [31], is 2EXPTIME-complete for both LTL and CTL specifications, and appears to be the main source of the high complexity of checking equilibrium properties. A discussion of the relationships between the verification of equilibrium properties and the solution of control and synthesis problems is appropriate, and therefore given in Section 9. We also discuss other temporal logics for strategic reasoning, in particular, their expressivity and the complexity of their model checking problems. We would also like to remark that, under our definition of satisfaction of branching-time goals, the use of non-deterministic strategies is necessary: if non-deterministic strategies were not allowed, then some of our hardness results would not be achievable, as one could not encode control-synthesis games in our framework.

8. Towards More Stable Strategic Reasoning

Undoubtedly, Nash equilibrium is the most prominent solution concept in game theory, and many relevant solution concepts in the literature are either generalisations or refinements of it. However, there are well known weaknesses of Nash equilibria; for instance, it is not guaranteed to be unique, it is arguably unstable (nothing is ensured if, *e.g.*, more than one player deviates or irrational moves are made), and it does not account for dynamic behaviour, amongst others. For this reason, we now explore the complexity of model checking EL formulae with respect to *dominant strategy equilibrium*, which is a solution concept known to be much more stable than Nash equilibrium; indeed, dominant strategy equilibrium ensures a player's best response *regardless of the behaviour of the other players.* Thus, unlike Nash equilibria, dominant strategies behave well even with respect to "irrational" moves of other players, or with respect to deviations of multiple players.

8.1. Dominant Strategies

Dominant strategy equilibrium [43] is a very appealing solution concept because of its stability/robustness: it defines a best response for each player, no matter how other players in the game behave. Formally, a strategy profile $\vec{\sigma} = (\sigma_1, \ldots, \sigma_i, \ldots, \sigma_n)$ is a *Dominant Strategy equilibrium* if for every strategy profile $(\sigma'_1, \ldots, \sigma'_n)$ and every player i, we have

$$(\sigma'_1, \ldots, \sigma'_i, \ldots, \sigma'_n) \leq_i (\sigma'_1, \ldots, \sigma_i, \ldots, \sigma'_n).$$

Intuitively, a dominant strategy equilibrium formalises the idea that no player can be better off by switching to a different strategy, no matter which strategies the other players in the game choose. Let $DS(G)$ be the set of dominant strategy equilibria of a game G.

Since equilibrium logics can be parametrised with a solution concept, we can consider equilibrium formulae with respect to dominant strategy equilibria; we now do so. We shall write $[\mathbf{DS}]\varphi$ for formulae, whose semantics is given with respect to dominant strategy equilibrium paths as follows:

$$\begin{array}{lll} (G, \varrho, t) \models_{\mathcal{E}} [\mathbf{DS}]\varphi & \text{iff} & (G, \varrho', t) \models_{\mathcal{P}} \varphi \\ & & \text{for all } \vec{\sigma} \text{ and } \varrho' \in [\![\vec{\sigma}]\!] \text{ such that} \\ & & \vec{\sigma} \in DS(G) \text{ and } \varrho[0 \ldots t] \in \varrho' \end{array}$$

Indeed, model checking equilibrium properties when considering dominant strategy equilibria is, as for Nash equilibria, a 2EXPTIME-hard problem. The proof of this result is obtained via a variation of the proof of Lemma 29. Essentially, via a reduction from the LTL games in [5] where the goals of the players in the constructed reactive game for dominant strategy equilibria are slightly different from the goals in the reactive game for Nash equilibria. Formally, we have the following result.

Lemma 32. EL* Model-Checking *for dominant strategy equilibrium, with players' goals given by LTL formulae is 2EXPTIME-hard.*

Obviously, this hardness result automatically transfers to reactive games where players' goals are given by linear-time μ-calculus formulae. And, for the same reasons given in the Nash equilibrium case, the hardness result also extends to the branching-time framework.

Formally, we have the following corollary:

Corollary 33. EL* Model-Checking *for dominant strategy equilibrium, with players' goals given by CTL formulae is 2EXPTIME-hard.*

Before we finish this section, we would like to remark that the technique used to obtain the 2EXPTIME-hardness results in this and the previous section, that is, via reductions from the games in [5, 31] will be likely to apply to other solutions concepts provided that the goals of the players in the constructed reactive game can be expressed as LTL/CTL formulae. For the sake of clarity and completeness, the general proof technique is outlined in the appendix, where the overall construction of the reactive game is described.

8.2. Equilibria Beyond Binary Pay-off Sets

As mentioned in the previous section, one way of addressing the issue of having a reasoning framework that is strategically unstable is by investigating solution concepts that are more stable than Nash equilibrium, for instance dominant strategies. Two other ways of trying to tackle this problem are to consider mixed strategies or fully quantitative pay-off sets (so far we have only considered binary pay-off sets given by a single temporal logic goal). However, as shown in [50, 49], considering either mixed strategies or full quantitative outcomes for a game immediately leads to undecidability results. Then, unfortunately, because of those undecidability results, our reasoning framework may be one of the most general ones that we could hope for.

Because of this computational limitation a number of intermediate solutions have been investigated. For instance, there is a fairly general multi-objective setting where decidable games can be defined: one where players are given not just a single temporal logic formula, but an ordered set of temporal logic goals. In this section, we discuss such a case. Formally, the new games are defined as follow. A *reactive game with ordered goals* is a structure:

$$G = (N, C, (C_i)_{i \in N}, (\Gamma_i)_{i \in N}, X, A)$$

where $\Gamma_i = \{\gamma_i^1, \ldots, \gamma_i^k\}$ is a set of size $k \in \mathbb{N}$, such that each $\gamma_i^m \in \Gamma_i$ is a temporal logic formula satisfying that $[\![\gamma_i^m]\!] <_i [\![\gamma_i^{m+1}]\!]$, with $1 \leq m < k$.

It should be easy to see that all of the problems we have studied for EL (in Sections 5, 7, and 8.1) are also 2EXPTIME-hard in this setting, since having only one single temporal logic goal for each player in the game is just a special case, that is, the case where Γ_i is a singleton set for each i. It is also not hard to show that the decision problems we studied in Section 4 remain within the same complexity classes. All one needs to do is to check, using the decision procedures already defined in Section 4, if a unilateral and beneficial deviation for each player i is possible with respect to any goal in Γ_i.

Lemma 34. *The complexity of* NE-Checking, Equivalence-Checking, *and* EL Model-Checking *with respect to reactive games with ordered goals is the same as the complexity of the same problems for reactive games.*

Note that in the definition of a reactive game with ordered sets, we have chosen a *strict* order: for each i, we have that $[\![\gamma_i^m]\!] <_i [\![\gamma_i^{m+1}]\!]$. However, note that because the complexities of the problems in Section 4 are all either PSPACE or EXPTIME, the way in which the goals of each player i are ordered within Γ_i can, in general, be using any *preorder* relation. Then, although the multi-objective setting we have defined in this section may not be as general and stable as those where either mixed strategies or quantitative pay-off sets are used, a multi-objective setting would substantially help mitigate the lack of stability when using Nash equilibria—interestingly, without paying an additional (worst-case) computational complexity cost.

9. Concluding Remarks and Related Work

Equilibrium checking vs. Control-synthesis problems. The main results of this paper clearly emphasise that model checking equilibrium properties is a computationally difficult problem (at least 2EXPTIME-hard), in particular, because it involves the solution of a control-synthesis problem: in order to model check an equilibrium logic formula, regardless of the solution concept under consideration, one may first have to synthesise a set of "controllers" (strategies)—see the semantics of equilibrium logic formulae. Then, the two problems, namely, equilibrium checking and control-synthesis are very closely related. A good reference for results and analyses of the relationships between these two problems can be found in [12]. On the other hand, as discussed in [30], the high computational complexity of solving synthesis problems may be, in fact, rather misleading, since specifications are most usually rather small when compared with systems to be model checked.

Complexity. We have shown that checking the equilibrium properties of a concurrent and multi-agent system is a computationally hard problem as any interesting property is at least in PSPACE. On the positive side, we have also shown that, in most cases, the difficulty of checking equilibrium properties is independent of whether the goals of the players are given by linear-time or branching-time formulae. Table 1 summarises our main complexity results.

As an aside, we should point out that there is substantial work on the complexity of Nash equilibrium in the algorithmic game theory community [42]. A central question in this community was the complexity of computing mixed (randomized) Nash equilibria in 2-person strategic games. This problem

	St–Check	NE–Check	Eq–Check	EL*–MC[NE]	EL*–MC[DS]
LTL	PSPACE	PSPACE	PSPACE	2EXPT−h	2EXPT−h
CTL	PSPACE	EXPT	PSPACE	2EXPT−h	2EXPT−h
TL_μ	PSPACE	PSPACE	PSPACE	2EXPT−h	2EXPT−h
L_μ	inEXPT	EXPT	inEXPT	2EXPT−h	2EXPT−h

Table 1: Summary of computational complexity results. In this table, TL_μ stands for the linear-time μ-calculus, L_μ for the modal μ-calculus, St–Check for Strategy-Checking, NE–Check for NE-Checking, Eq–Check for Equivalence-Checking, EL*–MC for EL* Model Checking, inEXPT for in EXPTIME, and 2EXPT−h for 2EXPTIME-hard.

received considerable attention before it was finally shown to be PPAD-complete in 2006 [11]. Our work is similar in spirit, but differs in many important respects. First, we are here concerned with *pure* strategy Nash equilibria, as opposed to mixed strategy equilibria. Second, our model of preferences is dichotomous, based on the use of temporal formulae γ_i to specific desirable temporal structures (paths or trees) that players wish to see satisfied—this contrasts with the utility-theoretic preference model of strategic games. Third, many of our problems implicitly embody highly complex temporal reasoning problems (such as synthesis or satisfiability) and the complexity of these problems tends to dominate the complexity of the strategic reasoning problems we consider. Finally, we are concerned with developing *logical languages for reasoning about temporal equilibrium properties*, which is not typically considered in algorithmic game theory.

Concurrency. There are a number of concurrency models similar to ours. We will mention a few that are relevant with respect to our model. The concurrent systems in [29] are defined by a set of independent sequential processes and, based on them, an interleaving global system is determined. In this model, all behaviours represented by the global system are allowed behaviours in the underlying concurrent processes. There are two main differences with our model. Firstly, because our processes have a strategic interpretation, they carry more structure than plain labelled transition systems. More specifically, our processes are machines that must behave, locally, as controllers in order to resolve non-deterministic choices in the global system. Secondly, we explicitly represent the global system by an arena (which has a game-theoretic interpretation), and such a global system only represents *potential* behaviours of the concurrent system; the particular reactive behaviour of the concurrent system is known only when strategies over such an arena are given. Similar considerations hold with respect to other models such as synchronous and asynchronous products of transitions systems [17].

Other models, like ours, come with game-theoretic interpretations: see, for instance, [2, 40]. These models are also transition-based frameworks and have been used for modelling and verification (synthesis and model-checking). Unlike ours, none of these models has a notion of control over variables. This notion, as used in our model, stems from models for AI and multi-agent systems, such as those similar to, or based on, the model of distributed control in Boolean games [25]. The interaction between players (processes/strategies) and the arena (global system) is also technically different from our model. In [40] there is a unique and hostile environment; we do not make such assumptions. On the other hand, in [2], the treatment of strategies more closely relates to traditional presentations of strategies as functions between histories of partial plays. Instead, our model formally justifies both all interactions between players and observable behaviours of the global system via manipulations of languages, for both word languages and tree languages. Because of this, we believe there is room for extensions of our model to the semantically finer "noninterleaving" concurrency framework via a generalisation to Mazurkiewicz trace languages [41]. Finally, good references to other game-based models for studying concurrent systems can be found in [19, 52].

Logic. Our work relates to logics either that are to reason about the behaviour of game-like systems or that are memoryfull. In [32] a memoryfull branching-time logic, called mCTL*, that extends CTL* was introduced. This logic has a special atomic proposition ("present") that allows one to reason about computations starting in the present or computations starting somewhere in the past (in CTL* all formulae are interpreted with respect to computations that start in the present). This logic is not more powerful than CTL*, but can be exponentially more succinct. No equilibrium issues are addressed there. In the linear-time framework, memoryfull logics have also been studied. In [33] and extension of LTL with past is extended with an operator ("Now") that allows one to "forget the past" when interpreting logical formulae. This logic is exponentially more succinct than LTL with past, which in turn is exponentially more succinct than LTL; all such logics are nevertheless equally expressive. Thus, it seems that adding memory to temporal logics, either in the linear-time or in the branching-time framework, can make them more succinct but not any more powerful. Another common feature of memoryfull temporal logics is that their verification problem is at least EXPSPACE. It follows from our 2EXPTIME-hardness results that checking equilibrium properties, which requires the use of a memoryfull logic, is a harder problem.

Another memoryfull logic, this time one designed to reason about game-like scenarios, is Strategy Logic [7]. Strategy Logic has strategies as first-class objects in the language and based on this feature one can indirectly reason about equilibrium computations of a system. In Equilibrium logic reasoning is done the other way around: we can directly refer to equilibrium computations of a system, and based on them reason about the strategic power of the players in a game. Verification is extremely hard in Strategy Logic too: it is $(d{+}1)$EXPTIME-complete, where d is the alternation between universal and existential strategy quantifiers in the logic, which has LTL as base language—a logic in the linear-time framework. Strategy Logic as defined in [7] is a logic with a two-player semantic game. This logic was later on extended to the multi-player setting in [38]. Recent works, for instance [36, 37], have focused on the discovery of syntactic fragments of Strategy Logic which are expressive enough, *e.g.*, to express the existence of equilibria, and yet with decidable satisfiability and model checking problems.

Finally, in [39] a memoryfull extension of ATL, called mATL*, is studied. Again, this logic identifies the need for memoryfull power when reasoning about strategic interactions. Since mATL* extends ATL, it is an alternating-time temporal logic [2]. This logic, as mCTL*, extends ATL* with a "present" proposition to differentiate between (and reason about) computations that start in the present state of play and computations that start far in the past—in a state previously visited. The verification problem for this logic, as it is for ATL*, is in 2EXPTIME. In addition, such logics (bisimuliation-invariant logics) cannot, in general, express the existence of Nash equilibria in multi-player games [22] when considering deterministic concurrent game structures as arenas, and strategies defined as functions from finite sequences of arena states to players' actions.

10. Future work

All complexity results in this paper are given for systems and players' goals in their full generality. It would be interesting to discover *"easy" classes of reactive games* for which the decision problems studied in the paper are computationally simpler. Because of the nature of the problems at hand, all relevant questions about equilibria are expected to be at least NP-hard.

Another interesting avenue for further work is the study of *other game-theoretic solution concepts*. For instance, solution concepts such as correlated or subgame-perfect Nash equilibrium, which are considered to have more attractive theoretical properties when compared with Nash equilibrium. In this paper, apart from Nash equilibrium, we also studied dominant strategy equilibria since they are known to be much more stable than Nash equilibria, and hence may characterise equilibrium computation paths which can be regarded as being more robust from a game theoretic point of view.

Certain features of our framework and decision problems related to equilibrium logics were only lightly discussed in this paper and, thus, could be studied in more detail. For instance, the study of richer preference relations as well as quantitative and imperfect information, which are well beyond the scope of this particular paper. From a more logical point of view, we would like to have a better understanding of the *expressivity of equilibrium logics*. They certainly offer a different paradigm for reasoning about equilibria in infinite, concurrent and multi-agent systems modelled as games, but whether such temporal logics can be translated into one of those already in the literature (or the other way around) is an open and interesting question.

Finally, our results naturally lead to further algorithmic solutions and a tool implementation, for instance, as done using temporal logics such as ATL and probabilistic variants of CTL* in model checkers such as PRISM [8], MCMAS [34], or MOCHA [4], just to name a few available online. In fact, a tool implementation of a solution to NE-Checking is presented in [48, 53] for the case where arenas and strategies are represented using the Reactive Modules specification language [3] and goals using CTL formulae. However, an implementation of the whole framework presented here is still needed.

Some of the questions and problems raised in this section are, in fact, already within reach. However, they are beyond the scope of this paper as each deserves a long and detailed presentation. Here we have, nevertheless, laid the foundations of our general framework and studied the most important computational and model-theoretic questions for equilibrium logics.

Acknowledgements. We thank the anonymous reviewers of the conference version [21] of this paper for their helpful comments. We also acknowledge the support of the ERC Advanced Grant 291528 ("RACE") at Oxford.

References

[1] Abramsky, S.: Semantics of interaction: an introduction to game semantics. In: Newton Institute Publications. Cambridge University (1996)

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

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

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

[5] Alur, R., La Torre, S., Madhusudan, P.: Playing games with boxes and diamonds. In: CONCUR. LNCS, vol. 2761, pp. 127–141. Springer (2003)

[6] Browne, M.C., Clarke, E.M., Grumberg, O.: Characterizing finite Kripke structures in propositional temporal logic. Theoretical Computer Science 59, 115–131 (1988)

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

[8] Chen, T., Forejt, V., Kwiatkowska, M.Z., Parker, D., Simaitis, A.: PRISM-games: a model checker for stochastic multi-player games. In: TACAS. LNCS, vol. 7795, pp. 185–191. Springer (2013)

[9] Clarke, E.M., Emerson, E.A.: Design and synthesis of synchronization skeletons using branching time temporal logic. In: Logics of Programs. LNCS, vol. 131, pp. 52–71. Springer-Verlag: Berlin, Germany (1981)

[10] Clarke, E.M., Grumberg, O., Peled, D.A.: Model Checking. The MIT Press: Cambridge, MA (2000)

[11] Daskalakis, C., Goldberg, P.W., Papadimitriou, C.H.: The complexity of computing a Nash equilibrium. In: STOC. Seattle, WA (2006)

[12] Ehlers, R., Lafortune, S., Tripakis, S., Vardi, M.: Reactive synthesis vs. supervisory control: Bridging the gap. Tech. Rep. UCB/EECS-2013-162, EECS Department, University of California, Berkeley (2013)

[13] Emerson, E.A.: Temporal and modal logic. In: Handbook of Theoretical Computer Science Volume B: Formal Models and Semantics, pp. 996–1072. Elsevier Science Publishers B.V.: Amsterdam (1990)

[14] Emerson, E.A., Halpern, J.Y.: 'Sometimes' and 'not never' revisited: on branching time versus linear time temporal logic. Journal of the ACM 33(1), 151–178 (1986)

[15] Emerson, E.A., Halpern, J.Y.: Decision procedures and expressiveness in the temporal logic of branching time. Journal of Computer and System Sciences 30(1), 1–24 (1985)

[16] Emerson, E.A., Jutla, C.S.: The complexity of tree automata and logics of programs (extended abstract). In: FOCS. pp. 328–337 (1988)

[17] Esparza, J., Heljanko, K.: Unfoldings - A Partial-Order Approach to Model Checking. EATCS Monographs in TCS, Springer (2008)

[18] Gabbay, D., Gärdenfords, P., Siekmann, J., Benthem, J.V., Vardi, M., Woods, J. (eds.): Handbook of Modal Logic, vol. 3. Elsevier (2007)

[19] Ghica, D.R.: Applications of game semantics: From program analysis to hardware synthesis. In: LICS. pp. 17–26. IEEE Computer Society (2009)

[20] Gutierrez, J., Harrenstein, P., Wooldridge, M.: Iterated Boolean games. In: IJCAI. IJCAI/AAAI (2013)

[21] Gutierrez, J., Harrenstein, P., Wooldridge, M.: Reasoning about equilibria in game-like concurrent systems. In: KR. AAAI (2014)

[22] Gutierrez, J., Harrenstein, P., Wooldridge, M.: Expresiveness and complexity results for strategic reasoning. In: CONCUR. LIPIcs, vol. 42, pp. 268–282. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik (2015)

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

[24] Habermehl, P.: On the complexity of the linear-time mu-calculus for Petri-nets. In: ICATPN. LNCS, vol. 1248, pp. 102–116. Springer (1997)

[25] Harrenstein, P., van der Hoek, W., Meyer, J.J., Witteveen, C.: Boolean games. In: Theoretical Aspects of Rationality and Knowledge (TARK VIII). pp. 287–298 (2001)

[26] Hyland, M.: Game semantics. In: Newton Institute Publications. Cambridge University (1997)

[27] Kozen, D.: Lower bounds for natural proof systems. In: FOCS. pp. 254–266. IEEE Computer Society (1977)

[28] Kozen, D.: Results on the propositional mu-calculus. Theoretical Computer Science 27, 333–354 (1983)

[29] Kupferman, O., Vardi, M.Y., Wolper, P.: An automata-theoretic approach to branching time model checking. Journal of the ACM 47(2), 312–360 (Mar 2000)

[30] Kupferman, O.: Recent challenges and ideas in temporal synthesis. In: SOFSEM. LNCS, vol. 7147, pp. 88–98. Springer (2012)

[31] Kupferman, O., Madhusudan, P., Thiagarajan, P.S., Vardi, M.Y.: Open systems in reactive environments: Control and synthesis. In: CONCUR. LNCS, vol. 1877, pp. 92–107. Springer (2000)

[32] Kupferman, O., Vardi, M.Y.: Memoryful branching-time logic. In: LICS. pp. 265–274. IEEE Computer Society (2006)

[33] Laroussinie, F., Markey, N., Schnoebelen, P.: Temporal logic with forgettable past. In: LICS. pp. 383–392. IEEE Computer Society (2002)

[34] Lomuscio, A., Qu, H., Raimondi, F.: MCMAS: a model checker for the verification of multi-agent systems. In: CAV. LNCS, vol. 5643, pp. 682–688. Springer (2009)

[35] MacLane, S.: Categories for the Working Mathematician. Springer (1998)

[36] Mogavero, F., Murano, A., Perelli, G., Vardi, M.Y.: What makes ATL* decidable? A decidable fragment of strategy logic. In: CONCUR. LNCS, vol. 7454, pp. 193–208. Springer (2012)

[37] Mogavero, F., Murano, A., Sauro, L.: On the boundary of behavioral strategies. In: LICS. pp. 263–272. IEEE Computer Society (2013)

[38] Mogavero, F., Murano, A., Vardi, M.Y.: Reasoning about strategies. In: FSTTCS. LIPIcs, vol. 8, pp. 133–144. Schloss Dagstuhl (2010)

[39] Mogavero, F., Murano, A., Vardi, M.Y.: Relentful strategic reasoning in alternating-time temporal logic. In: LPAR. LNCS, vol. 6355, pp. 371–386. Springer (2010)

[40] Mohalik, S., Walukiewicz, I.: Distributed games. In: FSTTCS. LNCS, vol. 2914, pp. 338–351. Springer (2003)

[41] Nielsen, M., Winskel, G.: Models for concurrency. In: Handbook of Logic in Computer Science, pp. 1–148. Oxford University Press (1995)

[42] Nisan, N., Roughgarden, T., Tardos, E., Vazirani, V.V. (eds.): Algorithmic Game Theory. Cambridge University Press (2007)

[43] Osborne, M.J., Rubinstein, A.: A Course in Game Theory. The MIT Press: Cambridge, MA (1994)

[44] Parikh, R.: The logic of games and its applications. In: Topics in the Theory of Computation. Elsevier (1985)

[45] Pnueli, A.: The temporal logic of programs. In: FOCS. pp. 46–57. IEEE (1977)

[46] Pnueli, A., Rosner, R.: On the synthesis of a reactive module. In: POPL. pp. 179–190. ACM Press (1989)

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

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

[49] Ummels, M., Wojtczak, D.: The complexity of nash equilibria in limit-average games. In: CONCUR. LNCS, vol. 6901, pp. 482–496. Springer (2011)

[50] Ummels, M., Wojtczak, D.: The complexity of nash equilibria in stochastic multiplayer games. Logical Methods in Computer Science 7(3) (2011)

[51] Vardi, M.Y.: A temporal fixpoint calculus. In: POPL. pp. 250–259. ACM Press (1988)

[52] Walukiewicz, I.: A landscape with games in the background. In: LICS. pp. 356–366. IEEE Computer Society (2004)

[53] Wooldridge, M., Gutierrez, J., Harrenstein, P., Marchioni, E., Perelli, G., Toumi, A.: Rational verification: From model checking to equilibrium checking. In: AAAI. pp. 4184–4190. AAAI (2016)

Appendix A. Proofs

Proposition 15. *Our models for arena games and players' strategies, with the structure-preserving morphism given in Definition 14, form a category.*

Proof. The morphism m preserves initial states (rule 1) and transitions, whenever α is defined (rule 2)—hence it preserves reachable states, *i.e.*, if v is reachable in M then $\kappa(v)$ is reachable in M'. Moreover, the identity morphism is the map where κ is the identity function on states, α is the identity function on transitions, and β is the identity function on the powerset of Ω. Finally, the composition of two morphisms m, m', where $m : M \to M'$ and $m' : M' \to M''$ is the map denoted by $m' \odot m : M \to M''$ where each internal map in m and m' composes as functions compose. It then follows that the composition of morphisms is reflexive and transitive, as needed. $\square$

Lemma 18. *The* Strategy-Checking *problem for LTL, CTL, and the linear-time μ-calculus is PSPACE-complete, and it is in EXPTIME for the modal μ-calculus. Moreover, the problem is PSPACE-hard even for formulae ψ of the form $\psi = \mathbf{F}\varphi$, where φ is a propositional logic formula.*

Proof. To check if $\vec{\sigma} \models \psi$ we need to check whether $Out(G, \vec{\sigma}) \subseteq [\![\psi]\!]$. Because of the definitions of outcomes and composition of strategies, we know that both $Out(G,\vec{\sigma}) \subseteq \mathcal{L}_o(A)$ and $Out(G,\vec{\sigma}) \subseteq \{\omega_A(q_A^0);\varrho \mid \varrho \in \mathcal{L}_i(\vec{\sigma})\}$—because of the alternation between the strategies and the arena.

Then, it follows that

$$Out(G,\vec{\sigma}) \subseteq (\mathcal{L}_o(A) \cap \omega_A(v^0);\mathcal{L}_i(\vec{\sigma})),$$

where $\omega_A(v^0);\mathcal{L}_i(\vec{\sigma})$ stands for $\{\omega_A(v^0);\varrho \mid \varrho \in \mathcal{L}_i(\vec{\sigma})\}$.

On the other hand, we know that

$$\mathcal{L}_i^{\vec{\sigma}}(A) = \mathcal{L}_o(\vec{\sigma})|_{\mathcal{L}_i(\vec{\sigma})\cap\mathcal{L}_o(A)[\vec{q}^0]}$$

where $\vec{q}^0 = (\tau_1(q_1^0),\ldots,\tau_n(q_n^0))$, and that

$$\mathcal{L}_o^{\vec{\sigma}}(A) = \mathcal{L}_o(A)|_{\mathcal{L}_i^{\vec{\sigma}}(A)},$$

and thus it follows that

$$(\omega_A(v^0);\mathcal{L}_i(\vec{\sigma}) \cap \mathcal{L}_o(A)) = Out(G,\vec{\sigma}).$$

Thus, $Out(G,\vec{\sigma}) \subseteq [\![\psi]\!]$ iff $(\omega_A(v^0);\mathcal{L}_i(\vec{\sigma}) \cap \mathcal{L}_o(A)) \subseteq [\![\psi]\!]$, and hence also iff

$$\omega_A(v^0);\mathcal{L}_i(\vec{\sigma}) \subseteq \{\varrho \mid \text{if } \varrho \in \mathcal{L}_o(A) \text{ then } \varrho \in [\![\psi]\!]\}.$$

Let φ_A be a formula (either of LTL or CTL) such that $[\![\varphi_A]\!] = \mathcal{L}_o(A)$. It is known that such formulae exist and are polynomial in the size of A (for instance, see [13] for the LTL case and [6] for the CTL case).

Thus, we have that

$$\vec{\sigma} \models \psi \quad \text{iff} \quad (\omega_A(v^0);\mathcal{L}_i(\vec{\sigma})) \subseteq [\![\varphi_A \to \psi]\!].$$

Since

$$\mathcal{L}_i(\vec{\sigma}) = \bigcap_{0\le j\le n} \mathcal{L}_i(\sigma_j)$$

we can translate every strategy σ_j into a "concurrent process" P_j as defined in [29, pp. 46] so that the output language of that "concurrent program" $P^+ = \prod_{1\le j\le n} P_j^+$ is, by definition, exactly $\mathcal{L}_i(\vec{\sigma})$. Now, let s_j^0 be the initial state of every P_j^+. We then simply add an additional state q_j^0 for every P_j^+, such that $q_j^0 \xrightarrow{\omega_A(v^0)} s_j^0$. Then, the output language of the new concurrent system, which we denote by P, is $\omega_A(v^0);\mathcal{L}_i(\vec{\sigma})$.

Therefore, we have that

$$\vec{\sigma} \models \psi \Leftrightarrow \omega_A(v^0); \mathcal{L}_i(\vec{\sigma}) \subseteq [\![\varphi_A \to \psi]\!] \Leftrightarrow P \models \varphi_A \to \psi.$$

It then immediately follows from [29] that Strategy-Checking is in PSPACE for LTL and CTL. Since CTL is a syntactic fragment of the modal μ-calculus, the formula φ_A is also a μ-calculus formula and thus, again from [29], it follows that Strategy-Checking is in EXPTIME for the μ-calculus. For the linear-time μ-calculus, rather than defining a concurrent program P, we use the usual translation of finite-state transition systems to 1-safe nets (*i.e.*, to synchronous products of transition systems)—*e.g.*, as in [17]—and use the fact that linear-time μ-calculus model checking for 1-safe nets is in PSPACE [24] (recall that since LTL is a syntactic fragment of the linear-time μ-calculus, then φ_A is also a linear-time μ-calculus formula).

Then, for each temporal logic we have considered here, we have reduced Strategy-Checking to a model checking problem for either concurrent programs (as defined in [29]) or a synchronous product of transitions systems (a 1-safe net as defined in [17]), for which the relevant complexities are known.

PSPACE-hardness for goals in LTL, CTL, or the linear-time μ-calculus follows from a reduction from the Finite automaton intersection problem [27], where finite automata are translated into strategies in a game in which $\psi = \mathbf{F}\varphi$ represents the situation where all automata accept the same word. $\square$

Lemma 19. *The* NE-Checking *problem for LTL and linear-time μ-calculus goals is PSPACE-complete. For CTL and μ-calculus goals the problem is EXPTIME-complete.*

Proof. We use the next algorithm:

```
1.  for $i := 1$ to $n$ do
2.      if $\vec{\sigma} \not\models \gamma_i$ then
3.          if $\gamma_i \wedge \varphi_A \wedge \bigwedge_{j \in N \setminus \{i\}} \varphi_j$ is satisfiable then
4.              return "no"
5.          end-if
6.      end-if
7.  end-for
8.  return "yes"
```

where the temporal logic formula φ_j is as in the proof of Lemma 18—an LTL formula in case γ_i is an LTL or a linear-time μ-calculus formula and a CTL formula in case γ_i is a CTL or a μ-calculus formula—namely, φ_j is a temporal logic formula that characterises the behaviour of σ_i. And, as in Lemma 18, the formula φ_A characterises the behaviour of A. Recall that φ_A has as models all, and only, the runs possible in A. The constructions of such formulae can be done, *e.g.*, as described in [13] and [6] for linear-time and for branching-time goals, respectively.

Because of Lemma 18, we know that line (2) of the algorithm can be solved using a PSPACE (EXPTIME) oracle for Strategy-Checking if the goals are LTL, CTL, or linear-time μ-calculus (modal μ-calculus); additionally, line (3) is solved using a PSPACE (EXPTIME) oracle for satisfiability for linear-time (branching-time) goals.

The correctness of the algorithm follows from the definition of Nash equilibrium with respect to binary preferences, that is, because the statement

$$(\vec{\sigma}_{-i}, \sigma'_i) \models \gamma_i \Longrightarrow \vec{\sigma} \models \gamma_i$$

must hold. Therefore,

$$\vec{\sigma} \not\models \gamma_i \Longrightarrow (\vec{\sigma}_{-i}, \sigma'_i) \not\models \gamma_i$$

for any strategy σ'_i, which is the statement that is checked by the algorithm above. Similar arguments and algorithm can be found in [20]. On the other hand, for hardness, one can use reductions from the LTL and CTL satisfiability problems, similar to the way is done in [20]. $\square$

Proposition 23. *Let G be a game and φ be a $[\mathbf{NE}]$-free EL* formula. Then, for all runs ϱ and $t \in \mathbb{N}$, we have that*

$$(G, \varrho, t) \models \mathbf{A}\varphi \text{ iff } (G, \varrho', 0) \models \varphi,$$

for all ϱ' starting at $\varrho[t]$.

Proof. One direction is trivial ($\Leftarrow$). The other direction ($\Rightarrow$) is proved by structural induction on φ. First, for ($\Rightarrow$), suppose that $(G, \varrho, t) \models_{\mathcal{P}} \mathbf{A}\varphi$. Then, we know that $(G, \varrho^*, t) \models_{\mathcal{P}} \varphi$ for all ϱ^* such that $\varrho[0 \dots t) \in \varrho^*$. Here a structural induction on φ proves the statement since only path and state formulae have to be considered.

For the other direction, ($\Leftarrow$), suppose $(G, \varrho', 0) \models \varphi$, for all ϱ' starting at $\varrho[t]$. Then, by the definition of the semantics, we know that $(G, \varrho, t) \models \mathbf{A}\varphi$ since, in particular, it holds for all runs of the form $\varrho[0 \dots t); \varrho'$, as desired. $\square$

Lemma 29. *The* EL* Model-Checking *problem where players' goals are LTL formulae is 2EXPTIME-hard.*

Proof. (Sketch) The proof uses a reduction from the LTL games in [5], which are 2EXPTIME-complete, to the EL* model checking problem with the EL formula $\varphi = \langle \mathbf{NE} \rangle \top$. Such games are two-player zero-sum perfect-information games played on a graph (which can be represented with our model of arenas) where one player, called player 0, wants to satisfy an LTL formula ψ while the other player, called player 1, wishes to refute such a formula ψ. The strategies in such a game can also be represented with our model of strategies.The main purpose of the reduction is to show that player 0 has a winning strategy in the given LTL game if and only if the constructed EL* Model-Checking problem for the formula $\varphi = \langle \mathbf{NE} \rangle \top$ has a positive solution, that is, if and only if there is a Nash equilibrium in the reactive game constructed from the given LTL game.

In order to do so we construct a 4-player game (with $N = \{0, 1, 2, 3\}$) where player 0/1 wants to satisfy/refute ψ and either to play consistently with the original LTL game or to make player 1/0 play inconsistently with the original LTL game; in addition, player 2/3 wants either to see ψ satisfied or to win a simple "matching pennies" game played against player 3/2. What is important in this proof is the fact that the goals of all players in the constructed reactive game can be represented as LTL formulae, and that only deterministic strategies need to be considered in the reactive game. $\square$

Lemma 30. *The* EL* Model-Checking *problem where players' goals are CTL formulae is 2EXPTIME-hard.*

Proof. (Sketch) The game is constructed, essentially, as in Lemma 29. However, player 1 is now allowed to use a non-deterministic strategy. Because of this, the computation generated by the interaction between player 0 and player 1 is now an infinite tree rather than an infinite word, as in the LTL game case. All other constructions remain the same, that is, as in the LTL case, except for the goals, which are now defined as CTL formulae. $\square$