Proceedings of the Fourteenth International Conference on Principles of Knowledge Representation and Reasoning

Reasoning about Equilibria in Game-Like Concurrent Systems

Julian Gutierrez and **Paul Harrenstein** and **Michael Wooldridge**
Department of Computer Science
University of Oxford

Abstract

Our aim is to develop techniques for reasoning about game-like concurrent systems, where the components of the system act rationally and strategically in pursuit of logically-specified goals. We first present a computational model for such systems, and investigate its properties. We then define and investigate a branching-time logic for reasoning about the equilibrium properties of such systems. The key operator in this logic is a path quantifier $[\mathbf{NE}]\varphi$, which asserts that φ holds on all Nash equilibrium computations of the system.

1 Introduction

Our goal 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 (Osborne and Rubinstein 1994). 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 (Emerson 1990) and model checking (Clarke, Grumberg, and Peled 2000) 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 strategic analysis of systems.

The AI/computer science literature is of course replete with logics intended for reasoning about game-like systems: Parikh's Game Logic was an early example (Parikh 1985), and more recently ATL (Alur, Henzinger, and Kupferman 2002) and Strategy Logic (Chatterjee, Henzinger, and Piterman 2010) 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 a temporal logic like ATL or Strategy Logic (or indeed LTL, CTL, ...) to define the goals of agents, and hence their preferences; but such languages don't provide any direct mechanism 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 (Osborne and Rubinstein 1994). A Nash equilibrium is an outcome that obtains because no player has a rational incentive to deviate from it. 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 (Gutierrez, Harrenstein, and Wooldridge 2013), 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 (Gutierrez, Harrenstein, and Wooldridge 2013) 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 (Gutierrez, Harrenstein, and Wooldridge 2013) is useful for the purposes of exposition, it is not a realistic model of concurrent programs. Moreover, (Gutierrez, Harrenstein, and Wooldridge 2013) 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 one can check useful properties of strategy profiles. We consider four logics for players' goals: LTL (Pnueli 1977), CTL (Clarke and Emerson 1981), the linear-time μ-calculus (Vardi 1988), and the branching-time μ-calculus (Kozen 1983). Based on our logical characterisation of equilibria in infinite games, 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 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 spectra. A summary of key results is given at the end of the paper. Note that most proofs are omitted due to lack of space.

2 Models

Before giving formal definitions, let us start by describing a situation that can naturally be modelled as a game-like concurrent and multi-agent system.

Example 1. Consider a situation in which two agents can request a resource from a dispatch centre infinitely often; assume both agents will always eventually need the resource. The centre's behaviour 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 gets the resource for ever after (thus, punishing 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 both agents get their goals achieved. Indeed, a game-theoretic analysis reveals that *in all equilibria of this system both agents get the resource infinitely often*.

In order to model this kind of situation, we will define a model for concurrent strategic interactions. Using this model, in Section 5 we present a formal model of the above example, and we then formally analyse its equilibrium properties (Example 14).

The basic formal model for capturing strategic interactions is a graph-based structure that is used for various distinct purposes. It will be used for both the *model* of game-like concurrent systems we are interested in modelling and the *strategies* of players in these systems. Depending on how we use and instantiate our model to each of the above applications they may acquire a particular name and have a refined structure. As will be clear from their 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 (Nielsen and Winskel 1995). Formally, let

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

be a Λ-labelled Ω-model M, where V is the set of vertices[1] of M, $v^0 \in V$ is the initial vertex, $E \in V \times V$ is the set of edges[2], 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*:

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

and the sets of sequences[3] of adjacent vertices and transitions starting at v^0, which we denote by V^*_M and T^*_M.

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

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

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

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 $\rho[0], \rho[1], \ldots$ for the first, second, ... element in the sequence; if ρ is finite, we write $last(\rho)$ to refer to its last element. We also write $|\rho|$ for the size of a sequence. The empty sequence is denoted by $\rho[\,] = \epsilon$ 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 $\rho[0 \ldots k]$ for the sequence $\rho[0], \rho[1], \ldots, \rho[k]$ (an initial segment of ρ), $\rho[k \ldots k']$ for the sequence $\rho[k], \ldots, \rho[k']$, and $\rho[k \ldots \infty]$ for the infinite sequence $\rho[k], \rho[k+1], \ldots$. We also write, *e.g.*, $\rho[k \ldots k')$ if the element $\rho[k']$ of the sequence is not included. Given a finite sequence ϱ, we write $\varrho \in \rho$ if $\varrho[k] = \rho[k]$ for all $0 \leq k \leq |\varrho|$. We also write $\varrho; \rho$ 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 on them, in particular, w.r.t. concatenation ";" with the empty sequence ϵ we have $\epsilon; \rho = \rho$, for any ρ, and $\rho; \epsilon = \rho$, for any finite ρ.

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

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

[1] We may write 'nodes' or 'states' when talking about vertices.

[2] We also call them 'events' or 'actions'.

[3] We also say 'words' or 'strings' when talking about sequences.

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

$$\bigwedge_{1\leq j\leq n}(v^0_j,a,v'_j),(v'_j,a',v''_j),\ldots \in T^*_{M_j},$$

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

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

where $\rho\in V^*_{\vec{M}}$ and $\omega_j(\rho[k])$, with $0\leq k<|\rho|$, is the application of the ω_j of M_j to the jth component of each $\rho[k]$.

Let $\mathcal{L}$ function equally for an input or output language of a model M or compound system $\vec{M}$; moreover, let $\mathcal{L}[\varrho]$ be the (sub)language of (sub)words $\{\rho[|\varrho|\ldots\infty]\mid\varrho\in\rho\in\mathcal{L}\}$.

There is an induced tree language $\mathit{TreeL}(\mathcal{L})$ for every (word) language $\mathcal{L}$, defined as:

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

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

Remark 2. 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, ..., and the infinite tree with infinitely many a-labelled branches. However, if non-determinism is not allowed (or at least 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:

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

3 Games and Strategies

Games. Using the model given in Section 2, we will define reactive games, a class of multi-player nonzero-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 is defined as follows. 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 of the game), $C=\{p,q,r,\ldots\}$ 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[4]) 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. And A—the *"arena" or "board"* where the game is played—is a Λ-total Ω-model such that $\Lambda=\mathit{Ch}$ and $\Omega=X$.

Note that Λ-totality ensures in a simple manner that a reactive game is played for an infinite number of rounds without imposing further consistency or validity conditions on strategies. Moreover, it does not limit our modelling power.

Remark 3. Reactive games can be considered as a metamodel of *infinite* games of *control* since their definition does not specify the logic 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, different kinds of games and results will arise from different choices with respect to these properties. All we 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 that (3) they have goals given in a *logical form*.

Strategies. Since in a reactive game a play is infinite, it is natural to think of a strategy for a player i as a function $f_i:E^*\to\mathit{Ch}_i$ or as $f'_i:V^*\to\mathit{Ch}_i$, that is, as a function from what has been played so far (or at least what a player *knows so far*) to a choice c_i for player i. To formalise this, 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.*, 'repeated games' in game theory (Osborne and Rubinstein 1994, pp. 140-143).

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

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

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*, $q^0_i=v^0$ is the *initial* state, $\delta_i:Q_i\times\Lambda\to 2^{Q_i}$, with $\Lambda=2^{X_A}$, is the *transition function* given by T_{M_i}, and $\tau_i=\omega:Q_i\to\mathit{Ch}_i$ 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.

[4] We will consider several logical temporal languages for γ_i, *e.g.*, LTL (Pnueli 1977), CTL (Clarke and Emerson 1981), or fixpoint linear-time and branching-time modal logics (Vardi 1988; Kozen 1983). At this point all definitions can be made leaving this choice open, which will make our framework more general.

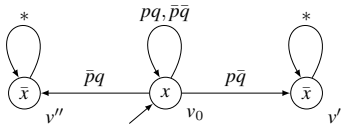

Figure 1: An arena, where $* = \{pq, \bar{p}q, p\bar{q}, \bar{p}\bar{q}\}$.

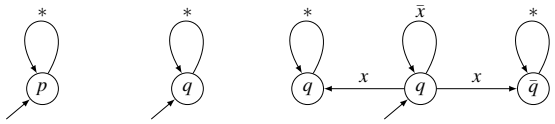

Figure 2: Strategy σ_1 for player 1 (left) and the strategies σ_2 (middle) and σ_2' (right) for player 2. Here $* = \{x, \bar{x}\}$.

Henceforth, given a game G with n players in N and a set 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 4. Consider a game with $N = \{1, 2\}$, $C_1 = \{p\}$, and $C_2 = \{q\}$ and arena as in Figure 1. There, we have $\omega_A(v^0) = x$, $\omega_A(v') = \omega_A(v'') = \bar{x}$. Moreover, the symbol $\bar{p}$ means $p := \bot$, p means $p := \top$ (and likewise for q and x). A possible strategy for player 1 would be to always play p and is depicted in Figure 2 (left). The exact outcome(s) of the game—to be defined next—can be determined only once the strategy for player 2 is given.

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

Lemma 5. *Let $\mathcal{T}$ be an ω-regular tree—i.e., the unfolding of a finite, total graph. There is σ such that $\mathcal{T} \in \mathit{TreeL}(\mathcal{L}_o(\sigma))$.*

It is important to point out that the strategy σ may be non-deterministic. However, with respect to word languages, only deterministic strategies are needed. Specifically, because ω-regular words are ω-regular trees that do not branch, the following is an easy corollary of Lemma 5.

Corollary 6. *Let w be an ω-regular word. There is σ such that $w = \mathcal{L}_o(\sigma)$.*

Lemma 5 and Corollary 6 will be used to ensure the existence of strategies in a reactive game with ω-regular goals.

Outcomes and composition of strategies. Given a 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 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 $\mathcal{L}_o^{\vec{\sigma}}(A)$ and defined as

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

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 define 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 $v = v^0$. Define the set

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

Thus, given a strategy profile $\vec{\sigma}$ in a reactive game G, the tree/branching outcomes of G are the trees in the set

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

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 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 $\mathit{Out}(G, \vec{\sigma})$ is not necessarily a singleton, as illustrated next.

Example 7. Consider again the game in Example 4 and the two strategies for player 2 depicted in Figure 2. The strategy profile $\vec{\sigma} = (\sigma_1, \sigma_2)$ induces the unique (word) outcome $x^\omega = x, x, \ldots$ in Out; the strategy profile $\vec{\sigma}' = (\sigma_1, \sigma_2')$, on the other hand, induces infinitely many outcomes, namely those 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 because σ_2' is non-deterministic; thus, multiple outcomes are possible even when 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.

Remark 8. Observe that the reactive games model strictly generalises the iBG model (Gutierrez, Harrenstein, and Wooldridge 2013), which can be represented as a reactive game where the arena is an *implicitly* defined clique whose nodes are the valuations of the variables the players control, goals are LTL formulae, and strategies are deterministic.

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}. Even though $\leq_i$ can 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.*, $X \leq_i X'$ 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—sometimes called solution concept—in game theory, namely, *Nash equilibrium*. However, many solution concepts can be found in the literature, *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 σ_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 the game G.

Remark 9. Note that since strategies or arenas can be nondeterministic—hence multiple outcomes can be induced, our definition of Nash equilibrium is phrased in terms of *sets* of outcomes, rather than in terms of single outcomes only. Even though the definition of equilibrium is given w.r.t. 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 as in this paper.

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 satisfy such goals, that is, we need to define a semantics of players' goals w.r.t. 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 notations 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 $\vec{\sigma}$ is a Nash equilibrium. We say that $\vec{\sigma}$ is a Nash equilibrium if, for every player i and for every strategy σ_i', we have that

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

Remark 10. Since our model generalises Kripke structures and Labelled Transition Systems (among other structures), it can be used to give the standard semantics of all usual 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 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 can be non-deterministic, and that the outcomes are tree outcomes.

The details of the semantics of these logics need not be given to obtain the results in this paper. All 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, whereas LTL and the linear-time μ-calculus are PSPACE; for model checking w.r.t. a product of transition systems, the logics LTL, CTL, and linear-time μ-calculus are PSPACE, while the μ-calculus is EXPTIME. The next question becomes relevant:

Given: Game G, strategy profile $\vec{\sigma}$, goal γ.
STRATEGY-CHECKING: Is it the case that $\vec{\sigma} \models \gamma$?

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

Lemma 11. *The* STRATEGY-CHECKING *problem for LTL, CTL, and linear-time μ-calculus goals is PSPACE-complete. For modal μ-calculus goals the problem is in EXPTIME.*

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 $\vec{\sigma} \in NE(G)$?

Formally, we have

Lemma 12. *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.*

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, iff $[\![\vec{\sigma}]\!] = [\![\vec{\sigma}']\!]$.

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, iff for every goal in $(\gamma_i)_{i\in N}$ of G we have that

$$\vec{\sigma} \models \gamma_i \Longleftrightarrow \vec{\sigma}' \models \gamma_i.$$

Formally, the decision 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?

An immediate consequence of Lemma 11 is:

Corollary 13. *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.*

5 Equilibrium logics

We now introduce two logics for expressing equilibrium properties of (reactive) games: these two logics are closely related to the branching time logics CTL and CTL* (see, *e.g.*, (Emerson and Halpern 1986)). 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 w.r.t. a set X of propositions, by the following grammars:

Path Formulae:	$\varphi ::= \psi \mid \theta \mid \neg\varphi \mid \varphi \vee \varphi \mid \mathbf{X}\varphi \mid \varphi\,\mathbf{U}\,\varphi$
State Formulae:	$\psi ::= x \mid \mathbf{A}\varphi$
Nash Formulae:	$\theta ::= [\mathbf{NE}]\varphi$

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. 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:[5] $\mathbf{E}\varphi \equiv \neg\mathbf{A}\neg\varphi$ and $\langle\mathbf{NE}\rangle\varphi \equiv \neg[\mathbf{NE}]\neg\varphi$.

Semantics. The semantics of EL formulae is given here w.r.t. 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 π 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

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

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

$(G, \varrho, t) \models_{\mathcal{S}} x$	iff	$x \in \omega_A(\varrho[t])$
$(G, \varrho, t) \models_{\mathcal{S}} \mathbf{A}\varphi$	iff	$(G, \varrho', t) \models_{\mathcal{P}} \varphi$ for all ϱ' such that $\varrho[0\ldots t) \in \varrho'$.

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

$(G, \varrho, t) \models_{\mathcal{E}} [\mathbf{NE}]\varphi$	iff	$(G, \varrho', t) \models_{\mathcal{P}} \varphi$ for all $\varrho' \in [\![\vec{\sigma}]\!]$ with both $\vec{\sigma} \in NE(G)$ and $\varrho[0\ldots t) \in \varrho'$

We say that G is a model of φ (in symbols $G \models \varphi$) if and only 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^*.

Example 14. 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 3. 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

[5] All usual abbreviations for the Boolean operators not explicitly given in the grammars above are also assumed to be defined.

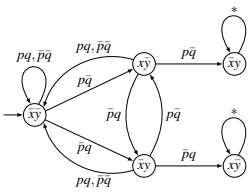

Figure 3: Formal model of the system in Example 1

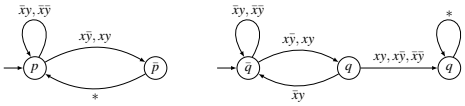

Figure 4: Strategies σ_1 (left) and σ_2 (right).

of the players are the LTL goals $\gamma_1 = \mathbf{GF}x$ and $\gamma_2 = \mathbf{GF}y$. The structures in Figure 4 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 strategy profile $\vec{\sigma} = (\sigma_1, \sigma_2)$ yields $\mathcal{L}_i^{\vec{\sigma}}(A) = \{(p\bar{q};\bar{p}q)^\omega\}$ and $\mathcal{L}_o^{\vec{\sigma}}(A) = \{\bar{x}\bar{y};(x\bar{y};\bar{x}y)^\omega\}$. The run $\rho = \bar{x}\bar{y}, x\bar{y}, \bar{x}y, x\bar{y}, \bar{x}y, \ldots$ 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 $\mathbf{A}(\gamma_1 \wedge \gamma_2)$, 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 specifically 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.

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 $\varrho[0\ldots t) \in \varrho'$." In the standard semantics of CTL* it should 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 15. *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 15, 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.*, (Kupferman and Vardi 2006).

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. Firstly it is known that neither ATL* (Alur, Henzinger, and Kupferman 2002) nor Strategy logic (Chatterjee, Henzinger, and Piterman 2010) can express the existence of a Nash equilibrium in a multi-player game with deterministic strategies and LTL players' goals. This property is simply stated as $\langle\mathbf{NE}\rangle\top$ in EL, where we restrict ourselves to reactive games with deterministic strategies and arenas and only LTL players' goals. Thus, we can conclude that both *Strategy logic and ATL* are not as expressive as Equilibrium logics*.

On the other hand, the existence of Nash equilibria in multi-player games and w.r.t. deterministic strategies and LTL goals can be expressed in the Strategy logic developed in (Mogavero, Murano, and Vardi 2010a). Using their logical specification of the existence of a Nash equilibrium, we can encode the $\langle\mathbf{NE}\rangle$ operator (and hence any $\langle\mathbf{NE}\rangle\psi$, with LTL ψ), again when restricted to LTL goals and deterministic games and strategies. This gives a 2EXPTIME upper bound for the model checking problem of this fragment of our logic. Note that the fragment is not only syntactic since the underlying reactive games model is also being restricted.

Letting EL*[LTL] be the fragment of EL* given above, we can show (w.r.t. deterministic arenas and strategies):

Proposition 16. *The model checking problem for EL*[LTL] formulae is in 2EXPTIME.*

Together with the hardness result that we will give in the next section, we obtain that model checking EL*[LTL] formulae is an 2EXPTIME-complete problem.

6 Complexity

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: Does $G \models \varphi$ hold?

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).

Hardness in the linear-time framework. In order to show 2EXPTIME-hardness of checking equilibrium properties given that the players of the game have linear-time goals, we provide a reduction from the LTL games in (Alur, La Torre, and Madhusudan 2003). Formally, we have

Lemma 17. *The* EL* MODEL-CHECKING *problem where players' goals are LTL formulae is 2EXPTIME-hard.*

Proof. (Sketch) We reduce the LTL games studied in (Alur, La Torre, and Madhusudan 2003), which are 2EXPTIME-complete, to EL* model checking $\varphi = \langle\mathbf{NE}\rangle\top$.

LTL games. An LTL game (Alur, La Torre, and Madhusudan 2003) is a two-player zero-sum game given by (G, ψ, u), where ψ is an LTL formula over a set of properties Σ, and

$$G = (V_G, V_0, V_1, \gamma : V \to 2^V, \mu : V \to \Sigma)$$

is a graph with vertices in V_G which are partitioned in player 0 vertices V_0 and player 1 vertices V_1. The transitions of the graph are given by γ; if $v' \in \gamma(v)$, for some $v, v' \in V_G$, then there is a transition from v to v', which we may assume is labelled by v'. Each vertex v has a set of properties p associated with it, which are given by μ; thus $p \in \Sigma$ holds in vertex v if $p = \mu(v)$. The graph G is assumed to be total, that is, for every $v \in V_G$, we have $\gamma(v) \neq \emptyset$.

The game is played for infinitely many rounds by each player choosing a successor vertex whenever it is their turn: player 0 plays in vertices in V_0 and player 1 plays in vertices in V_1. The game starts in the initial vertex $u \in V_G$. Playing the game defines an infinite word/sequence of adjacent vertices $w = v_0, v_1, v_2, \ldots$, such that $v_0 = u$ and for each v_k, with $k \in \mathbb{N}$, we have $v_{k+1} \in \gamma(v_k)$. An LTL game is won by player 0 if $w \models \psi$; otherwise player 1 wins. LTL games are determined, that is, there is always a winning strategy either for player 0 or for player 1. Checking whether player 0 has a winning strategy in an LTL game is a 2EXPTIME-complete problem (Alur, La Torre, and Madhusudan 2003).

We reduce the LTL game (G, ψ, u) to model checking the EL* formula $\varphi = \langle\mathbf{NE}\rangle\top$ in two stages. We first construct a reactive game G_1 that simulates the behaviour exhibited in the LTL game (G, ψ, u), and then transform G_1 into another reactive game G_2 with the property that player 0 has a winning strategy in (G, ψ, u) if and only if there is a Nash equilibrium in G_2, that is, if and only if $G_2 \models \varphi$.

Reactive games. The first construction (the one for G_1) comprises two players, 0 and 1, who control variables t_0 and t_1, respectively, whose domain is the set of vertices in the LTL with two additional actions, δ_0 and δ_1, to delay/skip when it is not their turn to play in the corresponding LTL game. Notice that in a reactive game they always must play, at each round, concurrently and synchronously. Their goals are $\gamma_0 = (\psi \wedge \mathbf{G}\neg l_0) \vee \mathbf{F} l_1$ and $\gamma_1 = (\neg\psi \wedge \mathbf{G}\neg l_1) \vee \mathbf{F} l_0$, respectively, where l_0 and l_1 are two fresh atomic propositions which hold only on two new states where, intuitively, player 0 loses the LTL game (state l_0) and player 1 loses the LTL game (state l_1). This translation from (G, ψ, u) to G_1 is polynomial. There are $|V_G| + 2$ vertices and $|\mu| + \mathcal{O}(|V_G|^3)$ transitions in the arena where G_1 is played. This first translation, from (G, ψ, u) to G_1, transforms a sequential zero-sum LTL game into a non-zero sum reactive game (where players make choices concurrently and synchronously) that is able to simulate the original LTL game. Players in G_1 are not forced to play consistently with plays in (G, ψ, u), but these players can be punished if they do not do so.

Then, we transform G_1 into G_2 in order to ensure that all (and only the) Nash equilibria of the reactive game G_2 correspond to plays where player 0 wins. The game G_2 comprises four players. The two new players, 3 and 4, control two Boolean variables t_2 and t_3, and have goals $\gamma_2 = \psi \vee (\mathbf{X}e)$ and $\gamma_3 = \psi \vee (\mathbf{X}\neg e)$, where e is a new atomic proposition. The arena of G_2 is linear in the size of the arena of G_1. It contains a copy of the vertices in game G_1 where the proposition $\neg e$ holds. In all other vertices—*i.e.,* those in G_1—the proposition e holds. The choice of where the game proceeds after the first round, namely, to the vertices already in G_1 or their copies, is ultimately controlled by the choices of players 2 and 3. In particular, in the former case, we have $t_2 = t_3$ (and move to a vertex where e holds) whereas, in the latter, we have $t_2 \neq t_3$ (and move to a vertex where $\neg e$ holds). This power allows these two players to deviate whenever ψ is not satisfied. As a consequence, all Nash equilibria of G_2 must satisfy formula ψ and hence the goals of players 2 and 3. Using this fact and Corollary 6, we build a winning strategy in the LTL game iff there is a Nash equilibrium in G_2. □

Hardness in the branching-time framework. Now, in order for us 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 (Kupferman et al. 2000). The control-synthesis problem for reactive environments is similar to the LTL game. However, in such a game player 1 may challenge player 0 with sets of successor vertices rather than with singletons, as in the LTL game. As a consequence, a play of the game can produce a tree instead of a word of vertices. In this game, player 0 wins if such a tree satisfies formula ψ; otherwise, player 1 wins. This game is also 2EXPTIME-complete and can be reduced to a reactive game in a way similar to that for LTL games. In particular, in this case, we can use Lemma 5 instead of Corollary 6 to ensure that strategies in these games always exist. Formally, we have

Lemma 18. *The* EL* MODEL-CHECKING *problem where players' goals are CTL formulae is 2EXPTIME-hard.*

Since LTL and CTL are syntactic fragments of, respectively, the linear-time μ-calculus and the modal μ-calculus, the hardness results can be transferred, to obtain:

Corollary 19. *The* EL* MODEL-CHECKING *problem, with players' goals given by linear-time μ-calculus or by 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, is 2EXPTIME-complete for both LTL and CTL specifications, and seems to be the main source of the high complexity of checking equilibrium properties.

7 Concluding Remarks and Related Work

Complexity. We have shown that checking the equilibrium properties of a concurrent and multi-agent system is a computationally very 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 temporal formulae. A summary of our complexity results is in Table 1.

	ST–CHECK	NE–CHECK	EQ–CHECK	EL*–MC
LTL	PSPACE	PSPACE	PSPACE	2EXPT$-$h
CTL	PSPACE	EXPT	PSPACE	2EXPT$-$h
TL$_\mu$	PSPACE	PSPACE	PSPACE	2EXPT$-$h
L$_\mu$	iEXPT	EXPT	iEXPT	2EXPT$-$h

Table 1: Overview of computational complexity results. In this table TL$_\mu$ stands for the linear-time μ-calculus, L$_\mu$ 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, and iEXPT for in EXPTIME.

Logic.
Our work relates to logics either that are to reason about the behaviour of game-like systems or that are memoryfull. In (Kupferman and Vardi 2006) 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 w.r.t. 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 (Laroussinie, Markey, and Schnoebelen 2002) 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 shared 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 even a harder problem. Another memoryfull logic, this time one designed to reason about game-like scenarios, is Strategy logic (Chatterjee, Henzinger, and Piterman 2010). 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 (Chatterjee, Henzinger, and Piterman 2010) is a logic with a two-player semantic game. This logic was later on extended to the multi-player setting in (Mogavero, Murano, and Vardi 2010a). Recent works, for instance (Mogavero et al. 2012; Mogavero, Murano, and Sauro 2013), 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. As with Equilibrium logics, the exact complexity upper bound of model checking a syntactic fragment that is able to express Nash equilibria is, presently, an open problem. Finally, in (Mogavero, Murano, and Vardi 2010b) 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 (Alur, Henzinger, and Kupferman 2002). 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.

Future work.
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 avenue for further work is the study of solution concepts specifically designed for the analysis of dynamic systems, in particular, we would like to investigate the well-known concept of *subgame-perfect Nash equilibrium*. We also leave for future work the study of richer preference relations as well as quantitative and imperfect information. Certain decision problems slightly discussed in this paper should also be studied in more detail. In this paper, we have investigated the complexity of some questions about reactive games and equilibrium computations in multi-agent and concurrent systems, and would like to understand better the *expressivity of equililibrium logics*. They certainly offer a different paradigm for reasoning about equilibria in infinite games, but whether they can be translated into one of the logics 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 logics such as ATL and probabilistic variants of CTL* in model checkers such as PRISM (Chen et al. 2013), MCMAS (Lomuscio, Qu, and Raimondi 2009), or MOCHA (Alur et al. 1998).

Acknowledgments

We thank the anonymous reviewers for their helpful comments. We also acknowledge the support of the ERC Ad-

vanced Investigator Grant 291528 ("RACE").

References

Alur, R.; Henzinger, T. A.; Mang, F. Y. C.; Qadeer, S.; Rajamani, S. K.; and Tasiran, S. 1998. MOCHA: modularity in model checking. In *CAV*, volume 1427 of *LNCS*, 521–525. Springer.

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

Alur, R.; La Torre, S.; and Madhusudan, P. 2003. Playing games with boxes and diamonds. In *CONCUR*, volume 2761 of *LNCS*, 127–141. Springer.

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

Chen, T.; Forejt, V.; Kwiatkowska, M. Z.; Parker, D.; and Simaitis, A. 2013. PRISM-games: a model checker for stochastic multi-player games. In *TACAS*, volume 7795 of *LNCS*, 185–191. Springer.

Clarke, E. M., and Emerson, E. A. 1981. Design and synthesis of synchronization skeletons using branching time temporal logic. In *Logics of Programs*, volume 131 of *LNCS*, 52–71. Springer-Verlag: Berlin, Germany.

Clarke, E. M.; Grumberg, O.; and Peled, D. A. 2000. *Model Checking*. The MIT Press: Cambridge, MA.

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

Emerson, E. A. 1990. Temporal and modal logic. In *Handbook of Theoretical Computer Science Volume B: Formal Models and Semantics*. Elsevier Science Publishers B.V.: Amsterdam, The Netherlands. 996–1072.

Gutierrez, J.; Harrenstein, P.; and Wooldridge, M. 2013. Iterated boolean games. In *IJCAI*. IJCAI/AAAI Press.

Kozen, D. 1983. Results on the propositional mu-calculus. *Theoretical Computer Science* 27:333–354.

Kupferman, O., and Vardi, M. Y. 2006. Memoryful branching-time logic. In *LICS*, 265–274. IEEE Computer Society.

Kupferman, O.; Madhusudan, P.; Thiagarajan, P. S.; and Vardi, M. Y. 2000. Open systems in reactive environments: Control and synthesis. In *CONCUR*, volume 1877 of *LNCS*, 92–107. Springer.

Laroussinie, F.; Markey, N.; and Schnoebelen, P. 2002. Temporal logic with forgettable past. In *LICS*, 383–392. IEEE Computer Society.

Lomuscio, A.; Qu, H.; and Raimondi, F. 2009. MCMAS: a model checker for the verification of multi-agent systems. In *CAV*, volume 5643 of *LNCS*, 682–688. Springer.

Mogavero, F.; Murano, A.; Perelli, G.; and Vardi, M. Y. 2012. What makes ATL* decidable? A decidable fragment of strategy logic. In *CONCUR*, volume 7454 of *LNCS*, 193–208. Springer.

Mogavero, F.; Murano, A.; and Sauro, L. 2013. On the boundary of behavioral strategies. In *LICS*, 263–272. IEEE Computer Society.

Mogavero, F.; Murano, A.; and Vardi, M. Y. 2010a. Reasoning about strategies. In *FSTTCS*, volume 8 of *LIPIcs*, 133–144. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik.

Mogavero, F.; Murano, A.; and Vardi, M. Y. 2010b. Relentful strategic reasoning in alternating-time temporal logic. In *LPAR*, volume 6355 of *LNCS*, 371–386. Springer.

Nielsen, M., and Winskel, G. 1995. Models for concurrency. In *Handbook of Logic in Computer Science*. Oxford University Press. 1–148.

Osborne, M. J., and Rubinstein, A. 1994. *A Course in Game Theory*. The MIT Press: Cambridge, MA.

Parikh, R. 1985. The logic of games and its applications. In *Topics in the Theory of Computation*. Elsevier.

Pnueli, A. 1977. The temporal logic of programs. In *FOCS*, 46–57. IEEE.

Vardi, M. Y. 1988. A temporal fixpoint calculus. In *POPL*, 250–259. ACM Press.