Nash Equilibrium and Bisimulation Invariance*

Julian Gutierrez[1], Paul Harrenstein[2], Giuseppe Perelli[3], and Michael Wooldridge[4]

1 **Department of Computer Science, University of Oxford, Oxford, United Kingdom**
julian.gutierrez@cs.ox.ac.uk
2 **Department of Computer Science, University of Oxford, Oxford, United Kingdom**
paul.harrenstein@cs.ox.ac.uk
3 **Department of Computer Science, University of Oxford, Oxford, United Kingdom**
giuseppe.perelli@cs.ox.ac.uk
4 **Department of Computer Science, University of Oxford, Oxford, United Kingdom**
mjw@cs.ox.ac.uk

Abstract

Game theory provides a well-established framework for the analysis of concurrent and multi-agent systems. The basic idea is that concurrent processes (agents) can be understood as corresponding to players in a game; plays represent the possible computation runs of the system; and strategies define the behaviour of agents. Typically, strategies are modelled as functions from sequences of system states to player actions. Analysing a system in such a way involves computing the set of (Nash) equilibria in the game. However, we show that, with respect to the above model of strategies—the standard model in the literature—*bisimilarity does not preserve the existence of Nash equilibria*. Thus, two concurrent games which are behaviourally equivalent from a semantic perspective, and which from a logical perspective satisfy the same temporal formulae, nevertheless have fundamentally different properties from a game theoretic perspective. In this paper we explore the issues raised by this discovery, and investigate three models of strategies with respect to which the existence of Nash equilibria is preserved under bisimilarity. We also use some of these models of strategies to provide new semantic foundations for logics for strategic reasoning, and investigate restricted scenarios where bisimilarity can be shown to preserve the existence of Nash equilibria with respect to the conventional model of strategies in the literature.

1998 ACM Subject Classification F.3.1 Specifying and Verifying and Reasoning about Programs, F.4.1 Mathematical logic (Temporal Logic), I.2.11 Distributed AI (Multiagent Systems)

Keywords and phrases Bisimulation, Nash Equilibrium, Multiagent Systems, Strategy Logic

Digital Object Identifier 10.4230/LIPIcs.CONCUR.2017.17

1 Introduction

The concept of *bisimilarity* plays a central role in both the theory of concurrency [18, 16] and logic [26, 16]. In the context of concurrency, bisimilar systems are regarded as *behaviourally equivalent*—appearing to have the same behaviour when interacting with an

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

arbitrary environment. From a logical/verification perspective, bisimilar systems are known to satisfy the *same temporal logic properties* with respect to languages such as LTL, CTL, or the μ-calculus. These features, in turn, make it possible to verify temporal logic properties of concurrent systems using bisimulation-based approaches. For example, temporal logic model checking techniques may be optimised by applying them to the smallest bisimulation-equivalent model of the system being analysed; or, indeed, to every other model within the system's bisimulation equivalence class. This is possible because the properties that one is interested in checking are *bisimulation invariant*.

Model checking is not the only verification technique that can benefit from bisimulation invariance. Consider abstraction and refinement techniques [9, 10] (where a set of states is either collapsed or broken down in order to build a somewhat simpler set of states); coinduction methods [25] (which can be used to check the correctness of an implementation with respect to a given specification); or reduced BDD representations of a system [7] (where isomorphic, and therefore bisimilar, subgraphs are merged, thereby eliminating part of the initial state space of the system). Bisimulation invariance is therefore a very important concept in the formal analysis and verification of concurrent and multi-agent systems.

Game theory [22] provides another important framework for the analysis of concurrent and multi-agent systems. Within this framework, a concurrent/multi-agent system is viewed as a game, where processes/agents correspond to players, system executions/computation runs to plays, and individual process behaviours are modelled as player strategies, which are used to resolve the possible nondeterministic choices available to each player. In logic and computer science, games have also been extensively used for synthesis and verification. In this case, one is usually focused on two-player zero-sum games where the desired solution concept is given by the existence of winning strategies. Instead, in this paper, we are more interested in the more general framework given by multi-player non-zero-sum games, where the standard solution concept is given by of strategies forming a Nash equilibrium.

A widely-used model for strategies in (concurrent) multi-player games is to view a strategy for a process i as a function f_i which maps finite histories $s_0, s_1, \ldots, s_k$ of system states to actions $f_i(s_0, s_1, \ldots, s_k)$ available to the process/agent/player i at state s_k. In what follows, we use the terms process, agent, and player interchangeably. We refer to this as the conventional model of strategies, as it is the best-known and most widely used model in logic, AI, and computer science (and indeed in extensive form games [22]). For instance, specification languages such as alternating-time temporal logic (ATL [4]), and formal models such as concurrent game structures [4] use this model of strategies. If we model a concurrent/multi-agent system as a game in this way, then the analysis and verification of the system reduces to computing the set of (Nash) equilibria in the associated multi-player game; in some cases, the analysis reduces to the computation of a winning strategy.

Because bisimilar systems are regarded as behaviourally equivalent, and bisimilar systems satisfy the same set of temporal logic properties, it is natural to ask whether the Nash equilibria of bisimilar structures are identical as well; that is, we ask the following question:

Is Nash equilibrium invariant under bisimilarity?

We show that, for the conventional model of strategies, the answer to this question is, in general, 'no'. More specifically, the answer critically depends on precisely how players' strategies are modelled. With the conventional model of strategies, the answer is positive only for some two-player games, but negative in general for games with more than two players. This means, for instance, that, in the general case, bisimulation-based techniques cannot be used when one is also reasoning about the Nash equilibria of concurrent systems that are formally modelled as (concurrent) multi-player games.

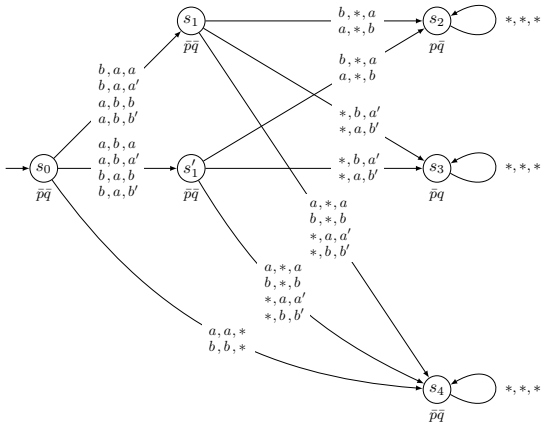

Figure 1 A 3-player game with a Nash equilibrium.

Motivated by this observation—which brings together in a striking way a fundamental concept in game theory and a fundamental concept in logic/concurrency—the purpose of the present paper is to investigate this issue in detail. We present three alternative models of strategies in which Nash equilibria and their existence are preserved under bisimilarity. We also study the above question for special classes of systems, and explore applications to logic. Specifically, we investigate the implications of replacing the conventional model of strategies with some of the models we propose in this paper in logics for strategic reasoning [19, 8], in particular, the semantic implications with respect to Strategy Logic (SL [19]). We also show that, within the conventional model of strategies, Nash equilibrium is preserved by bisimilarity in certain two-player games as well as in the class of concurrent game structures that are induced by iterated Boolean games [14], a logic-based framework for reasoning about the strategic behaviour of different kinds of multi-agent systems [28, 29].

A Motivating Example

So far we have mentioned some cases where one needs or desires a property to be invariant under bisimilarity. However, one may still wonder why it is so important that the particular property of having a Nash equilibrium is preserved under bisimilarity. One reason has its roots in automated formal verification. To illustrate this, imagine that the system of Figure 1 is given as input to a verification tool. It is likely that such a tool will try to perform as many optimisations as possible to the system before any analysis is performed. The simplest of such optimisations—for instance as done by virtually every model checking tool—is to reduce the input system by merging *isomorphic* subtrees (*e.g.*, when generating the ROBDD representation of a system). If such an optimisation is made, the tool will construct the (bisimilar) system in Figure 2. (Observe that the subgraphs rooted at s_1 and s_1' are isomorphic.) However, with respect to the existence of Nash equilibria, such a transformation is unsound in the general case.

For instance, suppose that the system in Figure 1 represents a 3-player game, where each transition is labelled by the choices x, y, z made by player 1, 2, and 3, respectively, an asterisk $*$ being a wildcard for any action for the player in the respective position. Thus, whereas players 1 and 2 can choose to play either a or b at each state, player 3 can choose between a, b, a', or b'. The states are labelled by valuations xy over $\{p, q\}$, where $\bar{x}$ indicates that x is set to false. Assume that player 1 would like p to be true sometime, that player 2

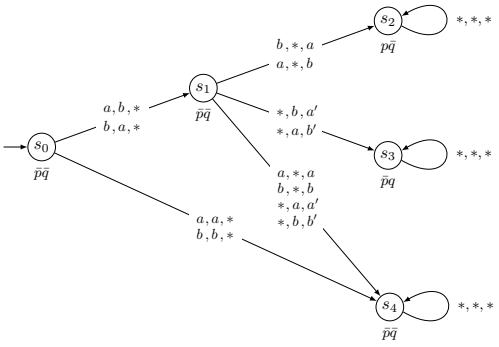

Figure 2 A 3-player game without Nash equilibria.

would like q to be true sometime, and that player 3 desires to prevent both player 1 and player 2 from achieving their goals. Accordingly, their preferences/goals can be formally represented by the LTL formulae $\gamma_1 = \mathsf{F}\, p$, $\gamma_2 = \mathsf{F}\, q$, and $\gamma_3 = \mathsf{G}\, \neg(p \vee q)$, respectively. Informally, $\mathsf{F}\, \varphi$ means "eventually φ holds" and $\mathsf{G}\, \varphi$ means "always φ holds." Moreover, given these players' goals and the conventional model of strategies, we will see later in Section 4.2 that the system in Figure 1 has a Nash equilibrium, whereas no Nash equilibria exists in the (bisimilar) system in Figure 2. This example illustrates a major issue when analysing Nash equilibria in the most widely used models of strategies and games studied in the literature, namely, that even the simplest optimisations commonly used in automated formal verification are not sound with respect to game theoretic analyses.

2 Preliminaries

We begin by introducing the main technical concepts and models used in this paper.

Concurrent Game Structures

We use the model of concurrent game structures, which are well-established in the literature (see, for instance, [4]). A *concurrent game structure (CGS)* is a tuple $M = (\mathrm{Ag}, \mathrm{AP}, \mathrm{Ac}, \mathrm{St}, s^0_M, \lambda, \delta)$, where $\mathrm{Ag} = \{1, \ldots, n\}$ is a set of *players* or agents, AP a set of *propositional variables*, Ac is a set of *actions*, St is a set of *states* containing a unique *initial state* s^0_M. With each player $i \in \mathrm{Ag}$ and each state $s \in \mathrm{St}$, we associate a non-empty set $\mathrm{Ac}_i(s)$ of *feasible* actions that, intuitively, i can perform when in state s. By a *direction* or *decision* we understand a profile of actions $d = (a_1, \ldots, a_n)$ in $\mathrm{Ac} \times \cdots \times \mathrm{Ac}$ and we let Dir denote the set of directions. A direction $d = (a_1, \ldots, a_n)$ is *legal at state* s if $a_i \in \mathrm{Ac}_i(s)$ for all players i. Unless stated otherwise, by "direction" we will henceforth generally mean "legal direction". Furthermore, $\lambda \colon \mathrm{St} \to 2^{\mathrm{AP}}$ is a *labelling function*, associating with every state s a *valuation* $v \in 2^{\mathrm{AP}}$. Finally, δ is a *deterministic transition function*, which associates with each state s and every legal direction $d = (a_1, \ldots, a_n)$ at s a state $\delta(a_1, \ldots, a_n)$. As such δ characterises the behaviour of the system when $d = (a_1, \ldots, a_n)$ is performed at state s.

Computations, Runs, and Traces

The possible behaviours exhibited by a concurrent game structure can be described at at least three different levels of abstraction. Thus, we distinguish between *computations*, *runs*, and *traces*. Computations carry the most information, while traces carry the least, in the

sense that every computation induces a unique run and every run induces a unique trace, but not necessarily the other way round.

A state s' is *accessible* from another state s whenever there is some $d = (a_1, \ldots, a_n)$ such that d is legal at s and $\delta(s, a_1, \ldots, a_n) = s'$. For easy readability we then also write $s \xrightarrow{d} s'$. An *(infinite) computation* is then an infinite sequence of directions $\kappa = d_0, d_1, d_2, \ldots$ such that there are states $s_0, s_1, \ldots$ such that $s_0 = s^0_M$ and $s_0 \xrightarrow{d_0} s_1 \xrightarrow{d_1} s_2 \xrightarrow{d_2} \cdots$. Observe that in every concurrent game model the states $s_0, s_1, \ldots$ in the above definition always exist and are unique. A *finite computation* is finite prefix of a computation κ. We also allow a finite computation to be the empty sequence ϵ of directions. We also use $\delta^*(s, d_0, d_1, \ldots d_k)$ to denote the unique state that is reached from the state s after applying the computation $d_0, d_1, \ldots d_k$.

An *(infinite) run* is an infinite sequence $\rho = s_0, s_1, s_2 \ldots$ of states of sequentially accessible states, with $s_0 = s^0_M$. We say that run $s_0, \ldots, s_k$ is *induced* by computation $d_0, \ldots, d_{k-1}$ if $s_0 \xrightarrow{d_0} s_1 \xrightarrow{d_1} s_2 \xrightarrow{d_2} \cdots$ and $s_0 = s^0_M$. Thus, every computation induces a unique run and every run is induced by at least one computation. A *finite run* is a finite prefix of a run. The sets of infinite and finite runs are denoted by $runs^\omega_M$ and $runs_M$, respectively.

An *(infinite) trace* is a sequence $\tau = v_0, v_1, v_2, \ldots$ of valuations such that there is a run $\rho = s_0, s_1, s_2, \ldots$ in $runs^\omega_M$ such that $v_k = \lambda(s_k)$ for every $k \geq 0$, that is, $\tau = \lambda(s_0), \lambda(s_1), \lambda(s_2), \ldots$. In that case we also say that trace τ is *induced* by run ρ, and if ρ is induced by computation κ, also that τ is induced by κ. By a *finite trace* we mean a finite prefix of a trace. We denote the sets of finite and infinite traces of a concurrent game structure M by $traces_M$ and $traces^\omega_M$, respectively. We use $\rho_M(\kappa)$ to denote the run induced by an infinite computation κ in M, and $\pi_M(\kappa)$, if κ is finite, on the understanding that $\pi_M(\epsilon) = s^0_M$. Also, if $\rho = s_0, s_1, s_2, \ldots$ is a run, by $\tau_M(\rho)$ we denote the trace $\lambda(s_0), \lambda(s_1), \lambda(s_2), \ldots$, and similarly for finite runs $\pi \in runs_M$. Finally, $\tau_M(\rho_M(\kappa))$ is abbreviated as $\tau_M(\kappa)$. When no confusion is likely, we sometimes omit the subscript M and the qualification 'finite'.

Bisimulations and Bisimilarity

One of the most important behavioural/observational equivalences in concurrency is bisimilarity, which is usually defined over Kripke structures or labelled transition systems (see, for instance, [18, 16]). However, the equivalence can be uniformly defined for general concurrent game structures, where decisions/directions play the role of "actions" in transition systems. Formally, let $M = (\mathrm{AP}, \mathrm{Ag}, \mathrm{Ac}, \mathrm{St}, s^0_M, \lambda, \delta)$ and $M' = (\mathrm{AP}, \mathrm{Ag}, \mathrm{Ac}, \mathrm{St}', s^0_{M'}, \lambda', \delta')$ be two concurrent game structures. A *bisimulation*, denoted by $\sim$, between states $s^* \in \mathrm{St}$ and $t^* \in \mathrm{St}'$ is a non-empty binary relation $R \subseteq \mathrm{St} \times \mathrm{St}'$, such that $s^* \; R \; t^*$ and for all $s, s' \in \mathrm{St}$, $t, t' \in \mathrm{St}'$, and $d \in \mathrm{Dir}$:

- $s \; R \; t$ implies $\lambda(s) = \lambda'(t)$,
- $s \; R \; t$ and $s \xrightarrow{d} s'$ implies $t \xrightarrow{d} t''$ for some $t'' \in \mathrm{St}'$ with $s' \; R \; t''$,
- $s \; R \; t$ and $t \xrightarrow{d} t'$ implies $s \xrightarrow{d} s''$ for some $s'' \in \mathrm{St}$ with $s'' \; R \; t'$.

Then, if there is a bisimulation between two states s^* and t^*, we say that they are *bisimilar* and write $s^* \sim t^*$ in such a case. We also say that concurrent game structures M and M' are *bisimilar*—in symbols $M \sim M'$—if $s^0_M \sim s^0_{M'}$. We furthermore say that runs $\rho = s_0, s_1, \ldots$ and $\rho' = s'_0, s'_1, \ldots$ are *statewise bisimilar* (in symbols $\rho \dot{\sim} \rho'$) if $s_k \sim s'_k$ for every $k \geq 0$. Both bisimilarity and statewise bisimilarity are equivalence relations, which is a standard result in the literature (see, for instance, [11, 5, 18]). We, moreover, find that the sets of (finite) computations as well as the sets of (finite) traces of two bisimilar concurrent game structures are *identical*. Moreover, every (finite) computation κ gives rise to statewise bisimilar (finite)

runs and identical (finite) traces in bisimilar concurrent game structures. However, as runs are sequences of states and the states of different concurrent game structures M and M' may be distinct, even if they are bisimilar, no identification of their sets $runs^{\omega}_{M}$ and $runs^{\omega}_{M'}$ of runs can generally be made.

3 Games on Concurrent Game Structures

Concurrent game structures specify the actions the players can take at each state and which states are reached if they all concurrently decide on an action. A full understanding of the system that is being modelled, however, also depends on what goals the players desire to achieve and on what strategies they may adopt in pursuit of these goals. We therefore augment concurrent game structures with preferences and strategies for the players. Thus they define a strategic game and as such they are amenable to game theoretic analysis by standard solution concepts, among which Nash equilibrium is probably the most prominent.

3.1 Strategies and Strategy Profiles

Based on the distinction between computations, runs, and traces, we can also distinguish three types of strategy: computation-based, run-based, and trace-based strategies. The importance of these distinctions is additionally corroborated by Bouyer et al. [6], who show how the specific model of strategies adopted affects the computational complexity of some standard decision problems related to multi-agent systems.

A *computation-based strategy* for a player i in a concurrent game structure M is a function $f_i^{comp}\colon comps_M \to \mathrm{Ac}$ such that $f_i^{comp}(\kappa) \in \mathrm{Ac}_i(s_k)$ for every finite $\kappa \in comps_M$ with $\pi_M(\kappa) = s_0, \ldots, s_k$. Thus, $f_i^{comp}(\epsilon) \in \mathrm{Ac}_i(s^0_M)$, where ϵ is the empty sequence of directions.

Similarly, a *run-based strategy* for player i is a function $f_i^{run}\colon runs_M \to \mathrm{Ac}$ where $f_i^{run}(s_0, \ldots, s_k) \in \mathrm{Ac}_i(s_k)$ for every finite run $(s_0, \ldots, s_k) \in runs$. Finally, a *trace-based strategy* for i is a function $f_i^{trace}\colon traces_M \to \mathrm{Ac}$ such that $f_i^{trace}(\tau) \in \mathrm{Ac}_i(s_k)$ for every trace $\tau \in traces_M$ and every run $\pi = s_0, \ldots, s_k$ such that $\tau = \lambda(s_0), \ldots, \lambda(s_k)$.

A *computation-based strategy profile* is then a tuple $f = (f_1, \ldots, f_n)$ that associates with each player i a computation-based strategy f_i. Run-based and trace-based strategy profiles are defined analogously. Every computation-based strategy profile $f = (f_1, \ldots, f_n)$ induces a unique computation $\kappa_M(f) = d_0, d_1, d_2, \ldots$ in M that is defined inductively such that $d_0 = (f_1(\epsilon), \ldots, f_n(\epsilon))$ and $d_{k+1} = (f_1(d_0, \ldots, d_k), \ldots, f_n(d_0, \ldots, d_k))$, for all $k \in \mathbb{N}$. Similarly, a run-based strategy profile $f = (f_1, \ldots, f_n)$ defines the unique computation $\kappa_M(f) = d_0, d_1, d_2, \ldots$ such that $d_0 = (f_1(s^0_M), \ldots, f_n(s^0_M))$ and $d_{k+1} = (f_1(\pi(d_0, \ldots, d_k)), \ldots, f_n(\pi(d_0, \ldots, d_k))$, for all $k \in \mathbb{N}$. Finally, the computation $\kappa_M(f)$ defined by a trace-based strategy profile f is such that $d_0 = (f_1(\lambda(s^0_M)), \ldots, f_n(\lambda(s^0_M)))$ and $d_{k+1} = (f_1(\tau(d_0, \ldots, d_k)), \ldots, f_n(\tau(d_0, \ldots, d_k))$, for all $k \in \mathbb{N}$. For $f = (f_1, \ldots, f_n)$ a profile of computation-based, run-based, or trace-based strategies, we write with a slight abuse of notation $\rho(f_1, \ldots, f_n)$ for $\rho(\kappa(f_1, \ldots, f_n))$ and $\tau(f_1, \ldots, f_n)$ for $\tau(\rho(f_1, \ldots, f_n))$.

Note that, as the computations and traces of bisimular concurrent games structures coincide, so do the computation-based and trace-based strategies available to the players. Moreover, the computations induced by them will be identical. With the states of bisimilar structures possibly being distinct, however, similar observations do not straightforwardly extend to run-based strategies and strategy profiles.

3.2 Preferences and Goals

We formally specify the preferences of a player i of a concurrent game structure M as a subset Γ_i of *computations*, that is, $\Gamma_i \subseteq comps^{\omega}_M$, and refer to Γ_i as player i's *goal set*. Player i is then understood to (strictly) prefer computations in Γ_i to those not in Γ_i and to be indifferent otherwise. Accordingly, each player's preferences are dichotomous, only distinguishing between the preferred computations in Γ_i and the not preferred ones not in Γ_i. Formally, player i is said to *weakly prefer* computation κ to computation κ' if $\kappa \in \Gamma_i$ whenever $\kappa' \in \Gamma_i$, and to *strictly prefer* κ to κ' if i weakly prefers κ to κ' but not vice versa. If player i weakly prefers κ to κ' and κ' to κ, we say that i is *indifferent* between κ and κ'.

The choice of modelling players' preferences as sets of *computations*—rather than say sets of runs or sets of traces—is for technical convenience and flexibility. Observe that every set of runs is induced by a set of computations, namely the set of computations that give rise to the same runs, and similarly for every set of traces. Thus, we say that a goal set $\Gamma_i \subseteq comps^{\omega}_M$ is *run-based* if for any two computations κ and κ' with $\rho(\kappa) = \rho(\kappa')$ we have that $\kappa \in \Gamma_i$ if and only if $\kappa' \in \Gamma_i$. Similarly, Γ_i is said to be *trace-based* whenever $\tau(\kappa) = \tau(\kappa')$ implies that $\kappa \in \Gamma_i$ if and only if $\kappa' \in \Gamma_i$. Thus, *run-based* goals are *computation-based* goals closed under induced runs, and *trace-based* goals are *computation-based* goals closed under induced traces.

Sometimes—as we did in the example in the introduction—players' goals are specified by *temporal logic formulae*. As the satisfaction of goals only depends on traces, they will directly correspond to trace-based goals, given our formalisation of goals and preferences.

3.3 Games and Nash Equilibrium

With the above definitions in place, we now define a *game on a concurrent game structure* M (also called a *CGS-game*) as a tuple, $G = (M, \Gamma_1, \ldots, \Gamma_n)$, where, for each player i, the goal set $\Gamma_i \subseteq comps^{\omega}_M$ specifies i's dichotomous preferences over the computations in M.

The players of a CGS-game either all play computation-based strategies, all play run-based strategies, or all play trace-based strategies. For each such choice of type of strategies, with the set of players and their preferences specified, every CGS-game defines a strategic game in the standard game theoretic sense. Observe that the set of strategies is infinite in general. Thus the game theoretic solution concept of Nash equilibrium becomes available for the analysis of games on concurrent game structures. If $f = (f_1, \ldots, f_n)$ is a strategy profile and g_i a strategy for player i, we write (f_{-i}, g_i) for the strategy profile $(f_1, \ldots, g_i, \ldots, f_n)$, which is identical to f except that i's strategy is replaced by g_i. Formally, given a CGS-game, we say that a profile $f = (f_1, \ldots, f_n)$ of computation-based strategies is a *Nash equilibrium in computation-based strategies* (or *computation-based equilibrium*) if, for every player i and every computation-based strategy g_i available to i, $\kappa_M(f_{-i}, g_i) \in \Gamma_i$ implies $\kappa_M(f) \in \Gamma_i$. The concepts of *Nash equilibrium in run-based strategies* and *Nash equilibrium in trace-based strategies* are defined analogously, where, importantly, f_i and g_i are required to be of the same type, that is, both run-based or both trace-based. If $\kappa(f) \notin \Gamma_i$ whereas $\kappa(f_{-i}, g_i) \in \Gamma_i$, we also say that player i would like to *deviate* from f_i to g_i. Thus, a run-based profile f is a *run-based equilibrium* whenever no player would like to deviate from it to some *run-based* strategy different from f_i. Similarly, a trace-based profile f is a trace-based equilibrium if no player would prefer to deviate to another *trace-based* strategy.

We furthermore say that a computation κ, run ρ, or a trace τ is *sustained by a Nash equilibrium* $f = (f_1, \ldots, f_n)$ (of any type) whenever $\kappa = \kappa(f)$, $\rho = \rho(f)$, and $\tau = \tau(f)$, respectively. We also refer to a computation, run, or trace that is sustained by a Nash equilibrium as an *equilibrium computation*, *equilibrium run*, and *equilibrium trace*, respectively.

Computation-based equilibrium is a weaker notion than run-based equilibrium, in the sense that, if f is a run-based equilibrium, there is also a corresponding computation-based equilibrium, but not necessarily the other way round. Run-based equilibrium, in turn, is in a similar way a weaker concept than trace-based equilibrium.

4 Invariance of Nash Equilibria under Bisimilarity

From a computational point of view, one may expect games based on bisimilar concurrent game structures and with identical players' preferences to display very similar behaviours, in particular with respect to their Nash equilibria. We find that while this is indeed the case for games with computation-based strategies as well as for games with trace-based strategies, for games with run-based strategies the situation is considerably more complicated. A key observation is that, by contrast to computation-based and trace-based strategies, there need not be a natural *one-to-one* mapping between the sets of run-based strategies in bisimilar concurrent game models. By restricting to so-called bisimulation-invariant run-based strategies, however, we find that order can be restored.

4.1 Computation-based and Trace-based Strategies

If strategies are computation-based, players can have their actions depend on virtually all information that is available in the system. In an important sense full transparency prevails and different actions can be chosen on bisimilar states provided that the computations that led to them are different. As, moreover, the strategies available to player in bisimilar concurrent game structures are identical, we obtain our first main preservation result.

▶ **Theorem 1.** *Let $G = (M, \Gamma_1, \ldots, \Gamma_n)$ and $G' = (M', \Gamma_1, \ldots, \Gamma_n)$ be games on bisimilar concurrent game structures M and M', respectively, and let $f = (f_1, \ldots, f_n)$ be a computation-based profile. Then, f is a Nash equilibrium in computation-based strategies in G if and only if f is a Nash equilibrium in computation-based strategies in G'.*

Since the sets of traces of two bisimilar concurrent game structure coincide, we can also directly compare their trace-based Nash-equilibria. We find that trace-based Nash equilibria are likewise preserved in CGS-games based on bisimilar concurrent game structures.

▶ **Theorem 2.** *Let $G = (M, \Gamma_1, \ldots, \Gamma_n)$ and $G' = (M', \Gamma_1, \ldots, \Gamma_n)$ be games on bisimilar concurrent game structures M and M', respectively, and $f = (f_1, \ldots, f_n)$ be a trace-based strategy profile. Then, f is a Nash equilibrium in trace-based strategies in G if and only if f is a Nash equilibrium in trace-based strategies in G'.*

As an immediate consequence of Theorems 1 and 2, we also find that the properties of computations and traces being sustained by, respectively, a computation-based equilibrium and a trace-based equilibrium are preserved under bisimulation.

4.2 Run-based Strategies

The positive results obtained using computation-based and trace-based strategies (with respect to both computation-based goals and trace-based goals) are now followed by a negative result, already mentioned in the introduction of the paper, which establishes that Nash equilibria in run-based strategies—probably the most widely-used strategy model in logic, computer science, and AI—are not preserved by bisimilarity. Previously we observed that the players' run-based strategies cannot straightforwardly be identified across two

different but bisimilar concurrent game structures. We would therefore have to establish a correspondence between the run-based strategies of different games in an arguably ad hoc way. To cut this Gordian knot, we therefore show the stronger result that the very *existence* of run-based equilibria is not preserved under bisimilarity. That is, there can be two CGS-games, G and G', that are based on bisimilar concurrent game structures but that G possesses a Nash equilibrium and G' does not.

▶ **Theorem 3.** *The existence of run-based Nash equilibria is* not *preserved under bisimilarity. That is, there are games* $(M, \Gamma_1, \ldots, \Gamma_n)$ *and* $(M', \Gamma_1, \ldots, \Gamma_n)$ *on bisimilar concurrent game structures* M *and* M' *such that a run-based equilibrium exists in* G *but not in* G'*.*

To see that the above statement holds, recall the three-player game G_0 on the concurrent game structure M_0 in Figure 1. Assume, as before, that player 1's goal set Γ_1 is given by those computations κ such that $\tau_{M_0}(\kappa) = v_0, v_1, v_2, \ldots$, implies $p \in v_k$ for some $k \geq 0$. Similarly, Γ_2 consists of all computations κ with $\tau_{M_0}(\kappa) = v_0, v_1, v_2, \ldots$ and $q \in v_k$ for some $k \geq 0$ and Γ_3 by those computations κ with $\tau_{M_0}(\kappa) = v_0, v_1, v_2, \ldots$ and $v_k = \emptyset$ for all $k \geq 0$. Recall that, consequently, the preferences of players 1, 2, and 3 are trace-based and can be represented by the LTL formulas $\gamma_1 = \mathsf{F}\, p$, $\gamma_2 = \mathsf{F}\, q$, and $\gamma_3 = \mathsf{G}\, \neg(p \vee q)$, respectively.

Let f_1^* and f_2^* be any run-based strategies for players 1 and 2 such that $f_1^*(s_0) = f_2^*(s_0) = a$. Let, furthermore, player 3's run-based strategy f_3^* be such that $f_3^*(s_0) = a$, $f_3^*(s_0, s_1) = a'$, and $f_3^*(s_0, s_1') = b$. Let $f^* = (f_1^*, f_2^*, f_3^*)$ and observe that $\rho_{M_0}(f^*) = s_0, s_4, s_4, s_4, \ldots$. Accordingly, player 3 has her goal achieved and does not want to deviate from f^*. Players 1 and 2 do not have their goals achieved, but do not want to deviate from f^* either. To see this, let g_1 be any run-based strategy for 1 such that $g_1(s_0) = b$; observe that this is required for any meaningful deviation from f^* by 1. Then $\rho_{M_0}(g_1, f_2^*, f_3^*) = s_0, s_1, s_3, s_3, s_3, \ldots$ or $\rho_{M_0}(g_1, f_2^*, f_3^*) = s_0, s_1, s_4, s_4, s_4, \ldots$, depending on whether $f_2^*(s_0, s_1) = b$ or $f_2^*(s_0, s_1) = a$, respectively. In either case, player 1 does not get her goal achieved and it follows that she does not want to deviate from f^*. An analogous argument—notice that the roles of player 1 and 2 are symmetric—shows that player 2 does not want to deviate from f^* either. We may thus conclude that f^* is a run-based equilibrium in G_0.

Now, consider the game G_1 on concurrent game structure M_1 in Figure 2 with the players' preferences as in M_0. It is easy to check that M_0 and M_1 are bisimilar. Still, there is no run-based equilibrium in G_1. To see this, consider an arbitrary run-based strategy profile $f = (f_1, f_2, f_3)$. First, assume that $\rho_{M_1}(f) = s_0, s_1, s_2, s_2, s_2, \ldots$. Then, player 1 gets his goal achieved and players 2 and 3 do not. If $f_1(s_0, s_1) = a$ then $f_3(s_0, s_1) = b$ and player 3 would like to deviate and play a strategy g_3 with $g_3(s_0, s_1) = a$. On the other hand, if $f_1(s_0, s_1) = b$, player 3 would like to deviate and play a strategy g_3 with $g_3(s_0, s_1) = b$. Player 3 would similarly like to deviate from f if we assume that $\rho_{M_1}(f) = s_0, s_1, s_3, s_3, s_3 \ldots$, in whose case it is player 2 who gets his goal achieved. Now, assume that $\rho_{M_1}(f) = s_0, s_1, s_4, s_4, s_4 \ldots$. In this case player 3 does get her goal achieved, but players 1 and 2 do not. However, player 1 would like to deviate to a strategy g_1 with $g_1(s_0, s_1) = b$ or $g_1(s_0, s_1) = a$, depending on whether $f_3(s_0, s_1) = a$ or $f_3(s_0, s_1) = b$; in a similar fashion, player 2 would like to deviate to a strategy g_2 with $g_2(s_0, s_1) = b$ if $f_1(s_0, s_1) = a'$, and to one with $g_2(s_0, s_1) = a$ if $f_1(s_0, s_1) = b'$. Finally, assume that $\rho_{M_1}(f) = s_0, s_4, s_4, s_4, \ldots$. Then, neither player 1 nor player 2 gets his goal achieved. Now either $f_3(s_0, s_1) \in \{a, b\}$ or $f_3(s_0, s_1) \in \{a', b'\}$. If the former, player 1 would like to deviate to a strategy g_1 with $g_1(s_0) \neq f_1(s_0)$ and $g_1(s_0, s_1) \neq f_3(s_0, s_1)$. If the latter, player 2 would like to deviate to a strategy g_2 with $g_2(s_0) \neq f_2(s_0)$ and either $g_2(s_0, s_1) = b$ if $f_3(s_0, s_1) = a'$ or $g_2(s_0, s_1) = a$ if $f_3(s_0, s_1) = b'$. We can then conclude that the CGS-game G_1 does not have any run-based Nash equilibria.

The main idea behind this counter-example is that in G_0 player 3 can distinguish which player deviates from f^* if the state reached after the first round is not s_4: if that state is s_1, player 1 deviated, otherwise player 2 did. By choosing either a' or b' at s_1, and either a or b at s_1', player 3 can guarantee that neither player 1 nor player 2 gets his goal achieved ("punish" them) and thus deter their deviating from f^*. This possibility to punish deviations from f^* by players 1 and 2 in a single strategy is not available in G_1: choosing from a and b will induce a deviation by player 1, choosing from a' and b' one by player 2.

Observe that the games G_0 and G_1 do *not* constitute a counter-example against the preservation under bisimilarity of either computation-based equilibria or trace-based equilibria. The reasons why such games fail to do so, however, are distinct. For the setting of computation-based strategies, player 3 can still distinguish and "punish" the deviating player in G_1 as (a, b, a) and (b, a, a) are different directions and player 3 can still have his action at s_1 depend on which of these is played at s_0. By contrast, if we assume trace-based strategies, player 3 has to choose the same action at both s_1 and s_1' in G_0. As a consequence, and contrarily to computation-based equilibria, trace-based equilibria exist in neither G_0 nor G_1.[1] Also note that, as the goal sets Γ_1, Γ_2, and Γ_3 are run-based as well as computation-based both in G_1 and G_2, the counter-example still holds if preferences are more fine-grained.

Observe at this point that s_1 and s_1' are bisimilar states. Yet, players are allowed to have run-based strategies (which depend on state histories only) that choose *different* actions at bisimilar states. The above counter-example shows how this relative richness of strategies makes a crucial difference. This raises the question as to whether we actually want players to adopt run-based strategies in which they choose different actions at bisimilar states. This observation leads us to the next section.

4.3 Bisimulation-invariant Run-based Strategies

Bisimilarity formally captures an informal concept of observational indistinguishability on the part of an external observer of the system. The players in a concurrent game structure are in essentially the same situation as an external observer, if they are assumed to be only able to observe the behaviour of the other players, without knowing their internal makeup.

Drawing on this idea of indistinguishability, it is natural to assume that players cannot distinguish statewise bisimilar runs and, as a consequence, can only adopt strategies that choose the same action at runs that are statewise bisimilar. The situation is comparable to the one in extensive games of imperfect information, in which players are required to choose the same action in histories that are in the same information set (cf., for instance, [22, 17]).

To make this idea precise, we say that a run-based strategy f_i is *bisimulation-invariant* if $f_i(\pi) = f_i(\pi')$ for all histories π and π' that are statewise bisimilar. The concept of Nash equilibrium is then similarly restricted to bisimulation-invariant strategies. Formally, a profile $f = (f_1, \ldots, f_n)$ of *bisimulation-invariant* strategies is a *Nash equilibrium in bisimulation-invariant strategies* (or a *bisimulation-invariant equilibrium*) in a game $(M, \Gamma_1, \ldots, \Gamma_n)$ if for all players i and every *bisimulation-invariant* strategy g_i for i, $\tau(f_{-i}, g_i) \in \Gamma_i$ implies $\tau(f) \in \Gamma_i$. In contrast to the situation for general run-based strategies, we find that computations and traces that are sustained by a bisimulation-invariant Nash equilibrium are preserved by bisimulation. Let M and M' be bisimilar concurrent game structures.

[1] Based on a similar example, Almagor et al. [1] also observe that the existence of Nash equilibria in a CGS-game may depend on the type of strategy that is being considered. Note, however, that this is quite different from our observation that Nash equilibria may differ in different but bisimilar CGS-games given a fixed type of strategy, *viz.*, run-based strategies.

Based on the concept of statewise bisimilarity, we associate with every bisimulation-invariant strategy f_i for player i in M, another bisimulation-invariant strategy $\tilde{f}_i$ for player i in M' such that for all $\pi \in runs_{M'}$ and $a \in \mathrm{Ac}$, we have $\tilde{f}_i(\pi) = a$ if $f_i(\pi') = a$ for some $\pi' \in runs_M$ with $\pi \dot{\sim} \pi'$. Transitivity of $\dot{\sim}$ guarantees that $\tilde{f}_i$ is well-defined. Given a profile $f = (f_1, \ldots, f_n)$, let $\tilde{f} = (\tilde{f}_1, \ldots, \tilde{f}_n)$. It can then be shown that f and $\tilde{f}$ induce identical computations, that is, $\kappa_M(f) = \kappa_{M'}(\tilde{f})$, which prepares the ground for the following preservation result.

▶ **Theorem 4.** *Let $G = (M, \Gamma_1, \ldots, \Gamma_n)$ and $G' = (M', \Gamma_1, \ldots, \Gamma_n)$ be games on bisimilar concurrent game structures M and M', respectively. Then, profile f is a bisimulation-invariant equilibrium in G if and only if $\tilde{f}$ is a bisimulation-invariant equilibrium in G'.*

As an immediate corollary of Theorem 4, it follows that the property of a computation or trace to be sustained by a bisimulation-invariant equilibria is also preserved under bisimilarity.

4.4 Two-player Games with Run-based Strategies

We now consider the setting of two-player games with run-based strategies and trace-based goals (which include temporal logic goals). This is an important special case, since run-based strategies, as we emphasised in the introduction, are the conventional model of strategies used in logics such as ATL* or Stragegy Logic (SL), as well as in multi-agent systems represented as concurrent game structures [4, 19].

The counterexample against the preservation of existence of Nash equilibria in Section 4.2 involved three players. We find that, if preferences are computation-based, the example can be adapted so as to involve only two players. Hence, we have the following result.

▶ **Theorem 5.** *There are two-player games (M, Γ_1, Γ_2) and (M', Γ_1, Γ_2) on bisimilar concurrent game structures M and M' with Γ_1 and Γ_2 computation-based such that a run-based Nash equilibrium exists in G but not in G'.*

By contrast, if players' preferences are required to be trace-based in any two models to be compared, sustenance of traces by run-based equilibrium—and hence also the existence of Nash equilibria—is preserved under bisimilarity. To establish this, we associate with each run-based profile $f = (f_1, f_2)$ a *bisimulation-invariant* profile $\hat{f} = (\hat{f}_1, \hat{f}_2)$. We then show that f and $\hat{f}$ induce the same traces and that $\hat{f}$ is a bisimulation invariant equilibrium, if f is a run-based equilibrium. We can then leverage Theorem 4, which yields the result.

We only give the definition of $\hat{f}_1$, as the construction of $\hat{f}_2$ is analogous. The key idea is to select for every finite run $\pi = s_0, \ldots, s_k$ a unique representative $\hat{\pi}^{s_0,\ldots,s_k}$ of the equivalence class $[\pi]_{\dot{\sim}}$ of runs statewise bisimilar to π and then define $\hat{f}_1$ such that $\hat{f}_1(\pi) = f_1(\hat{\pi}^{s_0,\ldots,s_k})$. This ensures that $\hat{f}_1$ is bisimulation-invariant. The representative $\hat{\pi}^{s_0,\ldots,s_k}$, however, has to be chosen carefully. Assuming that from every subset of actions we can always select a least element, we define inductively for every finite run $\pi = s_0, \ldots, s_k$ in $runs_M$ another statewise bisimilar finite run $\hat{\pi}^{s_0,\ldots,s_k} = t_0, \ldots, t_k$ in $runs_M$ such that $t_0 = s^0_M$ and, for every $0 \leq m < k$, $t_{m+1} = \delta(t_m, x_1, x_2)$, where x_1 and x_2 are actions available to players 1 and 2 at t_m such that:

- $x_1 = f_1(t_0, \ldots, t_m)$ and $x_2 = f_2(t_0, \ldots, t_m)$, if $\delta(t_m, f_1(t_0, \ldots, t_m), f_2(t_0, \ldots, t_m)) \sim s_{m+1}$,
- $x_1 = f_1(t_0, \ldots, t_m)$ and x_2 is the least action available to player 2 at t_m such that $\delta(t_m, f_1(t_0, \ldots, t_m), x_2) \sim s_{m+1}$, if $\delta(t_m, f_1(t_0, \ldots, t_m), f_2(t_0, \ldots, t_m)) \not\sim s_{m+1}$ and x_2 exists, and
- (x_1, x_2) is lexicographically the least pair of actions available to players 1 and 2 at t_m such that $\delta(t_m, x_1, x_2) \sim s_{m+1}$, otherwise.

By means of an inductive argument it can easily be shown that $s_0, \ldots, s_k \dot{\sim} \pi^{s_0,\ldots,s_k}$, which suffices for $\hat{\pi}^{s_0,\ldots,s_k}$ to be well-defined, that is, that x_1 and x_2 exist for every $0 \leq m < k$. As, $s_0, \ldots, s_k \dot{\sim} t_0, \ldots, t_k$ implies $\hat{\pi}^{s_0,\ldots,s_k} = \hat{\pi}^{t_0,\ldots,t_k}$, it follows that $\hat{f}_1$ is properly defined as a bisimulation-invariant strategy. Importantly, $\rho_M(\hat{f}_1, \hat{f}_2) = \rho_M(f_1, f_2)$, and hence also $\tau_M(\hat{f}_1, \hat{f}_2) = \tau_M(f_1, f_2)$. Moreover, we find by a non-trivial argument that, if $f = (f_1, f_2)$ is a run-based equilibrium, so is $\hat{f} = (\hat{f}_1, \hat{f}_2)$. Finally, as a bisimulation-invariant profile is a bisimulation-invariant equilibrium if and only if it is a run-based equilibrium, we are in a position to leverage Theorem 4 and obtain our result

▶ **Theorem 6.** *Let $G = (M, \Gamma_1, \Gamma_2)$ and $G' = (M', \Gamma_1, \Gamma_2)$ be two two-player games such that Γ_1 and Γ_2 are trace-based in both M and M' and let τ be a trace in $\mathrm{traces}^{\omega}_M$. Then, τ is sustained by a run-based equilibrium in G if and only if τ is sustained by a run-based equilibrium in G'.*

As an immediate consequence of Theorem 6 we also find that the *existence* of Nash equilibria is also preserved in two-player games with trace-based preferences. The case in which players' preferences are run-based, however, we have to leave as an open question.

5 Strategy Logics: New Semantic Foundations

Several logics for strategic reasoning have been proposed in the literature of computer science and AI, such as ATL* [4], Strategy Logic [19, 8], Coalition Logic [23], Coordination Logic [12], Game Logic [24], and Equilibrium Logic [15]. In several cases, the model of strategies that is used is the one that we refer to as run-based in this paper, that is, strategies are functions from finite sequences of states (of some arena) to actions/decisions/choices of players in a given game. As can be seen from our results so far, of the four options we have explored, run-based strategies form the least desirable model of strategies from a semantic point of view since in such a case Nash equilibrium is not preserved under bisimilarity.

This does not necessarily immediately imply that a particular logic with a run-based strategy model is not invariant under bisimilarity. For instance, ATL* is a bisimulation-invariant logic and, as shown in [13] one can reason about Nash equilibrium using ATL* only up-to bisimilarity. A question then remains: whether any of these logics for strategic reasoning becomes invariant under bisimilarity—as explained before, a desirable property—if one changes the model of strategies considered there to, for instance, computation-based or trace-based strategies. We find that this question has a satisfactory positive answer in some cases. In particular, we will consider the above question in the context of Strategy Logic as studied in [19], and in doing so we will provide new semantic foundations for strategy logics.

Let us start by introducing the syntax and semantics under the run-based model of strategies for Strategy Logic (SL [19]) as it has been given in [20]. Syntactically, SL extends LTL with two *strategy quantifiers*, $\langle\langle x \rangle\rangle$ and $[[x]]$, and an *agent binding* operator (i, x), where i is an agent and x is a variable. These operators can be read as *"there exists a strategy x"*, *"for all strategies x"*, and *"bind agent i to the strategy associated with the variable x"*, respectively. SL formulae are inductively built from a set of atomic propositions AP, variables Var, and agents Ag, using the following grammar, where $p \in \mathrm{AP}$, $x \in \mathrm{Var}$, and $i \in \mathrm{Ag}$:

$$\varphi ::= p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \mathsf{X}\varphi \mid \varphi \mathsf{U} \varphi \mid \langle\langle x \rangle\rangle \varphi \mid [[x]]\varphi \mid (i, x)\varphi.$$

We can now present the semantics of SL formulae, where Str denotes the set of all strategies of some specified type, *i.e.*, computation-based, run-based, or trace-based. Given a concurrent game structure M, for all SL formulae φ, states $s \in \mathrm{St}$ in M, and assignments

$\chi \in \text{Asg} = (\text{Var} \cup \text{Ag}) \to \text{Str}$, mapping variables and agents to strategies, the relation $M, \chi, s \models \varphi$ is defined as follows:

1. For the Boolean and temporal cases, the semantics is standard (see, *e.g.*, [19];
2. For all formulae φ and variables $x \in \text{Var}$ we have:
   a. $M, \chi, s \models \langle\langle x \rangle\rangle \varphi$ if there is a strategy $f \in \text{Str}$ such that $M, \chi[x \mapsto f], s \models \varphi$;
   b. $M, \chi, s \models [[x]]\varphi$ if for all strategies $f \in \text{Str}$ we have that $M, \chi[x \mapsto f], s \models \varphi$.
3. For all $i \in \text{Ag}$ and $x \in \text{Var}$, we have $M, \chi, s \models (i, x)\varphi$ if $M, \chi[i \mapsto \chi(x)], s \models \varphi$.

Intuitively, rules 2a and 2b, respectively, are used to interpret the existential $\langle\langle x \rangle\rangle$ and universal $[[x]]$ quantifiers over strategies, and rule 3 is used to bind an agent to the strategies associated with variable x. All other rules are as in LTL over concurrent game structures. Moreover, for a sentence φ, *i.e.*, a formula with no free variables or agents [19], we say that M satisfies φ, and write $M \models \varphi$, if $M, \emptyset, s_0 \models \varphi$, where $\emptyset$ is the empty assignment.

As it can be seen from its semantics, SL can be interpreted under different models of strategies and goals. As it was originally formulated, SL considers run-based strategies and trace-based preferences/goals [21]. More specifically, the model of goals is a proper subset of the trace-based one, represented by LTL goals over the set AP of variables. Let us fix an interpretation of strategies. We say that SL with that kind of model is invariant under bisimulation if, for all sentences φ and bisimilar models M_1 and M_2, it holds that $M_1 \models \varphi$ iff $M_2 \models \varphi$. In SL, it is possible to represent the existence of a Nash equilibrium in a concurrent game structure [19]. This implies, given Theorem 3, that SL under the standard (run-based model) interpretation is not invariant under bisimulation, as the formula expressing the existence of a Nash equilibrium can distinguish between two bisimilar models.

We now consider SL under the model of computation-based strategies, and find that in such a case SL becomes invariant under bisimilarity. Formally, we have the following result.

▶ **Theorem 7.** *SL with the computation-based model of strategies is invariant under bisimilarity.*

An analogous statement to the above Theorem can also be proved if we consider the model of trace-based strategies, leading to the next result on the semantics of SL.

▶ **Theorem 8.** *SL with the trace-based model of strategies is invariant under bisimilarity.*

6 Concluding Remarks and Related Work

We have shown that in the conventional model of strategies used in logic, computer science, and AI, the existence of a given Nash equilibrium is not preserved under bisimilarity, in particular for multi-player games played over concurrent games structures. With a few examples we also illustrated some of the adverse implications of this result, for instance, in the context of automated formal verification. To resolve this problem, we furthermore investigated alternative models of strategies which exhibit some desirable properties, in particular, *allowing for a formalisation of Nash equilibrium that is invariant under bisimilarity.*

We studied applications of these models and found that through their use, not only Nash equilibria becomes invariant under bisimilarity, but also full logics such as Strategy Logic [21, 8, 19]. This renders it possible to combine commonly used optimisation techniques for model checking with decision procedures for the analysis of Nash equilibria, thus overcoming a critical problem of this kind of logics regarding practical applications via automated verification. Some work also in the intersection between bisimulation equivalences, concurrent game structures, and Nash equilibria is summarised next.

Logics for Strategic Reasoning.

From the current (enormous) set of logics for strategic reasoning in the literature, ATL* [4] and SL [19] stand out, both due to their adoption by a number of practical tools for automated verification and because of their expressive power. On the one hand, ATL* is known to be invariant under bisimilarity using the conventional model of strategies. As such, Nash equilibria can be expressed within ATL* only up to bisimilarity [13]. On the other hand, SL, which is strictly more expressive than ATL*, allows for a simple specification of Nash equilibria, but suffers from not being invariant under bisimilarity with respect to the conventional model of strategies. In this paper, we have put forward a number of solutions to this problem. An additional advantage of replacing the model of strategies for SL (and therefore for concurrent game structures) is that other solution concepts in game theory also become invariant under bisimilarity. For instance, subgame-perfect Nash equilibria and strong Nash equilibria—which are widely used when considering, respectively, dynamic behaviour and cooperative behaviour in multi-agent systems—can also be expressed in SL. Our results therefore imply that these concepts are also invariant under bisimilarity, when considering games over concurrent game structures and goals given by LTL formulae.

Bisimulation Equivalences for Multi-Agent Systems.

Even though bisimilarity is probably the most widely used behavioural equivalence in concurrency, in the context of multi-agent systems other relations may be preferred, for instance, equivalence relations that take a detailed account of the independent interactions and behaviour of individual components in a multi-agent system. In such a setting, "alternating" relations with natural ATL* characterisations have been studied [3, 11]. Our results also apply to such alternating relations. On the one hand, the counter-example shown in Figures 1 and 2 also apply to such alternating (bisimulation) relations. On the other hand, because these alternating relations can be characterised in ATL*, they are at most as strong as bisimilarity. These two facts together imply that Nash equilibria is not preserved by the alternating (bisimulation) equivalence relations in [3, 11] either. Nevertheless, as discussed in [27], the right notion of equivalence for games and their game theoretic solution concepts is most certainly an important and interesting topic of debate.

Computations vs Traces.

A difference between computations and traces is that even though Nash equilibria and their existence are preserved under bisimilarity by three of the four strategy models we have studied, it is not the case that with each strategy model we obtain the same set of Nash equilibria in a given system, or that we can sustain the same set of computations or traces. For instance, consider the games in Figures 1 and 2. As we discussed above, if we consider the model of computation-based strategies and LTL goals (*i.e.*, trace-based goals) as shown in the example, then we obtain two games, each with an associated non-empty set of Nash equilibria, which is preserved by bisimilarity. However, if we consider, instead, the model of trace-based strategies and same LTL goals, then we obtain two games both with empty sets of Nash equilibria—thus, in this case, the non-existence of Nash equilibria is preserved by bisimilarity! To observe this, note that whereas in the case of computation-based strategies player 3 can implement a uniform "punishment" strategy for both player 1 and player 2, in the case of trace-based strategies player 3 cannot do so, even in the game in Figure 1.

Two-Player Games with Trace-Based Goals.

In this paper we also showed that if we consider two-player games together with the conventional model of strategies, the problems that arise with respect to the preservation of Nash equilibria disappear. This is indeed an important finding since most verification games (*e.g.*, model and module checking, synthesis, etc.) can be phrased in terms of two-player games together with temporal logic specifications (*e.g.*, using LTL, CTL, or ATL*). Our results, then, provide proof that, if only two-player games and temporal logic goals are needed, then all equilibrium analyses can be carried out using the conventional model of strategies—along with their associated reasoning tools and verification techniques.

Boolean Game Structures.

Boolean game structures are the special type of concurrent game structure in which each player i has unique control over a subset AP_i of propositional variables and the set $\mathrm{Ac}_i(s)$ of actions available to player i at state s is a non-empty subset of partial valuations in 2^{AP_i}. The key idea is that the choice player i makes with respect to the variables in AP_i at a state s defines the labelling of the subsequent state with respect to AP_i. Formally, for every direction $d' = (a_1, \ldots, a_n)$ in $2^{\mathrm{AP}_1} \times \cdots \times 2^{\mathrm{AP}_n}$ and every state s, it holds that $\delta(s, d') = s'$ implies $\lambda(s') = a_1 \cup \cdots \cup a_n$.

Boolean game structures are a particularly well-behaved class of games, of which *iterated Boolean games* [14] and multi-agent systems modelled using the *Reactive Modules specification language* [2] are special cases. We thus find that, in Boolean game structures, (existence of) Nash equilibrium is invariant under bisimilarity regardless of the model of strategies or goals that one chooses. As in this setting, it is readily shown that if two finite runs are statewise bisimilar, they have to be identical, it immediately follows that run-based strategies are bisimulation-invariant. Hence, by Theorem 4 that (the existence of) run-based equilibria is invariant under bisimulation. For the settings with computation-based and trace-based strategies, Theorems 1 and 2 give the result, respectively.

References

1. S. Almagor, G. Avni, and O. Kupferman. Repairing multi-player games. In L. Aceto and D. de Frutos Escri, editors, *Proceedings of the Twenty-Sixth Annual Conference on Concurrency Theory (CONCUR'15)*, Leibniz International Proceedings in Informatics (LIPIcs), pages 325–339, Madrid, Spain, 2015.
2. R. Alur and T. A. Henzinger. Reactive modules. *Formal Methods in System Design*, 15(1):7–48, 1999.
3. R. Alur, T. A. Henzinger, O. Kupferman, and M. Y. Vardi. Alternating refinement relations. In *Proceedings of the 9th International Conference on Concurrency Theory (CONCUR'98)*, volume 1466 of *LNCS*, pages 163–178. Springer-Verlag, Berlin, Germany, 1998.
4. R. Alur, T.A. Henzinger, and O. Kupferman. Alternating-Time Temporal Logic. *Journal of the ACM*, 49(5):672–713, 2002.
5. C. Baier and J.-P. Katoen. *Principles of Model Checking*. MIT Press, 2008.
6. P. Bouyer, R. Brenguier, N. Markey, and M. Ummels. Nash equilibria in concurrent games with büchi objectives. In *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'11), Mumbai, India*, pages 375–386, 2011.
7. R. E. Bryant. Symbolic Boolean manipulation with ordered binary-decision diagrams. *ACM Computing Surveys*, 24(3):293–318, 1992.
8. K. Chatterjee, T.A. Henzinger, and N. Piterman. Strategy logic. *Information and Computation*, 208(6):677–693, 2010.

9 Edmund M. Clarke, Orna Grumberg, and David E. Long. Model checking and abstraction. *ACM transactions on Programming Languages and Systems*, 16(5):1512–1542, 1994.

10 P. Cousot and R. Cousot. On abstraction in software verification. In *Forteenth International Conference on Computer Aided Verification (CAV'02)*, volume 2404 of *LNCS*, pages 37–56. Springer, 2002.

11 S. Demri, V. Goranko, and M. Lange. *Temporal Logics in Computer Science: Finite State Systems*, volume 58 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 2016.

12 B. Finkbeiner and S. Schewe. Coordination logic. In Anuj Dawar and Helmut Veith, editors, *International Workshop on Computer Science Logic (CSL'10)*, volume 6247 of *LNCS*, pages 305–319. Springer, 2010.

13 J. Gutierrez, P. Harrenstein, and M. Wooldridge. Expressiveness and complexity results for strategic reasoning. In *Proceedings of the Twenty-Sixth Annual Conference on Concurrency Theory (CONCUR'15)*, volume 42 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 268–282, Madrid, Spain, 2015.

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

15 J. Gutierrez, P. Harrenstein, and M. Wooldridge. Reasoning about equilibria in game-like concurrent systems. *Annals of Pure and Applied Logic*, 169(2):373–403, 2017.

16 M. Hennessy and R. A. Connolly Milner. Algebraic laws for nondeterminism and concurrency. *Journal of the ACM*, 32(1):137–161, 1985.

17 M. Maschler, E. Solan, and S. Zamir. *Game Theory*. Cambridge University Press, 2013.

18 R. Milner. *Communication and Concurrency*. Prentice Hall, 1989.

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

20 F. Mogavero, A. Murano, G. Perelli, and M. Y. Vardi. Reasoning about Strategies: On the Satisfiability Problem. *Logical Methods in Computer Science*, 13(1:9), 2017.

21 F. Mogavero, A. Murano, and M. Y. Vardi. Reasoning about strategies. In K. Lodaya and M. Mahajan, editors, *Proceedings of the IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'10)*, volume 8 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 133–144, 2010.

22 M.J. Osborne and A. Rubinstein. *A Course in Game Theory*. MIT Press, 1994.

23 M. Pauly. A modal logic for coalitional power in games. *Journal of Logic and Computation*, 12(1):149–166, 2002.

24 M. Pauly and R. Parikh. Game logic—an overview. *Studia Logica*, 75(2):165–182, 2003.

25 Davide Sangiorgi. On the origins of bisimulation and coinduction. *ACM Transactions on Programming Languages and Systems*, 31(4):111–151, 2009.

26 J. F. A. K. van Benthem. *Modal Correspondence Theory*. PhD thesis, University of Amsterdam, 1976.

27 J. F. A. K. van Benthem. Extensive games as process models. *Journal of Logic, Language and Information*, 11(3):289–313, 2002.

28 M. Wooldridge, J. Gutierrez, P. Harrenstein, E. Marchioni, G. Perelli, and A. Toumi. Rational verification: From model checking to equilibrium checking. In D. Schuurmans and M. Wellman, editors, *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI'16)*, pages 4184–4190, Phoenix, AZ, 2016.

29 M.J. Wooldridge. *Introduction to Multiagent Systems*. John Wiley & Sons, 2001.