NASH EQUILIBRIUM AND BISIMULATION INVARIANCE

JULIAN GUTIERREZ, PAUL HARRENSTEIN, GIUSEPPE PERELLI, AND MICHAEL WOOLDRIDGE

Department of Computer Science, University of Oxford
e-mail address: julian.gutierrez@cs.ox.ac.uk

Department of Computer Science, University of Oxford
e-mail address: paul.harrenstein@cs.ox.ac.uk

Department of Computer Science, University of Göteborg
e-mail address: perelli@chalmers.se

Department of Computer Science, University of Oxford
e-mail address: 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 setting involves computing the set of (Nash) equilibria in the concurrent game. However, we show that, with respect to the above model of strategies (arguably, the "standard" model in the computer science 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 logic formulae, may nevertheless have fundamentally different properties (solutions) from a game theoretic perspective. Our aim in this paper is to explore the issues raised by this discovery. After illustrating the issue by way of a motivating example, we present three models of strategies with respect to which the existence of Nash equilibria is preserved under bisimilarity. We 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 computer science literature.

1. INTRODUCTION

The concept of *bisimilarity* plays a central role in both the theory of concurrency [34, 28] and logic [44, 28]. In the context of concurrency, bisimilar systems are regarded as *behaviourally equivalent*—appearing to have the same behaviour when interacting with an 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 [40, 14, 29]. These features, in turn, make it possible to verify temporal logic

properties of concurrent systems using bisimulation-based approaches [42]. For example, temporal logic model checking techniques [13] may be optimised by applying them to the smallest bisimulation equivalent model of the system being analysed; or, indeed, to any 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 [13] is not the only verification technique that can benefit from bisimulation invariance: consider abstraction and refinement techniques [15, 16] (where a set of states is either collapsed or broken down in order to build a somewhat simpler set of states); coinduction methods [41] (which can be used to check the correctness of an implementation with respect to a given specification); or reduced BDD representations of a system [10] (where isomorphic, and therefore bisimilar, subgraphs are merged, thereby eliminating part of the initial state space of the system). Bisimulation invariance is therefore a powerful and fundamental concept in the formal analysis and verification of concurrent and multi-agent systems, which plays an important role in many verification tools.

Game theory provides another important framework for the analysis and verification 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 (that is, 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. A widely-used model for strategies in concurrent games is to view a strategy for a process/agent/player 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 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 [37]). For instance, specification languages such as Alternating-time Temporal Logic (ATL [1]), and formal models such as concurrent game structures [1] 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 game; in some cases, the analysis reduces to the computation of a winning strategy in the game, that is, a strategy that ensures that the players who follow such a plan will achieve their goal no matter how the other players in the system play, *i.e.*, against any other possible counter-strategy.

Now, 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 can be identified in a similar way; 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, we find 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 multi-player (concurrent) games.

For instance, given a concurrent and reactive system, represented as a collection of individual system components, say $P_1, \ldots, P_n$, one may want to know if a given temporal logic property, say φ, is satisfied by these system components whenever they choose to

use strategies that form an equilibrium, that is, we want to know whether for some/every computation run $\rho \in NE(P_1, \ldots, P_n)$ we have $\rho \models \varphi$, where $NE(P_1, \ldots, P_n)$ denotes the set of all computation runs that may be generated as a result of $P_1, \ldots, P_n$ selecting strategies that form a Nash equilibrium. Because we are interested in concurrent systems, and bisimilarity is one of the most important behavioural equivalences in concurrency [33, 28, 17, 47], it is highly desirable that properties which hold in equilibrium are sustained across all systems that are bisimilar to $P_1, \ldots, P_n$, meaning that for every (temporal logic) property φ and every process P_i', if P_i' is bisimilar to $P_i \in \{P_1, \ldots, P_n\}$, then φ is satisfied in equilibrium by $P_1, \ldots P_i \ldots P_n$ if and only if is also satisfied in equilibrium by $P_1, \ldots P_i' \ldots, P_n$, the system in which P_i is replaced by P_i', that is, across all bisimilar systems to $P_1, \ldots, P_n$. This property, called *invariance under bisimilarity*, has been widely used for decades for the semantic analysis (*e.g.*, for modular and compositional reasoning) and formal verification (*e.g.*, for temporal logic model checking) of concurrent systems. Unfortunately, as shown here, and already discussed in [22], the satisfaction of temporal logic properties in equilibrium is not invariant under bisimilarity, thus posing a verification challenge for the modular and compositional reasoning of concurrent systems, since individual system components in a concurrent system cannot be replaced by (behaviourally equivalent) bisimilar ones, while preserving the temporal logic properties that the overall system satisfies in equilibrium. This is also a problem from a synthesis point of view. Indeed, a strategy for a system component P_i may not be a valid strategy for a bisimilar system component P_i'. As a consequence, the problem of building strategies for individual processes in the concurrent system $P_1, \ldots P_i \ldots P_n$ may not, in general, be the same as building strategies for a bisimilar system $P_1, \ldots P_i' \ldots P_n$, again, dashing any hope of modular reasoning on concurrent systems.

Motivated by these observations—which bring 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 these issues in detail. We first present a motivating example, to illustrate the basic point that using the conventional model of strategies, bisimulation need not preserve Nash equilibria. We then present three alternative models of strategies in which Nash equilibria and their existence are preserved under bisimilarity. We also study the above question for different classes of systems, for instance deterministic and nondeterministic ones, 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 [35, 12], in particular, the semantic implications with respect to Strategy Logic (SL [35]). 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 [23], a framework that can be used to reason about the strategic behaviour of AI, autonomous, and multi-agent systems [48]. Our main invariance results are summarised in Table 1.

1.1. **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. Perhaps the simplest of such optimisations—as is being done by virtually every model checking tool—is to reduce the input system by merging *isomorphic*

subtrees. This is done, *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.
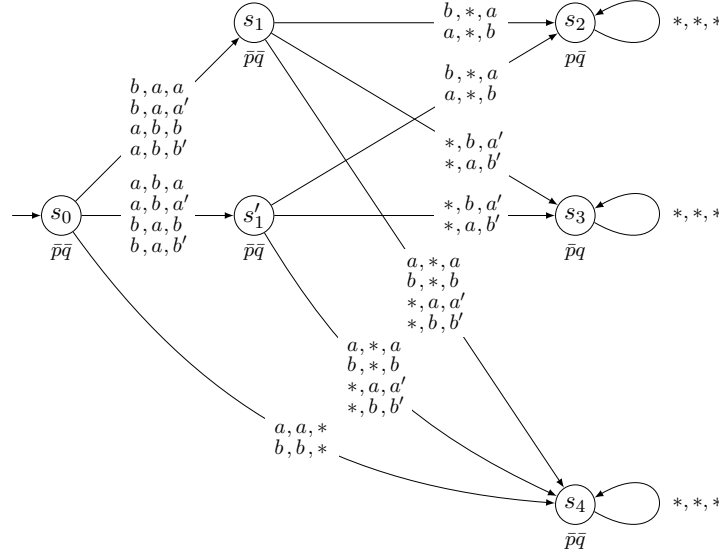

Figure 1: The game G_0 on concurrent game structure M_0 with a Nash equilibrium.

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, and 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 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, respectively, be formally represented by the LTL formulae

$$\gamma_1 = \mathbf{F}p, \qquad\qquad \gamma_2 = \mathbf{F}q, \qquad \text{and} \qquad \gamma_3 = \mathbf{G}\neg(p \vee q),$$

where, informally, $\mathbf{F}\varphi$ means "eventually φ holds" and $\mathbf{G}\varphi$ means "always φ holds". Moreover, given these players' goals and the conventional model of strategies, we will see later in Section 4 that the system in Figure 1 has a Nash equilibrium, whereas no Nash equilibria exists in the (bisimilar) concurrent system presented in Figure 2.

This example illustrates a major issue when analysing (the existence of) Nash equilibria in the most widely used models of strategies and multi-player games in the computer science literature, namely, that even the simplest and most innocuous optimisations commonly used in automated verification are not necessarily sound with respect to game-theoretic analyses.

Because the problem is so fundamental, one may wonder whether bisimilarity is not the right behavioural equivalence for multi-player games, or whether Nash equilibrium is not the right solution concept for game-theoretic analyses of concurrent and multi-agent systems modelled as multi-player games. We will discuss these questions in more detail
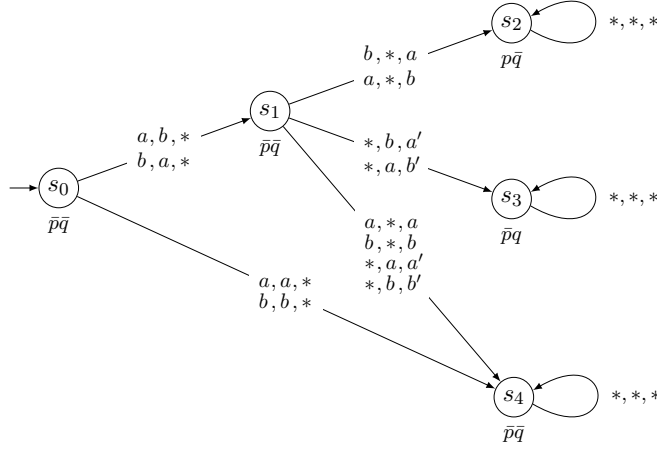
Figure 2: The game G_1 on concurrent game structure M_1 without a Nash equilibrium.

in Section 8, as we do not have a definite answer, but for now we would like to make a couple of observations. On the one hand, that our results also hold both for "alternating" (bisimilarity) relations, as defined in [3], which are intended to capture strategic behaviour in multi-player games, as well as for trace equivalence, as defined in CSP [9], an equivalence much weaker than bisimilarity. On the other hand, that our negative results also hold for solution concepts stronger than Nash equilibrium, *e.g.*, for strong and subgame-perfect Nash equilibria, suggesting that the problem is not a particular defect of Nash equilibrium. Indeed, we think that the issue underlying the mismatch between bisimilarity and Nash equilibrium lies elsewhere. We will propose a very general solution to this problem, that is, a way to reconcile bisimilarity and Nash equilibrium, based on a new definition of strategy in a multi-player game. To do this, some concepts and definitions have to be introduced first.

2. Preliminaries

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

Sets. Given any set $S = \{s, q, r, \ldots\}$, we use S^*, S^ω, and S^+ for, respectively, the sets of finite, infinite, and non-empty finite sequences of elements in S. If $w_1 \in S^*$ and w_2 is any other (finite or infinite) sequence, we write $w_1 w_2$ for their concatenation. The empty sequence is denoted by ϵ.

Concurrent Game Structures. We use the model of concurrent game structures, which are well-established in the logic and computer science literatures (see, for instance, [1]). A *concurrent game structure (CGS)* is a tuple $M = (\mathrm{Ag}, \mathrm{AP}, \mathrm{Ac}, \mathrm{St}, s_M^0, \lambda, \delta)$, 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_M^0. 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 every player 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 $\delta(s, a_1, \ldots, a_n)$. 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 CGS can be described at at least three different levels of abstraction. In what follows, 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. *The distinctions we make between computations, runs, and traces may appear to be insignificant, but are in fact central in our analysis of bisimilarity and Nash equilibrium.*

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 $s_0, s_1, \ldots$ with $s_0 = s_M^0$ and $s_0 \xrightarrow{d_0} s_1 \xrightarrow{d_1} s_2 \xrightarrow{d_2} \cdots$. Observe that, having assumed the transition function δ to be complete and deterministic, 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 any finite prefix of a computation κ. We also allow a finite computation to be the empty sequence ϵ of directions. The sets of infinite and finite computations are denoted by $comps_M^\omega$ and $comps_M$, respectively. 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 $s_0 = s_M^0$. 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 $s_0 = s_M^0$. Thus, every computation induces a unique run and every run is induced by at least one computation. By a *finite run* or *(finite) history* we mean a finite prefix of a run. The sets of infinite and finite runs are denoted by $runs_M^\omega$ 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 $runs_M^\omega$ 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 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 $traces_M^\omega$, respectively.

We use $\rho_M(\kappa)$ to denote the run induced by a computation κ in CGS M, and write $\pi_M(\kappa)$ if κ is finite on the understanding that $\pi_M(\epsilon) = s_M^0$. 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 omit the subscript M and the qualification 'finite'.

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, *e.g.*, [34, 28]). However, the equivalence can be uniformly defined

for general concurrent game structures, where decisions/directions play the role of, for instance, actions in transition systems. Formally, let $M = (\text{AP}, \text{Ag}, \text{Ac}, \text{St}, s_M^0, \lambda, \delta)$ and $M' = (\text{AP}, \text{Ag}, \text{Ac}, \text{St}', s_{M'}^0, \lambda', \delta')$ be two concurrent game structures. A *bisimulation*, denoted by $\sim$, between states $s^* \in \text{St}$ and $t^* \in \text{St}'$ is a non-empty binary relation $R \subseteq \text{St} \times \text{St}'$, such that $s^* R t^*$ and for all $s, s' \in \text{St}$, $t, t' \in \text{St}'$, and $d \in \text{Dir}$:

- $s R t$ implies $\lambda(s) = \lambda'(t)$,
- $s R t$ and $s \xrightarrow{d} s'$ implies $t \xrightarrow{d} t''$ for some $t'' \in \text{St}'$ with $s' R t''$,
- $s R t$ and $t \xrightarrow{d} t'$ implies $s \xrightarrow{d} s''$ for some $s'' \in \text{St}$ 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 $s_M^0 \sim s_{M'}^0$. Since the transition functions of concurrent game structures, as defined, are deterministic, we have the following simple but useful facts. We say that runs $\rho = s_0, s_1, \ldots$ and $\rho' = s_0', s_1', \ldots$ are *statewise bisimilar* (in symbols $\rho \stackrel{.}{\sim} \rho'$) 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, [18, 5, 34]).

We find, moreover, that the sets of (finite) computations as well as the sets of (finite) traces of two bisimilar concurrent game structures are *identical*. In order to see this, the following simple auxiliary result is useful.

Lemma 2.1. *Let $M \sim M'$, $s, s' \in \text{St}$ and $t, t' \in \text{St}'$, and d a direction. Then, $s \sim t$, $s \xrightarrow{d} s'$, and $t \xrightarrow{d} t'$ together imply $s' \sim t'$.*

Proof. Assume $s \sim t$, $s \xrightarrow{d} s'$, and $t \xrightarrow{d} t'$. As $M \sim M'$, there is a $t'' \in \text{St}$ such that $t \xrightarrow{d} t''$ and $s' \sim t''$. Since the transition function is deterministic, moreover, it follows that $t'' = t'$. Hence, $s' \sim t'$, as desired. $\square$

Using this observation we also have the following result.

Lemma 2.2. *Let M and M' be bisimilar concurrent game structures. Then,*

(1) $comps_M^\omega = comps_{M'}^\omega$ *and* $comps_M = comps_{M'}$,

(2) $traces_M^\omega = traces_{M'}^\omega$ *and* $traces_M = traces_{M'}$.

Proof. For part 1, let $\kappa = d_0, d_1, \ldots, d_2, \ldots$ be a computation in $comps_M^\omega$ and $s_0, s_1, s_2, \ldots$ the states in St_M such that $s_0 \xrightarrow{d_0} s_1 \xrightarrow{d_1} s_2 \xrightarrow{d_2} \cdots$. We show, by induction on k, that there are states $t_0, t_1, t_2, \ldots$ in $\text{St}_{M'}$ such that $t_0 \xrightarrow{d_0} t_1 \xrightarrow{d_1} t_2 \xrightarrow{d_2} \cdots$, where $t_0 = s_{M'}^0$. It suffices to prove by induction on k that for every $k \geq 0$ there is a t_{k+1} such that $t_0 \xrightarrow{d_0} \ldots \xrightarrow{d_k} t_{k+1}$ and $s_{k+1} \sim t_{k+1}$. For $k = 0$, we have $t_0 = s_{M'}^0$. Then, observe that, by definition, $s_0 = s_M^0$ and, as $M \sim M'$, it immediately follows that $s_0 \sim t_0$ and that there is a t_1 such that $t_0 \xrightarrow{d_0} t_1$ and $s_1 \sim t_1$. For the induction step, we may assume that there are $t_0, \ldots, t_k$ with $s_{M'}^0 = t_0 \xrightarrow{d_0} \ldots \xrightarrow{d_{k-1}} t_k$ and $s_k \sim t_k$. By bisimilarity of M and M' we then immediately obtain that there is a t_{k+1} such that $t_0 \xrightarrow{d_0} \ldots \xrightarrow{d_{k-1}} t_k \xrightarrow{d_k} t_{k+1}$. By Lemma 2.1 it then follows that $s_{k+1} \sim t_{k+1}$. Hence, $comps_M^\omega \subseteq comps_{M'}^\omega$. As the inclusion in the opposite direction is proven by an analogous argument, we may conclude that $comps_M^\omega = comps_{M'}^\omega$. It also follows that $comps_M = comps_{M'}$, the latter being defined as the finite prefixes of $comps_M^\omega$ and $comps_{M'}^\omega$, respectively.

Observe that from the argument in part 1 it also follows that $\rho_M(\kappa) \stackrel{.}{\sim} \rho_{M'}(\kappa)$ for every $\kappa \in comps^\omega_M = comps^\omega_{M'}$. For part 2, consider an arbitrary trace $\tau \in traces^\omega_M$. Then, there is a computation $\kappa \in comps^\omega_M$ such that $\tau_M(\kappa) = \tau$. By part 1, also $\kappa \in comps^\omega_{M'}$. Moreover, $\rho_M(\kappa) \stackrel{.}{\sim} \rho_{M'}(\kappa)$. By the definition of (statewise) bisimilarity it then follows that $\tau = \tau_M(\kappa) = \tau_{M'}(\kappa)$. Accordingly, $traces^\omega_M \subseteq traces^\omega_{M'}$ and the inclusion in the opposite direction ensues by an analogous argument. We then conclude that $traces_M = traces_{M'}$, the latter being defined as the finite prefixes of $traces^\omega_M$ and $traces^\omega_{M'}$, respectively. □

Moreover, every (finite) computation κ gives rise to statewise bisimilar (finite) runs and identical (finite) traces in bisimilar concurrent game structures.

Lemma 2.3. *Let M and M' be bisimilar concurrent game structures and $\kappa \in comps^\omega_M$ and $\kappa' \in comps_M$. Then,*

(1) *$\rho_M(\kappa) \stackrel{.}{\sim} \rho_{M'}(\kappa)$ and $\pi_M(\kappa') \stackrel{.}{\sim} \pi_{M'}(\kappa')$,*

(2) *$\tau_M(\kappa) = \tau_{M'}(\kappa)$ and $\tau_M(\kappa') = \tau_{M'}(\kappa')$.*

Proof. For part 1, first observe that by virtue of Lemma 2.2, we also have that $\kappa \in comps^\omega_{M'}$ and $\kappa \in comps_{M'}$. Let $\kappa = d_0, d_1, d_2, \ldots$, $\rho_M(\kappa) = s_0, s_1, s_2, \ldots$, and $\rho_{M'}(\kappa) = t_0, t_1, t_2, \ldots$. We prove by induction on k that $s_k \sim t_k$ for every $k \geq 0$. If $k = 0$, then $s_0 = s^0_M \sim s^0_{M'} = t_0$. For the induction step, we may assume that $s_k \sim t_k$. Then, $s_k \xrightarrow{d_k} s_{k+1}$ and $t_k \xrightarrow{d_k} t_{k+1}$. Lemma 2.1 now yields $s_{k+1} \sim t_{k+1}$, as desired. The argument for the second part of 1 proceeds by an analogous argument.

Part 2 then follows almost immediately from part 1. Let $\rho_M(\kappa) = s_0, s_1, s_2, \ldots$ and $\rho_{M'}(\kappa) = t_0, t_1, t_2, \ldots$. Now observe that for every $k \geq 0$ we have that $\lambda_M(s_k) = \lambda_{M'}(t_k)$. Accordingly, $\tau_M(\kappa) = \tau_{M'}(\kappa)$. For κ' a similar argument yields the result. □

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. In game theoretic terms, these structures loosely correspond to what are called *game forms*. A full understanding of the game-theoretic aspects of the system and the strategic behaviour of its constituent players—and therefore which computations/runs/traces will be generated in equilibrium—also essentially 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. In this way CGSs define fully fledged strategic games and as such they are amenable to game theoretic analysis by standard solution concepts, among which Nash equilibrium is arguably the most prominent.

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, 7], 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 $\pi_M(\kappa) = s_0, \dots, s_k$. Thus, in particular, $f_i^{comp}(\epsilon) \in \mathrm{Ac}_i(s_M^0)$, 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 $f_i^{run}(s_0, \dots, s_k) \in \mathrm{Ac}_i(s_k)$ for every finite run $(s_0, \dots, s_k) \in runs_M$. 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 $\pi = s_0, \dots, s_k$ such that $\tau = \lambda(s_0), \dots, \lambda(s_k)$.

A *computation-based strategy profile* is then a tuple $f = (f_1, \dots, f_n)$ 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 $f = (f_1, \dots, f_n)$ induces a unique computation
$$\kappa_M(f) = d_0, d_1, d_2, \dots$$
in M that is defined inductively as follows:
$$d_0 = (f_1(\epsilon), \dots, f_n(\epsilon))$$
$$d_{k+1} = (f_1(d_0, \dots, d_k), \dots, f_n(d_0, \dots, d_k)).$$
A run-based strategy profile $f = (f_1, \dots, f_n)$ defines a unique computation $\kappa_M(f) = d_0, d_1, d_2, \dots$ in a similar manner:
$$d_0 = (f_1(s_M^0), \dots, f_n(s_M^0)), \text{ and}$$
$$d_{k+1} = (f_1(\pi(d_0, \dots, d_k)), \dots, f_n(\pi(d_0, \dots, d_k))).$$
Finally, the computation $\kappa_M(f)$ defined by a trace-based strategy profile f is given by
$$d_0 = (f_1(\lambda(s_M^0)), \dots, f_n(\lambda(s_M^0)))$$
$$d_{k+1} = (f_1(\tau(d_0, \dots, d_k)), \dots, f_n(\tau(d_0, \dots, d_k))).$$
If M is clear from the context, we usually omit the subscript in $\kappa_M(f)$. For $f = (f_1, \dots, f_n)$ a profile of computation-based, run-based, or trace-based strategies, we write with a slight abuse of notation $\rho(f_1, \dots, f_n)$ for $\rho(\kappa(f_1, \dots, f_n))$ and $\tau(f_1, \dots, f_n)$ for $\tau(\rho(f_1, \dots, f_n))$.

As the computations of bisimilar concurrent games structures coincide (Lemma 2.2), we can now establish that a player's computation-based strategies coincide in bisimilar concurrent game structures. Moreover, the computations induced by them will be identical. Also, from the coincidence of traces between bisimilar concurrent game structures (Lemma 2.2), we can establish also trace-based strategies coincide in bisimilar concurrent game structures.

Lemma 3.1. *Let M and M' be bisimilar concurrent game structures and i a player. Then, every computation-based strategy for i in M is also a computation-based strategy for i in M', and every trace-based strategy for i in M is also a trace-based strategy for i in M'. Moreover, for every computation-based profile f for M we have that $\kappa_M(f) = \kappa_{M'}(f)$, and for every trace-based profile g that $\kappa_M(g) = \kappa_{M'}(g)$.*

Proof. First, let f_i be a computation-based strategy for i in M. We show that f_i is also a computation-based strategy for i in M'. To this end, consider an arbitrary $\kappa \in comps_{M'}$. Let $\pi_{M'}(\kappa) = t_0, \ldots, t_k$. It suffices to prove that $f_i(\kappa) \in \text{Ac}_i(t_k)$. To see this, first observe that by Lemma 2.2 also $\kappa \in comps_M$ and let $\pi_M(\kappa) = s_0, \ldots, s_k$. In virtue of Lemma 2.1, then $s_k \sim t_k$. Moreover, because f_i is a computation-based strategy for i in M, we have $f_i(\kappa) \in \text{Ac}_i(s_k)$. Now consider any legal direction $d_k = (a_1, \ldots, a_n)$ at s_k with $a_i = f_i(\kappa)$. Then, there is some state $s_{k+1} \in \text{St}_M$ such that $s_k \xrightarrow{d_k} s_{k+1}$. As $s_k \sim t_k$, moreover, there is also a state $t_{k+1} \in \text{St}_{M'}$ such that $t_k \xrightarrow{d_k} t_{k+1}$. Accordingly, d_k is legal at t_k in M' and in particular $a_i = f_i(\kappa) \in \text{Ac}_i(t_k)$ as desired.

The case if g_i is a trace-based strategy for i in M is similar. We then have to prove that g_i is also a trace-based strategy for i in M' as well. To this end, consider an arbitrary finite trace $\tau \in traces_{M'}$ and run $\pi = t_0, \ldots, t_k$ such that $\tau = \lambda_{M'}(t_0), \ldots, \lambda_{M'}(t_k)$. It then suffices to prove that $g_i(\tau) \in \text{Ac}_i(t_k)$. We may assume that π is induced by a computation $\kappa \in comps_{M'}$, that is, $\pi = \pi_{M'}(\kappa)$. By Lemma 2.2 we have $\kappa \in comps_M$ and by Lemma 2.3 both $\pi_M(\kappa) \overset{\cdot}{\sim} \pi_{M'}(\kappa)$ and $\tau_M(\kappa) = \tau_{M'}(\kappa)$. Let $\pi_M(\kappa) = s_0, \ldots, s_k$. Hence, $s_k \sim t_k$. As g_i is a run-based strategy for i in M we have $g_i(\tau) \in \text{Ac}_i(s_k)$. Let, furthermore, $d = (a_1, \ldots, a_k)$ be a legal direction at s_k with $a_i = g_i(\tau)$. Then, there is some state $s_{k+1} \in \text{St}_M$ such that $s_k \xrightarrow{d_k} s_{k+1}$. As $s_k \sim t_k$, there is also a state $t_{k+1} \in \text{St}_{M'}$ such that $t_k \xrightarrow{d_k} t_{k+1}$. Accordingly, d_k is legal at t_k in M' and in particular $a_i = g_i(\tau) \in \text{Ac}_i(t_k)$.

For the second part of the lemma, let $f = (f_1, \ldots, f_n)$ be a computation-based strategy profile in M. Then, f is a computation-based strategy profile in M' as well. Let $\kappa_M(f) = d_0, d_1, d_2, \ldots$ and $\kappa_{M'}(f) = d'_0, d'_1, d'_2, \ldots$. We show by induction on k that for every $k \geq 0$ we have $d_k = d'_k$. For $k = 0$, immediately, $d_0 = (f_1(\epsilon), \ldots, f_n(\epsilon)) = d'_0$. For the induction step we may assume that $d_0, \ldots, d_k = d'_0, \ldots, d'_k$. Hence,

$$d_{k+1} = (f_1(d_0, \ldots, d_k), \ldots, f_n(d_0, \ldots, d_k)) = (f_1(d'_0, \ldots, d'_k), \ldots, f_n(d'_0, \ldots, d'_k)) = d'_{k+1}.$$

Finally, let $g = (g_1, \ldots, g_n)$ be a trace-based strategy profile. Again we let $\kappa_M(f) = d_0, d_1, d_2, \ldots$ and $\kappa_{M'}(f) = d'_0, d'_1, d'_2, \ldots$ and show by induction on k that for every $k \geq 0$ we have that $d_k = d'_k$. If $k = 0$, observe that having assumed $M \sim M'$ also $s^0_M = s^0_{M'}$. Accordingly, $\lambda_M(s^0_M) = \lambda_{M'}(s^0_{M'})$ and, hence,

$$d_0 = (g_1(\lambda_M(s^0_M)), \ldots, g_n(\lambda_M(s^0_M)) = (g_1(\lambda_{M'}(s^0_{M'})), \ldots, g_n(\lambda_{M'}(s^0_{M'}))) = d'_{k+1}.$$

For the induction step, we may assume that $d_0, \ldots, d_k = d'_0, \ldots, d'_k$ and by Lemma 2.3, moreover, $\tau_M(d_0, \ldots, d_k) = \tau_{M'}(d'_0, \ldots, d'_k)$. Now the following equations hold:

$$d_{k+1} = (g_1(\tau_M(d_0, \ldots, d_k)), \ldots, g_n(\tau_M(d_0, \ldots, d_k)))$$
$$= (g_1(\tau_{M'}(d'_0, \ldots, d'_k)), \ldots, g_n(\tau_{M'}(d'_0, \ldots, d'_k)))$$
$$= d'_{k+1},$$

which concludes the proof. ☐

With the states of bisimilar structures possibly being distinct, however, a statement analogous to Lemma 3.1 cannot be shown to hold for run-based strategies.

Preferences and Goals. We assume the agents of a concurrent game structure to have preferences on basis of which they choose their strategies. Formally, we specify the preferences of a player i of a CGS M as a subset Γ_i of *computations*, that is, $\Gamma_i \subseteq comps^\omega_M$ and refer to Γ_i as 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 the other way round. If i both weakly prefers κ to κ' and weakly prefers κ' to κ, player i is said to be *indifferent* between κ and κ'.

Our choice to assume the players' preferences to be *computation-based* preferences—that is, to model their goals as sets of *computations* rather than, say, sets of runs or sets of traces—is for technical convenience and flexibility. Recall that every set of runs induces 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 $\kappa' \in \Gamma$. Similarly, Γ_i is said to be *trace-based* whenever $\tau(\kappa) = \tau(\kappa')$ implies $\kappa \in \Gamma_i$ if and only if $\kappa' \in \Gamma_i$. In other words, in our setting, formally, *run-based* goals are *computation-based* goals closed under induced runs, and *trace-based* goals are *computation-based* goals closed under induced traces.[1]

Sometimes—as we did in the example in the introduction—players' goals are specified by *temporal logic formulae* [18]. As the satisfaction of goals only depends on traces, they will directly correspond to trace-based goals, given our formalisation of goals and preferences.

Games and Nash Equilibrium. With the above definitions in place, we are now in a position to define a *game on a concurrent game structure M* (also called a *CGS-game*) with $\mathrm{Ag} = \{1, \ldots, n\}$ as a tuple

$$G = (M, \Gamma_1, \ldots, \Gamma_n),$$

where, for each player i in M, the set $\Gamma_i \subseteq comps^\omega_M$ is a goal set specifying i's dichotomous preferences over the computations in M.

In a CGS-game the players can all play either computation-based strategies, run-based strategies, or 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

---

[1] We do not directly consider sets of runs or sets of traces as possible models of players' preferences in this paper—formally, they are induced sets of computations. Accordingly, when talking about preferences, we need not make the distinction between 'run-based' ('trace-based') and 'run-invariant' ('trace-invariant') as we do for strategies. Our run-based preferences and trace-based preferences can with as much justification be referred to as run-invariant preferences and trace-invariant preferences, respectively.

every computation-based strategy g_i available to i,

$$\kappa_M(f_{-i}, g_i) \in \Gamma_i \text{ implies } \kappa_M(f) \in \Gamma_i.$$

The concepts of *Nash equilibrium in run-based strategies* and *Nash equilibrium in trace-based strategies* are defined analogously, where, importantly, the strategies in f_{-i} and g_i are required to be of the same type, that is, either they are all run-based or they are all 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* (and play g_i instead). Thus, a run-based profile f is a *run-based equilibrium* whenever no player would like to deviate from it and play some *run-based* strategy different from f_i. Similarly, a trace-based profile f is a trace-based equilibrium if no player likes to deviate and play another *trace-based* strategy.

We say that a computation κ, run ρ, or a trace τ is *sustained by a Nash equilibrium $f = (f_1, \dots, f_n)$* (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. As computation-based, run-based, and trace-based strategies are set-theoretically of different types, a comparison cannot be made directly. To make the comparison precise, we therefore identify two subclasses of computation-based strategies, *run-invariant strategies* and *trace-invariant strategies*, that characterise the behaviour of, respectively, run-based and trace-based strategies.

We say that a computation-based strategy $f_i \colon comps_M \to Ac_i$ is *run-invariant* in CGS M whenever $\pi_M(\kappa) = \pi_M(\kappa')$ implies $f_i(\kappa) = f_i(\kappa')$, for all computations $\kappa, \kappa' \in comps_M$. Similarly, f_i is *trace-invariant* in M whenever $\tau_M(\kappa) = \tau_M(\kappa')$ implies $f_i(\kappa) = f_i(\kappa')$, for all $\kappa, \kappa' \in comps_M$. Observe that thus a strategy f_i being trace-invariant implies f_i being run-invariant, but not necessarily the other way around.

We observe that there are one-to-one correspondences between run-based strategies on the one hand and run-invariant computation-based strategies on the other, and similarly between trace-based strategies and trace-invariant computation-based strategies. Let $f_i \colon runs_M \to Ac$ be a run-based strategy. Then define $\check{f}_i \colon comps_M \to Ac$ as the computation-based strategy such that for every finite computation $\kappa \in comps_M$ we have $\check{f}_i(\kappa) = f_i(\pi_M(\kappa))$. A similar statment holds if $g_i \colon traces_M \to Ac$ is a trace-based strategy. Then, define $\check{g}_i \colon comps \to Ac_i$ as the computation-based strategy such that for every finite computation $\kappa \in comps_M$ we have $\check{g}_i(\kappa) = g_i(\tau_M(\kappa))$.

Lemma 3.2. *For run-based strategies f_i and trace-based strategies g_i, the mapping that transforms f_i into $\check{f}_i$ and the mapping that transforms g_i into $\check{g}_i$ are both one-to-one.*

Proof. Let f_i a run-based strategy. We first show that $\check{f}_i$ is run-invariant. To this end, let $\kappa, \kappa' \in comps_M$ be computations such that $\pi_M(\kappa) = \pi_M(\kappa')$. Then,

$$\check{f}_i(\kappa) = f_i(\pi_M(\kappa)) = f_i(\pi_M(\kappa')) = \check{f}_i(\kappa').$$

To show that the mapping is onto, let g_i be an arbitrary run-invariant strategy. Now define run-based strategy $\hat{g}_i$ such that, for every run $\pi \in runs_M$ and $\kappa \in comps_M$ with $\pi = \pi_M(\kappa)$ we have $\hat{g}_i(\pi) = g_i(\kappa)$. Observe that $\hat{g}_i$ is well-defined since, , by run-invariance of g_i, for all $\kappa, \kappa' \in comps_M$ with $\pi_M(\kappa) = \pi_M(\kappa') = \pi$ we have that $g_i(\kappa) = g_i(\kappa')$.

Finally, to see that the mapping is injective, let f_i and f_i' be two distinct run-based strategies. Then, there is a run $\pi \in runs_M$ such that $f_i(\pi) \neq f_i'(\pi)$. We may assume the existence of a computation $\kappa \in comps_M$ such that $\pi_M(\kappa) = \pi$. Then,

$$\check{f}_i(\kappa) = f_i(\pi_M(\kappa)) = f_i(\pi) \neq f_i'(\pi) = f_i'(\pi_M(\kappa)) = \check{f}_i(\kappa),$$

as desired. This concludes the proof. $\square$

Furthermore, each profile of run-invariant strategies induces the same computation in a concurrent game structure as its run-based counterpart. A similar remark applies to trace-invariant and trace-based profiles.

Lemma 3.3. *Let* $f = (f_1, \ldots, f_n)$ *be a run-based profile and* $g = (g_1, \ldots, g_n)$ *a trace-based profile. Then,*

$$\kappa_M(f_1, \ldots, f_n) = \kappa_M(\check{f}_1, \ldots, \check{f}_n) \qquad and \qquad \kappa_M(f_1, \ldots, f_n) = \kappa_M(\check{f}_1, \ldots, \check{f}_n).$$

Proof. Let $\kappa_M(f_1, \ldots, f_n) = d_0, d_1, d_2, \ldots$ and $\kappa_M(\check{f}_1, \ldots, \check{f}_n) = d_0', d_1', d_2', \ldots$. We prove by induction on k that $d_0, \ldots, d_k = d_0', \ldots, d_k'$ for every $k \geq 0$.

If $k = 0$, recall that $\pi_M(\epsilon) = s_M^0$. Hence,

$$d_0 = (f_1(s_M^0), \ldots, f_n(s_M^0)) = (f_1(\pi_M(\epsilon)), \ldots, f_n(\pi_M(\epsilon))) = (\check{f}_1(\epsilon), \ldots, \check{f}_n(\epsilon)) = d_0'.$$

For the induction step, we may assume that $d_0, \ldots, d_k = d_0', \ldots, d_k'$. Now the following equalities hold.

$$
\begin{aligned}
d_{k+1} =\ & (f_1(\pi_M(d_0, \ldots, d_k)), \ldots, f_n(\pi_M(d_0, \ldots, d_k))) \\
=\ & (\check{f}_1(d_0, \ldots, d_k), \ldots, \check{f}_n(d_0, \ldots, d_k)) \\
=_{i.h.}\ & (\check{f}_1(d_0', \ldots, d_k'), \ldots, \check{f}_n(d_0', \ldots, d_k')) \\
=\ & d_{k+1}'.
\end{aligned}
$$

We may conclude that $d_0, \ldots, d_{k+1} = d_0', \ldots, d_{k+1}'$. The argument for trace-based and trace-invariant strategies runs along analogous lines, *mutatis mutandis*. $\square$

We say that a computation-based profile $f = (f_1, \ldots, f_n)$ is a *run-invariant equilibrium* in a CGS-game if f is run-invariant and no player i wishes to deviate from f and play another *run-invariant* strategy f_i'. Similarly, a computation-based profile $f = (f_1, \ldots, f_n)$ is a *trace-invariant equilibrium* in a CGS-game if f is trace-invariant and no player i wishes to deviate from f and play another *trace-invariant* strategy f_i'. As an immediate consequence of Lemmas 3.2 and 3.3 we have the following corollary.

Lemma 3.4. *Let* $f = (f_1, \ldots, f_n)$ *and* $g = (g_1, \ldots, g_n)$ *be a run-based profile, respectively, a trace-based profile in a CGS-game* $G = (M, \Gamma_1, \ldots, \Gamma_n)$ *based on* M. *Then,*

(1) f *is a run-based equilibrium if and only if* $\check{f}$ *is a run-invariant equilibrium ,*
(2) g *is a trace-based equilibrium if and only if* $\check{g}$ *is a trace-invariant equilibrium.*

Proof. For the run-based case, the following equivalences hold by virtue of Lemmas 3.2 and 3.3:

f is not a run-based equilibrium in G

iff $\quad \kappa_M(f) \notin \Gamma_i$ and $\kappa_M(f_{-i}, f_i') \in \Gamma_i$ for some run-based strategy f_i' for some player i

iff $\quad \kappa_M(\check{f}) \notin \Gamma_i$ and $\kappa_M(\check{f}_{-i}, \check{f}_i') \in \Gamma_i$ for some run-based strategy f_i' for some player i

iff $\kappa_M(\check{f}) \notin \Gamma_i$ and $\kappa_M(\check{f}_{-i}, f_i'') \in \Gamma_i$ for some run-invariant strategy f_i'' for some player i

iff $\check{f}$ is not a run-invariant equilibrium in G

The proof of the second part is by an analogous argument, *mutatis mutandis*. $\qquad\square$

Computation-based strategies grant a player more strategic flexibility than do run-invariant strategies. A similar remark applies to run-invariant strategies and trace-invariant strategies. Still, we find that, if a player i wishes to deviate from a computation-based profile f and play another computation-based strategy, i would also like to deviate by playing a run-invariant or even a trace-invariant strategy. This insight underlies the following result.[2]

Theorem 3.5. *Let* $f = (f_1, \ldots, f_n)$ *be a run-invariant profile and* $g = (g_1, \ldots, g_n)$ *a trace-invariant profile in CGS-game* $G = (M, \Gamma_1, \ldots, \Gamma_n)$ *based on* M*. Then,*

(1) f *is a run-invariant equilibrium if and only if* f *is a computation-based equilibrium,*
(2) g *is a trace-invariant equilibrium if and only if* g *is a computation-based equilibrium.*

Proof. For part 1, first assume that f is a run-invariant equilibrium in G. For a contradiction assume moreover that f is not a computation-based equilibrium. Then, there is a player i and a computation-based strategy f_i' such that $\kappa_M(f) \notin \Gamma_i$ whereas $\kappa_M(f_{-i}, f_i') \in \Gamma_i$. Let $\kappa_M(f_{-i}, f_i') = d_0', d_1', d_2', \ldots$. Observe that f_i' need not be run-invariant. We therefore define strategy f_i'' for player i such that $f_i''(\epsilon) = f_i'(\epsilon)$ and, for all finite computations $d_0, \ldots, d_k$,

$$f_i''(d_0, \ldots, d_k) = \begin{cases} f_i'(d_0', \ldots, d_k') & \text{if } \pi_M(d_0, \ldots, d_k) = \pi_M(d_0', \ldots, d_k'), \\ f_i(d_0, \ldots, d_k) & \text{otherwise.} \end{cases}$$

As f_i is run-invariant, this definition guarantees that f_i'' is run-invariant as well. Let $\kappa_M(f_{-i}, f_i'') = d_0'', d_2'', d_2'', \ldots$. We prove by induction on k that $d_0', \ldots, d_k' = d_0'', \ldots, d_k''$, for every $k \geq 0$, and hence that $\kappa_M(f_{-i}, f_i') = \kappa_M(f_{-i}, f_i'')$. If $k = 0$, we immediately obtain that

$$d_0' = (f_1(\epsilon), \ldots, f_i'(\epsilon), \ldots, f_n(\epsilon)) = (f_1(\epsilon), \ldots, f_i''(\epsilon), \ldots, f_n(\epsilon)) = d_0''.$$

For the induction step, we may assume that $d_0', \ldots, d_k' = d_0'', \ldots, d_k''$. and the following equalities hold:

$$\begin{aligned} d_{k+1}' &= & (f_1(d_1', \ldots, d_k'), \ldots, f_i'(d_1', \ldots, d_k'), \ldots, f_n(d_1', \ldots, d_k')) \\ &= & (f_1(d_1', \ldots, d_k'), \ldots, f_i''(d_1'', \ldots, d_k''), \ldots, f_n(d_1', \ldots, d_k')) \\ &=_{i.h.} & (f_1(d_1'', \ldots, d_k''), \ldots, f_i''(d_1'', \ldots, d_k''), \ldots, f_n(d_1'', \ldots, d_k'')) \\ &= & d_{k+1}''. \end{aligned}$$

Observe that the second equality holds by virtue of the definition of f_i'' and $\pi_M(d_0', \ldots, d_k') = \pi_M(d_0'', \ldots, d_k'')$. It would follow that $\kappa_M(f_{-i}, f_i'') \in \Gamma_i$ as well, and, as f_i'' is run-invariant, moreover that f is not a run-invariant equilibrium, a contradiction.

For the opposite direction, assume for contraposition that f is not a run-invariant equilibrium. Then, there is some player i who would like to deviate from f and play some run-invariant strategy f_i'. As run-invariant strategies are strategy-based by definition, it follows that f is not a computation-based equilibrium either.

---

[2]The situation can be compared to the relation between equilibria in *pure* and *mixed* (or *randomised*) strategies in game theory. There every equilibrium in pure strategies is also an equilibrium in mixed strategies, because, if a player wishes to deviate from a mixed profile, she wishes to deviate by playing a pure, that is, not randomised, strategy.

Part 2 follows by an analogous argument, *mutatis mutandis.* □

Theorem 3.5 does not preclude the existence of computation-based equilibria that fail to be run-invariant or trace-invariant, that is, the three equilibrium concepts—computation-based, run-invariant, and trace-invariant equilibrium—are not equivalent. However, they can be ordered with respect to how restrictive they are, that is, with respect to the sets of profiles they exclude as solutions.

Corollary 3.6. *Let $f = (f_1, \ldots, f_n)$ be a computation-based profile in some CGS-game $G = (M, \Gamma_1, \ldots, \Gamma_n)$ based on M. Then,*
(1) f is a run-invariant equilibrium implies f is a computation-based equilibrium,
(2) f is a trace-invariant equilibrium implies f is a run-invariant equilibrium.

Proof. Merely observe that if f is a run-invariant equilibrium, it is also a run-invariant profile. If f is moreover trace-invariant it is also run-invariant. The result then immediately follows from Theorem 3.5. □

On basis of the findings in this section, we may with justification claim that every trace-based equilibrium corresponds to a run-based equilibrium, and that every run-based equilibrium corresponds with some computation-based equilibrium, even if the converses of these statements do not generally hold.

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 exhibit similar properties, in particular with respect to their Nash equilibria. We find that that this is indeed the case for games with computation-based strategies as well as for games with trace-based strategies. Recall that (finite) computations and (finite) traces are unaffected by (state-splitting and state-merging) operations on CGSs that preserve bisimilarity (Lemma 2.2). As a consequence the sets of computation-based strategies and trace-based strategies available to an again are the same in bisimilar CGSs (Lemma 3.1), providing the intuitive basis for these observations.

For games with run-based strategies the situation is considerably more complicated. Here, 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 attention to so-called bisimulation-invariant run-based strategies, however, we find that order can be restored.

Invariance under Bisimilarity and Preference Types. We are primarily interested in the Nash equilibria of games that are the same up to bisimilarity of the underlying concurrent game structures. The Nash equilibria of a game, however, essentially depend on the players' preferences. Accordingly, the Nash equilibria of two bisimilar CGS-games can only be meaningfully compared if we also we assume that the players' preferences in these two games are identical. We formalised players' preferences as sets of computations, and, due to Lemma 2.2, this enables a straightforward comparison of players' goal sets across bisimilar concurrent game structures.

In Section 3, we also distinguished run-based and trace-based preferences, that is, goal sets closed under computations that induce the same runs and traces, respectively. We are also interested in the invariance of the existence of Nash equilibria in games on bisimilar
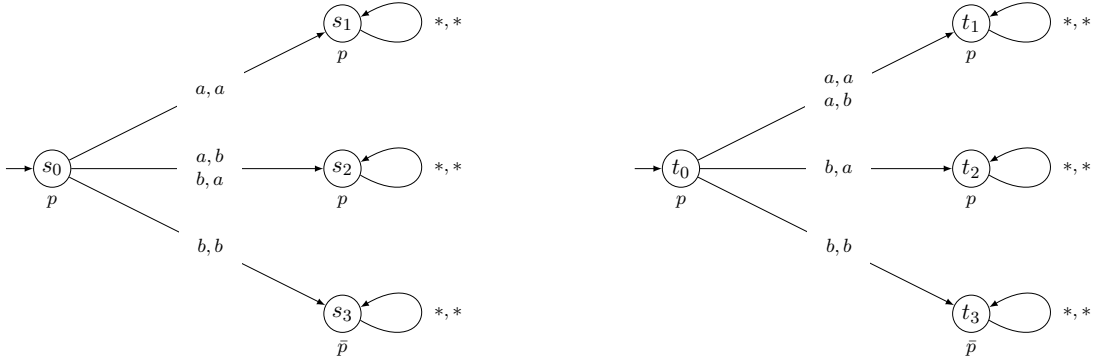
Figure 3: Two games G_2 (left) and G_3 (right) based on M_2 and M_3, respectively, showing that run-based preferences may not be preserved across bisimilar systems.

concurrent game structures where the players' preferences games are what we will call *congruent*, that is, both *the same* and *of the same type in both games*.

For computation-based and trace-based preferences the issue of congruence is moot. Observe that for bisimilar concurrent game structures M and M', if a goal set Γ_i is computation-based in M, then it is also computation-based in M'. Due to Lemma 2.3, the same holds for trace-based preferences.

This preservation of preference type under bisimilarity, however, does not extend to run-based preferences. To see this, consider Figure 3 and let the goal set Γ_i of some player i be given by all computations $\kappa = d_0, d_1, d_2, \ldots$ with $d_0 = (a, a)$. Then, obviously, Γ_i is run-based in the game G_2 based on M_2 on the left, but not in the game G_3 based on M_3 to the right. To see the latter, consider any computation $\kappa' = d'_0, d'_1, d'_2, \ldots$ with $d'_0 = (a, b)$. Then, $\kappa' \notin \Gamma_i$, but, nevertheless, in G_3 we have $\rho_{M_3}(\kappa) = \rho_{M_3}(\kappa')$. By contrast, the goal set given by all computations $\kappa = d_0, d_1, d_2, \ldots$ such that $d_0 \neq (b, b)$ is run-based in both games.

Computation-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. Moreover, the strategies available to players in bisimilar concurrent game structures are identical. Thus we obtain our first main result.

Theorem 4.1. *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'.*

Proof. First assume for contraposition that $f = (f_1, \ldots, f_n)$ is *not* a Nash equilibrium in computation-based strategies in M'. Then, there is a player i and a strategy g_i for i in M' such that $\kappa_{M'}(f) \notin \Gamma_i$ and $\kappa_{M'}(f_{-i}, g_i) \in \Gamma_i$. Observe that, as the computation-based strategies of players in bisimilar structures coincide (cf., Lemma 3.1), f is also a strategy profile in M and g_i a strategy for i in M. Moreover, recall that the computations induced by the same strategy profile in different but bisimilar structures are identical (cf., second part

of Lemma 3.1). This yields $\kappa_M(f) = \kappa_{M'}(f)$ and $\kappa_M(f_{-i}, g_i) = \kappa_{M'}(f_{-i}, g_i)$. Accordingly, $\kappa_M(f) \notin \Gamma_i$ whereas $\kappa_M(f_{-i}, g_i) \in \Gamma_i$. We may conclude that f is not a computation-based equilibrium in M either. The opposite direction follows by an analogous argument. $\square$

Theorem 4.1 holds for computation-based preferences. As run-based preferences and trace-based preferences are computation-based preferences of a special kind, the result immediately extends to games in which the players' preferences are run-based in both games or trace-based preferences in both games. As a consequence of Theorem 4.1, moreover, we find that sustenance of runs and traces by computation-based equilibrium is also preserved under bisimilarity.

Corollary 4.2. *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, $\kappa \in comps_M^\omega$, and $\tau \in traces_M^\omega$. Then,*
(1) *κ is sustained by computation-based equilibrium in G if and only if κ is sustained by a computation-based equilibrium in G'.*
(2) *τ is sustained by a computation-based equilibrium in G if and only if τ is sustained by a computation-based equilibrium in G'.*

Proof. Recall that by Lemma 2.2, we have that $\kappa \in comps_{M'}^\omega$ and $\tau \in traces_{M'}^\omega$. For part 1, let $\kappa_M(f) = \kappa$, where f is a computation-based equilibrium in G'. Then, by Theorem 4.1, profile f is a computation-based equilibrium in G' as well. By virtue of Lemma 3.1, moreover, $\kappa_M(f) = \kappa_{M'}(f)$, which gives the result. The implication in the other direction follows by the same argument *mutatis mutandis*.

The argument for part 2 runs along analogous lines. First assume that τ is sustained by computation-based equilibrium f, that is, $\tau = \tau_M(f)$. By Theorem 4.1, we have that f is a computation-based equilibrium in G' as well. Now consider $\kappa_M(f)$. By Lemma 3.1 then $\kappa_{M'}(f) = \kappa_M(f)$. Lemma 2.3 then yields $\tau_M(\kappa_M(f)) = \tau_{M'}(\kappa_{M'}(f))$. It thus follows that τ is sustained by f, a computation based Nash equilibrium, in G'. The argument in the opposite direction is analogous, giving the result. $\square$

Trace-based Strategies. As we saw in Lemma 2.2, the sets of (finite) traces of two bisimilar concurrent game structures coincide. Lemma 3.1 shows that the same holds for the trace-based strategies that are available to the players. As a consequence, we can directly compare their trace-based Nash-equilibria. We find that, like computation-based equilibria, trace-based Nash equilibria are preserved in CGS-games based on bisimilar concurrent game structures.

Theorem 4.3. *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'.*

Proof. The proof is analogous to the one for Theorem 4.1 for the computation-based case. First assume for contraposition that $f = (f_1, \ldots, f_n)$ is *not* a Nash equilibrium in trace-based strategies in M'. Then, there is a player i and a trace-based strategy g_i for i in M' such that $\kappa(f) \notin \Gamma_i$ and $\kappa(f_{-i}, g_i) \in \Gamma_i$. Observe that, as the trace-based strategies of players in bisimilar structures coincide (cf., Lemma 3.1), we have that f is also a trace-based strategy profile in M and g_i a trace-based strategy for i in M. By the second part of Lemma 3.1, moreover, $\kappa_M(f) = \kappa_{M'}(f)$ and $\kappa_M(f_{-i}, g_i) = \kappa_{M'}(f_{-i}, g_i)$. Accordingly, $\kappa_M(f) \notin \Gamma_i$

whereas $\kappa_M(f_{-i}, g_i) \in \Gamma_i$. We may conclude that f is not a trace-based equilibrium in M either. The opposite direction follows by an analogous argument. □

Like Theorem 4.1, this result is for computation-based preferences in general, and as such immediately extends to the case in which the players' preferences are stipulated to be run-based in both games or trace-based in both games. Theorem 4.3 has further the following result as an immediate consequence, which is analogous to Corollary 4.2.

Corollary 4.4. *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, $\kappa \in comps_M^\omega$, and $\tau \in traces_M^\omega$. Then,*
(1) *κ is sustained by trace-based equilibrium in G if and only if κ is sustained by a trace-based equilibrium in G'.*
(2) *τ is sustained by a trace-based equilibrium in G if and only if τ is sustained by a trace-based equilibrium in G'.*

Proof. The proof is analogous to the one for Corollary 4.2. □

Run-based Strategies. The positive results obtained using computation-based and trace-based strategies are now followed by a negative result, already mentioned in the introduction of the paper, which establishes that Nash equilibria in run-based strategies—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 concurrent game structures, even if they are bisimilar. We would therefore have to establish a correspondence between the run-based strategies in the one game and the run-based strategies in the other in an arguably ad hoc way. To cut this Gordian knot, we therefore show in this section the stronger result that the very *existence* of run-based equilibria is not preserved under bisimilarity. That is, we can have two bisimilar concurrent game structures, say M and M', on which we base two games G and G' with congruent preferences, such that G has a Nash equilibrium and G' does not.

Theorem 4.5. *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 Nash equilibrium in run-based strategies exists in G but not in G'.*

To see that the above statement holds, consider again 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 $\gamma_1 = \mathbf{F}p$, $\gamma_2 = \mathbf{F}q$, and $\gamma_3 = \mathbf{G}\neg(p \vee q)$, 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

$$f_3^*(s_0) = a, \quad f_3^*(s_0, s_1) = a', \quad \text{and} \quad f_3^*(s_0, s_1') = b.$$

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 his goal achieved and it follows that he 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 and play 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 and play 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 and play 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 and play 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 could distinguish which player deviates from f^* if the state reached after the first round is not s_4: if that state is s_1, it was player 1 who deviated, otherwise player 2. 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 them from deviating from f^*. This possibility to punish deviations from f^* by players 1 and 2 in a single strategy is not available in the game on M_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 either the preservation under bisimilarity of computation-based equilibria or the preservation of trace-based equilibria. The reasons why such games fail to do so, however, are different. 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. Also note that 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. Accordingly, the counter-example also applies to settings wherein the players' preferences are assumed to be finer-grained in these two ways.

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.

Bisimulation-invariant Run-based Strategies. Bisimilarity formally captures an informal concept of observational indistinguishability on the part of an external observer of the system. Now, 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 structure or their interaction.

Drawing on this idea of indistinguishability, it is natural 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, that is, histories that cannot be distinguished (cf., *e.g.*, [37, 32]).

To make this idea formally 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. 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,

$$\tau(f_{-i}, g_i) \in \Gamma_i \text{ implies } \tau(f) \in \Gamma_i$$

That is, f is a bisimulation-invariant equilibrium if no player wishes to deviate from f by playing a different bisimulation-invariant strategy. 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. We show this by establishing a one-to-one correspondence between the bisimulation-invariant strategies available to the players in two bisimilar structures.

Based on the concept of state-wise bisimilarity, we associate with every bisimulation-invariant strategy f_i for player i in concurrent game structure M, another bisimulation-invariant strategy $\tilde{f}_i$ for player i in any bisimilar concurrent game structure M' such that for all $\pi \in runs_{M'}$ and $a \in \mathrm{Ac}$,

$$\tilde{f}_i(\pi) = a \quad \text{if } f_i(\pi') = a \text{ for some } \pi' \in runs_M \text{ with } \pi \mathbin{\dot\sim} \pi'.$$

Transitivity of $\dot\sim$ guarantees that $\tilde{f}_i$ is well defined. To see this, observe that for all $\pi', \pi'' \in runs_M$ with $\pi' \mathbin{\dot\sim} \pi$ and $\pi'' \mathbin{\dot\sim} \pi$, we have $\pi' \mathbin{\dot\sim} \pi''$. Having assumed that f_i is bisimulation-invariant, then $f_i(\pi') = f_i(\pi'')$. By very much the same argument, $\tilde{f}_i$ is bisimulation-invariant, if f_i is.

Lemma 4.6. *Let M and M' be bisimilar concurrent game structures and let f_i be a bisimulation-invariant strategy for player i in M. Then, $\tilde{f}_i$ is a bisimulation-invariant strategy in M'.*

Proof. Consider two statewise bisimilar runs $\pi, \pi' \in runs_{M'}$, that is, $\pi \mathbin{\dot\sim} \pi'$. Then, there are computations $\kappa, \kappa' \in comps_{M'}$ such that $\pi_{M'}(\kappa) = \pi$ and $\pi_{M'}(\kappa') = \pi'$. By Lemma 2.3, we have $\pi_M(\kappa) \mathbin{\dot\sim} \pi_{M'}(\kappa)$ and $\pi_M(\kappa') \mathbin{\dot\sim} \pi_{M'}(\kappa')$. Now, transitivity of $\dot\sim$ yields $\pi_M(\kappa) \mathbin{\dot\sim} \pi_M(\kappa')$.

Having assumed that f_i is bisimulation-invariant, we obtain that $f_i(\pi_M(\kappa)) = f_i(\pi_M(\kappa'))$. Finally, we may conclude that $\tilde{f}_i(\pi) = \tilde{f}_i(\pi')$, as desired. $\qquad\square$

Moreover, it is easily appreciated that the mapping that transforms a strategy f_i into strategy $\tilde{f}_i$ is one-to-one. We will find that this is an essential property for bisimulation-invariant equilibria to be preserved under bisimilarity.

For a profile of bisimulation-invariant strategies $f = (f_1, \ldots, f_n)$ in M we denote $\tilde{f} = (\tilde{f}_1, \ldots, \tilde{f}_n)$. We then find that profiles f and $\tilde{f}$ of bisimulation-invariant strategies induce identical computations.

Lemma 4.7. *Let M and M' be bisimilar concurrent game structures, $f = (f_1, \ldots, f_n)$ a bisimulation-invariant strategy profile. Then, $\kappa_M(f) = \kappa_{M'}(\tilde{f})$.*

Proof. Let $\kappa_M(f) = d_0, d_1, d_2, \ldots$ and $\kappa_{M'}(\tilde{f}) = d'_0, d'_1, d'_2, \ldots$. We prove by induction on k that $d_k = d'_k$ for every $k \geq 0$. If $k = 0$, we have $d_0 = (f_1(s^0_M), \ldots, f_n(s^0_M))$. Observe that, as $M \sim M'$ also $s^0_M \sim s^0_{M'}$ and, hence, $f_i(s^0_M) = \tilde{f}_i(s^0_{M'})$. Therefore,

$$d'_0 = (\tilde{f}_1(s^0_{M'}), \ldots, \tilde{f}_n(s^0_{M'})) = (f_1(s^0_M), \ldots, f_n(s^0_M)) = d_0.$$

For the induction step, we may assume that $d_0, \ldots, d_k = d'_0, \ldots, d'_k$. By Lemma 2.3, then $\pi_M(d_0, \ldots, d_k) \mathrel{\dot\sim} \pi_{M'}(d'_0, \ldots, d'_k)$. Accordingly, for every player i we have that $f_i(\pi_M(d_0, \ldots, d_k)) = \tilde{f}_i(\pi_{M'}(d'_0, \ldots, d'_k))$. It thus follows that

$$\begin{aligned} d'_{k+1} &= (\tilde{f}_1(\pi_{M'}(d'_0, \ldots, d'_k)), \ldots, \tilde{f}_n(\pi_{M'}(d'_0, \ldots, d'_k))) \\ &= (f_1(\pi_M(d_0, \ldots, d_k)), \ldots, f_n(\pi_M(d_0, \ldots, d_k))) \\ &= d_{k+1}. \end{aligned}$$

This concludes the proof. $\qquad\square$

We are now in a position to state an equilibrium preservation theorem for bisimulation-invariant strategies in a similar way as we were able to obtain Theorem 4.1, the analogous result for computation-based and trace-based strategies.

Theorem 4.8. *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, f is a bisimulation-invariant equilibrium in G if and only if $\tilde{f}$ is a bisimulation-invariant equilibrium in G'.*

Proof. First assume for contraposition that $\tilde{f} = (\tilde{f}_1, \ldots, \tilde{f}_n)$ is *not* a Nash equilibrium in bisimulation-invariant strategies in G'. Then, there is a player i and a bisimulation-invariant strategy g_i for i in M' such that $\kappa_{M'}(\tilde{f}) \notin \Gamma_i$ and $\kappa_{M'}(\tilde{f}_{-i}, g_i) \in \Gamma_i$. As the mapping that transforms a strategy f_i into strategy $\tilde{f}_i$ is one-to-one, there is a bisimulation-invariant strategy f'_i for i in M with $\tilde{f}'_i = g_i$. Accordingly, $\kappa_{M'}(\tilde{f}_{-i}, \tilde{f}'_i) \in \Gamma_i$. Lemma 4.7 then yields that $\kappa_M(f_{-i}, f'_i) \in \Gamma_i$ and $\kappa_M(f) \notin \Gamma_i$. As f'_i is bisimulation-invariant, it follows that f is not an equilibrium in bisimulation-invariant strategies in G.

The proof in the opposite direction runs along analogous lines. $\qquad\square$

As an immediate corollary of Theorem 4.8, it follows that the property of a computation or trace to be sustained by a bisimulation-invariant equilibria is also preserved under bisimilarity.

Corollary 4.9. *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, for every computation $\kappa \in comps^\omega_M = comps^\omega_{M'}$ and every trace $\tau \in traces^\omega_M = traces^\omega_{M'}$,*

(1) κ *is sustained by a bisimulation invariant equilibrium in G if and only if κ is sustained by a bisimulation-invariant equilibrium in G'.*
(2) τ *is sustained by a bisimulation-invariant equilibrium in G if and only if τ is sustained by a bisimulation-invariant equilibrium in G'.*

Proof. For part 1, assume that $\kappa_M(f) = \kappa$. Then in virtue of Lemma 4.7 also $\kappa_{M'}(\tilde{f}) = \kappa$. Moreover, by Theorem 4.8 we have that profile f is a Nash equilibrium in bisimulation-invariant strategies in G if and only if $\tilde{f}$ is a Nash equilibrium in bisimulation-invariant strategies in G', which gives the result.

The argument for part 2 runs along analogous lines. First assume that τ is sustained by bisimulation-invariant Nash equilibrium f. Let $\kappa = \kappa_M(f)$. Then, $\tau = \tau_M(\kappa)$. By Theorem 4.8, moreover, $\tilde{f}$ is a bisimulation-invariant Nash equilibrium in G'. An application of Lemma 4.7 yields $\kappa_M(\tilde{f}) = \kappa_M(f)$. By Lemma 2.3 then $\tau_M(\kappa) = \tau_{M'}(\kappa)$. It follows that τ is sustained by $\tilde{f}$, a bisimulation-invariant Nash equilibrium, in G'. The argument in the opposite direction is analogous, giving the result. □

5. Special Cases

In the previous section we provided results about the preservation of a given Nash equilibrium under bisimilarity, specifically, as long as we do not consider run-based strategies or goals. In this section we study two important special scenarios where this is not the case.

Firstly, consider the scenario where we have two-player games with run-based strategies and trace-based 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 SL, as well as in systems represented as concurrent game structures. In particular, we show that with respect to two-player games with run-based strategies and trace-based goals (which include temporal logic goals), the setting coincides with the one with bisimulation-invariant strategies and trace-based goals, for which the preservation of Nash equilibria under bisimilarity holds. A key observation in this case is that in two-player games the existence of Nash equilibria can be characterised in terms of the existence of certain winning strategies, which are preserved across bisimilar systems.

Secondly, we also study the scenario where concurrent game structures are restricted to those that are induced by iterated Boolean games [23] and Reactive Modules games [48], two frameworks for the strategic analysis of AI and multi-agent systems, in particular, using model checking techniques.[3] In this case, we show that bisimulation-invariant strategies also coincide with run-based strategies, and therefore, that the positive results for bisimulation-invariant strategies presented in the previous section also transfer to this special case.

Two-Player Games. This section concerns the preservation under bisimilarity of Nash equilibria under bisimulation in two-player games. We deal with the cases in which the players' preferences are computation-based, trace-based, and run-based separately.

---

[3]For instance, Reactive Modules games provide a game semantics to formal specification languages such as Reactive Modules [2], which is widely used in model checking tools, such as MOCHA [4] and PRISM [31].
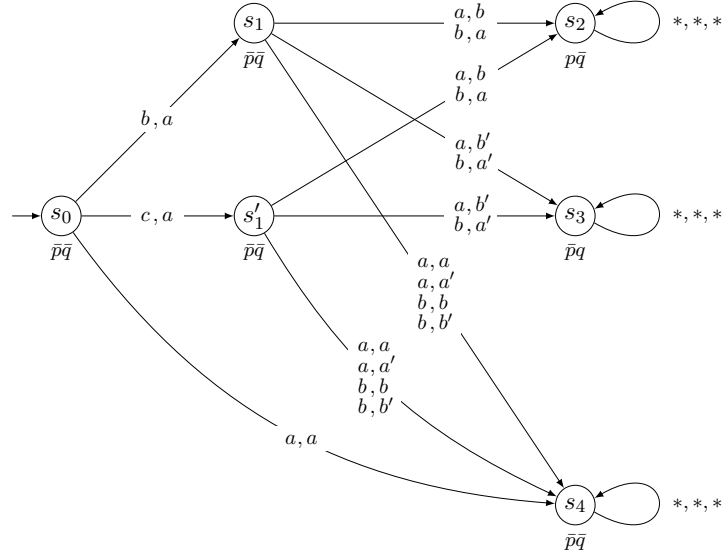
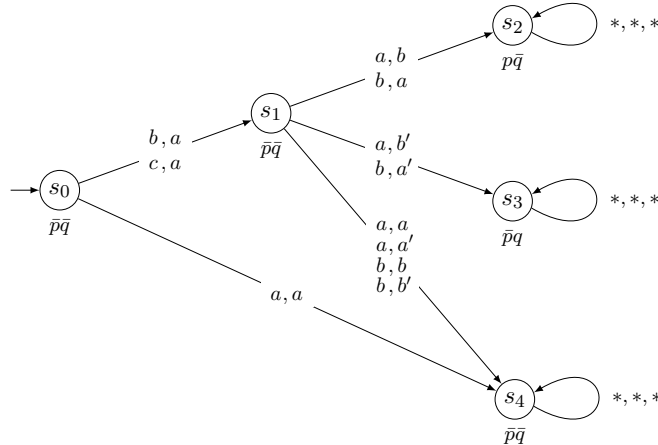Figure 4: The concurrent game structure M_4 underlying the game G_4.

Figure 5: The concurrent game structure M_5 underlying the game G_5.

Computation-based Preferences. The counter-example against the preservation of the existence of Nash equilibria in Section 4 involved three players. We find that, if preferences are computation-based, the example can be adapted so as to involve only two players, which gives rise to the following result.

Theorem 5.1. *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'.*

Proof. Consider the concurrent game structures M_4 and M_5 depicted in Figures 4 and 5, respectively. Observe that $\mathrm{Ac}_1(s_0) = \{a, b, c\}$ and $\mathrm{Ac}_1(s) = \{a, b\}$ at all states s distinct

from s_0, and that $Ac_2(s_0) = \{a\}$ and $Ac_2(s) = \{a, a', b, b'\}$ at all states s distinct from s_0. We define the games $G_4 = (M_4, \Gamma_1, \Gamma_2)$ and $G_5 = (M_5, \Gamma_1, \Gamma_2)$ by letting Γ_1 contain exactly those computations $\kappa = d_0, d_1, d_2, \ldots$ such that either $d_0 = (b, a)$ and $d_1 \in \{(a, b), (b, a)\}$ or $d_0 = (c, a)$ and $d_1 \in \{(a, b'), (b, a')\}$, and letting Γ_2 consist precisely of those computations $\kappa = d_0, d_1, d_2, \ldots$ such that $d_0 = (a, a)$ or $d_1 \in \{(a, a), (a, a'), (b, b), (b, b')\}$. By an argument analogous to that presented in Section 4, it can then be appreciated that G_4 has a run-based equilibrium, whereas G_5 has not.

To see the former, observe that any run-based profile $f^* = (f_1^*, f_2^*)$ will be an equilibrium if $f_1^*(s_0) = f_2^*(s_0) = a$, $f_2^*(s_1) \in \{a', b'\}$, and $f_2^*(s_1') \in \{a, b\}$. For every strategy g_1 for player 1, we then have $\kappa_{M_4}(g_1, f_2^*) \notin \Gamma_1$, whereas $\kappa_{M_4}(f_1^*, f_2^*) \in \Gamma_2$.

To see that G_5 has no run-based equilibrium, first let $f = (f_1, f_2)$ be a run-based profile such that $\kappa_{M_5}(f_1, f_2) \notin \Gamma_1$. Then, if $f_2(s_0, s_1) = a$, player 1 would like to deviate and play a strategy g_1 with $g_1(s_0) = g_1(s_0, s_1) = b$; if $f_2(s_0, s_1) = a'$, to deviate and play a strategy g_1' with $g_1'(s_0) = c$ and $g_1'(s_0, s_1) = b$; if $f_2(s_0, s_1) = b$ to deviate and play a strategy g_1'' with $g_1''(s_0) = b$ and $g_1''(s_0, s_1) = a$; and, finally, if $f_2(s_0, s_1) = b'$ to deviate and play a strategy g_1''' with $g_1'''(s_0) = c$ and $g_1'''(s_0, s_1) = a$. On the other hand, if $\kappa_{M_3}(f_1, f_2) \notin \Gamma_2$, it must be the case that $f_1(s_0) \in \{b, c\}$. Observe, however, that player 2 would then like to deviate and play any strategy g_2 with $g_2(s_0, s_1) = a$ if $f_1(s_0, s_1) = a$ and to a strategy g_2' with $g_2'(s_0, s_1) = b$ if $f_1(s_0, s_1) = b$. As, furthermore, Γ_1 and Γ_2 are disjoint, that is, the goals players 1 and 2 cannot simultaneously be satisfied, it follows that G_5 has no run-based equilibria. □

Run-based Preferences. We now address the preservation (of the existence) of Nash equilibria in two-player CGS-games where both preferences and strategies are run-based. In contrast to our findings in the previous section, we find that, under a natural closure restriction on the players' preferences, we are able to obtain a positive result. Our proof relies on the equivalence of run-based profiles and run-invariant profiles as expounded in Section 3.

As already noted above, run-based strategies cannot generally be identified directly across bisimilar CGS-games. The reason for this complication is that runs are sequences of states, and the sets of states of the two CGS-games need not coincide. In Section 3, we saw, however, how run-based strategies correspond to run-invariant strategies, which are computation-based by definition. Lemma 2.2, moreover, allows us to compare computation-based strategies, even if they may be run-invariant in the one model but not in the other.

Let $f = (f_1, f_2)$ be a given run-invariant equilibrium in a CGS-game $G = (M, \Gamma_1, \Gamma_2)$ based on M and let $G' = (M', \Gamma_1, \Gamma_2)$ be a CGS-game based on a concurrent game structure M' bisimilar to M. We define a (computation-based) profile $f^K = (f_1^K, f_2^K)$ that is a run-invariant equilibrium in both G and G'. To prove that $f^K = (f_1^K, f_2^K)$ is a run-invariant equilibrium if $f = (f_1, f_2)$ is, we exploit a characterisation of Nash equilibria in terms of winning strategies.[4] We say that a run-invariant strategy f_i for player i is *winning against player j* whenever $\kappa_M(f_i, f_j) \notin \Gamma_j$ for all run-invariant strategies f_j of player j. We then have the following lemma, which is independent of the type of preferences that players have.

Lemma 5.2. *Let $G = (M, \Gamma_1, \Gamma_2)$ be a game. Then, a profile $f = (f_1, f_2)$ is a run-invariant equilibrium if and only if both*

(1) $\kappa_M(f_1, f_2) \notin \Gamma_1$ implies that f_2 is a winning strategy against player 1, and

---

[4]Winning strategies have also been used to characterise the existence of Nash equilibria in other two-player games with binary outcomes—see, *e.g.*, [22, 27].

(2) $\kappa_M(f_1, f_2) \notin \Gamma_2$ *implies that f_1 is a winning strategy against player 2.*

Proof. For the "if" direction assume for contraposition that $f = (f_1, f_2)$ is not a run-invariant equilibrium. Then, either $\kappa_M(f_1, f_2) \notin \Gamma_1$ and $\kappa_M(g_1, f_2) \in \Gamma_1$ for some run-invariant strategy g_1 for player 1, or $\kappa_M(f_1, f_2) \notin \Gamma_2$ and $\kappa_M(f_1, g_2) \in \Gamma_2$ for some run-invariant strategy g_2 for player 2. If the former, f_2 is not winning against player 1, refuting 1. If the latter, f_1 is not winning against player 2, which refutes 2.

The opposite direction is also by contraposition. Assume that either 1 or 2 is not satisfied. Without loss of generality we may assume the former. Then, $\kappa_M(f_1, f_2) \notin \Gamma_1$ and f_2 is not a winning strategy against player 1. Accordingly, there is some run-invariant strategy g_1 for player 1 such that $\kappa_M(g_1, f_2) \in \Gamma_1$ and it follows that $f = (f_1, f_2)$ is not a run-invariant equilibrium. □

In order to have a formally convenient characterisation of the goal sets Γ_1 and Γ_2 to be run-based in two bisimilar CGS-games and to define the profile $f^K = (f_1^K, f_2^K)$, we furthermore introduce the following notations and auxiliary concepts. For a concurrent game structure M and finite computations $\kappa, \kappa' \in comps_M$, we write $\kappa \equiv_M \kappa'$ if $\pi_M(\kappa) = \pi_M(\kappa)$. Furthermore, we say that finite computations κ and κ' are *finitely congruent in M and M'*, in symbols $\kappa \equiv_{K_{M,M'}} \kappa'$, whenever there are (not necessarily distinct) *intermediate computations* $\kappa_0, \ldots, \kappa_m$ such that

(1) $\kappa = \kappa_0$,
(2) $\kappa' = \kappa_m$, and
(3) $\kappa_j \equiv_M \kappa_{j+1}$ or $\kappa_j \equiv_{M'} \kappa_{j+1}$, for every $0 \le j < m$.

As $\equiv_M$ and $\equiv_{M'}$ are equivalence relations, we may assume that here $\equiv_M$ and $\equiv_{M'}$ alternate and $\kappa_j \equiv_M \kappa_{j+1}$ if j is even, and $\kappa_j \equiv_{M'} \kappa_{j+1}$ if j is odd. We will generally omit the subscript in $K_{M,M'}$ when M and M' are clear from the context. For an example, see again Figure 3. Consider the (one-step) computations $\kappa_1 = (a, a)$, $\kappa_2 = (a, b)$, $\kappa_3 = (b, a)$, and $\kappa_4 = (b, b)$. Then, $\kappa_1 \equiv_{M_2, M_3} \kappa_3$, because $\kappa_1 \equiv_{M_3} \kappa_2$ and $\kappa_2 \equiv_{M_2} \kappa_3$. On the other hand, some reflection reveals that $\kappa_1 \not\equiv_{K_{M_1, M_2}} \kappa_4$. It is worth noting that finite congruence of two computations implies statewise bisimilarity of the runs induced, that is, $\kappa \equiv_{K_{M,M'}} \kappa'$ implies $\pi_M(\kappa) \stackrel{.}{\sim} \pi_M(\kappa')$.

Lemma 5.3. *Let M and M' be two bisimilar concurrent game structures and $\kappa = d_0, \ldots, d_k$ and $\kappa' = d'_0, \ldots, d'_k$. Then, $\kappa \equiv_{K_{M,M'}} \kappa'$ implies $\pi_M(\kappa) \stackrel{.}{\sim} \pi_M(\kappa')$.*

Proof. Assume $\kappa \equiv_{K_{M,M'}} \kappa'$. Then there are $\kappa_0, \ldots, \kappa_m$ such that $\kappa = \kappa_0$, $\kappa_m = \kappa'$, and, for all $0 \le \ell < m$, $\pi_M(\kappa_\ell) = \pi_M(\kappa_{\ell+1})$ if ℓ is even and $\pi_{M'}(\kappa_\ell) = \pi_{M'}(\kappa_{\ell+1})$ if ℓ is odd. We assume that m is even; the case where m is odd follows by the same argument *mutatis mutandis*. By virtue of Lemma 2.3-1, we have $\pi_M(\kappa_\ell) \stackrel{.}{\sim} \pi_{M'}(\kappa_\ell)$ for every $0 \le \ell < m$. Hence,

$$\pi_M(\kappa_0) = \pi_M(\kappa_1) \stackrel{.}{\sim} \pi_{M'}(\kappa_1) = \cdots = \pi_{M'}(\kappa_{m-1}) \stackrel{.}{\sim} \pi_M(\kappa_{m-1}) = \pi_M(\kappa_m).$$

As obviously $\pi_M(\kappa_\ell) = \pi_M(\kappa_{\ell+1})$ and $\pi_{M'}(\kappa_\ell) = \pi_{M'}(\kappa_{\ell+1})$ imply, respectively, $\pi_M(\kappa_\ell) \stackrel{.}{\sim} \pi_M(\kappa_{\ell+1})$ and $\pi_{M'}(\kappa_\ell) \stackrel{.}{\sim} \pi_{M'}(\kappa_{\ell+1})$, also

$$\pi_M(\kappa_0) \stackrel{.}{\sim} \pi_M(\kappa_1) \stackrel{.}{\sim} \pi_{M'}(\kappa_1) \stackrel{.}{\sim} \cdots \stackrel{.}{\sim} \pi_{M'}(\kappa_{m-1}) \stackrel{.}{\sim} \pi_M(\kappa_{m-1}) \stackrel{.}{\sim} \pi_M(\kappa_m).$$

By transitivity of $\stackrel{.}{\sim}$ we may conclude that $\pi_M(\kappa) \stackrel{.}{\sim} \pi_M(\kappa)$. □

For bisimilar concurrent game structures M and M', we say that a computation-based strategy f is $K_{M,M'}$-*invariant* if $\kappa \equiv_{K_{M,M'}} \kappa'$ implies $f(\kappa) = f(\kappa')$, for all finite computations $\kappa, \kappa' \in comps_M$. We find that $K_{M,M'}$-invariance exactly captures the concept of a strategy that is run-invariant in two bisimilar concurrent game structures.

Lemma 5.4. *Let M and M' be bisimilar concurrent game structures and f_i a computation-based strategy for player i. Then, f_i is $K_{M,M'}$-invariant if and only if f_i is run-invariant in both M and M'.*

Proof. For the "only if"-direction, assume that f_i is $K_{M,M'}$-invariant and consider arbitrary $\kappa, \kappa' \in comps_M$ such that $\pi_M(\kappa) = \pi_M(\kappa')$, that is, $\kappa \equiv_M \kappa'$. By $K_{M,M'}$-invariance of f_i then immediately $f_i(\kappa) = f_i(\kappa')$. Accordingly, f_i is run-invariant in M. The argument for f_i being run-invariant in M' is analogous.

For the "if"-direction, assume that f_i is run-invariant in both M and M', and consider arbitrary $\kappa, \kappa' \in comps$ such that $\kappa \equiv_K \kappa'$. Then, we may assume that there are $\kappa_0, \ldots, \kappa_m$ such that

$$\kappa = \kappa_0 \equiv_M \kappa_1 \equiv_{M'} \kappa_2 \equiv_M \cdots \equiv_{M'} \kappa_m = \kappa'.$$

Having assumed that f_i is run-invariant in both M and M', then also

$$f_i(\kappa) = f_i(\kappa_0) = f_i(\kappa_1) = f_i(\kappa_2) = \cdots = f_i(\kappa_m) = f_i(\kappa'),$$

from which follows that f_i is $K_{M,M'}$-invariant. $\qquad\square$

As we have argued in Section 1, for the question whether the Nash equilibria across two bisimilar CGS-games are preserved to make sense, the players' preferences in the two games have to be congruent, that is, they have to be the same and of the same type in both games. In this section we deal with run-based preferences. We have already seen in Section 4, that identity of a player's computation-based preferences in two CGSs does not guarantee their being congruent as run-based preferences. By imposing an additional closedness restriction, however, we find that a computation-based goal set can be guaranteed to be run-based in two CGS-games based on bisimilar concurrent game structures M and M'. Accordingly, call a goal set Γ_i $K_{M,M'}$-*closed* if for all computations $\kappa = d_0, d_1, d_2, \ldots$ and $\kappa' = d'_0, d'_1, d'_2, \ldots$, we have that $\kappa \in \Gamma_i$ implies $\kappa' \in \Gamma_i$ whenever $d_0, \ldots, d_k \equiv_{K_{M,M'}} d'_0, \ldots, d'_k$ for all $k \geq 0$.

Lemma 5.5. *Let $G = (M, \Gamma_1, \ldots, \Gamma_n)$ and $G' = (M', \Gamma_1, \ldots, \Gamma_n)$ be CGS-games on bisimilar concurrent game structures M and M', and Γ_i is $K_{M,M'}$-closed for some player i. Then, Γ_i is run-based in both M and M'.*

Proof. Assume that Γ_i is $K_{M,M'}$-invariant and consider arbitrary infinite computations $\kappa = d_0, d_1, d_2, \ldots$ and $\kappa' = d'_0, d'_1, d'_2, \ldots$ such that $\rho_M(\kappa) = \rho_M(\kappa')$. Also assume that $\kappa \in \Gamma_i$. Let $\rho_M(\kappa) = s_0, s_1, s_2, \ldots$ and $\rho_M(\kappa') = t_0, t_1, t_2, \ldots$. Then, for every $k \geq 0$, we also have that

$$\pi_M(d_0, \ldots, d_k) = s_0, \ldots, s_{k+1} = t_0, \ldots, t_{k+1} = \pi_M(d'_0, \ldots, d'_k).$$

Having assumed Γ_i to be $K_{M,M'}$-invariant, it follows that $\kappa' \in \Gamma_i$, as desired. $\qquad\square$
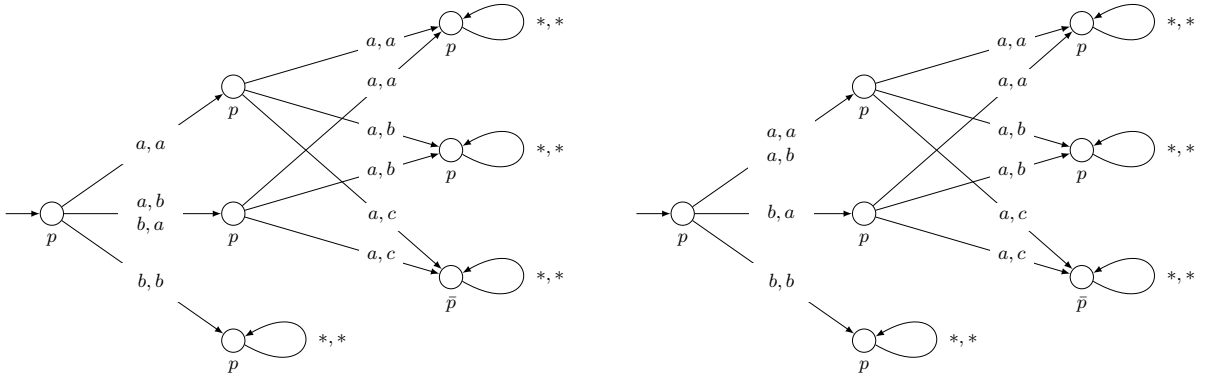
Figure 6: Two games G_6 (left) and G_7 (right) based on M_6 and M_7, respectively.

For the remainder, let $G = (M, \Gamma_1, \Gamma_2)$ and $G' = (M', \Gamma_1, \Gamma_2)$ be two two-player CGS-games based on bisimilar concurrent game structures M and M' such that both Γ_1 and Γ_2 are $K_{M,M'}$-closed (and thus, in particular, run-based). We prove that if there is a run-invariant equilibrium in M, then there is also a K-invariant profile that is a run-invariant equilibrium in M. We construct for a given strategy profile $f = (f_1, f_2)$ that is run-invariant in M a $K_{M,M'}$-invariant profile $f^K = (f_1^K, f_2^K)$ such that

  (i) $\kappa_M(f_1, f_2) = \kappa_M(f_1^K, f_2^K)$,
 (ii) if f_1 is a winning strategy against player 2, then so is f_1^K,
(iii) if f_2 is a winning strategy against player 1, then so is f_2^K.

On basis of Lemma 5.2 we may then conclude that f^K corresponds to a run-invariant equilibrium in M. Having defined f^K formally as a computation-based profile, by Theorem 4.1 it follows that f^K is also a computation-based equilibrium in G'. Finally, because f^K is $K_{M,M'}$-invariant, we know that it furthermore corresponds to a *run-invariant* equilibrium in both M and M'.

For an example consider the games G_6 and G_7 depicted in Figure 6. The underlying concurrent game structures M_6 and M_7 only differ with respect to direction (a, b) at the initial state and their bisimilarity is easily established. Assume that the goal of player 1 is to see p false at some point in the future, that is,

$$\Gamma_1 = \{d_0, d_1, d_2, \cdots \in comps^\omega : d_0 \in \{(a, a), (a, b), (b, a)\} \text{ and } d_1 = (a, c)\},$$

and that player 2 tries to prevent this, that is, $\Gamma_2 = comps^\omega \setminus \Gamma_1$. Observe that defined thus, the players' preferences are run-based in both G_6 and G_7. Concentrating on G_6 first, define the run-invariant strategy profile $f = (f_1, f_2)$, such that,

$$\begin{array}{llll} f_1(\epsilon) = b & f_1(a, a) = a & f_1(a, b) = a & f_1(b, a) = a \\ f_2(\epsilon) = b & f_2(a, a) = a & f_2(a, b) = b & f_2(b, a) = b \end{array}$$

We find that $f = (f_1, f_2)$ is a run-invariant equilibrium in G_6. Observe, however, that $f = (f_1, f_2)$ is *not* run-invariant in G_7, as $f_2(a, a) \neq f_2(a, b)$ even though $\pi_{M_7}(a, a) = \pi_{M_7}(a, b)$. Accordingly, $f = (f_1, f_2)$ fails as a K_{M_6, M_7}-invariant equilibrium. Let g_2 be defined such that,

$$\begin{array}{llll} g_2(\epsilon) = b & g_2(a, a) = b & g_2(a, b) = b & g_2(b, a) = b \end{array}$$

Then, (f_1, g_2) is readily seen to be a K_{M_6,M_7}-invariant equilibrium in G_6. We will find that under the conditions specified, this is no coincidence and that K-invariant equilibria can be constructed from run-invariant equilibria in a systematic fashion.

We first define the strategy profile $f^K = (f_1^K, f_2^K)$. The underlying idea is to carefully choose for each finite computation $d_0, \ldots, d_k$ computations $\kappa_{f_1}^{d_0,\ldots,d_k}$ and $\kappa_{f_2}^{d_0,\ldots,d_k}$ from the equivalence class of $d_0, \ldots, d_k$ under $\equiv_{K_{M,M'}}$, and set $f_1^K(d_0, \ldots, d_k) = f_1(\kappa_{f_1}^{d_0,\ldots,d_k})$ and $f_2^K(d_0, \ldots, d_k) = f_2(\kappa_{f_2}^{d_0,\ldots,d_k})$, respectively. This guarantees that $f^K = (f_1^K, f_2^K)$ is $K_{M,M'}$-invariant. Here we give priority to prefixes of $\kappa(f_1, f_2)$, that is if $\kappa(f_1, f_2) = d'_0, d'_1, d'_2, \ldots$ and $d'_0, \ldots, d'_k \equiv_{K_{M,M'}} d_0, \ldots, d_k$, then $\kappa_{f_1}^{d_0,\ldots,d_k} = \kappa_{f_2}^{d_0,\ldots,d_k} = d'_0 \ldots, d'_k$. In a similar way, we also choose $\kappa_{f_1}^{d_0,\ldots,d_k}$ and $\kappa_{f_2}^{d_0,\ldots,d_k}$ so as to preserve the two players' punishment strategies. This guarantees that $f^K = (f_1^K, f_2^K)$ is an equilibrium, as it inherits this property from $f = (f_1, f_2)$.[5] To do so we assume a well-ordering of the action sets Ac_1 and Ac_2 of players 1 and 2, respectively. Then, for all finite computations $\kappa = d_0, \ldots, d_k$ in $comps_M$, we define inductively and simultaneously computations $\kappa_{f_1}^{d_0,\ldots,d_k}$ and $\kappa_{f_2}^{d_0,\ldots,d_k}$ as follows. For $\kappa = \epsilon$ we have $\kappa_{f_1}^\epsilon = \kappa_{f_2}^\epsilon = \epsilon$, and, for $\kappa = d_0, \ldots, d_{k+1}$,

$$\kappa_{f_1}^{d_0,\ldots,d_k,d_{k+1}} = d'_0, \ldots, d'_k, (x_1, x_2)$$
$$\kappa_{f_2}^{d_0,\ldots,d_k,d_{k+1}} = d''_0, \ldots, d''_k, (y_1, y_2),$$

where $\kappa_{f_1}^{d_0,\ldots,d_k} = d'_0, \ldots, d'_k$, $\kappa_{f_2}^{d_0,\ldots,d_k} = d''_0, \ldots, d''_k$, and

(i.1) $x_1 = f_1(d'_0, \ldots, d'_k)$ and $x_2 = f_2(d'_0, \ldots, d'_k)$, if
$$d'_0, \ldots, d'_k, (f_1(d'_0, \ldots, d'_k), f_2(d'_0, \ldots, d'_k)) \equiv_K d_0, \ldots, d_k, d_{k+1},$$

(i.2) $y_1 = f_1(d''_0, \ldots, d''_k)$ and $y_2 = f_2(d''_0, \ldots, d''_k)$, if
$$d''_0, \ldots, d''_k, (f_1(d''_0, \ldots, d''_k), f_2(d''_0, \ldots, d''_k)) \equiv_K d_0, \ldots, d_k, d_{k+1},$$

(ii.1) $x_1 = f_1(d'_0, \ldots, d'_k)$ and x_2 is the least action available to player 2 such that
$$d'_0, \ldots, d'_k, (x_1, x_2) \equiv_K d_0, \ldots, d_k, d_{k+1},$$
if such an action x_2 exists and (i.1) does not apply,

(ii.2) $y_2 = f_2(d''_0, \ldots, d''_k)$ and y_1 is the least action available to player 1 such that
$$d''_0, \ldots, d''_k, (y_1, y_2) \equiv_K d_0, \ldots, d_k, d_{k+1},$$
if such an action y_1 exists and (i.2) does not apply,

(iii.1) x_1 and x_2 are the least actions available to players 1 and 2, respectively, such that $d'_0, \ldots, d'_k, (x_1, x_2) \equiv_K d_0, \ldots, d_k, d_{k+1}$, if neither ($i$.1) nor ($ii$.1) apply,

(iii.2) y_1 and y_2 are the least actions available to players 1 and 2, respectively, such that $d''_0, \ldots, d''_k, (x_1, x_2) \equiv_K d_0, \ldots, d_k, d_{k+1}$, if neither ($i$.2) nor ($ii$.2) apply,

Observe that the actions x_1 and x_2 in the definition above always exist. The reason for this is that, if $d_0, \ldots, d_k \equiv_{K_{M,M'}} d'_0, \ldots, d'_k$, by Lemma 5.3 also $\pi_M(d_0, \ldots, d_k) \dot\sim \pi_M(d'_0, \ldots, d'_k)$. Let $\pi(d_0, \ldots, d_k) = s_0, \ldots, s_k$, $\pi(d'_0, \ldots, d'_k) = s'_0, \ldots, s'_k$, and $d'_{k+1} = (a_1, a_2)$. Then,

---

[5]The precise definition is rather involved, and the reader may want to skip to page 33, where the main theorem of the section is stated and proven.

obviously, a_1 and a_2 are available to players 1 and 2 respectively at state s'_k. As $s_k \sim s'_k$, that should also be the case at s_k. A similar argument applies to the case where $d_0, \ldots, d_k \equiv_{K_{M,M'}} d''_0, \ldots, d''_k$.

We now define strategies f_1^K and f_2^K such that, for all finite computations $\kappa = d_0, \ldots, d_k$ in $comps_M$,

$$f_1^K(d_0, \ldots, d_k) = f_1(\kappa_{f_1}^{d_0, \ldots, d_k}) \qquad \text{and} \qquad f_2^K(d_0, \ldots, d_k) = f_2(\kappa_{f_2}^{d_0, \ldots, d_k}).$$

To illustrate how $f^K = (f_1^K, f_2^K)$ is constructed from $f = (f_1, f_2)$, recall the games M_6 and G_7 in Figure 6, and assume that actions for both players are ordered alphabetically. For the empty computation ϵ, we have

$$f_1^K(\epsilon) = f_1(\kappa_{f_1}^\epsilon) = f_1(\epsilon) = b \qquad \text{and} \qquad f_2^K(\epsilon) = f_2(\kappa_{f_2}^\epsilon) = f_2(\epsilon) = b$$

Now consider the finite computations of length one. For computation (b, b), we find that $(b, b) \equiv_{K_{M_6,M_7}} (f_1(\kappa_{f_1}^\epsilon), f_2(\kappa_{f_1}^\epsilon))$. Now case $(i.1)$ applies and we obtain $\kappa_{f_1}^{(b,b)} = (b, b)$. Similarly, $\kappa_{f_2}^{(b,b)} = (b, b)$. For the other one-step computations $(x_1, x_2) \in \{(a, a), (a, b), (b, a)\}$, we find that $(x_1, x_2) \not\equiv_{K_{M_6,M_7}} (f_1(\epsilon), f_2(\epsilon))$. First consider (a, a), and to determine $\kappa_{f_1}^{(a,a)}$, first observe that $\kappa_{f_1}^\epsilon = d'_0, \ldots, d'_k = \epsilon$, $d_0, \ldots, d_k = \epsilon$, and $d_{k+1} = (a, a)$. Now, for $x_1 = f_1(\epsilon) = b$ we have that

$$(x_1, a) = (b, a) \equiv_{M_6} (a, b) \equiv_{M_7} (a, a).$$

Accordingly, $(x_1, a) \equiv_{M_6, M_7} (a, a) = d_{k+1}$, and case $(ii.1)$ applies. With a being moreover player 2's alphabetically least action, we may therefore conclude that $\kappa_{f_1}^{(a,a)} = (b, a)$. In a similar way we obtain $\kappa_{f_1}^{(a,b)} = \kappa_{f_1}^{(b,a)} = (b, a)$ as well as $\kappa_{f_2}^{(a,a)} = \kappa_{f_2}^{(a,b)} = \kappa_{f_2}^{(b,a)} = (a, b)$. Hence,

$$f_1^K(a, a) = f_1(\kappa_{f_1}^{(a,a)}) = f_1(b, a) = a \qquad f_2^K(a, a) = f_2(\kappa_{f_2}^{(a,a)}) = f_2(a, b) = b$$

$$f_1^K(a, b) = f_1(\kappa_{f_1}^{(a,b)}) = f_1(b, a) = a \qquad f_2^K(a, b) = f_2(\kappa_{f_2}^{(a,b)}) = f_2(a, b) = b$$

$$f_1^K(b, a) = f_1(\kappa_{f_1}^{(b,a)}) = f_1(b, a) = a \qquad f_2^K(b, a) = f_2(\kappa_{f_2}^{(b,a)}) = f_2(a, b) = b$$

$$f_1^K(b, b) = f_1(\kappa_{f_1}^{(b,b)}) = f_1(b, b) = b \qquad f_2^K(b, b) = f_2(\kappa_{f_2}^{(b,b)}) = f_2(b, b) = b$$

We thus find that $f^K = (f_1^K, f_2^K)$ coincides with the K_{M_6,M_7}-invariant equilibrium (f_1, g_2) that we identified above.

The definition of strategies f_1^K and f_2^K ensures that $f^K = (f_1^K, f_2^K)$ is $K_{M,M'}$-invariant.

Lemma 5.6. *Let $\kappa = d_0, \ldots, d_k$ be s finite computation in $comps_M$. Then, for $i = 1, 2$,*

$$d_0, \ldots, d_k \equiv_K \kappa_{f_i}^{d_0, \ldots, d_k}.$$

Accordingly, $f^K = (f_1^K, f_2^K)$ is $K_{M,M'}$-invariant.

Proof. Strategies f_1^K and f_2^K have been defined so as to be K-invariant. The claim then follows by induction on the length of κ. Let $i = 1$; the case for $i = 2$ is analogous. If the length of κ is 0, we have $\kappa = \kappa_{f_1}^\epsilon = \epsilon$, and it immediately follows that $\kappa \equiv_K \kappa_{f_1}^\epsilon$. For the induction step, let $\kappa = d_0, \ldots, d_k, d_{k+1}$ and assume that $d_0, \ldots, d_k \equiv_K \kappa_{f_1}^{d_0, \ldots, d_k}$. Let $\kappa_{f_1}^{d_0, \ldots, d_{k+1}}$ be denoted by $d'_0, \ldots, d'_k, d'_{k+1}$ where $d'_{k+1} = (x_1, x_2)$. There are three possibilities.

First, assume that $d'_0, \ldots, d'_k, (f_1(d'_0, \ldots, d'_k), f_2(d'_0, \ldots, d'_k)) \equiv_K d_0, \ldots, d_k, d_{k+1}$. Then, clause $(i.1)$ applies and we have $x_1 = f_1(d'_0, \ldots, d'_k)$ and $x_2 = f_2(d'_0, \ldots, d'_k)$. It then follows

that $d'_0, \ldots, d'_k, d'_{k+1} \equiv_K d_0, \ldots, d_k, d_{k+1}$. Second, assume that $d'_0, \ldots, d'_k, (f_1(d'_0, \ldots, d'_k), y_2) \equiv_K d_0, \ldots, d_k, d_{k+1}$ for some action $y_2 \in \mathrm{Ac}_2$, but $d'_0, \ldots, d'_k, (f_1(d'_0, \ldots, d'_k), f_2(d'_0, \ldots, d'_k)) \not\equiv_K d_0, \ldots, d_k, d_{k+1}$. Then clause $(ii.1)$ applies and we have that $x_1 = f_1(d'_0, \ldots, d'_k)$ and x_2 is the least action available to player 2 such that $d'_0, \ldots, d'_k, (f_1(d'_0, \ldots, d'_k), y_2) \equiv_K d_0, \ldots, d_k, d_{k+1}$. Again, it immediately follows that $d'_0, \ldots, d'_k, d'_{k+1} \equiv_K d_0, \ldots, d_k, d_{k+1}$.

Finally, assume that neither of the above, then clause $(iii.1)$ applies and we have that x_1 and x_2 are the least actions available to players 1 and 2 such that $d'_0, \ldots, d'_k, (x_1, x_2) \equiv_K d_0, \ldots, d_{k+1}$. As such actions exist, we conclude that $d'_0, \ldots, d'_k, d'_{k+1} \equiv_K d_0, \ldots, d_k, d_{k+1}$. $\square$

We are now in a position to prove the following crucial lemmas.[6]

Lemma 5.7. *Let $f = (f_1, f_2)$ be a run-invariant strategy profile for concurrent game structure M and f^K defined as above. Then, $\kappa_M(f_1, f_2) = \kappa_M(f_1^K, f_2^K)$.*

Proof. Let us use the following notations: $\kappa_M(f_1, f_2) = d_0, d_1, \ldots$; $\kappa(f_1^K, f_2^K) = d_0^K, d_1^K, \ldots$; $\kappa_{f_1}^{d_0, \ldots, d_k} = d'_0, d'_1, \ldots$; $\kappa_{f_2}^{d_0, \ldots, d_k} = d''_0, d''_1, \ldots$; where, for every $k \geq 0$,

$$d_k = (a_1^k, a_2^k), \qquad d_k^K = (b_1^k, b_2^k), \qquad d'_k = (x_1^k, x_2^k), \qquad d''_k = (y_1^k, y_2^k).$$

It then suffices to prove by induction on k that for every $k \geq 0$, we have

$$d_k = d_k^K = d'_k = d''_k.$$

For $k = 0$, let $d_0 = (a_1^0, a_2^0)$ and $d_0^K = (b_1, b_2)$. Then,

$$a_1^0 = f_1(\epsilon) = f_1(\kappa_{f_1}^\epsilon) = f_1^K(\epsilon) = b_1^0.$$

In a similar way we find that $a_2^0 = b_2^0$, and hence $d_0 = d_0^K$. Because both $f_1(\epsilon) = a_1^0$ and $f_2(\epsilon) = a_2^0$, we also have $d'_0 = d_0$, and, a fortiori, $d'_0 \equiv_K d_0$. Hence, clause $(i.1)$ applies and therefore $d'_0 = (f_1(\epsilon), f_2(\epsilon)) = (a_1^0, a_2^0) = d_0$. In a similar way we can establish that $d''_0 = d_0$.

For the induction step, we may assume that

$$d_0, \ldots, d_k = d_0^K, \ldots, d_k^K = d'_0, \ldots, d'_k = d''_0, \ldots, d''_k. \qquad (i.h.)$$

Then,

$$a_1^{k+1} = f_1(d_0, \ldots, d_k) =_{i.h.} f_1(d'_0, \ldots, d'_k) = f_1^K(d_0, \ldots, d_k) =_{i.h.} f_1^K(d'_0, \ldots, d'_k) = b_1^{k+1}.$$

Observe that the third equality holds because $d'_0, \ldots, d'_k = \kappa_{f_1}^{d_0, \ldots, d_k}$. For player 2, the following equalities hold:

$$a_2^{k+1} = f_2(d_0, \ldots, d_k) =_{i.h.} f_2(d''_0, \ldots, d''_k) = f_2^K(d_0, \ldots, d_k) =_{i.h.} f_2^K(d'_0, \ldots, d'_k) = b_2^{k+1}.$$

Now the third equality holds since $d''_0, \ldots, d''_k = \kappa_{f_2}^{d_0, \ldots, d_k}$.

Finally, by the induction hypothesis, $d'_0, \ldots, d'_k = d_0, \ldots, d_k$. From the above, we already had $f_1(d'_0, \ldots, d'_k) = a_1^{k+1}$ and $f_2(d''_0, \ldots, d''_k) = a_2^{k+1}$ Hence, $(f_1(d'_0, \ldots, d'_k), f_2(d''_0, \ldots, d''_k)) = d^{k+1}$. It follows that, $d'_0, \ldots, d'_k, (f_1(d'_0, \ldots, d'_k), f_2(d''_0, \ldots, d''_k)) = d_0, \ldots, d_k, d_{k+1}$. In particular, $d'_0, \ldots, d'_k, (f_1(d'_0, \ldots, d'_k), f_2(d''_0, \ldots, d''_k)) \equiv_K d_0, \ldots, d_k, d_{k+1}$. Therefore, clause $(i.1)$ is applicable, and both

$$x_1^{k+1} = f_1(d'_0, \ldots, d'_k) = a_1^{k+1} \qquad \text{and} \qquad x_2^{k+1} = f_2(d''_0, \ldots, d''_k) = a_2^{k+1},$$

signifying that $d_{k+1} = d'_{k+1}$. By an analogous reasoning we show that $d_{k+1} = d''_{k+1}$. $\square$

---

[6]The proofs of these two lemmas extend over a couple of pages and the reader may skip to page 33, where the running text continues.

The next lemma establishes that f_1^K and f_2^K are winning run-invariant strategies against player 2 and player 1, respectively, if the goal sets Γ_1 and Γ_2 are to be run-based. Notice that this result requires Γ_1 and Γ_2 to be $K_{M,M'}$-closed.

Lemma 5.8. *Let $f = (f_1, f_2)$ be a run-invariant strategy profile in game $G = (M, \Gamma_1, \Gamma_2)$ with Γ_1 and Γ_2 run-based and K-closed, and f^K defined as above. Then,*

(1) *if f_1 is a winning (run-invariant) strategy against player 2, then so is f_1^K,*

(2) *if f_2 is a winning (run-invariant) strategy against player 1, then so is f_2^K.*

Proof. For part 1, assume for contraposition that f_1^K is not winning against player 2. Then, there is a strategy g_2 for player 2 such that $\kappa(f_1^K, g_2) \in \Gamma_2$. We define a K-invariant strategy g_2^K for player 2 such that $d_0, \ldots, d_k \equiv_K e_0, \ldots, e_k$ for every $k \geq 0$, where

$$\kappa(f_1^K, g_2) = d_0, d_1, d_2, \ldots \qquad \text{and} \qquad d_k = (a_1, a_2),$$
$$\kappa(f_1, g_2^K) = e_0, e_1, e_2, \ldots. \qquad \text{and} \qquad e_k = (b_1, b_2).$$

By Γ_2 being K-closed it then follows that also $\kappa(f_1, g_2^K) \in \Gamma_2$, which contradicts our initial assumption that f_1 is winning strategy against player 2. For each $k \geq 0$, let furthermore

$$\kappa^{d_0, \ldots, d_k} = d_0', \ldots, d_k' \qquad \text{and} \qquad d_k' = (c_1, c_2).$$

In order to define the strategy g_2^K, we may assume the existence of some K-invariant strategy h_2 for player 2. For the empty computation ϵ we have $g_2^K(\epsilon) = x_2^0$ where

$(i.3)$ $x_2^0 = f_2(\epsilon)$, if $(f_1(\epsilon), f_2(\epsilon)) \equiv_K d_0$,

$(ii.3)$ x_2^0 is the least action y_2^0 available to player 2 such that $(f_1(\epsilon), y_2^{k+1}) \equiv_K d_0$, if such an action y_2^{k+1} exists and case (i) does not apply,

$(iii.3)$ $x_2^{k+1} = h(\epsilon)$ in all other cases.

For every finite computation $d_0, \ldots, d_k$, we have that $g_2^K(d_0, \ldots, d_k) = x_2^{k+1}$, where

$(i.4)$ $x_2^{k+1} = f_2(d_0, \ldots, d_k)$, if

$$d_0, \ldots, d_k, (f_1(d_0, \ldots, d_k), f_2(d_0, \ldots, d_k)) \equiv_K d_0, \ldots, d_k, d_{k+1},$$

$(ii.4)$ x_2^{k+1} is the least action y_2^{k+1} available to player 2 such that

$$d_0, \ldots, d_k, (f_1(d_0, \ldots, d_k), y_2^{k+1}) \equiv_K d_0, \ldots, d_k, d_{k+1},$$

if such an action y_2^{k+1} exists and case (i) does not apply,

$(iii.4)$ $x_2^{k+1} = h(d_0, \ldots, d_k)$ in all other cases.

Observe that g_2^K is K-invariant by construction.

We now prove by induction on k that $e_0, \ldots, e_k = d_0', \ldots, d_k'$ for every $k \geq 0$. If $k = 0$, recall that $e_0 = (b_1^0, b_2^0)$ and $d_0^0 = (c_1^0, c_2^0)$. Observe that $f_1^K(\epsilon) = f_1(\kappa^\epsilon) = f_1(\epsilon)$. Thus we have $d_0 = (f_1(\epsilon), g_2(\epsilon)) = (f_1(\epsilon), g_2(\epsilon))$, and in particular $d_0 \equiv_K (f_1(\epsilon), g_2(\epsilon))$. Hence, there is some y_2^0 such that $(f_1(\epsilon), y_2^0) \equiv_K d_0$.

First consider the case where $(f_1(\epsilon), f_2(\epsilon)) \equiv_K d_0$. Now clause $(i.1)$ applies and $c_1^0 = f_1(\epsilon)$ and $c_2^0 = f_2(\epsilon)$. Accordingly, clause $(i.3)$ is applicable as well, and we obtain both

$$b_1^0 = f_1(\epsilon) = c_1^0 \qquad \text{and} \qquad b_2^0 = g_2^k(\epsilon) = f_2(\epsilon) = c_2^0.$$

Otherwise, there is some least x_2^0 such that $(f_1(\epsilon), x_2^0) \equiv_K d_0$. Thus, due to clause $(ii.1)$ we have $c_1^0 = f_1(\epsilon)$ and $c_2^0 = x_2^0$. Clause $(ii.3)$ now also applies and we obtain:

$$b_1^0 = f_1(\epsilon) = c_1^0 \qquad \text{and} \qquad b_2^0 = g_2^K(\epsilon) = x_2^0 = c_2^0.$$

The induction step runs along similar lines. We may assume that

$$e_0, \ldots, e_k = d_0^0, \ldots, d_k. \qquad (i.h.)$$

Observe that

$$f_1^K(d_0, \ldots, d_k) = f_1(\kappa^{d_0, \ldots, d_k}) = f_1(d_0', \ldots, d_k').$$

Thus,

$$d_{k+1} = (f_1^K(d_0, \ldots, d_k), g_2(d_0, \ldots, d_k)) = (f_1(d_0', \ldots, d_k'), g_2(d_0, \ldots, d_k)).$$

By Lemma 5.6, moreover, $d_0, \ldots, d_k \equiv_K d_0', \ldots, d_k'$, and it follows that

$$d_0, \ldots, d_k, d_{k+1} \equiv_K d_0', \ldots, d_k', (f_1(d_0', \ldots, d_k'), g_2(d_0, \ldots, d_k)).$$

Hence, there is some y_2^{k+1} such that

$$d_0, \ldots, d_k, d_{k+1} \equiv_K d_0', \ldots, d_k', (f_1(d_0', \ldots, d_k'), y_2^{k+1}). \qquad (*)$$

First assume that equation $(*)$ holds for $y_2^{k+1} = f_2(d_0', \ldots, d_k')$. Then clause $(i.1)$ applies and for $d_{k+1}' = (c_1^{k+1}, c_2^{k+1})$ we have

$$c_1^{k+1} = f_1(d_0', \ldots, d_k') \qquad \text{and} \qquad c_2^{k+1} = f_2(d_0', \ldots, d_k').$$

Recall that $e^{k+1} = (b_1^{k+1}, b_2^{k+1})$. Now for player 1 we find that

$$b_1^{k+1} = f_1(e_0, \ldots, e_k) =_{i.h.} f_1(d_0', \ldots, d_k') = c_1^{k+1}.$$

For player 2, observe that, in the case we are considering,

$$e_0, \ldots, e_k, (f_1(d_0', \ldots, d_k'), f_2(d_0', \ldots, d_k')) \quad =_{i.h.} \quad d_0', \ldots, d_k', (f_1(d_0', \ldots, d_k'), f_2(d_0', \ldots, d_k'))$$
$$= \quad d_0', \ldots, d_k', d_{k+1}'.$$

Accordingly, clause $(i.4)$ applies, that is, $g_2^K(e_0, \ldots, e_k) = f_2(d_0', \ldots, d_k')$. Hence,

$$b_2^{k+1} = g_2^K(e_0, \ldots, e_k) = f_2(d_0', \ldots, d_k') = c_2^{k+1},$$

and we may conclude that $e_{k+1} = (b_1^{k+1}, b_2^{k+1}) = (c_1^{k+1}, c_2^{k+1}) = d_{k+1}'$.

Finally, assume that equation $(*)$ does not hold for $y_2^{k+1} = f_2(d_0', \ldots, d_k')$. Then, let x_2^{k+1} be the least action for player 2 for which equation $(*)$ does hold with $y^{k+1} = x_2^{k+1}$. As in this case clause $(i.2)$ applies and for $d_{k+1}' = (c_1^{k+1}, c_2^{k+1})$, we have,

$$c_1^{k+1} = f_1(d_0', \ldots, d_k') \qquad \text{and} \qquad c_2^{k+1} = x_2^{k+1}.$$

Recall that $e_{k+1} = (b_1^{k+1}, b_2^{k+1})$. For player 1 we again find that,

$$b_1^{k+1} = f_1(e_0, \ldots, e_k) =_{i.h.} f_1(d_0', \ldots, d_k') = c_1^{k+1}.$$

For player 2, observe that, in the case we are considering,

$$e_0, \ldots, e_k, (f_1(d_0', \ldots, d_k'), x_2^{k+1}) \quad =_{i.h.} \quad d_0', \ldots, d_k', (f_1(d_0', \ldots, d_k'), x_2^{k+1})$$
$$= \quad d_0', \ldots, d_k', d_{k+1}'.$$

Accordingly, clause $(ii.4)$ applies and we have $g_2^K(e_0, \ldots, e_k) = x_2^{k+1}$. It then follows that,

$$b_2^{k+1} = g_2^K(e_0, \ldots, e_k) = f_2(d_0', \ldots, d_k') = c_2^{k+1}.$$

Again we may conclude that $e_{k+1} = (b_1^{k+1}, b_2^{k+1}) = (c_1^{k+1}, c_2^{k+1}) = d_{k+1}'$, as desired.

The proof for part 2 is analogous to that of part 1. $\qquad \square$

The ground has now been cleared for the main result of this paper that the existence of run-invariant equilibria is preserved under bisimulation in two-player games provided that the run-based preferences of the players are $K_{M,M'}$-closed.

Theorem 5.9. *Let $G = (M, \Gamma_1, \Gamma_2)$ and $G' = (M', \Gamma_1, \Gamma_2)$ be two two-player games on bisimilar concurrent game structures such that Γ_1 and Γ_2 are run-based and $K_{M,M'}$-closed. Then, if $f = (f_1, f_2)$ is a run-invariant strategy profile in M, then $f^K = (f_1^K, f_2^K)$ is a run-invariant equilibrium in M'.*

Proof. Assume that $f = (f_1, f_2)$ is a run-invariant strategy profile in M. By Lemma 5.2, then both

(1) $\kappa_M(f_1, f_2) \notin \Gamma_1$ implies that f_2 is a winning strategy against player 1 in M, and

(2) $\kappa_M(f_1, f_2) \notin \Gamma_2$ implies that f_1 is a winning strategy against player 2 in M.

On basis of 1, Lemma 5.7 yields $\kappa_M(f_1, f_2) = \kappa_M(f_1^K, f_2^K)$. Now assume $\kappa_M(f_1^K, f_2^K) \notin \Gamma_1$. Then, also $\kappa_M(f_1, f_2) \notin \Gamma_1$ and we may assume that f_2 is a winning strategy against player 1 in M. In virtue of Lemma 5.8 we may then conclude that f_2^K is a winning strategy against player 1 in M. Assuming that $\kappa_M(f_1^K, f_2^K) \notin \Gamma_2$, we can reason analogously and infer that f_1 is a winning strategy against player 2 in M. Hence,

(1′) $\kappa_M(f_1^K, f_2^K) \notin \Gamma_1$ implies that f_2^K is a winning strategy against player 1 in M, and

(2′) $\kappa_M(f_1^K, f_2^K) \notin \Gamma_2$ implies that f_1^K is a winning strategy against player 2 in M.

Accordingly, f^K is a computation-based equilibrium in M. By Theorem 4.1 we may infer that f^K is also a computation-based equilibrium in M'. Lemma 5.6 then guarantees that f^K is $K_{M,M'}$-invariant, and it follows that f^K is run-invariant in M' as well. By virtue of Theorem 3.5 we may finally conclude that f^K is also a run-invariant equilibrium in M'. □

As an immediate consequence of Theorem 5.9, we have the following result, which is phrased in terms of run-based strategies instead of run-invariant strategies.

Corollary 5.10. *Let $G = (M, \Gamma_1, \Gamma_2)$ and $G' = (M', \Gamma_1, \Gamma_2)$ be two two-player games on bisimilar concurrent game structures M and M' such that Γ_1 and Γ_2 are run-based and $K_{M,M'}$-closed. Let furthermore $\rho \in runs_M^\omega$ be a run in M that is sustained by a run-based equilibrium in M. Then, there is a run $\rho' \in runs_{M'}^\omega$ in M' that is statewise bisimilar to ρ and that is also sustained by a run-based equilibrium in M'.*

Proof. Let run $\rho \in runs_M^\omega$ be sustained by a run-based equilibrium $f = (f_1, f_2)$ in G and let $\check{f} = (\check{f}_1, \check{f}_2)$ be the run-invariant strategy profile corresponding to f. Lemma 3.4 guarantees that $\check{f} = (\check{f}_1, \check{f}_2)$ is a run-invariant equilibrium in G. Now construct profile $\check{f}^K = (\check{f}_1^K, \check{f}_2^K)$, which by virtue of Theorem 5.9 is then $K_{M,M'}$-invariant and is a run-invariant equilibrium in both G and G'. By virtue Lemma 2.2-1, it moreover follows that $\rho_M(g_1^K, g_2^K) \stackrel{.}{\sim} \rho_{M'}(g_1^K, g_2^K)$, that is, $\rho_M(g_1^K, g_2^K)$ and $\rho_{M'}(g_1^K, g_2^K)$ are statewise bisimilar, which concludes the proof. □

A further corollary of Theorem 5.9 is that the *existence* of run-based equilibria is preserved in two-player games with run-based and $K_{M,M'}$-closed preferences.

Trace-based Preferences. We find that, with a couple of slight modifications, essentially the same construction as in the previous section can be leveraged to prove that run-based equilibria are also preserved under bisimulation in two-player games with *trace-based* preferences. It be emphasised that here we do not require the preferences to satisfy any other condition than being trace-based.

Let two CGS-games $G = (M, \Gamma_1, \Gamma_2)$ and $G' = (M', \Gamma_1, \Gamma_2)$ on bisimilar concurrent game structures M and M' and with Γ_1 and Γ_2 trace-based be given. For a run-invariant equilibrium $f = (f_1, f_2)$ in game G, we define the $K_{M,M'}$-invariant strategy-profile $f^K = (f_1^K, f_2^K)$ as in the previous section. We prove that $f^K = (f_1^K, f_2^K)$ is also a run-invariant equilibrium in G'. To this end, we adapt Lemma 5.8 so as to apply to trace-based preferences instead of preferences that are both run-based and $K_{M,M'}$-closed.

Lemma 5.11. *Let $f = (f_1, f_2)$ be a run-invariant strategy profile in game $G = (M, \Gamma_1, \Gamma_2)$ with Γ_1 and Γ_2 trace-based, and f^K defined as above. Then,*

(1) *if f_1 is a winning (run-invariant) strategy against player 2, then so is f_1^K,*

(2) *if f_2 is a winning (run-invariant) strategy against player 1, then so is f_2^K.*

Proof. For part 1—part 2 follows by an analogous argument—assume for contraposition that f_1^K is not a winning strategy against player 2. Then, there is a strategy g_2 for player 2 such that $\kappa(f_1^K, g_2) \in \Gamma_2$. We define a $K_{M,M'}$-invariant strategy g_2^K for player 2 exactly as in the proof of Lemma 5.8. Accordingly, $d_0, \ldots, d_k \equiv_{K_{M,M'}} e_0, \ldots, e_k$ for every $k \geq 0$, where

$$\kappa(f_1^K, g_2) = d_0, d_1, d_2, \ldots \qquad \text{and} \qquad d_k = (a_1, a_2),$$
$$\kappa(f_1, g_2^K) = e_0, e_1, e_2, \ldots . \qquad \text{and} \qquad e_k = (b_1, b_2).$$

Now consider an arbitrary $k \geq 0$. Then, by Lemma 5.3, also $\pi_M(d_0, \ldots, d_k) \overset{\cdot}{\sim} \pi_M(e_0, \ldots, e_k)$. Letting $\pi_M(d_0, \ldots, d_k) = s_0, \ldots, s_k$ and $\pi_M(d'_0, \ldots, d'_k) = s'_0, \ldots, s'_k$, we then also have $s_k \sim s'_k$. It follows that $\rho_M(f_1^K, g_2) \overset{\cdot}{\sim} \rho_M(f_1, g_2^K)$ and hence $\tau_M(f_1^K, g_2) = \tau_M(f_1, g_2^K)$. As a consequence of Γ_2 being trace-based, we obtain $\kappa(f_1, g_2^K) \in \Gamma_2$, which contradicts our initial assumption that f_1 is winning strategy against player 2. $\square$

We are now in a position to prove the counterpart of Theorem 5.9 for trace-based preferences, showing that run-invariant equilibria are preserved under bisimulation if the players' preferences are trace-based.

Theorem 5.12. *Let $G = (M, \Gamma_1, \Gamma_2)$ and $G' = (M', \Gamma_1, \Gamma_2)$ be two two-player games on bisimilar concurrent game structures such that Γ_1 and Γ_2 are trace-based. Then, if $f = (f_1, f_2)$ is run-invariant in M, then $f^K = (f_1^K, f_2^K)$ is a run-invariant equilibrium in M'.*

Proof. The proof is fully analogous to that for Theorem 5.9, invoking Lemma 5.11 instead of Lemma 5.8. $\square$

As an immediate consequence of Theorem 5.12, we find that also the *existence* of run-invariant equilibria is preserved in two-player games with trace-based preferences. Furthermore, also the counterpart of Corollary 5.10 for trace-based preferences can easily be demonstrated.

Boolean Game Structures. We now consider a subclass of concurrent game structures in which Nash equilibrium is invariant under bisimilarity. Specifically, we study games played over the class of concurrent game structures induced by iterated Boolean games [23], a framework that can be used to reason about Nash equilibria in games and multi-agent systems modelled using the Reactive Modules specification language [2].

By a *Boolean game structure* we understand a special type of concurrent game structure $M = (\mathrm{Ag}, \mathrm{AP}, \mathrm{Ac}, \mathrm{St}, s_M^0, \lambda, \delta)$ for which there is a partition $\{\mathrm{AP}_1, \ldots, \mathrm{AP}_n\}$ of AP such

that $\mathrm{Ac}_i(s) \subseteq (2^{\mathrm{AP}_i} \setminus \emptyset)$ for all players i and states s and 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

$$\delta(s, d') = s' \text{ implies } \lambda(s') = a_1 \cup \cdots \cup a_n.$$

Then, informally, in a Boolean game structure, choice profiles correspond to system states, which is not generally the case in concurrent game structures. In other words, in a Boolean game structure M, if a strategy profile induces a run $s_M^0, s_1, s_2, \ldots$, then we know that it has been induced by the computation $s_1, s_2, \ldots$. Even more, we also know that the trace induced by such a computation is precisely $s_M^0, s_1, s_2, \ldots$. This very strong correspondence between computations, runs, and traces is key to the proof that in Boolean game structures all strategies for a player are in fact bisimulation-invariant. This result, in turn, can also be used to show that Nash equilibrium is invariant under bisimilarity, regardless of the model of strategies or goals that one chooses. To see this, the following preliminary results will be useful.

Lemma 5.13. *Let $M = (\mathrm{Ag}, \mathrm{AP}, \mathrm{Ac}, \mathrm{St}, s^0, \lambda, \delta)$ be a Boolean game structure with partition $\{\mathrm{AP}_1, \ldots, \mathrm{AP}_n\}$. Let $\pi = s_0, \ldots, s_k$ and $\pi' = s_0', \ldots, s_k'$ be statewise bisimilar finite histories, that is, $\pi \dot{\sim} \pi'$. Then, $\pi = \pi'$.*

Proof. We may assume that there are finite computations $\kappa = d_0, \ldots, d_{k-1}$ and $\kappa' = d_0', \ldots, d_{k-1}'$ such that $s_0 \xrightarrow{d_0} \cdots \xrightarrow{d_{k-1}} s_k$ and $s_0' \xrightarrow{d_0'} \cdots \xrightarrow{d_{k-1}'} s_k'$. We show by induction that $s_m = s_m'$ for all $0 \leq m \leq k$. For the basis, we have $s_0 = s_M^0 = s_0'$. For the induction step we may assume that $s_m = s_m'$. Moreover, as $s_{m+1} \sim s_{m+1}'$, also $\lambda(s_{m+1}) = \lambda(s_{m+1}')$. Furthermore, $s_m \xrightarrow{d_m} s_{m+1}$ and $s_m' \xrightarrow{d_m'} s_{m+1}'$. As M is a Boolean game structure, it follows that $d_m = (\lambda(s_{m+1}) \cap \mathrm{AP}_1, \ldots, \lambda(s_{m+1}) \cap \mathrm{AP}_n)$ and $d' = (\lambda(s_{m+1}') \cap \mathrm{AP}_1, \ldots, \lambda(s_{m+1}') \cap \mathrm{AP}_n)$ and, hence, $d_m = d_m'$. By determinism of δ, we may conclude that $s_{m+1} = \delta(s_m, d_m) = \delta(s_m', d_m') = s_{m+1}'$. $\square$

The above lemma can be used to show that in fact, for Boolean game structures, all models of strategies collapse to the model of bisimulation-invariant strategies.

Lemma 5.14. *In Boolean game structures, all strategies for every player are bisimulation-invariant.*

Proof. Consider an arbitrary strategy f_i of a player i in a Boolean game structure M along with arbitrary statewise bisimilar histories $\pi, \pi' \in runs_M$, that is, $\pi \dot{\sim} \pi'$. By Lemma 5.13, then $\pi = \pi'$. Hence, trivially, $f_i(\pi) = f_i(\pi')$. $\square$

We can now present the main result of this section.

Theorem 5.15. *In Boolean game structures, (the existence of a) Nash equilibrium is invariant under bisimilarity.*

Proof. Observe that because of Lemma 5.14, in Boolean game structures, a strategy profile $f = (f_1, \ldots, f_n)$ is a Nash equilibrium if and only if f is a Nash equilibrium in bisimulation-invariant strategies. The result then immediately follows from Corollary 4.9. $\square$

		Preferences					
		computation-based		run-based		trace-based	
Strategies	computation-based	$+$	(Th. 4.1)	$+$	(Th. 4.1)	$+$	(Th. 4.1)
	run-based (general)	$-$	(Th. 4.5)	$-$	(Th. 4.5)	$-$	(Th. 4.5)
	run-based (two players)	$-$	(Th. 5.1)	$+^\dagger$	(Th. 5.9)	$+$	(Th. 5.12)
	trace-based	$+$	(Th. 4.3)	$+$	(Th. 4.3)	$+$	(Th. 4.3)
	bisimulation-invariant	$+$	(Th. 4.8)	$+$	(Th. 4.8)	$+$	(Th. 4.8)

†Assuming the players' (run-based) preferences to be $K_{M,M'}$-closed.

Table 1: Summary of main bisimulation-invariance results for multi-player games in deterministic systems as well as the results in this paper they are based on. In this figure, $+$ means that Nash equilibria are preserved in (computation/run/trace)-based strategy profiles with preferences given by sets of computations/runs/traces, while $-$ indicates that they are not for such a pair.
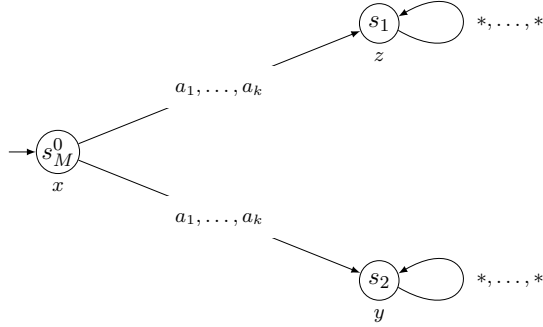
Figure 7: Nondeterministic system with sets of infinite runs given by $s_M^0.(s_1^\omega + s_2^\omega)$ and infinite traces given by $x.(z^\omega + y^\omega)$.

6. NONDETERMINISM

Our results so far, summarised in Table 1, apply to profiles of deterministic strategies and deterministic systems. In this section, we investigate the case of nondeterministic systems. In this more general setting, most of our notations and definitions remain the same, except for two that are particularly relevant: the notions of *outcome* of a game and Nash *equilibrium*.

Note that in a deterministic system, a profile of deterministic strategies induces a *unique* system path (and therefore a unique computation, run, and trace). However, if the system is nondeterministic, a profile of deterministic strategies might, instead, determine a *set* of paths of the system: all those complying with the profile of strategies. For instance, in the system in Figure 7, the deterministic strategy profile where every player i chooses to play a_i at the beginning determines two different runs and traces of the system.

Therefore, formally, a deterministic strategy profile f on a nondeterministic system M may determine a set of computations in $comps_M^\omega$. To simplify notations, we will write $\kappa_M(f) \subseteq comps_M^\omega$ for such a set, which will correspond to the set of computations that could result in M when playing strategy profile f. Likewise, we will write $\rho_M(f) \subseteq runs_M^\omega$ and $\tau_M(f) \subseteq traces_M^\omega$, respectively, for the sets of runs and traces determined by f on M.

These three sets of computations, runs, and traces determined by f, namely $\kappa_M(f)$, $\rho_M(f)$, and $\tau_M(f)$, will correspond to our more general notion of (computation, run, trace) *outcome* of a multi-player game. Clearly, for deterministic systems, these sets of computations, runs, and traces will correspond to the special case where outcomes are singleton sets.[7]

Our more general definition of outcome call for a (just slightly) more general definition of equilibrium. The generalisation is rather simple. With respect to a nondeterministic system M, we will define the preferences Γ_i of a player i as a set of sets of computations of M, rather than just a set of computations, as in the deterministic case. In other words, while in a deterministic system we have $\Gamma_i \subseteq comps_M^\omega$, in a nondeterministic system we have $\Gamma_i \subseteq 2^{comps_M^\omega}$. With this definition in place, we can define a Nash equilibrium in exactly the same way that it is defined for deterministic systems, that is, as a strategy profile $f = (f_1, \ldots, f_n)$ such that for every player i and every strategy g_i available to i,

$$\kappa_M(f_{-i}, g_i) \in \Gamma_i \text{ implies } \kappa_M(f) \in \Gamma_i.$$

As for deterministic systems, the concepts of *Nash equilibrium in run-based strategies* and *Nash equilibrium in trace-based strategies* are defined analogously.

We first note that all negative results for deterministic systems immediately carry over to this more general setting as those are simply the case when deterministic strategy profiles induce a unique computation (a singleton set of computations). On the other hand, although positive results for computations and traces also carry over to nondeterministic systems, this is not something that one can immediately conclude. A couple of technical results are needed. In the reminder of this section we will study why positive results for computations and traces do carry over to nondeterministic systems.

The first observation to make is that the set of strategy profiles across bisimilar systems is invariant, that is, that every collection of (computation-based, trace-based) strategies $f = (f_1, \ldots, f_n)$ is a strategy profile in a system M if and only if f is a strategy profile in M', for every M' that is bisimilar to M.

Lemma 6.1. *Let M and M' be two bisimilar systems. For all (computation-based, trace-based) strategy profiles f:*

> *f is a strategy profile in M if and only if f is a strategy profile in M'.*

Proof. By induction on the length of computations/traces, and noting that, for every player, the set of actions available to a player in bisimilar states is the same (as otherwise the two states would not be bisimilar). □

The second observation is that, despite nondeterminism, the outcome of games across bisimilar systems is invariant. Formally, we have the following result.

Lemma 6.2. *Let M and M' be two bisimilar systems. For all (computation-based, trace-based) strategy profiles f we have*

$$\kappa_M(f) = \kappa_{M'}(f) \qquad \qquad and \qquad \qquad \tau_M(f) = \tau_{M'}(f) \ .$$

Proof. There are four different cases to consider here: either f is *computation-based* or f is *trace-based*, and either the outcome of the game is taken to be the set of *computations*, or the outcome of the game is taken to be the set of *traces*.

---

[7]Later on, in this section, we will present some examples of how sets of computations/runs/traces can be induced by (deterministic) computation-based/run-based/trace-based strategies in nondeterministic systems.

By double inclusion, we show the first case: f being *computation-based* and the outcome of the game taken to be the set of *computations*. To show that $\kappa_M(f) \subseteq \kappa_{M'}(f)$, with f computation-based, reason by contradiction. Suppose that there is a computation κ^* in $\kappa_M(f)$ that is not in $\kappa_{M'}(f)$. Since M and M' are bisimilar, κ^* is also a computation of M', and due to Lemma 6.1, for every prefix κ_k^* of κ^*, we know that $f(\kappa_k^*)$ is defined. Since f is functional, $f(\kappa_k^*)$ in M' is the same as $f(\kappa_k^*)$ in M, which must be precisely the last direction of κ_{k+1}^*. By an inductive argument we can conclude that κ^* must also be a computation of $\kappa_{M'}(f)$, which is a contradiction to our hypothesis, proving the statement. We can reason in a symmetric way to prove the inclusion in the other direction. Note that for computation-based strategies not only $\kappa_M(f) = \kappa_{M'}(f)$, but also they are *singleton* sets.

The second case we consider is when f is *trace-based* and the outcome of the game is taken to be the set of *traces*. To show this case, we can reason similarly, but, unlike for computation-based strategies, the sets $\tau_M(f)$ and $\tau_{M'}(f)$ may not be singleton sets. We, again, show the result by double inclusion, and each direction by contradiction. Thus, first, suppose that there is a trace τ^* in $\tau_M(f)$ that is not in $\tau_{M'}(f)$. Since M and M' are bisimilar, τ^* is also a trace of M', and due to Lemma 6.1, for every prefix τ_k^* of τ^*, we know that $f(\tau_k^*)$ is defined. Let τ_k^* be the smallest prefix of τ^* that is not a prefix of any trace in $\tau_{M'}(f)$, and let s be any state that can be reached after following the finite trace τ_{k-1}^* from s_M^0, the initial state of M. Then, we know that $s \xrightarrow{f(\tau_{k-1}^*)} q$, for some q such that $\lambda(q)$ is the last element of τ_k^*. Necessarily, the prefix τ_{k-1}^* is the prefix of some trace in $\tau_{M'}(f)$ that leads to a state, say s', that is bisimilar to s. Because s and s' are bisimilar, $s' \xrightarrow{f(\tau_{k-1}^*)} q'$ for some state q' that is bisimilar to q. Lemma 6.1 ensures that f is defined at τ_{k-1}^* in M'. Since q and q' are bisimilar, it also follows that $\lambda(q) = \lambda(q')$, and therefore that τ_k^*, with $\lambda(q')$ being the last element of τ_k^*, is the prefix of some trace in $\tau_{M'}(f)$, which is a contradiction to our hypothesis. Therefore, via induction on the length of traces, we can conclude, in particular, that $\tau^* \in \tau_{M'}(f)$, and in general that every trace in $\tau_M(f)$ must also be a trace in $\tau_{M'}(f)$. The inclusion in the other direction is, as before, obtained by symmetric reasoning.

The third case we consider is when f is *computation-based* and the outcome of the game is taken to be the set of *traces*. This proof is *almost* identical to the previous case. To show this case we, again, show the result by double inclusion, and each direction by contradiction. First, suppose that there is a trace τ^* in $\tau_M(f)$ that is not in $\tau_{M'}(f)$. Since M and M' are bisimilar, τ^* is also a trace of M', and due to Lemma 6.1, for every prefix τ_k^* of τ^* and every computation $\kappa_k \in \kappa(\tau_k^*)$, we know that $f(\kappa_k)$ is defined. Let τ_k^* be the smallest prefix of τ^* that is not a prefix of any trace in $\tau_{M'}(f)$, and let s be any state that can be reached after following the finite trace τ_{k-1}^* from s_M^0, the initial state of M. Then, we know that for some computation $\kappa_{k-1} \in \kappa(\tau_{k-1}^*)$, we have $s \xrightarrow{f(\kappa_{k-1})} q$, for some q such that $\lambda(q)$ is the last element of τ_k^*. Necessarily, the prefix τ_{k-1}^* is the prefix of some trace in $\tau_{M'}(f)$ that leads to a state, say s', that is bisimilar to s. Because s and s' are bisimilar, $s' \xrightarrow{f(\kappa_{k-1})} q'$ for some state q' that is bisimilar to q. Lemma 6.1 ensures that f is defined at κ_{k-1} in M'. Since q and q' are bisimilar, it also follows that $\lambda(q) = \lambda(q')$, and therefore that τ_k^*, with $\lambda(q')$ being the last element of τ_k^*, is the prefix of some trace in $\tau_{M'}(f)$, which is a contradiction to our hypothesis. Therefore, via induction on the length of traces, we can conclude, in particular, that $\tau^* \in \tau_{M'}(f)$, and in general that every trace in $\tau_M(f)$ must also be a trace in $\tau_{M'}(f)$. The inclusion in the other direction is also obtained by symmetric reasoning.

The fourth and final case we consider is when f is *trace-based* and the outcome of the game is taken to be the set of *computations*. This proof is also *almost* identical to the previous two cases. To show this case we, again, show the result by double inclusion, and each direction by contradiction. First, suppose that there is a computation κ^* in $\kappa_M(f)$ that is not in $\kappa_{M'}(f)$. Since M and M' are bisimilar, κ^* is also a computation of M', and due to Lemma 6.1, for every prefix κ_k^* of κ^* and every trace $\tau_k \in \tau(\kappa_k^*)$, we know that $f(\tau_k)$ is defined. Let κ_k^* be the smallest prefix of κ^* that is not a prefix of any computation in $\kappa_{M'}(f)$, and let s be any state that can be reached after following the finite computation κ_{k-1}^* from s_M^0, the initial state of M, while complying with τ_{k-1}^*, that is, any state s such that

$$\Theta_{k-1} = s_M^0 \xrightarrow{f(\lambda(s_M^0))} s_1 \xrightarrow{f(\lambda(s_M^0),\lambda(s_1))} \ldots \xrightarrow{f(\tau_{k-1})} s$$

with

$$\tau_{k-1}^* = \lambda(s_M^0), \lambda(s_1), \ldots, \lambda(s)$$

and

$$\kappa_{k-1}^* = f(\lambda(s_M^0)), f(\lambda(s_M^0), \lambda(s_1)), \ldots, f(\tau_{k-1}) \ .$$

Then, for trace $\tau_{k-1} \in \tau(\kappa_{k-1}^*)$ as above, we have $s \xrightarrow{f(\tau_{k-1}^*)} q$, for some q such that $\lambda(q)$ is the last element of τ_k^*. Necessarily, the prefix κ_{k-1}^* is the prefix of some computation in $\kappa_{M'}(f)$ that leads to a state, say s', that is bisimilar to s, that is, a computation

$$\kappa_{k-1}^* = f(\lambda(s_{M'}^0)), f(\lambda(s_{M'}^0), \lambda(s_1')), \ldots, f(\tau_{k-1})$$

with

$$\tau_{k-1}^* = \lambda(s_{M'}^0), \lambda(s_1'), \ldots, \lambda(s')$$

and

$$\Theta_{k-1}' = s_{M'}^0 \xrightarrow{f(\lambda(s_{M'}^0))} s_1' \xrightarrow{f(\lambda(s_{M'}^0),\lambda(s_1'))} \ldots \xrightarrow{f(\tau_{k-1})} s'$$

and $\Theta_{k-1}(i)$ bisimilar to $\Theta_{k-1}'(i)$, for every $0 \leq i \leq k-1$.

Because f is functional and

$$\lambda(s_M^0), \lambda(s_1), \ldots, \lambda(s) = \lambda(s_{M'}^0), \lambda(s_1'), \ldots, \lambda(s')$$

it follows that $f(\lambda(s_M^0), \lambda(s_1), \ldots, \lambda(s_k)) = f(\lambda(s_{M'}^0), \lambda(s_1'), \ldots, \lambda(s'))$ and that $s' \xrightarrow{f(\tau_{k-1}^*)} q'$ for some state q' bisimilar to s'. Lemma 6.1 ensures that f is defined at κ_{k-1} in M'. Since q and q' are bisimilar, it also follows that $\lambda(q) = \lambda(q')$, and therefore that τ_k^*, with $\lambda(q')$ being the last element of τ_k^*, is the prefix of some trace in $\tau_{M'}(f)$. Since τ_k^* is indeed a trace in $\tau_{M'}(f)$ which can be obtained following some computation κ_k^* in M', then we can conclude that κ_k^* is the prefix of some computation κ^* in $\kappa_{M'}(f)$, which is a contradiction to our hypothesis. Therefore, via induction on the length of computations, we can infer, in particular, that $\kappa^* \in \tau_{M'}(f)$, and in general that every computation in $\kappa_M(f)$ must also be a computation in $\kappa_{M'}(f)$. As in all previous cases, because M and M' are bisimilar, the inclusion in the other direction is also obtained by symmetric reasoning. $\square$

Figure 8: Two non-bisimilar systems where every computation-based strategy profile determines two different (infinite) trace outcomes, namely, x^ω in the system on the left (M) and y^ω in the system on the right (M').

Using Lemma 6.1 and Lemma 6.2, we can then show that the set of (computation-based, trace-based) Nash equilibria across bisimilar systems remains invariant too.

Theorem 6.3. *Let $G = (M, \Gamma_1, \ldots, \Gamma_n)$ and $G' = (M', \Gamma_1, \ldots, \Gamma_n)$ be games on bisimilar nondeterministic concurrent game structures M and M', respectively. Let further f^κ be a computation-based strategy profile and f^τ be a trace-based strategy profile. Then,*
(1) *f^κ is a computation-based Nash equilibrium in G if and only if f^κ is a computation-based equilibrium in G', and*
(2) *f^τ is a trace-based Nash equilibrium in G if and only if f^τ is a trace-based equilibrium in G'.*

Proof. Both proofs are by double implication, where each direction is proved by contradiction. For part (1), first assume that there is some computation-based equilibrium f^κ in G that is not a computation-based equilibrium in G'. Because of Lemma 6.2, every player i who gets its goal achieved in M also gets its goal achieved in M'. Then, they will not deviate in M'. Therefore, there must be a player j who does not get its goals achieved in M and has a beneficial deviation g_j in M', that is, while $\kappa_M(f^\kappa) \notin \Gamma_j$, we have $\kappa_{M'}(f^\kappa_{-j}, g_j) \in \Gamma_j$. Lemma 6.1 then ensures that g_i is also a strategy in M, and Lemma 6.2 that $\kappa_M(f^\kappa_{-j}, g_j) \in \Gamma_j$, which is a contradiction with f^κ being a computation-based Nash equilibrium in M. We can reason in a symmetric way to show the implication in the opposite direction. Notice that because of Lemmas 6.1 and 6.2, the result holds for both computation-based preferences and trace-based preferences. Finally, the proof for trace-based strategies (part (2)), with either computation-based or trace-based preferences, follows the exact same reasoning. □

We note that the main idea behind the proofs in this section is that if a given strategy profile f, whether computation-based or trace-based, does not determine the same set of computations and traces in bisimilar systems, then that computation or trace could be used to show that the two systems are in fact not bisimilar. This is the main argument behind the four cases in Lemma 6.2, each requiring slightly different proofs that such a witness to the non-bisimilarity of M and M' does not exist. However, it is also important to note that if M and M' are not bisimilar, then a given strategy profile f, well defined in both systems, may not determine the same set of outcomes—see, for instance, the example in Figure 8.

We would also like to note that even though for deterministic systems, computation-based strategies strictly generalise run-based strategies, and run-based strategies strictly generalise trace-base strategies, for nondeterministic systems this is no longer the case. Run-based strategies still generalise trace-based strategies, but not other relation like this holds between any other pair of models of strategies. For instance, as shown in the example in Figure 9, there is a system in which, for instance, a trace-based strategy can be defined (as well as a run-based strategy) while a computation-based strategy cannot.
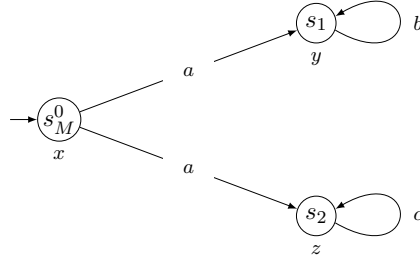
Figure 9: A nondeterministic system in which no computation-based strategy can be defined, but where both run-based and trace-based strategies can be defined.
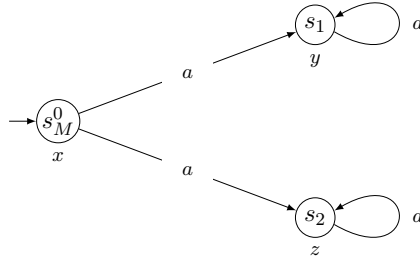

Figure 10: A system in which any computation-based strategy induces a set of runs and a set of traces containing, respectively, runs $s_M^0 s_1^\omega$ and $s_M^0 s_2^\omega$ and traces xy^ω and xz^ω.

In case strategies are well defined, as mentioned before, they can induce sets of computations, runs, and traces in a nondeterministic system, except for one case: whenever defined, a computation-based strategy always determines a unique computation of the system, whether deterministic or nondeterministic. Examples of all other cases (8 in total) are easy to build. For instance, in the nondeterministic system in Figure 9, any run-based or trace-based strategy will induce a set of computations containing both ab^ω and ac^ω. Correspondingly, they will also induce a set of runs and a set of traces, namely, those containing, respectively, $s_M^0 s_1^\omega$ and $s_M^0 s_2^\omega$ in case of runs, and xy^ω and xz^ω in case of traces. For the two remaining cases, a set of runs and a set of traces induced by a computation-based strategy, consider the nondeterministic system in Figure 10, which is almost the same as the system in Figure 9, save that a computation-based strategy can be defined. In such a system, any well defined computation-based strategy will induce a set of runs and a set of traces containing, respectively, $s_M^0 s_1^\omega$ and $s_M^0 s_2^\omega$ in case of runs, and xy^ω and xz^ω in case of traces.

Finally, the reader may have noticed that in this section we did not study the case considering run-based preferences (for run-based strategies we know that the negative results for deterministic systems carry over). The reason is that, as shown for deterministic systems, we can ensure invariance of (the existence of) Nash equilibria with respect to bisimilarity only if the sets of run-based preferences are congruent between bisimilar systems. As this is regarded as a major drawback, even for deterministic systems as illustrated in the example in Figure 11, it is really not an interesting question to be investigated any further.
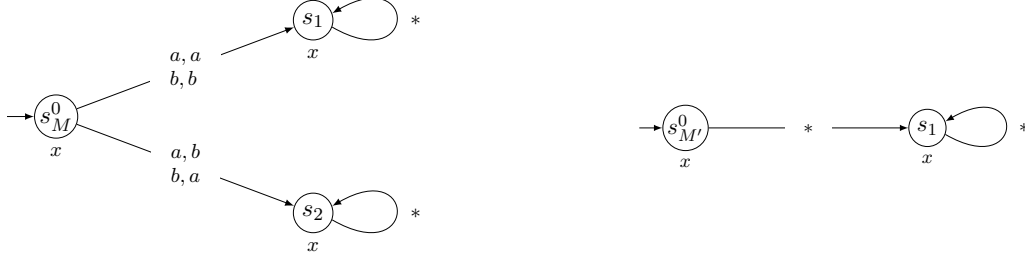
Figure 11: A pair of bisimilar systems, M and M', where the sets of run-based preferences given by $\Gamma_1 = \{s_M^0, s_1, s_1, \ldots\}$ for player 1 and $\Gamma_2 = \{s_M^0, s_2, s_2, \ldots\}$ for player 2 in system M, do not have a congruent counterpart in system M'.

7. Strategy Logics: New Semantic Foundations

Several logics for strategic reasoning have been proposed in the literature of computer science and AI, such as ATL* [1], Strategy Logic [35, 12], Coalition Logic [38], Coordination Logic [19], Game Logic [39], and Equilibrium Logic [25]. 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 [22] 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 [35], 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 [35]) as it has been given in [36]. 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. Intuitively, these operators can be understood as *"there exists a strategy x"*, *"for all strategies x"*, and *"bind agent i to the strategy associated with the variable x"*, respectively. Formally, SL formulae are inductively built from a set of atomic propositions AP, variables Var, and agents Ag, using the following grammar, where $p \in$ AP, $x \in$ Var, and $i \in$ Ag:

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

We also use the usual abbreviations for LTL formulae, that is, those for Boolean and temporal logic formulae.

We can now present the semantics of SL formulae. Given a concurrent game structure M, for all SL formulae φ, states $s \in \mathrm{St}$ in M, and assignments $\chi \in \mathrm{Asg} = (\mathrm{Var} \cup \mathrm{Ag}) \to \mathrm{Str}$, mapping variables and agents to strategies, the relation $M, \chi, s \models \varphi$ is defined as follows:

(1) $M, \chi, s \models p$ if $p \in \lambda(s)$, with $p \in \mathrm{AP}$.

(2) For all formulae φ, φ_1, and φ_2, we have:
   (a) $M, \chi, s \models \neg\varphi$ if not $M, \chi, s \models \varphi$;
   (b) $M, \chi, s \models \varphi_1 \wedge \varphi_2$ if $M, \chi, s \models \varphi_1$ and $M, \chi, s \models \varphi_2$.

(3) For all formulae φ and variables $x \in \mathrm{Var}$ we have:
   (a) $M, \chi, s \models \langle\langle x \rangle\rangle \varphi$ if there is a strategy $f \in \mathrm{Str}$ such that $M, \chi[x \mapsto f], s \models \varphi$;
   (b) $M, \chi, s \models [[x]]\varphi$ if for all strategies $f \in \mathrm{Str}$ we have that $M, \chi[x \mapsto f], s \models \varphi$.

(4) For all $i \in \mathrm{Ag}$ and $x \in \mathrm{Var}$, we have $M, \chi, s \models (i, x)\varphi$ if $M, \chi[i \mapsto \chi(x)], s \models \varphi$.

(5) Moreover, for all formulas φ, φ_1, and φ_2, we have:
   (a) $M, \chi, s \models \mathbf{X}\varphi$ if $M, (\chi, s)^1, \delta(s, d) \models \varphi$, where d is the decision taken from s by following χ and $(\chi, s)^1$ is the update of the assignment function as described in [35];
   (b) $M, \chi, s \models \varphi_1 \mathbf{U} \varphi_2$ if there exist $k \in \mathbb{N}$ such that $M, (\chi, s)^k, \delta(s, \vec{d}) \models \varphi_2$ and, for all $h \in \mathbb{N}$ with $h \leq k$, we have $M, (\chi, s)^h, \delta(s, \vec{d}_{\leq h}) \models \varphi_1$, where $\vec{d}$ is the sequence of decisions identified by the assignment function χ starting from s, and $(\chi, s)^k$ is the update of the assignment given by the execution of k steps of the strategy profile in χ starting from s.

Intuitively, rules 3a and 3b, respectively, are used to interpret the existential $\langle\langle x \rangle\rangle$ and universal $[[x]]$ quantifiers over strategies, and rule 4 is used to bind an agent to the strategies associated with variable x. All other rules are as in LTL over concurrent game structures.

As 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. 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. In SL, it is possible to represent the existence of a Nash equilibrium in a concurrent game structure [35]. This implies, given Theorem 4.5, that SL under the standard interpretation is not invariant under bisimulation, as the formula expressing the existence of a Nash equilibrium can distinguish between two bisimilar models.

Given the semantics of SL formulae given above, 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 7.1. *Let $M_1 = (\mathrm{Ag}, \mathrm{AP}, \mathrm{Ac}, \mathrm{St}_1, s_1^0, \lambda_1, \delta_1)$ and $M_2 = (\mathrm{Ag}, \mathrm{AP}, \mathrm{Ac}, \mathrm{St}_2, s_2^0, \lambda_2, \delta_2)$ be two bisimilar CGSs. Moreover, let χ be an assignment of strategies and $s_1 \sim s_2$ be two bisimilar states. Then, for all $\varphi \in SL$, it holds that*

$$M_1, \chi, s_1, \models \varphi \qquad \text{if and only if} \qquad M_2, \chi, s_2 \models \varphi.$$

Proof. The proof proceeds by induction on the structure of φ. First note that we do not need to prove all the cases, as, for example, we have that $\psi_1 \vee \psi_2 = \neg(\neg\psi_1 \wedge \neg\psi_2)$ and $[[x]]\varphi = \neg\langle\langle x \rangle\rangle\neg\varphi$. Moreover, recall from Lemma 3.1 that every computation-based strategy in M_1 is also a computation-based strategy in M_2 and vice-versa. We have the following.

- $\varphi = p \in \mathrm{AP}$. We have that $M_1, \chi, s_1 \models p$ if and only if $p \in \lambda_1(s_1) = \lambda_2(s_2)$ if and only if $M_2, \chi, s_2 \models p$;
- $\varphi = \neg\psi$. We have that $M_1, \chi, s_1 \models \neg\psi$ if and only if $M_1, \chi, s_1 \not\models \psi$ if and only if, by induction hypothesis, $M_2, \chi, s_2 \not\models \psi$ if and only if $M_2, \chi, s_2 \models \neg\psi$.

- $\varphi = \psi_1 \wedge \psi_2$. We have that $M_1, \chi, s_1 \models \psi_1 \wedge \psi_2$ if and only if $M_1, \chi, s_1 \models \psi_1$ and $M_1, \chi, s_1 \models \psi_2$, which holds, by induction hypothesis, if and only if $M_2, \chi, s_2 \models \psi_1$ and $M_2, \chi, s_2 \models \psi_2$ if and only if $M_2, \chi, s_2 \models \psi_1 \wedge \psi_2$.
- $\varphi = \mathbf{X}\psi$. We have that $M_1, \chi, s_1 \models \mathbf{X}\psi$ if and only if $M_1, (\chi)^1, \delta_1(s_1, d) \models \psi$, where $d = (\chi(1)(\epsilon), \ldots \chi(n)(\epsilon))$ is the decision taken by the agents on the first round of the game, according the assignment χ. By bisimilarity, we have that $\delta_1(s_1, d) \sim \delta_2(s_2, d)$, and so, by induction hypothesis, that $M_2, (\chi)^1, \delta_2(s_2, d) \models \psi$, that holds if and only if $M_2, \chi, s_2 \models \mathbf{X}\psi$.
- $\varphi = \varphi_1 \mathbf{U} \varphi_2$. We have that $M_1, \chi, s_1 \models \varphi_1 \mathbf{U} \varphi_2$ if and only if there exists $k \in \mathbb{N}$ such that $M_1, \chi^k, \delta_1^*(s_1, \vec{d}) \models \varphi_2$ and $M_1, \chi^h, \delta_1^*(s_1, \vec{d}_{\leq h}) \models \varphi_1$ for every $h < k$, where $\vec{d}$ is the unique sequence of decisions identified by the k-steps application of the transition function that follows χ. Observe that, for each $h \leq k$, we have that $\delta_1(s_1, \vec{d}_{\leq h}) \sim \delta_2(s_2, \vec{d}_{\leq h})$ and so, by induction hypothesis, we have that $M_2, \chi^k, \delta_2^*(s_2, \vec{d}) \models \varphi_2$ and $M_2, \chi^h, \delta_2^*(s_1, \vec{d}_{\leq h}) \models \varphi_1$ for every $h < k$, that is, if and only if $M_2, \chi, s_2 \models \varphi_1 \mathbf{U} \varphi_2$.
- $\varphi = (i, x)\psi$. We have that $M_1, \chi, s_1 \models (i, x)\psi$ if and only if $M_1, \chi[i \mapsto \chi(x)], s_1 \models \psi$ if and only if, by induction hypothesis, $M_1, \chi[i \mapsto \chi(x)], s_2 \models \psi$ if and only if $M_2, \chi, s_2 \models (i, x)\psi$.
- $\varphi = \langle\langle x \rangle\rangle \psi$. We have that $M_1, \chi, s_1 \models \langle\langle x \rangle\rangle \psi$ if and only if there exists a strategy $f \in \text{Str}$ such that $M_1, \chi[x \mapsto f], s_1 \models \psi$ if and only if, by induction hypothesis $M_2, \chi[x \mapsto f], s_2 \models \psi$, if and only if $M_2, \chi, s_2 \models \langle\langle x \rangle\rangle \psi$.

This concludes the proof. $\square$

As an immediate corollary, we then obtain the following result about the semantic relationship between the properties that can be expressed in SL and the concept of bisimilarity.

Corollary 7.2. *SL with the computation-based model of strategies is invariant under bisimilarity.*

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

Corollary 7.3. *SL with the trace-based model of strategies is invariant under bisimilarity.*

8. CONCLUDING REMARKS AND RELATED WORK

In this paper we showed that with the conventional model of strategies used in the logic, computer science, and AI literatures, the existence of Nash equilibria is not necessarily preserved under bisimilarity—in particular this is the case for multi-player games played over deterministic concurrent games structures. By way of some examples, we also illustrated some of the implications of this result—for example, in the context of automated formal verification. To resolve this difficulty, 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*, even on nondeterministic systems. We studied applications of these models and found that through their use, not only Nash equilibria become invariant under bisimilarity, but also full logics such as Strategy Logic. 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, Nash equilibria, and automated formal verification is summarised next.

Logics for Strategic Reasoning. There is now a large literature on logics for strategic reasoning. From this literature, ATL* [1] and SL [35] stand out, both due to their use within 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 [22]. 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 (which correspond to preferences over traces).

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]. Our results also apply to such alternating equivalence relations. Alternating bisimulation is very similar to bisimilarity on labelled transition systems [33, 28], only that when defined on concurrent game structures, instead of action profiles taken as possible transitions, one allows individual player's actions, which must be matched in the bisimulation game. Because of this, it immediately follows that any alternating bisimulation as defined in [3] is also a bisimilarity as defined here. Despite having a different formal definition, a simple observation can be made: that the counter-example shown in Figures 1 and 2 also apply to such alternating (bisimulation) relations. This immediately implies that Nash equilibria are not preserved by the alternating (bisimulation) equivalence relations in [3] either. Nevertheless, as discussed in [45], the "right" notion of equivalence for games and their game theoretic solution concepts is, undoubtedly, an important and interesting topic of debate, which deserves to be investigated further.

Computations vs. Traces. An important remark about the 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 again 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 are preserved by bisimilarity. However, if we consider, instead, the model of trace-based strategies and the same LTL goals, then we obtain two concurrent 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. 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 zero-sum two-player games together with temporal logic specifications (*e.g.*, using LTL, CTL, or ATL*). Our results, then, provide conclusive 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 formal verification techniques.

Nondeterminism. We extended our main bisimulation-invariant results to nondeterministic systems, making it possible to analyse more complex systems. This was possible, in turn, because our two main models of strategies, namely computation-based and trace-based, are themselves oblivious to nondeterministic choices. As a consequence, given a particular strategy (or strategy profile, more generally), the set of outcomes of a multi-player game across bisimilar structures remains the same. Indeed, the definitions of strategies in the computation-based and trace-based models can be used to show that the set of Nash equilibria in strategy profiles given by these two models is invariant across systems that are equivalent with respect to equivalences for concurrency that are weaker than bisimilarity; for instance, across trace equivalent systems as defined in CSP [9]. Thus, with respect to this kind of systems, all our positive results also carry over, even for nondeterministic processes.

Tools for Model Checking and Equilibrium Analysis. Due to the success of temporal logics and model checking in the verification of concurrent and multi-agent systems, some model checking tools have been extended to cope with the strategic analysis of concurrent systems modelled as multi-player games. For instance, tools such as MCMAS [11], EAGLE [43], PRALINE [8], MOCHA [4], and PRISM [30], allow for the analysis of *some* strategic properties in a system. Because all of these tools rely on underlying algorithms for temporal logic model checking, hardly any optimisations are possible when moving to the more complex game-theoretic setting where Nash equilibria needs to be analysed. In this way, our results find a powerful, and immediate, practical application. Indeed, based on the work presented in this paper, we have developed a new tool for temporal equilibrium analysis [26], which uses the computation-based model of strategies studied here.

As mentioned before, we have developed a new tool for temporal equilibrium analysis, which we call EVE [26] (Equilibrium Verification Environment). EVE uses the computation-based model of strategies and trace-based preferences given by LTL formulae. EVE is a formal verification tool for the automated analysis of temporal equilibrium properties of concurrent and multi-agent systems modelled using the Simple Reactive Module Language (SRML [2, 46]) as a collection of independent system components (players/agents in a game). In particular, EVE automatically solves three key decision problems in rational synthesis and verification [24, 48, 20]: NON-EMPTINESS, E-NASH, and A-NASH. These problems ask, respectively, whether a multi-player game has at least one Nash equilibrium, whether an LTL formula holds on *some* Nash equilibrium, and whether an LTL formula holds on *all* Nash

equilibria. EVE uses a technique based on parity games to check for the existence of Nash equilibria in a concurrent and multi-player game, which crucially relies on the underlying model of strategies being bisimulation invariant.

Acknowledgment

References

[1] R. Alur, T. Henzinger, and O. Kupferman. Alternating-Time Temporal Logic. *Journal of the ACM*, 49(5):672–713, 2002.

[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 *CONCUR*, volume 1466 of *LNCS*, pages 163–178. Springer, 1998.

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

[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 S. Chakraborty and A. Kumar, editors, *31st International Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'11)*, pages 375—386, 2011.

[7] P. Bouyer, R. Brenguier, N. Markey, and M. Ummels. Pure Nash equilibria in concurrent deterministic games. *Logical Methods in Computer Science*, 11(2:9):1–72, 2015.

[8] R. Brenguier. PRALINE: A tool for computing nash equilibria in concurrent games. In *CAV*, volume 8044 of *LNCS*, pages 890–895. Springer, 2013.

[9] S. D. Brookes, C. A. R. Hoare, and A. W. Roscoe. A theory of communicating sequential processes. *Journal of the ACM*, 31(3):560–599, 1984.

[10] R. E. Bryant. Symbolic boolean manipulation with ordered binary-decision diagrams. *ACM Comput. Surv.*, 24(3):293–318, 1992.

[11] P. Cermák, A. Lomuscio, F. Mogavero, and A. Murano. MCMAS-SLK: A model checker for the verification of strategy logic specifications. In *CAV*, volume 8559 of *LNCS*, pages 525–532. Springer, 2014.

[12] K. Chatterjee, T. A. Henzinger, and N. Piterman. Strategy logic. *Inf. Comput.*, 208(6):677–693, 2010.

[13] E. Clarke, O. Grumberg, and D. Peled. *Model Checking*. MIT Press, 2002.

[14] E. M. Clarke and E. A. Emerson. Design and synthesis of synchronization skeletons using branching-time temporal logic. In *Logics of Programs*, volume 131 of *LNCS*, pages 52–71. Springer, 1981.

[15] E. M. Clarke, O. Grumberg, and D. E. Long. Model checking and abstraction. *ACM Trans. Program. Lang. Syst.*, 16(5):1512–1542, 1994.

[16] P. Cousot and R. Cousot. On abstraction in software verification. In *CAV*, volume 2404 of *LNCS*, pages 37–56. Springer, 2002.

[17] R. De Nicola and F. W. Vaandrager. Three logics for branching bisimulation. *Journal of the ACM*, 42(2):458–487, 1995.

[18] S. Demri, V. Goranko, and M. Lange. *Temporal Logics in Computer Science*, volume 58 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 2016.

[19] B. Finkbeiner and S. Schewe. Coordination logic. In A. Dawar and H. Veith, editors, *CSL*, volume 6247 of *LNCS*, pages 305–319. Springer, 2010.

[20] D. Fisman, O. Kupferman, and Y. Lustig. Rational synthesis. In *TACAS*, volume 6015 of *LNCS*, pages 190–204. Springer, 2010.

[21] J. Gutierrez, P. Harrenstein, G. Perelli, and M. Wooldridge. Nash equilibrium and bisimulation invariance. In *CONCUR*, volume 85 of *LIPIcs*, pages 17:1–17:16. Schloss Dagstuhl, 2017.

[22] J. Gutierrez, P. Harrenstein, and M. Wooldridge. Expresiveness and complexity results for strategic reasoning. In *CONCUR*, volume 42 of *LIPIcs*, pages 268–282. Schloss Dagstuhl, 2015.

[23] J. Gutierrez, P. Harrenstein, and M. Wooldridge. Iterated Boolean games. *Inf. Comput.*, 242:53–79, 2015.

[24] J. Gutierrez, P. Harrenstein, and M. Wooldridge. From model checking to equilibrium checking: Reactive modules for rational verification. *Artificial Intelligence*, 248:123–157, 2017.

[25] J. Gutierrez, P. Harrenstein, and M. Wooldridge. Reasoning about equilibria in game-like concurrent systems. *Ann. Pure Appl. Logic*, 168(2):373–403, 2017.

[26] J. Gutierrez, M. Najib, G. Perelli, and M. Wooldridge. EVE: A tool for temporal equilibrium analysis. In *ATVA*, LNCS. Springer, 2018.

[27] J. Gutierrez and M. Wooldridge. Equilibria of concurrent games on event structures. In *CSL-LICS*, pages 46:1–46:10. ACM, 2014.

[28] M. Hennessy and R. Milner. Algebraic laws for nondeterminism and concurrency. *J. ACM*, 32(1):137–161, 1985.

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

[30] M. Kwiatkowska, D. Parker, and C. Wiltsche. Prism-games 2.0: A tool for multi-objective strategy synthesis for stochastic games. In *TACAS*, volume 9636 of *LNCS*, pages 560–566. Springer, 2016.

[31] M. Z. Kwiatkowska, G. Norman, and D. Parker. PRISM 4.0: Verification of probabilistic real-time systems. In *CAV*, volume 6806 of *LNCS*, pages 585–591. Springer, 2011.

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

[33] R. Milner. *A Calculus of Communicating Systems*, volume 92 of *LNCS*. Springer, 1980.

[34] R. Milner. *Communication and Concurrency*. Prentice Hall, 1989.

[35] F. Mogavero, A. Murano, G. Perelli, and M. Vardi. Reasoning About Strategies: On the Model-Checking Problem. *ACM Transactions On Computational Logic*, 15(4):34:1–42, 2014.

[36] F. Mogavero, A. Murano, G. Perelli, and M. Y. Vardi. Reasoning about Strategies: on the Satisfiability Problem. *CoRR*, abs/1611.08541, 2016.

[37] M. Osborne and A. Rubinstein. *A Course in Game Theory*. MIT Press, 1994.

[38] M. Pauly. A modal logic for coalitional power in games. *J. Log. Comput.*, 12(1):149–166, 2002.

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

[40] A. Pnueli. The temporal logic of programs. In *FOBS*, pages 46–57. IEEE Computer Society, 1977.

[41] D. Sangiorgi. On the origins of bisimulation and coinduction. *ACM Trans. Program. Lang. Syst.*, 31(4), 2009.

[42] D. Sangiorgi and J. Rutten, editors. *Advanced Topics in Bisimulation and Coinduction*, volume 52 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 2012.

[43] A. Toumi, J. Gutierrez, and M. Wooldridge. A tool for the automated verification of nash equilibria in concurrent games. In *ICTAC*, volume 9399 of *LNCS*, pages 583–594. Springer, 2015.

[44] J. van Benthem. *Modal Correspondence Theory*. PhD thesis, University of Amsterdam, 1976.

[45] J. van Benthem. Extensive games as process models. *Journal of Logic, Language and Information*, 11(3):289–313, 2002.

[46] W. van der Hoek, A. Lomuscio, and M. Wooldridge. On the complexity of practical ATL model checking. In *AAMAS*, pages 201–208. ACM, 2006.

[47] R. J. van Glabbeek and W. P. Weijland. Branching time and abstraction in bisimulation semantics. *Journal of the ACM*, 43(3):555–600, 1996.

[48] M. Wooldridge, J. Gutierrez, P. Harrenstein, E. Marchioni, G. Perelli, and A. Toumi. Rational verification: From model checking to equilibrium checking. In *AAAI*, pages 4184–4191. AAAI Press, 2016.