Logical Methods in Computer Science
Vol. 13(4:15)2017, pp. 1–40
https://lmcs.episciences.org/

Submitted Nov. 03, 2016
Published Nov. 22, 2017

GAMES FOR BISIMULATIONS AND ABSTRACTION

DAVID DE FRUTOS ESCRIG, JEROEN J.A. KEIREN, AND TIM A.C. WILLEMSE

Dpto. Sistemas Informáticos y Computación - Facultad CC. Matemáticas, Universidad Complutense de Madrid, Spain
e-mail address: defrutos@sip.ucm.es

Open University in the Netherlands and Radboud University, Nijmegen, The Netherlands
e-mail address: Jeroen.Keiren@ou.nl

Eindhoven University of Technology, Eindhoven, The Netherlands
e-mail address: T.A.C.Willemse@TUe.nl

Abstract. Weak bisimulations are typically used in process algebras where silent steps are used to abstract from internal behaviours. They facilitate relating implementations to specifications. When an implementation fails to conform to its specification, pinpointing the root cause can be challenging. In this paper we provide a generic characterisation of branching-, delay-, η- and weak-bisimulation as a game between Spoiler and Duplicator, offering an operational understanding of the relations. We show how such games can be used to assist in diagnosing non-conformance between implementation and specification. Moreover, we show how these games can be extended to distinguish divergences.

1. Introduction

Abstraction is a powerful, fundamental concept in process theories. It facilitates reasoning about the conformance between an implementation and specification of a (software) system, described by a transition system. Essentially, it allows one to ignore (*i.e.*, abstract from) implementation details that are unimportant from the viewpoint of the specification.

There is a wealth of behavioural equivalences (and preorders), each treating abstraction in slightly different manners [29]. Some prototypical equivalences have been incorporated in contemporary tool sets that implement verification technology for (dis)proving the correctness of software systems. These equivalences include branching bisimilarity [32] and branching bisimilarity with explicit divergence [30], which are both used in tool sets such as CADP [13], μCRL [5], and mCRL2 [7], and weak bisimilarity [20] which is used in, for example, FDR [15].

The key idea behind these weak behavioural equivalences is that they abstract from 'internal' events (events that are *invisible* to the outside observer of a system). At the same time, they preserve essential parts of the branching structure of the transition systems.

2012 ACM CCS: [**Theory of computation**]: Concurrency; Algorithmic game theory; Semantics and reasoning.

Key words and phrases: behavioural equivalences, bisimulation games, abstraction, branching bisimulation, divergence.

LOGICAL METHODS IN COMPUTER SCIENCE DOI:10.23638/LMCS-13(4:15)2017

The behavioural equivalences differ in the extent to which the branching structure is preserved. Allowing abstraction from invisible actions can make it difficult to explain why a particular pair of states is not equivalent, as one must somehow capture the loss of potential future computations in the presence of internal actions. While (theoretical) tools such as distinguishing formulae can help to understand why two states are distinguishable, these may not always be very accessible and, to date, the idea of integrating such formulae in tool sets seems not to have caught on, as witnessed by their absence in the four prominent tool sets for analysing labelled transition systems, *viz.* CADP [13], μCRL [5], mCRL2 [7] and FDR [15]; the only tool supporting distinguishing formulae seems to be the (education-focussed) tool CAAL [1], which uses both games and formulae. We note that distinguishing formulae and games for behavioural equivalences are complementary. Formulae offer a high-level explanation for the inequivalence of two systems in terms of their 'capabilities', whereas the game characterisation allows for explaining their inequivalence at the level of the structure of the transition systems.

Contributions. We provide game-based views on four weak behavioural equivalences. Such a game-based view provides an alternative point of view on the traditional coinductive definitions. Moreover, we argue, using some examples, how such games can be used to give an operational explanation of the inequivalence of states following the ideas in [26], thereby remaining close to the realm of transition systems. Our first main contribution can be summarised as follows:

> *Contribution 1.* We show that branching-, η-, delay-, and weak-bisimilarity can be characterised by Ehrenfeucht-Fraïssé games [28] in Theorem 5.2.

We do not obtain just a collection of (more or less) similar game-based characterisations. Instead, we present a generic, parameterised characterisation that captures all these (closely related) semantics in a single framework, resulting in a single definition for all four games. In addition, this unified presentation enables a generic proof of the correctness of our game characterisation.

We furthermore study the notion of divergence for all four weak behavioural equivalences. A divergence can be understood as a computation that only involves unobservable behaviour. From the perspective of an observer, a system that diverges can thus appear to be 'stuck'. For that reason, it may be undesirable to abstract from divergences. While coinductive definitions of divergence-aware weak behavioural equivalences treat divergence—in some sense—as a separate issue, we show that divergence is obtained through a natural modification of our generic game-based characterisation of these equivalences. Our second main contribution is therefore as follows:

> *Contribution 2.* We generalise branching bisimilarity with explicit divergence [30] to branching-, η-, delay-, and weak-bisimilarity with explicit divergence and we show that all four notions can be characterised by Ehrenfeucht-Fraïssé games in Theorem 6.10.

This paper is an extended and enhanced version of [9], where we defined games for branching bisimilarity, and branching bisimilarity with explicit divergence and proved that these games characterise branching bisimilarity and branching bisimilarity with explicit divergence, respectively. A major difference between the games of the current paper and the game-based characterisations of [9] is that in *ibid.* we employed the stuttering condition in the definition of branching bisimulation, whereas in this paper we need to resort to a slightly different mechanism for dealing with the four weak behavioural equivalences in a unified way. The stuttering condition underlies branching bisimilarity and increases its local coinductive character, thus allowing to arrive at a 'simple' game. In Theorem 5.14, we nevertheless show how the branching bisimulation game of [9] can be recovered from our generic game. The proof of this theorem exposes the close connection between the two games.

Related work. The idea of using two-player games for bisimulation originates with Stirling, who described *strong bisimulation* games [27]. Yin *et al.* recently presented a branching bisimulation game, in the context of normed process algebra [35], but their game uses moves that consist of sequences of silent steps, rather than single steps. As argued convincingly by Namjoshi [22], local reasoning using single steps often leads to simpler arguments.

The two-player game for divergence-blind stuttering bisimilarity [10] (a relation for Kripke structures that in essence is the same as branching bisimilarity), provided by Bulychev *et al.* in [6] comes closer to our work on games for branching bisimilarity. However, their game-based definition is sound only for transition systems that are essentially free of divergences, so that in order to deal with transition systems containing divergences they need an additional step that precomputes and eliminates these divergences. Such a preprocessing step is a bit artificial, and makes it hard to present the user with proper diagnostics, as it fundamentally modifies the structure of the transition system. Moreover, infinite state transition systems cannot be dealt with this way. As far as we are aware, ours is the first work that tightly integrates dealing with divergences in a game-based characterisation of a behavioural equivalence.

In a broader context, the type of games we study in this paper have been used for characterising different types of relations in settings other than labelled transition systems. For instance, in [18], simulation games on Büchi automata are defined with the purpose of approximating the inclusion of trace closures of languages accepted by finite-state automata. Similarly, in [11], several algorithms based on games characterising a variety of simulation relations on the state space of a Büchi automaton are studied. In [12], a delayed simulation game for parity games is defined and investigated, and in [8], the branching bisimulation games we studied in [9] are lifted to the setting of parity games.

There is a large number of behavioural equivalences in the context of abstraction, each having its own applications and properties. An extensive overview of these was given by van Glabbeek [29]. In practice, behavioural equivalences are used that can be computed efficiently, yet that still allow to equate a large number of systems. Weak bisimilarity [20] and branching bisimilarity [32] are the dominant equivalences in use. The notions of η-bisimilarity [2] and delay bisimilarity [21] are between weak- and branching bisimilarity in terms of distinguishing power.

Divergence has been studied as a concept orthogonal to bisimulation. Different notions of divergence have been defined in the literature, differing in their abilities to, for example, distinguish livelocks (infinite, internal computations) from deadlock. Milner, *e.g.* studied a notion of observation equivalence with divergence [21]. In the branching bisimulation setting, the notions of branching bisimilarity with explicit divergence [32] and divergence sensitive branching bisimilarity [10] have been studied. For the first, different, yet equivalent characterisations have been explored by van Glabbeek *et al.* [30].[1]

Outline of the paper. Labelled transition systems, and strong- and branching bisimilarity are introduced in Section 2. Subsequently, we introduce bisimulation games, using Stirling's games as an example, in Section 3. In the same section, we present Bulychev's version of branching bisimulation games, and explain its shortcomings, and we introduce the branching bisimulation games from [9]. Next, in Section 4, we introduce the coinductive definitions of weak-, delay-, and η-bisimilarity, and present a generic bisimulation definition that covers all three bisimulations as well as branching bisimulation. In Section 5 we introduce a generic bisimulation game that is parameterised such that it covers all four bisimulations. We discuss the concepts of divergence and simulation, and the changes to our generic bisimulation game required to capture these in Section 6. We illustrate our game using a small example in Section 7. Finally, we conclude in Section 8.

2. Preliminaries

We are concerned here with relations on labelled transition systems that include both *observable* transitions, and *internal* transitions labelled by the special action τ.

Definition 2.1. A *Labelled Transition System* (LTS) is a structure $L = \langle S, A, \to \rangle$ where:
- S is a set of states,
- A is a set of actions containing a special action τ,
- $\to \subseteq S \times A \times S$ is the transition relation.

As usual, we write $s \xrightarrow{a} t$ to stand for $(s, a, t) \in \to$. The reflexive-transitive closure of the $\xrightarrow{\tau}$ relation is denoted by $\twoheadrightarrow$ and the transitive closure of the $\xrightarrow{\tau}$ relation is denoted $\twoheadrightarrow^+$. Given a relation $R \subseteq S \times S$ on states, we simply write $s\ R\ t$ to represent $(s, t) \in R$. We say that a labelled transition system is *divergent* if it admits an infinite sequence $s \xrightarrow{\tau} s_1 \xrightarrow{\tau} s_2 \cdots$ from some state $s \in S$, and we say it is *non-divergent* if it contains no such sequence.

Strong bisimulation, due to Park [23], is arguably the finest meaningful behavioural equivalence defined for labelled transition systems.

Definition 2.2 ([23])**.** A symmetric relation $R \subseteq S \times S$ is said to be a *strong bisimulation* whenever for all $s\ R\ t$, if $s \xrightarrow{a} s'$ then there exists a state t' such that $t \xrightarrow{a} t'$ and $s'\ R\ t'$. We write $s \leftrightarrow t$ and say that s and t are strongly bisimilar if and only if there is a strong bisimulation relation R such that $s\ R\ t$.

In this paper we are mainly concerned with weaker bisimulations, that allow to abstract from *internal* transitions. The finest such bisimulation is *branching bisimulation*, that was introduced by van Glabbeek and Weijland in [32].

Definition 2.3 ([32])**.** A symmetric relation $R \subseteq S \times S$ is said to be a *branching bisimulation* whenever $s\ R\ t$ and $s \xrightarrow{a} s'$ imply either:
- $a = \tau$ and $s'\ R\ t$, or
- there exist states t', t_1 such $t \twoheadrightarrow t_1 \xrightarrow{a} t'$, $s\ R\ t_1$ and $s'\ R\ t'$;

[1] Note that the literature uses the same names for different notions of sensitivity to divergence, but also uses different names for the same notion of divergence, so when studying these issues one needs to tread carefully.

We write $s \leftrightarrow_b t$ and say that s and t are branching bisimilar, iff there is a branching bisimulation R such that $s \mathrel{R} t$. Typically we simply write $\leftrightarrow_b$ to denote *branching bisimilarity.*

Basten [3], finally showed that branching bisimilarity is indeed an equivalence relation.

Theorem 2.4 ([3, Corollary 13])**.** *Branching bisimilarity, $\leftrightarrow_b$, is an equivalence relation.*

Branching bisimilarity has the *stuttering property*, see [32, Lemma 2.5]. This means that the condition $s \mathrel{R} t''$ imposed on the weak transition $t \twoheadrightarrow t''$ at Definition 2.3 can be strengthened, so that all the intermediate states t''_i along the weak transition $t \twoheadrightarrow t''$, and not just the final state t'', will have to satisfy the condition $s \mathrel{R} t''_i$. This fact is quite useful when studying other properties of branching bisimilarity, and in particular plays a central role when developing the characterisation of branching bisimilarity by means of a branching bisimulation game.

Definition 2.5 ([32, Lemma 2.5], [25, 30])**.** A relation R has the *stuttering property* if, whenever $t_0 \xrightarrow{\tau} t_1 \cdots \xrightarrow{\tau} t_k$ with $t_0 \mathrel{R} t_k$, then $t_i \mathrel{R} t_j$, for all $0 \leq i, j \leq k$.

3. Games for Strong and Branching Bisimulation

In the previous section we have reiterated the coinductive definitions of strong- and branching bisimulation. There are several alternatives for defining behavioural equivalences. Relations can, for example be defined using a fixed point characterisation, or using a two-player game. In this paper we are concerned with such two-player games. We first give some background on the games, and recall strong bisimulation games, before we proceed to the branching bisimulation games that we introduced in [9].

3.1. Games.

The games we consider in this paper are, essentially, Ehrenfeucht-Fraïssé games, whose use in computer science has been reviewed in [28]. They are instances of two-player infinite-duration games with ω-regular winning conditions, played on game arenas that can be represented by graphs. In these games each vertex is assigned to one of two players, here called Spoiler and Duplicator. The players move a token over the vertices as follows. The player that 'owns' the vertex where the token is pushes it along an edge to an adjacent vertex, and this continues as long as possible, possibly forever. The winner of the play is decided from the resulting sequence of vertices visited by the token, depending on the predetermined winning criterion. We say that a player *can win from a given vertex* if she has a strategy such that any play with the token initially at that vertex will be won by her. The games that we consider here are *memoryless* and *determined*: every vertex is won by (exactly) one player, and the winning player has a *positional* winning strategy, meaning that she can decide her winning moves based only on the vertex where the token currently resides, without inspecting the previous moves of the play. A strategy of a player can thus be given by a function that maps each vertex owned by this player to one of its adjacent vertices. A play is *consistent* with a strategy of a given player whenever on this play, the successor of each vertex owned by this player is given by the strategy. A strategy of a player induces a *solitaire game* by restricting the edges in the graph that emanate from this player's set of vertices to those used by her strategy. Note that a play that is consistent both with a strategy of Spoiler and a strategy of Duplicator is a unique path in the graph. Winning strategies can be efficiently computed while solving the game. We refer to [16] for a more in-depth treatment of the underlying theory.

3.2. **Stirling Games for Strong Bisimulation.** The most well-known, and simplest game-based characterisation of a behavioural equivalence is Stirling's notion of a *strong bisimulation game* [27].

The game is played on pairs of states (s,t) in a labelled transition system, where DUPLICATOR tries to prove that states s and t are strongly bisimilar, whereas SPOILER tries to disprove this. To achieve this, SPOILER plays first, and challenges DUPLICATOR with a move from either s or t, and DUPLICATOR is required to match this move. If play continues indefinitely, this means DUPLICATOR is always able to match SPOILER's challenges, and the states are equivalent. If not, this means SPOILER found a way to distinguish the states. The game is formally defined as follows.

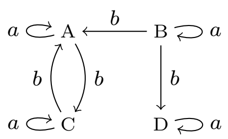
Definition 3.1. A *strong bisimulation (sb) game* on an LTS L is played by players SPOILER and DUPLICATOR on an arena of SPOILER-owned configurations $[\,(s,t)\,]$ and DUPLICATOR-owned configurations $\langle\,(s,t),c\,\rangle$, where $(s,t) \in \mathit{Position}$ and $c \in \mathit{Challenge}$, and $\mathit{Position} = S \times S$ is the set of *positions* and $\mathit{Challenge} = (A \times S)$ the set of *pending challenges*.

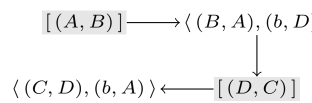
- SPOILER moves from a configuration $[\,(s,t)\,]$ by:
  (1) selecting $s \xrightarrow{a} s'$ and moving to $\langle\,(s,t),(a,s')\,\rangle$, or
  (2) selecting $t \xrightarrow{a} t'$ and moving to $\langle\,(t,s),(a,t')\,\rangle$.
- DUPLICATOR responds from a configuration $\langle\,(u,v),(a,u')\,\rangle$ by playing $v \xrightarrow{a} v'$ and continuing in configuration $[\,(u',v')\,]$.

Finite plays are won by DUPLICATOR if and only if SPOILER gets stuck. All infinite plays are won by DUPLICATOR. Full plays of the game start in a configuration $[\,(s,t)\,]$; we say that DUPLICATOR wins the sb-game for a position (s,t), if the configuration $[\,(s,t)\,]$ is won by her; in this case, we write $s \equiv_s t$. Otherwise, we say that SPOILER wins that game.

We illustrate the above game using a small example.

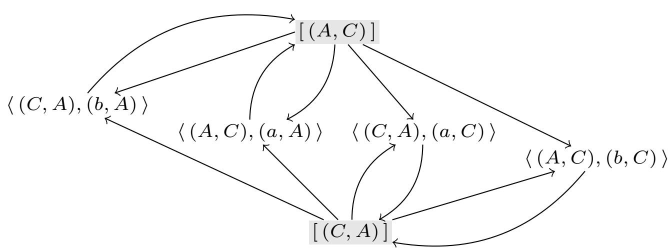
Example 3.2. Consider the four-state transition system depicted below (left).





Observe that states A and B are not strongly bisimilar. This can be seen by the (solitaire) game, depicted right, in which SPOILER plays her winning strategy, first challenging DUPLICATOR to match a b-transition, and, when she does so by moving to C, switch positions and challenging her to match another b-transition (which DUPLICATOR cannot).

States A and C, on the other hand, are strongly bisimilar. While it is easy to check that, indeed, the relation $R = \{(A,C),(C,A)\}$ is a strong bisimulation relation, the fact that both states are strongly bisimilar also follows from the solitaire game depicted below, in which DUPLICATOR plays her winning strategy.



3.3. **Branching Bisimulation Games for Non-Divergent LTSs.** The first attempt to define two-player games for weak bisimulations was made by Bulychev *et al.* [6]. Their work defines a game for *(divergence blind) stuttering equivalence*, which is the counterpart of branching bisimulation for Kripke structures (which have state labels instead of edge labels). The following definition is a direct translation of the one from [6] to labelled transition systems. In fact, this definition weakens the strong bisimulation game in two ways: first, it allows a τ-transition to be mimicked by DUPLICATOR by not moving, and second, when SPOILER challenges with an a-transition, DUPLICATOR can reply by making a τ move. These are the first and third move for DUPLICATOR in the following definition. This game is limited to labelled transition systems in which there are no divergences (*i.e.*, no infinite sequences of τ-transitions are possible).

Definition 3.3. A *limited branching bisimulation (lbb) game* on an LTS L is played by players SPOILER and DUPLICATOR on an arena of SPOILER-owned configurations $[\,(s,t)\,]$ and DUPLICATOR-owned configurations $\langle\,(s,t),c\,\rangle$, where $(s,t) \in \mathit{Position}$ and $c \in \mathit{Challenge}$, and $\mathit{Position}$ and $\mathit{Challenge}$ are as before.

- SPOILER moves from a configuration $[\,(s,t)\,]$ by:
  (1) selecting $s \xrightarrow{a} s'$ and moving to $\langle\,(s,t),(a,s')\,\rangle$, or
  (2) selecting $t \xrightarrow{a} t'$ and moving to $\langle\,(t,s),(a,t')\,\rangle$.
- DUPLICATOR responds from a configuration $\langle\,(u,v),(a,u')\,\rangle$ by:
  (1) not moving if $a = \tau$ and continuing in configuration $[\,(u',v)\,]$, or
  (2) playing $v \xrightarrow{a} v'$ if available, and continuing in configuration $[\,(u',v')\,]$, or
  (3) moving $v \xrightarrow{\tau} v'$ if possible, and continuing in configuration $[\,(u,v')\,]$.

Finite plays are won by DUPLICATOR if and only if SPOILER gets stuck. All infinite plays are won by DUPLICATOR. We say that a configuration is won by a player when she has a strategy that wins all plays starting in it. Full plays of the game start in a configuration $[\,(s,t)\,]$; we say that DUPLICATOR wins the lbb-game for a position (s,t) if the configuration $[\,(s,t)\,]$ is won by her; in this case we write $s \equiv_{lb} t$. Otherwise we say that SPOILER wins that game.

By adapting Bulychev *et al.*'s proof of [6, Theorem 1] to LTSs and branching bisimulation, we can show that this game-based definition characterises branching bisimulation, provided the labelled transition system does not have divergences, *i.e.* is void of infinite τ-paths.

Theorem 3.4. *Let $L = \langle S, A, \rightarrow\rangle$ be a non-divergent LTS. We have for all $s,t \in S$ that $s \mathrel{\underline{\leftrightarrow}}_b t$ if and only if $s \equiv_{lb} t$.*

We conclude this section with an example showing that the definition does not characterise branching bisimilarity for arbitrary LTSs (*i.e.* those that exhibit divergence).

Example 3.5. Consider the following LTS, in which the states are labelled with their name.

$$\tau \circlearrowright t \xleftarrow{\;a\;} \overset{\downarrow}{s} \xrightarrow{\;a\;} u \xrightarrow{\;a\;} v$$

Observe that there is a divergence because of the τ-loop in t. Furthermore, t and u are not branching bisimilar since $u \xrightarrow{a} v$ can never be mimicked from t. However, DUPLICATOR can win the lbb-game starting in $[\,(t,u)\,]$ according to Definition 3.3 (regardless of SPOILER's moves). When starting in $[\,(t,u)\,]$, SPOILER can choose to play on u or on t. We distinguish these two cases:

- $[\,(t,u)\,] \to \langle\,(u,t),(a,v)\,\rangle$, then DUPLICATOR can respond with $\langle\,(u,t),(a,v)\,\rangle \to [\,(u,t)\,]$ by playing $t \xrightarrow{\tau} t$ according to her third move, or
- $[\,(t,u)\,] \to \langle\,(t,u),(\tau,t)\,\rangle$, then DUPLICATOR can respond with $\langle\,(t,u),(\tau,t)\,\rangle \to [\,(t,u)\,]$ by not moving according to her first move.

In both cases, the game ends up in a configuration for which we have shown DUPLICATOR can respond, and the game will continue indefinitely, hence DUPLICATOR wins, even though $t \not\leftrightarrow_b u$.

3.4. **Branching Bisimulation Games.** Since all vertices on a strongly connected component reachable through τ-transitions are branching bisimilar, any finite LTS can be preprocessed and turned into a non-divergent LTS. Such preprocessing is often part of a state space minimisation algorithm. However, for use cases such as debugging a specification or an implementation, it is desirable to avoid any preprocessing, and stay as close as possible to any user-provided specification. Furthermore, for infinite LTSs, such a preprocessing step is impossible altogether. In [9] we therefore investigated an alternative game-based characterisation for branching bisimulation. We introduce this alternative in this section.

First, let us take a closer look at Definition 3.3. Observe that, when SPOILER challenges DUPLICATOR by playing, *e.g.*, $s \xrightarrow{a} s'$, DUPLICATOR can get away with not ever playing an a-transition, by playing a τ-transition instead. Intuitively, this means that DUPLICATOR never truly answers to SPOILER's challenge, since this challenge is forgotten after she plays the τ-transition in her third move. The key idea in our branching bisimulation game is to preserve an unanswered challenge after DUPLICATOR's move. To facilitate this, we also include the challenge in SPOILER's configurations. Next, we also introduce a reward for DUPLICATOR that she earns whenever she actually matches SPOILER's challenge. The need for this reward is illustrated by the following example.

Example 3.6. Consider the first LTS depicted in Figure 1. Observe that s_0 and t_0 are branching bisimilar. Suppose SPOILER tries (in vain) to disprove that s_0 and t_0 are branching bisimilar and challenges DUPLICATOR by playing $s_0 \xrightarrow{a} c_1$. DUPLICATOR may respond with an infinite sequence of τ-steps, moving between t_0 and t_1, so long as SPOILER sticks to her challenge. In this way she would win the play following the rules in Definition 3.3, but such *procrastinating* behaviour of DUPLICATOR is not rewarded in our game. Instead, DUPLICATOR has to eventually move to c_1, matching the challenge, if she wants to win the play.

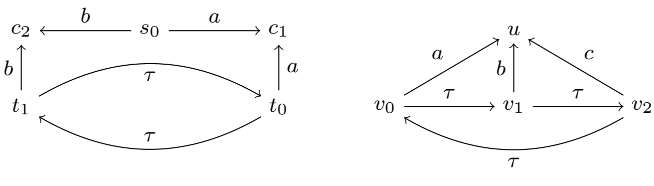


Figure 1: LTSs illustrating some consequences and subtleties of using challenges.

Definition 3.7. A *branching bisimulation (bb) game* on an LTS L is played by players SPOILER and DUPLICATOR on an arena of SPOILER-owned configurations $[\,(s,t),c,r\,]$ and DUPLICATOR-owned configurations $\langle\,(s,t),c,r\,\rangle$, where $((s,t),c,r) \in \mathit{Position} \times \mathit{Challenge}_{\dagger} \times$

Reward, and *Position* and *Challenge* are as before, $\mathit{Challenge}_\dagger = \mathit{Challenge} \cup \{\dagger\}$ is the set of challenges where $\dagger$ signifies absence of a challenge, and $\mathit{Reward} = \{*, \checkmark\}$ is a set of *rewards*. By convention, we write $((s,t),c,r)$ if we do not care about the owner of the configuration.

- SPOILER moves from a configuration $[\,(s,t),c,r\,]$ by:
  (1) selecting $s \xrightarrow{a} s'$ and moving to $\langle\,(s,t),(a,s'),*\,\rangle$ if $c = (a,s')$ or $c = \dagger$, and to $\langle\,(s,t),(a,s'),\checkmark\,\rangle$, otherwise; or
  (2) picking some $t \xrightarrow{a} t'$ and moving to $\langle\,(t,s),(a,t'),\checkmark\,\rangle$.

- DUPLICATOR responds from a configuration $\langle\,(u,v),(a,u'),r\,\rangle$ by:
  (1) not moving if $a = \tau$ and continuing in configuration $[\,(u',v),\dagger,\checkmark\,]$, or,
  (2) moving $v \xrightarrow{a} v'$ if available and continuing in configuration $[\,(u',v'),\dagger,\checkmark\,]$, or
  (3) moving $v \xrightarrow{\tau} v'$ if available and continuing in configuration $[\,(u,v'),(a,u'),*\,]$.

DUPLICATOR wins a finite play starting in a configuration $((s,t),c,r)$ if SPOILER gets stuck, and she wins an infinite play if the play yields infinitely many $\checkmark$ rewards. All other plays are won by SPOILER. We say that a configuration is won by a player when she has a strategy that wins all plays starting in it. *Full* plays of the game start in a configuration $[\,(s,t),\dagger,*\,]$; we say that DUPLICATOR wins the bb-game for a position (s,t), if the configuration $[\,(s,t),\dagger,*\,]$ is won by her; in this case, we write $s \equiv_b t$. Otherwise, we say that SPOILER wins that game.

Note that by definition both players strictly alternate their moves along plays.

We have not yet explained the rewards that are handed to DUPLICATOR in some of SPOILER's moves. The need for those are illustrated by the following example.

Example 3.8. Again consider the first LTS depicted in Figure 1. Suppose SPOILER tries to disprove (again in vain) that s_0 and t_0 are branching bisimilar, and challenges DUPLICATOR by playing $s_0 \xrightarrow{b} c_2$. The only response for DUPLICATOR is to move $t_0 \xrightarrow{\tau} t_1$, for which she gets a $*$ reward, and the pending challenge (b,c_2) is kept, generating the new configuration $[\,(s_0,t_0),(b,c_2),*\,]$. If SPOILER again plays $s_0 \xrightarrow{b} c_2$, DUPLICATOR can win by playing $t_1 \xrightarrow{b} c_2$. If SPOILER changes, and plays $s_0 \xrightarrow{a} c_1$, then DUPLICATOR can only respond with $t_1 \xrightarrow{\tau} t_0$, getting a $*$ reward and continuing from configuration $[\,(s_0,t_0),(a,c_1),*\,]$. SPOILER can continue switching between the moves $s_0 \xrightarrow{b} c_2$ and $s_0 \xrightarrow{a} c_1$, and DUPLICATOR will never be able to match the move directly. However, note that $\checkmark$s are only awarded whenever SPOILER switches away from his current challenge, and DUPLICATOR still wins this play. If we were not to award $\checkmark$s whenever SPOILER switches away from the current challenge, SPOILER would win the game starting in (s_0,t_0), whereas $s_0 \leftrightarrow_b t_0$.

Now consider the second LTS in Figure 1. In this LTS, $v_0 \leftrightarrow_b v_1 \leftrightarrow_b v_2$. Now, if the game starts in $[\,(v_1,v_0),\dagger,*\,]$, SPOILER can play to $\langle\,(v_0,v_1),(a,u),\checkmark\,\rangle$, to which DUPLICATOR can only respond by playing to $[\,(v_0,v_2),(a,u),*\,]$. This play can be extended, each time not leaving any choice for DUPLICATOR, as $[\,(v_0,v_2),(a,u),*\,] \to \langle\,(v_2,v_0),(c,u),\checkmark\,\rangle \to [\,(v_2,v_1),(c,u),*\,] \to \langle\,(v_1,v_2),(b,u),\checkmark\,\rangle \to [\,(v_1,v_0),(b,u),*\,] \to \langle\,(v_0,v_1),(a,u),\checkmark\,\rangle \to \cdots$. If we do not reward DUPLICATOR when SPOILER plays the second move, this play is winning for SPOILER, whereas $v_1 \leftrightarrow_b v_0$.

In [9] we proved that the bb-game captures branching bisimilarity. The result also follows from our results concerning the generic games in Section 5.

Theorem 3.9. *We have* $\leftrightarrow_b = \equiv_b$.

4. A Generic Bisimulation for Abstraction

Branching bisimilarity induces a weak semantics of systems; it achieves this by partially abstracting from τ moves so that they need not be matched one by one as is the case for strong bisimulation. A much more direct way of abstracting from τ moves is to modify strong bisimulation in such a way that each concrete transition $\xrightarrow{a}$ is 'weakly' matched using a weak transition $\xRightarrow{a}$, where the weak transition relation $\Longrightarrow$ is essentially $\twoheadrightarrow \circ \longrightarrow \circ \twoheadrightarrow$. The relation obtained this way is called *weak bisimulation*, which is the basis for weak bisimilarity.

As explained in detail in [32], the definition of branching bisimulation can be obtained from the definition of weak bisimulation by imposing additional conditions on the relation. By varying these conditions, two further behavioural equivalences can be obtained, *viz. delay bisimilarity* and *η-bisimilarity.* This is nicely visualised in [32] using the diagram of Figure 2.

$$\begin{tikzcd}
 & s \arrow[r, "a"] \arrow[dl, dotted, dash] \arrow[d, dotted, dash, "(1)"] & s' \arrow[d, dotted, dash, "(2)"'] \arrow[dr, dotted, dash] & \\
t \arrow[r, ->>] & t_1 \arrow[r, "a"] & t_2 \arrow[r, ->>] & t'
\end{tikzcd}$$

Figure 2: Weakly matching a concrete transition $s \xrightarrow{a} s'$ by an abstract transition from $t \xRightarrow{a} t'$. By requiring (1) and (2) to hold, or not, variations on weak bisimulations are obtained: η-bisimulation is obtained by requiring (1), delay bisimulation is obtained by requiring (2) and branching bisimulation is obtained by requiring both (1) and (2).

In this section we give a parametric bisimulation, which we refer to as (x, y)*-generic bisimulation*, that gives a unified definition for branching bisimulation and the three other abstract bisimulations. This definition in essence reflects the diagram of Figure 2: we use parameters x and y to indicate which extra conditions (if any) are imposed on the definition of weak bisimulation. Before we formally state our parametric bisimulation, we briefly recall the definitions of weak bisimulation, delay bisimulation and η-bisimulation and state a few well-known facts about them.

Definition 4.1 ([20])**.** A symmetric relation $R \subseteq S \times S$ is a *weak bisimulation* whenever $s \mathrel{R} t$ and $s \xrightarrow{a} s'$ imply either:

- $a = \tau$ and $s' \mathrel{R} t$, or
- there exist states t', t_1, t_2 such that $t \twoheadrightarrow t_1 \xrightarrow{a} t_2 \twoheadrightarrow t'$ and $s' \mathrel{R} t'$.

We write $s \leftrightarroweq_w t$ and say that s and t are weak bisimilar iff there is a weak bisimulation R such that $s \mathrel{R} t$. Typically we simply write $\leftrightarroweq_w$ to denote *weak bisimilarity.*[2]

Note that compared to branching bisimulation, weak bisimulation drops all conditions imposed on states reached silently before *and* after the execution of a weakly matching a-transition. The difference between the two relations is nevertheless subtle, as illustrated by the LTS depicted in Figure 3, taken from Korver [19]. It is not too hard to check that states 0 and 5 are weak bisimilar (and, in fact, also delay bisimilar). Nonetheless, the argument why these states are not branching bisimilar is more subtle and is best explained using the branching bisimulation game of the previous section, as we also indicated in [9]. Indeed,

[2]The definition in [20] uses the weak transition $s \xRightarrow{a} s'$ to denote $s \twoheadrightarrow s'' \xrightarrow{a} s''' \twoheadrightarrow s'$ in our notation.

$$
\begin{tikzcd}
1 & 0 \arrow[l, "a"'] \arrow[r, "b"] \arrow[d, "\tau"] & 4 \\
2 & 3 \arrow[l, "a"'] &
\end{tikzcd}
\qquad\qquad
\begin{tikzcd}
6 \arrow[d, "a"'] & 5 \arrow[l, "\tau"'] \arrow[r, "b"] & 8 \\
7 & &
\end{tikzcd}
$$

Figure 3: An illustration of the difference between weak bisimulation and branching bisimulation: states 0 and 5 are weak bisimilar but not branching bisimilar.

Spoiler wins the branching bisimulation game by moving $0 \xrightarrow{a} 1$. Duplicator can only respond to this challenge by moving $5 \xrightarrow{\tau} 6$. Now, continuing from $[\,(0,5),(a,1),*\,]$ Spoiler plays her second option and challenges Duplicator to mimic move $0 \xrightarrow{b} 4$, something that Duplicator cannot match.

Definition 4.2 ([21, 32])**.** A symmetric relation $R \subseteq S \times S$ is a *delay bisimulation* whenever $s\ R\ t$ and $s \xrightarrow{a} s'$ imply either:

- $a = \tau$ and $s'\ R\ t$, or
- there exist states t', t_1 such that $t \twoheadrightarrow t_1 \xrightarrow{a} t'$ and $s'\ R\ t'$.

We write $s \mathrel{\underline{\leftrightarrow}}_d t$ and say that s and t are delay bisimilar iff there is a delay bisimulation R such that $s\ R\ t$. Typically we simply write $\mathrel{\underline{\leftrightarrow}}_d$ to denote *delay bisimilarity.*

We remark that the original 'delay bisimulation' originates with Milner in [21], where it is called *observation equivalence*. It is not presented in a coinductive way but by means of a sequence of *approximations*. Moreover, the definition from [21] is sensitive to divergences, and coincides with our definition only for non-divergent systems. The fact that *weak transitions* where the execution of an observable action can be preceded, but not followed, by a finite sequence of internal actions, is indeed the seed for the–by now–'classical' notion of delay bisimulation that we present above, and which stems from [32].

Observe that delay bisimulation essentially renders the existentially quantified state t_2 in the definition of weak bisimulation superfluous as it requires it to coincide with t' in that definition. We note that from this it also immediately follows that every delay bisimulation is a weak bisimulation, but not *vice versa*.

Definition 4.3 ([2])**.** A symmetric relation $R \subseteq S \times S$ is an *η-bisimulation* whenever $s\ R\ t$ and $s \xrightarrow{a} s'$ imply either:

- $a = \tau$ and $s'\ R\ t$, or
- there exist states t', t_1, t_2 such that $t \twoheadrightarrow t_1 \xrightarrow{a} t_2 \twoheadrightarrow t'$, $s\ R\ t_1$ and $s'\ R\ t'$.

We write $s \mathrel{\underline{\leftrightarrow}}_e t$ and say that s and t are η-bisimilar iff there is an η-bisimulation R such that $s\ R\ t$. Typically we simply write $\mathrel{\underline{\leftrightarrow}}_e$ to denote *η-bisimilarity.*

As shown in [32], each of the relations we define above are equivalence relations.

Theorem 4.4 ([32])**.** *The relations $\mathrel{\underline{\leftrightarrow}}_e, \mathrel{\underline{\leftrightarrow}}_d$ and $\mathrel{\underline{\leftrightarrow}}_w$ are equivalence relations.*

From their definitions it immediately follows that the four equivalences $\mathrel{\underline{\leftrightarrow}}_b, \mathrel{\underline{\leftrightarrow}}_e, \mathrel{\underline{\leftrightarrow}}_d$ and $\mathrel{\underline{\leftrightarrow}}_w$, are ordered according to the lattice depicted in Figure 4, as was also shown in [25]. The equivalences lower in the lattice are coarser than the equivalences higher in the lattice.

We next exploit the similarity between the four abstract bisimulations by explicitly giving a parameterised definition from which the original definitions can be recovered. Before we state this parameterised definition, we introduce some auxiliary notation to facilitate a concise parametric definition.

$$
\begin{tikzcd}[column sep=small]
 & \underline{\leftrightarrow}_b \arrow[dl, dash] \arrow[dr, dash] & \\
\underline{\leftrightarrow}_d \arrow[dr, dash] & & \underline{\leftrightarrow}_e \arrow[dl, dash] \\
 & \underline{\leftrightarrow}_w &
\end{tikzcd}
$$

Figure 4: The lattice of abstract behavioural equivalences.

Let $R \subseteq S \times S$ and let $s, s', t \in S$. We write $s \twoheadrightarrow_{o,R,t} s'$ iff $s \twoheadrightarrow s'$, while we write $s \twoheadrightarrow_{b,R,t} s'$ iff $s \twoheadrightarrow s'$, $t\ R\ s$ and $t\ R\ s'$. In this way, when $x \in \{o, b\}$, a *general weak transition* $s \twoheadrightarrow_{x,R,t} s'$ is just a plain weak transition $s \twoheadrightarrow s'$, when $x = o$, while, when $x = b$, we additionally impose the 'context' conditions $t\ R\ s$ and $t\ R\ s'$. In a similar way, we write $s \Longrightarrow_{x,R,t} s'$ iff $s \twoheadrightarrow s'$ and either $x = o$, or $x = b$ and there is a finite sequence of transitions $s = s_0 \xrightarrow{\tau} s_1 \xrightarrow{\tau} \cdots \xrightarrow{\tau} s_n = s'$ such that for all i, $t\ R\ s_i$. It is clear that $s \Longrightarrow_{x,R,t} s'$ implies $s \twoheadrightarrow_{x,R,t} s'$, because we are not only imposing $t\ R\ s_0$ and $t\ R\ s_n$, but also that $t\ R\ s_i$ for any intermediate state. Instead, the implication from right to left generally does not hold, because those additional conditions for the intermediate states are not (explicitely) imposed by the definition of our general weak transitions.

Definition 4.5. For $x, y \in \{o, b\}$, a symmetric relation $R \subseteq S \times S$ is an (x, y)-*generic bisimulation*, whenever $s\ R\ t$ and $s \xrightarrow{a} s'$ imply either:

- $a = \tau$ and $s'\ R\ t$, or
- there exist states t', t_1, t_2 such that $t \twoheadrightarrow_{x,R,s} t_1 \xrightarrow{a} t_2 \twoheadrightarrow_{y,R,s'} t'$ and $s'\ R\ t'$.

Now, for the corresponding values of x and y, we write $s \underline{\leftrightarrow}_{(x,y)} t$ and say that s and t are (x, y)-generic bisimilar iff there is an (x, y)-generic bisimulation R such that $s\,R\,t$. Typically, we simply write $\underline{\leftrightarrow}_{(x,y)}$ to denote (x, y)-*generic bisimilarity.*

As formally claimed by the proposition below, the above parametric bisimulation definition captures all four abstract bisimulations.

Proposition 4.6. *We have the following correspondences for $R \subseteq S \times S$:*

- *R is a weak bisimulation iff it is an (o, o)-generic bisimulation;*
- *R is a delay bisimulation iff it is an (o, b)-generic bisimulation;*
- *R is an η bisimulation iff it is a (b, o)-generic bisimulation;*
- *R is a branching bisimulation iff it is a (b, b)-generic bisimulation.*

Proof. First observe that the definition of (o, o)-generic bisimulation coincides exactly with the definition of weak bisimulation. Likewise, the definition of (b, o)-generic bisimulation reduces to the definition of η-bisimulation.

As for the remaining two relations, we find that the implication from right to left follows because the requirements for being a (b, b)-generic bisimulation (resp. an (o, b)-generic bisimulation) are stronger than the requirements for being a branching bisimulation (resp. a delay bisimulation). For the implication from left to right we note that for any delay bisimulation (resp. a branching bisimulation) R, then R is an (o, b)-generic bisimulation (resp. a (b, b)-generic bisimulation) by simply choosing $t_2 = t'$. $\square$

It follows from the above theorem that, by equipping the set $\{o, b\}$ with an ordering $\leq$, where $x \leq x$ for all x and $o < b$, and lifting this ordering to pairs taken from $\{o, b\} \times \{o, b\}$

we can recover the lattice of Figure 4. We furthermore note that in view of Proposition 4.6 and Theorems 2.4 and 4.4 we have the following corollary.

Corollary 4.7. *Let $x, y \in \{o, b\}$. Then $\leftrightarroweq_{(x,y)}$ is an equivalence relation.*

Additionally, it is not hard to prove that (x, y)-generic bisimilarity, like branching bisimilarity, satisfies the stuttering property.

Lemma 4.8. *Let $x, y \in \{o, b\}$. Then $\leftrightarroweq_{(x,y)}$ satisfies the stuttering property (Definition 2.5).*

Proof. The proof is essentially a translation to the general case of that given in [25, Lemma 4.9.2], for the particular case of branching bisimulation. Let $t_0 \xrightarrow{\tau} t_1 \cdots \xrightarrow{\tau} t_k$ with $t_0 \leftrightarroweq_{(x,y)} t_k$. We define relation R as follows:

$$R = \{(t_0, t_i), (t_i, t_0) \mid 0 \leq i < k\} \cup \leftrightarroweq_{(x,y)}$$

Let us see that R is an (x, y)-generic bisimulation relation, by proving the transfer condition for all pairs $t_0 \; R \; t_i$.

- When we have $t_i \xrightarrow{a} t_i'$, we immediately obtain $t_0 \twoheadrightarrow_{x,R,t_i} t_i \xrightarrow{a} t_i' \twoheadrightarrow_{y,R,t_i'} t_i'$, which is trivially true because $t_i \; R \; t_i$. Moreover, it is equally trivial that $t_i' \; R \; t_i'$.
- When we have $t_0 \xrightarrow{a} t_0'$, from $t_0 \leftrightarroweq_{(x,y)} t_k$ we infer that we have either:
  – $a = \tau$ and $t_0' \; R \; t_k$, or
  – there exist states t_k', t_k'', t_k''' such that $t_k \twoheadrightarrow_{x,R,t_0} t_k' \xrightarrow{a} t_k'' \twoheadrightarrow_{y,R,t_0'} t_k'''$ and $t_0' \; R \; t_k'''$.
  In both cases we can start the matching computation from t_i by means of the weak transition $t_i \twoheadrightarrow t_k$, thus getting either:
  – $a = \tau$ and we have $t_i \twoheadrightarrow_{x,R,t_0} t_{k-1} \xrightarrow{a} t_k \twoheadrightarrow_{y,R,t_0'} t_k$, because $t_0 \; R \; t_{k-1}$ and $t_0' \; R \; t_k$, or
  – there exist states t_k', t_k'', t_k''' such that $t_i \twoheadrightarrow_{x,R,t_0} t_k' \xrightarrow{a} t_k'' \twoheadrightarrow_{y,R,t_0'} t_k'''$ and $t_0' \; R \; t_k'''$,
  thus concluding the proof. □

Using the stuttering property, we find that we may rephrase (x, y)-generic bisimilarity as per the following theorem. Essentially, this says we may require the intermediate states along the stuttering paths of the transfer condition to be related without changing the definition.

Theorem 4.9. *For every $x, y \in \{o, b\}$ and $\bar{s}, \bar{t} \in S$, we have $\bar{s} \leftrightarroweq_{(x,y)} \bar{t}$ iff $\bar{s} \; R \; \bar{t}$ for some symmetric relation $R \subseteq S \times S$ satisfying that whenever $s \; R \; t$ and $s \xrightarrow{a} s'$ we have:*

- *$a = \tau$ and $s' \; R \; t$, or*
- *there exist states t', t_1, t_2 such that $t \Longrightarrow_{x,R,s} t_1 \xrightarrow{a} t_2 \Longrightarrow_{y,R,s'} t'$ and $s' \; R \; t'$.*

Proof. Immediate. Even if the original definition of the parameterised bisimulations used $\twoheadrightarrow_{x,R,s}$ and $\twoheadrightarrow_{y,R,s'}$ instead of $\Longrightarrow_{x,R,s}$ and $\Longrightarrow_{y,R,s'}$, the additional conditions added by the latter relations follow from those imposed by the former ones per Lemma 4.8. □

As a consequence of the above theorem, we may henceforth and without loss of generality, use a stronger definition for (x, y)-generic bisimilarity than the one stated in Definition 4.5, *viz.* the one implied by Theorem 4.9.

5. Generic Bisimulation Games

As we illustrated in the previous section, branching bisimulation is one of four bisimulations for providing a weak semantics to systems. Rather than defining a dedicated game for each of these bisimulations, we show that these four relations are captured by instances of a generic bisimulation game, which we introduce in Section 5.1. In particular, in Section 5.2 we show that this game is sound and complete for (x, y)-generic bisimilarity. Moreover, in Section 5.3 we shall see that the branching bisimulation game that we defined in Section 3 specialises our generic bisimulation game. Finally, in Section 5.4 we discuss two variations on the game for characterising (x, y)-generic bisimilarity.

5.1. A Generic Bisimulation Game.

As we explained in Section 3, Spoiler's and Duplicator's role in the games we consider is to *refute* and *prove*, respectively, that states in an LTS can be related. In general, given some game characterising a relation on states, we can obtain a weaker relation by either restricting Spoiler's capabilities, or by offering more liberal options to Duplicator. The game we present in this section essentially does the latter: it will—depending on which relation we wish to capture—give Duplicator fewer or more options to respond to Spoiler's challenges. Before we move to defining our generic

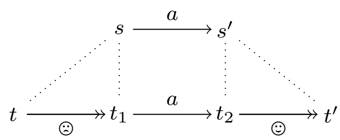


Figure 5: Diagram that illustrates the different scenarios that need to be considered when 'weakly' matching a challenge by Spoiler

bisimulation game, consider the diagram above (see also Figure 2). It illustrates that a 'challenge' $s \xrightarrow{a} s'$, posed by Spoiler, is to be matched 'weakly' through $t \twoheadrightarrow t_1 \xrightarrow{a} t_2 \twoheadrightarrow t'$. Intuitively, we can capture this weak match by recording the progress made by Duplicator using a 'pebble' which Duplicator can push along the τ or a-edges in the transition system. Moreover, we need to record whether Duplicator is pushing the pebble along the '☹-decorated' τ-path or the '☺-decorated' τ-path if we wish to prevent her from meeting Spoiler's challenge by taking multiple (or no) a-transitions. The relation one intuitively obtains in this way characterises weak bisimilarity.

On the other hand, we must disallow Duplicator to merely push the pebble when we wish to characterise branching bisimilarity. Instead, she must move to a configuration with an updated position so that Spoiler can renew her challenge for this new position. Indeed, since branching bisimilarity has the stuttering property, Duplicator should be able to move along τ-paths in which every state is related to the state from which Spoiler posed her challenge. A similar observation can be made for η-bisimilarity: in this case, Duplicator must update her position while traversing the τ-path to the state from which a matching a-action can be performed, while, after the match, she may move 'freely' by pushing the pebble to a state that matches Spoiler's target state. In delay bisimilarity, Duplicator can first push the pebble along the ☹-decorated τ-path to a state from which a matching action can be performed, but she must update the position immediately upon choosing this matching transition.

This means that we can tweak Duplicator's powers by enabling or disabling those rules that would allow her to merely push pebbles. That is, in case of branching bisimilarity, she is never allowed to use such rules, whereas for weak bisimilarity, she can always employ such rules. For the other two relations, we can disable the rules associated with pushing the pebble essentially by relying on whether Duplicator is on a $\frownie$ or $\smiley$ τ-path. Below, we make these ideas more precise.

Definition 5.1. For each $E \subseteq \{\smiley, \frownie\}$, a *generic bisimulation (gb) game* (also noted as E-gb game when we want to stress the corresponding set E) on an LTS L is played by players Spoiler and Duplicator on an arena of Spoiler-owned configurations $[\,(s,t), c, m, r\,]$ and Duplicator-owned configurations $\langle\,(s,t), c, m, r\,\rangle$, where $(s,t) \in \mathit{Position}$, $c \in \mathit{Challenge}_\dagger$, $m \in \mathit{Match}_\dagger$ and $r \in \mathit{Reward}$, and $\mathit{Position}$, $\mathit{Challenge}_\dagger$ and Reward are as before, and $\mathit{Match}_\dagger = (S \times \{\frownie, \smiley\}) \cup \{\dagger\}$ is the set of *partial matches*. We write $((s,t), c, m, r)$ if we do not care about the owner of the configuration. Spoiler's and Duplicator's possible moves are given by the following rules:

- From a configuration $[\,(s,t), c, m, r\,]$, Spoiler can:
  (1) move to $\langle\,(s,t), c, m, *\,\rangle$ if $c \neq \dagger$, or
  (2) for some $s \xrightarrow{a} s'$, move to either:
    (a) $\langle\,(s,t), (a,s'), (t,\frownie), *\,\rangle$, if $c = \dagger$, or
    (b) $\langle\,(s,t), (a,s'), (t,\frownie), \checkmark\,\rangle$, if $c \neq (a,s')$
  (3) for some $t \xrightarrow{a} t'$, move to: $\langle\,(t,s), (a,t'), (s,\frownie), \checkmark\,\rangle$.

- From a configuration $\langle\,(u,v), (a,u'), (\bar{v}, f), r\,\rangle$, Duplicator can:
  (1) move to $[\,(u', \bar{v}), \dagger, \dagger, \checkmark\,]$ when $a = \tau$, or
  (2) if $f = \frownie$ and $\bar{v} \xrightarrow{a} v'$, move to one of the following:
    (a) $[\,(u',v'), (a,u'), (v',\smiley), *\,]$, in any case, or
    (b) $[\,(u',v'), \dagger, \dagger, \checkmark\,]$, in any case, or
    (c) $[\,(u,v), (a,u'), (v',\smiley), *\,]$, only if $\smiley \in E$
  (3) for some $\bar{v} \xrightarrow{\tau} v'$, move to one of the following:
    (a) $[\,(u,v'), (a,u'), (v',f), *\,]$, in any case, or
    (b) $[\,(u',v'), \dagger, \dagger, \checkmark\,]$, only if $f = \smiley$
    (c) $[\,(u,v), (a,u'), (v',f), *\,]$, only if $f \in E$

Duplicator wins a finite play starting in a configuration $((s,t), c, m, r)$ if Spoiler gets stuck, and she wins an infinite play if the play yields infinitely many $\checkmark$ rewards. All other plays are won by Spoiler. A player wins a configuration when she has a strategy that wins all plays starting in it. *Full* plays of the game start in a configuration $[\,(s,t), \dagger, \dagger, *\,]$; we say that Duplicator wins the game for a position (s,t), if the configuration $[\,(s,t), \dagger, \dagger, *\,]$ is won by her. In this case, we write $s \equiv_E t$. Otherwise, we say that Spoiler wins that game.

Note that the moves in the game, like in the branching bisimulation game of Definition 3.7, alternate between Spoiler and Duplicator. Spoiler typically issues a challenge, which, in case of rule (1) consists of a previously coined challenge, in case of rule (2a) is a fresh challenge, in case of rule (2b) overrides a pre-existing challenge, and in case of rule (3), is a fresh challenge but issued from the second state in the position of the configuration. Intuitively, when Duplicator is to respond from a configuration, she will try to 'weakly' match the challenge using any of her rules. Using rule (1), she can decide to instantly meet a τ-challenge and leave the task of disproving that the states in the resulting position are not related, to Spoiler. Alternatively, Duplicator can also face a challenge by making (at least) one (explicit) move. The actual matching (or cancelling) of Spoiler's challenge is captured in rules (2a)-(2c). Additionally, Duplicator can use rules (3a)-(3c) to 'push' the pebble from the position on the partial match closer to a state in which she can finally match the challenge as above (when the pebble is still on a ☹-decorated path in terms of Figure 5), or she can continue to 'push' the pebble closer to the final position where she can finally cancel the challenge. Note that in rules (2a), (2c), (3a), and (3c) the state v' at the match reflects the progress of Spoiler until the completion of the matching that cancels it. The matching is completed by updating the position in Duplicator's rules (1), (2b), and (3b).

Let the set $E(x, y)$ be the smallest set such that ☹ $\in E(o, y)$ and ☺ $\in E(x, o)$ for all $x, y \in \{o, b\}$. Using this notation, we can instantiate our gb-games to games for the four instances of (x, y)-generic bisimulation. That is, we have:

- the branching bisimulation game, when $E(x, y) = \emptyset$
- the delay bisimulation game, when $E(x, y) = \{$☹$\}$
- the η-bisimulation game, when $E(x, y) = \{$☺$\}$
- the weak bisimulation game, when $E(x, y) = \{$☺, ☹$\}$.

Therefore, the *faces* ☹ and ☺, included in each of the sets $E(x, y)$, state exactly the degree of freedom of the corresponding bisimulation relations. They govern *when* (before and/or after the matching a actions) we do not have new bisimulation obligations that would generate new *positions* to check. That means that the *fewer* faces we have, the *finer* the bisimulation relation we get.

To be exact, our generic game could be presented as a collection of (four) games in a single family. The 'unity' of this family is supported by the single (parameterised) definition of its elements. This allows us to claim the following correspondence between our gb-game and (x, y)-generic bisimilarity.

Theorem 5.2. *For all $x, y \in \{o, b\}$, we have $s \leftrightarroweq_{(x,y)} t$ iff $s \equiv_{E(x,y)} t$.*

The proof of this claim is discussed in detail in the next section. We first illustrate the definition using a small example.

Example 5.3. Consider the transition system depicted to the left below. Note that the transition system is essentially the result of merging all deadlocking and 'determined' states of the two transition systems depicted in Figure 3. That is, state A represents state 0 of Figure 3, state B represents the deadlock states 1, 2, 4, 7 and 8 of Figure 3, state C represents state 5 of Figure 3 and state D represents states 3 and 6 of Figure 3.

$$\begin{tikzcd}
\mathrm{A} \arrow[r, bend left, "a"] \arrow[r, bend right, "b"'] \arrow[dr, bend right, "\tau"'] & \mathrm{B} & \mathrm{C} \arrow[l, "b"'] \arrow[dl, bend left, "\tau"] \\
& \mathrm{D} \arrow[u, "a"'] &
\end{tikzcd}$$

$$\begin{tikzcd}
{} \arrow[r] & {[\,(A,C),\dagger,\dagger,*\,]} \arrow[r, "2\mathrm{a}"] & {\langle\,(A,C),(a,B),(C,☹),*\,\rangle} \arrow[d, "3\mathrm{a}"] \\
& {\langle\,(A,D),(b,B),(D,☹),\checkmark\,\rangle} & {[\,(A,D),(a,B),(D,☹),*\,]} \arrow[l, "2\mathrm{b}"']
\end{tikzcd}$$

The states A and C are delay bisimilar and weak bisimilar, but they are not branching bisimilar or η-bisimilar as we explained earlier. Spoiler's strategy to win the branching bisimulation game, which we explained on page 11, can be replayed in the gb-game, as illustrated by the (solitaire) game graph depicted next to the transition system: for Spoiler-owned configurations (marked grey), Spoiler uses her rule (2a), followed by rule (2b), and

DUPLICATOR can only use rule (3a). Notice that SPOILER's strategy is winning in both the $\equiv_{E(b,b)}$ game and the $\equiv_{E(b,o)}$ game; in both cases, the same (solitaire) game graph is obtained.

We elaborate on DUPLICATOR's winning strategy in the $\equiv_{E(o,o)}$ game in Example 5.6 on page 19.

5.2. **Soundness and Completeness.** For our completeness result, we essentially need to show that any pair of states s, t related through a generic bisimulation, yields a configuration $[\,(s,t), \dagger, \dagger, *\,]$ in our generic bisimulation game that is won by DUPLICATOR. A play generally passes through configurations that have a challenge and partial match different from $\dagger$, so in our proof of completeness, we must deal with such configurations, too. More specifically, our completeness proof deals with configurations of a particular shape; we call such configurations *good*.

Definition 5.4. Let $((s,t), c, m, r)$ be an arbitrary configuration in a gb-game. We say this configuration is (x,y)-*good* with respect to a given relation $R \subseteq S \times S$ iff $s\ R\ t$, $m = \dagger$ implies $c = \dagger$, and for $c = (a, s')$ and $m = (t_1, f)$, either $a = \tau$ and $s'\ R\ t_1$, or there is some t' such that $s'\ R\ t'$ and:

- if $f = ☺$ then $t_1 \Longrightarrow_{y,R,s'} t'$;
- if $f = ☹$ then there are t_2, t_3 so that $t_1 \Longrightarrow_{x,R,s} t_2 \xrightarrow{a} t_3 \Longrightarrow_{y,R,s'} t'$.

When the relation R and both x and y are clear from the context, we simply write that a configuration is *good* rather than (x,y)-*good* for R.

Lemma 5.5 (Completeness). *For all $x, y \in \{o, b\}$, whenever we have $s \leftrightarrow_{(x,y)} t$, we also have $s \equiv_{E(x,y)} t$.*

Proof. We design a partial strategy for DUPLICATOR and show that this strategy is winning for configurations that are *good* with respect to $\leftrightarrow_{(x,y)}$. Since $[\,(s,t), \dagger, \dagger, *\,]$ is *good* follows from $s \leftrightarrow_{(x,y)} t$, it follows that DUPLICATOR wins the $\equiv_{E(x,y)}$ game and we thus have $s \equiv_{E(x,y)} t$.

Fix $x, y \in \{o, b\}$. For convenience, we write R for the relation $\leftrightarrow_{(x,y)}$. We first show that from every configuration that is *good* with respect to R, SPOILER can only move to a configuration that is again *good*. Suppose $[\,(s,t), c, m, r\,]$ is such a configuration. Then SPOILER may play such that the play will continue in either:

- $\langle\,(s,t), c, m, *\,\rangle$ if $c \neq \dagger$ and SPOILER passes on the configuration unchanged using rule (1);
- $\langle\,(s,t), (a,s'), (t, ☹), *\,\rangle$ for some $s \xrightarrow{a} s'$ if $c = \dagger$ when SPOILER uses rule (2a);
- $\langle\,(s,t), (a,s'), (t, ☹), \checkmark\,\rangle$ for some $s \xrightarrow{a} s'$ for which $c \neq (a, s')$ when SPOILER uses rule (2b);
- $\langle\,(t,s), (a,t'), (s, ☹), \checkmark\,\rangle$ for some $t \xrightarrow{a} t'$ if SPOILER uses rule (3).

In the first case, the fact that $\langle\,(s,t), c, m, r\,\rangle$ is *good* follows from the fact that $[\,(s,t), c, m, r\,]$ is *good*. The configurations of the remaining three cases can be seen to be *good* by an immediate application of the conditions stating when two states are related by an (x,y)-generic bisimulation, as a consequence of the fact that $s\ R\ t$ is imposed by definition of *good* configurations. Recall that R here is (the largest) (x,y)-generic bisimulation. Moreover, note that every DUPLICATOR-owned configuration reached by SPOILER—when starting from a configuration that is *good*—has a non-$\dagger$ challenge; consequently, such configurations—being *good*—also carry a non-$\dagger$ partial match.

We next focus on configurations of the form $\langle\,(s,t),(a,s'),(\bar{t},f),r\,\rangle$ that are *good* and we argue that Duplicator can always move to a Spoiler-owned configuration that is again *good*. Let $\langle\,(s,t),(a,s'),(\bar{t},f),r\,\rangle$ be such a configuration. We distinguish three (mutually exclusive) cases.

(1) Suppose $a = \tau$ and $s' \mathrel{R} \bar{t}$. Then, using rule (1), Duplicator plays to configuration $[\,(s',\bar{t}),\dagger,\dagger,\checkmark\,]$. Clearly this configuration is *good*.

(2) Case $f = ☹$, and $a \neq \tau$ or not $s' \mathrel{R} \bar{t}$. Since $\langle\,(s,t),(a,s'),(\bar{t},f),r\,\rangle$ is *good* and $f = ☹$, there are t', t_2, t_3 so that $\bar{t} \Longrightarrow_{x,R,s} t_2 \xrightarrow{a} t_3 \Longrightarrow_{y,R,s'} t'$ and $s' \mathrel{R} t'$. Fix states t', t_2 and t_3 on a shortest path witnessing these properties. We distinguish three further cases:
   - Case $t_2 = \bar{t}$ but $t_3 \neq t'$. If $☺ \notin E(x,y)$ then Duplicator plays to configuration $[\,(s',t_3),(a,s'),(t_3,☺),*\,]$ using rule (2a); otherwise she moves to configuration $[\,(s,t),(a,s'),(t_3,☺),*\,]$ using rule (2c). Both configurations are *good*.
   - Case $t_2 = \bar{t}$ and $t_3 = t'$. Then Duplicator plays to configuration $[\,(s',t_3),\dagger,\dagger,\checkmark\,]$ using rule (2b), which is again *good*.
   - Case $t_2 \neq \bar{t}$. We consider the set of paths underlying $\bar{t} \Longrightarrow_{x,R,s} t_2$, and in particular the *shortest* paths in this set. Now we take as t^* the τ-successor of $\bar{t}$ in any of these shortest paths. Duplicator then can play to configuration $[\,(s,t^*),(a,s'),(t^*,☹),*\,]$ if $☹ \notin E(x,y)$, using rule (3a), and to configuration $[\,(s,t),(a,s'),(t^*,☹),*\,]$, otherwise using rule (3c). Again, both configurations are *good*.

(3) Case $f = ☺$ and not $s' \mathrel{R} \bar{t}$. Since $\langle\,(s,t),(a,s'),(\bar{t},f),r\,\rangle$ is *good* and $f = ☺$, there is some t' such that $\bar{t} \Longrightarrow_{y,R,s'} t'$ and $s' \mathrel{R} t'$. Consider a t' closest to $\bar{t}$ (with respect to the lengths of the paths underlying $\Longrightarrow_{y,R,s'}$) with the property $s' \mathrel{R} t'$. At this point, we can draw two conclusions:
   - $y = o$ since $y = b$ and $\bar{t} \Longrightarrow_{y,R,s'} t'$ contradicts not $s' \mathrel{R} \bar{t}$;
   - $\bar{t} \neq t'$ since $s' \mathrel{R} t'$ but not $s' \mathrel{R} \bar{t}$.

   From the first, it follows that $☺ \in E(x,y)$ so Duplicator may play either rule (3a) or rule (3b). From the second, it follows that $\bar{t} \Longrightarrow_{y,R,s'} t'$ reduces to $\bar{t} \twoheadrightarrow t'$ and since $\bar{t} \neq t'$ there must be some immediate τ-successor t^* of $\bar{t}$ on the shortest τ-path from $\bar{t}$ to t'. Then Duplicator plays to configuration $[\,(s,t),(a,s'),(t^*,☺),*\,]$ if this $t^* \neq t'$, and configuration $[\,(s',t'),\dagger,\dagger,\checkmark\,]$ otherwise. We observe that both configurations are again *good*.

Since in all cases, Duplicator can move to a configuration that is again *good*, it follows that if Spoiler starts in a configuration that is *good*, no play can pass along configurations that are not *good*. Moreover, Duplicator never gets stuck playing her strategy.

Finally, we argue that Duplicator wins all plays starting in configurations that are *good*. Let Π be the set of all plays consistent with Duplicator's strategy that start in some Spoiler-owned configuration that is *good*. Since on plays in Π, Duplicator never gets stuck, all finite plays in Π are won by Duplicator. It thus suffices to prove that Duplicator wins all infinite plays in Π.

Towards a contradiction, assume that $\pi \in \Pi$ is a play won by Spoiler. That means that there are only finitely many configurations on π in which Duplicator earns a $\checkmark$. Consequently, Duplicator plays rules (1), (2b) or (3b) only finitely often. Furthermore, there are only finitely many configurations on π with a non-$\dagger$ challenge. This can be seen as follows: only Spoiler-owned configurations can have $\dagger$ challenges and Duplicator only produces a $\dagger$ challenge by applying one of the rules (1), (2b) or (3b), which she does only finitely often. Thus, π must have an infinite suffix without $\checkmark$ rewards and only non-$\dagger$

challenges. On this suffix, SPOILER's only moves consist of passing on the configuration unchanged to DUPLICATOR.

Consider the i-th DUPLICATOR-owned configuration $\langle\, (s_i, t_i), (a, s'), (\bar{t}_i, f_i), * \,\rangle$ on this suffix. Since this configuration is *good*, there must be some t' such that $s' \mathrel{R} t'$ and there must be some finite path from $\bar{t}_i$ to t'. Now DUPLICATOR can simply follow this path, as it was done in cases 2 and 3 in the analysis we conducted to prove that goodness can be preserved by DUPLICATOR. In particular, DUPLICATOR will apply either rule (2b) or (3b) when t' is finally reached, but in such a case she earns a ✓, which means a contradiction. So π is won by DUPLICATOR. □

We illustrate DUPLICATOR's winning strategy as constructed in the proof above using a small example.

Example 5.6. Reconsider the states A and C of Example 5.3, depicted again in Figure 6 for convenience. As we noted before, these states are not branching bisimilar, but they are weak bisimilar, and, hence, they are (o, o)-bisimilar. Following the strategy explained in the proof of Lemma 5.5, this is illustrated by the $E(o, o)$-bisimulation subgame starting in $[\,(A, C), \dagger, \dagger, *\,]$, depicted in Figure 6 when DUPLICATOR plays according to the aforementioned strategy. Note also that SPOILER can only employ move 2a in those configurations in which there is no pending challenge. She can, however, at all times change her challenge by playing either move 2b or move 3.

We next focus on the soundness of our game; that is, we will show that DUPLICATOR cannot win configurations $[\,(s, t), \dagger, \dagger, *\,]$ for states s and t that are not (x, y)-generic bisimilar. Before we give a formal proof of soundness, we first state two useful observations concerning the gb-games.

Proposition 5.7. *Configurations* $[\,(s, t), c, m, *\,]$ *and* $[\,(s, t), c, m, \checkmark\,]$ *are both won by the same player. So are* $\langle\, (s, t), c, m, * \,\rangle$ *and* $\langle\, (s, t), c, m, \checkmark \,\rangle$*.*

Proof. This follows immediately from the Büchi winning condition: any player that wins some suffix of an infinite play also wins the infinite play itself. Furthermore, note that neither SPOILER nor DUPLICATOR can get stuck playing a game by changing a reward from $*$ to ✓ or *vice versa.* □

Proposition 5.8. *If* DUPLICATOR *wins a configuration* $[\,(s, t), c, m, r\,]$*, then she also wins the configuration* $[\,(s, t), \dagger, \dagger, \checkmark\,]$*.*

Proof. Let $[\,(s, t), c, m, r\,]$ be a SPOILER-owned consistent configuration that is won by DUPLICATOR. We simply observe that any configuration reached by a move by SPOILER from the configuration $[\,(s, t), \dagger, \dagger, \checkmark\,]$, can also be reached from the given $[\,(s, t), c, m, r\,]$ up to possibly the value of r. Next we simply apply Proposition 5.7 to conclude the result. □

Remark 5.9. The converse of the result above does not hold in general. To illustrate this, consider a transition system with states s, s' and one transition $s \xrightarrow{a} s'$, where $a \neq \tau$. Clearly, DUPLICATOR wins $[\,(s, s), \dagger, \dagger, \checkmark\,]$ but she does not, for instance, win $[\,(s, s), (a, s'), (s', ☹), *\,]$.

Note that as a consequence of Proposition 5.8, DUPLICATOR's rule (2a) is redundant. That is, a game that lacks rule (2a) is equivalent (from the point of view of $\equiv_E$) to the game we define in Definition 5.1; we briefly return to this matter in Section 5.3. Although our game contains some redundancy, we believe that our presentation of the game better reflects the diagram and intuition of Figure 2.

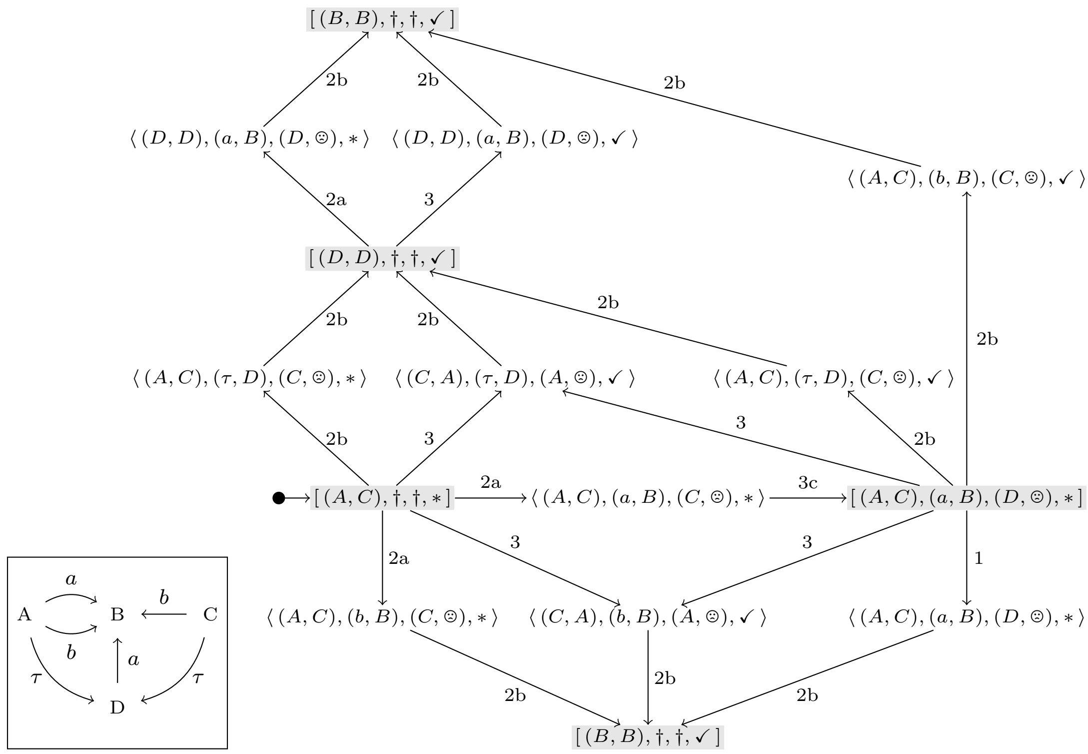


Figure 6: The (solitaire) game underlying $A \equiv_{E(o,o)} C$ in which Duplicator plays according to the (memoryless) winning strategy of Lemma 5.5. Spoiler-owned vertices are marked grey, Duplicator-owned vertices are not marked, and edges are annotated with the rule Spoiler or Duplicator used to move to the next configuration. The 'starting configuration' of the game is indicated by the $\bullet\!\!\rightarrow$ symbol. For quick reference, the Labelled Transition System of Figure 3 serving as the basis for this game is depicted in the below left corner.

Lemma 5.10 (Soundness). *For all $x, y \in \{o, b\}$, the relation $\equiv_{E(x,y)}$ is an (x, y)-generic bisimulation.*

Proof. We proceed as follows: we construct a relation $R \subseteq S \times S$ such that all pairs of states related by R represent some Spoiler-owned configuration that is won by Duplicator. Next, we show that this relation is an (x, y)-generic bisimulation.

Fix some $x, y \in \{o, b\}$ and define $R \subseteq S \times S$ as follows:

$$R = \{(s, t) \mid \text{Duplicator wins } [\,(s, t), \dagger, \dagger, *\,]\}$$

We show that R meets the conditions of Definition 4.5. First observe that symmetry of R follows from Proposition 5.7 and the fact that starting from a configuration $[\,(s, t), \dagger, \dagger, *\,]$, Spoiler has the same options as when starting from configuration $[\,(t, s), \dagger, \dagger, *\,]$.

Suppose $s \mathrel{R} t$, and assume that $s \xrightarrow{a} s'$ holds. It suffices to prove that R meets the transfer condition, *i.e.* either:

T1 $a = \tau$ and $s' \mathrel{R} t$, or

T2 $t \twoheadrightarrow_{x,R,s} t_1 \xrightarrow{a} t_2 \twoheadrightarrow_{y,R,s'} t'$ for some t_1, t_2, t' such that $s' \mathrel{R} t'$.

Since $s\ R\ t$, we know that DUPLICATOR wins $[\,(s,t),\dagger,\dagger,*\,]$. Fix a strategy ϱ for DUPLICATOR that is winning from this configuration. Now, consider a play π that emerges by DUPLICATOR playing according to ϱ and SPOILER using her rule (2a) to move from $[\,(s,t),\dagger,\dagger,*\,]$ to $\langle\,(s,t),(a,s'),(t,☹),*\,\rangle$ and her rule (1) on all subsequent non-$\dagger$ configurations. Note that, by assumption, $s \xrightarrow{a} s'$ holds, so SPOILER's rule (2a) is applicable, and a play π of the required form therefore is guaranteed to exist.

We will show that π contains all information necessary for concluding that either (T1) or (T2) holds. Notice that since DUPLICATOR wins π, the play π must have a prefix π' of the following form (for some $n \geq 0$ and appropriately chosen $s_i, t_i, \bar{t}_i, f_i$ and t'):

$$\begin{array}{rl} \pi' = & [\,(s,t),\dagger,\dagger,*\,] \\ & \langle\,(s_0,t_0),(a,s'),(\bar{t}_0,f_0),*\,\rangle \\ & [\,(s_1,t_1),(a,s'),(\bar{t}_1,f_1),*\,] \\ & \cdots \\ & \langle\,(s_n,t_n),(a,s'),(\bar{t}_n,f_n),*\,\rangle \\ & [\,(s',t'),\dagger,\dagger,\checkmark\,] \end{array}$$

where either $t' = \bar{t}_n$, or t' is an immediate τ-successor or a-successor of $\bar{t}_n$. Moreover, for all odd i, we have $\bar{t}_i = \bar{t}_{i+1}$ and, for even $i < n-1$, either $\bar{t}_i \xrightarrow{\tau} \bar{t}_{i+2}$ or $\bar{t}_i \xrightarrow{a} \bar{t}_{i+2}$. Fix this prefix π'. Note that $[\,(s',t'),\dagger,\dagger,\checkmark\,]$ is again won by DUPLICATOR, and by Proposition 5.7 and by definition of R we find $s'\ R\ t'$. We next distinguish cases, essentially based on the length of π'.

In case $n = 0$, DUPLICATOR must have either used rule (1) or rule (2b). If DUPLICATOR used rule (1), then $a = \tau$ and $t' = \bar{t}_0 = t$, and thus R meets transfer condition (T1). When DUPLICATOR used rule (2b), then we must have had $t = \bar{t}_0 \xrightarrow{a} t'$, in which case R meets transfer condition (T2).

In case $n > 0$, we find $\bar{t}_0 \twoheadrightarrow \bar{t}_{k-1} \xrightarrow{a} \bar{t}_k \twoheadrightarrow t'$ for some k so it suffices to prove that also $\bar{t}_0 \twoheadrightarrow_{x,R,s} \bar{t}_{k-1} \xrightarrow{a} \bar{t}_k \twoheadrightarrow_{y,R,s'} t'$. We distinguish two cases:

(1) For all $i \leq n$, we have $f_i = ☹$ in π'. In that case, we find that DUPLICATOR only played according to rules (3a) or (3c) for all i-th DUPLICATOR owned configurations ($i < n$) and rule (2b) for its n-th configuration. Consider the sequence of transitions $t = \bar{t}_0 \xrightarrow{\tau} \bar{t}_2 \xrightarrow{\tau} \cdots \xrightarrow{\tau} \bar{t}_n \xrightarrow{a} t'$.
   - Case $x = o$. Then we can immediately conclude $t \twoheadrightarrow_{x,R,s} \bar{t}_n \xrightarrow{a} t' \twoheadrightarrow_{y,R,s'} t'$ and since $s'\ R\ t'$, R meets transfer condition (T2).
   - Case $x = b$. Then also $☹ \notin E(x,y)$ and therefore DUPLICATOR can only have applied rule (3a). In that case we have $s_i = s$ and $t_i = \bar{t}_i$ for all $i \leq n$. Furthermore, since DUPLICATOR wins all configurations on π', she also wins all configurations $[\,(s,t_i),(a,s'),(\bar{t}_i,☹),*\,]$. By Proposition 5.8, she also wins $[\,(s,t_i),\dagger,\dagger,*\,]$ so we find $s\ R\ t_i$ (and therefore $s\ R\ \bar{t}_i$) for all $i \leq n$. As a result, we find that $t \Longrightarrow_{x,R,s} \bar{t}_n$ holds and therefore $t \twoheadrightarrow_{x,R,s} \bar{t}_n \xrightarrow{a} t' \twoheadrightarrow_{y,R,s'} t'$, which, together with $s'\ R\ t'$, is all we needed for concluding that R meets transfer condition (T2).

(2) There is some k ($1 \leq k \leq n$) such that $f_k = ☺$ in π'; fix the smallest k with this property. We note that k must be odd since the first configuration with a ☺ is owned by SPOILER. Observe that this means that $\bar{t}_{k-1} \xrightarrow{a} \bar{t}_{k+1}$ must hold since DUPLICATOR used either rule (2a) or rule (2c) at configuration $\langle\,(s_{k-1},t_{k-1}),(a,s'),(\bar{t}_{k-1},f_{k-1}),*\,\rangle$. At all other configurations (except for the last), DUPLICATOR can only have played rule (3a) or (3c), meaning we must have had $\bar{t}_i \xrightarrow{\tau} \bar{t}_{i+2}$ for all $i \neq k-1$. For the last configuration,

DUPLICATOR could only have used rule (3b). First, consider the sequence of transitions $t = \bar{t}_0 \xrightarrow{\tau} \bar{t}_2 \xrightarrow{\tau} \cdots \xrightarrow{\tau} \bar{t}_{k-1}$.

- Case $k = 1$. Then $t = \bar{t}_{k-1}$ and since $s \mathrel{R} t$ we can then immediately conclude that $t \twoheadrightarrow_{x,R,s} \bar{t}_{k-1}$ holds.
- Case $k > 1$. Following the argument we used in case 1 (reading $k-1$ for n), we conclude that $t \twoheadrightarrow_{x,R,s} \bar{t}_{k-1}$.

Second, consider the sequence of transitions $\bar{t}_{k+1} \xrightarrow{\tau} \bar{t}_{k+3} \xrightarrow{\tau} \cdots \xrightarrow{\tau} \bar{t}_n \xrightarrow{\tau} t'$. Then we can use similar arguments as in case 1 (observing that $y = b$ implies that DUPLICATOR could only have used rule (3a)), to conclude that $\bar{t}_{k+1} \twoheadrightarrow_{y,R,s'} \bar{t}_n$. Moreover, from $\bar{t}_{k+1} \twoheadrightarrow_{y,R,s'} \bar{t}_n$, $\bar{t}_n \xrightarrow{\tau} t'$ and $s' \mathrel{R} t'$, we also obtain $\bar{t}_{k+1} \twoheadrightarrow_{y,R,s'} t'$.

Concluding, we find that we have $t \twoheadrightarrow_{x,R,s} \bar{t}_{k-1} \xrightarrow{a} \bar{t}_{k+1} \twoheadrightarrow_{y,R,s'} t'$. Since we already had established that we have $s' \mathrel{R} t'$, we conclude that R meets transfer condition (T2).

In both cases R meets the transfer condition. Thus, R is an (x, y)-generic bisimulation relation, and therefore, since for any $s, t \in S$ such that $s \equiv_{E(x,y)} t$ we have $s \mathrel{R} t$, we also have $s \leftrightarrow_{(x,y)} t$. □

We are now in a position to prove our claimed correspondence.

Theorem 5.2. *For all $x, y \in \{o, b\}$, we have $s \leftrightarrow_{(x,y)} t$ iff $s \equiv_{E(x,y)} t$.*

Proof. The implication from left to right follows from Lemma 5.5. The implication from right to left follows from Lemma 5.10. □

5.3. **Relating Branching and Generic Bisimulation Games.** As we have shown in the previous section, our generic bisimulation game exactly characterises (x, y)-generic bisimilarity. More specifically, this implies that $\equiv_\emptyset$ coincides with branching bisimilarity. Since we already have a game that captures branching bisimilarity, one may wonder what the relation between $\equiv_\emptyset$ and that game is. In this section, we formally relate the game play in the generic bisimulation game to the game play in the branching bisimulation game.

First, observe that by definition, the game for $\equiv_\emptyset$ consists of all rules by SPOILER, but only rules (1), (2a), (2b), (3a) and (3b) of DUPLICATOR. Intuitively, this one-but-last move coincides with DUPLICATOR's third rule in the branching bisimulation game. Rule (2b) can be seen to match with DUPLICATOR's second rule in the bb-game. However, rule (2a) and (3b) have no counterpart in the bb-game. From these observations, we can expect that every strategy in a bb-game has a matching strategy in our gb-game, but not *vice versa*.

We next formalise these arguments. Consider the abstraction function f which maps configurations in the gb-game to configurations in the bb-game, where f is defined by $f(((s, t), c, m, r)) = ((s, t), c, r)$. The function f can be lifted from configurations to plays in the natural manner. Using f, we can claim that the generic game simulates the branching bisimulation game. More formally, we establish that any strategy of DUPLICATOR in the bb-game induces a 'matching' strategy in the $\emptyset$-gb-game that is such that, after abstraction using f, all plays resulting from this matching strategy are plays in the bb-game. This would allow us to conclude that a winning strategy for DUPLICATOR in the bb-game induces a winning strategy for her in the generic game.

Proposition 5.11. *For every strategy ϱ_{bb} of* DUPLICATOR *in the bb-game, there is a strategy ϱ_{gb} of* DUPLICATOR *in the $\emptyset$-gb-game such that for every ϱ_{gb}-consistent play π, $f(\pi)$ is a ϱ_{bb}-consistent play in the bb-game.*

Proof. Let ϱ_{bb} be an arbitrary strategy of DUPLICATOR in the bb-game. Consider the partial strategy ϱ_{gb}, defined as follows:

$$\begin{aligned} &\varrho_{gb}(\langle\, (s,t),(a,s'),(t,\frownie),r\,\rangle) \\ = \\ &\begin{cases} [\,(s,t'),(a,s'),(t',\frownie),r'\,] & \text{iff } \varrho_{bb}(\langle\,(s,t),(a,s'),r\,\rangle) = [\,(s,t'),(a,s'),r'\,] \\ [\,(s',t'),\dagger,\dagger,\checkmark\,] & \text{iff } \varrho_{bb}(\langle\,(s,t),(a,s'),r\,\rangle) = [\,(s',t'),\dagger,\checkmark\,] \end{cases} \end{aligned}$$

One can check that ϱ_{gb} is well-defined. Now, consider an arbitrary play π starting in some configuration $[\,(s,t),\dagger,\dagger,*\,]$ that is consistent with ϱ_{gb}. We next use induction on the length of the prefix of π to show that $f(\pi)$ is a play in the bb-game that is consistent with ϱ_{bb} and π satisfies the following invariant for all configurations $((s_0,t_0),c_0,m_0,r_0)$ on π:

$$c_0 = m_0 = \dagger, \text{ or } m_0 = (t_0,\frownie) \text{ and } c_0 = (a,s_0') \text{ for some } a, s_0' \text{ such that } s_0 \xrightarrow{a} s_0'$$

Clearly, the prefix of length 1 of π, *viz.* $[\,(s,t),\dagger,\dagger,*\,]$, induces a prefix of length 1, *viz.* $f([\,(s,t),\dagger,\dagger,*\,])$ of some play in the bb-game that is consistent with ϱ_{bb}. Moreover, the single configuration on this prefix satisfies the invariant.

Next, assume that a given prefix π' of length n of π induces a prefix $f(\pi')$ of a play, consistent with ϱ_{bb}, in the bb-game, and we assume that on this prefix π', the above invariant holds. Consider the prefix $\pi'\ ((s',t'),c',m',r')$ of π. We show that all configurations on this prefix meet the invariant and, second, we show that also $f(\pi'\ ((s',t'),c',m',r'))$ is a prefix of a play, consistent with ϱ_{bb}, in the bb-game.

Suppose $[\,(s_0,t_0),c_0,m_0,r_0\,]$ is the last configuration on π'. Since DUPLICATOR and SPOILER alternate their moves, we find that $((s',t'),c',m',r')$ is a configuration owned by DUPLICATOR. We analyse the rule used by SPOILER and show that she could have played to a similar configuration in the bb-game:

- If SPOILER used her first rule, then $c_0 \neq \dagger$ and by our invariant, $c_0 = (a,s_0')$ for some $s_0 \xrightarrow{a} s_0'$, and $m_0 = (t_0,\frownie)$. Thus SPOILER played to configuration $\langle\,(s_0,t_0),(a,s_0'),(t_0,\frownie),*\,\rangle$. Note that this configuration again meets the invariant. Moreover, observe that configuration $f(\langle\,(s_0,t_0),(a,s_0'),(t_0,\frownie),*\,\rangle) = \langle\,(s_0,t_0),(a,s_0'),*\,\rangle$ is a configuration SPOILER could have reached from $[\,(s_0,t_0),(a,s_0'),r_0\,]$ in the bb-game using her first rule, since $s_0 \xrightarrow{a} s_0'$ follows from our invariant.
- If SPOILER used her second rule, then $c_0 = \dagger$ or $c_0 \neq c'$. We first analyse the case when $c_0 = \dagger$. Then for some s_0' such that $s_0 \xrightarrow{a} s_0'$, SPOILER played to $\langle\,(s_0,t_0),(a,s_0'),(t_0,\frownie),*\,\rangle$. Note that the resulting configuration again adheres to the invariant. Moreover, in the bb-game, SPOILER could have played to $\langle\,(s_0,t_0),(a,s_0'),*\,\rangle$.

  If $c_0 \neq c'$ then she played to configuration $\langle\,(s_0,t_0),(a,s_0'),(t_0,\frownie),\checkmark\,\rangle$ for some a, s_0' such that $s_0 \xrightarrow{a} s_0'$ and $c_0 \neq (a,s_0')$. Again, the resulting configuration satisfies our invariant. It immediately follows that SPOILER could have played to $\langle\,(s_0,t_0),(a,s_0),\checkmark\,\rangle$ in the bb-game.
- If SPOILER used her third rule, then she must have played to $\langle\,(t_0,s_0),(a,t_0'),(s_0,\frownie),\checkmark\,\rangle$ for some a, t_0' such that $t_0 \xrightarrow{a} t_0'$. This configuration satisfies the invariant. Moreover, we find that SPOILER could have played to $\langle\,(t_0,s_0),(a,t_0'),\checkmark\,\rangle$ in the branching bisimulation game.

Next, suppose that $\langle\,(s_0,t_0),c_0,m_0,r_0\,\rangle$ is the last configuration on π'. First note that ϱ_{gb} is defined for $(\langle\,(s_0,t_0),c_0,m_0,r_0\,\rangle)$ since, by our invariant, we find that $m_0 = c_0 = \dagger$, or $m_0 = (t_0,\frownie)$ and $c_0 = (a,s_0')$ for some a, s_0' satisfying $s_0 \xrightarrow{a} s_0'$. Since configuration

$f(\langle\,(s_0,t_0),c_0,m_0,r_0\,\rangle)$ is the last configuration on $f(\pi')$ and DUPLICATOR's strategy from $\langle\,(s_0,t_0),c_0,m_0,r_0\,\rangle$ matches $f(\langle\,(s_0,t_0),c_0,m_0,r_0\,\rangle)$, the result follows immediately.

Consequently, in both cases we find that $f(\pi'\ ((s',t'),c',m',r'))$ is a prefix of length $n+1$ of a play consistent with ϱ_{bb}. We thus find that all prefixes of π induce prefixes of $f(\pi)$ that are ϱ_{bb}-consistent and we thus conclude that $f(\pi)$ is a play consistent with ϱ_{bb} in the bb-game. $\square$

From the above proposition, we find that whenever DUPLICATOR wins the bb-game, she also wins the generic game. This follows from the fact that any winning strategy by DUPLICATOR in the bb-game induces a winning strategy for DUPLICATOR in the gb-game.

For the reverse, we first observe that DUPLICATOR can follow an *eager* strategy in the $\equiv_\emptyset$ game: whenever she uses rule (2a) to play to a configuration $[\,(s_0,t_0),(a,s_0'),(t_0,☺),*\,]$, she could also have used rule (2b) to play to configuration $[\,(s_0,t_0),\dagger,\dagger,\checkmark\,]$ instead without changing the outcome of the play. This follows immediately from Proposition 5.8. We say that DUPLICATOR follows an *eager* strategy if she never uses rule (2a). The following result essentially follows immediately from the above.

Lemma 5.12. *If* DUPLICATOR *wins a configuration* $[\,(s,t),\dagger,\dagger,*\,]$*, she has an eager winning strategy to win this configuration.*

Observe that DUPLICATOR's rule (3b) is only 'enabled' once she plays rule (2a). As a consequence, this rule is effectively disabled on all plays in which DUPLICATOR follows an eager strategy. Thus, if DUPLICATOR wins a configuration $[\,(s,t),\dagger,\dagger,*\,]$ she can do so without using rules (2a) and (3b).

Next, consider an augmentation function g which takes configurations from the bb-game and yields configurations in the gb-game, where $g(((s,t),c,r)) = ((s,t),c,(t,☹),r)$ if $c \neq \dagger$ and $g(((s,t),\dagger,r)) = ((s,t),\dagger,\dagger,r)$. We again lift g from configurations to plays in the natural manner.

Proposition 5.13. *For every eager strategy ϱ_{gb} of* DUPLICATOR *in the gb-game, there is a strategy ϱ_{bb} of* DUPLICATOR *in the bb-game such that for every ϱ_{bb}-consistent play π, $g(\pi)$ is a ϱ_{gb}-consistent play in the gb-game.*

Proof. Analogous to the proof of Proposition 5.11. $\square$

As a result, every eager strategy in the gb-game induces a strategy in the bb-game. We then have the following result.

Theorem 5.14. *We have $s \equiv_b t$ iff $s \equiv_\emptyset t$.*

Proof. The implication from left to right follows from Proposition 5.11. This can be seen as follows: assuming that $s \equiv_b t$ holds, then DUPLICATOR has a strategy to win configuration $[\,(s,t),\dagger,*\,]$. Consequently, DUPLICATOR has a strategy in the gb-game such that every play π that is consistent with this play induces a play $f(\pi)$ in the bb-game that is consistent with DUPLICATOR's winning strategy. Since $f(\pi)$ is won by DUPLICATOR in the bb-game, π is won by her in the gb-game. Therefore, DUPLICATOR wins $[\,(s,t),\dagger,\dagger,*\,]$ in the gb-game and thus $s \equiv_\emptyset t$.

Using identical arguments, the implication from right to left can be shown to follow from Proposition 5.13. $\square$

5.4. **Variations on the Generic Bisimulation Games.** We finish this section with a brief discussion on two possible alternative definitions for our gb-game. The definition we presented in Section 5.1 is, as we illustrated in the previous subsection, closely related to the bb-game we presented in Section 3.

A first alternative definition that is similar in spirit to the current definition is obtained by uncoupling 'termination' from the walk along the edges in the transition system. More specifically, we can drop DUPLICATOR's rules (2b) and (3b) and rephrase rule (1) as follows:
- move to $[\,(s_0', s), \dagger, \dagger, \checkmark\,]$ if $a = \tau$ or $f = ☺$.

The resulting game definition is more concise than the original one. While it is not too hard to see that the resulting game is still the same as the original game play, the additional round that DUPLICATOR needs to conclude matching SPOILER's challenge makes that it is a bit more involved to show that it generalises the bb-game.

The second alternative we present is further removed from our current definition. Rather than parameterising DUPLICATOR's rules, one can also parameterise the rules of SPOILER. This means that DUPLICATOR's role is reduced to updating the partial match, moving along the transition system taking a number of τ actions and the action occurring in SPOILER's challenge. Depending on the relation that is being characterised, SPOILER can 'prematurely' decide to pose a new challenge to DUPLICATOR and continue the game play from a fresh position. This way, *e.g.* branching bisimilarity is captured by allowing SPOILER to pose new challenges at any point in the game play, whereas, *e.g.* η-bisimilarity is captured by allowing SPOILER to pose new challenges so long as the configurations contain partial matches of the form $(s, ☹)$. Formally, the game is as follows.

Definition 5.15. For each $E \subseteq \{☹, ☺\}$, a *dual generic bisimulation (dgb) game* on an LTS L (or E-dgb-game) is played by SPOILER and DUPLICATOR on an arena of SPOILER-owned configurations $[\,(s,t), c, m, r\,]$ and DUPLICATOR-owned configurations $\langle\,(s,t), c, m, r\,\rangle$, where $(s,t) \in \mathit{Position}$, $c \in \mathit{Challenge}_\dagger$, $m \in \mathit{Match}_\dagger$ and $r \in \mathit{Reward}$. We write $((s,t), c, m, r)$ if we do not care about the owner of the configuration. SPOILER's and DUPLICATOR's moves are given by the following rules:

- SPOILER moves from a configuration $[\,(s,t), c, m, r\,]$ by:
  (1) if $c \neq \dagger$, moving to $\langle\,(s,t), c, m, r\,\rangle$, or
  (2) if $c = \dagger$, either:
    – selecting $s \xrightarrow{a} s'$ and moving to $\langle\,(s,t), (a,s'), (t, ☹), *\,\rangle$, or
    – selecting $t \xrightarrow{a} t'$ and moving to $\langle\,(t,s), (a,t'), (s, ☹), \checkmark\,\rangle$.
  (3) if $c \neq \dagger$, $m = (\bar{t}, ☹)$ and $☹ \notin E$, either:
    – selecting $s \xrightarrow{a} s'$ and moving to $\langle\,(s,\bar{t}), (a,s'), (\bar{t}, ☹), \checkmark\,\rangle$, or
    – selecting $\bar{t} \xrightarrow{a} t'$ and moving to $\langle\,(\bar{t},s), (a,t'), (s, ☹), \checkmark\,\rangle$.
  (4) if $c = (a, s')$, $m = (\bar{t}, ☺)$ and $☺ \notin E$, either:
    – selecting $s' \xrightarrow{b} s''$ and moving to $\langle\,(s',\bar{t}), (b,s''), (\bar{t}, ☹), \checkmark\,\rangle$
    – selecting $\bar{t} \xrightarrow{b} t'$ and moving to $\langle\,(\bar{t},s'), (b,t'), (s', ☹), \checkmark\,\rangle$

- DUPLICATOR responds from a configuration $\langle\,(u,v), (a,u'), (\bar{v}, f), r\,\rangle$ by:
  (1) not moving if $a = \tau$ or $f = ☺$ and continuing in $[\,(u', \bar{v}), \dagger, \dagger, \checkmark\,]$, or
  (2) moving $\bar{v} \xrightarrow{a} v'$ if available and continuing in $[\,(u,v), (a,u'), (v', ☺), *\,]$ if $f = ☹$, or
  (3) moving $\bar{v} \xrightarrow{\tau} v'$ if available and continuing in configuration $[\,(u,v), (a,u'), (v', f), *\,]$.

Duplicator wins a finite play starting in a configuration $((s,t),c,m,r)$ if Spoiler gets stuck, and she wins an infinite play if the play yields infinitely many ✓ rewards. All other plays are won by Spoiler. We say that a configuration is won by a player when she has a strategy that wins all plays starting in it. *Full* plays of the game start in a configuration $[\,(s,t),\dagger,\dagger,*\,]$; we say that Duplicator wins the E-dgb-game for a position (s,t), if the configuration $[\,(s,t),\dagger,\dagger,*\,]$ is won by her; in this case, we write $s \equiv_{dE} t$. Otherwise, we say that Spoiler wins that game.

We claim, without further proof, that the dual generic game characterises (again) branching bisimilarity, when $E = \emptyset$; weak bisimilarity, when $E = \{☹, ☺\}$; η-bisimilarity, when $E = \{☺\}$; and delay bisimilarity, whenever $E = \{☹\}$.

6. Extensions

In this section we investigate how our generic bisimulation games can be modified to obtain other relations.

We observe that all relations discussed so far are not sensitive to divergences, in the sense that a state in which a divergence (an infinite τ-path) is possible, cannot be distinguished from a state in which no divergence is possible, but which otherwise exhibits the exact same behaviour. Our first modification of our game is thus to make it such that it can distinguish divergent states from non-divergent states. For the branching bisimulation case our modified game characterises *branching bisimilarity with explicit divergence* (also sometimes called divergence preserving branching bisimilarity). Second, we show that, with minimal changes, our game can be modified to obtain the *simulation* counterparts of the bisimulations we discussed so far.

6.1. **Explicit Divergences.**

Intuitively, branching bisimulation with explicit divergence imposes the following additional constraint on top of branching bisimulation. Whenever there is a divergent path from one of the states in a related pair for which all states on that path are related, the related state is also part of such a divergent path. In other words, these branching bisimulations will preserve the existence of infinite executions of internal actions through states with the same (behavioural) potentials.

Note that this is not the simplest possible way of defining a refinement of branching bisimulation that captures divergences. In fact, there are several competing proposals in the literature. For instance, the one studied by Bergstra *et al.* [4] simply imposes that in order to be related by such a bisimulation, whenever one of the states is divergent, *i.e.*, admits an infinite sequence of τ-steps, then the other is too. This, however, produces a coarser divergence sensitive branching bisimulation than the one we study in this section, which is essentially based on [30, 32, 29, 35]. In addition to the (technical) argument that the resulting relation is a congruence for parallel composition and distinguishes livelocked states from deadlocked states [31], it is argued that branching bisimulations are able to see (up to branching bisimulation itself) all the intermediate states along a computation, and therefore this idea should be preserved when considering divergent computations. Other authors preferred the opposite approach, where an even coarser treatment of divergence to that in [4] is proposed. For instance, Walker made a quite detailed study of bisimulation and divergence in [33], where a notion of *local divergence* that takes into account the possibility of executing new observable actions in the future, is considered. This notion was already proposed by Hennessy and Plotkin as early as in 1980 [17], although its development had to wait ten more years.

There are also various formalisations of branching bisimilarity with explicit divergence. Van Glabbeek *et al.* investigated these variations in [30], proving that all such variations were in essence equivalent. We here use one of their (many) equivalent characterisations:

Definition 6.1 ([30, Condition D_4]). A symmetric relation $R \subseteq S \times S$ is a *branching bisimulation with explicit divergence* if and only if R is a branching bisimulation and for all $s\ R\ t$, if there is an infinite sequence $s = s_0 \xrightarrow{\tau} s_1 \xrightarrow{\tau} s_2 \cdots$, then there is a state t' such that $t \twoheadrightarrow^{+} t'$ and for some k, $s_k\ R\ t'$. We write $s \leftrightarrows_b^{ed} t$ iff there is a branching bisimulation with explicit divergence R such that $s\ R\ t$.

We here opt to use condition D_4 instead of, *e.g.* their condition D, in which all states on the divergent paths are related. Partly, this is to allow for local arguments in the game based definition and the corresponding proof, and partly because not all their conditions turn out to generalise straightforwardly to our (x, y)-generic bisimulation. For the moment we defer a discussion on this topic to Section 6.2.

In the previous section we have presented our (x, y)-generic bisimulation as a natural way of capturing, in a generic way, four kinds of bisimulation capturing abstraction. We continue this generalising approach in this section, defining (x, y)-*generic bisimulation with explicit divergences* by adding the constraint on divergent computations from Definition 6.1 to our definition of (x, y)-generic bisimulation from Definition 4.5.

Definition 6.2. A symmetric relation $R \subseteq S \times S$ is an (x, y)-*generic bisimulation with explicit divergence* if and only if R is an (x, y)-generic bisimulation and for all $s\ R\ t$, if there is an infinite sequence $s = s_0 \xrightarrow{\tau} s_1 \xrightarrow{\tau} s_2 \cdots$, then there is a state t' such that $t \twoheadrightarrow^{+} t'$ and for some k, $s_k\ R\ t'$. We write $s \leftrightarrows_{(x,y)}^{ed} t$ iff there is an (x, y)-generic bisimulation with explicit divergence R such that $s\ R\ t$.

Let us next argue why this is an adequate way of defining a divergence sensitive extension of all the instances of our (x, y)-generic bisimulation. One could argue that we are adding a condition which is too strong for some of the relations since it originates from the branching bisimulation case. However, the added condition uses the transitive τ-closure and refers to the particular relation we are defining in each case, and therefore it allows to consider the intermediate states along a divergent computation with respect to the corresponding equivalence. Essentially, we are just applying a 'categorical' approach, by asking for the preservation of divergent computations including the 'observation' of the semantical information that we are capturing in each case. Moreover, in this way we are obtaining a uniformly defined extension of all the instances of our (x, y)-generic bisimulation, something that cannot be claimed if for any reason we would prefer the alternative finer (or coarser) extension for some (but not all) of the instances of our (x, y)-generic bisimulation.

Note that with the above definition at hand, it is not too hard to prove that $\leftrightarrows_{(x,y)}^{ed}$ is an equivalence relation. Moreover, it follows immediately that any relation R that is a (b, b)-generic bisimulation with explicit divergence is also a branching bisimulation with explicit divergence. Also, $\leftrightarrows_{(x,y)}^{ed}$ has the stuttering property.

Lemma 6.3. *Let $x, y \in \{o, b\}$. Then $\leftrightarrows_{(x,y)}^{ed}$ satisfies the stuttering property (Definition 2.5).*

Proof. Let $t_0 \xrightarrow{\tau} t_1 \cdots \xrightarrow{\tau} t_k$ with $t_0 \leftrightarrow^{ed}_{(x,y)} t_k$. We define relation R as follows:

$$R = \{(t_0, t_i), (t_i, t_0) \mid 0 \leq i < k\} \cup \leftrightarrow^{ed}_{(x,y)}$$

We can prove that R is a (x, y)-generic bisimulation relation with explicit divergence. Proving the transfer condition for all pairs $t_0\ R\ t_i$ is again analogous to the proof of [25, Lemma 4.9.2]. For condition D_4 we sketch the proof. We distinguish two cases. Suppose $i > 0$ (for $i = 0$ the result follows immediately from reflexivity).

- $t_0\ R\ t_i$. Suppose there exists an infinite sequence $t_0 = \bar{t}_0 \xrightarrow{\tau} \bar{t}_1 \xrightarrow{\tau} \cdots$. First, observe that $t_0 \leftrightarrow^{ed}_{(x,y)} t_k$, so there exists $t_k \twoheadrightarrow^+ t'$ and ℓ such that $t' \leftrightarrow^{ed}_{(x,y)} \bar{t}_\ell$. Since $t_i \twoheadrightarrow t_k$, also $t_i \twoheadrightarrow^+ t'$.
- $t_i\ R\ t_0$. Suppose there exists an infinite sequence $t_i = \bar{t}_0 \xrightarrow{\tau} \bar{t}_1 \xrightarrow{\tau} \cdots$. Since $i > 0$ and $t_0 \twoheadrightarrow t_i$, also $t_0 \twoheadrightarrow^+ t_i$, and the result follows from $t_i \leftrightarrow^{ed}_{(x,y)} t_i$ due to reflexivity. □

In the remainder of this section we present a generic game characterisation of (x, y)-generic bisimilarity with explicit divergence and prove its correctness, seeing that the proofs presented in [9], for the particular case of branching bisimilarity can be easily transferred to the general case. We first recall the branching bisimulation with explicit divergence game introduced in [9].

Definition 6.4. A *branching bisimulation with explicit divergence (bbed) game* on an LTS L is played by players SPOILER and DUPLICATOR on an arena of SPOILER-owned configurations $[\,(s,t), c, r\,]$ and DUPLICATOR-owned configurations $\langle\,(s,t), c, r\,\rangle$, where $((s,t), c, r) \in \mathit{Position} \times \mathit{Challenge}_\dagger \times \mathit{Reward}$, and $\mathit{Position}$, $\mathit{Challenge}_\dagger$ and Reward are as before. By convention, we write $((s,t), c, r)$ if we do not care about the owner of the configuration.

- SPOILER moves from a configuration $[\,(s,t), c, r\,]$ by:
  1. selecting $s \xrightarrow{a} s'$ and moving to $\langle\,(s,t), (a, s'), *\,\rangle$ if $c = (a, s')$ or $c = \dagger$, and to $\langle\,(s,t), (a, s'), \checkmark\,\rangle$, otherwise; or
  2. picking some $t \xrightarrow{a} t'$ and moving to $\langle\,(t,s), (a, t'), \checkmark\,\rangle$.
- DUPLICATOR responds from a configuration $\langle\,(u,v), (a, u'), r\,\rangle$ by:
  1. not moving if $a = \tau$ and continuing in configuration $[\,(u', v), \dagger, *\,]$, or,
  2. moving $v \xrightarrow{a} v'$ if available and continuing in configuration $[\,(u', v'), \dagger, \checkmark\,]$, or
  3. moving $v \xrightarrow{\tau} v'$ if available and continuing in configuration $[\,(u, v'), (a, u'), *\,]$.

DUPLICATOR wins a finite play starting in a configuration $((s,t), c, r)$ if SPOILER gets stuck, and she wins an infinite play if the play yields infinitely many $\checkmark$ rewards. All other plays are won by SPOILER. We say that a configuration is won by a player when she has a strategy that wins all plays starting in it. *Full* plays of the game start in a configuration $[\,(s,t), \dagger, *\,]$; we say that DUPLICATOR wins the bbed-game for a position (s, t), if the configuration $[\,(s,t), \dagger, *\,]$ is won by it; in this case, we write $s \equiv^{ed}_b t$. Otherwise, we say that SPOILER wins that game.

Comparing this definition with Definition 3.7, it is hard to pinpoint the difference due to the similarities between the games. In fact, the only difference is in DUPLICATOR's first rule (rule (1) in Definition 3.7), in which idling in response to an internal move is no longer rewarded, whereas it was in the bb-game. As we proved in [9], this single change is sufficient to capture branching bisimulation with explicit divergence. We here only repeat this result.

Theorem 6.5. *We have* $\leftrightarrow^{ed}_b\ = \equiv^{ed}_b$.

While the formal proof of correctness of this game in [9] is fairly involved, the intuition is quite simple. First, observe that in order to check a divergence, Spoiler can challenge Duplicator by presenting the internal steps on the divergence one by one. Duplicator is forced to reply to infinitely many of these internal steps by an internal step of her own, otherwise she loses the game since she does not collect infinitely many ✓ rewards. Instead, in the original bb-game Duplicator could win by replying to a divergence by just remaining idle, because her first rule also rewarded her with a ✓. We also need to check that when not rewarding 'isolated' τ moves that Duplicator replies to by remaining idle, Duplicator cannot (incorrectly) be declared loser of a game. This cannot happen, simply because an infinite play that is not checking a divergence will include infinitely many challenges that correspond to visible actions, and Duplicator should match all of them, so that she will collect infinitely many ✓ rewards that she needs to win the play, even if she does not get the ✓ rewards corresponding to the idling responses to τ moves.

We next use the idea from Definition 6.4 to introduce the *generic bisimulation with explicit divergence (gbed) game.*

Definition 6.6. Let $E \subseteq \{☺, ☹\}$. A *generic bisimulation with explicit divergence (gbed) game* (or E-bisimulation game with explicit divergences) on an LTS L is played by players Spoiler and Duplicator on an arena of Spoiler-owned configurations $[\,(s,t), c, m, r\,]$ and Duplicator-owned configurations $\langle\,(s,t), c, m, r\,\rangle$, where $(s,t) \in \mathit{Position}$, $c \in \mathit{Challenge}_\dagger$, $m \in \mathit{Match}_\dagger$, $r \in \mathit{Reward}$, and $\mathit{Position}, \mathit{Challenge}_\dagger, \mathit{Reward}$ and $\mathit{Match}_\dagger$ are as before. We write $((s,t), c, m, r)$ if we do not care about the owner of the configuration.

- From a configuration $[\,(s,t), c, m, r\,]$, Spoiler can:
  1. move to $\langle\,(s,t), c, m, *\,\rangle$ if $c \neq \dagger$, or
  2. for some $s \xrightarrow{a} s'$, move to either:
     - (a) $\langle\,(s,t), (a,s'), (t, ☹), *\,\rangle$, if $c = \dagger$, or
     - (b) $\langle\,(s,t), (a,s'), (t, ☹), ✓\,\rangle$, if $c \neq (a,s')$
  3. for some $t \xrightarrow{a} t'$, move to: $\langle\,(t,s), (a,t'), (s, ☹), ✓\,\rangle$.

- From a configuration $\langle\,(u,v), (a,u'), (\bar{v}, f), r\,\rangle$, Duplicator can:
  1. move to $[\,(u', \bar{v}), \dagger, \dagger, *\,]$ when $a = \tau$, or
  2. if $f = ☹$ and $\bar{v} \xrightarrow{a} v'$, move to one of the following:
     - (a) $[\,(u',v'), (a,u'), (v', ☺), *\,]$, in any case, or
     - (b) $[\,(u',v'), \dagger, \dagger, ✓\,]$, in any case, or
     - (c) $[\,(u,v), (a,u'), (v', ☺), *\,]$, only if $☺ \in E$
  3. for some $\bar{v} \xrightarrow{\tau} v'$, move to one of the following:
     - (a) $[\,(u,v'), (a,u'), (v', f), *\,]$, in any case, or
     - (b) $[\,(u',v'), \dagger, \dagger, ✓\,]$, only if $f = ☺$
     - (c) $[\,(u,v), (a,u'), (v', f), *\,]$, only if $f \in E$

Duplicator wins a finite play starting in a configuration $((s,t), c, m, r)$ if Spoiler gets stuck, and she wins an infinite play if the play yields infinitely many ✓ rewards. All other plays are won by Spoiler. We say that a configuration is won by a player when she has a strategy that wins all plays starting in it. *Full* plays of the game start in a configuration $[\,(s,t), \dagger, \dagger, *\,]$; we say that Duplicator wins the game for a position (s,t), if the configuration $[\,(s,t), \dagger, \dagger, *\,]$ is won by her. In this case, we write $s \equiv_E^{ed} t$. Otherwise, we say that Spoiler wins that game.

As before for the bb-game, the modification compared to Definition 5.1 is marginal: only the ✓ reward in DUPLICATOR's rule (1) has been turned into $*$. The idea here is the same as for the bbed-game.

We next prove that for all $x, y \in \{o, b\}$ the relation induced by the $E(x, y)$-bisimulation game exactly captures (x, y)-generic bisimilarity with explicit divergence. We split the proof obligations into three separate lemmata, first addressing completeness (Lemma 6.7) and next addressing soundness (Lemmata 6.8 and 6.9).

Lemma 6.7 (Completeness)**.** *For all $x, y \in \{o, b\}$, whenever we have $s \leftrightarroweq^{ed}_{(x,y)} t$, we also have $s \equiv^{ed}_{E(x,y)} t$.*

Proof. We again design a partial strategy for DUPLICATOR for the gbed-game that starts in $[\,(s,t), \dagger, \dagger, *\,]$. We first construct a DUPLICATOR strategy using the same construction as used for defining the strategy in the proof of Lemma 5.5 to win the corresponding gb-game, but using $\leftrightarroweq^{ed}_{(x,y)}$ instead of $\leftrightarroweq_{(x,y)}$ as relation R in the construction. However, if we do not change anything in this strategy, it could be the case that SPOILER now wins the gbed-game, since the strategy does not take divergences into account. Let us see which changes are needed to guarantee that DUPLICATOR will also win the gbed-game.

First, note that all the *positions* along any play consistent with that winning strategy for DUPLICATOR contain two $\leftrightarroweq_{(x,y)}$ equivalent states, reusing the proof in Lemma 5.10. Second, observe that we start from a configuration $[\,(s,t), \dagger, \dagger, *\,]$ containing two $\leftrightarroweq^{ed}_{(x,y)}$ equivalent states, and in order to be able to repeat our arguments after any move of DUPLICATOR, we must preserve the $\leftrightarroweq^{ed}_{(x,y)}$ relation, and not just $\leftrightarroweq_{(x,y)}$, as in the proof of Lemma 5.5.

If we apply the strategy that we have defined for the gb-game, the only case in which DUPLICATOR loses the game is that in which she is generating infinitely many † challenges, but only finitely many ✓ rewards. In particular, there would be some suffix of a play in which DUPLICATOR generates infinitely many † challenges, but no ✓ reward. We consider that suffix as a full play, and denote by (s_0, t_0) the position at the configuration that suffix starts.

Let us first make a few observations about the moves played by both players along this suffix:

- SPOILER only plays moves (1) or (2a);
- DUPLICATOR only plays move (1)

We analyse why this must be the case. First, observe that in SPOILER's moves (2b) and (3) and DUPLICATOR's moves (2b) and (3b) immediately a ✓ is rewarded, which is a contradiction to the assumption that there are no ✓ rewards on the suffix. Now, suppose DUPLICATOR plays any of the moves (2a), (2c), (3a), or (3c). In each of these cases, when playing according to the strategy defined in the proof of Lemma 5.5, when this move is played, eventually the ✓ is obtained through either DUPLICATOR's move (2b) or (3b).[3]

Now, since DUPLICATOR is always playing using rule (1), all challenges involved in the considered infinite suffix concern τ actions (by definition of rule (1)), and generate a divergent sequence $s_0 \xrightarrow{\tau} s_1 \xrightarrow{\tau} s_2 \xrightarrow{\tau} \cdots$, and DUPLICATOR always responds by leaving t_0 the same, so that the invariance of $\leftrightarroweq^{ed}_{(x,y)}$ implies that $s_i \leftrightarroweq^{ed}_{b} t_0$, for all i. But then, by definition of $\leftrightarroweq^{ed}_{(x,y)}$, there must be some sequence of transitions $t_0 \twoheadrightarrow^{+} t'$ such that for some

[3]This can be checked using a tedious but elementary analysis on the cases in the defined strategy.

k, $s_k \leftrightarroweq^{ed}_{(x,y)} t'$. Write this sequence as $t_0 \xrightarrow{\tau} t_1 \xrightarrow{\tau} \cdots \xrightarrow{\tau} t_\ell = t'$ (for $\ell > 0$). Since $s_i \leftrightarroweq^{ed}_{(x,y)} t_0$ for all i, and $\leftrightarroweq^{ed}_{(x,y)}$ is an equivalence relation, we have $s_i \leftrightarroweq^{ed}_{(x,y)} s_j$ for all i, j. In a similar vein, we can also conclude $t_0 \leftrightarroweq^{ed}_{(x,y)} t_\ell$. Since $\leftrightarroweq^{ed}_{(x,y)}$ has the stuttering property according to Lemma 6.3, we now find that $t_i \leftrightarroweq^{ed}_{(x,y)} t_j$ for all $0 \leq i, j \leq \ell$, and hence $s_0 \leftrightarroweq^{ed}_{(x,y)} t_i$ for $0 \leq i \leq \ell$.

This allows us to modify the strategy Duplicator plays from the configurations $\langle\, (s_0, t_i), (\tau, s_1), (t_i, \frownie), * \,\rangle$ for $i < \ell - 1$ in such a way that she plays the τ-step from t_i using rule (3a) to $[\, (s_0, t_{i+1}), (\tau, s_1), (t_i, \frownie), * \,]$. From $\langle\, (s_0, t_{\ell-1}), (\tau, s_1), (t_{\ell-1}, \frownie), * \,\rangle$, she now plays to $[\, (s_1, t_\ell), \dagger, \dagger, \checkmark \,]$.

Note that all configurations that we play to in this changed strategy are *good*. From $[\, (s_1, t_\ell), \dagger, \dagger, *) \,]$, Duplicator can proceed as she would from $[\, (s_1, t_\ell), \dagger, \dagger, *) \,]$, and apply the same modifications as described above.

In this way we get a revised strategy for Duplicator that will allow her to win the gbed-game that starts in $[\, (s, t), \dagger, \dagger, * \,]$, thus proving $s \equiv^{ed}_{E(x,y)} t$. □

We continue by showing that the relation induced by the *gbed*-game is, in fact, an (x, y)-generic bisimulation (without the divergence requirement). Note that this result follows more or less by design since the gbed-game is stricter than our original gb-game since it rejects plays that were once winning for Duplicator.

Lemma 6.8. *For all $x, y \in \{o, b\}$, the relation $\equiv^{ed}_{E(x,y)}$ is an (x, y)-generic bisimulation.*

Proof. Since the $E(x, y)$-bisimulation with explicit divergence games are obtained from the corresponding $E(x, y)$-bisimulation games simply by turning some ✓ rewards into $*$, and in this way any configuration that is won by Duplicator at the former is also a winning configuration for her at the latter, we can repeat the reasoning in the proof of Lemma 5.10 substituting the ✓ reward by a $*$ reward whenever Duplicator resorts to choosing her first option, to obtain the proof that $\equiv^{ed}_{E(x,y)}$ is an (x, y)-generic bisimulation. □

The following lemma confirms that our new game is indeed capable of discerning states that are divergent and states that are non-divergent. More specifically, it states that the relation induced by $\equiv^{ed}_{E(x,y)}$ meets divergence condition D_4.

Lemma 6.9. *Let $s \equiv^{ed}_{E(x,y)} t$, and assume that we have a divergent sequence $s = s_0 \xrightarrow{\tau} s_1 \xrightarrow{\tau} s_2 \xrightarrow{\tau} \cdots$. Then there is some t' and some k such that $t \twoheadrightarrow^+ t'$ and $s_k \equiv^{ed}_{E(x,y)} t'$.*

Proof. Towards a contradiction, suppose that for all t' for which $t \twoheadrightarrow^+ t'$, and for all k, we have $s_k \not\equiv^{ed}_{E(x,y)} t'$. Consider Spoiler's partial strategy to use rule (2a) with challenge $s_i \xrightarrow{\tau} s_{i+1}$ for all configurations of the form $[\, (s_i, t'), \dagger, \dagger, r \,]$, and for configurations of the form $[\, (s_i, t'), (\tau, s_{i+1}), (\bar{t}, f), r \,]$, with $t \twoheadrightarrow^+ \bar{t}$, use rule (1) when $t' = t$, and when $t' \neq t$, the strategy that mimics Spoiler's winning strategy for $[\, (s_i, t'), \dagger, \dagger, * \,]$. Note that since Duplicator wins $[\, (s_0, t), \dagger, \dagger, * \,]$, all plays consistent with Duplicator's winning strategy visit Spoiler-owned configurations for which the above strategy is defined.

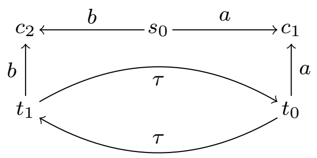
Consider the play that emerges from $[\, (s_0, t), \dagger, \dagger, * \,]$ by following Spoiler's strategy and Duplicator's winning strategy. Since Spoiler would only make moves generated by τ transitions when following her fixed strategy, Duplicator can only reply using τ transitions as well. Moreover, to win the play she needs to collect infinitely many ✓ rewards. In order to collect the first of them, she needs to sometime apply either rule (2b) or (3b). When doing this she is advancing her state by executing a τ transition, thus moving to a configuration with position (s_k, t'), for some $k \geq 0$ and t' such that $t \twoheadrightarrow^+ t'$. But by definition, Spoiler's strategy is winning for configurations with such positions, and therefore the play is won by Spoiler. As a result we find $s_0 \not\equiv^{ed}_{E(x,y)} t$, which contradicts our assumptions. So there must be some k and some t' such that $t \twoheadrightarrow^+ t'$ and $s_k \equiv^{ed}_{E(x,y)} t'$. $\square$

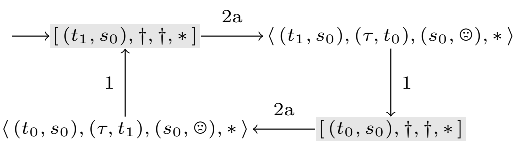
By combining the three preceding lemmata, we can next conclude that the gbed-game characterises our relational definition of (x, y)-generic bisimilarity with explicit divergence.

Theorem 6.10. *For all $x, y \in \{o, b\}$, we have $\leftrightarrow^{ed}_{(x,y)} = \equiv^{ed}_{E(x,y)}$.*

Proof. The implication from left to right follows from Lemma 6.7, the implication from right to left follows from Lemmata 6.8 and 6.9. $\square$

Example 6.11. Recall the labelled transition system from Figure 1 (depicted left below), with the corresponding gbed-game with $E = \emptyset$ (shown right below).





Observe that s_0 and t_1 are branching bisimilar, but not branching bisimilar with explicit divergence. Spoiler tries to show they are not equivalent by challenging Duplicator with τ transitions from t_1, and subsequently t_0 using her rule (2a). Since in s_0 Duplicator cannot play any τ transitions, she can only respond with her move (1), and she loses since she never gains any ✓ reward.

In Section 7, we present a larger application of the gbed-game.

6.2. **A Note on Divergence Conditions.** Branching bisimilarity with explicit divergence can be defined in terms of a relational characterisation as we did in the previous section, but also in terms of a modal characterisation or in terms of coloured traces. The relational characterisation of branching bisimilarity with explicit divergence essentially builds on the relational definition of branching bisimulation, adding an orthogonal divergence condition to it. The literature, however, defines several incomparable divergence conditions which are all used to this effect. It turns out that all of these alternatives (referred to as condition D, D_0, D_1, D_2, D_3 and D_4) give rise to the same behavioural equivalence relation, see [30].

Our definition of (x, y)-generic bisimilarity with explicit divergence is based on divergence condition D_4, see also Definition 6.1. Yet, a natural question is whether the alternative divergence conditions also give rise to equivalent characterisations for our (x, y)-generic bisimilarity with explicit divergence. The answer to this question is negative: using condition D_2 gives rise to a different behavioural relation than condition D_4.

Formally condition D_2 requires of a relation R that for every $s\ R\ t$ for which we have $s = s_0 \xrightarrow{\tau} s_1 \xrightarrow{\tau} \ldots$, there is some immediate τ-successor t' of t such that $s_k\ R\ t'$ for some k. Note that this strengthens condition D_4, which requires that t' is some state that can be reached via one or more τ-steps.

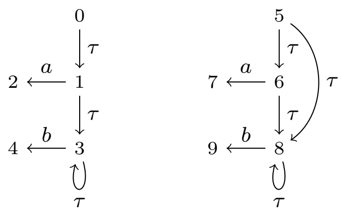


Figure 7: An illustration of the difference between condition D_2 and D_4.

The difference between these two conditions is illustrated through the transition system depicted in Figure 7. While we cannot expect the divergence condition D_2 and D_4 for (b,b)-generic bisimilarity with explicit divergence to give different results, they do give different results for (o,o)-generic bisimilarity with explicit divergence. First, observe that we have $0 \mathrel{\underline{\leftrightarrow}}_{(o,o)} 5$. States 0 and 5 can, however, not be related through a weak bisimulation relation that meets condition D_2. The offending behaviour here is exactly the divergent computation $5 \xrightarrow{\tau} 8 \xrightarrow{\tau} 8 \xrightarrow{\tau} \ldots$ which cannot be mimicked from state 0 because state 1 cannot be related to any state on the divergent computation. On the other hand, we do have $0 \mathrel{\underline{\leftrightarrow}}^{ed}_{(o,o)} 5$. This can be seen using our game characterisation: if Spoiler challenges Duplicator to mimic the divergent computation from state 5, then Duplicator can respond by moving from state 0 to state 3 using rule (3c) and subsequently using rule (2b).

The above observation raises several questions. For instance, it is not clear whether there is a game characterisation of (x,y)-generic bisimilarity with explicit divergence based on divergence condition D_2. The same can be asked for some of the other divergence conditions. Moreover, one can wonder for which of the conditions the various instances of (x,y)-generic bisimilarity with explicit divergence would coincide; do they all give rise to equivalence relations, *etcetera*. We defer answering these questions to future research.

6.3. **Simulation Games.** So far we have considered four bisimulation relations and showed that explicit divergences can be added orthogonally. All of these relations are equivalence relations. When checking an implementation relation, sometimes it is desirable to drop the symmetry requirement, and use simulation relations instead. In this section we sketch how our generic bisimulation game can be adapted to reflect the corresponding simulation relations. We here forego a formal treatment, since most of the required modifications are straightforward, and the corresponding proofs only require minor changes compared to what we have presented so far.

The similarity versions of the four instances of our generic bisimulation already appeared in the preliminary version of [29], and some of them we also mentioned later, see, *e.g.* [14], where the induced bisimilarity notions were studied in depth. Of course, these similarity notions generate *preorders*, although the corresponding *kernels* would again provide equivalence relations. Van Glabbeek and Weijland [32] also briefly mention the possibility of defining η, delay- and branching bisimulation preorders with explicit divergences. However, in this case the starting point is not a simulation relation, but the corresponding bisimulation equivalence, that does not (still) take into account divergences. Then, the treatment of these divergences induces a finer preorder, where behaviours without divergences are 'better' than the equivalent (up to divergences) ones that contain some divergent computations. Also, the notion of divergence preserving branching simulation defined in [24] comes quite close to a branching bisimulation preorder with explicit divergences.

We next define *generic simulation*, by dropping the symmetry requirement from Definition 4.5.

Definition 6.12. A relation $R \subseteq S \times S$ is an (x, y)-*generic simulation*, whenever $s\, R\, t$ and $s \xrightarrow{a} s'$ imply either:

- $a = \tau$ and $s'\, R\, t$, or
- there exist states t', t_1, t_2 such that $t \twoheadrightarrow_{x,R,s} t_1 \xrightarrow{a} t_2 \twoheadrightarrow_{y,R,s'} t'$ and $s'\, R\, t'$.

Now, for the corresponding values of x and y, we write $s \leq_{(x,y)} t$ and say that s is (x, y)-simulated by t iff there is an (x, y)-generic simulation R such that $s\, R\, t$. Typically, we simply write $\leq_{(x,y)}$ to denote (x, y)-*generic similarity*, and we write $\sim_{(x,y)}$ to denote the (x, y)-*generic simulation equivalence* $\leq_{(x,y)} \cap \leq_{(x,y)}^{-1}$.

As before, we have the following correspondences for $R \subseteq S \times S$:

- R is a weak simulation iff it is an (o, o)-generic simulation;
- R is a delay simulation iff it is an (o, b)-generic simulation;
- R is an η-simulation iff it is a (b, o)-generic simulation;
- R is a branching simulation iff it is a (b, b)-generic simulation.

Van Glabbeek already proved in [29], using his testing characterisations, that the preorders induced by the first two classes of simulations above are the same, and that this is also the case for those preorders induced by the latter two. Next we provide a direct proof based on the definition of our (coinductive) generic simulations.

Proposition 6.13. *For all $x, y, z \in \{o, b\}$ we have $s \leq_{(z,x)} t \Leftrightarrow s \leq_{(z,y)} t$.*

Proof. For each relation R we define the derived relation $R^{\twoheadleftarrow}$ by

$$R^{\twoheadleftarrow} = \{(s, t) \mid \exists t' \colon t \twoheadrightarrow t' \wedge s\, R\, t'\}.$$

Let us see that whenever R is a (z, o)-generic simulation, the derived relation $R^{\twoheadleftarrow}$ is too. Indeed, when we have $s\, R^{\twoheadleftarrow}\, t$, we will have some t' with both $t \twoheadrightarrow t'$ and $s\, R\, t'$. Now, if $s \xrightarrow{a} s'$, then there exist states t'', t_1', t_2' such that $t' \twoheadrightarrow_{z,R,s} t_1' \xrightarrow{a} t_2' \twoheadrightarrow_{o,R,s'} t''$ and $s'\, R\, t''$. We also have $t' \twoheadrightarrow_{z,R^{\twoheadleftarrow},s} t_1' \xrightarrow{a} t_2'$, since for each intermediate state t''' in the computation $t \twoheadrightarrow t'$ we always have $s\, R^{\twoheadleftarrow}\, t'''$. And from $s'\, R\, t''$ we can infer $s'\, R^{\twoheadleftarrow}\, t_2'$. In this way we finally obtain $t' \twoheadrightarrow_{z,R^{\twoheadleftarrow},s} t_1' \xrightarrow{a} t_2' \twoheadrightarrow_{o,R^{\twoheadleftarrow},s'} t_2'$ with $s'\, R^{\twoheadleftarrow}\, t_2'$. But since the last weak transition in the obtained weak computation is an empty transition, we also have $t' \twoheadrightarrow_{z,R^{\twoheadleftarrow},s} t_1' \xrightarrow{a} t_2' \twoheadrightarrow_{b,R^{\twoheadleftarrow},s'} t_2'$ with $s'\, R^{\twoheadleftarrow}\, t_2'$, thus proving that $R^{\twoheadleftarrow}$ is also a (z, b)-generic simulation. $\square$

A generic weak simulation game can be obtained from Definition 3.7 by disallowing Spoiler to choose her third option.

We remark that we could use this game characterisation to provide an alternative proof for Proposition 6.13 along the following line of reasoning. Observe that Duplicator can avoid the use of move (2c) even when playing an E-simulation game with ☺ $\in E$. The moves that she plays when playing those games are always valid for the E'-game, where $E' = E \setminus \{☺\}$, hence proving the desired game equivalence.

Note that when playing move (2a) instead of move (2c), Duplicator allows the next challenge from Spoiler to correspond to the reached pair of states (u', v'), instead of some pair (u', v''), with $v' \twoheadrightarrow v''$, that she could propose after playing the sequence of (3c) moves corresponding to that weak computation. In the case of the simulation game SPOILER can, however, not take any advantage from a new challenge starting from (u', v') instead of (u', v''), since her move always uses a transition $u' \xrightarrow{a} u''$, and then DUPLICATOR can start her reply playing the moves corresponding to the weak computation $v' \twoheadrightarrow v''$ again.[4]

Example 6.14. If we now reconsider the example we took from [19] in Figure 3 on page 11, we note that state 0 is not branching simulated by state 5, which can be proved following the same arguments as used in that section. Instead, state 5 is branching simulated by state 0, as the last can copy any move from the former, eventually arriving at states that are trivially equivalent.

A game characterisation of the simulation equivalences can equally straightforwardly be obtained from our definitions, by only allowing SPOILER to choose her third option for her moves (in the bisimulation game) during the first round of the game, and disallowing this option in any subsequent rounds. Of course, the corresponding simulation equivalence relation that one obtains in this way is coarser than the corresponding bisimulation: SPOILER has a much bigger power if she can switch the board at any round.

Similarly, by combining the modification to obtain the gbed-game (Definition 6.6) with the change needed to obtain the simulation game, we could obtain games for the corresponding simulation relations with explicit divergence and the corresponding simulation equivalence by restricting SPOILER's options.

7. A Small Application

The game characterisation we discussed in the preceding sections first and foremost provide a useful operational understanding of the four weak bisimulations. In particular, our games allow for a nice intuitive operational explanation of the *inequivalence* of states: SPOILER's winning strategy essentially guides one to the offending states. Our operational explanation of the inequivalence of states can be used complementary to a distinguishing formula, if available for the equivalence used to compare the two states.[5]

A distinguishing formula is a modal formula that holds for exactly one of the two (inequivalent) transition systems being compared and it provides a high-level explanation for their inequivalence. By high-level, we mean that it will not only hold for one of the two transition systems at hand, but by their nature, it will also hold for every transition system that is equivalent to the one for which it holds. Since such transition systems are in general structurally quite different, the formula can therefore only explain the difference between the two given transition systems in terms of their capabilities and not in terms of their structure.

In contrast, SPOILER's winning strategy is specific for the two given transition systems and therefore provides a much more fine-grained explanation as to the inequivalence of the two transition systems than a distinguishing formula. Such a structure-based operational

[4]In the bisimulation game, of course, SPOILER is much stronger, as she can pose a challenge using any transition $v' \xrightarrow{a} v'''$, changing the side on which she plays.

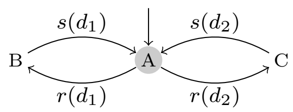
[5]We are not aware of distinguishing formulae for η-bisimulation and delay-bisimulation, nor of any of their divergence-aware variants or their simulation equivalence variants. Distinguishing formulae for branching bisimulation can be found in [19] and distinguishing formulae for branching bisimulation with explicit divergence are studied in [30]. For weak bisimulation, distinguishing formulae are essentially obtained by a 'weak' Hennessy-Milner logic.

explanation of inequivalence can be more accessible than a distinguishing formula, pointing at concrete states that cause the inequivalence. We will forego a precise comparison between the game-theoretical and the various logical approaches, leaving a comparison and deeper connections between the two approaches to future work. Instead, we will illustrate some of the differences between the two techniques through two examples.

As a first example, reconsider the LTSs of Figure 3 on page 11, taken from [19]. The distinguishing formula given in [19] for illustrating that states 0 and 5 are not branching bisimilar is $\neg((tt\langle b\rangle tt)\langle a\rangle tt)$. This formula explains the inequivalence between states 0 and 5 by asserting that state 0 (for which the formula fails) may 'engage in an a-step, while in all intermediate states (state 0 in this case) a b-step is available' [19], whereas this is not true of state 5 (for which the formula holds).[6]

Of course, conceptually, the argument is the same as the strategy we gave on page 10, but it is given in a syntax and semantics that is quite remote from the concept of a transition system. Whereas the formula explains the inequivalence in terms of high-level concepts such as 'behavioural potentials', *i.e.* the ability to execute certain behaviour, the game-theoretical approach works at the level of the excution of concrete states and transitions.

Next we give another example that illustrates how the bbed-game (and, of course, the gbed-game) can be put to use to explain inequivalence of states in a more involved setting. Consider the LTS below, which depicts a simple specification of a one-place buffer for exchanging two types of message (d_1 and d_2).



Suppose one tries to implement this one-place buffer using the Alternating Bit Protocol (see Figure 8), only to find out that states A and 0 are not branching bisimilar with explicit divergence. Note that states A and 0 *are* branching bisimilar, so the difference must be in the lack of divergence in the specification. Indeed, a distinguishing formula states just that: formula $tt\ \langle r(d_1)\rangle\ (\Delta tt)$,[7] holds in state 0 but not in state A. The formula concisely explains the inequivalence, but it does not point at a concrete cause for the divergence in the involved transition systems and which states are involved. In this case, SPOILER's winning strategy can be used to 'play' against the designer of the implementation in a way similar to that of [26], allowing the designer to better understand the reason why this implementation is not satisfactory. By solving the (automatically generated) bbed-game we obtain a winning strategy for player SPOILER, which can be used in an interactive setting as follows:

```
Spoiler moves A --r(d1)--> B
You respond with 0 --r(d1)--> 1
Spoiler switches positions and moves 1 --tau--> 3
You respond by not moving
```

[6]Informally, a formula of the form $\phi\langle a\rangle\psi$ states that there is a finite τ-path on which ϕ holds until action a, after which ψ holds. By nesting these *until*-operators one can state more complex properties: $tt\langle b\rangle tt$ means that a b-action can be 'weakly' executed, and using this formula as the first argument in $((tt\langle b\rangle tt)\langle a\rangle tt)$, we state that a can be fired as first visible action, possibly after an internal computation that only passes intermediate states where b can be weakly executed.

[7]The unary divergence predicate $\Delta\ \phi$ of [30] holds in a state iff it admits an infinite τ-path on which ϕ holds.

```
...
Spoiler moves 19 --tau--> 1
You respond by not moving
You explored all options. You lose.
```

In a similar vein, one can check also that states B and 9 are not branching bisimilar with explicit divergence.

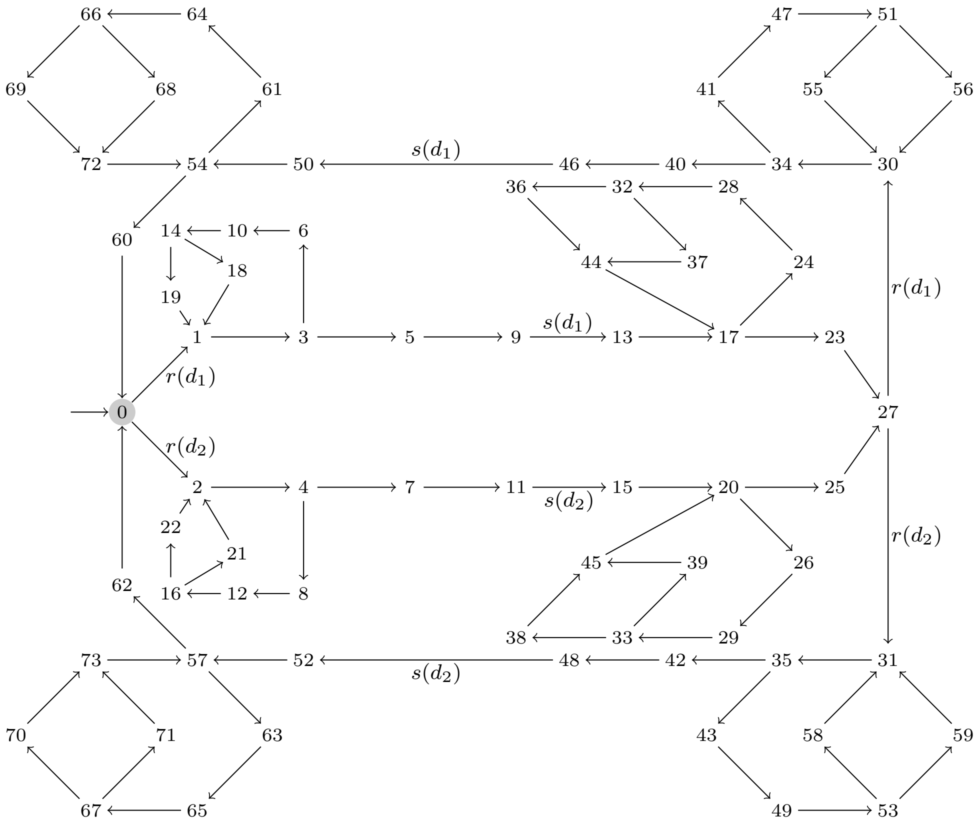


Figure 8: The ABP with two messages; unlabelled transitions are τ transitions.

8. Closing Remarks

In this paper we introduced a generic, game-theoretic definition of weak bisimulation relations. In particular, we showed how our games can be instantiated to obtain games that characterise branching-, η-, delay- and weak bisimilarity. We also illustrated that our generic bisimulation games generalise the branching bisimulation games we presented in [9].

The definition that we have presented does not require any transitive closure of τ-transitions in the game definition, so that a 'local' assessment is obtained when two states are found to be inequivalent. Any sequence of τ-transitions is handled explicitly in the game, in a step-by-step fashion. This enables a more straightforward use of our generic bisimulation games for debugging whenever states are not equivalent.

We generalised our generic bisimulation games to deal with divergence as a first-class citizen: no precomputation of divergences, and subsequent modification of the game, is needed. In particular, we showed that our game, when suitably instantiated, characterises branching bisimulation with explicit divergences, just as it was the case for the game we previously presented in [9].

Additionally, we illustrated the (simple) generalisation of our games to cover simulation relations.

Future work. We have experimented with a prototype of the game-theoretic definitions of branching bisimulation (also with explicit divergence), both using an interactive, command-line application, as well as with a graphical user interface. We intend to make a proper implementation available in the mCRL2 toolset [7].

In this paper, we have used condition D_4 from [30] to obtain an extension sensitive to divergence. As we illustrated in Section 6.2, using any of their other conditions is non-trivial; indeed, it is far from immediate that the use of any of their other conditions would produce equivalent relations. These conditions should be explored further to answer the questions raised in this paper. Additionally, the notion of divergence considered in this paper is not the only notion of divergence considered in the literature. One could, for example, drop the requirement that vertices on divergent paths are related [4]. Game-based characterisations for this and other notions such as *divergence sensitive branching bisimulation* [10], but also notions such as the *next-preserving branching bisimulations* of [34] can be studied. Finally, it would be interesting to extend the approach of [8] and investigate whether the games we have studied in this paper can be applied to Büchi automata or parity games, *i.e.* models other than labelled transition systems.

Acknowledgements and Dedicatory. The authors would like to thank the reviewers for their constructive feedback and their insights which enabled us to improve the paper considerably. The first author wants to dedicate this paper to the memory of his mother Virginia, that passed away during its elaboration.

References

[1] J.R. Andersen, N. Andersen, S. Enevoldsen, M.M. Hansen, K.G. Larsen, S.R. Olesen, J. Srba, and J.K. Wortmann. CAAL: Concurrency Workbench, Aalborg Edition. In *ICTAC*, volume 9399 of *Lecture Notes in Computer Science*, pages 573–582. Springer, 2015.

[2] J.C.M. Baeten and R.J. van Glabbeek. Another look at abstraction in process algebra. In T. Ottmann, editor, *Proc. ICALP'87*, volume 267 of *LNCS*, pages 84–94. Springer, 1987.

[3] T. Basten. Branching bisimilarity is an equivalence indeed! *Inform. Process. Lett.*, 58(3):141–147, May 1996.

[4] J.A. Bergstra, J.W. Klop, and E.R. Olderog. Failures without chaos: A new process semantics for fair abstraction. In *Formal Description of Programming Concepts III - Proceedings of the IFIP TC2/WG 2.2 Working Conference*, pages 77–103. North Holland, 1987.

[5] S.C.C. Blom, W.J. Fokkink, J.F. Groote, I. van Langevelde, B. Lisser, and J.C. van de Pol. μCRL: A toolset for analysing algebraic specifications. In *Proc. CAV'01*, volume 2102 of *LNCS*, pages 250–254, 2001.

[6] P.E. Bulychev, I.V. Konnov, and V.A. Zakharov. Computing (bi)simulation relations preserving CTL*-X for ordinary and fair Kripke structures. *Inst. for Syst. Progr., Russian Acad. of Sci., Math. Meth. and Algor.*, 12, 2007.

[7] S. Cranen, J.F. Groote, J.J.A. Keiren, F.P.M. Stappers, E.P. de Vink, J.W. Wesselink, and T.A.C. Willemse. An overview of the mCRL2 toolset and its recent advances. In *Proc. TACAS'13*, volume 7795 of *LNCS*, pages 199–213, 2013.

[8] S. Cranen, J.J.A. Keiren, and T.A.C. Willemse. Parity game reductions. *Acta Informatica*, pages 1–44, 8 2017.
[9] D. de Frutos Escrig, J.J.A. Keiren, and T.A.C. Willemse. Branching bisimulation games. In E. Albert and I. Lanese, editors, *Proc. FORTE'16*, pages 142–157. Springer, 2016.
[10] R. de Nicola and F.W. Vaandrager. Three logics for branching bisimulation. *J. ACM*, 42(2):458–487, March 1995.
[11] K. Etessami, Th. Wilke, and R.A. Schuller. Fair simulation relations, parity games, and state space reduction for Büchi automata. In *ICALP*, volume 2076 of *Lecture Notes in Computer Science*, pages 694–707. Springer, 2001.
[12] C. Fritz and Th. Wilke. Simulation relations for alternating parity automata and parity games. In *Developments in Language Theory*, volume 4036 of *Lecture Notes in Computer Science*, pages 59–70. Springer, 2006.
[13] H. Garavel, F. Lang, R. Mateescu, and W. Serwe. CADP 2011: a toolbox for the construction and analysis of distributed processes. *Int. Journ. on Softw. Tools for Techn. Transfer*, 15(2):89–107, April 2013.
[14] R. Gerth, R. Kuiper, D. Peled, and W. Penczek. A partial order approach to branching time logic model checking. *Inform. and Comput.*, 150(2):132–152, 1999.
[15] T. Gibson-Robinson, P. Armstrong, A. Boulgakov, and A.W. Roscoe. FDR3: a parallel refinement checker for CSP. *International Journal on Software Tools for Technology Transfer*, 18(2):149–167, 2016.
[16] E. Grädel, W. Thomas, and T. Wilke, editors. *Automata Logics, and Infinite Games*, volume 2500 of *LNCS*. Springer, 2002.
[17] M. Hennessy and G.D. Plotkin. A term model for CCS. In *Mathematical Foundations of Computer Science 1980 (MFCS'80), Proceedings of the 9th Symposium*, volume 88 of *Lecture Notes in Computer Science*, pages 261–274. Springer, 1980.
[18] M. Hutagalung, N. Hundeshagen, D. Kuske, M. Lange, and É. Lozes. Multi-buffer simulations for trace language inclusion. In *GandALF*, volume 226 of *EPTCS*, pages 213–227, 2016.
[19] H. Korver. Computing distinguishing formulas for branching bisimulation. In *Proc. CAV'92*, volume 575, pages 13–23. Springer, 1992.
[20] R. Milner. *A Calculus of Communicating Systems*, volume 92 of *LNCS*. Springer, 1980.
[21] R. Milner. A modal characterisation of observable machine-behaviour. In E. Astesiano and C. Bhm, editors, *CAAP '81*, volume 112 of *LNCS*, pages 25–34. Springer Berlin Heidelberg, 1981.
[22] K.S. Namjoshi. A simple characterization of stuttering bisimulation. In *Proc. FSTTCS'97*, volume 1346 of *LNCS*, pages 284–296. Springer, 1997.
[23] D. Park. Concurrency and automata on infinite sequences. In *Proc. TCS'81*, volume 104 of *LNCS*, pages 167–183. Springer, 1981.
[24] M.A. Reniers, R. Schoren, and T.A.C. Willemse. Results on embeddings between state-based and event-based systems. *Comput. J.*, 57(1):73–92, 2014.
[25] D. Sangiorgi. *Introduction to Bisimulation and Coinduction*. Cambridge Universtity Press, 2012.
[26] P. Stevens and C. Stirling. Practical model-checking using games. In *Proc. TACAS'98*, volume 1384 of *LNCS*, pages 85–101. Springer, 1998.
[27] C. Stirling. Bisimulation, modal logic and model checking games. *Logic Journal of IGPL*, 7(1):103–124, January 1999.
[28] W. Thomas. On the Ehrenfeucht-Fraïssé game in theoretical computer science. In *Proc. TAPSOFT'93*, volume 668 of *LNCS*, pages 559–568. Springer, 1993.
[29] R.J. van Glabbeek. The linear time - Branching time spectrum II. In E. Best, editor, *CONCUR'93*, volume 715 of *LNCS*, pages 66–81. Springer, 1993. An extended preliminary version can be reached at http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.29.1931.
[30] R.J. van Glabbeek, B. Luttik, and N. Trçka. Branching bisimilarity with explicit divergence. *Fundam. Inform.*, 93(4):371–392, 2009.
[31] R.J. van Glabbeek, B. Luttik, and N. Trçka. Computation tree logic with deadlock detection. *Logical Methods in Computer Science*, 5(4), 2009.
[32] R.J. van Glabbeek and W.P. Weijland. Branching time and abstraction in bisimulation semantics. *J. ACM*, 43(3):555–600, May 1996.
[33] D.J. Walker. Bisimulation and divergence. *Inf. Comput.*, 85(2):202–241, 1990.

[34] N. Yatapanage and K. Winter. Next-preserving branching bisimulation. *Theor. Comput. Sci.*, 594:120–142, 2015.

[35] Q. Yin, Y. Fu, C. He, M. Huang, and X. Tao. Branching bisimilarity checking for PRS. In *Proc. ICALP'14*, volume 8573 of *LNCS*, pages 363–374. Springer, 2014.