Hereditary History-Preserving Bisimilarity: Logics and Automata*

Paolo Baldan and Silvia Crafa

Dipartimento di Matematica, Università di Padova, Italy
baldan,crafa@math.unipd.it

Abstract. We study hereditary history-preserving (hhp-) bisimilarity, a canonical behavioural equivalence in the true concurrent spectrum, by means of logics and automata. We first show that hhp-bisimilarity on prime event structures can be characterised in terms of a simple logic whose formulae just observe events in computations and check their executability. The logic suggests a characterisation of hhp-bisimilarity based on history-dependent automata, a formalism for modelling systems with dynamic allocation and deallocation of resources, where the history of resources is traced over time. Prime event structures can be naturally mapped into history-dependent automata in a way that hhp-bisimilarity exactly corresponds to the canonical behavioural equivalence for history-dependent automata.

1 Introduction

Behavioural equivalences play a key role in the formal analysis of system specifications. They can be used to equate specifications that, although syntactically different, denote the same system behaviour, or to formally state that a system enjoys a desired property. A number of behavioural equivalences have been defined which take into account different concurrency features of computations. In particular, true concurrent equivalences (see, e.g., [1]) are a natural choice when one is interested in analysing properties concerning the dependencies between computational steps (e.g. causality). They can be convenient also because they provide some relief to the so-called state-space explosion problem in the analysis of concurrent systems (see, e.g., [2]).

Hereditary history preserving (hhp-)bisimilarity [3], the finest equivalence in the true concurrent spectrum in [1], has been shown to arise as a canonical behavioural equivalence when considering partially ordered computations [4]. True concurrent models, such as Winskel's event structures [5], often describe the behaviour of systems in terms of events in computations and dependency relations between such events, like causal dependency or concurrency. Hhp-bisimilarity then precisely captures the interplay between branching, causality and concurrency. Roughly, hhp-bisimilarity requires that events of one system are simulated by events of the other system with the same causal history and the same

* Work partially supported by the MIUR PRIN project CINA.

J. Garrigue (Ed.): APLAS 2014, LNCS 8858, pp. 469–488, 2014.

concurrency. The last constraint is often captured by means of a sort of backtracking condition: for any two related computations, the computations obtained by reversing a pair of related events must be related too. As a consequence, hhp-bisimilarity, together with other variants of forward-reverse equivalences, are considered appropriate behavioural equivalences for systems with reversible computations [6,7,8].

Recently, the logical characterisation of hhp-bisimilarity has received a renewed interest and corresponding event-based logics have been introduced, where formulae include variables which can be bound to events. The logic $\mathcal{L}$ in [9] explicitly refers to relations between events, namely causality and concurrency. More precisely, $\mathcal{L}$ includes two main operators. The formula $(x, \overline{y} < \mathsf{a}\, z)\varphi$ is satisfied in a state when an a-labelled future event exists, which causally depends on the event bound to x, and is independent from the event bound to y; such an event is bound to variable z and then φ is required to hold. In general, x and y can be replaced by tuples of variables. The formula $\langle z \rangle\, \varphi$ says that the event bound to z is enabled in the current state, and after its execution φ holds. Instead, the logic EIL (Event Identifier Logic) in [10] relies on a backward step modality: the formula $\langle\langle x \rangle\, \varphi$ holds when the event bound to x can be undone and then φ holds. This is similar to the past tense or future perfect modality studied in [4,11,3,12].

In this paper we provide a logical characterisation of hhp-bisimilarity in terms of a simple logic $\mathcal{L}_{\mathbf{0}}$, a core fragment of $\mathcal{L}$, which only predicates over existence and executability of events, without explicitly referring to their dependencies. Formally, the operator $(x, \overline{y} < \mathsf{a}\, z)\varphi$ is replaced by $(\mathsf{a}\, z)\varphi$. Syntactically, $\mathcal{L}_{\mathbf{0}}$ is also a subset of EIL, but it is different in spirit (as quantification is performed only on future events and it does not include a backward modality). In particular, although all such logics characterise hhp-bisimilarity, the modalities of EIL and $\mathcal{L}$ are not interdefinable.

The fact that the logic $\mathcal{L}_{\mathbf{0}}$ allows one to observe and track events in computations suggests a connection with *history-dependent automata (HD-automata)* [13], a computational formalism for modelling systems with dynamic allocation and deallocation of resources, tracing the history of such resources over time. Indeed, by considering events in computations as resources manipulated by automata, we identify a class of HD-automata, called HDE-automata, where prime event structures (PESs) can be naturally mapped, in a way that the canonical behavioural equivalence for HD-automata coincides with hhp-bisimilarity over PESs. More precisely, transitions of HDE-automata correspond to planning an activity or event (which could be not immediately executable due to unsatisfied dependencies with other activities), executing a previously planned activity and dismissing a planned activity (without executing it). We provide an encoding of any prime event structure $\mathcal{E}$ into an HDE-automaton $\mathcal{H}(\mathcal{E})$ such that two prime event structures are hhp-bisimilar if and only if the corresponding HDE-automata are bisimilar. The proof relies on a logical characterisation of bisimilarity on HDE-automata in terms of a logic $\mathcal{L}_{hd}$, a slight variant of the logic $\mathcal{L}_{\mathbf{0}}$, which adds an operator for deallocation, i.e., for forgetting an event

planned and not yet executed. Mappings of logic $\mathcal{L}_{\mathbf{0}}$ into $\mathcal{L}_{hd}$ and back are provided, in a way that a PES $\mathcal{E}$ satisfies a formula in $\mathcal{L}_{\mathbf{0}}$ if and only if $\mathcal{H}(\mathcal{E})$ satisfies the corresponding formula in $\mathcal{L}_{hd}$ and vice versa. Although developed for a specific class of HD-automata, in our opinion the logical characterisation of HD-bisimilarity has an interest which goes beyond the specific application in this paper and deserves to be further investigated.

Moreover, our characterisation of hhp-bisimilarity in terms of HD-automata, besides shedding light on the nature of this behavioural equivalence, can be helpful in studying the decidability boundary for hhp-bisimilarity, which is undecidable for many basic models of concurrency, even in the finite state case (e.g., it is known that hhp-bisimilarity is undecidable for safe finite Petri nets [14]). Indeed, the characterisation in terms of HD-automata naturally suggests effective approximations of hhp-bisimilarity, which can be obtained by establishing bounds k on the distance in the future of planned events. The detailed study of such approximations is postponed to the extended version of the paper. We focus here on an insightful investigation about the logical and the automata-theoretic characterisations of hhp-bisimilarity.

The rest of the paper is structured as follows. In Section 2 we review the definition of hhp-bisimilarity over prime event structures. In Section 3 we define the logic $\mathcal{L}_{\mathbf{0}}$ and show that hhp-bisimilarity is the logical equivalence induced by $\mathcal{L}_{\mathbf{0}}$ on (image finite) PESs. In Section 4 we study HDE-automata: the class of HD-automata operating over resources which can be seen as activities or events in a computation. In Section 5 we provide a bisimilarity-preserving encoding of prime event structures into HDE-automata. In Section 6 we comment on some related work and outline future research.

2 Event Structures and hhp-Bisimilarity

Prime event structures [5] are a widely known model of concurrency. They describe the behaviour of a system in terms of events and dependency relations between such events. Throughout the paper $\mathbb{E}$ is a fixed countable set of events, Λ a set of labels ranged over by $\mathsf{a}, \mathsf{b}, \mathsf{c} \ldots$ and $\lambda : \mathbb{E} \to \Lambda$ a labelling function.

Definition 1 (prime event structure). *A (Λ-labelled)* prime event structure *(*PES*) is a tuple $\mathcal{E} = \langle E, \leq, \# \rangle$, where $E \subseteq \mathbb{E}$ is the set of* events *and $\leq$, $\#$ are binary relations on E, called* causality *and* conflict *respectively, such that:*

1. *$\leq$ is a partial order and $\lceil e \rceil = \{e' \in E \mid e' \leq e\}$ is finite for all $e \in E$;*
2. *$\#$ is irreflexive, symmetric and hereditary with respect to $\leq$, i.e., for all $e, e', e'' \in E$, if $e \# e' \leq e''$ then $e \# e''$.*

In the following, we will assume that the components of an event structure $\mathcal{E}$ are named as in the definition above, possibly with subscripts.

Definition 2 (consistency, concurrency). *Let $\mathcal{E}$ be a* PES*. We say that $e, e' \in E$ are* consistent*, written $e \frown e'$, if $\neg(e \# e')$. A subset $X \subseteq E$ is called* consistent *if $e \frown e'$ for all $e, e' \in X$. We say that e and e' are* concurrent*, written $e \,||\, e'$, if $e \frown e'$ and $\neg(e \leq e')$, $\neg(e' \leq e)$.*

Causality and concurrency will be sometimes used on set of events. Given $X \subseteq E$ and $e \in E$, by $X < e$ we mean that for all $e' \in X$, $e' < e$. Similarly $X \,||\, e$, resp. $X \frown e$, means that for all $e' \in X$, $e' \,||\, e$, resp. $e' \frown e$.

The concept of (concurrent) computation for event structures is captured by the notion of configuration.

Definition 3 (configuration). *Let $\mathcal{E}$ be a* PES. *A* (finite) configuration *in $\mathcal{E}$ is a (finite) consistent subset of events $C \subseteq E$ closed w.r.t. causality (i.e., $\lceil e \rceil \subseteq C$ for all $e \in C$). The set of finite configurations of $\mathcal{E}$ is denoted by $\mathcal{C}(\mathcal{E})$.*

Observe that the empty set of events $\emptyset$ is always a configurations, which can be interpreted as the initial state of the computation. Hereafter, unless explicitly stated otherwise, all configurations will be assumed to be finite.

Definition 4 (residual). *Let $\mathcal{E}$ be a* PES. *For a configuration $C \in \mathcal{C}(\mathcal{E})$, the* residual *of $\mathcal{E}$ after C, is defined as $\mathcal{E}[C] = \{e \mid e \in E \setminus C \ \wedge \ C \frown e\}$.*

Concurrent behavioural equivalences can then be defined on the transition system where configurations are states.

Definition 5 (transition system). *Let $\mathcal{E}$ be a* PES *and let $C \in \mathcal{C}(\mathcal{E})$. Given $e \in \mathcal{E}[C]$, if $C \cup \{e\} \in \mathcal{C}(\mathcal{E})$ then we write $C \xrightarrow{e} C \cup \{e\}$.*

A PES $\mathcal{E}$ is called *image finite* if for every $C \in \mathcal{C}(\mathcal{E})$ and $\mathsf{a} \in \Lambda$, the set of events $\{e \in E \mid C \xrightarrow{e} C' \ \wedge \ \lambda(e) = \mathsf{a}\}$ is finite. All the PESs considered in this paper will be assumed to be image finite, a standard requirement for getting a logical characterisation of a behavioural equivalence based on a finitary logic.

Several equivalences have been defined in order to capture the concurrency features of a system to different extents (see, e.g., [1]). Hereditary history-preserving (hhp-)bisimilarity arises as a canonical equivalence for PESs [4] which fully takes into account the interplay between causality, concurrency and nondeterminism of events.

We need to fix some further notation. A consistent subset $X \subseteq E$ of events will be often seen as a *pomset* (partially ordered multiset) $(X, \leq_X, \lambda_X)$, where $\leq_X$ and λ_X are the restrictions of $\leq$ and λ to X. Given $X, Y \subseteq E$ we will write $X \sim Y$ if X and Y are isomorphic as pomsets and write $f : X \xrightarrow{\sim} Y$ for a pomset isomorphism.

Definition 6 (posetal product). *Given two* PESs *$\mathcal{E}_1$, $\mathcal{E}_2$, the* posetal product *of their configurations, denoted $\mathcal{C}(\mathcal{E}_1) \bar{\times} \mathcal{C}(\mathcal{E}_2)$, is defined as*

$$\{(C_1, f, C_2) \mid C_1 \in \mathcal{C}(\mathcal{E}_1),\ C_2 \in \mathcal{C}(\mathcal{E}_2),\ f : C_1 \xrightarrow{\sim} C_2\}$$

A subset $R \subseteq \mathcal{C}(\mathcal{E}_1) \bar{\times} \mathcal{C}(\mathcal{E}_2)$ is called a posetal relation. *We say that R is* downward closed *whenever for any $(C_1, f, C_2), (C_1', f', C_2') \in \mathcal{C}(\mathcal{E}_1) \bar{\times} \mathcal{C}(\mathcal{E}_2)$, if $(C_1, f, C_2) \subseteq (C_1', f', C_2')$ pointwise and $(C_1', f', C_2') \in R$ then $(C_1, f, C_2) \in R$.*

Given a function $f : X_1 \to X_2$ we will use the notation $f[x_1 \mapsto x_2] : X_1 \cup \{x_1\} \to X_2 \cup \{x_2\}$ for the function defined by $f[x_1 \mapsto x_2](x_1) = x_2$ and $f[x_1 \mapsto x_2](z) = f(z)$ for $z \in X_1 \setminus \{x_1\}$. Note that this can represent an update of f, when $x_1 \in X_1$, or an extension of its domain, otherwise.

Definition 7 ((hereditary) history-preserving bisimulation). *A* history-preserving (hp-)bisimulation *is a posetal relation* $R \subseteq \mathcal{C}(\mathcal{E}_1) \bar{\times} \mathcal{C}(\mathcal{E}_2)$ *such that if* $(C_1, f, C_2) \in R$ *and* $C \xrightarrow{e_1} C_1'$ *then* $C_2 \xrightarrow{e_2} C_2'$, *with* $(C_1', f[e_1 \mapsto e_2], C_2') \in R$, *and vice versa. We say that* $\mathcal{E}_1$, $\mathcal{E}_2$ *are* history preserving (hp-)bisimilar *and write* $\mathcal{E}_1 \sim_{hp} \mathcal{E}_2$ *if there exists a hp-bisimulation* R *such that* $(\emptyset, \emptyset, \emptyset) \in R$.

A hereditary history-preserving (hhp-)bisimulation *is a downward closed hp-bisimulation. When* $\mathcal{E}_1, \mathcal{E}_2$ *are* hereditary history-preserving (hhp-)bisimilar *we write* $\mathcal{E}_1 \sim_{hhp} \mathcal{E}_2$.

3 A Logic for hhp-Bisimilarity

In this section we introduce the syntax and the semantics of a logic $\mathcal{L}_\mathbf{0}$, used to characterise hhp-bisimilarity. The formulae of $\mathcal{L}_\mathbf{0}$ predicate over existence and executability of events in computations. As already mentioned, $\mathcal{L}_\mathbf{0}$ is a small core of the logic $\mathcal{L}$ in [9], where the operators do not explicitly refer to the dependencies between events. Still $\mathcal{L}_\mathbf{0}$ is sufficiently powerful to capture such dependencies and its logical equivalence is the same as that of the full logic in that they both correspond to hhp-bisimilarity.

Definition 8 (syntax). *Let Var be a countable set of variables ranged over by* $x, y, z, \ldots$ *The logic* $\mathcal{L}_\mathbf{0}$ *over the set of labels* Λ *is defined by the following syntax:*

$$\varphi \ ::= \ \mathsf{T} \ \mid \ \varphi \wedge \varphi \ \mid \ \neg\varphi \ \mid \ (\mathsf{a}\, z)\, \varphi \ \mid \ \langle z \rangle\, \varphi$$

Disjunction $\varphi \vee \psi$ is defined, as usual, by duality as the formula $\neg(\neg\varphi \wedge \neg\psi)$. Similarly, we write F for $\neg\mathsf{T}$.

The operator $(\mathsf{a}\, z)$ acts as a binder for the variable z. Accordingly, the free variables of a formula φ are defined as follows:

$$fv((\mathsf{a}\, z)\varphi) = fv(\varphi) \setminus \{z\} \qquad fv(\langle z \rangle\, \varphi) = fv(\varphi) \cup \{z\}$$

$$fv(\mathsf{T}) = \emptyset \quad fv(\neg\varphi) = fv(\varphi) \quad fv(\varphi_1 \wedge \varphi_2) = fv(\varphi_1) \cup fv(\varphi_2)$$

Formulae are considered up to α-conversion of bound variables. The logic $\mathcal{L}_\mathbf{0}$ is interpreted over PESs. In particular, the satisfaction of a formula is defined with respect to pairs (C, η), where $C \in \mathcal{C}(\mathcal{E})$ is a configuration representing the state of the computation, and $\eta : Var \to E$ is a function, called *environment*, that maps the free variables of φ to events.

Since, intuitively, a formula φ describes possible future computations, the environment should map variables to events consistent with C and pairwise consistent. The first condition ensures that the formula actually refers to events that belong to the future (residual) of the current state. The second condition prevents the direct observation of conflicts, in accordance with the observational power of hhp-bisimilarity (Some examples are provided below, after defining the semantics.) Formally, this is captured by the notion of legal pair.

Definition 9 (legal pair). *Given a* PES $\mathcal{E}$, *let* $Env_\mathcal{E}$ *denote the set of environments, i.e., of functions* $\eta : Var \to E$. *Given a formula* φ *in* $\mathcal{L}_\mathbf{0}$, *a pair* $(C, \eta) \in \mathcal{C}(\mathcal{E}) \times Env_\mathcal{E}$ *is* legal *for* φ *if* $C \cup \eta(fv(\varphi))$ *is a consistent set of events. We write* $lp_\mathcal{E}(\varphi)$ *for the set of legal pairs for* φ.

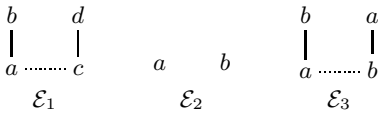

Fig. 1. The PES $\mathcal{E}_1$ for $a.b + c.d$, $\mathcal{E}_2$ for $a \mid b$ and $\mathcal{E}_3$ for $a.b + b.a$

We omit the subscripts and write *Env* and $lp(\varphi)$ when the PES $\mathcal{E}$ is clear from the context.

Definition 10 (semantics). *Let $\mathcal{E}$ be a* PES. *The denotation of a formula φ, written $\{\!|\varphi|\!\}^{\mathcal{E}} \in 2^{\mathcal{C}(\mathcal{E})\times Env}$ is defined inductively as follows:*

$$
\begin{aligned}
\{\!|\mathsf{T}|\!\}^{\mathcal{E}} &= \mathcal{C}(\mathcal{E}) \times Env_{\mathcal{E}} \\
\{\!|\varphi_1 \wedge \varphi_2|\!\}^{\mathcal{E}} &= \{\!|\varphi_1|\!\}^{\mathcal{E}} \cap \{\!|\varphi_2|\!\}^{\mathcal{E}} \cap lp(\varphi \wedge \psi) \\
\{\!|\neg\varphi|\!\}^{\mathcal{E}} &= lp(\varphi) \setminus \{\!|\varphi|\!\}^{\mathcal{E}} \\
\{\!|(\mathsf{a}\, z)\,\varphi|\!\}^{\mathcal{E}} &= \{(C,\eta) \mid \exists e \in \mathcal{E}[C].e \frown \eta(fv(\varphi)\setminus\{z\}) \\
&\qquad\qquad \lambda(e) = \mathsf{a} \wedge\ (C, \eta[z \mapsto e]) \in \{\!|\varphi|\!\}^{\mathcal{E}}\} \\
\{\!|\langle z\rangle\ \varphi|\!\}^{\mathcal{E}} &= \{(C,\eta) \mid\ C \xrightarrow{\eta(z)} C' \ \wedge\ (C',\eta) \in \{\!|\varphi|\!\}^{\mathcal{E}}\ \}
\end{aligned}
$$

When $(C,\eta) \in \{\!|\varphi|\!\}^{\mathcal{E}}$ we say that the PES *$\mathcal{E}$ satisfies the formula φ in the configuration C and environment $\eta : Var \to E$, and write $\mathcal{E}, C \models_\eta \varphi$. For closed formulae φ, we write $\mathcal{E} \models \varphi$, when $\mathcal{E}, \emptyset \models_\eta \varphi$ for some η.*

In words, the formula $(\mathsf{a}\, z)\,\varphi$ holds in (C,η) when in the future of the configuration C there is an a-labelled event e consistent with the events already observed (which are bound to free variables in φ) and binding such event e to the variable z, the formula φ holds. The formula $\langle z\rangle\,\varphi$ states that the event bound to z is currently enabled, hence it can be executed producing a new configuration which satisfies the formula φ. An environment η is a total function, but it can be shown that the semantics of a formula φ depends only on the value of the environment on the free variables $fv(\varphi)$. In particular, for closed formulae the environment is irrelevant. Moreover, it can be easily seen that α-equivalent formulae have the same semantics.

As an example, consider the PES $\mathcal{E}_1$ in Fig. 1 corresponding to the CCS process $a.b + c.d$, where dotted lines represent immediate conflict and the causal order proceeds upwards along the straight lines. The empty configuration satisfies the formula $\varphi = (\mathsf{b}\, x)\mathsf{T}$, i.e., $\mathcal{E}_1 \models \varphi$ since in the future of the empty configuration there is a b-labelled event. However $\mathcal{E}_1 \not\models (\mathsf{b}\, x)\langle x\rangle\, \mathsf{T}$ since such event is not immediately executable.

Observe also that $\mathcal{E}_1 \models (\mathsf{b}\, x)\mathsf{T} \wedge (\mathsf{d}\, y)\mathsf{T}$, since there are two possible (incompatible) future computations starting from the empty configuration that contain, respectively, a b-labelled and a d-labelled event. For a similar reason, we have

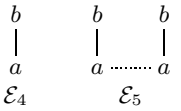

Fig. 2. The PES $\mathcal{E}_4$ for $a.b$, $\mathcal{E}_5$ for $a.b + a.b$.

also $\mathcal{E}_1 \models (\mathsf{a}\,x)\langle x\rangle\,\mathsf{T} \wedge (\mathsf{c}\,y)\langle y\rangle\,\mathsf{T}$. Finally observe that $\mathcal{E}_1 \models (\mathsf{a}\,x)(\mathsf{c}\,y)\mathsf{T}$ since in this case, after binding the variable x to the a-labelled event, we can bind y to the c-labelled event because x is not free in the remaining subformula T.

As a further example, consider the PESs $\mathcal{E}_2$ and $\mathcal{E}_3$ in Fig. 1, corresponding to the CCS processes $a \mid b$ and $a.b + b.a$, respectively. They are distinguished by the formula $(\mathsf{a}\,x)(\mathsf{b}\,y)(\langle x\rangle\,\langle y\rangle\,\mathsf{T} \wedge \langle y\rangle\,\langle x\rangle\,\mathsf{T})$ that states that there are two events, labelled a and b, that can be executed in any order. The formula is satisfied by the first but not by the second PES. In a similar way, the processes $a \mid a$ and $a.a$ are distinguished by the formula $(\mathsf{a}\,x)(\mathsf{a}\,y)(\langle x\rangle\,\langle y\rangle\,\mathsf{T} \wedge \langle y\rangle\,\langle x\rangle\,\mathsf{T})$.

On the other hand, the PESs $\mathcal{E}_4$ and $\mathcal{E}_5$ in Fig. 2, corresponding to the processes $a.b$ and $a.b + a.b$, are hhp-equivalent; accordingly, they both satisfy the formula $\varphi_1 = (\mathsf{a}\,x)(\mathsf{a}\,y)\mathsf{T}$ and falsify $\varphi_2 = (\mathsf{a}\,x)(\mathsf{a}\,y)\langle x\rangle\,\langle y\rangle\,\mathsf{T}$. In particular, for $\mathcal{E}_4$ to satisfy φ_1 both x and y must be bound to the unique a-labelled event. These PESs can be also used for clarifying the need of restricting to legal pairs in the semantics. Consider the formula $\varphi = (\mathsf{a}\,x)(\mathsf{b}\,y)\langle x\rangle\,\neg\langle y\rangle\,\mathsf{T}$. While, clearly, $\mathcal{E}_4 \not\models \varphi$, one could believe that $\mathcal{E}_5 \models \varphi$ since after binding the variable x to the right a-labelled event, we could think of binding y to the left b-labelled event, thus satisfying the remaining subformula $\langle x\rangle\,\neg\langle y\rangle\,\mathsf{T}$. However, this is not correct: since x occurs free in the subformula $\langle x\rangle\,\neg\langle y\rangle\,\mathsf{T}$, the event bound to y must be consistent to that bound to x in order to lead to a legal pair, hence the only possibility is to choose the b-labelled event caused by that bound to x.

Roughly speaking, the logic $\mathcal{L}_\mathbf{0}$ observes conflicting futures, as long as conflicting events are kept separate and not combined in a computation. This corresponds to the observational power of hhp-bisimilarity, which captures the interplay between branching and causality/concurrency without explicitly observing conflicts. We observe that the fragment $\mathcal{L}_\mathbf{0}$ is less expressive than the full logic $\mathcal{L}$. For instance, it can be shown that the formula $(\mathsf{a}\,x)(x < \mathsf{a}\,y)\mathsf{T}$ in $\mathcal{L}$, which states the existence of two causally dependent a-labelled events at arbitrary causal distance, is not encodable by a finite formula of $\mathcal{L}_\mathbf{0}$. Still, $\mathcal{L}_\mathbf{0}$ is sufficiently expressive to capture the same logical equivalence of $\mathcal{L}$, i.e., hhp-bisimilarity.

In the following we will denote lists of variables like $x_1, ..., x_n$ by $\boldsymbol{x}$.

Theorem 1 (hhp-bisimilarity, logically). *Let $\mathcal{E}_1, \mathcal{E}_2$ be two* PES*s. Then $\mathcal{E}_1 \sim_{hhp} \mathcal{E}_2$ iff $\mathcal{E}_1$ and $\mathcal{E}_2$ satisfy the same closed formulae in $\mathcal{L}_\mathbf{0}$.*

Proof (Sketch). The only if part follows from [9, Theorem 1], since the logic $\mathcal{L}_\mathbf{0}$ is a fragment of $\mathcal{L}$. For the converse implication, fix a surjective environment $\eta_1 : \mathit{Var} \to E_1$. Then given an event $e \in E_1$, we let x_e denote a chosen variable

such that $\eta_1(x_e) = e$. For a configuration $C_1 = \{e_1, \ldots, e_n\}$ we denote by X_{C_1} the set of variables $\{x_{e_1}, \ldots, x_{e_n}\}$.
Then one can prove that the posetal relation $R \subseteq \mathcal{C}(\mathcal{E}_1) \bar{\times} \mathcal{C}(\mathcal{E}_2)$ defined by:

$$R = \{ (C_1, f, C_2) \mid \forall \varphi \in \mathcal{L}_{\mathbf{0}}.\ fv(\varphi) \subseteq X_{C_1} \quad (\mathcal{E}_1, \emptyset \models_{\eta_1} \varphi \textit{ iff } \mathcal{E}_2, \emptyset \models_{f \circ \eta_1} \varphi) \} \tag{1}$$

is a hhp-bisimulation. Above, given an isomorphism of pomsets $f : C_1 \to C_2$, we denote by $f \circ \eta_1$ an environment such that $f \circ \eta_1(x) = f(\eta_1(x))$ for $x \in X_{C_1}$ and $f \circ \eta_1(x)$ has any value, otherwise (the semantics of φ only depends on the value of the environment on $fv(\varphi)$ and $fv(\varphi) \subseteq X_{C_1}$ by construction). Note that R relates two configurations C_1 and C_2 when the same formulae φ are satisfied by the empty configuration (rather than by the configurations C_1 and C_2 themselves). The formulae φ considered in (1) refer to events in C_1 and in C_2 by means of their free variables. This is according to the intuition that hhp-bisimilarity does not only compare the future of two configurations but also their alternative evolutions, that is evolutions from the past. □

Similarly to what has been done in [9] for the full logic $\mathcal{L}$, one can identify fragments of $\mathcal{L}_{\mathbf{0}}$ that characterise various other behavioural equivalences in the true concurrent spectrum [1]. First of all notice that the standard Hennessy-Milner logic can be recovered as the following fragment of $\mathcal{L}_{\mathbf{0}}$, where whenever we state the existence of an event we are forced to execute it:

$$\varphi \ ::= \ \mathsf{T} \ \mid \ \varphi \wedge \varphi \ \mid \ \neg\varphi \ \mid \ (\mathsf{a}\, x)\langle x \rangle\, \varphi$$

In such a fragment variables are irrelevant: the formula $(\mathsf{a}\, x)\langle x \rangle\, \varphi$ states the existence of an a-labelled event, which is immediately executable from the current configuration and whose execution produces a new configuration in which φ holds. The event is bound to variable x which, however, is no longer referred to in the formula. Hence $(\mathsf{a}\, x)\langle x \rangle$ is completely analogous to the diamond modality of standard Hennessy Milner logic and the induced logical equivalence is (interleaving) bisimilarity [15].

Along the lines of [9, Theorem 4], one can prove that history-preserving bisimilarity (Definition 7) corresponds to the logical equivalence induced by the following fragment of $\mathcal{L}_{\mathbf{0}}$:

$$\varphi \ ::= \ \mathsf{T} \ \mid \ \varphi \wedge \varphi \ \mid \ \neg\varphi \ \mid \ \langle\!| \boldsymbol{x}, \overline{\boldsymbol{y}} < \mathsf{a}\, z |\!\rangle \varphi$$

where $\boldsymbol{x}$, $\boldsymbol{y}$ are lists of variables and $\langle\!| \boldsymbol{x}, \overline{\boldsymbol{y}} < \mathsf{a}\, z |\!\rangle \varphi$ denotes the formula

$$(\mathsf{a}\, z)(\langle \boldsymbol{x} \rangle\, \langle z \rangle\, \langle \boldsymbol{y} \rangle\, \mathsf{T} \ \wedge \ \langle \boldsymbol{x} \rangle\, \langle \boldsymbol{y} \rangle\, \mathsf{T} \ \wedge \ \bigwedge_{\boldsymbol{x}' \subset \boldsymbol{x}} \neg \langle \boldsymbol{x}' \rangle\, \langle z \rangle\, \mathsf{T} \ \wedge \ \varphi). \tag{2}$$

Above, given a list of variables $\boldsymbol{x} = x_1 \ldots x_n$ the abbreviation $\langle \boldsymbol{x} \rangle$ is used as a shortcut for $\langle x_1 \rangle \ldots \langle x_n \rangle$. Intuitively, the formula (2) states the existence of an a-labelled event, which is bound to z, that causally depends on the events bound to $\boldsymbol{x}$ and that is concurrent with the events bound to $\boldsymbol{y}$. In fact, z can be

executed only after $\boldsymbol{x}$, while $\boldsymbol{y}$ can be executed after or before z. The event is required to be immediately executable, and once executed, formula φ holds. For the above to work, the events bound to $\boldsymbol{x}$ and $\boldsymbol{y}$ must form a $\leq$-closed set, i.e., $\lceil \eta(w) \rceil \subseteq \eta(\boldsymbol{x} \cup \boldsymbol{y})$ for any $w \in \boldsymbol{x} \cup \boldsymbol{y}$. More formally, it is not difficult to prove that $\{\!|\langle\!\langle \boldsymbol{x}, \overline{\boldsymbol{y}} < \mathsf{a}\, z \rangle\!\rangle \varphi |\!\}^{\mathcal{E}}$ is

$$\begin{aligned}\{\langle C, \eta\rangle \mid \exists e \in \mathcal{E}[C].\ & e \frown \eta(fv(\varphi) \setminus \{z\}) \wedge \\ & C \xrightarrow{e} C \cup \{e\} \ \wedge\ \lambda(e) = \mathsf{a} \wedge \\ & \eta(\boldsymbol{x}) < e \ \wedge\ \eta(\boldsymbol{y}) \,||\, e \ \wedge\ \eta(\boldsymbol{x} \cup \boldsymbol{y}) \leq\text{-closed} \wedge \\ & \langle C \cup \{e\}, \eta[z \mapsto e]\rangle \in \{\!|\varphi|\!\}^{\mathcal{E}}\}\end{aligned}$$

Incidentally, this derived operator illustrates how $\mathcal{L}_{\mathbf{0}}$ formulae can be used to express causal (in)dependence between events.

The fact that this fragment has a reduced expressivity, corresponding exactly to hd-bisimilarity can be intuitively explained as follows. As noticed above, a formula in the fragment can "observe" an event only by executing it. Hence the observation of an event automatically discards all future conflictual events. As a consequence it is not possible to observe alternative conflicting futures involving common events, namely the fragment cannot fully describe the interplay between causality/concurrency and branching as required for hhp-bisimilarity. Still, it allows one to capture the dependencies of the observed events with previously executed events, a capability which corresponds to the observational power of hp-bisimilarity.

Analogously, fragments of $\mathcal{L}_{\mathbf{0}}$ inducing pomset and step bisimilarity can be identified.

4 History-Dependent Automata over Events

The logic $\mathcal{L}_{\mathbf{0}}$ for hhp-bisimilarity, singled out in the previous section, allows one to trace events in computations and check their executability. This hints at a connection with HD-automata, a generalised model of automata that has been indeed introduced to describe systems with dynamic allocation and deallocation of resources, tracing the history of such resources over time [13]. In this section we lay the basis of such a connection by identifying a class of HD-automata where PESs can be naturally mapped and providing a logical characterisation of bisimilarity for this class of automata in terms of a mild extension of $\mathcal{L}_{\mathbf{0}}$.

4.1 HDE-Automata and HD-Bisimilarity

HD-automata extend ordinary automata in order to manipulate resources generically identified as names. The allocation of a resource is modelled by the generation of a fresh name and the usage of a resource in a transition is modelled by observing the corresponding name in the transition label. Concretely, with respect to an ordinary automaton, states of an HD-automaton are enriched with a set of local names corresponding to the resources that are active at that states.

Transitions, in turn, modify these sets and explicitly trace the correspondence between the local names of the source and the target states.

We introduce a class of HD-automata, referred to as HDE-automata, where PESs will be naturally encodable. In HDE-automata the names can be thought of as activities or events in a computation. HDE transitions are of three kinds: $\mathtt{plan}(e)\,, \mathtt{exec}(e)\,, \mathtt{drop}(e)$ which can be interpreted, respectively, as planning an activity or event e (which might be not immediately executable due to unsatisfied dependencies with other activities), executing a previously planned activity and dismissing a planned activity (without executing it).

Formally, as before, we fix a countable set $\mathbb{E}$ whose elements are thought of as activities, labelled by $\lambda : \mathbb{E} \to \Lambda$. Given two subsets $A_1, A_2 \subseteq \mathbb{E}$, a *labelled bijection*, denoted $\delta : A_1 \xrightarrow{\sim} A_2$, is a bijection such that for any $e_1 \in A_1$, it holds that $\lambda(e_1) = \lambda(\delta(e_1))$. Let $R(\mathbb{E})$ be the set of *renamings*, i.e., label preserving partial injective functions $\rho : \mathbb{E} \to \mathbb{E}$. Given $\rho \in R(\mathbb{E})$, we write $dom(\rho)$ and $cod(\rho)$ for the domain and codomain of ρ, respectively. The set of labels for the automata transitions is $L(\mathbb{E}) = \{\mathtt{plan}(e)\,, \mathtt{exec}(e)\,, \mathtt{drop}(e) \mid e \in \mathbb{E}\}$.

Definition 11 (HDE-automata). *A* HDE-automaton $\mathcal{H}$ *is a tuple* $\langle Q, n, q_0, \to\rangle$ *where* Q *is a set of states,* $n : Q \to 2^{\mathbb{E}}$ *associates with each state a set of activities and* $\to \subseteq Q \times L(\mathbb{E}) \times R(\mathbb{E}) \times Q$ *is a transition relation, written* $q \xrightarrow{\ell}_{\rho} q'$ *for* $(q, \ell, \rho, q') \in \to$*, such that* $dom(\rho) \subseteq n(q')$ *and* $cod(\rho) \subseteq n(q)$ *(hence* ρ *is a partial injection* $n(q') \to n(q)$*) and*

- *if* $q \xrightarrow{\mathtt{plan}(e)}_{\rho} q'$ *then* $cod(\rho){=}n(q)$, $dom(\rho){=}n(q') \setminus \{e\}$*;*
- *if* $q \xrightarrow{\mathtt{exec}(e)}_{\rho} q'$ *or* $q \xrightarrow{\mathtt{drop}(e)}_{\rho} q'$ *then* $cod(\rho){=}n(q) \setminus \{e\}$ *and* $dom(\rho){=}n(q')$*.*

For a $\mathtt{plan}(e)$ transition the mapping ρ is a bijection between $n(q)$ and $n(q')\backslash\{e\}$. Intuitively, e is the newly planned activity, while $n(q')\backslash\{e\}$ represents, via the renaming ρ, activities already planned in q. In an $\mathtt{exec}(e)$ transition the activity e is executed, while in a $\mathtt{drop}(e)$ transition the activity e is dropped without being executed. In both cases the other activities planned in the source state are kept, and the correspondence between source and target is established by ρ which is a bijection between $n(q)\backslash\{e\}$ and $n(q')$.

Note that when dealing with event structures, states of a computation are given by configurations, namely sets of events which have been already executed. Logic $\mathcal{L}_{\mathbf{0}}$ observes events in the future of a configuration, but these are not part of the state and are implicitly garbage collected when they are no longer referred by the formula. Instead, the states of a HDE-automaton have a richer structure as they explicitly include a set of activities planned but not yet executed (which intuitively correspond to events observed and not yet executed). As a consequence, also dismissing a planned activity is an explicit operation which requires a $\mathtt{drop}(\cdot)$ transition.

We write $q \to_{\rho} q'$ when $q \xrightarrow{\ell}_{\rho} q'$ for some label $\ell \in L(\mathbb{E})$, and we denote by $\to^*_{\rho}$ the reflexive and transitive closure of the transition relation, with ρ resulting as the composition of the involved renamings, i.e., $q \to^*_{id} q$ and if $q \to^*_{\rho} q' \to_{\rho'} q''$ then $q \to^*_{\rho' \circ \rho} q''$.

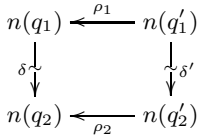

Fig. 3. HD-bisimulation

The theory of HD-automata [13] provides a notion of behavioural equivalence, which we specialise in the following to the case of HDE-automata. First, according to the general theory, it is not restrictive to assume that all HDE-automata are *irredundant*, i.e. that all names occurring in a state are eventually used. Actually, we work with a slightly strengthened notion of irredundancy, i.e., we will assume that for any $e \in n(q)$ there exists a state reachable from q where e can be executed. Formally, we assume that for any $q \in Q$ and any $e \in n(q)$ there exists some $q' \in Q$ such that $q \rightarrow_\rho^* q'$ and $q' \xrightarrow{\mathtt{exec}(e')}_{\rho'} q''$ with $\rho(e') = e$.

Definition 12 (HD-bisimilarity). *Let $\mathcal{H}_1$ and $\mathcal{H}_2$ be two HDE-automata. A* HD-bisimulation *is a relation*

$$R = \{(q_1, \delta, q_2) \mid q_1 \in Q_1 \ \wedge \ q_2 \in Q_2 \ \wedge \ \delta : n_1(q_1) \xrightarrow{\sim} n_2(q_2)\}$$

such that, whenever $(q_1, \delta, q_2) \in R$,

1. *if $q_1 \xrightarrow{\mathtt{plan}(e_1)}_{\rho_1} q'_1$, then there exists a transition $q_2 \xrightarrow{\mathtt{plan}(e_2)}_{\rho_2} q'_2$ such that $(q'_1, \delta', q'_2) \in R$;*
2. *if $q_1 \xrightarrow{\mathtt{exec}(e_1)}_{\rho_1} q'_1$, resp. $q_1 \xrightarrow{\mathtt{drop}(e_1)}_{\rho_2} q'_1$, then there exists a transition $q_2 \xrightarrow{\mathtt{exec}(e_2)}_{\rho_1} q'_2$, resp. $q_2 \xrightarrow{\mathtt{drop}(e_2)}_{\rho_2} q'_2$, such that $\delta(e_1){=}e_2$ and $(q'_1, \delta', q'_2){\in}R$;*

where both for 1) and 2) it holds $\rho_2 \circ \delta' = \delta \circ \rho_1$ (see Fig. 3). Dually, transitions of $\mathcal{H}_2$ are simulated in $\mathcal{H}_1$.

We say that $\mathcal{H}_1$ and $\mathcal{H}_2$ are HD-bisimilar, written $\mathcal{H}_1 \sim_{hd} \mathcal{H}_2$, when there exists a HD-bisimulation R such that $(q_{01}, \delta, q_{02}) \in R$ for some δ.

Observe that, by commutativity of the diagram in Fig. 3, in case (1) we get that $\delta' = \rho_2^{-1} \circ \delta \circ \rho_1 \cup \{(e_1, e_2)\}$ and in case (2) $\delta = \rho_2 \circ \delta' \circ \rho_1^{-1} \cup \{(e_1, e_2)\}$. Hence, since the δ-component in R is a labelled bijection, whenever we match two transitions, the involved activities are required to have the same label.

The behavioural equivalence is referred to as HD-bisimilarity rather than HDE-bisimilarity since it is just the general notion [13] instantiated to our specific subclass of HD-automata.

4.2 Logical Characterisation of HD-Bisimilarity

We next show that HD-bisimilarity admits a natural logical characterisation in terms of a mild extension of the logic $\mathcal{L}_\mathbf{0}$ introduced in Section 3.

Definition 13 ($\mathcal{L}_{hd}$ syntax). *Let Var be a countable set of variables ranged over by $x, y, z....$ The logic $\mathcal{L}_{hd}$ over the set of labels Λ is defined as:*

$$\varphi \ ::= \ \mathsf{T} \ \mid \ \varphi \wedge \varphi \ \mid \ \neg\varphi \ \mid \ (\mathsf{a}\, z)\, \varphi \ \mid \ \langle z \rangle\, \varphi \ \mid \downarrow z\, \varphi$$

The logic $\mathcal{L}_{hd}$, besides the operators of $\mathcal{L}_{\mathbf{0}}$ for planning and executing activities, includes an additional operator $\downarrow z$ that represents the dismissal of a planned activity. More precisely, the formula $\downarrow z\, \varphi$ holds if $\eta(z) \in n(q)$, i.e., $\eta(z)$ is a planned activity in the current state (namely, an active name), and after dismissing such activity (i.e., forgetting the corresponding name) φ holds.

The free variables of a formula in $\mathcal{L}_{hd}$ are defined as in Section 3, with the additional clause $fv(\downarrow z\, \varphi) = fv(\varphi) \cup \{z\}$. Concerning the semantics, $\mathcal{L}_{hd}$ formulae are now interpreted over HDE-automata. More precisely, let *Env* be the set of environments, i.e., functions $\eta : Var \to \mathbb{E}$. Given a HDE-automaton $\mathcal{H}$ and a formula φ in $\mathcal{L}_{hd}$, the denotation of φ will be a set of pairs $(q, \eta) \in Q \times Env$. Note that the semantics of $\mathcal{L}_{hd}$ does not involve a notion of legal pair, which was essential in Section 3 to correctly deal with the conflict relation distinctive of PESs. In fact, as observed before, the states of HDE-automata explicitly include a set of planned activities which intuitively corresponds to events observed and not yet executed in PESs. The activities planned in a state are pairwise consistent by construction: for a state q of a HDE-automaton, the fact that a new activity e is in conflict with some activities which have been already planned in q is represented by the absence of a $\mathtt{plan}(e)$ transition from state q, i.e., by the impossibility of planning activity e in state q.

Below given a renaming $\rho \in R(\mathbb{E})$ and an environment $\eta : Var \to \mathbb{E}$ we write $\eta; \rho^{-1}$ for the environment defined by $\eta; \rho^{-1}(x) = \rho^{-1}(\eta(x))$ when $\eta(x) \in cod(\rho)$ and $\eta; \rho^{-1}(x) = \eta(x)$, otherwise.

Definition 14 (semantics). *Let $\mathcal{H}$ be a HDE-automaton. The denotation of a formula φ, written $\{\!|\varphi|\!\}^{\mathcal{H}} \in 2^{Q \times Env}$, is inductively defined as follow:*

$$\begin{aligned}
\{\!|\mathsf{T}|\!\}^{\mathcal{H}} &= Q \times Env \\
\{\!|\varphi_1 \wedge \varphi_2|\!\}^{\mathcal{H}} &= \{\!|\varphi_1|\!\}^{\mathcal{H}} \cap \{\!|\varphi_2|\!\}^{\mathcal{H}} \\
\{\!|\neg\varphi|\!\}^{\mathcal{H}} &= (Q \times Env) \setminus \{\!|\varphi|\!\}^{\mathcal{H}} \\
\{\!|(\mathsf{a}\, z)\, \varphi|\!\}^{\mathcal{H}} &= \{(q, \eta) \mid \exists q \xrightarrow{\mathtt{plan}(e)}_{\rho} q' \ \wedge \ \lambda(e) = \mathsf{a} \ \wedge \\
&\qquad (q', \eta; \rho^{-1}[z \mapsto e]) \in \{\!|\varphi|\!\}^{\mathcal{H}} \ \} \\
\{\!|\langle z \rangle\ \varphi|\!\}^{\mathcal{H}} &= \{(q, \eta) \mid q \xrightarrow{\mathtt{exec}(\eta(z))}_{\rho} q' \wedge (q', \eta; \rho^{-1}) \in \{\!|\varphi|\!\}^{\mathcal{H}}\} \\
\{\!|\downarrow z\ \varphi|\!\}^{\mathcal{H}} &= \{(q, \eta) \mid q \xrightarrow{\mathtt{drop}(\eta(z))}_{\rho} q' \wedge (q', \eta; \rho^{-1}) \in \{\!|\varphi|\!\}^{\mathcal{H}}\}
\end{aligned}$$

When $(q, \eta) \in \{\!|\varphi|\!\}^{\mathcal{H}}$ we say that the automaton $\mathcal{H}$ satisfies the formula φ in the state q and environment $\eta : Var \to E$, and write $\mathcal{H}, q \models_{\eta} \varphi$. For closed formulae φ, we write $\mathcal{H} \models \varphi$, when $\mathcal{H}, q_0 \models_{\eta} \varphi$ for some η.

The logical equivalence induced by $\mathcal{L}_{hd}$ over HDE-automata can be shown to be HD-bisimilarity. Actually, as it commonly happens when dealing with a finitary logic (with finite conjunctions), the result holds under suitable hypotheses which restrict the branching cardinality of HDE-automata. The standard requirement is image-finiteness, which, however, for HDE-automata would be too restrictive as $\mathtt{plan}(\cdot)$ steps allow one to plan activities which are executable unboundedly far in the future. Instead, we assume the following weaker notion of boundedness for HDE-automata.

Definition 15 (bounded HDE-automata). *A HDE-automaton $\mathcal{H}$ is called* bounded *if for any $q \in Q$, $k \in \mathbb{N}$ and $A \subseteq_{fin} \Lambda$ the set below is finite:*

$$q(k,A) = \{e \in \mathbb{E} \mid q \xrightarrow{\mathtt{plan}(e)}_{\rho} q' \xrightarrow{\ell_1}_{\rho_1} q_1 \ldots \xrightarrow{\ell_k}_{\rho_k} q_k \xrightarrow{\mathtt{exec}(e')} q_{k+1} \\ \wedge\ \rho_k \circ \ldots \circ \rho_1(e') = e \ \wedge\ \lambda(e) \in A \ \wedge \\ \textit{for } i \in \{1,\ldots,k\},\ \textit{if } \ell_i = \mathtt{plan}(e_i)\ \textit{ then } \lambda(e_i) \in A\}.$$

In words, $q(k,A)$ is the set of activities labelled over A which can be planned in the current state and executed in k steps, using only already planned activities or new activities labelled in A. This set is required to be finite when A is finite. We will show later that the automaton corresponding to a PES is bounded if and only if the original PES is image finite. Under the boundedness hypothesis, we can prove that the logical equivalence induced by $\mathcal{L}_{hd}$ on HDE-automata is HD-bisimilarity.

Proposition 1 (HD-bisimilarity, logically). *Let $\mathcal{H}_1$, $\mathcal{H}_2$ be bounded HDE-automata. Then $\mathcal{H}_1 \sim_{hd} \mathcal{H}_2$ iff $\mathcal{H}_1$, $\mathcal{H}_2$ satisfy the same closed formulae in $\mathcal{L}_{hd}$.*

The boundedness hypothesis is essential to ensure the existence of a finite formula distinguishing any two non bisimilar HDE-automata. Roughly, the point is that a $\mathtt{plan}(e_1)$ transition of an automaton could be simulated, in principle, by infinitely many $\mathtt{plan}(e_2)$ transitions of the other. However, by the irredundancy assumption on the class of HDE-automata, we know that e_1 is executable in some reachable state. Let k be the number of transitions of a run leading to a state where e_1 is executable and let A be the set of labels of events planned in such run. Then it is not difficult to see that the event e_2 of the simulating transition $\mathtt{plan}(e_2)$ must be itself executable within k steps, involving only already planned events or events labelled in the set A. By the boundedness hypothesis there are only finitely many such events, a fact which plays a basic role in the proof of Proposition 1.

5 Hhp-Bisimilarity via HD-Automata

In order to obtain a characterisation of hhp-bisimilarity in terms of HD-automata we proceed as follows: first we provide an encoding of PESs into the class of HDE-automata. Then we encode the logic $\mathcal{L}_{\mathbf{0}}$ into $\mathcal{L}_{hd}$ and back, in a way that a PES satisfies a formula in $\mathcal{L}_{\mathbf{0}}$ if and only if the corresponding automaton satisfies the

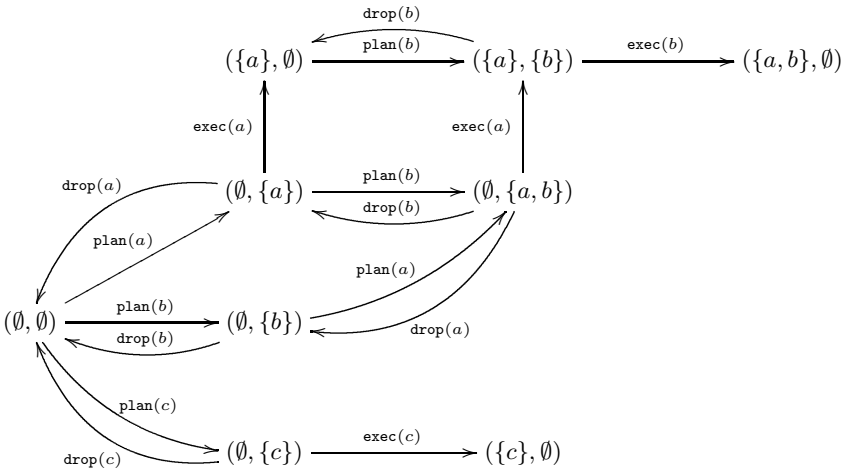

Fig. 4. HDE automaton corresponding to the CCS process $a.b + c$.

formula in $\mathcal{L}_{hd}$. Finally we rely on the logical characterisations of HD-bisimilarity and of hhp-bisimilarity to show that two PESs are hhp-bisimilar if and only if their corresponding HDE-automata are HD-bisimilar.

5.1 From Event Structures to HDE-Automata

We next provide an encoding of PESs into HDE-automata which is later shown to preserve and reflect behavioural equivalence. Throughout this section, the correspondence between activities in the source, label and target of a transition are given by (partial) identities and hence kept implicit.

Definition 16 (from PES to HDE-automata). *Let $\mathcal{E}$ be a* PES*. The HDE-automaton $\mathcal{H}(\mathcal{E})=(Q, q_0, n, \rightarrow)$ is defined as*

- $Q=\{\langle C, X\rangle \mid C \in \mathcal{C}(\mathcal{E}) \wedge X \subseteq_{fin} \mathcal{E}[C] \wedge X \times X \subseteq \frown \}$
- $q_0 = (\emptyset, \emptyset)$
- $n(\langle C, X\rangle) = X$
- *the transition relation is given as follows where it is assumed that $e \notin X$*
  - $\langle C, X\rangle \xrightarrow{\texttt{plan}(e)} \langle C, X \cup \{e\}\rangle$ *when $e \in \mathcal{E}[C]$ and $e \frown X$;*
  - $\langle C, X \cup \{e\}\rangle \xrightarrow{\texttt{exec}(e)} \langle C \cup \{e\}, X\rangle$ *when $C \cup \{e\} \in \mathcal{C}(\mathcal{E})$;*
  - $\langle C, X \cup \{e\}\rangle \xrightarrow{\texttt{drop}(e)} \langle C, X\rangle$.

In words, a PES $\mathcal{E}$ corresponds to an automaton $\mathcal{H}(\mathcal{E})$ whose states are pairs $\langle C, X\rangle$ where $C \in \mathcal{C}(\mathcal{E})$ represents the current state of the computation, and X is a set of events belonging to a possible future computation extending C, planned but not yet executed. Note that, in order to represent a set of events which can occur in a computation starting from C, the events in X must be both pairwise

consistent and consistent with C. Instead, we do not require X to be causally closed, that is we do not require $C \cup X \in \mathcal{C}(\mathcal{E})$.

According to this intuition, given a state $\langle C, X\rangle$, the transition $\mathtt{plan}(e)$ allows one to plan a new event e whenever e is compatible both with C and its future X. On the other hand, any event planned and not yet executed, i.e., any event $e \in X$ can be dismissed by means of a $\mathtt{drop}(e)$ transition. Finally, an event e can be executed if it belongs to the planned future X and it is enabled by the configuration C. As an example, the automaton corresponding to the (PES associated with the) CCS process $a.b+c$ is given in Fig. 4. Observe that the HDE-automaton obtained from a PES is irredundant. Roughly, $\mathtt{plan}(\cdot)$ and $\mathtt{drop}(\cdot)$ transitions allow one to construct alternative futures of the current configuration. The concurrent structure of such futures can then be analysed by means of $\mathtt{exec}(\cdot)$ moves.

Note that, as mentioned above, the ρ-component of transitions is omitted and it is implicitly assumed to be a partial identity. More precisely when $\langle C, X\rangle \xrightarrow{\ell} \langle C', X'\rangle$, the renaming is $\rho = id_{X\cap X'}$. For instance, when $\langle C, X\rangle \xrightarrow{\mathtt{plan}(e)} \langle C, X \cup \{e\}\rangle$, the renaming $\rho : X \cup \{e\} \to X$ is defined by $\rho(e') = e'$ for $e' \in X$ and $\rho(e)$ undefined.

The image finiteness property for PESs exactly corresponds, through the encoding, to the boundedness property for HDE-automata as introduced in Definition 15.

Proposition 2 (image finiteness). *Let $\mathcal{E}$ a* PES. *Then $\mathcal{E}$ is image finite iff $\mathcal{H}(\mathcal{E})$ is bounded.*

5.2 From $\mathcal{L}_0$ to $\mathcal{L}_{hd}$ and Back: Hhp-Bisimilarity via hd-Bisimilarity

In order to prove that behavioural equivalence is preserved and reflected by the encoding of PESs into HDE-automata we rely on the logical characterisation of such equivalences, which is given in terms of very similar logics. Specifically, here we prove that a tight link exists between satisfaction of $\mathcal{L}_0$ formulae by PESs and satisfaction of $\mathcal{L}_{hd}$ formulae by the corresponding HDE-automata, in a way that the two logical equivalences can then be shown to coincide. Below, we write $\models^{\mathcal{L}_0}$ and $\models^{\mathcal{L}_{hd}}$ in order to clarify to which notion of satisfaction we are referring to.

First of all, notice that although $\mathcal{L}_0$ is syntactically a subset of $\mathcal{L}_{hd}$, for a PES $\mathcal{E}$ and a formula φ in $\mathcal{L}_0$, it is not the case that if $\mathcal{E} \models^{\mathcal{L}_0} \varphi$ then $\mathcal{H}(\mathcal{E}) \models^{\mathcal{L}_{hd}} \varphi$. As an example, consider the PES $\mathcal{E}_1$ in Fig. 1 associated with the process $a.b + c.d$ and the formula $\varphi = (\mathsf{a}\,x)((\mathsf{c}\,y)\langle y\rangle\,\mathsf{T} \wedge (\mathsf{b}\,z)\langle x\rangle\,\langle z\rangle\,\mathsf{T})$. Then $\mathcal{E}_1 \models^{\mathcal{L}_0} \varphi$, because $\emptyset \models^{\mathcal{L}_0}_{\eta[x\to a]} (\mathsf{c}\,y)\langle y\rangle\,\mathsf{T}$ and $\emptyset \models^{\mathcal{L}_0}_{\eta[x\to a]} (\mathsf{b}\,z)\langle x\rangle\,\langle z\rangle\,\mathsf{T}$. In fact, for the first subformula, note that y can be bound to the c-labelled event even though it is in conflict with a, since x is no longer free in the subformula.

Instead, $\mathcal{H}(\mathcal{E}_1) \not\models^{\mathcal{L}_{hd}} \varphi$ since satisfaction reduces to $(\emptyset, \{a\}) \models^{\mathcal{L}_{hd}}_{\eta[x\to a]} (\mathsf{c}\,y)\langle y\rangle\,\mathsf{T}$ and $(\emptyset, \{a\}) \models^{\mathcal{L}_{hd}}_{\eta[x\to a]} (\mathsf{b}\,z)\langle x\rangle\,\langle z\rangle\,\mathsf{T}$. The first is false since the automaton cannot perform a $\mathtt{plan}(c)$ step as long as the conflicting event a belongs to the planned future. However, $\mathcal{H}(\mathcal{E}_1) \models^{\mathcal{L}_{hd}} (\mathsf{a}\,x)(\downarrow x\,(\mathsf{c}\,y)\langle y\rangle\,\mathsf{T} \wedge (\mathsf{b}\,z)\langle x\rangle\,\langle z\rangle\,\mathsf{T})$ since, in this

case, after planning a, the left branch forgets it in a way that b can be planned and executed.

More generally, a $\mathcal{L}_{\mathbf{0}}$ formula φ can be encoded into a $\mathcal{L}_{hd}$ formula that uses the $\downarrow$ operator to explicitly drop planned events that intuitively no longer pertain to the future that the formula describes, i.e., events planned but no longer referred to by free variables in the remaining part of the formula. Formally, given $\varphi \in \mathcal{L}_{\mathbf{0}}$, we define an encoding of φ into $\mathcal{L}_{hd}$ which is parametric on a set of variables X such that $fv(\varphi) \subseteq X$, representing the events planned in the past. Given a set of variables $Z = \{z_1, \ldots, z_n\}$ we write $\downarrow Z$ for $\downarrow x_1 \ldots \downarrow x_n$.

Definition 17 (from $\mathcal{L}_{\mathbf{0}}$ to $\mathcal{L}_{hd}$). *The encoding function $[\![\cdot]\!] : \mathcal{L}_{\mathbf{0}} \times 2^{Var} \to \mathcal{L}_{hd}$ is inductively defined as follows:*

$$
\begin{aligned}
[\![\mathsf{T}]\!]_X &= \mathsf{T} \\
[\![\neg\varphi]\!]_X &= \neg[\![\varphi]\!]_X \\
[\![\varphi_1 \wedge \varphi_2]\!]_X &= [\![\varphi_1]\!]_X \wedge [\![\varphi_2]\!]_X \\
[\![\langle x\rangle\, \varphi]\!]_X &= \langle x\rangle\, [\![\varphi]\!]_X \\
[\![(\mathsf{a}\, x)\varphi]\!]_X &= \downarrow Z\, (\mathsf{a}\, x)[\![\varphi]\!]_{fv(\varphi)\cup\{x\}}
\end{aligned}
$$

where, in the last clause, $Z = X \backslash (fv(\varphi) \backslash \{x\})$.

In words, before binding a new event to x, the $\mathcal{L}_{hd}$ encoding drops any (previously planned) event that is not bound to the free variables of the subformula.

As an example, consider the formula $\varphi = (\mathsf{a}\, x)((\mathsf{c}\, y)\langle y\rangle\, \mathsf{T} \wedge (\mathsf{b}\, z)\langle x\rangle\, \langle z\rangle\, \mathsf{T})$ in $\mathcal{L}_{\mathbf{0}}$ discussed at the beginning of the section, satisfied by $\mathcal{E}_1$ but not by $\mathcal{H}(\mathcal{E}_1)$. The formula $[\![\varphi]\!]_\emptyset$ is exactly the $\mathcal{L}_{hd}$ formula previously constructed by hand in order to be satisfied by the automaton, i.e., $(\mathsf{a}\, x)(\downarrow x\, (\mathsf{c}\, y)\langle y\rangle\, \mathsf{T} \wedge (\mathsf{b}\, z)\langle x\rangle\, \langle z\rangle\, \mathsf{T})$. As a further example, consider the $\mathcal{L}_{\mathbf{0}}$ formulae $\varphi_1 = (\mathsf{a}\, x)(\mathsf{b}\, y)(\mathsf{c}\, z)\langle z\rangle\, \mathsf{T}$ and $\varphi_2 = (\mathsf{a}\, x)(\mathsf{b}\, y)(\langle y\rangle\, \mathsf{T} \wedge (\mathsf{c}\, z)\langle z\rangle\, \mathsf{T})$, which are both true for the PES consisting of three pairwise conflicting events. Then we have that $[\![\varphi_1]\!]_\emptyset = (\mathsf{a}\, x) \downarrow x\, (\mathsf{b}\, y) \downarrow y\, (\mathsf{c}\, z)\langle z\rangle\, \mathsf{T}$ and $[\![\varphi_2]\!]_\emptyset = (\mathsf{a}\, x) \downarrow x\, (\mathsf{b}\, y)(\langle y\rangle\, \mathsf{T} \wedge \downarrow y\, (\mathsf{c}\, z)\langle z\rangle\, \mathsf{T})$.

We next prove a technical lemma. It roughly asserts that, given a formula $\varphi \in \mathcal{L}_{\mathbf{0}}$, the satisfaction of its encoding in $\mathcal{L}_{hd}$ by a state of the HD-automaton does not depend on planned events bound to variables which are not free in the formula, as long as the encoding takes care of dropping such events.

Lemma 1. *Let $\mathcal{E}$ be a* PES. *Let $\varphi \in \mathcal{L}_{hd}$ be a formula, $\eta : Var \to \mathbb{E}$ an environment and $X_1, X_2 \subseteq Var$ sets of variables such that $fv(\varphi) \subseteq X_i$ and $C \cup \eta(X_i)$ is compatible for $i \in \{1, 2\}$. Then in the HDE-automata $\mathcal{H}(\mathcal{E})$ it holds*

$$\langle C, \eta(X_1) \setminus C\rangle \models_\eta [\![\varphi]\!]_{X_1} \quad \textit{iff} \quad \langle C, \eta(X_2) \setminus C\rangle \models_\eta [\![\varphi]\!]_{X_2}.$$

Then we can prove the following.

Lemma 2 (from $\mathcal{L}_{\mathbf{0}}$ to $\mathcal{L}_{hd}$). *Let $\mathcal{E}$ be a* PES. *For any closed formula $\varphi \in \mathcal{L}_{\mathbf{0}}$ it holds $\mathcal{E} \models^{\mathcal{L}_{\mathbf{0}}} \varphi$ iff $\mathcal{H}(\mathcal{E}) \models^{\mathcal{L}_{hd}} [\![\varphi]\!]_\emptyset$.*

Conversely, we show how formulae of $\mathcal{L}_{hd}$ can be encoded in $\mathcal{L}_{\mathbf{0}}$. This is somehow more difficult since the notion of satisfaction in $\mathcal{L}_{\mathbf{0}}$ relies on simpler states, those of PESs, consisting only of a configuration (executed events), while states of HDE-automata, where $\mathcal{L}_{hd}$ satisfaction is defined, include explicitly also those events which have been planned and not executed. In order to fill this gap the idea is to "keep" events planned but not yet executed as free variables in the formulae of $\mathcal{L}_{\mathbf{0}}$.

Definition 18 (from $\mathcal{L}_{hd}$ to $\mathcal{L}_{\mathbf{0}}$). *Given a set of variables $X=\{x_1,\ldots,x_n\} \subseteq Var$, let $st(X)$ denote the formula in $\mathcal{L}_{\mathbf{0}}$*

$$st(X) = (\bigvee_{i=1}^{n} \langle x_i \rangle \mathsf{T}) \vee \mathsf{T}$$

The encoding function $\|\cdot\| : \mathcal{L}_{hd} \times 2^{Var} \to \mathcal{L}_{\mathbf{0}}$ is inductively defined as follows:

$$\begin{aligned}
\|\mathsf{T}\|_X &= \mathsf{T} \\
\|\neg\varphi\|_X &= \neg\|\varphi\|_X \\
\|\varphi_1 \wedge \varphi_2\|_X &= \|\varphi_1\|_X \wedge \|\varphi_2\|_X \\
\|(\mathsf{a}\, x)\varphi\|_X &= (\mathsf{a}\, x)(\|\varphi\|_{X\cup\{x\}} \wedge st(X)) \\
\|{\downarrow}x\ \varphi\|_X &= \begin{cases} \|\varphi\|_{X\setminus\{x\}} & \text{if } x \in X \\ \mathsf{F} & \text{otherwise} \end{cases} \\
\|\langle x\rangle\ \varphi\|_X &= \begin{cases} \langle x\rangle\ \|\varphi\|_{X\setminus\{x\}} & \text{if } x \in X \\ \mathsf{F} & \text{otherwise} \end{cases}
\end{aligned}$$

Observe that the encoding of a formula of $\mathcal{L}_{hd}$ into $\mathcal{L}_{\mathbf{0}}$ is parametric w.r.t. a set of variables which represent those events which have been planned but not yet dropped or executed. In order to understand this, note that in the formula $st(X)$ the disjunction with T does not make it trivially equivalent to true. In fact $fv(st(X)) = X$, and thus $st(X)$ is satisfied only by pairs (C,η) which are legal, i.e., such that $\eta(X) \subseteq C[E]$ and pairwise consistent. The role of $st(X)$ is exactly to keep alive the events associated with variable in X and impose that they are consistent. It can be proved inductively that, more generally, $fv(\|\varphi\|_X) \subseteq X$.

Lemma 3 (from $\mathcal{L}_{hd}$ to $\mathcal{L}_{\mathbf{0}}$). *Let $\mathcal{E}$ be a* PES*, let $\mathcal{H}(\mathcal{E})$ be the corresponding automaton. For any closed formula $\varphi \in \mathcal{L}_{hd}$, $\mathcal{H}(\mathcal{E}) \models^{\mathcal{L}_{hd}} \varphi$ iff $\mathcal{E} \models^{\mathcal{L}_{\mathbf{0}}} \|\varphi\|_\emptyset$.*

Combining the results above we can immediately deduce that hhp-bisimilarity between PESs is faithfully captured by bisimilarity of the corresponding HDE-automata.

Theorem 2 (hhp-bisimilarity vs. hd-bisimilarity). *Let $\mathcal{E}_1$ and $\mathcal{E}_2$ be* PES*s. Then $\mathcal{E}_1 \sim_{hhp} \mathcal{E}_2$ iff $\mathcal{H}(\mathcal{E}_1) \sim_{hd} \mathcal{H}(\mathcal{E}_2)$.*

6 Conclusions: Related and Future Work

We studied hhp-bisimilarity, a canonical behavioural equivalence in the true concurrent spectrum, by means of logics and automata. We provided a characterisation in terms of an event-based logic $\mathcal{L}_{\mathbf{0}}$ that predicates over the existence and executability of events. This in turn suggests a connection with HD-automata. More precisely, we defined a class of HD-automata whose transitions allow one to plan the execution of an activity, execute a planned activity and to dismiss a planned activity. We then showed that PESs can be mapped into such class of automata in a way that the canonical behavioural equivalence for HD-automata coincides with hhp-bisimilarity over the corresponding PESs.

Both characterisations show that, in order to capture hhp-bisimilarity, the observer must be able to compare states by checking unboundedly large concurrent computations in the future of such states. Intuitively, this can be seen as a source of ineffectiveness of hhp-bisimilarity which indeed is known to be undecidable for many basic models of concurrency, even in the finite state case (e.g., it is known that hhp-bisimilarity is undecidable for safe finite Petri nets [14]).

The results in the paper can be helpful in the study of decidable approximations of hhp-bisimilarity, possibly opening the road to the development of verification techniques. This represents an interesting line of future research. Indeed, some preliminary investigations show that fixing a bound on the distance of the future that an observer is allowed to check, one gets effective approximations of hhp-bisimilarity. More precisely, when fixing such a bound, regular PESs (which typically arise as semantics of finite state systems [16]) can be transformed into finite HD-automata for which bisimilarity checking is decidable. On these bases, algorithms for checking such approximations of $\sim_{hhp}$-bisimilarity can be obtained by simply providing an explicit construction of the finite HD-automata for specific formalisms. E.g., for finite (n-)safe Petri nets this could be done along the lines of the work in [17,18] for history preserving bisimilarity. The construction could also take inspiration from that in [19], used for proving decidability of approximations of hhp-bisimilarity on finite safe Petri nets.

The fact that HDE-automata deal with infinite sets of events, but with the possibility of testing only equality and labels, suggests a connection with register automata and, more generally, with the recent line of work on nominal automata (see, e.g., [20] and references therein), which would be interesting to deepen.

In order to capture hhp-bisimilarity in the setting of HD-automata, we provided a characterisation of HD-bisimilarity in terms of a logic $\mathcal{L}_{hd}$ that enriches $\mathcal{L}_{\mathbf{0}}$ with an operator for explicitly dropping activities planned but not yet executed. Interestingly, even if it is defined over HDE-automata, we think that the logic $\mathcal{L}_{hd}$ will be useful to establish a precise connection with the logic EIL in [10], which includes a reverse step-modality which is related to the drop transitions and the $\downarrow\cdot$ modality in $\mathcal{L}_{hd}$. We believe a further investigation of the relation between $\mathcal{L}_{\mathbf{0}}$ and EIL (and the other logics for concurrency in the literature) can bring some interesting insights, at least at conceptual level. This, despite the fact that it is clear that some modalities of $\mathcal{L}_{\mathbf{0}}$ and EIL are not interdefinable. For instance, the formula $(x : \mathsf{a})\varphi$ in EIL which binds x to an a-labelled event in

the current configuration is not encodable in $\mathcal{L}_{\mathbf{0}}$. Conversely, the formula $(\mathsf{a}\, x)\varphi$ where x is bound to an a-labelled event in the future of the current configuration is not encodable in EIL. A connection to be further investigated seems to exist also with the work on higher-dimensional automata and ST-configuration structures in [21], where a logic, again with backward step modalities, is proposed for hhp-bisimilarity.

We also believe that the logical characterisation of HD-bisimilarity has an interest which goes beyond the specific class of HD-automata considered in this paper and deserves to be studied further.

Acknowledgments. We are grateful to Alberto Meneghello for several insightful discussions on this work at its early stages of development. We are also indebted with the anonymous reviewers for providing detailed comments and insightful suggestions which helped us to improve our work.

References

1. van Glabbeek, R., Goltz, U.: Refinement of actions and equivalence notions for concurrent systems. Acta Informatica 37(4/5), 229–327 (2001)
2. Esparza, J., Heljanko, K.: Unfoldings - A Partial order Approach to Model Checking. EACTS Monographs. Springer (2008)
3. Bednarczyk, M.A.: Hereditary history preserving bisimulations or what is the power of the future perfect in program logics. Technical report, Polish Academy of Sciences (1991)
4. Joyal, A., Nielsen, M., Winskel, G.: Bisimulation from open maps. Information and Computation 127(2), 164–185 (1996)
5. Winskel, G.: Event Structures. In: Brauer, W., Reisig, W., Rozenberg, G. (eds.) APN 1986. LNCS, vol. 255, pp. 325–392. Springer, Heidelberg (1987)
6. Phillips, I., Ulidowski, I.: A hierarchy of reverse bisimulations on stable configuration structures. Mathematical Structures in Computer Science 22(2), 333–372 (2012)
7. Phillips, I., Ulidowski, I.: Reversing algebraic process calculi. Journal of Logic and Algebraic Programming 73(1-2), 70–96 (2007)
8. Cristescu, I., Krivine, J., Varacca, D.: A compositional semantics for the reversible p-calculus. In: Proc. of LICS 2013, pp. 388–397. IEEE Computer Society (2013)
9. Baldan, P., Crafa, S.: A logic for true concurrency. Journal of the ACM 61(4), 24:1–24:36 (2014)
10. Phillips, I., Ulidowski, I.: Event identifier logic. Mathematical Structures in Computer Science 24(2), 1–51 (2014)
11. Nielsen, M., Clausen, C.: Games and logics for a noninterleaving bisimulation. Nordic Journal of Computing 2(2), 221–249 (1995)
12. Hennessy, M., Stirling, C.: The power of the future perfect in program logics. Information and Control 67(1-3), 23–52 (1985)
13. Montanari, U., Pistore, M.: History-Dependent automata: An introduction. In: Bernardo, M., Bogliolo, A. (eds.) SFM-Moby 2005. LNCS, vol. 3465, pp. 1–28. Springer, Heidelberg (2005)
14. Jurdzinski, M., Nielsen, M., Srba, J.: Undecidability of domino games and hhp-bisimilarity. Information and Computation 184(2), 343–368 (2003)

15. Hennessy, M., Milner, R.: Algebraic laws for nondeterminism and concurrency. Journal of the ACM 32(1), 137–161 (1985)
16. Thiagarajan, P.S.: Regular event structures and finite petri nets: A conjecture. In: Brauer, W., Ehrig, H., Karhumäki, J., Salomaa, A. (eds.) Formal and Natural Computing. LNCS, vol. 2300, pp. 244–256. Springer, Heidelberg (2002)
17. Vogler, W.: Deciding history preserving bisimilarity. In: Leach Albert, J., Monien, B., Rodríguez-Artalejo, M. (eds.) ICALP 1991. LNCS, vol. 510, pp. 495–505. Springer, Heidelberg (1991)
18. Montanari, U., Pistore, M.: Minimal transition systems for history-preserving bisimulation. In: Reischuk, R., Morvan, M. (eds.) STACS 1997. LNCS, vol. 1200, pp. 413–425. Springer, Heidelberg (1997)
19. Fröschle, S., Hildebrandt, T.: On plain and hereditary history-preserving bisimulation. In: Kutyłowski, M., Wierzbicki, T., Pacholski, L. (eds.) MFCS 1999. LNCS, vol. 1672, pp. 354–365. Springer, Heidelberg (1999)
20. Bojanczyk, M., Klin, B., Lasota, S.: Automata with group actions. In: Proc. of LICS 2011, pp. 355–364. IEEE Computer Society (2011)
21. Prisacariu, C.: The glory of the past and geometrical concurrency. CoRR abs/1206.3136 (2012)