A Logic for True Concurrency*

Paolo Baldan and Silvia Crafa

Department of Pure and Applied Math, University of Padova

Abstract. We propose a logic for true concurrency whose formulae predicate about events in computations and their causal dependencies. The induced logical equivalence is hereditary history preserving bisimilarity, and fragments of the logic can be identified which correspond to other true concurrent behavioural equivalences in the literature: step, pomset and history preserving bisimilarity. Standard Hennessy-Milner logic, thus (interleaving) bisimilarity, is also recovered as a fragment. We believe that this contributes to a rational presentation of the true concurrent spectrum and to a deeper understanding of the relations between the involved behavioural equivalences.

1 Introduction

In the semantics of concurrent and distributed systems, a major dichotomy opposes the *interleaving approaches*, where concurrency of actions is reduced to the non-deterministic choice among their possible sequentializations, to *true-concurrent approaches*, where concurrency is taken as a primitive notion. In both cases, on top of the operational models a number of behavioural equivalences have been defined by abstracting from aspects which are considered unobservable [1, 2].

For the interleaving world, a systematic and impressive picture is taken in the linear-time branching-time spectrum [1]. Quite interestingly, the equivalences in the spectrum can be uniformly characterised in logical terms. Bisimilarity, the finest equivalence, corresponds to Hennessy-Milner (HM) logic: two processes are bisimilar if and only if they satisfy the same HM logic formulae [3]. Coarser equivalences correspond to suitable fragments of HM logic.

In the true-concurrent world, relying on models like event structures or transition systems with independence [4], several behavioural equivalences have been defined, ranging from hereditary history preserving (hhp-) bisimilarity, to pomset and step bisimilarity. Correspondingly, a number of logics have been studied, but, to the best of our knowledge, a unifying logical framework encompassing the main true-concurrent equivalences is still missing. The huge amount of work on the topic makes it impossible to give a complete account of related approaches. Just to give a few references (see Section 5 for a wider discussion), [5] proposes a general framework encompassing a number of temporal and modal logics that

* Supported by the MIUR Projects `SisteR` and `AIDA2007`, and the project `AVIAMO` of the University of Padova.

characterize pomset and weak history preserving bisimilarities as well as interleaving bisimilarity. However, finer equivalences are not considered and a single unitary logic is missing. History preserving (hp-) bisimilarity has been characterised in automata-theoretic terms using HD-automata [6] or Petri nets [7]. More recently, hp-bisimilarity has been obtained as a logical equivalence, using a separation fixpoint modal logic where it is possible to check the causal dependence/independence of the last executed action [8, 9]. Concerning hhp-bisimilarity, several logics with modalities corresponding to the "retraction" or "backward" execution of computations have been proposed [10–12]. In absence of autoconcurrency they are shown to capture hhp-bisimilarity, while the general case complicates the picture and requires some adjustments.

In this paper we propose a behavioural logic for concurrency and we show that it allows to characterise a relevant part of the truly concurrent spectrum. More specifically, the full logic is shown to capture hhp-bisimilarity, the finest equivalence in the spectrum [2]. Then suitable fragments of the logic are shown to scale down to the characterisation of other coarser equivalences, i.e., history preserving, pomset and step bisimilarity. Standard HM logic, and thus (interleaving) bisimilarity, is also recovered as a fragment.

Our logic allows to predicate about events in computations together with their causal and independence relations. It is interpreted over prime event structures, but it could naturally be interpreted over any formalism with a notion of event, causality and consistency. A formula is evaluated in a configuration representing the current state of the computation, and it predicates on a set of possible future evolutions starting from that state. The logic is event-based in the sense that it contains a binder that allows to refer later to the events the formula predicates on. In this respect, it is reminiscent of the modal analogue of independence-friendly modal logic as considered in [13].

The logic contains two main operators. The formula $(x, \overline{y} < \mathsf{a}\, z)\varphi$ declares that 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 so that it can be later referred to in φ. In general, x and y can be replaced by lists of variables. A second operator allows to "execute" events previously bound to 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.

Different behavioural equivalences are induced by fragments of the logics where we suitably restrict the set of possible futures the formulae are able to refer to. Namely, hhp-bisimilarity, that is captured by the full logic, corresponds to the ability of observing the existence of a number of legal but (possibly) incompatible futures, without executing such futures. Interestingly, the definition of hhp-bisimilarity is normally given in terms of backward transitions, whereas our logical characterization has a "forward flavor". By restricting to a fragment where future events can be observed only by executing them (any occurrence of the binding operator is immediately followed by a corresponding execution), we get hp-bisimilarity. Pomset bisimilarity is induced by a fragment of the logic obtained by further restricting that for hp-bisimilarity, with the requirement

that propositional connectives are used only on closed (sub)formulae. Roughly speaking, this fragment predicates about the possibility of executing pomset transitions and the closedness requirement prevents pomset transitions from being causally linked to the events in the past. Finally, quite intuitively, step bisimilarity corresponds to the possibility of observing only currently enabled concurrent actions.

We believe that this work contributes to the definition of a logical counterpart of the true concurrent spectrum and shades further light on the relations between the involved behavioural equivalences.

2 Background

In this section we provide the basics of prime event structures which will be used as models for our logic. Then we define some common behavioural concurrent equivalences which will play a basic role in the paper.

2.1 Event structures

Prime event structures [14] 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 Λ denotes a fixed set of labels ranged over by $\mathsf{a}, \mathsf{b}, \mathsf{c} \ldots$

Definition 1 (prime event structure). *A (Λ-labelled)* prime event structure *(*PES*) is a tuple $\mathcal{E} = \langle E, \leq, \#, \lambda \rangle$, where E is a set of* events*, $\lambda : E \to \Lambda$ is a labelling function 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. Subscripts carry over the components.

Definition 2 (consistency, concurrency). *Let $\mathcal{E}$ be a* PES*. We say that $e, e' \in E$ are* consistent*, written $e \frown e'$, if $\neg(e \# e')$. Moreover, we say that e and e' are* concurrent*, written $e \,||\, e'$, if $\neg(e \leq e')$, $\neg(e' \leq e)$ and $\neg(e \# e')$.*

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. $e' \frown e$. We write $\lceil X \rceil$ for $\bigcup_{e \in X} \lceil e \rceil$.

The idea of (concurrent) computation is captured, in event structures, by the notion of configuration.

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

Hereafter, unless explicitly stated otherwise, all configurations will be assumed to be finite. A pairwise consistent subset $X \subseteq E$ of events will be always seen as the *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.

Definition 4 (pomset transition and step). *Let $\mathcal{E}$ be a* PES *and let $C \in \mathcal{C}(\mathcal{E})$. Given $\emptyset \neq X \subseteq E$, if $C \cap X = \emptyset$ and $C' = C \cup X \in \mathcal{C}(\mathcal{E})$ we write $C \xrightarrow{X} C'$ and call it a* pomset transition *from C to C'. When the events in X are pairwise concurrent, we say that $C \xrightarrow{X} C'$ is a* step*. When $X = \{e\}$ we write $C \xrightarrow{e} C'$ instead of $C \xrightarrow{\{e\}} C'$.*

A PES $\mathcal{E}$ is called *image finite* if for any $C \in \mathcal{C}(\mathcal{E})$ 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. As it commonly happens when relating modal logics and bisimilarities, this assumption is crucial for getting the logical characterisation of the various bisimulation equivalences in Section 4.

2.2 Concurrent behavioural equivalences

Behavioural equivalences which capture to some extent the concurrency features of a system, can be defined on the transition system where states are configurations and transitions are pomset transitions.

Definition 5 (pomset, step bisimulation). *Let $\mathcal{E}_1$, $\mathcal{E}_2$ be* PES*s. A* pomset bisimulation *is a relation $R \subseteq \mathcal{C}(\mathcal{E}_1) \times \mathcal{C}(\mathcal{E}_2)$ such that if $(C_1, C_2) \in R$ and $C_1 \xrightarrow{X_1} C_1'$ then $C_2 \xrightarrow{X_2} C_2'$, with $X_1 \sim X_2$ and $(C_1', C_2') \in R$, and vice versa. We say that $\mathcal{E}_1$, $\mathcal{E}_2$ are* pomset bisimilar*, written $\mathcal{E}_1 \sim_p \mathcal{E}_2$, if there exists a pomset bisimulation R such that $(\emptyset, \emptyset) \in R$.*

Step bisimulation *is defined analogously, replacing general pomset transitions with steps. We write $\mathcal{E}_1 \sim_s \mathcal{E}_2$ when $\mathcal{E}_1$ and $\mathcal{E}_2$ are step bisimilar.*

While pomset and step bisimilarity only consider the causal structure of the current step, (hereditary) history preserving bisimilarities are sensible to the way in which the executed events depend on events in the past. In order to define history preserving bisimilarities the following definition is helpful.

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

$$\{(C_1, f, C_2) : C_1 \in \mathcal{C}(\mathcal{E}_1),\ C_2 \in \mathcal{C}(\mathcal{E}_2),\ f : C_1 \to C_2 \text{ isomorphism}\}$$

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 when 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* $(C_1, f, C_2) \in R$.

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 an hp-bisimulation* R *such that* $(\emptyset, \emptyset, \emptyset) \in R$.

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

3 A logic for true concurrency

In this section we introduce the syntax and the semantics of our logic, where formulae predicate about events in computations and their dependencies as primitive concepts. The logic is interpreted over PESs, but it could be equivalently interpreted over any formalism with a notion of event, causality and consistency.

As a first step we define a quite general syntax and the semantics of the two distinctive operators. Our logic for concurrency will then be obtained by filtering out formulae which do not satisfy a suitable well-formedness condition.

In order to keep the notation simple, lists of variables like $x_1, \ldots, x_n$ will be denoted by $\boldsymbol{x}$ and, abusing the notation, lists will be sometimes used as sets.

Definition 8 (syntax). *Let* Var *be a denumerable set of variables ranged over by* $x, y, z, \ldots$. *The syntax of the formulae over the set of labels* Λ *is defined as follows:*

$$\varphi \ ::= \ (\boldsymbol{x}, \overline{\boldsymbol{y}} < \mathsf{a}\, z)\, \varphi \ \mid \ \langle z \rangle\, \varphi \mid \ \varphi \wedge \varphi \ \mid \ \neg\varphi \ \mid \ \top$$

The operator $(\boldsymbol{x}, \overline{\boldsymbol{y}} < \mathsf{a}\, z)$ acts as a binder for variable z, as clarified by the following notion of free variables in a formula.

Definition 9 (free variables). *The set of free variables of a formula* φ *is denoted* $fv(\varphi)$ *and it is inductively defined by:*

$$fv((\boldsymbol{x}, \overline{\boldsymbol{y}} < \mathsf{a}\, z)\, \varphi) = (fv(\varphi) \setminus \{z\}) \cup \boldsymbol{x} \cup \boldsymbol{y} \qquad fv(\langle z \rangle\, \varphi) = fv(\varphi) \cup \{z\}$$
$$fv(\varphi_1 \wedge \varphi_2) = fv(\varphi_1) \cup fv(\varphi_2) \qquad fv(\top) = \emptyset \qquad fv(\neg\varphi) = fv(\varphi)$$

The satisfaction of a formula is defined with respect to a configuration, representing the state of the computation. Moreover a partial function $\eta : Var \to E$ is fixed, called an *environment*, in order to bind free variables in φ to events.

Definition 10 (semantics). *Let* $\mathcal{E} = \langle E, \leq, \#, \lambda \rangle$ *be a* PES. *For* $C \in \mathcal{C}(\mathcal{E})$ *a configuration,* φ *a formula and* $\eta : Var \to E$ *an environment such that* $fv(\varphi) \subseteq dom(\eta)$, *satisfaction* $\mathcal{E}, C \models_\eta \varphi$ *is defined as follows:*

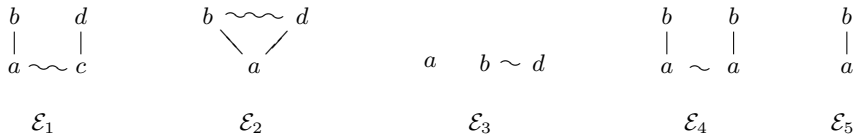

Fig. 1.

$\mathcal{E}, C \models_\eta (\boldsymbol{x}, \overline{\boldsymbol{y}} < \mathsf{a}\, z)\, \varphi$ *if there is* $e \in E \setminus C$*, such that*
- $\lambda(e) = \mathsf{a}$ *and* $C \frown e$*,* $\eta(\boldsymbol{x}) < e$*,* $\eta(\boldsymbol{y}) \,||\, e$
- $\mathcal{E}, C \models_{\eta'} \varphi$*, where* $\eta' = \eta[z \mapsto e]$

$\mathcal{E}, C \models_\eta \langle z \rangle\, \varphi$ *if* $C \xrightarrow{\eta(z)} C'$ *and* $\mathcal{E}, C' \models_\eta \varphi$

The semantics of propositional connectives is defined as usual. We say that a PES $\mathcal{E}$ *satisfies a closed formula* φ*, written* $\mathcal{E} \models \varphi$*, when* $\mathcal{E}, \emptyset \models_\emptyset \varphi$*.*

Intuitively, the formula

$$(x_1 \dots x_n, \overline{y_1 \dots y_m} < \mathsf{a}\, z)\, \varphi$$

holds in C if in the future of such a configuration there is an a-labelled event e, and binding e to the variable z, the formula φ holds. Such an event is required to be caused (at least) by the events already bound to $x_1 \dots x_n$, and to be independent (at least) from those bound to $y_1 \dots y_m$. We stress that the event e might not be currently enabled; it is only required to be consistent with the current configuration, meaning that it could be enabled in the future of the current configuration. The formula $\langle z \rangle\, \varphi$ says that the event bound to z is currently enabled, hence it can be executed producing a new configuration which satisfies formula φ. To simplify the notation we write $(\mathsf{a}\, z)\, \varphi$ for $(\ < \mathsf{a}\, z)\, \varphi$ and we often omit trailing $\top$ at the end of the formulae.

As an example, consider the PES $\mathcal{E}_1$ in Fig. 1, corresponding to the CCS process $a.b + c.d$, where curly lines represent immediate conflict and the causal order proceeds upwards along the straight lines. The empty configuration satisfies the closed formula $(\mathsf{b}\, x)$, i.e., $\mathcal{E}_1 \models (\mathsf{b}\, x)$, even if the b-labelled event is not immediately enabled. Also $\mathcal{E}_1 \models (\mathsf{b}\, x) \wedge (\mathsf{d}\, y)$, since there are two possible (incompatible) computations that starts from the empty configuration and contain, respectively, a b-labelled and a d-labelled event. On the other hand, if $\varphi = (\mathsf{a}\, z)\langle z \rangle\, ((\mathsf{b}\, x) \wedge (\mathsf{d}\, y))$ then $\mathcal{E}_1 \not\models \varphi$ since after the execution of the a-labelled event, $\mathcal{E}_1$ reaches a configuration that does not admit a future containing an event labelled by d. As a further example, the formula φ above is satisfied by the PESs $\mathcal{E}_2$ and $\mathcal{E}_3$ in Fig. 1 corresponding respectively to the process $a.(b + d)$ and $a \mid (b + d)$, whereas the formula $(\mathsf{a}\, z)\langle z \rangle\, (\overline{z} < \mathsf{b}\, x)$ is satisfied only by $\mathcal{E}_3$.

It is worth noticing that the semantics of the binding operator does not prevent from choosing for z an event e that has been already bound to a different

variable, i.e., the environment function η needs not to be injective. This is essential to avoid the direct observation of conflicts. Consider for instance the PESs associated to the hhp-equivalent processes $a + a$ and a: in order to be also logically equivalent, they both must satisfy the formula $(\mathsf{a}\, z)(\mathsf{a}\, z')$. Hence for the second PES, both z and z' should be bound to the unique a-labelled event. On the other hand, observe that both PESs falsify the formula $(\mathsf{a}\, z)(\mathsf{a}\, z')\langle z\rangle\, \langle z'\rangle$, since either z and z' will be bound to the same event, which cannot be executed twice, or they will be bound to conflicting events.

Still, the logic as it is defined up to now is too powerful since it allows to observe conflicts through a combination of the binder and the execution modality. For instance, consider the PESs $\mathcal{E}_4$ and $\mathcal{E}_5$ in Fig. 1, corresponding to the processes $a.b + a.b$ and $a.b$ and take formula $\varphi = (\mathsf{a}\ x)(\mathsf{b}\ y)\langle x\rangle\, \neg\langle y\rangle$. Then $\mathcal{E}_5 \not\models \varphi$, while $\mathcal{E}_4 \models \varphi$, since only in $\mathcal{E}_4$ the variables x and y can be bound to conflicting events (e.g., x could be bound to the a-labelled event on the left and y to the b-labelled event on the right). In a similar way, the logic allows one to distinguish the PESs corresponding to any process from that corresponding to the non-deterministic choice between that process and itself, which instead are equated by virtually any behavioural equivalence.

In order to disallow the observation of conflicts and avoid this phenomenon, we restrict our logic to well-formed formulae, that "syntactically" ensure that (i) the free variables in any subformula will always refer to events consistent with the current configuration and (ii) the variables which are used as causes/non-causes, i.e., $\boldsymbol{x}$ and $\boldsymbol{y}$ in $(\boldsymbol{x}, \overline{\boldsymbol{y}} < \mathsf{a}\, z)\, \varphi$, will be bound to be pairwise consistent events.

This is formalised by the definition below. Consistency constraints are represented by a relation on variables $Co \subseteq Var \times Var$, where $(x, y) \in Co$ means that x and y must be bound to consistent events. We write $(\boldsymbol{x}, \boldsymbol{y})$ for the set $\{(x, y) : x \in \boldsymbol{x}\ \wedge\ y \in \boldsymbol{y}, x \neq y\}$.

Definition 11 ($\mathcal{L}$: the logic of well-formed formulae). *A formula φ is called* well-formed *if $Co \vdash \varphi$, for some $Co \subseteq Var \times Var$, where the entailment relation $\vdash$ is defined by the rules below:*

$$\frac{(\boldsymbol{x} \cup \boldsymbol{y}, \boldsymbol{x} \cup \boldsymbol{y}) \subseteq Co \qquad Co \cup (z, \boldsymbol{x} \cup \boldsymbol{y}) \vdash \varphi}{Co \vdash (\boldsymbol{x}, \overline{\boldsymbol{y}} < \mathsf{a}\, z)\varphi} \qquad \frac{(\{z\}, fv(\varphi)) \in Co \qquad Co \vdash \varphi}{Co \vdash \langle z\rangle\, \varphi}$$

$$\frac{Co \vdash \varphi}{Co \vdash \neg\varphi} \qquad \frac{Co \vdash \varphi_1 \qquad Co \vdash \varphi_2}{Co \vdash \varphi_1 \wedge \varphi_2} \qquad \frac{}{Co \vdash \top}$$

We denote by $\mathcal{L}$ the logic consisting of the well-formed formulae.

According to the first rule, the formula $(\boldsymbol{x}, \overline{\boldsymbol{y}} < \mathsf{a}\, z)\varphi$ is well-formed if the constraints in Co ensure that $\boldsymbol{x} \cup \boldsymbol{y}$ is pairwise consistent and φ can be proved well-formed using also the fact that the chosen z will be consistent with $\boldsymbol{x}$ and $\boldsymbol{y}$. The second rule, instead, says that $\langle z\rangle\, \varphi$ is well-formed when the constraints in Co ensure that z is consistent with the free-variables used in φ and φ itself is well-formed in Co.

As an example, the formula $(\mathsf{a}\, x)(\mathsf{b}\ y)\langle x\rangle\,\langle y\rangle$ is not well-formed (since it "executes" x and y which, in principle, can be bound to events in conflict) as opposed to $(\mathsf{a}\, x)(x < \mathsf{b}\ y)\langle x\rangle\,\langle y\rangle$, where the two executed events are certainly consistent (they are causally dependent). Notice that the formula $(\mathsf{a}\, x)(x < \mathsf{b}\ y)\langle y\rangle\,\langle x\rangle$ is also well formed, even if it is always false since it tries to execute an event before executing one of its causes. Finally, according to the rules $(\mathsf{a}\, x)\langle x\rangle\,(\mathsf{b}\ y)\langle y\rangle$ is also deemed a well-formed formula even though there is no constraint on y. This can be understood recalling that the semantics of the binding operator requires y to be bound to an event that is consistent with the current state, hence consistent with the event bound to x.

When dealing with the semantics of formulae in $\mathcal{L}$ we will always consider environments η that reflect the corresponding consistency constraints.

Definition 12 (legal environment). *Let $\mathcal{E}$ be a* PES*. Given a configuration $C \in \mathcal{C}(\mathcal{E})$ and a formula φ in $\mathcal{L}$, a* legal environment *for C and φ is an environment $\eta : Var \to E$ such that $fv(\varphi) \subseteq dom(\eta)$, $\eta(fv(\varphi))$ is consistent with C and there exists Co such that $Co \vdash \varphi$ and $\forall (x, y) \in Co$, $\eta(x) \frown \eta(y)$.*

The semantics of the logic $\mathcal{L}$ is then formally defined as in Definition 10, where η is assumed to be a legal environment for C and φ. It is easy to see that this assumption properly fits with Definition 10, i.e., whenever we recur on a subformula, we are surely checking satisfiability in an environment legal for the configuration and the formula.

Definition 13 (logical equivalence). *Let $\mathcal{L}'$ be a fragment of $\mathcal{L}$. We say that two* PES *$\mathcal{E}_1, \mathcal{E}_2$ are* logically equivalent *in $\mathcal{L}'$, written $\mathcal{E}_1 \equiv_{\mathcal{L}'} \mathcal{E}_2$ when they satisfy the same closed formulae.*

3.1 Examples and notation

In this subsection we provide some more examples illustrating the expressiveness of the logic. We start by introducing some handy notation, which will improve the readability of the formulae.

Immediate execution. We will write

$$\langle\!\langle \boldsymbol{x}, \overline{\boldsymbol{y}} < \mathsf{a}\, z \rangle\!\rangle\, \varphi$$

for the formula $(\boldsymbol{x}, \overline{\boldsymbol{y}} < \mathsf{a}\, z)\langle z\rangle\, \varphi$ that chooses a consistent event not belonging to the current configuration, and immediately executes it.

Steps. We introduce a notation also to predicate the existence, resp., the immediate execution, of concurrent events, specifying also their dependencies. We will write

$$((\boldsymbol{x}, \overline{\boldsymbol{y}} < \mathsf{a}\, z) \otimes (\boldsymbol{x}', \overline{\boldsymbol{y}'} < \mathsf{b}\, z'))\, \varphi \qquad \text{and} \qquad (\langle\!\langle \boldsymbol{x}, \overline{\boldsymbol{y}} < \mathsf{a}\, z \rangle\!\rangle \otimes \langle\!\langle \boldsymbol{x}', \overline{\boldsymbol{y}'} < \mathsf{b}\, z' \rangle\!\rangle)\varphi$$

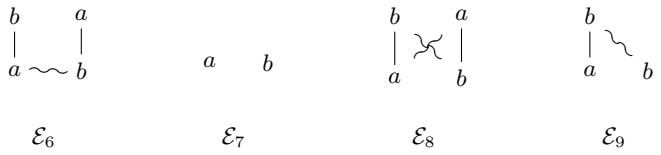

Fig. 2.

to declare the existence, resp., the immediate execution, of two concurrent events, labelled a and b, which are bound to z and z', and then φ holds. These notations correspond, respectively, to the formulae $(\boldsymbol{x}, \overline{\boldsymbol{y}} < \mathsf{a}\, z)(\boldsymbol{x}', \overline{\boldsymbol{y}', z} < \mathsf{b}\, z')\varphi$ and $(\,(\boldsymbol{x}, \overline{\boldsymbol{y}} < \mathsf{a}\, z) \otimes (\boldsymbol{x}', \overline{\boldsymbol{y}'} < \mathsf{b}\, z')\,)\langle z\rangle\,\langle z'\rangle\,\varphi$. In particular, the ability to perform a step consisting of two concurrent events labelled by a and b is simply expressed by the formula $\langle\!|\mathsf{a}\, x|\!\rangle \otimes \langle\!|\mathsf{b}\, y|\!\rangle$. Clearly, this notation can be generalised to the quantification and the immediate execution of any number of concurrent events.

Example 1 (Interleaving vs. True-concurrency). As a first example, consider the PESs $\mathcal{E}_6$ and $\mathcal{E}_7$ in Fig. 2. They are equated by interleaving equivalences and kept distinct by any true-concurrent equivalence. The formula $\varphi_1 = \langle\!|\mathsf{a}\, x|\!\rangle\langle\!|\overline{x} < \mathsf{b}\, y|\!\rangle = (\langle\!|\mathsf{a}\, x|\!\rangle \otimes \langle\!|\mathsf{b}\, y|\!\rangle)$ is true only on $\mathcal{E}_7$, while $\varphi_2 = \langle\!|\mathsf{a}\, x|\!\rangle\langle\!|x < \mathsf{b}\, y|\!\rangle$ is true only on $\mathcal{E}_6$.

Example 2 (Dependent vs Independent Events). The need of considering both causal dependency and independency in the binding operator of our logic is exemplified below. Consider the PESs $\mathcal{E}_6$ and $\mathcal{E}_8$ in Fig. 2. They are distinguished by any true-concurrent equivalence, but since they have the same causal structure, in order to pinpoint how they differ, the logic must be able to express the presence of two concurrent events: indeed $\mathcal{E}_8 \models \langle\!|\mathsf{a}\, x|\!\rangle \otimes \langle\!|\mathsf{b}\, y|\!\rangle$, while $\mathcal{E}_6 \not\models \langle\!|\mathsf{a}\, x|\!\rangle \otimes \langle\!|\mathsf{b}\, y|\!\rangle$. On the other hand, expressing causal dependencies between events is essential to distinguish, for instance, the PESs $\mathcal{E}_7$ and $\mathcal{E}_9$. These are equated by step bisimulation and distinguished by any equivalence which observes causality, e.g., pomset bisimulation.

Example 3 (Conflicting Futures). Consider the following two PESs, which can be proved to be hp-bisimilar but not hhp-bisimilar:

Intuitively, they differ since the causes of the c-labelled and d-labelled events are in conflict in the first PES and independent in the second one. This is captured by the formula $\varphi = ((\mathsf{a}\, x) \otimes (\mathsf{b}\, y))((x < \mathsf{c}\, z_1) \wedge (y < \mathsf{d}\, z_2))$, which is satisfied only by the right-most PES. Notice that the formula φ exploits the ability of the logic $\mathcal{L}$ of quantifying over events in conflict with previously bound events: formula φ is satisfied in the rightmost PES by binding x and y to the rightmost a-labelled and

b-labelled events; then both z_1 and z_2 are bound to events which are in conflict with either x or y. For this, the possibility of quantifying over an event without executing it is essential: the formula $\varphi' = (\langle\!|\mathsf{a}\, x|\!\rangle \otimes \langle\!|\mathsf{b}\, y|\!\rangle)((x < \mathsf{c}\, z_1) \wedge (y < \mathsf{d}\, z_2))$ would be false in both PESs since the execution of the first two events leads to a configuration that is no further extensible.

As a final example, consider the two CCS processes $P = a|(b{+}c){+}a|b{+}b|(a{+}c)$ and $Q = a|(b{+}c) + b|(a{+}c)$. They contain no causal dependencies, but they exhibit a different interplay between concurrency and branching. Accordingly, the corresponding PESs can be proved to be hp-bisimilar but not hhp-bisimilar. Intuitively, this difference comes from the fact that only the process P includes two concurrent events a and b such that, independently from their execution order, no c-labelled event will be enabled. Such a difference can be expressed in $\mathcal{L}$ by the formula $((\mathsf{a}\, x) \otimes (\mathsf{b}\, y))(\neg(\overline{x} < \mathsf{c}\, z) \wedge \neg(\overline{y} < \mathsf{c}\, z'))$, which is satisfied only by the PES corresponding to P.

4 A logical characterisation of concurrent equivalences

In this section we study the logical equivalences induced by fragments of $\mathcal{L}$. We have already argued that no formula in $\mathcal{L}$ distinguishes the PESs a and $a\#a$, hence the logical equivalence induced by $\mathcal{L}$ is surely coarser than isomorphism. We next show that the logical equivalence induced by $\mathcal{L}$ is hhp-bisimulation. Moreover, we identify suitable fragments of $\mathcal{L}$ corresponding to coarser equivalences.

Theorem 1 (hhp-bisimilarity). *Let $\mathcal{E}_1$ and $\mathcal{E}_2$ be PESs. Then $\mathcal{E}_1 \sim_{hhp} \mathcal{E}_2$ iff $\mathcal{E}_1 \equiv_{\mathcal{L}} \mathcal{E}_2$.*

4.1 From Hennessy-Milner logic to HP-logic

Hhp-bisimilarity is the finest equivalence in the spectrum of true concurrent equivalences in [2]. Coarser equivalences such as step, pomset and hp-bisimilarity, can be captured by suitable fragments of $\mathcal{L}$ summarised in Fig. 3, which can be viewed as the logical counterpart of the true concurrent spectrum.

Interestingly, in each of these fragments after predicating the existence of an event we must execute it. As a consequence, differently from what happens in the full logic, in the fragments it is impossible to refer to events in conflict with already observed events. Intuitively, behavioural equivalences up to hp-bisimilarity observe events only by executing them. Hence they cannot fully capture the interplay between concurrency and branching, which is indeed distinctive of hhp-equivalence.

Hennessy-Milner Logic. A first simple observation is that Hennessy-Milner logic can be recovered as the fragment of $\mathcal{L}$ where only the derived modality $\langle\!|\mathsf{a}\, x|\!\rangle\varphi$ (with no references to causally dependent/concurrent events) is allowed. In words, whenever we state the existence of an enabled event we are forced to execute it. Moreover, since no dependencies can be expressed, the bound variable

HM Logic	$\mathcal{L}_{HM}$	$\varphi \ ::= \ \langle\!\langle \mathsf{a}\, x \rangle\!\rangle \varphi \ \mid \ \varphi \wedge \varphi \ \mid \ \neg\varphi \ \mid \ \top$
Step Logic	$\mathcal{L}_s$	$\varphi \ ::= \ (\langle\!\langle \mathsf{a_1}\, x_1 \rangle\!\rangle \otimes \cdots \otimes \langle\!\langle \mathsf{a_n}\, x_n \rangle\!\rangle)\ \varphi \ \mid \ \varphi \wedge \varphi \ \mid \ \neg\varphi \ \mid \ \top$
Pomset Logic	$\mathcal{L}_p$	$\varphi \ ::= \ \langle\!\langle \boldsymbol{x}, \overline{\boldsymbol{y}} < \mathsf{a}\, z \rangle\!\rangle \varphi \ \mid \ \neg\varphi \ \mid \ \varphi \wedge \varphi \ \mid \ \top$ where $\neg$, $\wedge$ are used only on closed formulae.
HP Logic	$\mathcal{L}_{hp}$	$\varphi \ ::= \ \langle\!\langle \boldsymbol{x}, \overline{\boldsymbol{y}} < \mathsf{a}\, z \rangle\!\rangle \varphi \ \mid \ \neg\varphi \ \mid \ \varphi \wedge \varphi \ \mid \ \top$

Fig. 3. Fragments corresponding to behavioral equivalences

x is irrelevant. The induced logical equivalence is thus bisimilarity [3] (recall that we consider only image finite PES's).

Step logic. A fragment $\mathcal{L}_s$ corresponding to step bisimilarity naturally arises as a generalisation of HM logic, where we can refer to sets of concurrently enabled events. More precisely, as shown in Fig. 3, $\mathcal{L}_s$ is the fragment of $\mathcal{L}$ where only the derived modality $\langle\!\langle \mathsf{a_1}\, x_1 \rangle\!\rangle \otimes \cdots \otimes \langle\!\langle \mathsf{a_n}\, x_n \rangle\!\rangle$ is used, allowing to predicate on the possibility of performing a parallel step, but without any reference to causal dependencies. Note that all formulae in $\mathcal{L}_s$ are closed, and thus environments are irrelevant in their semantics (as well as the names of the bound variables).

As an example, consider the two PESs $\mathcal{E}_6$ and $\mathcal{E}_7$ in Fig. 2. They are bisimilar but not step bisimilar since only $\mathcal{E}_7$ can execute the step consisting of a and b; accordingly, the formula $\langle\!\langle \mathsf{a} \rangle\!\rangle \otimes \langle\!\langle \mathsf{b} \rangle\!\rangle$ in $\mathcal{L}_s$ is true only on $\mathcal{E}_7$.

Theorem 2 (step bisimilarity). *Let $\mathcal{E}_1$ and $\mathcal{E}_2$ be PESs. Then $\mathcal{E}_1 \sim_s \mathcal{E}_2$ iff $\mathcal{E}_1 \equiv_{\mathcal{L}_s} \mathcal{E}_2$.*

Pomset logic. The logic $\mathcal{L}_p$ for pomset bisimilarity consists of a fragment of $\mathcal{L}$ which essentially predicates about the possibility of executing pomset transitions. Even in $\mathcal{L}_p$ the events must be immediately executed when quantified, but it is now possible to refer to dependencies between events. However, propositional connectives (negation and conjunction) can be used only on closed formulae.

Intuitively, in this logic closed formulae characterize the execution of a pomset. Then, the requirement that the propositional operators are used only on closed (sub)formulae prevents pomset transitions from being causally linked to the events in the past. These ideas are formalised by the results below, starting with a lemma showing that the execution of a pomset can be characterized as a corresponding formula in $\mathcal{L}_p$.

Definition 14 (pomsets as formulae in $\mathcal{L}_p$). *Let $p_x = (\{x_1, \ldots, x_n\}, \leq_{p_x}, \lambda_{p_x})$ be a labelled poset, whose elements $\{x_1, \ldots, x_n\}$ are variables ordered by $\leq_{p_x}$. Given a formula $\varphi \in \mathcal{L}_p$, we denote by $\langle\!\langle\, p_x \rangle\!\rangle \varphi$ the formula inductively defined as follows. If p_x is empty then $\langle\!\langle\, p_x \rangle\!\rangle \varphi = \varphi$. If $p_x = p'_x \cup \{x\}$, where x is*

maximal with respect to $\leq_{p_x}$*, then if* $\boldsymbol{y} = \{x' \in p'_x \mid x' \leq_{p_x} x\}$*,* $\boldsymbol{z} = p'_x \setminus \boldsymbol{y}$*, and* $\lambda_{p_x}(x) = \mathsf{a}$*, then* $\langle\!\langle\, p_x \,\rangle\!\rangle \varphi = \langle\!\langle\, p'_x \,\rangle\!\rangle \langle\!\langle \boldsymbol{y}, \overline{\boldsymbol{z}} < \mathsf{a}\, x \rangle\!\rangle \varphi$*.*

Lemma 1 (pomsets in $\mathcal{L}_p$). *Let* $\mathcal{E}$ *be a* PES *and let* $C \in \mathcal{C}(\mathcal{E})$ *be a configuration. Given a labelled poset* $p_x = (\{x_1, \ldots, x_n\}, \leq_{p_x}, \lambda_{p_x})$*, then*

$$\mathcal{E}, C \models_\eta \langle\!\langle\, p_x \,\rangle\!\rangle\, \varphi \quad \textit{iff}$$
$$C \xrightarrow{X} C' \textit{ where } X = \{e_1, \ldots, e_n\} \textit{ is a pomset s.t. } X \sim p_x$$
$$\textit{and } \mathcal{E}, C' \models_{\eta'} \varphi, \textit{ with } \eta' = \eta[x_1 \mapsto e_1, \ldots, x_n \mapsto e_n]$$

As an example, consider the two PESs $\mathcal{E}_7$ and $\mathcal{E}_9$ in Fig. 2. They are step bisimilar but not pomset bisimilar since only the second one can execute the pomset $(\{\mathsf{a}, \mathsf{b}\}, \mathsf{a} < \mathsf{b})$. Accordingly, the formula $\varphi = \langle\!\langle \mathsf{a}\, x \rangle\!\rangle \langle\!\langle x < \mathsf{b}\, y \rangle\!\rangle$ in $\mathcal{L}_p$ is satisfied only by $\mathcal{E}_9$.

Theorem 3 (pomset bisimilarity). *Let* $\mathcal{E}_1$ *and* $\mathcal{E}_2$ *be* PES*s. Then* $\mathcal{E}_1 \sim_p \mathcal{E}_2$ *iff* $\mathcal{E}_1 \equiv_{\mathcal{L}_p} \mathcal{E}_2$*.*

History preserving logic. The fragment $\mathcal{L}_{hp}$ corresponding to hp-bisimilarity is essentially the same as for pomset logic, where we relax the condition about closedness of the formulae the propositional connectives are applied to. Intuitively, in this way a formula $\varphi \in \mathcal{L}_{hp}$, besides expressing the possibility of executing a pomset p_x, also predicates about its dependencies with previously executed events (bound to the free variables of φ).

The two PESs below can be proved to be pomset equivalent but not hp-equivalent:

Intuitively, they allow the same pomset transitions, but they have a different "causal branching". Indeed, only in the left-most PESs after the execution of an a-labelled event we can choose between an independent and a dependent b-labelled event. Formally, the formula $\langle\!\langle \mathsf{a}\, x \rangle\!\rangle (\langle\!\langle \overline{x} < \mathsf{b}\, y \rangle\!\rangle \wedge (\langle\!\langle x < \mathsf{b}\, z \rangle\!\rangle)$ in $\mathcal{L}_{hp}$ is true only on the left-most PES.

Theorem 4 (hp-bisimilarity). *Let* $\mathcal{E}_1$ *and* $\mathcal{E}_2$ *be* PES*s. Then* $\mathcal{E}_1 \sim_{hp} \mathcal{E}_2$ *iff* $\mathcal{E}_1 \equiv_{\mathcal{L}_{hp}} \mathcal{E}_2$*.*

5 Conclusions: related and future work

We have introduced a logic for true concurrency, which allows to predicate on events in computation and their mutual dependencies (causality and concurrency). The logic subsumes standard HM logic and provides a characterisation of the most widely known true concurrent behavioural equivalences: hhp-bisimilarity is the logical equivalence induced by the full logic, and suitable

fragments are identified which induce hp-bisimilarity, pomset and step bisimilarity.

As we mentioned in the introduction, there is a vast literature relating logical and operational views of true concurrency, however, to the best of our knowledge, a uniform logical counterpart of the true concurrent spectrum is still missing. An exhaustive account of the related literature is impossible; we just recall here the approaches that most closely relate to our work.

In [5, 15, 16] the causal structure of concurrent systems is pushed into the logic. The paper [5] considers modalities which describe pomset transitions, thus providing an immediate characterization of pomset bisimilarity. Moreover, [5, 15, 16] show that by tracing the history of states and adding the possibility of reverting pomset transitions, one obtains an equivalence coarser than hp-bisimilarity and incomparable with pomset bisimilarity, called weak hp-bisimilarity. Our logic intends to be more general by also capturing the interplay between concurrency and branching, which is not observable at the level of hp-bisimilarity.

A recent work [8, 9] introduces a fixpoint modal logic for true concurrent models, called Separation Fixpoint Logics (SFL). This includes modalities which specify the execution of an action causally dependent/independent on the last executed one. Moreover, a "separation operator" deals with concurrently enabled actions. The fragment of the logic without the separation operator is shown to capture hp-bisimilarity, while the full logic induces an equivalence which lies in between hp- and hhp-bisimilarity, still being decidable for finite state systems. The approach of [8, 9] is inspired by the so-called Independence-Friendly Modal Logic (IFML) [13], which includes a modality that allows to specify that the current executed action is independent from a number of previously executed ones. In this sense IFML is similar in spirit to our logic. Although, most of the equivalences induced by fragments of IFML are not standard in the true concurrent spectrum, a deeper comparison with this approach represents an interesting open issue.

Several classical papers have considered temporal logics with modalities corresponding to the "retraction" or "backward" execution of computations. In particular [10–12] study a so-called path logic with a future perfect (also called past tense) modality: $@\mathsf{a}\,\varphi$ is true when φ holds in a state which can reach the current one with an a-transition. When interpreted over transition systems with independence, in absence of autoconcurrency, the logic characterises hhp-bisimilarity. In [10] it is shown that, taking autoconcurrency into account, the result can be extended at the price of complicating the logic (roughly, the logic needs an operator to undo a specific action performed in the past).

Compared to these works, the main novelty of our approach resides in the fact that the logic $\mathcal{L}$ provides a characterisation of the different equivalences in a simple, unitary logical framework. In order to enforce this view, we intend to pursue a formal comparison with the logics for concurrency introduced in the literature. It is easy to see that the execution modalities of [8, 9] can be encoded in $\mathcal{L}$ since they only refer to the last executed event, while the formulae in $\mathcal{L}$ can refer to any event executed in the past. On the other hand, the "separation

operator" of [8, 9], as well as the backward modalities mentioned above (past tense, future perfect, reverting pomset transitions) are not immediately encodable in $\mathcal{L}$. A deeper investigation would be of great help in shading further light on the truly concurrent spectrum. Moreover $\mathcal{L}$ suggests an alternative, forward-only, operational definition of hhp-bisimulation, which could be inspiring also for other reverse bisimulations [17].

As a byproduct of such an investigation, we foresee the identification of interesting extensions of the concurrent spectrum, both at the logical and at the operational side. For instance, a preliminary investigation ([18]) suggests that the fragment of $\mathcal{L}$ where only events consistent with the current environment can be quantified induces an equivalence which admits a natural operational definition and lies in between hp- and hhp-bisimilarity, still being different from the equivalences in [8, 9]. Moreover, the logic in its present form only allows to describe properties of finite, bounded computations. A more powerful specification logic, well-suited for describing properties of unbounded, possibly infinite computations can be obtained by enriching $\mathcal{L}$ with some form of recursion. In particular, from some first experiments ([18]), the idea of "embedding" our logic into a first order modal mu-calculus in the style of [19, 20] seems promising. For this purpose, also the fixpoint extension of the Independence-Friendly Modal Logic in [21] could be inspiring. The resulting logic would allow to express non-trivial causal properties, like "any a action can be always followed by a causally related b action in at most three steps", or "an a action can be always executed in parallel with a b action".

Connected to this, model-checking and decidability issues are challenging directions of future investigation (see [22] for a survey focussed on partial order temporal logics). It is known that hhp-bisimilarity is undecidable, even for finite state systems [23], while hp-bisimilarity is decidable. Characterising decidable fragments of the logic could be helpful in drawing a clearer separation line between decidability and undecidability of concurrent equivalences. A promising direction is to impose a bound on the "causal depth" of the future which the logic can quantify on. In this way one gets a chain of equivalences, coarser than hhp-bisimilarity, which should be closely related with n-hhp bisimilarities introduced and shown to be decidable in [24]. As for verification, we aim at investigating the automata-theoretic counterpart of the logic. In previous papers, hp-bisimilarity has been characterised in automata-theoretic terms using HD-automata [6] or Petri nets [7]. It seems that HD-automata [6] could provide a suitable automata counterpart of the fragment $\mathcal{L}_{hp}$. Also the game-theoretical approach proposed in [8, 9] for the fixpoint separation logic can be a source of inspiration.

References

1. van Glabbeek, R.: The linear time – branching time spectrum I; the semantics of concrete, sequential processes. In: Handbook of Process Algebra. Elsevier (2001) 3–99
2. van Glabbeek, R., Goltz, U.: Refinement of actions and equivalence notions for concurrent systems. Acta Informatica **37**(4/5) (2001) 229–327

3. Hennessy, M., Milner, R.: Algebraic laws for nondeterminism and concurrency. Journal of the ACM **32** (1985) 137–161
4. Winskel, G., Nielsen, M.: Models for concurrency. In: Handbook of logic in Computer Science. Volume 4. Clarendon Press (1995)
5. De Nicola, R., Ferrari, G.: Observational logics and concurrency models. In: Proceedings of FST-TCS'90. Volume 472 of LNCS., Springer (1990) 301–315
6. Montanari, U., Pistore, M.: Minimal transition systems for history-preserving bisimulation. In: Proceedings of STACS'97. Volume 1200 of LNCS., Springer (1997) 413–425
7. Vogler, W.: Deciding history preserving bisimilarity. In: Proceedings of ICALP'91. Volume 510 of LNCS., Springer (1991) 495–505
8. Gutierrez, J., Bradfield, J.C.: Model-checking games for fixpoint logics with partial order models. In: Proceedings of CONCUR'09. Volume 5710 of LNCS., Springer (2009) 354–368
9. Gutierrez, J.: Logics and bisimulation games for concurrency, causality and conflict. In: Proceedings of FoSSaCS'09. Volume 5504 of LNCS., Springer (2009) 48–62
10. Nielsen, M., Clausen, C.: Games and logics for a noninterleaving bisimulation. Nordic Journal of Computing **2**(2) (1995) 221–249
11. 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)
12. Hennessy, M., Stirling, C.: The power of the future perfect in program logics. Information and Control **67**(1-3) (1985) 23–52
13. Bradfield, J., Fröschle, S.: Independence-friendly modal logic and true concurrency. Nordic Journal of Computing **9**(1) (2002) 102–117
14. Winskel, G.: Event Structures. In: Petri Nets: Applications and Relationships to Other Models of Concurrency. Volume 255 of LNCS., Springer (1987) 325–392
15. Pinchinat, S., Laroussinie, F., Schnoebelen, P.: Logical characterization of truly concurrent bisimulation. Technical Report 114, LIFIA-IMAG, Grenoble (1994)
16. Cherief, F.: Back and forth bisimulations on prime event structures. In: Proceedings of PARLE'92. Volume 605 of LNCS., Springer (1992) 843–858
17. Phillips, I., Ulidowski, I.: Reverse bisimulations on stable configuration structures. In: Proc. of SOS'09. Volume 18 of Electronic Proceedings in Theoretical Computer Science. (2010) 62–76
18. Baldan, P., Crafa, S.: A logic for true concurrency. Full version. In preparation. (2010)
19. Dam, M.: Model checking mobile processes. Information and Computation **129**(1) (1996) 35–51
20. Dam, M., Fredlund, L.Å., Gurov, D.: Toward parametric verification of open distributed systems. In: Proceedings of COMPOS'97. Volume 1536 of LNCS., Springer (1998) 150–185
21. Bradfield, J., Kreutzer, K.: The complexity of independence-friendly fixpoint logic. In: Proceedings of CLS'05. Volume 3634 of LNCS., Springer (2005) 355–368
22. Penczek, W.: Branching time and partial order in temporal logics. In: Time and Logic: A Computational Approach, UCL Press (1995) 179–228
23. Jurdzinski, M., Nielsen, M., Srba, J.: Undecidability of domino games and hhp-bisimilarity. Information and Computation **184**(2) (2003) 343–368
24. Fröschle, S., Hildebrandt, T.: On plain and hereditary history-preserving bisimulation. In: Proceedings of MFCS'99. Volume 1672 of LNCS., Springer (1999) 354–365