Local Model Checking in a Logic for True Concurrency

Paolo Baldan and Tommaso Padoan(✉)

Dipartimento di Matematica, Università di Padova, Padova, Italy
{baldan,padoan}@math.unipd.it

Abstract. We provide a model-checking technique for a logic for true concurrency, whose formulae predicate about events in computations and their causal dependencies. The logic, that represents the logical counterpart of history-preserving bisimilarity, is naturally interpreted over event structures. It includes minimal and maximal fixpoint operators and thus it can express properties of infinite computations. Global algorithms are not convenient in this setting, since the event structure associated with a system is typically infinite (even if the system is finite state), a fact that makes also the decidability of model-checking non-trivial. Focusing on the alternation free fragment of the logic, along the lines of some classical work for the modal mu-calculus, we propose a local model-checking algorithm. The algorithm is given in the form of a tableau system, for which, over a class of event structures satisfying a suitable regularity condition, we prove termination, soundness and completeness.

1 Introduction

When dealing with concurrent and distributed systems, a partial order approach to the semantics can be appropriate for providing a precise account of the behavioural steps and of their dependencies, like causality and concurrency. This is normally referred to as a true concurrent approach to the semantics and opposed to the so-called interleaving approach where concurrency of actions is reduced to the non-deterministic choice among their possible sequentializations. True concurrent models can be convenient also because, thanks to an explicit representation of concurrency, they provide some relief to the so-called state-space explosion problem in the analysis of concurrent systems, which instead occurs more severely in interleaving approaches (see, e.g., [1]).

A number of true concurrent behavioural equivalences have been defined which take into account different concurrency features of computations (see, e.g., [2]). On the logical side, various behavioural logics have been proposed capable of expressing causal properties of computations (see, e.g., [3–8] just to mention a few and [9,10] for more references) and corresponding model-checking problems have been considered (see, e.g., [11–15]).

Recently, the logical characterisation of true concurrent behavioural equivalences has received a renewed interest and corresponding event-based logics

Partially supported by the University of Padova project ANCORE.

J. Esparza and A.S. Murawski (Eds.): FOSSACS 2017, LNCS 10203, pp. 407–423, 2017.
DOI: 10.1007/978-3-662-54458-7_24

have been introduced [9,10], interpreted over event structures [16]. Logic formulae include variables which can be bound to events in computations and describe their dependencies, namely causality and concurrency. The expressiveness of such logics is sufficient to provide a logical characterisation of the main behavioural equivalences in the true concurrent spectrum [2]. Hereditary history preserving (hhp-)bisimilarity [4], the finest equivalence in the spectrum in [2], corresponds to the full logics, i.e., two systems are hhp-bisimilar if and only if they satisfy the same logical formulae, and fragments can be identified corresponding to coarser behavioural equivalences. The corresponding model-checking problem, instead, has not been investigated. Decidability itself is an issue, since event structure models are infinite even for finite state systems and problems in this framework have often revealed to sit on the border between decidability and undecidability.

In this paper we focus on a fragment of the event-based logic in [10], referred to as $\mathcal{L}_{hp}$, corresponding to a classical equivalence in the spectrum, namely history preserving (hp-)bisimilarity [17–19]. The logic is extended with minimal and maximal fixpoint operators, in mu-calculus style, in order to express interesting properties of infinite computations. Hp-bisimilarity is known to be decidable for finite safe Petri nets [20,21]. However, the question remains open on whether the corresponding model checking problem for $\mathcal{L}_{hp}$ is decidable over some interesting class of systems.

We answer positively to the question, defining a model checking procedure for $\mathcal{L}_{hp}$ over a class of event structures satisfying a suitable regularity condition.

Since the model checking procedure acts on event structures which are normally infinite even when generated from finite state systems, global algorithms fully exploring the structure are not a convenient choice. We are naturally led to focus on local algorithms, in the style of [22], exploring only the part of a model needed to asses the property. Along the lines of [22], the model checking procedure is given in the form of a tableau system. In order to check whether a system model satisfies a given formula a set of proof trees is constructed by applying a suitable set of rules that reduce the satisfaction of a formula in a given state to the satisfaction of suitably generated subformulae. In this way, the state space is explored "on demand" only in the part relevant for deciding the satisfaction of the formula. The presence of fixpoint operators makes the issue of sound termination quite delicate and non-trivial already in the original approach that works on finite transition systems. In the setting of this paper, this is further complicated by the infiniteness of the event structure model of any non-trivial system.

In order to single out a setting that ensures termination, soundness and completeness of the model checking procedure we take two key choices. The first is the restriction to a class of event structures having a finitary flavour, which we call strongly regular event structures. Recall that regular event structures [23] are characterised by the fact that the number of sub-structures arising as residuals of the original event structure after some steps of computations is finite up to isomorphism. The intuition is that, after going sufficiently in depth, the event structure starts repeating cyclically. For strongly regular event structures the

requirement is strengthened by asking the finiteness of the residuals extended with an event from the past. This is important in our setting since $\mathcal{L}_{hp}$ formulae can express history-based properties that depend not only on the future but also on past events. The second choice is the use of fixpoints in an alternation free fashion: minimal and maximal fixpoints can be nested, but without creating mutual dependencies (technically, the propositional variable bound by a minimal fixpoint operator cannot be in the scope of a maximal fixpoint operator and vice versa). When dealing with finite structures, alternation freeness allows for efficient verification procedures [24]. Here the essential fact is that it ensures that, over finitely branching transition systems, the formulae are continuous in the sense of [25,26], hence fixpoints are reached in at most ω steps (higher ordinals are never needed).

The paper is organised as follows. In Sect. 2 we recall some basics on (prime) event structures and we define the regularity property of interest. In Sect. 3 we introduce the true concurrent logic $\mathcal{L}_{hp}$, its fixpoint extension and the alternation free fragment. In Sect. 4 we define our model checking procedure as a tableau system, and we prove its soundness, completeness and termination. Finally, in Sect. 5 we discuss some related work and outline directions of future research.

2 Event Structures and Regularity

Prime event structures [16] 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 finite 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 $e \# e''$.*

The PES*s $\mathcal{E}_1 = \langle E_1, \leq_1, \#_1 \rangle$, $\mathcal{E}_2 = \langle E_2, \leq_2, \#_2 \rangle$ are* isomorphic*, written $\mathcal{E}_1 \sim \mathcal{E}_2$, when there is a bijection $\iota : E_1 \to E_2$ such that for all $e_1, e_1' \in E_1$ it holds $e_1 \leq_1 e_1'$ iff $\iota(e_1) \leq_2 \iota(e_1')$ and $e_1 \#_1 e_1'$ iff $\iota(e_1) \#_2 \iota(e_1')$ and $\lambda(e_1) = \lambda(\iota(e_1))$.*

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* 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 $\mathcal{C}(\mathcal{E})$.*

The empty set of events $\emptyset$ is always a configuration, which can be interpreted as the initial state of the computation. The evolution of a system can be represented by a transition system where configurations are states.

Definition 4 (Transition system). *Let $\mathcal{E}$ be a* PES *and let $C \in \mathcal{C}(\mathcal{E})$. Given $e \in E \backslash C$ such that $C \cup \{e\} \in \mathcal{C}(\mathcal{E})$, and $X, Y \subseteq C$ with $X < e$, $Y \parallel e$ we write $C \xrightarrow{X, \overline{Y} < e}_{\lambda(e)} C \cup \{e\}$, possibly omitting X, Y or the label $\lambda(e)$. The set of* enabled *events at a configuration C is defined as $en(C) = \{e \in E \mid C \xrightarrow{e} C'\}$.*

Transitions are labelled by the executed event e. In addition, they can report its label $\lambda(e)$, a subset of causes X and a set of events $Y \subseteq C$ concurrent with e.

We already mentioned that the PES associated with a non-trivial system exhibiting a cyclic behaviour is infinite. We next introduce a class of PESs enjoying a finitary property referred to as strong regularity.

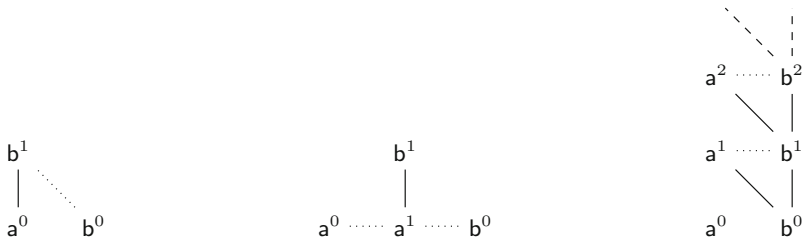
Definition 5 (Residual). *Let $\mathcal{E}$ be a* PES. *For a configuration $C \in \mathcal{C}(\mathcal{E})$, the* residual *of $\mathcal{E}$ after C, is defined as $\mathcal{E}[C] = \{e \in E \backslash C \;\mid\; C \frown e\}$.*

The residual of $\mathcal{E}$ can be seen as a PES, endowed with the restriction of the causality and conflict relations of $\mathcal{E}$. Intuitively, it represents the PES that remains to be executed after the computation expressed by C.

Recall that a PES $\mathcal{E}$ is *regular* [23] when the set of residuals $\{\mathcal{E}[C] \mid C \in \mathcal{C}(\mathcal{E})\}$ is finite up to isomorphism and there exists an integer k such that $|en(C)| \leq k$ for all $C \in \mathcal{C}(\mathcal{E})$. The first condition roughly means that there is a finite number of residuals over which the computation cycles. The second condition requires that the transition system of configurations is boundedly branching, with a finite bound. Here we strengthen the first condition by requiring the finiteness of the residuals extended with an event from the past. Given $C \in \mathcal{C}(\mathcal{E})$ and $e \in C$, we denote by $\mathcal{E}[C] \cup \{e\}$ the PES obtained from $\mathcal{E}[C]$ by adding event e with the causal dependencies it had in the original PES $\mathcal{E}$.

Definition 6 (Strong regularity). *A* PES *$\mathcal{E}$ is called* strongly regular *when the set $\{\mathcal{E}[C] \cup \{e\} \mid C \in \mathcal{C}(\mathcal{E}) \;\wedge\; e \in C\}$ is finite up to isomorphism of* PES*s and there exists an integer k such that $|en(C)| \leq k$ for all $C \in \mathcal{C}(\mathcal{E})$.*

Clearly, each strongly regular PES is regular. Strongly regular PESs can be shown, e.g., to include regular trace PESs, the class of event structures associated with finite safe Petri nets [23].



(a) $\mathcal{E}_1$ (b) $\mathcal{E}_2$ (c) $\mathcal{E}_3$

Fig. 1. Some examples of PESs.

Some simple PESs are depicted in Fig. 1. Graphically, dotted lines represent immediate conflicts and the causal partial order proceeds upwards along the straight lines. Events are denoted by their labels, possibly with superscripts. For instance, in the PES $\mathcal{E}_1$, there are two b-labelled events, b^0 and b^1 in conflict. The first is caused by the only a-labelled event a^0 and the other is concurrent with it. The infinite PES $\mathcal{E}_3$ corresponds to the CCS process $\mathsf{a} \parallel \mathsf{b}.C$ where $C = \mathsf{a} + \mathsf{b}.C$. It is strongly regular since it has seven (equivalence classes of) residuals extended with an event from the past $\mathcal{E}_3[\{\mathsf{a}^0\}] \cup \{\mathsf{a}^0\} = \mathcal{E}_3[\{\mathsf{b}^0\}] \cup \{\mathsf{b}^0\} = \mathcal{E}_3$, $\mathcal{E}_3[\{\mathsf{a}^0, \mathsf{b}^0\}] \cup \{\mathsf{a}^0\}$, $\mathcal{E}_3[\{\mathsf{a}^0, \mathsf{b}^0\}] \cup \{\mathsf{b}^0\}$, $\mathcal{E}_3[\{\mathsf{b}^0, \mathsf{a}^1\}] \cup \{\mathsf{b}^0\}$, $\mathcal{E}_3[\{\mathsf{b}^0, \mathsf{a}^1\}] \cup \{\mathsf{a}^1\}$, $\mathcal{E}_3[\{\mathsf{a}^0, \mathsf{b}^0, \mathsf{a}^1\}] \cup \{\mathsf{a}^0\} \simeq \mathcal{E}_3[\{\mathsf{a}^0, \mathsf{b}^0, \mathsf{a}^1\}] \cup \{\mathsf{a}^1\}$, and $\mathcal{E}_3[\{\mathsf{a}^0, \mathsf{b}^0, \mathsf{a}^1\}] \cup \{\mathsf{b}^0\}$.

3 A Logic for True Concurrency

In this section we introduce the syntax and the semantics of the logic for concurrency of interest in the paper. Originally defined in [10], the logic has formulae that predicate over executability of events in computations and on their dependency relations (causality and concurrency).

3.1 Syntax

As already mentioned, logic formulae include event variables from a fixed denumerable set *Var*, denoted by $x, y, \ldots$. In order to keep the notation simple, tuples of variables like $x_1, \ldots, x_n$ will be denoted by a corresponding boldface letter $\mathbf{x}$ and, abusing the notation, tuples will be often used as sets. The logic, in positive form, besides the standard propositional connectives $\wedge$ and $\vee$, includes diamond and box modalities. The formula $\langle\!|\mathbf{x}, \overline{\mathbf{y}} < \mathsf{a}\, z|\!\rangle\, \varphi$ holds when in the current configuration an a-labelled event e is enabled which causally depends on the events bound to the variables in $\mathbf{x}$ and is concurrent with those in $\mathbf{y}$. Event e is executed and bound to variable z, and then the formula φ must hold in the resulting configuration. Dually, $[\![\mathbf{x}, \overline{\mathbf{y}} < \mathsf{a}\, z]\!]\, \varphi$ is satisfied when all a-labelled events causally dependent on $\mathbf{x}$ and concurrent with $\mathbf{y}$ bring to a configuration where φ holds.

Fixpoint operators resort to propositional variables. In order to let them interact correctly with event variables, whose values can be passed from an iteration to the next one in the recursion, we use abstract propositions.

We fix a denumerable set $\mathcal{X}^a$ of *abstract propositions*, ranged over by X, Y, ..., that are intended to represent formulae possibly containing (unnamed) free event variables. Each abstract proposition has an arity $ar(X)$, which indicates the number of free event variables in X. An abstract proposition X can be turn into a formula by specifying a name for its free variables. For $\mathbf{x}$ such that $|\mathbf{x}| = ar(X)$, we write $X(\mathbf{x})$ to indicate the abstract proposition X whose free event variables are named $\mathbf{x}$. When $ar(X) = 0$ we will write X instead of $X(\epsilon)$ omitting the trailing empty tuple of variables. We call $X(\mathbf{x})$ a *proposition* and denote by $\mathcal{X}$ the set of all propositions.

Definition 7 (Syntax). *The syntax of $\mathcal{L}_{hp}$ over the sets of event variables Var, abstract propositions $\mathcal{X}^a$ and labels Λ is defined as follows:*

$$\begin{array}{rl} \varphi ::= X(\mathbf{x}) \mid \mathsf{T} \mid \varphi \wedge \varphi \mid \langle\!\langle \mathbf{x}, \overline{\mathbf{y}} < \mathsf{a}\, z \rangle\!\rangle \varphi \mid \nu X(\mathbf{x}).\varphi \\ \quad \mid \mathsf{F} \mid \varphi \vee \varphi \mid [\![\mathbf{x}, \overline{\mathbf{y}} < \mathsf{a}\, z]\!] \varphi \mid \mu X(\mathbf{x}).\varphi \end{array}$$

The free event variables of a formula φ are denoted by $fv(\varphi)$ and defined in the obvious way. Just note that the modalities act as binders for the variable representing the event executed, hence $fv(\langle\!\langle \mathbf{x}, \overline{\mathbf{y}} < \mathsf{a}\, z \rangle\!\rangle \varphi) = fv([\![\mathbf{x}, \overline{\mathbf{y}} < \mathsf{a}\, z]\!] \varphi) = (fv(\varphi) \backslash \{z\}) \cup \mathbf{x} \cup \mathbf{y}$. For formulae $\nu X(\mathbf{x}).\varphi$ and $\mu X(\mathbf{x}).\varphi$ we require that $fv(\varphi) = \mathbf{x}$. The free propositions in φ, i.e., the propositions not bound by μ or ν, are denoted by $fp(\varphi)$. When both $fv(\varphi)$ and $fp(\varphi)$ are empty we say that φ is *closed*.

The model checking procedure presented in the paper is shown to work in the so-called alternation free fragment of the logic. The idea is that propositions introduced in least (greatest) fixpoints should not occur free in nested greatest (least) fixpoints. More precisely, call an occurrence of an abstract proposition X a μ-proposition or ν-proposition when it is bound by a μ or ν operator, respectively. Then the definition is as follows.

Definition 8 (Alternation free formula). *A formula of $\mathcal{L}_{hp}$ is called* alternation free *when no subformula includes both a free μ-proposition and a free ν-proposition.*

An example of formula with alternation is the following

$$[\![\mathsf{a}\, x]\!] \mu Y(x).(\nu Z(x).[\![x < \mathsf{a}\, y]\!] Y(y) \wedge [\![x < \mathsf{b}\, y]\!] Z(y))$$

In fact, Z is a ν-proposition, Y is a μ-proposition and they both occur free in the subformula $[\![x < \mathsf{a}\, y]\!] Y(y) \wedge [\![x < \mathsf{b}\, y]\!] Z(y)$. It expresses that along every run starting with an a and consisting of an infinite causal chain of events, labelled a or b, only a finite number of a-labelled events can occur. Already for the propositional mu-calculus, it can be proved that alternation increases the expressiveness of the logic and also its complexity (see, e.g., [27]). Still, as argued in the same paper, the alternation free fragment is quite useful, in practice, as many behavioural properties of interest can be expressed in such fragment.

3.2 Semantics

Since the logic $\mathcal{L}_{hp}$ is interpreted over PESs, the satisfaction of a formula is defined with respect to a configuration C, representing the state of the computation, and a (total) function $\eta : \mathit{Var} \to E$, called an *environment*, that binds free variables in φ to events in C. Namely, if $\mathit{Env}_{\mathcal{E}}$ denotes the set of environments, the semantics of a formula will be a set of pairs in $\mathcal{C}(\mathcal{E}) \times \mathit{Env}_{\mathcal{E}}$. The semantics of $\mathcal{L}_{hp}$ also depends on a proposition environment providing a semantic interpretation for propositions.

Definition 9 (Proposition environments). *Let $\mathcal{E}$ be a* PES. *A* proposition environment *is a function $\pi : \mathcal{X} \to 2^{\mathcal{C}(\mathcal{E}) \times \mathit{Env}_{\mathcal{E}}}$ such that if $(C, \eta) \in \pi(X(\mathbf{x}))$ and $\eta'(\mathbf{y}) = \eta(\mathbf{x})$ pointwise, then $(C, \eta') \in \pi(X(\mathbf{y}))$. We denote by $\mathit{PEnv}_{\mathcal{E}}$ the set of proposition environments, ranged over by π.*

The condition imposed on proposition environments ensures that the semantics of a formula only depends on the events that the environment associates to its free variables and that it does not depend on the naming of the variables.

We can now give the semantics of the logic $\mathcal{L}_{hp}$. Given an event environment η and an event e we write $\eta[x \mapsto e]$ to indicate the updated environment which maps x to e. Similarly, for a proposition environment π and $S \subseteq \mathcal{C}(\mathcal{E}) \times \mathit{Env}_{\mathcal{E}}$, we write $\pi[Z(\mathbf{x}) \mapsto S]$ for the corresponding update.

Definition 10 (Semantics). *Let $\mathcal{E}$ be a* PES. *The denotation of a formula φ in $\mathcal{L}_{hp}$ is given by the function $\{\!|\cdot|\!\}^{\mathcal{E}} : \mathcal{L}_{hp} \to \mathit{PEnv}_{\mathcal{E}} \to 2^{\mathcal{C}(\mathcal{E}) \times \mathit{Env}_{\mathcal{E}}}$ defined inductively as follows, where we write $\{\!|\varphi|\!\}^{\mathcal{E}}_{\pi}$ instead of $\{\!|\varphi|\!\}^{\mathcal{E}}(\pi)$:*

$$\{\!|\mathsf{T}|\!\}^{\mathcal{E}}_{\pi} = \mathcal{C}(\mathcal{E}) \times \mathit{Env}_{\mathcal{E}} \qquad \{\!|\mathsf{F}|\!\}^{\mathcal{E}}_{\pi} = \emptyset \qquad \{\!|Z(\mathbf{y})|\!\}^{\mathcal{E}}_{\pi} = \pi(Z(\mathbf{y}))$$

$$\{\!|\varphi_1 \wedge \varphi_2|\!\}^{\mathcal{E}}_{\pi} = \{\!|\varphi_1|\!\}^{\mathcal{E}}_{\pi} \cap \{\!|\varphi_2|\!\}^{\mathcal{E}}_{\pi} \qquad \{\!|\varphi_1 \vee \varphi_2|\!\}^{\mathcal{E}}_{\pi} = \{\!|\varphi_1|\!\}^{\mathcal{E}}_{\pi} \cup \{\!|\varphi_2|\!\}^{\mathcal{E}}_{\pi}$$

$$\{\!|\langle\!\langle \mathbf{x}, \overline{\mathbf{y}} < \mathsf{a}\, z \rangle\!\rangle \varphi|\!\}^{\mathcal{E}}_{\pi} = \{(C, \eta) \mid \exists e.\, C \xrightarrow{\eta(\mathbf{x}), \overline{\eta(\mathbf{y})} < e}_{\mathsf{a}} C' \ \wedge\ (C', \eta[z \mapsto e]) \in \{\!|\varphi|\!\}^{\mathcal{E}}_{\pi}\}$$

$$\{\!|[\![\mathbf{x}, \overline{\mathbf{y}} < \mathsf{a}\, z]\!] \varphi|\!\}^{\mathcal{E}}_{\pi} = \{(C, \eta) \mid \forall e.\, C \xrightarrow{\eta(\mathbf{x}), \overline{\eta(\mathbf{y})} < e}_{\mathsf{a}} C' \ \Rightarrow\ (C', \eta[z \mapsto e]) \in \{\!|\varphi|\!\}^{\mathcal{E}}_{\pi}\}$$

$$\{\!|\nu Z(\mathbf{x}).\varphi|\!\}^{\mathcal{E}}_{\pi} = \mathit{gfp}(f_{\varphi, Z(\mathbf{x}), \pi}) \qquad \{\!|\mu Z(\mathbf{x}).\varphi|\!\}^{\mathcal{E}}_{\pi} = \mathit{lfp}(f_{\varphi, Z(\mathbf{x}), \pi})$$

where $f_{\varphi, Z(\mathbf{x}), \pi} : 2^{\mathcal{C}(\mathcal{E}) \times \mathit{Env}_{\mathcal{E}}} \to 2^{\mathcal{C}(\mathcal{E}) \times \mathit{Env}_{\mathcal{E}}}$ is the semantic function *of φ, $Z(\mathbf{x})$, π defined by $f_{\varphi, Z(\mathbf{x}), \pi}(S) = \{\!|\varphi|\!\}^{\mathcal{E}}_{\pi[Z(\mathbf{x}) \mapsto S]}$ and $\mathit{gfp}(f_{\varphi, Z(\mathbf{x}), \pi})$ (resp. $\mathit{lfp}(f_{\varphi, Z(\mathbf{x}), \pi})$) denotes the corresponding greatest (resp. least) fixpoint. When $(C, \eta) \in \{\!|\varphi|\!\}^{\mathcal{E}}_{\pi}$ we say that the* PES *$\mathcal{E}$ satisfies the formula φ in the configuration C and environments η, π, and we write $C, \eta \models^{\mathcal{E}}_{\pi} \varphi$.*

The semantics of boolean operators is as usual. The formula $\langle\!\langle \mathbf{x}, \overline{\mathbf{y}} < \mathsf{a}\, z \rangle\!\rangle \varphi$ holds in (C, η) when from configuration C there is an enabled a-labelled event e that is causally dependent on (at least) the events bound to the variables in $\mathbf{x}$ and concurrent with (at least) those bound to the variables in $\mathbf{y}$ and can be executed producing a new configuration $C' = C \cup \{e\}$ which, paired with the environment $\eta' = \eta[z \mapsto e]$, satisfies the formula φ. Dually, $[\![\mathbf{x}, \overline{\mathbf{y}} < \mathsf{a}\, z]\!] \varphi$ holds

when all a-labelled events executable from C, caused by $\mathbf{x}$ and concurrent with $\mathbf{y}$ bring to a configuration where φ is satisfied.

The fixpoints corresponding to the formulae $\nu Z(\mathbf{x}).\varphi$ and $\mu Z(\mathbf{x}).\varphi$ are guaranteed to exist by Knaster-Tarski theorem, since the set $2^{\mathcal{C}(\mathcal{E})\times Env_{\mathcal{E}}}$ ordered by subset inclusion is a complete lattice and the functions, of which the fixpoints are calculated, are monotonic.

For example, the formula $\langle\!|\mathsf{a}\, x|\!\rangle\langle\!|\overline{x} < \mathsf{b}\, y|\!\rangle\mathsf{T}$ says that we can execute an a-labelled event and a b-labelled event, concurrently. It is satisfied by all the PES in Fig. 1. The formula $\langle\!|\mathsf{a}\, x|\!\rangle(\langle\!|\overline{x} < \mathsf{b}\, y|\!\rangle\mathsf{T} \wedge \langle\!|x < \mathsf{b}\, z|\!\rangle\mathsf{T})$ expresses a simple "history dependent" property: it requires that, after the execution of an a-labelled event, one can choose between a concurrent and a causally dependent b-labelled event. This holds for $\mathcal{E}_1$ in Fig. 1a, while it is false on $\mathcal{E}_2$ in Fig. 1b where the choice is already determined by the execution of the a-labelled event.

As an example of property of infinite computations consider the formula $[\![\mathsf{b}\, x]\!]\nu Z(x).\langle\!|\mathsf{a}\, z|\!\rangle\langle\!|\overline{z} < \mathsf{a}\, y|\!\rangle\mathsf{T}\wedge[\![x < \mathsf{b}\, y]\!]Z(y)$, expressing that all non-empty causal chain of b-labelled events reach a state where is possible to execute two concurrent a-labelled events. This holds for the PES $\mathcal{E}_3$ in Fig. 1c. Now consider the formula $\mu X.\langle\!|\_\, z|\!\rangle X \vee \langle\!|\mathsf{b}\, x|\!\rangle\langle\!|x < \mathsf{a}\, y|\!\rangle\nu Y.\langle\!|\_\, z|\!\rangle Y$, where we use $\langle\!|\_\, z|\!\rangle\varphi$ as a shortcut for $\bigvee_{\mathsf{c}\in\Lambda}\langle\!|\mathsf{c}\, z|\!\rangle\,\varphi$. The formula, requiring the existence of an infinite run containing a b-labelled event immediately followed by a causally dependent a-labelled event, turns out to be false in the same PES. Intuitively this is because any a-labelled event causally dependent on a b-labelled event is in conflict with the rest of the infinite chain of events, and then, after its execution, the computation is guaranteed to terminate. A variant of the formula $\mu X.\langle\!|\_\, z|\!\rangle X \vee \langle\!|\mathsf{b}\, x|\!\rangle\langle\!|\overline{x} < \mathsf{a}\, y|\!\rangle\nu Y.\langle\!|\_\, z|\!\rangle Y$ requiring the existence of an infinite run containing a b-labelled event immediately followed by a concurrent a-labelled event, would be again true in $\mathcal{E}_3$. Observe that all formulae in the examples above are alternation free.

4 A Local Model Checker for the Logic

The model checker is given as a tableau system for testing whether an alternation free formula of the logic $\mathcal{L}_{hp}$ is satisfied by a semantic model of a system given in the form of a PES.

4.1 Constants and Definition Lists

As a first step, along the lines of [22], we extend the logic with propositional constants which are useful in defining the tableau rules.

Let $\mathcal{K}$ be a set of propositional constant symbols, ranged over by upper case letters like $U, V, W \ldots$. Similarly to abstract propositions, constants are meant to represent formulae of the logic, possibly containing (unnamed) free variables. Each constant U has an arity $ar(U)$, which indicates the number of free event variables in the associated formula. It can be used as a formula by specifying the names for its free variables, writing $U(\mathbf{x})$ where $|\mathbf{x}| = ar(U)$.

Definition 11 (Extended logic). *We denote by $\mathcal{L}^e_{hp}$ the logic defined as in Definition 7, with the addition of constants $U(\mathbf{x})$ as atomic formulae.*

Constants are interpreted at syntactical level, by means of a list of declarations that associates constants with formulae of the logic.

Definition 12 (Definition list). *A* definition list *is a sequence Δ of declarations $U_1(\mathbf{x}_1) = \varphi_1, \ldots, U_n(\mathbf{x}_n) = \varphi_n$, where $U_i \neq U_j$ whenever $i \neq j$ and each φ_k is a formula of the extended logic $\mathcal{L}^e_{hp}$ such that $|\mathbf{x}_k| = ar(U_k)$ and $fv(\varphi_k) = \mathbf{x}_k$, for $k \in \{1, \ldots, n\}$. We write $dom(\Delta) = \{U_1, \ldots, U_n\}$ for the set of constants declared in Δ, and $K(\varphi)$ for the set of constants appearing in φ. The definition list is* well-formed *if for all $k \in \{1, \ldots, n\}$ it holds that $K(\varphi_k) \subseteq \{U_1, \ldots, U_{k-1}\}$.*

Clearly, a prefix of a well-formed definition list is itself a well-formed definition list. Hereafter all definition lists will be implicitly assumed to be well-formed.

Definition 13 (Admissibility). *Let φ be a formula of the extended logic $\mathcal{L}^e_{hp}$. We say that a definition list Δ is* admissible *for φ if $K(\varphi) \subseteq dom(\Delta)$.*

A definition list is a sort of syntactical environment for constants, but, differently from environments, it is not total. The admissibility of Δ for φ simply means that each constant occurring in φ is declared in Δ, possibly with different names for its free variables. Given a constant $U \notin dom(\Delta)$ such that Δ is admissible for φ, $|fv(\varphi)| = ar(U)$ and $fv(\varphi) = \mathbf{x}$, we denote by $\Delta \cdot (U(\mathbf{x}) = \varphi)$ the definition list resulting by appending the declaration $U(\mathbf{x}) = \varphi$ to Δ.

Given a formula φ in the extended logic $\mathcal{L}^e_{hp}$ and an admissible definition list Δ, we can transform φ into a formula of the original logic $\mathcal{L}_{hp}$ by repeatedly replacing each constant with the corresponding definition. The substitution of a constant U according to its definition $U(\mathbf{x}) = \psi$ in a formula φ, denoted $\varphi[U := \psi]$, is the formula obtained from φ by replacing any occurrence of $U(\mathbf{y})$ with $\psi[\mathbf{y}/\mathbf{x}]$.

Definition 14 (Expansion). *Let φ be a formula in the extended logic $\mathcal{L}^e_{hp}$ and let Δ be $U_1(\mathbf{x}_1) = \psi_1, \ldots, U_n(\mathbf{x}_n) = \psi_n$, an admissible definition list for φ. The formula φ_Δ in $\mathcal{L}_{hp}$ is obtained by applying recursively n substitutions, starting from $\varphi_{(n)} = \varphi[U_n := \psi_n]$ and then $\varphi_{(i-1)} = \varphi_{(i)}[U_{i-1} := \psi_{i-1}]$, until $\varphi_{(1)}$ is calculated. Then $\varphi_\Delta = \varphi_{(1)}$.*

We will often write $\Delta(U(\mathbf{y}))$ to indicate the formula of the extended logic associated with $U(\mathbf{y})$ in Δ. Free variables and free propositions of a formula φ in the extended logic, in an admissible definition list Δ, are defined as before, with the addition of the clauses $fv(U(\mathbf{y})) = \mathbf{y}$ and $fp(U(\mathbf{y})) = fp(\Delta(U(\mathbf{y})))$.

Concerning the semantics, we say that a PES $\mathcal{E}$ satisfies the formula φ with the admissible definition list Δ in the configuration C and environments η, π, written $C, \eta, \Delta \models^{\mathcal{E}}_{\pi} \varphi$, when $(C, \eta) \in \{\!|\varphi_\Delta|\!\}^{\mathcal{E}}_{\pi}$.

4.2 Tableau Rules

The tableau system works on judgements $\Gamma \models^{\mathcal{E}} \varphi$, where $\Gamma = \langle C, \eta, \Delta \rangle$, referred to as a *context*, is a tuple consisting of a configuration $C \in \mathcal{C}(\mathcal{E})$, an environment η and a definition list Δ, admissible for φ, such that φ_Δ is closed. It consists in a set of rules of the form

$$\frac{C, \eta, \Delta \models^{\mathcal{E}} \varphi}{C_1, \eta_1, \Delta_1 \models^{\mathcal{E}} \varphi_1 \ \ldots \ C_k, \eta_k, \Delta_k \models^{\mathcal{E}} \varphi_k} \ \delta$$

where $k > 0$ and δ is a possible side condition required to hold. The intuition is that the validity of the premise sequent reduces to the validity of its consequents. The index $\mathcal{E}$, when the model $\mathcal{E}$ is clear from the context, will be dropped.

In [22,28], the finiteness of the model is an essential ingredient that concurs to the finiteness of the tableaux. In our case, as already mentioned, the PES model is commonly infinite, even for finite-state systems. However, working on strongly regular PESs we have that (1) only a finite part of the model is used in every step of the tableau construction, thanks to the fact that strongly regular PESs are boundedly branching and (2) after going sufficiently in depth, the PES starts "repeating" cyclically the same structure. These facts will allow to conclude that the satisfaction of a formula can be determined by checking only a finite part of the PES (as formally proved later in Sect. 4.4).

The tableau rules for the logic $\mathcal{L}^{e}_{hp}$, are reported in Table 1. The rules for the propositional connectives are straightforward. They reduce the satisfaction of the formula in the premise to the satisfaction of subformulae in an obvious way. For instance $\varphi \vee \psi$ is satisfied when either φ is satisfied or ψ is satisfied. The context is not altered.

Similarly, the rules for the modal operators generate sequents involving the subformulae after the modal operators with a context that changes according to the semantics. The formula $\langle\!\langle \mathbf{x}, \overline{\mathbf{y}} < \mathsf{a}\, z \rangle\!\rangle \varphi$ holds when there is at least a transition leading to a context where φ is satisfied, while $[\![\mathbf{x}, \overline{\mathbf{y}} < \mathsf{a}\, z]\!] \varphi$ is satisfied when all transitions lead to a context where φ is satisfied.

For each kind of fixpoint formula $\alpha Z(\mathbf{x}).\varphi$, with $\alpha \in \{\mu, \nu\}$, there are two rules. The first rule introduces a constant, which is added to the definition list and used as a formula in the consequent. The second rule unfolds the fixpoint by unrolling the associated constant and also reassigns its variables by updating the environment. The side condition involves an unspecified part γ, the so-called *stop* condition, that prevents the reduction to continue unboundedly and will be described in the next section.

The tableau rules are backwards sound, namely the premise is true if all the consequents are true, a property that will play a basic role in Sect. 4.4.

Lemma 1 (Backwards soundness). *Every rule of the tableau system is backwards sound.*

4.3 The Stop Condition

In order to ensure the finiteness of the tableaux generated by a formula, the unfolding of the fixpoints has to be stopped when a context is reached that is

Table 1. The tableau rules for logic $\mathcal{L}^{e}_{hp}$.

$$\frac{C,\eta,\Delta \models \varphi \wedge \psi}{C,\eta,\Delta \models \varphi \quad C,\eta,\Delta \models \psi}$$

$$\frac{C,\eta,\Delta \models \varphi \vee \psi}{C,\eta,\Delta \models \varphi} \qquad \frac{C,\eta,\Delta \models \varphi \vee \psi}{C,\eta,\Delta \models \psi}$$

$$\frac{C,\eta,\Delta \models \langle\!\langle \mathbf{x},\overline{\mathbf{y}} < \mathsf{a}\, z \rangle\!\rangle \varphi}{C',\eta[z \mapsto e],\Delta \models \varphi} \quad \exists\, e.\ C \xrightarrow{\eta(\mathbf{x}),\overline{\eta(\mathbf{y})} < e}_{\mathsf{a}} C'$$

$$\frac{C,\eta,\Delta \models [\![\mathbf{x},\overline{\mathbf{y}} < \mathsf{a}\, z]\!] \varphi}{C_1,\eta_1,\Delta \models \varphi \ \ldots\ C_n,\eta_n,\Delta \models \varphi}$$

where $\{(C_1,\eta_1),\ldots,(C_n,\eta_n)\} = \{(C',\eta[z \mapsto e]) \mid \exists\, e.\ C \xrightarrow{\eta(\mathbf{x}),\overline{\eta(\mathbf{y})} < e}_{\mathsf{a}} C'\}$

$$\frac{C,\eta,\Delta \models \alpha Z(\mathbf{x}).\varphi}{C,\eta,\Delta' \models U(\mathbf{x})} \quad \Delta' \text{ is } \Delta \cdot (U(\mathbf{x}) = \alpha Z(\mathbf{x}).\varphi) \text{ for } \alpha \in \{\nu,\mu\}$$

$$\frac{C,\eta,\Delta \models U(\mathbf{y})}{C,\eta',\Delta \models \varphi[Z \coloneqq U(\mathbf{y})]} \quad \neg\gamma \text{ and } \Delta(U(\mathbf{x})) = \alpha Z(\mathbf{x}).\varphi \text{ for } \alpha \in \{\nu,\mu\} \text{ and } \eta' = \eta[\mathbf{x} \mapsto \eta(\mathbf{y})],$$

equivalent, in a suitable sense to be defined, to a context occurring in an ancestor of the current node of the tableau, for the same fixpoint formula. The notion of equivalence should prevent the tableau generation to continue infinitely but it should be chosen carefully not to break the soundness of the technique.

Surely two contexts $\Gamma = \langle C,\eta,\Delta\rangle$ and $\Gamma' = \langle C',\eta',\Delta'\rangle$ for a formula φ, in order to be considered equivalent, must have isomorphic futures, i.e., the residuals $\mathcal{E}[C]$ and $\mathcal{E}[C']$ are isomorphic as PESs. This is not sufficient, though, since φ can express history dependent properties that relates the future with the past events. Therefore we additionally ask that event variables of φ are mapped to events in C and C', respectively, which have the same relations (causality and concurrency) with the corresponding futures.

In order to formalise this intuition we need to set up some notions.

Definition 15 (Contextualized residual). *Let $\mathcal{E}$ be a* PES*. Given a configuration $C \in \mathcal{C}(\mathcal{E})$, an environment η, a finite set of variables $\mathbf{x} \subseteq Var$, we refer to $\mathcal{E}[C,\eta,\mathbf{x}] = \langle \mathcal{E}[C],\eta,\mathbf{x}\rangle$ as the $(\eta,\ \mathbf{x})$-*contextualised residual *of $\mathcal{E}$ after C.*

The contextualised residuals $\mathcal{E}[C,\eta,\mathbf{x}]$ and $\mathcal{E}[C',\eta',\mathbf{x}']$ are isomorphic when $\mathbf{x} = \mathbf{x}'$ and there is an isomorphism between $\mathcal{E}[C]$ and $\mathcal{E}[C']$ "compatible" with the way environments map the variables in $\mathbf{x}$ into events.

Definition 16 (Isomorphism of contextualized residuals). *Let $\mathcal{E}$ be a* PES. *Two contextualised residuals $\mathcal{E}[C_1, \eta_1, \mathbf{x}_1]$ and $\mathcal{E}[C_2, \eta_2, \mathbf{x}_2]$ are isomorphic, written $\mathcal{E}[C_1, \eta_1, \varphi] \sim \mathcal{E}[C_2, \eta_2, \varphi]$, when $\mathbf{x}_1 = \mathbf{x}_2$ and there is an isomorphism $\iota : \mathcal{E}[C_1] \to \mathcal{E}[C_2]$ such that for any $x \in \mathbf{x}_1$, $e_1 \in \mathcal{E}[C_1]$ it holds $\eta_1(x) \leq_1 e_1 \iff \eta_2(x) \leq_2 \iota(e_1)$.*

A key observation is that isomorphic contextualised residuals satisfy exactly the same formulae, in the sense of the following theorem.

Lemma 2 (Equivalent contexts, logically). *Let $\mathcal{E}$ be a* PES, *let φ be a formula and let $\Gamma_1 = \langle C_1, \eta_1, \Delta_1 \rangle$ and $\Gamma_2 = \langle C_2, \eta_2, \Delta_2 \rangle$ be contexts, where $C_1, C_2 \in \mathcal{C}(\mathcal{E})$, $\Delta_1 \subseteq \Delta_2$ and Δ_1 (hence Δ_2) admissible for φ. For all proposition environments π_1, π_2 such that $\forall\ Z \in fp(\varphi_{\Delta_1})$, $(C_1, \eta_1) \in \pi_1(Z(\mathbf{y})) \iff (C_2, \eta_2) \in \pi_2(Z(\mathbf{y}))$, if $\mathcal{E}[C_1, \eta_1, fv(\varphi)] \sim \mathcal{E}[C_2, \eta_2, fv(\varphi)]$ then $\Gamma_1 \models^{\mathcal{E}}_{\pi_1} \varphi \iff \Gamma_2 \models^{\mathcal{E}}_{\pi_2} \varphi$.*

Note that the condition on the proposition environments π_1, π_2 is vacuously satisfied for formulae in the sequents of a tableau. In fact, the sequent labelling the root contains a closed formula and thus, by construction, formulae do not contain free propositions and neither do those associated to constants.

The results above motivate the definition of the stop condition.

Definition 17 (Stop condition). *The* stop condition γ *for a rule where the premise is $C, \eta, \Delta \models U(\mathbf{y})$, is as follows:*

$$\textit{There is an ancestor of the node labelled with } C', \eta', \Delta' \models U(\mathbf{z}), \textit{ such that } \mathcal{E}[C', \eta'[\mathbf{y} \mapsto \eta'(\mathbf{z})], \mathbf{y}] \sim \mathcal{E}[C, \eta, \mathbf{y}].$$

Informally, the stop condition holds when in a previous step of the construction of the tableau the same constant has been unfolded in a context equivalent to the current one, possibly after some renaming of variables. Hence we can safely avoid to continue along this path. Instead, when the stop condition fails, it makes sense to further unroll the fixpoint since the current context is still "different enough" from those previously encountered.

4.4 Model Checking a Formula Through Tableaux

For checking whether a closed formula φ in $\mathcal{L}_{hp}$ is satisfied by a PES $\mathcal{E}$, we proceed by building a tableau for the sequent $\emptyset, \eta, \emptyset \models^{\mathcal{E}} \varphi$, where η is any environment (irrelevant since the formula does not have free variables). A maximal tableau is a proof tree where all leaves are labelled by sequents to which no rule applies.

We next clarify when a maximal tableau is considered successful.

Definition 18 (Successful tableau). *A* successful *tableau is a finite maximal tableau where every leaf is labelled by a sequent $C, \eta, \Delta \models^{\mathcal{E}} \varphi$ such that:*

1. $\varphi = \mathsf{T}$; *or*
2. $\varphi = [\![\mathbf{x}, \overline{\mathbf{y}} < \mathsf{a}\, z]\!]\, \psi$; *or*
3. $\varphi = U(\mathbf{y})$ *and* $\Delta(U(\mathbf{x})) = \nu Z(\mathbf{x}).\psi$.

We will prove that in a successful tableau all leaves are labelled by true sequents, a fact that, together with backwards soundness of the rules (Lemma 1), will guarantee the truth of the sequent labelling the root.

Note that the choice of the rule to be applied at a step of the construction of a tableau is non-deterministic in the case of $\langle\!|\mathbf{x}, \overline{\mathbf{y}} < \mathsf{a}\, z|\!\rangle\, \varphi$ and $\varphi \vee \psi$. This means that there can be various maximal tableaux for the same sequent. However, when we work on strongly regular PESs, the fact that they are boundedly branching ensures that at each step the number of possible non-deterministic choices is finite and bounded. In turn this is used to deduce that there can be only a finite number of maximal tableaux for a sequent, up to renaming of constants.

We next focus the finiteness issue and then move on to the soundness and completeness of the technique.

Finiteness. We first aim at proving that all tableaux for a sequent $\emptyset, \eta, \emptyset \models^{\mathcal{E}} \varphi$ are finite. A first basic observation is that an infinite tableau for a sequent $C, \eta, \Delta \models^{\mathcal{E}} \varphi$ necessarily includes a path where the same constant is unfolded infinitely many times.

Lemma 3 (Infinite repetitions of constants in infinite tableaux). *Given an infinite tableau for a sequent $C, \eta, \Delta \models^{\mathcal{E}} \varphi$ there exists a constant U and an infinite path in the tableau such that U occurs in an unfolding rule infinitely many often, possibly each time with different event variables $\mathbf{x}_i$.*

Along the lines of [22], the proof relies on the introduction of a notion of degree for a sequent, that intuitively estimates the length of the longest path in the tableau that starts from the sequent itself and ends at the first sequent whose formula is a constant previously introduced (not new) or at a leaf. We already observed that the fact that strongly regular PESs are boundedly branching implies that also the constructed tableaux are finitely branching. Then, by König lemma, an infinite tableau necessarily include an infinite path. The observation that the degree is finite, non-negative and it decreases along a path until a previously introduced constant is met again, allows one to conclude.

A crucial observation is now that, for strongly regular PESs, the number of contextualised residuals is finite up to isomorphism. Using this fact, if there were an infinite path in a tableau and thus, by Lemma 3, a constant occurring infinitely often in such a path, then the constant would occur infinitely often within isomorphic contextualised residuals, leading to a contradiction. In fact, at the first repetition the stop condition (see Definition 17) would have prescribed of terminating the branch.

Theorem 1 (Finite number of contextualised residuals). *Let $\mathcal{E}$ be a strongly regular* PES *and let $\mathbf{x} \subseteq Var$ be a finite set of variables. Then the class of $(\eta, \mathbf{x})$-contextualised residuals of $\mathcal{E}$ after C, i.e., $\{\mathcal{E}[C, \eta, \mathbf{x}] \mid \eta \in Env_{\mathcal{E}} \wedge C \in \mathcal{C}(\mathcal{E})\}$ is finite up to isomorphism.*

We can finally deduce the finiteness of the tableaux for a sequent that in turn implies that the number of tableaux is finite (up to constant renaming). This fact is essential for termination of the model checking procedure.

Theorem 2 (Tableaux finiteness). *Given a strongly regular* PES $\mathcal{E}$ *and a formula* φ*, every tableau for a sequent* $\Gamma \models^{\mathcal{E}} \varphi$ *is finite. Hence the number of tableaux for* $\Gamma \models^{\mathcal{E}} \varphi$ *is finite up to constant renaming.*

Soundness and Completeness. We finally prove the soundness and completeness of the tableau system. For this we use the possibility of reducing the satisfaction of a formula to the satisfaction of a finite approximant. While on finite models this is immediate, here we need the (co)continuity of the semantic functions associated to formulae (see Definition 10) ensuring that the fixpoints will be reached in at most ω steps.

Let X be a set. A subset $A \subseteq 2^X$ is called *directed* if for any $S_1, S_2 \in A$ there exists $S \in A$ such that $S_1, S_2 \subseteq S$. A function $f : 2^X \to 2^X$ is *continuous* when for any directed set $A \subseteq 2^X$ it holds that $f(\bigcup A) = \bigcup\{f(S) \mid S \in A\}$. We call it *co-continuous* when it is continuous in the reverse (superset) order.

The semantic functions associated with formulae of the logic $\mathcal{L}_{hp}$ (see Definition 10) are neither continuous nor co-continuous in general. However, when requiring alternation freeness, μ-formulae, i.e., formulae where all ν-operators are in the scope of a μ-operator, are continuous and ν-formulae, defined dually, are co-continuous.

Lemma 4 ((Co-)continuous semantic operators). *Let* $\mathcal{E}$ *be a* PES*. Given an alternation free* μ*-formula* ψ*, a proposition* $Z(\mathbf{z}) \in \mathcal{X}$ *and a proposition environment* $\pi \in PEnv_{\mathcal{E}}$*, the semantic function* $f_{\psi, Z(\mathbf{z}), \pi}$ *is continuous. Dually, if* ψ *is an alternation free* ν*-formula the function* $f_{\psi, Z(\mathbf{z}), \pi}$ *is co-continuous.*

Note that the continuous fragment here is wider than that in [25] as, in particular, it includes the box modality. The difference is motivated by the fact that strongly regular PESs are finitely branching.

By Kleene Theorem the least fixpoint of a continuous function on a lattice requires up to ω iterations to be reached, namely if $f : 2^X \to 2^X$ is continuous then $lfp(f) = \bigcup_{i \in \mathbb{N}} f^i(\emptyset)$. Similarly, if f is co-continuous $gfp(f) = \bigcap_{i \in \mathbb{N}} f^i(X)$. This fact plays a role in the proof of the main result below.

Theorem 3 (Soundness and completeness of the tableaux system). *Given a strongly regular* PES $\mathcal{E}$ *and a closed alternation free formula* φ *of* $\mathcal{L}_{hp}$*, a sequent* $C, \eta, \emptyset \models^{\mathcal{E}} \varphi$ *has a successful tableau if and only if* $(C, \eta) \in \{\!|\varphi|\!\}^{\mathcal{E}}$*.*

5 Conclusions

We provided a tableau system for model checking the alternation free fragment of a logic for true concurrency $\mathcal{L}_{hp}$ over strongly regular PESs, proving finiteness of the tableaux and soundness and completeness of the rule system. Such results, together with Theorem 2, that ensures that a sequent has a(n essentially) finite number of tableaux, leads to a decision procedure for the alternation free fragment of $\mathcal{L}_{hp}$ over strongly regular event structures. A concrete procedure

requires the effectiveness of the transition relation over configurations and of the equivalence of contextualised residuals, that we have if we focus on regular trace PESs. Indeed, a natural instantiation of the model checking procedure can be given on finite safe Petri nets. For space reasons its presentation is delayed to the full version of the paper.

While regular trace event structures can be shown to be strongly regular, we still do not know whether also the converse holds. Some preliminary investigations lead us to conjecture that this is the case.

Soundness and completeness of the tableau system rely on the (co)continuity of the semantic functions associated with the formulae that we obtained by considering the alternation free fragment of the logic. While continuity fails outside this fragment, we conjecture that, exploiting the regularity property of the PES models, a sort of continuity up to isomorphism can be proved, allowing to extend the procedure to the full logic $\mathcal{L}_{hp}$.

Another open issue concerns the possibility of generalising the tableau-based technique to the full logic $\mathcal{L}$ in [10]. This is quite challenging: the full logic $\mathcal{L}$ induces a behavioural equivalence – hhp-bisimilarity – which is known to be undecidable already for finite state Petri nets [29]. Note that this does not imply undecidability of the corresponding model checking problem. On the semantic side one could try to relax the restriction to strongly regular PESs. However, we tend to believe that few results can be obtained when the realm of regular structures is abandoned.

Model checking on event structures has been considered by several other authors. In [30] a finite representation of the PES corresponding to the behaviour of a suitable class of programs is proposed, showing how discrete event logics can be model-checked on such structures. The paper [15] shows that first order logic and a restricted form of monadic second order (MSO) logic are decidable on regular trace event structures. The fact that the mu-calculus, in the propositional case, corresponds to the bisimulation invariant fragment of MSO logic [31] suggests the possibility of exploiting the mentioned result in our setting. This would require an encoding of $\mathcal{L}_{hp}$ into the MSO logic of [15]. More generally, understanding which are the bisimulation invariant fragments of MSO over event structures, with respect to the various concurrent bisimulations, represents an interesting program in itself. The work in [14] develops higher-order games as a mean for local model-checking a concurrent logic over partial order semantics. Despite the fact that such logic is different (and of incomparable expressive power with ours as explained in [10]), exploring the possibility of adopting a game-theoretical approach in our setting appears an interesting venue of further research.

Finally, a lot of work exists in the literature for the propositional mu-calculus, proposing efficient automata-based model checking techniques [32,33], that reduce the model checking problem to the non-emptiness problem of parity tree automata. Trying to develop a similar approach for the logic $\mathcal{L}_{hp}$, with the idea that the automata would somehow inherit the regularities in the structure of the PESs, represents a stimulating direction of future research.

References

1. Esparza, J., Heljanko, K.: Unfoldings - A Partial order Approach to Model Checking. EACTS Monographs. Springer, Heidelberg (2008)
2. van Glabbeek, R., Goltz, U.: Refinement of actions and equivalence notions for concurrent systems. Acta Inform. **37**(4/5), 229–327 (2001)
3. Nicola, R., Ferrari, G.L.: Observational logics and concurrency models. In: Nori, K.V., Veni Madhavan, C.E. (eds.) FSTTCS 1990. LNCS, vol. 472, pp. 301–315. Springer, Heidelberg (1990). doi:10.1007/3-540-53487-3_53
4. 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)
5. Pinchinat, S., Laroussinie, F., Schnoebelen, P.: Logical characterization of truly concurrent bisimulation. Technical report 114, LIFIA-IMAG, Grenoble (1994)
6. Penczek, W.: Branching Time and Partial Order in Temporal Logics. Time and Logic: A Computational Approach, pp. 179–228. UCL Press, London (1995)
7. Nielsen, M., Clausen, C.: Games and logics for a noninterleaving bisimulation. Nordic J. Comput. **2**(2), 221–249 (1995)
8. Bradfield, J., Fröschle, S.: Independence-friendly modal logic and true concurrency. Nordic J. Comput. **9**(1), 102–117 (2002)
9. Phillips, I., Ulidowski, I.: Event identifier logic. Math. Struct. Comput. Sci. **24**(2), 1–51 (2014)
10. Baldan, P., Crafa, S.: A logic for true concurrency. J. ACM **61**(4), 24:1–24:36 (2014)
11. Alur, R., Peled, D., Penczek, W.: Model-checking of causality properties. In: Proceedings of LICS 1995, pp. 90–100. IEEE Computer Society (1995)
12. Gutierrez, J., Bradfield, J.: Model-checking games for fixpoint logics with partial order models. In: Bravetti, M., Zavattaro, G. (eds.) CONCUR 2009. LNCS, vol. 5710, pp. 354–368. Springer, Heidelberg (2009). doi:10.1007/978-3-642-04081-8_24
13. Gutierrez, J.: Logics and bisimulation games for concurrency, causality and conflict. In: Alfaro, L. (ed.) FoSSaCS 2009. LNCS, vol. 5504, pp. 48–62. Springer, Heidelberg (2009). doi:10.1007/978-3-642-00596-1_5
14. Gutierrez, J.: On bisimulation and model-checking for concurrent systems with partial order semantics. Ph.D. thesis, University of Edinburgh (2011)
15. Madhusudan, P.: Model-checking trace event structures. In: Proceedings of LICS 2003, pp. 371–380. IEEE Computer Society (2003)
16. Winskel, G.: Event structures. In: Brauer, W., Reisig, W., Rozenberg, G. (eds.) ACPN 1986. LNCS, vol. 255, pp. 325–392. Springer, Heidelberg (1987). doi:10.1007/3-540-17906-2_31
17. Best, E., Devillers, R., Kiehn, A., Pomello, L.: Fully concurrent bisimulation. Acta Inform. **28**, 231–261 (1991)
18. Rabinovich, A.M., Trakhtenbrot, B.A.: Behaviour structures and nets. Fundam. Inform. **11**, 357–404 (1988)
19. Degano, P., Nicola, R., Montanari, U.: Partial orderings descriptions and observations of nondeterministic concurrent processes. In: Bakker, J.W., Roever, W.-P., Rozenberg, G. (eds.) REX 1988. LNCS, vol. 354, pp. 438–466. Springer, Heidelberg (1989). doi:10.1007/BFb0013030
20. Vogler, W.: Deciding history preserving bisimilarity. In: Albert, J.L., Monien, B., Artalejo, M.R. (eds.) ICALP 1991. LNCS, vol. 510, pp. 495–505. Springer, Heidelberg (1991). doi:10.1007/3-540-54233-7_158

21. 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). doi:10.1007/11419822_1
22. Stirling, C., Walker, D.: Local model checking in the modal mu-calculus. Theor. Comput. Sci. **89**(1), 161–177 (1991)
23. 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 - Essays Dedicated to Grzegorz Rozenberg. LNCS, vol. 2300, pp. 244–253. Springer, Heidelberg (2002). doi:10.1007/3-540-45711-9_14
24. Cleaveland, R., Steffen, B.: A linear-time model-checking algorithm for the alternation-free modal mu-calculus. Form. Methods Syst. Des. **2**(2), 121–147 (1993)
25. Fontaine, G.: Continuous fragment of the mu-calculus. In: Kaminski, M., Martini, S. (eds.) CSL 2008. LNCS, vol. 5213, pp. 139–153. Springer, Heidelberg (2008). doi:10.1007/978-3-540-87531-4_12
26. Carreiro, F., Facchini, A., Venema, Y., Zanasi, F.: Weak MSO: automata and expressiveness modulo bisimilarity. In: Henzinger, T.A., Miller, D. (eds.) Proceedings of CSL-LICS 2014, pp. 27:1–27:27. ACM Press (2014)
27. Bradfield, J., Stirling, C.: Modal mu-calculi. In: Blackburn, P., van Benthem, J., Wolter, F. (eds.) Handbook of Modal Logic, pp. 721–756. Elsevier, New York (2006)
28. Clarke, E.M., Schlingloff, B.H.: Model checking. In: Robinson, A., Voronkov, A. (eds.) Handbook of Automated Reasoning. Elsevier, Amsterdam (2001)
29. Jurdzinski, M., Nielsen, M., Srba, J.: Undecidability of domino games and hhp-bisimilarity. Inf. Comput. **184**(2), 343–368 (2003)
30. Penczek, W.: Model-checking for a subclass of event structures. In: Brinksma, E. (ed.) TACAS 1997. LNCS, vol. 1217, pp. 145–164. Springer, Heidelberg (1997). doi:10.1007/BFb0035386
31. Janin, D., Walukiewicz, I.: On the expressive completeness of the propositional mu-calculus with respect to monadic second order logic. In: Montanari, U., Sassone, V. (eds.) CONCUR 1996. LNCS, vol. 1119, pp. 263–277. Springer, Heidelberg (1996). doi:10.1007/3-540-61604-7_60
32. Emerson, E.A., Jutla, C.S., Sistla, A.P.: On model checking for the μ-calculus and its fragments. Theor. Comput. Sci. **258**(1–2), 491–522 (2001)
33. Streett, R., Emerson, E.A.: An automata theoretic decision procedure for the propositional mu-calculus. Inf. Comput. **81**(3), 249–264 (1989)