Relating some Logics for True Concurrency

Tommaso Padoan

Dipartimento di Matematica, Università di Padova, Padua, Italy
padoan@math.unipd.it

Abstract. We study some logics for true concurrency recently defined by several authors to characterise a number of known or meaningful behavioural equivalences, with special interest in history-preserving bisimilarity. All the considered logics are event-based, naturally interpreted over event structures or any formalism which can be given a causal semantics, like Petri nets. Operators of incomparable expressiveness from different logics can be combined into a single logic, more powerful than the original ones. Since the event structure associated with a system is typically infinite (even if the system is finite state), already the known decidability results of model-checking in the original logics are non-trivial. Here we show, using a tableaux-based approach, that the model-checking problem for the new logic is still decidable over a class of event structures satisfying a suitable regularity condition, referred to as strong regularity.

1 Introduction

In the analysis and verification of concurrent and distributed systems partial order semantics can be an appropriate choice since they provide a precise account of the possible steps in the evolution of the system and of their dependencies, like causality and concurrency. This approach is normally referred to as true concurrency and it is opposed to the so-called interleaving approach, where concurrency of actions is reduced to the non-deterministic choice among their possible sequentializations. In the true concurrent world, a widely used foundational model is given by Winskel's event structures [1]. They describe the behaviour of a system in terms of events in computations and two dependency relations: a partial order modelling causality and an additional relation modelling conflict.

Several true concurrent behavioural equivalences have been defined which allow to abstract operational models taking 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., [3,4,5,6,7,8]) and some verification techniques have been considered (see, e.g., [9,10,11]). Recently, event-based logics have been introduced [12,13], capable of uniformly characterising a relevant part of the true concurrent spectrum. Some of these logics, together with those in [10,11], are among the most expressive true concurrent logics for which verification techniques have been proved to be decidable over suitable classes of true concurrent models.

Interestingly, these logics provides the logical characterisation of a number of different meaningful behavioural equivalences, some of them incomparable.

The logic referred to as $\mathcal{L}_{hp}$, corresponding to a classical equivalence in the spectrum, i.e., history-preserving (hp-)bisimilarity [14,15,16], is a fragment of the more general logic in [12], representing instead hereditary history-preserving (hhp-)bisimilarity [4]. $\mathcal{L}_{hp}$ allows to predicate over executability of events in computations and their dependency relations (causality and concurrency). Formulae include variables which can be bound to events in computations. The logic includes two modalities, diamond and box, which allows to explicitly assert the dependency relations between the computational steps. The formula $\langle\!|x, \overline{y} < \mathsf{a}\, z|\!\rangle\varphi$ declares that is possible to execute an a-labelled event, which causally depends on the event bound to the variable x and is concurrent with the event bound to y, and, binding such an event to z, the formula φ holds. In general, x and y can be replaced by (possibly empty) tuples of variables. The presence of least and greatest fixpoint operators, in mu-calculus style, allows one to express properties of infinite computations. Recent results [17,18] proved that the model-checking problem for $\mathcal{L}_{hp}$ is decidable over a class of event structures satisfying a suitable regularity condition [19] referred to as strong regularity.

In [10] two true concurrent logics have been introduced, namely separation fixpoint logic (SFL) and trace fixpoint logic ($\mathbb{L}_\mu$). Both are based on a core logic with modalities that allow to express properties about causality and concurrency. The difference w.r.t. $\mathcal{L}_{hp}$ is that such modalities can only express causality and concurrency between consecutive steps. For example, the formula $\langle\mathsf{a}\rangle_{nc}\,\varphi$ declares the possibility to execute an a-labelled event, which is concurrent with (not caused by) the one executed before. The logics SFL and $\mathbb{L}_\mu$ differ for the way in which they capture the duality between concurrency and conflict, relying on operators on conflict-free sets of events. SFL uses a separating operator $*$ (dual $\bowtie$) that behaves as a structural conjunction, allowing for local reasoning on conflict-free sets of executable events. The formula $\varphi * \psi$ requires the existence of two concurrent disjunct subsets of the executable events, such that the subformula φ (resp. ψ) holds on the first (resp. second) subset. $\mathbb{L}_\mu$ instead has a second-order modality $\langle\otimes\rangle$ (dual $[\otimes]$) that recognises maximal concurrent subsets of the executable events. The formula $\langle\otimes\rangle\varphi$ requires that the subformula φ holds when restricting, locally, the computation of the system to a set of events that can actually execute all concurrently. The two logics have incomparable expressive power, as they characterise incomparable behavioural equivalences [10], in turn incomparable with hp-bisimilarity. Also in this case, model-checking has been proved decidable for both logics [10] over regular trace event structures [19].

In [13] the authors propose an extension of Hennessy-Milner logic called event identifier logic (EIL). The logic is again an event-based modal logic, but this time reverse as well as forward modalities are allowed. The two modalities, i.e., $\langle x : \mathsf{a}\rangle\rangle$ (forward) and $\langle\langle x\rangle$ (backward), are not capable to assert explicitly the dependency relations between the computational steps, which are instead captured by a proper sequencing of such operators. The meaning of the two operator is quite clear: $\langle x : \mathsf{a}\rangle\rangle\varphi$ declares the executability of an a-labelled event,

which is bound to x, and then φ holds; $\langle\langle x\rangle\varphi$ requires that the event bound to x can be undone and then φ holds. There is also a third operator $(x : \mathsf{a})\varphi$ which states that there is an a-labelled event executed in the past, such that, binding it to x, φ holds. The expressiveness of such logic is sufficient to provide a logical characterisation of hhp-bisimilarity, intuitively because the possibility of performing backward steps can be a mean of exploring alternative different futures. Moreover, a fragment of the logic, referred to as $\mathrm{EIL_h}$, where forward modalities are no longer allowed after backward modalities, corresponds to hp-bisimilarity. In the spirit of the paper we will focus on $\mathrm{EIL_h}$. The corresponding model-checking problem has not yet been investigated and it will be proved to be decidable here as a secondary result, for strongly regular event structures.

In this work we study the mentioned logics, comparing their expressive power. The focus will be on the logic $\mathcal{L}_{hp}$, used as a benchmark. We will study a logic, $\mathcal{L}_{hp}^{*\otimes}$, which combines the operators of $\mathcal{L}_{hp}$, SFL, and $\mathbb{L}_\mu$. The logic $\mathrm{EIL_h}$ will be shown to be encodable in $\mathcal{L}_{hp}^{*\otimes}$. Hence, $\mathcal{L}_{hp}^{*\otimes}$ is more powerful than all the considered logics ($\mathcal{L}_{hp}$, SFL, $\mathbb{L}_\mu$, $\mathrm{EIL_h}$). Still, we conjecture that the logical equivalence for $\mathcal{L}_{hp}^{*\otimes}$ is coarser than hhp-bisimilarity and thus that $\mathcal{L}_{hp}^{*\otimes}$ is less expressive than the full logic EIL.

We also show the decidability of model-checking in $\mathcal{L}_{hp}^{*\otimes}$, providing a local model-checking procedure for strongly regular event structures. The problem is not obvious since event structure models are infinite even for finite state systems and the possibility of expressing properties that depends on the past often leads to undecidability [20]. Indeed, even the results for the three original logics combined together in $\mathcal{L}_{hp}^{*\otimes}$ are non-trivial. The model-checking procedure is given in the form of a tableau system along the lines of [17] originally inspired by [21]. In order to check whether a system model satisfies a formula, a set of proof trees is constructed by applying suitable rules that reduce the satisfaction of a formula in a given state to the satisfaction of proper subformulae.

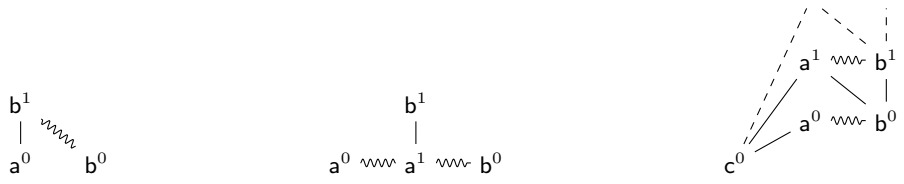
2 Event Structures

Prime event structures [1] 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 $\mathsf{a}, \mathsf{b}, \ldots$ and $\lambda : \mathbb{E} \to \Lambda$ a labelling function.

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

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

In the following, we will assume that the components of a PES $\mathcal{E}$ are named as in the definition above, possibly with subscripts. The concept of (concurrent) computation for event structures is captured by the notion of configuration.



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

Fig. 1. Some examples of finite (a)(b) and infinite (c) PESs.

Definition 2 (configuration). *A* configuration *of a* PES $\mathcal{E}$ *is a finite set of events* $C \subseteq E$ consistent *(i.e.,* $\neg(e \# e')$ *for all* $e, e' \in C$*) and* causally closed *(i.e.,* $\lceil e \rceil \subseteq C$ *for all* $e \in C$*). The set of configurations of* $\mathcal{E}$ *is denoted by* $\mathcal{C}(\mathcal{E})$.

The evolution of a system can be represented by a transition system where configurations are states.

Definition 3 (transition system). *Let* $\mathcal{E}$ *be a* PES *and let* $C \in \mathcal{C}(\mathcal{E})$. *Given* $e \in E \setminus C$ *such that* $C \cup \{e\} \in \mathcal{C}(\mathcal{E})$ *and* $X, Y \subseteq C$ *with* $X \subseteq \lceil e \rceil$, $Y \cap \lceil e \rceil = \emptyset$, *we write* $C \xrightarrow{X, \overline{Y} < e}_{\lambda(e)} C \cup \{e\}$. *The set of* enabled *events at a configuration* C *is defined as* $en(C) = \{e \in E \setminus C \mid C \cup \{e\} \in \mathcal{C}(\mathcal{E})\}$. *The* PES *is called* k-bounded *for some* $k \in \mathbb{N}$ *(or simply* bounded*) if* $|en(C)| \leq k$ *for all* $C \in \mathcal{C}(\mathcal{E})$.

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

For any configuration it is possible to identify the substructure of the PES corresponding to the transition system rooted in such configuration.

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

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

Some simple PESs are depicted in Fig. 1. Graphically, curly 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 $\mathcal{E}_3$, the event c^0, labelled by c, causes a^0 and it is concurrent with b^0. Events a^0 and b^0 are in conflict.

3 True Concurrent Logics

In this section we introduce the syntax and the semantics of the logic $\mathcal{L}_{hp}^{*\otimes}$ which arises as a join of the logics for true concurrency $\mathcal{L}_{hp}$ [12], SFL and $\mathbb{L}_\mu$ [10]. The logic has formulae that predicate over executability of events in computations and their dependency relations (causality, concurrency), and provide second order power on conflict-free sets of events in two different flavours.

3.1 Syntax

As already mentioned, formulae of $\mathcal{L}_{hp}$ include event variables, and so do formulae of $\mathcal{L}_{hp}^{*\otimes}$. They belong to a fixed denumerable set *Var*, denoted by $x, y, \ldots$. Tuples of variables like $x_1, \ldots, x_n$ will be denoted by the corresponding boldface letter $\mathbf{x}$ and, abusing the notation, tuples will be often used as sets. The logic, in positive form, includes the diamond and box modalities from $\mathcal{L}_{hp}$, the separating operators from SFL, and the second-order modalities from $\mathbb{L}_\mu$, described before.

Fixpoint operators resort to propositional variables, expressed by abstract propositions to let them interact correctly with event variables. *Abstract propositions* belong to a fixed denumerable set $\mathcal{X}^a$, ranged over by $X, Y, \ldots$. Each abstract proposition X has an arity $ar(X)$ and represents a formula with $ar(X)$ (unnamed) free event variables. For $\mathbf{y}$ such that $|\mathbf{y}| = ar(X)$, $X(\mathbf{y})$ indicates the abstract proposition X whose free event variables are named $\mathbf{y}$. We call $X(\mathbf{y})$ a *proposition* and denote by $\mathcal{X}$ the set of all propositions.

Definition 5 (syntax). *The syntax of $\mathcal{L}_{hp}^{*\otimes}$ over the sets of event variables Var, abstract propositions $\mathcal{X}^a$ and labels Λ is defined as follows:*

$$\begin{aligned}\varphi \;::=\; & Z(\mathbf{y}) \mid \mathsf{T} \mid \varphi \wedge \varphi \mid \langle\!|\mathbf{x}, \overline{\mathbf{y}} < \mathsf{a}\, z|\!\rangle\, \varphi \mid \varphi * \varphi \mid \langle\otimes\rangle\, \varphi \mid (\nu Z(\mathbf{x}).\varphi)(\mathbf{y}) \\ & \mid \mathsf{F} \mid \varphi \vee \varphi \mid [\![\mathbf{x}, \overline{\mathbf{y}} < \mathsf{a}\, z]\!]\, \varphi \mid \varphi \bowtie \varphi \mid [\otimes]\, \varphi \mid (\mu Z(\mathbf{x}).\varphi)(\mathbf{y})\end{aligned}$$

The free event variables of a formula φ are denoted $fv(\varphi)$ and defined in the obvious way. Just note that the modalities act as binders for the variable representing the event executed, hence $fv(\langle\!|\mathbf{x}, \overline{\mathbf{y}} < \mathsf{a}\, z|\!\rangle\, \varphi) = fv([\![\mathbf{x}, \overline{\mathbf{y}} < \mathsf{a}\, z]\!]\, \varphi) = (fv(\varphi) \smallsetminus \{z\}) \cup \mathbf{x} \cup \mathbf{y}$. The free propositions in φ, not bound by ν or μ, are denoted by $fp(\varphi)$. Hereafter α ranges over $\{\nu, \mu\}$. For formulae $(\alpha Z(\mathbf{x}).\varphi)(\mathbf{y})$ we require that $fv(\varphi) = \mathbf{x}$. Intuitively, the fixpoint part $\alpha Z(\mathbf{x}).\varphi$ defines a recursive formula $Z(\mathbf{x})$ whose free variables are then instantiated with $\mathbf{y}$. The formula $(\alpha Z(\mathbf{x}).\varphi)(\mathbf{x})$ will be abbreviated as $\alpha Z(\mathbf{x}).\varphi$. When both $fv(\varphi)$ and $fp(\varphi)$ are empty we say that φ is *closed*. When $\mathbf{x}$ or $\mathbf{y}$ are empty they are omitted, e.g., we write $\langle\!|\mathsf{a}\, z|\!\rangle\varphi$ for $\langle\!|\emptyset, \overline{\emptyset} < \mathsf{a}\, z|\!\rangle\varphi$ and $\alpha Z.\varphi$ for $(\alpha Z(\emptyset).\varphi)(\emptyset)$.

For example, the formula $\varphi_1 = \langle\!|\mathsf{c}\, x|\!\rangle(\langle\!|x < \mathsf{a}\, y|\!\rangle\mathsf{T} \wedge \langle\!|\overline{x} < \mathsf{b}\, z|\!\rangle\mathsf{T})$ requires that, after the execution of a c-labelled event, one can choose between a causally dependent a-labelled event and a concurrent b-labelled event. This is satisfied by $\mathcal{E}_3$ in Fig. 1c. Instead, $\varphi_2 = \langle\otimes\rangle(\mathsf{F} \bowtie \mathsf{F})$ requiring the existence of a maximal conflict-free set of enabled events which cannot be further separated, is false. Moving to infinite computations, consider $\varphi_3 = [\![\mathsf{b}\, x]\!]\, \nu Z(x).(\langle\!|\mathsf{c}\, y|\!\rangle\mathsf{T} * (\langle\!|x < \mathsf{b}\, z|\!\rangle\mathsf{T} \wedge [\![x < \mathsf{b}\, w]\!]Z(w)))$, expressing that all non-empty causal chains of b-labelled events reach a state where the system can be separated into two parallel components, one continuing the chain of b events, while the other can execute a c-labelled event. Then, φ_3 is satisfied by $\mathcal{E}_3$.

3.2 Semantics

Before defining the semantics of $\mathcal{L}_{hp}^{*\otimes}$, we need some notions, taken from [10], about conflict-free sets of enabled events, providing specific kinds of second-order quantification over them. The most general is the concept of support set.

Definition 6 (support set). *Given a* PES $\mathcal{E}$ *and a configuration* $C \in \mathcal{C}(\mathcal{E})$*, a* support set R *for* C *is either the set of enabled events* $en(C)$ *or a non-empty conflict-free set of enabled events, either way* $R \subseteq en(C)$*. We call* $\mathcal{R}(C)$ *the set of all support sets for a configuration* C*, and* $\mathcal{R}_{\mathcal{E}} = \bigcup_{C \in \mathcal{C}(\mathcal{E})} \mathcal{R}(C)$ *the set of all support sets for all possible configurations of a* PES $\mathcal{E}$*.*

Support sets are used in the logic for local reasoning on executable events. According to the definition they can be conflict-free sets, where local reasoning becomes possible since they can be decomposed into smaller ones with the same property. Alternatively a support set can contain conflicts when it is the whole set of enabled events. In the latter case proper maximal conflict-free subsets can be isolated using so-called complete supsets.

Definition 7 (complete supset). *Let* $\mathcal{E}$ *be a* PES *and* $C \in \mathcal{C}(\mathcal{E})$ *be a configuration. Given a support set* $R \in \mathcal{R}(C)$*, a* complete supset M *of* R*, denoted by* $M \sqsubseteq R$*, is a conflict-free support set* $M \in \mathcal{R}(C)$ *such that* $M \subseteq R$ *and for all* $e \in R \setminus M$ *there exists* $e' \in M$ *s.t.* $e \# e'$*. We call* $\mathcal{M}(R)$ *the set of all complete supsets of a support set* R*.*

Intuitively, to decompose conflict-free sets into smaller ones means to separate different parallel components of systems, to allow local reasoning on them. This decomposition is captured by the notion of separation.

Definition 8 (separation). *Let* $\mathcal{E}$ *be a* PES *and* $C \in \mathcal{C}(\mathcal{E})$ *be a configuration. Given a support set* $R \in \mathcal{R}(C)$*, a* separation (R_1, R_2) *of* R *is a pair of support sets* $R_1, R_2 \in \mathcal{R}(C)$ *such that* $R_1 \cap R_2 = \emptyset$ *and* $R_1 \cup R_2 \in \mathcal{M}(R)$*. We call* $\mathsf{Sep}(R)$ *the set of all possible separations of a support set* R*.*

For instance, consider $\mathcal{E}_3$ in Fig. 1c. For the initial configuration $\emptyset$ we have three possible support sets $R_1 = \{\mathsf{b}^0, \mathsf{c}^0\}$, $R_2 = \{\mathsf{b}^0\}$, and $R_3 = \{\mathsf{c}^0\}$. Among those only R_1 admits a separation (just one), i.e., $\mathsf{Sep}(R_1) = \{(R_2, R_3)\}$.

Since the logic $\mathcal{L}^{*\otimes}_{hp}$ is interpreted over PESs, the satisfaction of a formula φ is defined w.r.t. a configuration C, a support set R for C, 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 triples in $\mathcal{C}(\mathcal{E}) \times \mathcal{R}_{\mathcal{E}} \times \mathit{Env}_{\mathcal{E}}$. Given $S \subseteq \mathcal{C}(\mathcal{E}) \times \mathcal{R}_{\mathcal{E}} \times \mathit{Env}_{\mathcal{E}}$ and two tuples of variables $\mathbf{x}$ and $\mathbf{y}$, with $|\mathbf{x}| = |\mathbf{y}|$, we define $S[\mathbf{y}/_{\mathbf{x}}] = \{(C, R, \eta') \mid \exists\, (C, R, \eta) \in S \wedge \eta(\mathbf{x}) = \eta'(\mathbf{y})\}$. The semantics of $\mathcal{L}^{*\otimes}_{hp}$ also depends on a *proposition environment* $\pi : \mathcal{X} \to 2^{\mathcal{C}(\mathcal{E}) \times \mathcal{R}_{\mathcal{E}} \times \mathit{Env}_{\mathcal{E}}}$ providing an interpretation for propositions. To ensure that the semantics of a formula depends only on the events associated with its free variables and is independent on the naming of the variables, it is required that for all tuples of variables $\mathbf{x}$, $\mathbf{y}$ with $|\mathbf{x}| = |\mathbf{y}| = ar(X)$ it holds $\pi(X(\mathbf{y})) = \pi(X(\mathbf{x}))[\mathbf{y}/_{\mathbf{x}}]$. We denote by $\mathit{PEnv}_{\mathcal{E}}$ the set of proposition environments, ranged over by π.

With $\eta[x \mapsto e]$ we indicate the updated environment obtained from η where x is mapped to the event e. Similarly, for $S \subseteq \mathcal{C}(\mathcal{E}) \times \mathcal{R}_{\mathcal{E}} \times \mathit{Env}_{\mathcal{E}}$, we write $\pi[Z(\mathbf{x}) \mapsto S]$ for the corresponding update of π. For a triple $(C, R, \eta) \in \mathcal{C}(\mathcal{E}) \times \mathcal{R}_{\mathcal{E}} \times \mathit{Env}_{\mathcal{E}}$ and variables $\mathbf{x}$, $\mathbf{y}$, z, we define the $(\mathbf{x}, \overline{\mathbf{y}} < \mathsf{a}z)$-successors of (C, R, η), as

$$\mathsf{Succ}_{\mathcal{E}}^{\mathbf{x},\overline{\mathbf{y}}<\mathsf{a}z}(C,R,\eta) = \{(C', en(C'), \eta[z \mapsto e]) \mid e \in R \wedge C \xrightarrow{\eta(\mathbf{x}),\overline{\eta(\mathbf{y})}<e}_{\mathsf{a}} C'\}.$$

Definition 9 (semantics). *Let $\mathcal{E}$ be a* PES. *The denotation of a formula φ in $\mathcal{L}_{hp}^{*\otimes}$ is given by the function $\{\!|\cdot|\!\}^{\mathcal{E}} : \mathcal{L}_{hp}^{*\otimes} \to PEnv_{\mathcal{E}} \to 2^{\mathcal{C}(\mathcal{E})\times\mathcal{R}_{\mathcal{E}}\times Env_{\mathcal{E}}}$ defined inductively as follows, where we write $\{\!|\varphi|\!\}_{\pi}^{\mathcal{E}}$ instead of $\{\!|\varphi|\!\}^{\mathcal{E}}(\pi)$:*

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

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

$$\{\!|\langle\!\langle \mathbf{x}, \overline{\mathbf{y}} < \mathsf{a}\, z \rangle\!\rangle \varphi|\!\}_{\pi}^{\mathcal{E}} = \{(C,R,\eta) \mid \mathsf{Succ}_{\mathcal{E}}^{\mathbf{x},\overline{\mathbf{y}}<\mathsf{a}z}(C,R,\eta) \cap \{\!|\varphi|\!\}_{\pi}^{\mathcal{E}} \neq \emptyset\}$$

$$\{\!|[\![\mathbf{x}, \overline{\mathbf{y}} < \mathsf{a}\, z]\!] \varphi|\!\}_{\pi}^{\mathcal{E}} = \{(C,R,\eta) \mid \mathsf{Succ}_{\mathcal{E}}^{\mathbf{x},\overline{\mathbf{y}}<\mathsf{a}z}(C,R,\eta) \subseteq \{\!|\varphi|\!\}_{\pi}^{\mathcal{E}}\}$$

$$\{\!|\varphi_1 * \varphi_2|\!\}_{\pi}^{\mathcal{E}} = \{(C,R,\eta) \mid \exists\, (R_1,R_2) \in \mathsf{Sep}(R).\ \bigwedge_{i\in\{1,2\}} (C,R_i,\eta) \in \{\!|\varphi_i|\!\}_{\pi}^{\mathcal{E}}\}$$

$$\{\!|\varphi_1 \bowtie \varphi_2|\!\}_{\pi}^{\mathcal{E}} = \{(C,R,\eta) \mid \forall\, (R_1,R_2) \in \mathsf{Sep}(R).\ \bigvee_{i\in\{1,2\}} (C,R_i,\eta) \in \{\!|\varphi_i|\!\}_{\pi}^{\mathcal{E}}\}$$

$$\{\!|\langle\otimes\rangle \varphi|\!\}_{\pi}^{\mathcal{E}} = \{(C,R,\eta) \mid \exists\, M \in \mathcal{M}(R).\, (C,M,\eta) \in \{\!|\varphi|\!\}_{\pi}^{\mathcal{E}}\}$$

$$\{\!|[\otimes]\, \varphi|\!\}_{\pi}^{\mathcal{E}} = \{(C,R,\eta) \mid \forall\, M \in \mathcal{M}(R).\, (C,M,\eta) \in \{\!|\varphi|\!\}_{\pi}^{\mathcal{E}}\}$$

$$\{\!|(\nu Z(\mathbf{x}).\varphi)(\mathbf{y})|\!\}_{\pi}^{\mathcal{E}} = \nu(f_{\varphi,Z(\mathbf{x}),\pi})[\mathbf{y}/_{\mathbf{x}}] \qquad \{\!|(\mu Z(\mathbf{x}).\varphi)(\mathbf{y})|\!\}_{\pi}^{\mathcal{E}} = \mu(f_{\varphi,Z(\mathbf{x}),\pi})[\mathbf{y}/_{\mathbf{x}}]$$

where $f_{\varphi,Z(\mathbf{x}),\pi} : 2^{\mathcal{C}(\mathcal{E})\times\mathcal{R}_{\mathcal{E}}\times Env_{\mathcal{E}}} \to 2^{\mathcal{C}(\mathcal{E})\times\mathcal{R}_{\mathcal{E}}\times Env_{\mathcal{E}}}$ is the semantic function *of φ, $Z(\mathbf{x})$, π defined by $f_{\varphi,Z(\mathbf{x}),\pi}(S) = \{\!|\varphi|\!\}_{\pi[Z(\mathbf{x})\mapsto S]}^{\mathcal{E}}$ and $\nu(f_{\varphi,Z(\mathbf{x}),\pi})$ (resp. $\mu(f_{\varphi,Z(\mathbf{x}),\pi})$) denotes the corresponding greatest (resp. least) fixpoint. We say that a* PES *$\mathcal{E}$ satisfies φ if $(\emptyset, en(\emptyset), \eta) \in \{\!|\varphi|\!\}_{\pi}^{\mathcal{E}}$ for all environments η and π.*

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,R,η) when an a-labelled event e included in the set R (hence enabled in configuration C), that causally depends on (at least) the events bound to the variables in $\mathbf{x}$ and is concurrent with (at least) those bound to the variables in $\mathbf{y}$, can be executed producing a new configuration $C' = C \cup \{e\}$ which, together with the events enabled in C' and 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 in R, caused by $\mathbf{x}$ and concurrent with $\mathbf{y}$ bring to a configuration where φ is satisfied.

The formula $\varphi_1 * \varphi_2$ is satisfied by (C,R,η) if there is a separation (R_1,R_2) of R such that each formula φ_i holds in the corresponding R_i with the same configuration C and environment η. Dually, $\varphi_1 \bowtie \varphi_2$ holds if in all the possible separations of R at least one component satisfies the corresponding subformula.

The operator $\langle\otimes\rangle\varphi$ is satisfied by (C,R,η) simply when the formula φ holds after restricting R to one of its complete supsets M. Similarly, the dual $[\otimes]\varphi$ requires that φ holds for all possible restrictions of R to complete supsets.

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

Hereafter we assume that in every formula different bound propositions have different names, so that we can refer to the fixpoint subformula quantifying an abstract proposition. This requirement can always be fulfilled by alpha-renaming.

Fragments of $\mathcal{L}_{hp}^{*\otimes}$ can be easily identified which correspond to the three original logics. Hence $\mathcal{L}_{hp}^{*\otimes}$ is indeed more powerful than all of them.

3.3 Encoding $\mathrm{EIL_h}$

Here we prove that $\mathrm{EIL_h}$ [13] can be encoded in $\mathcal{L}^{*\otimes}_{hp}$ which is thus more expressive than such logic. More precisely, we show that $\mathrm{EIL_h}$ can be encoded in $\mathcal{L}_{hp}$. This also implies that the model-checking in $\mathrm{EIL_h}$ is decidable by reduction to $\mathcal{L}_{hp}$.

The encoding uses functions to remember the variables bound by forward modalities, their causal dependency with past variables, and their labels. Function $\gamma : \mathit{Var} \to 2^{\mathit{Var}}$ associates variables with the set of past variables causing them. Function $l : \mathit{Var} \to \Lambda$ associates variables with their labels. γ and l are actually partial functions defined only on a subset of variables, denoted by their dominion. Then, the procedure is as follows:

$$
\begin{aligned}
[\mathsf{T}](\gamma, l) &= \mathsf{T} & [\mathsf{F}](\gamma, l) &= \mathsf{F} \\
[\varphi \wedge \psi](\gamma, l) &= [\varphi](\gamma, l) \wedge [\psi](\gamma, l) & [\varphi \vee \psi](\gamma, l) &= [\varphi](\gamma, l) \vee [\psi](\gamma, l) \\
[\langle x : \mathsf{a} \rangle \rangle\, \varphi](\gamma, l) &= \bigvee_{\mathbf{y} \subseteq dom(\gamma)} \langle\!\langle \mathbf{y}, \overline{\mathbf{y}^c} < \mathsf{a}\, x \rangle\!\rangle\, [\varphi](\gamma[x \mapsto \mathbf{y}], l[x \mapsto \mathsf{a}]) \\
[[x : \mathsf{a}]]\, \varphi](\gamma, l) &= \bigwedge_{\mathbf{y} \subseteq dom(\gamma)} [\![\mathbf{y}, \overline{\mathbf{y}^c} < \mathsf{a}\, x]\!]\, [\varphi](\gamma[x \mapsto \mathbf{y}], l[x \mapsto \mathsf{a}]) \\
[(x : \mathsf{a})\, \varphi](\gamma, l) &= \bigvee_{z \in dom(\gamma).\, l(z) = \mathsf{a}} [\varphi[^{z}\!/_{x}]](\gamma, l) \\
[\langle\langle x \rangle\, \varphi](\gamma, l) &= \begin{cases} [\varphi](\gamma|_{\{x\}^c}, l|_{\{x\}^c}) & \text{if } \forall\, z \in dom(\gamma).\, x \notin \gamma(z) \text{ and } \mathit{fv}(\varphi) \subseteq \{x\}^c \\ \mathsf{F} & \text{otherwise} \end{cases}
\end{aligned}
$$

where $V^c = dom(\gamma) \setminus V$, for a set of variables V, and $\varphi[^{z}\!/_{x}]$ is the formula φ where all free occurrences of the variable x are substituted with variable z.

We assume that in every formula of $\mathrm{EIL_h}$ different bound variables have different names. This requirement can always be fulfilled by alpha-renaming.

The procedure allows for the encoding of any closed formula φ of $\mathrm{EIL_h}$ by computing $[\varphi](\emptyset, \emptyset)$. The correctness of the encoding can be proved by induction on the formula φ, observing that the necessary properties are preserved at each step and they vacuously hold when $\gamma = l = \emptyset$. Thus, we obtain the following.

Proposition 1 (encoding $\mathrm{EIL_h}$). *Let $\mathcal{E}$ be a* PES *and let φ be a closed formula of EIL_h, $\mathcal{E}$ satisfies φ iff it satisfies $[\varphi](\emptyset, \emptyset)$.*

4 Model Checking $\mathcal{L}^{*\otimes}_{hp}$

In this section we provide a model-checking procedure for $\mathcal{L}^{*\otimes}_{hp}$, showing that it is sound and complete over a class of PES satisfying a suitable regularity condition.

In many model-checking algorithms (e.g., [22,21]) the finiteness of the model is an essential ingredient that concurs to termination or correctness of the method. We will work on a subclass of PESs identified by finitarity requirements on the possible substructures. Given a configuration $C \in \mathcal{C}(\mathcal{E})$ and a subset $X \subseteq C$, we denote by $\mathcal{E}[C] \cup X$ the PES obtained from the residual $\mathcal{E}[C]$ by adding the events in X with the causal dependencies they had in the original PES $\mathcal{E}$.

$$(\wedge)\ \frac{C,R,\eta,\Delta \models^{\mathcal{E}} \varphi \wedge \psi}{C,R,\eta,\Delta \models^{\mathcal{E}} \varphi \quad C,R,\eta,\Delta \models^{\mathcal{E}} \psi} \qquad (\vee)\ \frac{C,R,\eta,\Delta \models^{\mathcal{E}} \varphi_1 \vee \varphi_2}{C,R,\eta,\Delta \models^{\mathcal{E}} \varphi_i} \quad i \in \{1,2\}$$

$$(\Diamond)\ \frac{C,R,\eta,\Delta \models^{\mathcal{E}} \langle\!\langle \mathbf{x}, \overline{\mathbf{y}} < \mathsf{a}\, z \rangle\!\rangle \varphi}{C', en(C'), \eta[z \mapsto e], \Delta \models^{\mathcal{E}} \varphi} \quad e \in R \text{ and } C \xrightarrow{\eta(\mathbf{x}), \overline{\eta(\mathbf{y})} < e}_{\mathsf{a}} C'$$

$$(\Box)\ \frac{C,R,\eta,\Delta \models^{\mathcal{E}} [\![\mathbf{x}, \overline{\mathbf{y}} < \mathsf{a}\, z]\!] \varphi}{C_1, en(C_1), \eta_1, \Delta \models^{\mathcal{E}} \varphi \ \ldots\ C_n, en(C_n), \eta_n, \Delta \models^{\mathcal{E}} \varphi}$$

$$\text{where } \{(C_1, en(C_1), \eta_1), \ldots, (C_n, en(C_n), \eta_n)\} = \mathsf{Succ}_{\mathcal{E}}^{\mathbf{x}, \overline{\mathbf{y}} < \mathsf{a}z}(C, R, \eta)$$

$$(*)\ \frac{C,R,\eta,\Delta \models^{\mathcal{E}} \varphi * \psi}{C,R',\eta,\Delta \models^{\mathcal{E}} \varphi \qquad C,R'',\eta,\Delta \models^{\mathcal{E}} \psi} \quad (R', R'') \in \mathsf{Sep}(R)$$

$$(\bowtie)\ \frac{C,R,\eta,\Delta \models^{\mathcal{E}} \varphi_1 \bowtie \varphi_2}{C,R_1^{p_1},\eta,\Delta \models^{\mathcal{E}} \varphi_{p_1} \ \ldots\ C,R_n^{p_n},\eta,\Delta \models^{\mathcal{E}} \varphi_{p_n}}$$

$$\text{where } \{(R_1^1, R_1^2), \ldots, (R_n^1, R_n^2)\} = \mathsf{Sep}(R) \text{ and } \forall\ i \in [1,n].\ p_i \in \{1,2\}$$

$$(\langle\otimes\rangle)\ \frac{C,R,\eta,\Delta \models^{\mathcal{E}} \langle\otimes\rangle \varphi}{C,M,\eta,\Delta \models^{\mathcal{E}} \varphi} \quad M \in \mathcal{M}(R)$$

$$([\otimes])\ \frac{C,R,\eta,\Delta \models^{\mathcal{E}} [\otimes] \varphi}{C,M_1,\eta,\Delta \models^{\mathcal{E}} \varphi \ \ldots\ C,M_n,\eta,\Delta \models^{\mathcal{E}} \varphi} \quad \text{where } \{M_1, \ldots, M_n\} = \mathcal{M}(R)$$

$$(\mathsf{Int})\ \frac{C,R,\eta,\Delta \models^{\mathcal{E}} (\alpha Z(\mathbf{x}).\varphi)(\mathbf{y})}{C,R,\eta,\Delta' \models^{\mathcal{E}} Z(\mathbf{y})} \quad \Delta' = \Delta[Z(\mathbf{x}) \mapsto \alpha Z(\mathbf{x}).\varphi]$$

$$(\mathsf{Unf}_\alpha)\ \frac{C,R,\eta,\Delta \models^{\mathcal{E}} Z(\mathbf{y})}{C,R,\eta[\mathbf{x} \mapsto \eta(\mathbf{y})],\Delta \models^{\mathcal{E}} \varphi} \quad \neg\gamma \text{ and } \Delta(Z(\mathbf{x})) = \alpha Z(\mathbf{x}).\varphi$$

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

Definition 10 (strong regularity). *A* PES $\mathcal{E}$ *is called* strongly regular *when it is bounded and for each* $k \in \mathbb{N}$ *the set* $\{\mathcal{E}[C] \cup \{e_1, \ldots, e_k\} \mid C \in \mathcal{C}(\mathcal{E}) \ \wedge\ e_1, \ldots, e_k \in C\}$ *is finite up to isomorphism of* PES*s.*

Strong regularity [17] is obtained from the notion of regularity in [19], by replacing residuals with residuals extended with a bounded number of events from the past. Intuitively, this is important since we are interested in history dependent properties. Clearly, each strongly regular PES is regular. In [18] it is shown that the PESs associated with finite safe Petri nets are strongly regular.

The model-checking procedure is given in the form of a tableau system. It follows closely the lines of [17]. The tableau rules are reported in Table 1.

Sequents contain a context C, R, η, Δ and a formula φ of $\mathcal{L}_{hp}^{*\otimes}$ which they assert to be satisfied by such context. C, R, η are the usual elements from the semantics, while Δ is a finite set of definitions $Z(\mathbf{x}) = \psi$. In such case $\Delta(Z(\mathbf{x}))$

denotes the formula ψ. In a tableau built starting from a closed formula, Δ associates a free proposition with the fixpoint subformula where it is quantified. Intuitively, Δ is like a proposition environment at syntax level. $\Delta[Z(\mathbf{x}) \mapsto \psi]$ denotes the updated definition set obtained from Δ by removing the previous definition of Z, if any, and adding $Z(\mathbf{x}) = \psi$. Δ is updated every time a fixpoint formula is encountered, i.e., when rule (Int) is applied. In such case we say that the sequent *introduces* the corresponding proposition. Then, given a sequent, if $Z(\mathbf{x}) = \psi$ is in Δ, we denote by $\Delta^{\uparrow}(Z)$ the closest ancestor introducing Z.

We next clarify when a fully constructed tableau is considered successful.

Definition 11 (successful tableau). *A* successful *tableau is a finite tableau where no more rules can be applied and every leaf is labelled by a sequent $C, R, \eta, \Delta \models^{\mathcal{E}} \varphi$ such that one of the following holds:*

1. $\varphi = \mathsf{T}$
2. $\varphi = [\![\mathbf{x}, \overline{\mathbf{y}} < \mathsf{a}\, z]\!]\, \psi$
3. $\varphi = \psi_1 \bowtie \psi_2$
4. $\varphi = Z(\mathbf{y})$ *and* $\Delta(Z(\mathbf{x})) = \nu Z(\mathbf{x}).\psi$.

Note the absence of the case $\varphi = [\otimes]\psi$ because every support set R has at least one complete supset, i.e., $\mathcal{M}(R) \neq \emptyset$, hence the sequent could not be a leaf.

For checking whether a closed formula φ of $\mathcal{L}^{*\otimes}_{hp}$ is satisfied by a strongly regular PES $\mathcal{E}$, one must build a tableau for the formula, i.e., a successful tableau rooted in the sequent $\emptyset, en(\emptyset), \eta, \emptyset \models^{\mathcal{E}} \varphi$ (where η is irrelevant since φ is closed).

Theorem 1 (tableau system). *Given a strongly regular* PES $\mathcal{E}$ *and a closed formula φ of $\mathcal{L}^{*\otimes}_{hp}$, $\mathcal{E}$ satisfies φ iff φ admits a successful tableau for $\mathcal{E}$.*

Here we just sketch the main ingredients of the proof.

Tableaux are guaranteed to be finitely branching by finitarity of the formulae of the logic and the fact that strongly regular PESs are bounded. However, tableaux could contain infinite paths because of repeated applications of the rule (Unf_α), which unfolds propositions according to their definition in Δ. To avoid this, rule (Unf_α) has a side condition involving an unspecified part γ, the so-called *stop condition*. It is intended to prevent the unfolding of a proposition when a context is reached that is equivalent, in a suitable sense to be defined, to a context occurring in an ancestor sharing the same formula.

Intuitively, two contexts are equivalent for a formula if they share the satisfaction of the formula. Recall that properties of $\mathcal{L}^{*\otimes}_{hp}$ predicate over executability of events and their dependency relations while imposing some second-order constraints. The notion below captures exactly the properties that two contexts must meet to be deemed equivalent.

Definition 12 (isomorphism of pointed supported residuals). *Given a* PES $\mathcal{E}$*, and two contexts C, R, η, Δ and C', R', η', Δ' for a formula φ, we say that they have isomorphic pointed supported residuals, written $\mathcal{E}[\langle C, R, \eta|_{fv(\varphi)}\rangle] \approx \mathcal{E}[\langle C', R', \eta'|_{fv(\varphi)}\rangle]$, if there is an isomorphism of* PES*s $\iota : \mathcal{E}[C] \to \mathcal{E}[C']$ such that $R' = \iota(R)$ and for all $x \in fv(\varphi)$, $e \in \mathcal{E}[C]$ we have $\eta(x) \leq e$ iff $\eta'(x) \leq \iota(e)$.*

It can be shown that two contexts with isomorphic pointed supported residuals for a formula either both satisfy it or none of them does. These results motivate the definition of the stop condition.

Definition 13 (stop condition). *The* stop condition γ *for rule* (Unf_α) *in Table 1 is as follows: there is an ancestor of the premise* $C, R, \eta, \Delta \models^{\mathcal{E}} Z(\mathbf{z})$ *labelled* $C', R', \eta', \Delta' \models^{\mathcal{E}} Z(\mathbf{y})$*, such that* $\Delta^{\uparrow}(Z) = \Delta'^{\uparrow}(Z)$ *and* $\mathcal{E}[\langle C, R, \eta[\mathbf{x} \mapsto \eta(\mathbf{z})]|_{\mathbf{x}}\rangle] \approx \mathcal{E}[\langle C', R', \eta'[\mathbf{x} \mapsto \eta'(\mathbf{y})]|_{\mathbf{x}}\rangle]$.

Informally, the stop condition holds when in a previous step of the construction of the tableau an instance of the same abstract proposition has been unfolded in an equivalent context, without being reintroduced. Then we can safely avoid to continue along this path because it would not add new information.

Now, a crucial observation is that, for strongly regular PESs, the number of pointed supported residuals is finite up to isomorphism. From this and the previous facts it can be shown that all tableaux are finite and the number of possible tableaux for a sequent is also finite. Moreover, we can prove the correctness of the tableau system, relying on the reduction of the semantics of fixpoint formulae to that of finite approximants and the backwards soundness of the rules.

5 Conclusions

We studied some expressive logics for true concurrency, proposed by several authors in the literature and we showed how they can be combined into a single, more powerful logic $\mathcal{L}_{hp}^{*\otimes}$. We showed also that $\mathrm{EIL_h}$ can be encoded in $\mathcal{L}_{hp}$. Except for fixpoint operators, we conjecture that also $\mathcal{L}_{hp}$ is encodable in $\mathrm{EIL_h}$.

We proved the decidability of the model-checking problem for $\mathcal{L}_{hp}^{*\otimes}$ over strongly regular PESs, providing a decision procedure in the form of a tableau system, which is correct and terminating. A concrete procedure requires the effectiveness of the transition relation over configurations and of the equivalence of pointed supported residuals, that we have if we focus on regular trace PESs, which are known to be included in (but possibly equal to) strongly regular PESs.

In [11] another logic for concurrency, called monadic trace logic (MTL), is proposed as a fragment of monadic second-order logic (MSOL), where second-order quantification is allowed only on conflict-free sets of events. Still, the possibility of directly observing conflicts and thus of distinguishing behaviourally equivalent PESs (e.g., those consisting of a single or two conflicting copies of an event), and the presence in $\mathcal{L}_{hp}^{*\otimes}$ of propositions which are non-monadic with respect to event variables, make these logics not immediate to compare. Nevertheless, some investigations point us to conjecture that they are in fact incomparable.

The intimate relation between $\mathcal{L}_{hp}^{*\otimes}$ and $\mathcal{L}_{hp}$, one of the logics combined into $\mathcal{L}_{hp}^{*\otimes}$, suggests that the model-checking procedure based on automata proposed for $\mathcal{L}_{hp}$ in [18] could be adapted for the model-checking in $\mathcal{L}_{hp}^{*\otimes}$. Furthermore, also the tool implementing such technique, presented in the same work, could be adjusted to allow the verification of $\mathcal{L}_{hp}^{*\otimes}$ properties. However, some considerations on the equivalence of pointed supported residuals lead us to think that a naive implementation would have a too high complexity to be useful in practice.

Acknowledgements. I am grateful to Paolo Baldan for insightful discussions and inspiring suggestions and to the anonymous reviewers for their comments.

References

1. Winskel, G.: Event Structures. In Brauer, W., Reisig, W., Rozenberg, G., eds.: Petri Nets: Applications and Relationships to Other Models of Concurrency. Volume 255 of LNCS., Springer (1987) 325–392
2. van Glabbeek, R., Goltz, U.: Refinement of actions and equivalence notions for concurrent systems. Acta Informatica **37**(4/5) (2001) 229–327
3. De Nicola, R., Ferrari, G.: Observational logics and concurrency models. In Nori, K.V., Madhavan, C.E.V., eds.: FSTTCS'90. Volume 472 of LNCS., Springer (1990) 301–315
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. In: Time and Logic: A Computational Approach, UCL Press (1995) 179–228
7. Nielsen, M., Clausen, C.: Games and logics for a noninterleaving bisimulation. Nordic Journal of Computing **2**(2) (1995) 221–249
8. Bradfield, J., Fröschle, S.: Independence-friendly modal logic and true concurrency. Nordic Journal of Computing **9**(1) (2002) 102–117
9. Alur, R., Peled, D., Penczek, W.: Model-checking of causality properties. In: Proceedings of LICS'95, IEEE Computer Society (1995) 90–100
10. Gutierrez, J.: On bisimulation and model-checking for concurrent systems with partial order semantics. PhD thesis, University of Edinburgh (2011)
11. Madhusudan, P.: Model-checking trace event structures. In: Proceedings of LICS 2013, IEEE Computer Society (2003) 371–380
12. Baldan, P., Crafa, S.: A logic for true concurrency. J. ACM **61**(4) (2014) 24:1–24:36
13. Phillips, I., Ulidowski, I.: Event identifier logic. Mathematical Structures in Computer Science **24**(2) (2014) 1–51
14. Best, E., Devillers, R., Kiehn, A., Pomello, L.: Fully concurrent bisimulation. Acta Informatica **28** (1991) 231–261
15. Rabinovich, A.M., Trakhtenbrot, B.A.: Behaviour structures and nets. Fundamenta Informaticae **11** (1988) 357–404
16. Degano, P., De Nicola, R., Montanari, U.: Partial orderings descriptions and observations of nondeterministic concurrent processes. In de Bakker, J.W., de Roever, W.P., Rozenberg, G., eds.: REX'88. Volume 354 of LNCS., Springer (1988) 438–466
17. Baldan, P., Padoan, T.: Local model checking in a logic for true concurrency. In Esparza, J., Murawski, A.S., eds.: FoSSaCS'17. Volume 10203 of LNCS., Springer (2017) 407–423
18. Baldan, P., Padoan, T.: Automata for true concurrency properties. In Baier, C., Dal Lago, U., eds.: FoSSaCS'18. Volume 10803 of LNCS., Springer (2018) 165–182
19. 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. Volume 2300 of LNCS., Springer (2002) 244–256
20. Jurdzinski, M., Nielsen, M., Srba, J.: Undecidability of domino games and hhp-bisimilarity. Information and Computation **184**(2) (2003) 343–368
21. Stirling, C., Walker, D.: Local model checking in the modal mu-calculus. Theoretical Computer Science **89**(1) (1991) 161–177
22. Clarke, E.M., Schlingloff, B.H.: Model checking. In Robinson, A., Voronkov, A., eds.: Handbook of Automated Reasoning. Elsevier (2001)