(Un)Decidability for History Preserving True Concurrent Logics

Paolo Baldan
University of Padova, Italy

Alberto Carraro
ITIS Zuccante, Venezia, Italy

Tommaso Padoan
University of Padova, Italy

Abstract

We investigate the satisfiability problem for a logic for true concurrency, whose formulae predicate about events in computations and their causal (in)dependencies. Variants of such logics have been studied, with different expressiveness, corresponding to a number of true concurrent behavioural equivalences. Here we focus on a mu-calculus style logic that represents the counterpart of history-preserving (hp-)bisimilarity, a typical equivalence in the true concurrent spectrum of bisimilarities.

It is known that one can decide whether or not two 1-safe Petri nets (and in general finite asynchronous transition systems) are hp-bisimilar. Moreover, for the logic that captures hp-bisimilarity the model-checking problem is decidable with respect to prime event structures satisfying suitable regularity conditions. To the best of our knowledge, the problem of satisfiability has been scarcely investigated in the realm of true concurrent logics.

We show that satisfiability for the logic for hp-bisimilarity is undecidable via a reduction from domino tilings. The fragment of the logic without fixpoints, instead, turns out to be decidable. We consider these results a first step towards a more complete investigation of the satisfiability problem for true concurrent logics, which we believe to have notable solvable cases.

2012 ACM Subject Classification Theory of computation → Logic and verification; Theory of computation → Modal and temporal logics

Keywords and phrases Event structures, history-preserving bisimilarity, true concurrent behavioural logics, satisfiability, decidability, domino systems

Digital Object Identifier 10.4230/LIPIcs.MFCS.2021.13

Funding This work is supported by the Ministero dell'Universtà e della Ricerca Scientifica of Italy, under Grant No. 201784YSZ5, PRIN2017 – ASPRA.

1 Introduction

When dealing with concurrent and distributed systems, the so-called true concurrent models are used to provide a precise account of the computational steps and of their dependencies, like causality and concurrency. An early and widely used foundational model in this class is given by Winskel's event structures [41]. 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 representing conflict. A survey on the applications of such causal models can be found in [42]. Recently they have been used in the study of concurrency in weak memory models [31, 20], for process mining and differencing [14], in the study of atomicity [15] and information flow [2] properties.

In the true concurrent approach numerous behavioural equivalences have been defined ranging from hereditary history-preserving bisimilarity to the coarser pomset and step equivalences (see, e.g., [37]). Correspondingly, behavioural logics have been proposed including operators that allow one to express causal properties of computations (see, e.g., [12, 7, 32, 29, 25, 11, 33] just to mention a few).

In particular, event-based logics have been introduced [3, 30], interpreted over event structures, expressive enough to provide a logical characterization of the main behavioural equivalences in the true concurrent spectrum [37], from hereditary history-preserving (hhp-)bisimilarity [7] to the coarser equivalences. Formulae of such logics include variables which can be bound to events in computations and describe their dependencies.

The relation between operational models, behavioural equivalences, and true concurrent logics has been widely studied and the model-checking problem has been investigated for various logics describing true concurrency properties (see, e.g.,[1, 18, 16, 17, 23, 6, 27]). The decidability of true concurrent equivalences has also been settled in various papers.

A natural problem that, to the best of our knowledge, has been scarcely investigated for true concurrent logics is satisfiability, which has been historically referred to as the *classical decision problem* [10] in the context of first-order logic. The satisfiability problem for true concurrent logics is the quest for an algorithm that, given as input any formula φ, determines whether or not there exists an event structure that satisfies φ. From this point of view, formulae are intended as abstract specifications of desired properties and event structures are abstractions of actual systems that, if implemented, should have those properties. An algorithm for satisfiability is therefore a sort of oracle that can tell system designers whether or not their desires can be realized. Obviously, checking satisfiability allows one also to verify whether two requirements, despite being syntactically different, are equivalent.

In this paper we tackle the satisfiability problem for the logic proposed in [3], referred to as $\mathcal{L}_{hp}$, corresponding to history-preserving (hp-)bisimilarity [9, 34, 13], a classical equivalence in the spectrum. The logic is endowed with least and greatest fixpoint operators, in mu-calculus style, in order to express interesting properties of infinite computations. For the propositional mu-calculus, corresponding to ordinary interleaving bisimilarity, satisfiability is decidable and every satisfiable formula has a finite model. For $\mathcal{L}_{hp}$ instead the finite model property fails, essentially because of the presence of an interpreted transitive relation (causality).

Still, hp-bisimilarity and the related logic has been shown to have good decidability properties. The equivalence itself is known to be decidable for finite safe Petri nets [38, 19, 24] (while hhp-bisimilarity is undecidable [21]). Additionally, the model-checking problem has been proved decidable for $\mathcal{L}_{hp}$ over event structures satisfying a suitable regularity property [5].

Here we show that satisfiability for $\mathcal{L}_{hp}$ is undecidable via a reduction from a well-known undecidable tiling problem [8], similarly to what was done for some two-variable logics [26]. Despite the first-order features of $\mathcal{L}_{hp}$, the reduction is not trivial since quantifications can be used in a quite restricted way: formulae can refer only to events enabled in the current configuration and execute them, inspecting their relations with (a limited number) of past events. In particular, it is impossible to relate events which are not consistent (in conflict). The "local" nature of quantifications makes it hard to constrain the event structure model to have a "grid shape". Still, we can show that, given a domino system, it is possible to construct a formula which is satisfied only by event structures embedding an event-based representation of a valid tiling for such domino. Consequently, the formula is satisfiable if and only if the domino system admits a tiling, whence the undecidability of satisfiability.

We show that, instead, for $\mathcal{L}^f_{hp}$, the fragment of $\mathcal{L}_{hp}$ without fixpoints, satisfiability is decidable. The result relies on the fact that $\mathcal{L}^f_{hp}$ can be encoded into first-order logic and the fact that it enjoys the finite model property.

We foresee that our results can be of help for settling the decidability status of other similar logics for true concurrency, like other fragments of the logic in [3], or the event identifier logic of [30] and the mu-calculi for true concurrency in [18, 16, 17], for which satisfiability has not yet been investigated.

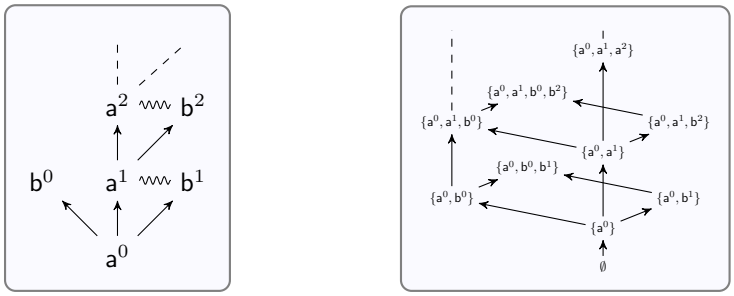

Figure 1 A simple PES $\mathcal{E}$ and (part of) its set of configurations.

2 Event Structures

We recap the basics of prime event structures [41], a widely known model of concurrency. Throughout the paper $\mathbb{E}$ is a fixed countable set of events from which all events are taken, Λ a set of labels ranged over by $\mathsf{a}, \mathsf{b}, \mathsf{c}, \ldots$, and $\lambda : \mathbb{E} \to \Lambda$ a labelling function.

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

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

Hereafter, we will assume that the components of a PES $\mathcal{E}$ are named as in the definition above, possibly with subscripts. Concurrency is a derived relation, defined as follows.

▶ **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. $e' \frown e$.

A simple PES is depicted in Fig. 1(left). Graphically, curly lines represent immediate conflicts and the causal partial order proceeds along the arrows. Events are denoted by their labels, possibly with superscripts. For instance, in $\mathcal{E}$, the events a^1 and b^1, labelled by a and b, respectively, are in conflict. Event a^0 causes the event b^0 which, in turn, is concurrent with each a^i and b^i (for $i \geq 1$).

A state of a system modelled as a PES is represented as the set of events executed to reach the state. It is formalised 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 $\mathcal{C}(\mathcal{E})$.*

In words, a configuration cannot contain events in conflict and it must be closed with respect to causality. 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 $e \in E \setminus C$ such that $C \cup \{e\} \in \mathcal{C}(\mathcal{E})$, and $X, Y \subseteq C$ with $X < e$, $Y \,||\, e$ we write $C \xrightarrow{X,\overline{Y} < e}_{\lambda(e)} C \cup \{e\}$, possibly omitting X, Y or the label $\lambda(e)$.*

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. When X or Y are empty they are normally omitted, e.g., we write $C \xrightarrow{X < e}_{\lambda(e)} C'$ for $C \xrightarrow{X,\overline{\emptyset} < e}_{\lambda(e)} C'$ and $C \xrightarrow{e}_{\lambda(e)} C'$ for $C \xrightarrow{\emptyset,\overline{\emptyset} < e}_{\lambda(e)} C'$. Some configurations of the PES $\mathcal{E}$ in Fig. 1 (left) can be found in the same figure, on the right. Examples of transitions are $\{\mathsf{a}^0, \mathsf{b}^0\} \xrightarrow{\mathsf{a}^0, \overline{\mathsf{b}^0} < \mathsf{b}^1}_{\mathsf{a}} \{\mathsf{a}^0, \mathsf{b}^0, \mathsf{b}^1\}$ and $\{\mathsf{a}^0, \mathsf{b}^0\} \xrightarrow{\mathsf{a}^0 < \mathsf{a}^1}_{\mathsf{a}} \{\mathsf{a}^0, \mathsf{b}^0, \mathsf{a}^1\}$.

A PES is called image-finite when every configuration enables a finite number of events for each fixed label. In the rest of the paper all PESs will be assumed to be image-finite. This assumption, as it commonly happens for modal logics, is crucial to have a logical characterisation of bisimilarity in terms of a finitary logic.

▶ **Definition 5** (image-finiteness). *A PES $\mathcal{E}$ is called* image-finite *when, for every configuration $C \in \mathcal{C}(E)$ and label $\mathsf{a} \in \Lambda$, the set $\{C' \in \mathcal{C}(\mathcal{E}) \mid \exists e \in E.\, C \xrightarrow{e}_{\mathsf{a}} C'\}$ is finite.*

3 A Logic for True Concurrency

We review the logic for concurrency of interest in the paper, a Hennessy-Milner style logic, originally introduced in [3], which corresponds to history-preserving bisimilarity. Its formulae predicate over executability of events in computations and their mutual relations (causality and concurrency).

Syntax

In order to specify dependencies between events in computation, formulae include event variables, from 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, besides standard propositional connectives, includes a diamond modality (and, dually, a box modality). 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.

Fixpoint operators refer 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. For dealing with fixpoint operators we fix a denumerable set $\mathcal{X}^a$ of *abstract propositions*, ranged over by X, Y, ... Each abstract proposition X has an arity $ar(X)$ and it represents a formula with $ar(X)$ (unnamed) free event variables. Then, 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 $\mathbf{x}$.

▶ **Definition 6** (hp-logic). *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:*

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

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 $fv(\langle\!|\mathbf{x}, \overline{\mathbf{y}} < \mathsf{a}\, z|\!\rangle\, \varphi) = fv([\![\mathbf{x}, \overline{\mathbf{y}} < \mathsf{a}\, z]\!]\, \varphi) = (fv(\varphi) \setminus \{z\}) \cup \mathbf{x} \cup \mathbf{y}$. The free propositions

in φ, i.e., the propositions not bound by μ, are denoted by $fp(\varphi)$. In fixpoint formulae, like $(\mu X(\mathbf{x}).\varphi)(\mathbf{y})$, we require that the tuple $\mathbf{x}$ does not include multiple occurrences of the same variable and correspond exactly to the free event variables of the inner formula φ, i.e., $fv(\varphi) = \mathbf{x}$. Intuitively, the fixpoint part $\mu X(\mathbf{x}).\varphi$ defines a recursive formula $X(\mathbf{x})$ whose free variables are then instantiated with $\mathbf{y}$. The formula $(\mu X(\mathbf{x}).\varphi)(\mathbf{x})$ will be abbreviated as $\mu X(\mathbf{x}).\varphi$. 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 often omitted, e.g., we write $\langle\!|\mathsf{a}\, z|\!\rangle\, \varphi$ for $\langle\!|\emptyset, \overline{\emptyset} < \mathsf{a}\, z|\!\rangle\, \varphi$.

Given a formula φ and variables $x, y \in Var$, we denote by $\varphi[y/x]$ the formula obtained from φ via a (capture avoiding) substitution of the free occurrences of x in φ by y. Similarly, given a proposition $Z(\mathbf{x}) \in \mathcal{X}$ and a formula ψ such that $fv(\psi) \subseteq \mathbf{x}$, we denote by $\varphi[Z(\mathbf{x}) \coloneqq \psi]$ the formula obtained from φ by replacing free occurrences of $Z(\mathbf{y})$ by $\psi[\mathbf{y}/_{\mathbf{x}}]$.

In the logic we can easily represent the possibility of performing concurrent events. Borrowing the notation from [3], we write $(\langle\!|\mathsf{a}\, z|\!\rangle \otimes \langle\!|\mathsf{b}\, z'|\!\rangle)\, \varphi$ for the formula $\langle\!|\mathsf{a}\, z|\!\rangle\langle\!|\overline{z} < \mathsf{b}\, z'|\!\rangle\varphi$ that declares the existence of two concurrent events labelled by a and b, respectively, such that if we execute such events and bind them to z and z', respectively, then φ holds.

Consider again the PES in Fig. 1. Let $\psi_{2b} = (\langle\!|\mathsf{b}\, x|\!\rangle \otimes \langle\!|\mathsf{b}\, y|\!\rangle)\mathsf{T}$ be the formula stating that two concurrent b-events can be executed. Then $\mathcal{E}$ satisfies the formula $\langle\!|\mathsf{a}\, z|\!\rangle\, \psi_{2b}$, which states that after executing an a-labelled event one can execute two concurrent b-labelled events. It satisfies also the formula $[\![\mathsf{a}\, x]\!](\nu X(x).(\psi_{2b} \wedge [\![x < \mathsf{a}\, z]\!]X(z)))$ stating that after any causal chain of a-labelled events ψ_{2b} holds. Instead the formula $\mu X.(([\![\mathsf{a}\, x]\!] \otimes [\![\mathsf{a}\, y]\!])\mathsf{T} \vee \langle\!|\mathsf{a}\, z|\!\rangle X)$ that asks for the reachability of a state where two concurrent a-labelled events can be executed, is false in $\mathcal{E}$. As a final example, the formula $\langle\!|\mathsf{a}\, x|\!\rangle(\nu X(x)\langle\!|x < \mathsf{a}\, y|\!\rangle X(y))$ asks for the existence of an infinite causal chain of a-labelled events and it is satisfied by $\mathcal{E}$.

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 an *environment* $\eta : Var \to E$, that binds free variables in the formula to events in C. Namely, if $Env_{\mathcal{E}}$ denotes the set of environments, the semantics of a formula will be a set of pairs in $\mathcal{C}(\mathcal{E}) \times Env_{\mathcal{E}}$. Given a set of pairs $S \subseteq \mathcal{C}(\mathcal{E}) \times Env_{\mathcal{E}}$ and two tuples of variables $\mathbf{x}$ and $\mathbf{y}$, with $|\mathbf{x}| = |\mathbf{y}|$, we define $S[\mathbf{y}/_{\mathbf{x}}] = \{(C, \eta') \mid \exists (C, \eta) \in S \ \wedge \eta(\mathbf{x}) = \eta'(\mathbf{y})\}$. The semantics of $\mathcal{L}_{hp}$ also depends on a proposition environment providing a semantic interpretation for propositions.

▶ **Definition 7** (proposition environment)**.** *Let $\mathcal{E}$ be a PES. A* proposition environment *is a function $\pi : \mathcal{X} \to 2^{\mathcal{C}(\mathcal{E}) \times Env_{\mathcal{E}}}$ such that for all abstract propositions X and tuples of variables $\mathbf{x}$, $\mathbf{y}$ with $|\mathbf{x}| = |\mathbf{y}| = ar(X)$ it holds $\pi(X(\mathbf{y})) = \pi(X(\mathbf{x}))[\mathbf{y}/_{\mathbf{x}}]$. The set of proposition environments, ranged by π, is denoted $PEnv_{\mathcal{E}}$.*

The condition posed on proposition environments ensures that the semantics of a formula only depends on the events that the environment associates with 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 $S \subseteq \mathcal{C}(\mathcal{E}) \times Env_{\mathcal{E}}$, we write $\pi[Z(\mathbf{x}) \mapsto S]$ for the corresponding update. For a pair $(C, \eta) \in \mathcal{C}(\mathcal{E}) \times Env_{\mathcal{E}}$ and variables $\mathbf{x}$, $\mathbf{y}$, z, we define the $(\mathbf{x}, \overline{\mathbf{y}} < \mathsf{a}z)$-successors of (C, η), as

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

In words $\mathsf{Succ}_{\mathcal{E}}^{\mathbf{x},\overline{\mathbf{y}}<\mathsf{a}z}(C,\eta)$ consists of the pairs (C',η') where C' is a configuration reachable from C, by executing an event e satisfying the requirement expressed by $\mathbf{x},\overline{\mathbf{y}} < \mathsf{a}z$, namely events in $\eta(\mathbf{x})$ are causes of e and events in $\eta(\mathbf{y})$ are concurrent with e. The environment η' is the update of η where event e has been bound to variable z.

▶ **Definition 8** (semantics). *Let $\mathcal{E}$ be a PES. The denotation of a formula in $\mathcal{L}_{hp}$ is given by the function $\{\!|\cdot|\!\}^{\mathcal{E}} : \mathcal{L}_{hp} \to PEnv_{\mathcal{E}} \to 2^{\mathcal{C}(\mathcal{E})\times Env_{\mathcal{E}}}$ defined inductively as follows, where we write $\{\!|\varphi|\!\}^{\mathcal{E}}_{\pi}$ instead of $\{\!|\varphi|\!\}^{\mathcal{E}}(\pi)$:*

$$\begin{array}{rclcrcl}
\{\!|\mathsf{T}|\!\}^{\mathcal{E}}_{\pi} & = & \mathcal{C}(\mathcal{E}) \times Env_{\mathcal{E}} & \qquad \{\!|\mathsf{F}|\!\}^{\mathcal{E}}_{\pi} = \emptyset & \qquad \{\!|Z(\mathbf{y})|\!\}^{\mathcal{E}}_{\pi} & = & \pi(Z(\mathbf{y})) \\
\{\!|\varphi_1 \wedge \varphi_2|\!\}^{\mathcal{E}}_{\pi} & = & \{\!|\varphi_1|\!\}^{\mathcal{E}}_{\pi} \cap \{\!|\varphi_2|\!\}^{\mathcal{E}}_{\pi} & & \{\!|\varphi_1 \vee \varphi_2|\!\}^{\mathcal{E}}_{\pi} & = & \{\!|\varphi_1|\!\}^{\mathcal{E}}_{\pi} \cup \{\!|\varphi_2|\!\}^{\mathcal{E}}_{\pi} \\
\{\!|\langle\!|\mathbf{x},\overline{\mathbf{y}} < \mathsf{a}\, z|\!\rangle\!\rangle\, \varphi|\!\}^{\mathcal{E}}_{\pi} & = & \{(C,\eta) \in \mathcal{C}(\mathcal{E}) \times Env_{\mathcal{E}} \mid \mathsf{Succ}_{\mathcal{E}}^{\mathbf{x},\overline{\mathbf{y}}<\mathsf{a}z}(C,\eta) \cap \{\!|\varphi|\!\}^{\mathcal{E}}_{\pi} \neq \emptyset\} & & & & \\
\{\!|[\![\mathbf{x},\overline{\mathbf{y}} < \mathsf{a}\, z]\!]\, \varphi|\!\}^{\mathcal{E}}_{\pi} & = & \{(C,\eta) \in \mathcal{C}(\mathcal{E}) \times Env_{\mathcal{E}} \mid \mathsf{Succ}_{\mathcal{E}}^{\mathbf{x},\overline{\mathbf{y}}<\mathsf{a}z}(C,\eta) \subseteq \{\!|\varphi|\!\}^{\mathcal{E}}_{\pi}\} & & & & \\
\{\!|\nu Z(\mathbf{x}).\varphi|\!\}^{\mathcal{E}}_{\pi} & = & \nu(f_{\varphi,Z(\mathbf{x}),\pi}) & & \{\!|\mu Z(\mathbf{x}).\varphi|\!\}^{\mathcal{E}}_{\pi} & = & \mu(f_{\varphi,Z(\mathbf{x}),\pi})
\end{array}$$

where $f_{\varphi,Z(\mathbf{x}),\pi} : 2^{\mathcal{C}(\mathcal{E})\times Env_{\mathcal{E}}} \to 2^{\mathcal{C}(\mathcal{E})\times Env_{\mathcal{E}}}$ is the function defined by $f_{\varphi,Z(\mathbf{x}),\pi}(S) = \{\!|\varphi|\!\}^{\mathcal{E}}_{\pi[Z(\mathbf{x})\mapsto S]}$, that we refer to as the semantic function *of φ, $Z(\mathbf{x})$, π. Moreover, $\alpha(f_{\varphi,Z(\mathbf{x}),\pi})$, for $\alpha \in \{\mu,\nu\}$, denotes the corresponding (least or greatest) 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.*

The semantics of boolean connectives is standard. The formula $\langle\!|\mathbf{x},\overline{\mathbf{y}} < \mathsf{a}\, z|\!\rangle\, \varphi$ holds in (C,η) when configuration C enables an 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 φ. The semantics of $[\![\mathbf{x},\overline{\mathbf{y}} < \mathsf{a}\, z]\!]\, \varphi$ is dual. When φ is closed (so that the environments η,π are irrelevant) and $\mathcal{E}$ satisfies the formula φ in the empty configuration, we simply say that $\mathcal{E}$ satisfies φ.

4 Undecidability of $\mathcal{L}_{hp}$

In this section we study the satisfiability problem for the logic $\mathcal{L}_{hp}$, i.e., the problem of determining whether a closed formula in $\mathcal{L}_{hp}$ is satisfied by some (image-finite) PES. We prove it to be undecidable by reduction from domino tilings.

Domino systems

Tiling problems are a simple and general form of combinatorial problems introduced in [39, 40] for proving the unsolvability of the $\forall\exists\forall$-prefix class in the pure predicate calculus. Along the years they revealed to be a powerful tool for proving undecidability results for fragments of first-order logic and for decision problems in mathematical theories (see, e.g., [10]).

▶ **Definition 9** (dominoes). *A* domino system *is a tuple $\mathcal{D} = (D,H,V)$ where D is a finite set and $H,V \subseteq D^2$ are binary relations.*

The elements of D should be thought of as square tiles (called domino pieces, or *Wang tiles*) with a color on each side and a fixed orientation. Hence for $d,e \in D$ read $(d,e) \in V$, or dVe, as "e can stand immediately above d" because the upper color of d and the lower color of e are the same. Similarly $(d,e) \in H$, or dHe, should be read as "the left side of e can be attached to the right side of d". A tiling is thus a covering of $\mathbb{N}\times\mathbb{N}$ thought as an infinite board on which each point is a spot to be occupied by a domino.

The plane $\mathbb{N}\times\mathbb{N}$ together with the binary relations $H = \{((p,q),(p+1,q)) : p,q \in \mathbb{N}\}$ and $V = \{((p,q),(p,q+1)) : p,q \in \mathbb{N}\}$ can be seen as domino system referred to as the *grid* $\mathfrak{S}_{\mathbb{N}}$. Given two domino systems $\mathcal{D} = (D,H,V)$ and $\mathcal{D}' = (D',H',V')$ a homomorphism $f : \mathcal{D} \to \mathcal{D}'$ is a function $f : D \to D'$ such that for all $(d_1,d_2) \in H$ it holds that $(f(d_1),f(d_2)) \in H'$ and, similarly, for all $(d_1,d_2) \in V$ it holds that $(f(d_1),f(d_2)) \in V'$. Tilings can be then formalised relying on the notion of homomorphism.

▶ **Definition 10** (tiling). *A* tiling *of a domino system $\mathcal{D}$ is a homomorphism $T : \mathfrak{S}_{\mathbb{N}} \to \mathcal{D}$.*

Given a tiling $T : \mathfrak{S}_{\mathbb{N}} \to \mathcal{D}$, intuitively $T(p,q) = d$ means that the point (p,q) is filled with a copy of the piece d. Our undecidability proof relies on the following well-known result about domino tilings.

▶ **Theorem 11** (undecidability of tiling [8]). *The problem of establishing the existence of a tiling for a given domino system is undecidable.*

Reduction from Domino Tiling

In order to show that $\mathcal{L}_{hp}$ is undecidable, we associate with each domino system a formula of $\mathcal{L}_{hp}$ such that the formula is satisfiable if and only if the domino system admits a tiling.

For various fragments of first-order logic (even with only two variables), the approach consists in using suitable unary predicates to establish a correspondence between elements of the model and domino pieces, and binary predicates to represent adjacency. For example, undecidability results for two-variables logics are obtained in [26, 22] using in a crucial way the transitivity of some predicates in the chosen relational vocabulary. A suitable interplay between existential and universal quantifications allows to build formulae whose models are *grid-like*, i.e., such that $\mathfrak{S}_{\mathbb{N}}$ can be embedded into these models.

Similarly, the idea here is to embed the grid inside a PES. Events are associated, via their label, with pieces of the domino and adjacency is suitably represented with arrangements of causality and concurrency between events. This is far from trivial since formulae of the logic can refer only to events enabled in the current configuration and execute them, checking their relations with (a limited number) of past events. As a consequence, quantification has a "local" nature and only the relations between consistent events can be inspected. This is quite a subtle point, since similar restrictions upon quantifications can yield decidable logics, like the guarded fragment with transitive guards studied in [35].

Greatest fixpoints like $(\nu X(\mathbf{x}).\varphi)(\mathbf{y})$ can be used to ensure that what is being predicated by $\varphi(\mathbf{x})$ holds repeatedly throughout an unbounded number of reachable configurations. For a given domino system, we define formulae that verify the proper adjacency of pieces by exploring the events along diagonal slices of the grid, starting from the bottom row and ending at the leftmost column. All slices are finite causal chains but overall they grow unboundedly in length. The formula for a domino system is quite complex and not immediate to read, but the underlying intuition will be explained in detail after the definition.

▶ **Definition 12** (formula for a domino). *Let $\mathcal{D}$ be a domino system with $D = \{d_1,\dots,d_n\}$. Define the set of $6n$ labels $A = \{\mathsf{b}^s_k \mid \mathsf{b} \in \{\mathsf{a},\mathsf{i},\mathsf{j}\} \wedge k \in \{1,\dots,n\} \wedge s \in \{0,1\}\}$. For $s \in \{0,1\}$ we let $\bar{s}$ abbreviate $1-s$. Consider the following formulae (1)–(4).*

$$\bigvee_{\substack{d_k\ H\ d_h\\ d_k\ V\ d_v}} \langle\!|\mathsf{i}^0_k\, x|\!\rangle\langle\!|x < \mathsf{i}^1_h\, y|\!\rangle\langle\!|y < \mathsf{j}^1_v\, z|\!\rangle\mathsf{T} \tag{1}$$

$$\bigwedge_{\substack{\mathsf{b}\in\{\mathsf{a},\mathsf{j}\}\\ d_k,d_l\in D\\ s\in\{0,1\}}} [\![\mathsf{i}_k^s\, x]\!][\![x < \mathsf{b}_l^s\, y]\!]\,(\nu X(x,y,z).$$

$$\bigvee_{d_k\, H\, d_h} \langle\!\langle x, \overline{y} < \mathsf{i}_h^{\bar{s}}\, u\rangle\!\rangle \mathsf{T} \wedge \bigwedge_{\substack{\mathsf{c}\in\{\mathsf{a},\mathsf{j}\}\\ d_m\in D}} [\![z < \mathsf{c}_m^s\, v]\!] X(x,y,v))(x,y,y) \tag{2}$$

$$\bigwedge_{\substack{\mathsf{b}\in\{\mathsf{a},\mathsf{i}\}\\ \mathsf{c}\in\{\mathsf{a},\mathsf{j}\}\\ d_l,d_m\in D\\ d_k\, H\, d_h\\ s\in\{0,1\}}} [\![\mathsf{b}_k^s\, x]\!][\![x < \mathsf{a}_l^s\, y]\!][\![y < \mathsf{c}_m^s\, z]\!][\![x, \overline{y} < \mathsf{b}_h^{\bar{s}}\, w]\!]\,(\nu X(y,z,w,u).$$

$$\bigvee_{\substack{d_k\, V\, d_p\\ d_l\, H\, d_p}} \langle\!\langle y, w, \overline{z} < \mathsf{a}_p^{\bar{s}}\, r\rangle\!\rangle \mathsf{T} \wedge \bigwedge_{\substack{\mathsf{e}\in\{\mathsf{a},\mathsf{j}\}\\ d_q\in D}} [\![u, \overline{w} < \mathsf{e}_q^s\, v]\!] X(y,z,w,v))(y,z,w,z) \tag{3}$$

$$\bigwedge_{\substack{\mathsf{b}\in\{\mathsf{a},\mathsf{i}\}\\ d_l\in D\\ d_k\, H\, d_h\\ s\in\{0,1\}}} [\![\mathsf{b}_k^s\, x]\!][\![x < \mathsf{j}_l^s\, y]\!][\![x, \overline{y} < \mathsf{b}_h^{\bar{s}}\, z]\!] \bigvee_{\substack{d_k\, V\, d_p\\ d_l\, H\, d_p\\ d_l\, V\, d_q}} \langle\!\langle y, z < \mathsf{a}_p^{\bar{s}}\, v\rangle\!\rangle \langle\!\langle v < \mathsf{j}_q^{\bar{s}}\, w\rangle\!\rangle \mathsf{T} \tag{4}$$

Calling ψ_i the corresponding formula (i) above, we denote the formula for the domino system $\mathcal{D}$ by $\varphi_{\mathcal{D}} = \psi_1 \wedge \nu Z.(\psi_2 \wedge \psi_3 \wedge \psi_4 \wedge [\![A\, z]\!]Z)$, where we write $[\![A\, z]\!]Z$ for $\bigwedge_{\mathsf{b}_k^s\in A} [\![\mathsf{b}_k^s\, z]\!]Z$.

Intuitively, the formula $\varphi_{\mathcal{D}}$ requires a model to contain a grid of consistent events, as depicted in Fig. 2. The formula also arranges causal dependencies that constrain the order of exploration, i.e., of execution, of the grid as an ever growing right-angled triangle. This allows to build the grid one diagonal at a time, starting from the leftmost smallest diagonal which consists of a single event at coordinates $(0,0)$ in Fig. 2. Every diagonal, except the first, is delimited by two events with special labels: the first at the bottom of the diagonal is labelled i, the last at the top is labelled j. Inner events are, instead, all labelled a. Actually, labels include also a subscript and a superscript. Subscripts $k \in \{1,\ldots,n\}$ represent the associated domino piece d_k. Superscripts $s \in \{0,1\}$, instead, are used to distinguish events in a diagonal from those in the next and previous ones. So the superscript for all the events in a diagonal starting at coordinates $(t,0)$ is simply $s = t \bmod 2$, as shown in the figure.

To explain how the formula works we first comment on how the satisfaction of the formula implies the existence of a grid in the model. For this, the superscripts on labels play no role and can be safely ignored. They will become relevant later for the converse implication.

The first two diagonals are determined by the first subformula ψ_1 (1). This formula simply requires the existence of three events, executable from the initial state, such that each one causes the next. Such events represent the first three domino pieces in the tiling. So, they are required to be labelled $\mathsf{i}_k, \mathsf{i}_h, \mathsf{j}_v$, in a way that the adjacency constraints are respected, i.e., $d_k\, H\, d_h$ and $d_k\, V\, d_v$. Each diagonal beyond the first two is, instead, jointly defined by the other three subformulae ψ_2, ψ_3, ψ_4, which are guaranteed to be checked on every reachable state of the computation via the outermost greatest fixpoint of $\varphi_{\mathcal{D}}$.

In Fig. 3 we give a graphical representation of such three subformulae. In each sub-figure, greyed out events represent the configuration to which the subformula is intended to apply. Events highlighted in red are universally quantified by box operators in the formula, while events highlighted in blue are existentially quantified by diamond operators. The picture also reports, close to events, the variables to which such events are bound in the formula.

The subformula ψ_2 (2) starts a new diagonal. The first event in the new diagonal must be labelled i_p, for some domino piece d_p horizontally compatible with the one corresponding to the first event in the previous diagonal, labelled i_l in the figure. Moreover, the new event i_p must be caused by i_l and concurrent with the second event in the previous diagonal, labelled a_m in the figure. In addition, using a fixpoint subformula, we ask that the new event is executed after (hence consistent with) the whole red diagonal. This implies that i_p is necessarily concurrent with all the events in the red diagonal, although not explicitly required. In fact, since it is consistent and executed after the diagonal, the only alternative would be that i_p were caused by some event in the diagonal. But this would mean that also a_m causes i_p, while we know that they are concurrent.

The subformula ψ_3 (3) continues the "construction" of the new diagonal starting from the second event and, with each successive application, up to the third-to-last event of the new diagonal. Basically, it builds the a-labelled part of the new diagonal, except the last a-labelled event. In particular, a single application of ψ_3 ensures the executability of an event labelled a_q, for some domino piece d_q vertically compatible with the one corresponding to the event just below in the grid, which is necessarily part of the previous diagonal, labelled i_l in the figure, and horizontally compatible with the one just after the latter in the previous diagonal, that is a_m in the example. The new event a_q must be caused by the one just before it in the new diagonal, i_p in this case. Furthermore, as in the previous property, the new event must be caused by the event a_m at the same height in the previous diagonal, and concurrent with the events coming after it in such diagonal.

Finally, the subformula ψ_4 (4) concludes the construction of the new diagonal, requiring the executability of the last two events labelled a_s and j_t, respectively. As before, the corresponding domino pieces d_s and d_t must be compatible with those of the adjacent events below and on the left, all of which belonging to the previous diagonal. Moreover, the new event a_s must be caused by the one just before it in the new diagonal and by the last event of the previous diagonal. Instead, j_t is just required to be caused by a_s, which, however, by transitivity of the causality relation, means that j_t is caused by every other event in the new diagonal and all those in the previous diagonals.

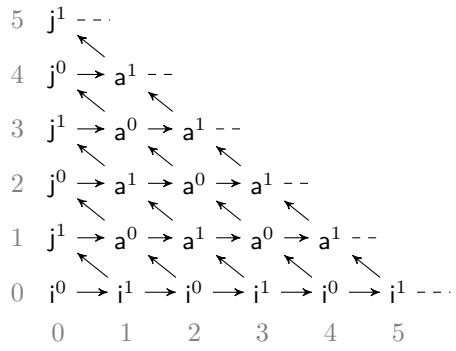

Figure 2 Grid of events for the domino tiling (with simplified event labels).

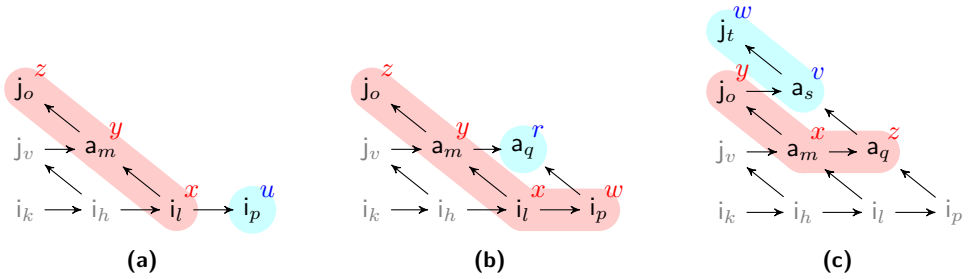

Figure 3 Graphical representation of the properties, from left to right, (2), (3) and (4).

Note that the subformulae ψ_2 and ψ_3 use a (greatest) fixpoint in order to fully explore a diagonal, which is unbounded. This ensures the consistency of each newly added event with the previous diagonal and thus with the whole right-angled triangle up to such diagonal.

Relying on the intuitions described above we can prove the desired result: given a domino system $\mathcal{D}$, the formula $\varphi_{\mathcal{D}}$ is satisfiable if and only if $\mathcal{D}$ admits a tiling. We next present a sketch of the proof.

For proving the first implication we show how, given a PES which satisfies the formula $\varphi_{\mathcal{D}}$, one can build a tiling for the domino system $\mathcal{D}$. Recall that a tiling of $\mathcal{D}$ is a function $T : \mathbb{N} \times \mathbb{N} \to D$ complying with the adjacency relations H and V of $\mathcal{D}$. Proceeding, as mentioned before, by diagonals, we can inductively define an infinite chain of functions f_i, for all $i \in \mathbb{N}^+$, whose domain is the right-angled triangle up to the $(i+1)$-th diagonal, i.e. $\{(x, y) \in \mathbb{N} \times \mathbb{N} \mid x + y \leq i\}$. The first function f_1 is obtained directly from the events guaranteed to exist by the satisfaction of the subformula ψ_1 of $\varphi_{\mathcal{D}}$. Every other function f_i is defined extending f_{i-1} and using the events whose existence is required by the other subformulae of $\varphi_{\mathcal{D}}$. The join of the f_i's is defined on the whole grid and provides a tiling.

▶ **Theorem 13** (satisfiability implies tiling). *Let $\mathcal{D}$ be a domino system with $D = \{d_1, \ldots, d_n\}$, if the corresponding formula $\varphi_{\mathcal{D}}$ is satisfiable, then $\mathcal{D}$ admits a tiling.*

For the converse implication, we need to show how to transform a tiling T of $\mathcal{D}$ into a PES which satisfies the property $\varphi_{\mathcal{D}}$. From the grid of domino pieces corresponding to the tiling T we define a PES $\mathcal{E}$ whose events are $E = \mathbb{N} \times \mathbb{N}$. Events are labelled i_k^s if they belong to the bottom row, j_k^s if in the left-most column, except $(0, 0)$, or a_k^s otherwise, where k is the subscript of the corresponding domino piece d_k occupying the same position, and s is the index corresponding to the diagonal to which the event belongs. Explicitly, if the coordinates of the event are (x, y), then the index is $s = (x + y) \bmod 2$, hence it is the same for all the events in a same diagonal. The PES has empty conflict relation, while causality is defined as in Fig. 4. In this way, every event

- is caused exactly by those at lower or equal height in the smallest right-angled triangle containing the event,
- causes those at higher or equal height outside of such triangle or above the event along its diagonal,
- is concurrent with all the others.

For instance, consider the event at position $(4, 2)$ in Fig. 4, labelled a_k^0. It is caused by the events highlighted in red, it causes those highlighted in green, and it is concurrent with those highlighted in blue. It is easy to see that $\mathcal{E}$ is well-defined, i.e. it is a PES, in fact, the set of causes of each event is clearly finite. Moreover, $\mathcal{E}$ is image-finite since every configuration enables a finite number of events (bounded by 1 plus half the size of the configuration).

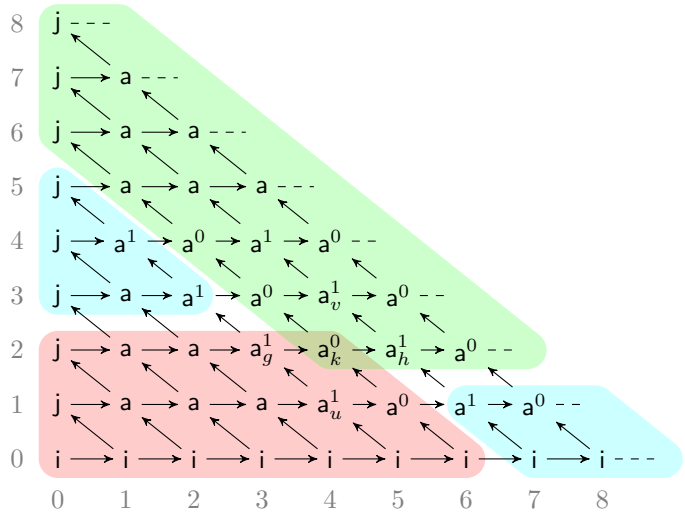

Figure 4 PES for the canonical grid of the domino tiling (with simplified event labels).

Formalising the ideas outlined above, one can prove that $\varphi_{\mathcal{D}}$ holds in the initial state of $\mathcal{E}$. In particular, the formulae ψ_2, ψ_3 and ψ_4, inside the outermost fixpoint, can be shown to hold in every reachable state, sometimes vacuously. For example, consider the configuration C consisting of all the events appearing in Fig. 4 except those highlighted in green. Let us focus on the subformula ψ_3 (3) and argue that it holds in configuration C. Observe that the initial three box modalities in ψ_3 require a specific structure to be executable, in absence of which the formula holds vacuously. Such structure consists of a causal chain of three events, labelled with some combination of letters $\mathsf{i}, \mathsf{a}, \mathsf{j}$ but all with the same superscript s. Inspecting the structure in the figure, it occurs that there are only two possible causal chains of three events executable from C: both starting with $(4, 2)$, and then going either along its diagonal up to $(2, 4)$, or along its row up to $(6, 2)$. However, since the three events must be labelled with the same superscript s, only the chain along the diagonal is actually considered by the formula (along rows events alternate superscripts instead). Then, exploiting the similarities with the graphical representation of ψ_3 in Fig. 3b, it is possible to see that after binding those events the rest of the property holds.

▶ **Theorem 14** (tiling implies satisfiability). *Let $\mathcal{D}$ be a domino system with $D = \{d_1, \ldots, d_n\}$, if $\mathcal{D}$ admits a tiling, then the formula $\varphi_{\mathcal{D}}$ is satisfiable.*

By the theorems above and the undecidability of the domino problem we conclude.

▶ **Corollary 15** (undecidability of $\mathcal{L}_{hp}$). *The satisfiability problem for $\mathcal{L}_{hp}$ is undecidable.*

5 Decidability without fixpoints

In this section we show that, in absence of fixpoint operators, the logic $\mathcal{L}_{hp}$ has the finite model property. Moreover, we provide an encoding of formulae into first-order logic preserving their (un)satisfiability. As a consequence we deduce tha satisfiability for the logic without fixpoints becomes decidable.

In the following we will denote by $\mathcal{L}^f_{hp}$ the fragment of the logic $\mathcal{L}_{hp}$ without fixpoint operators (and propositions). Consequently, the semantics of formulae in $\mathcal{L}^f_{hp}$ can be defined without proposition environments π, since there are no propositions to interpret.

In order to prove the finite model property the idea consists in showing that every model of a formula can be reduced to a finite one by restricting to a suitable chosen subset of events. To this aim, we introduce a way to truncate PESs by keeping only events up to a certain causal level and with a specific subset of labels. The causal level of an event e is inductively defined as $lev(e) = \max\{lev(e') + 1 \mid e' \in E \wedge e' < e\}$, where it is intended that $\max \emptyset = 0$.

▶ **Definition 16** (prefix of a PES). *Let E be a PES. For $k \in \mathbb{N}$ and $A \subseteq \Lambda$, consider the set of events $E^{(A,k)} = \{e \in E \mid lev(e) \leq k \wedge \forall e' \in \lceil e \rceil . \lambda(e') \in A\}$. Then, the A-*labelled *k-*prefix *of $\mathcal{E}$ is the PES defined as $\mathcal{E}^{(A,k)} = \langle E^{(A,k)}, <|_{E^{(A,k)}}, \#|_{E^{(A,k)}} \rangle$.*

Note that, by the very definition of causal level, $lev(e') < lev(e)$ for all $e' < e$; hence, the (A-labelled) k-prefix $\mathcal{E}^{(A,k)}$ of a PES $\mathcal{E}$ is a causally closed subset of $\mathcal{E}$. From this observation, it immediately follows that $\mathcal{E}^{(A,k)}$ is indeed a PES, i.e., the definition is well-given.

Notably, when a PES is image-finite, the same holds for all its prefixes. Hence, for every $k \in \mathbb{N}$ and finite A, the A-labelled k-prefix $\mathcal{E}^{(A,k)}$ can be shown to be finite.

▶ **Lemma 17** (finiteness of prefixes). *Let $\mathcal{E}$ be a image-finite PES. For all $k \in \mathbb{N}$ and $A \subseteq \Lambda$, if A is finite, then $\mathcal{E}^{(A,k)}$ is finite.*

Now, in order to prove the finite model property of $\mathcal{L}^f_{hp}$ it is enough to show that the satisfaction of formulae of $\mathcal{L}^f_{hp}$ is preserved when truncating a PES up to a suitable level k and set of labels A. Both k and A can be obtained directly from the formula. Let the *modal depth* of a formula φ, denoted by $d(\varphi)$, be defined as usual. If φ is T or F, its modal depth is 0. If it is a conjunction or disjunction, the modal depth is the maximum of those of the conjuncts, resp. disjuncts. Otherwise, when φ consists of a modality followed by a subformula ψ, the modal depth is $d(\varphi) = 1 + d(\psi)$. Let $A(\varphi)$ be the (finite) set of labels appearing in the formula φ. Then, whenever a formula φ is satisfied by a PES $\mathcal{E}$, it is also satisfied by the $A(\varphi)$-labelled $d(\varphi)$-prefix of $\mathcal{E}$, which, by the previous lemma, is finite.

▶ **Theorem 18** (finite model property of $\mathcal{L}^f_{hp}$). *Let φ be a closed formula of $\mathcal{L}^f_{hp}$. If φ is satisfiable, then there exists a finite PES satisfying φ.*

The result above implies that satisfiability for $\mathcal{L}^f_{hp}$ is semi-decidable. In fact, finite PESs are denumerable and checking whether a finite PES satisfies a formula is decidable. Then, to conclude it is sufficient to observe that the axioms of PESs are expressible as first-order formulae and $\mathcal{L}^f_{hp}$, as it happens for many modal logics, can be encoded into first-order logic, hence also unsatisfiability is semi-decidable.

First, as mentioned, for a fixed formula in $\mathcal{L}^f_{hp}$ the set of labels appearing in it is finite. Once the finite set of labels A is fixed, the theory of prime event structures, apart from the axiom of finite causes, is expressible as a finite set of first-order axioms.

▶ **Definition 19** (first-order theory of PESs). *Let $A \subseteq \Lambda$ be a finite set of labels. The first order theory of theory of PESs over A, consists of the following axioms with $<$ and $\#$ as binary predicates and labels as unary predicates.*

1. $\forall x, y, z.\ (x < y) \wedge (y < z) \rightarrow (x < z)$
2. $\forall x.\ \neg(x < x)$
3. $\forall x.\ \neg(x \# x)$
4. $\forall x, y.\ (x \# y) \rightarrow (y \# x)$
5. $\forall x, y, z.\ (x < y) \wedge (x \# z) \rightarrow (y \# z)$
6. $\forall x.\ \left(\bigvee_{\mathsf{a} \in A} \mathsf{a}(x) \ \wedge \ \bigwedge_{\mathsf{a},\mathsf{b} \in A, \mathsf{a} \neq \mathsf{b}} \neg(\mathsf{a}(x) \wedge \mathsf{b}(x))\right)$

We denote by $T_{PES}(A)$ the conjunction of the above axioms.

Axioms (1) and (2) state that $<$ is a (strict) partial order. Axioms (3) and (4) ask conflict $\#$ to be irreflexive and symmetric, while (5) requires conflict to be inherited along causality. Finally (6) asks that each event has exactly one label.

Now, given a set of variables $X \subseteq \mathit{Var}$ and a variable $y \in \mathit{Var}$, let us write $y \in X$ as an abbreviation for $\bigvee_{x \in X}(x = y)$. Then we can express the property of being a configuration as:

$$\begin{aligned} \mathit{conf}(X) \equiv\ & \forall x, y.((x \in X) \wedge (y < x) \rightarrow (y \in X)) \wedge \\ & \forall x, y.((x \# y) \rightarrow \neg((x \in X) \wedge (y \in X)) \end{aligned}$$

We can finally provide the translation of $\mathcal{L}^f_{hp}$ into first-order formulae.

▶ **Definition 20** (compiling $\mathcal{L}^f_{hp}$ to first-order logic)**.** *Let X be a finite set of variables and let φ be a formula of $\mathcal{L}^f_{hp}$. We denote by $(\varphi)_X$ the first-order formula inductively defined as follows:*

- $(\mathsf{T})_X = \mathsf{T}$ *and* $(\mathsf{F})_X = \mathsf{F}$
- $(\varphi \wedge \psi)_X = (\varphi)_X \wedge (\psi)_X$ *and* $(\varphi \vee \psi)_X = (\varphi)_X \vee (\psi)_X$
- $(\langle\!|\mathbf{x}, \overline{\mathbf{y}} < \mathsf{a}\, z|\!\rangle\varphi)_X = \exists z.\ \ \bigwedge_{x \in \mathbf{x}}(x < z)\ \wedge \bigwedge_{y \in \mathbf{y}} \neg(y < z) \wedge \mathsf{a}(z) \wedge \neg(z \in X) \wedge \mathit{conf}(X \cup \{z\}) \wedge (\varphi)_{X \cup \{z\}}$
- $([\![\mathbf{x}, \overline{\mathbf{y}} < \mathsf{a}\, z]\!]\varphi)_X = \forall z.\ \ (\bigwedge_{x \in \mathbf{x}}(x < z)\ \wedge \bigwedge_{y \in \mathbf{y}} \neg(y < z) \wedge \mathsf{a}(z) \wedge \neg(z \in X) \wedge \mathit{conf}(X \cup \{z\})) \rightarrow (\varphi)_{X \cup \{z\}}$

Then, given a closed formula φ of the logic $\mathcal{L}^f_{hp}$ we can obtain an equisatisfiable first-order formula by taking the conjuction of the first order theory of PESs and the encoding of φ defined above.

▶ **Proposition 21.** *Let φ be a closed formula of $\mathcal{L}^f_{hp}$ and let A be the set of labels occurring in φ. It holds that φ is satisfiable iff the first-order formula $T_{PES}(A)\ \wedge\ (\varphi)_\emptyset$ is satisfiable.*

The proof is straightforwardly based on the observation that a PES $\mathcal{E}$ satisfying φ can be seen as a first-order structure satisfying $T_{PES}(A)\ \wedge\ (\varphi)_\emptyset$, and vice versa. The only delicate aspects is the absence of the axiom of finite causes in $T_{PES}(A)$. Hence, it could happen that $T_{PES}(A)\ \wedge\ (\varphi)_\emptyset$ is satisfiable by a structure which, seen as a PES $\mathcal{E}$, includes events with infinitely many causes. However, in this case, since these events would never be executable, it is clear that also the PES $\mathcal{E}'$ obtained from $\mathcal{E}$ by removing all events with infinitely many causes is a model for φ.

We can finally deduce that the satisfiability for $\mathcal{L}^f_{hp}$ is decidable.

▶ **Corollary 22** (decidability of $\mathcal{L}^f_{hp}$)**.** *The satisfiability problem for the logic fragment $\mathcal{L}^f_{hp}$ is decidable and every satisfiable formula has a finite model.*

The proof combines the results proved above. First, by Theorem 18, when a formula φ in $\mathcal{L}^f_{hp}$ is satisfiable it has a finite model labelled over the finite set A of labels occurring in φ. Since the finite PESs labelled over a finite alphabet are denumerable and checking whether a finite PES satisfies a formula is decidable, we can semi-decide satisfiability of a formula φ by enumerating the finite PESs labelled over A and checking whether each of the generated PES satisfies φ. Moreover, by Proposition 21, unsatisfiability of a formula φ in $\mathcal{L}^f_{hp}$ is reducible to unsatisfiability of the first-order formula $T_{PES}(A)\ \wedge\ (\varphi)_\emptyset$, and thus it is semi-decidable. We conclude that satisfiability for $\mathcal{L}^f_{hp}$ is decidable.

6 Conclusions and Perspectives

The logic $\mathcal{L}_{hp}$ investigated in this paper is one of a number of fragments of a logic introduced in [3]. Other fragments $\mathcal{L}_p$ and $\mathcal{L}_s$ can be obtained by syntactical restrictions, characterising coarser true concurrent notions of bisimilarity, namely pomset and step bisimilarity. The full logic, instead, corresponds to hereditary hp-bisimilarity, the finest behavioural equivalence in the true concurrent spectrum of [37], finer than hp-bisimilarity. Each logic fragment admits a variant with fixpoints and a variant without fixpoints. However, the induced logical equivalences for image-finite PESs are the same with or without fixpoints. We proved that the satisfiability problem for $\mathcal{L}_{hp}$ is undecidable, and thus the same holds also for the logic for hereditary hp-bisimilarity in [3]. On the other hand, satisfiability is decidable for $\mathcal{L}^f_{hp}$, the fragment of $\mathcal{L}_{hp}$ without fixpoints.

Some preliminary investigations suggest that the step logic $\mathcal{L}_s$ (with fixpoints) is decidable via a reduction to the propositional μ-calculus. The same technique appears to be promising for the pomset logic $\mathcal{L}_p$ (with fixpoints) but this case is more complex and unresolved to this day. Similar logics for true concurrent properties are event identifier logic of [30] and the mu-calculi for true concurrency in [18, 16, 17]. Also in this case, to the best of our knowledge satisfiability has not yet been investigated. This offer a range of open questions that, when answered, would draw an interesting picture of problems across the decidability border.

In a sense there are "two dimensions" to the satisfiability problem: one is the syntax and the other the semantics, so that there are also many interesting variants and facets of the satisfiability question when the restrictions are imposed on the model side. For example a notable semantics is that of *regular* models in the sense of [36]. Investigating whether restricting the semantics with the constraint of regularity affects decidability is an intriguing direction of future work.

A formalisation of the semantics of the logics in terms of suitable (parity) games is often a source of inspiration for facing complexity and decidability issues for modal logics. Currently, there is no established game-theoretical characterisation of the semantics of the logic in [3], of which $\mathcal{L}_{hp}$ is a fragment. However, such a development could be naturally guided by the approach in [16, 17] and by the relation between fixpoint games [4] and logics for concurrency, hinted at in [28]. This also appears as an interesting route to explore.

References

1 Rajeev Alur, Doron A. Peled, and Wojciech Penczek. Model-checking of causality properties. In *Proceedings of LICS'95*, pages 90–100. IEEE Computer Society, 1995.

2 Paolo Baldan and Alberto Carraro. A causal view on non-intereference. *Fundamenta Informaticae*, 140(1):1–38, 2015.

3 Paolo Baldan and Silvia Crafa. A logic for true concurrency. *Journal of the ACM*, 61(4):24:1–24:36, 2014.

4 Paolo Baldan, Barbara König, Christina Mika-Michalski, and Tommaso Padoan. Fixpoint games in continuous lattices. *PACMPL*, 3(POPL):26:1–26:29, 2019.

5 Paolo Baldan and Tommaso Padoan. Local model checking in a logic for true concurrency. In Javier Esparza and Andrzej S. Murawski, editors, *Proceedings of FoSSaCS'17*, volume 10203 of *LNCS*, pages 407–423. Springer, 2017.

6 Paolo Baldan and Tommaso Padoan. Model checking a logic for true concurrency. *ACM Trans. Comput. Log.*, 21(4):34:1–34:49, 2020.

7 Marek A. Bednarczyk. Hereditary history preserving bisimulations or what is the power of the future perfect in program logics. Technical report, Polish Academy of Sciences, 1991.

8 Robert Berger. *The Undecidability of the Domino Problem*. Memoirs ; No 1/66. American Mathematical Society, 1966.

9 Eike Best, Raymond Devillers, Astrid Kiehn, and Lucia Pomello. Fully concurrent bisimulation. *Acta Informatica*, 28:231–261, 1991.

10 Egon Börger, Erich Grädel, and Yuri Gurevich. *The Classical Decision Problem*. Universitext. Springer Berlin Heidelberg, 2001.

11 J. Bradfield and S. Fröschle. Independence-friendly modal logic and true concurrency. *Nordic Journal of Computing*, 9(1):102–117, 2002.

12 R. De Nicola and G. Ferrari. Observational logics and concurrency models. In K. V. Nori and C. E. V. Madhavan, editors, *Proceedings of FST-TCS'90*, volume 472 of *LNCS*, pages 301–315. Springer, 1990.

13 Pierpaolo Degano, Rocco De Nicola, and Ugo Montanari. Partial orderings descriptions and observations of nondeterministic concurrent processes. In Jaco W. de Bakker, Willem P. de Roever, and Grzegorz Rozenberg, editors, *REX Workshop*, volume 354 of *LNCS*, pages 438–466, Heidelberg, DE, 1988. Springer.

14 Marlon Dumas and Luciano García-Bañuelos. Process mining reloaded: Event structures as a unified representation of process models and event logs. In R. R. Devillers and A. Valmari, editors, *Petri Nets 2015*, volume 9115 of *LNCS*, pages 33–48. Springer, 2015.

15 Azadeh Farzan and P. Madhusudan. Causal atomicity. In T. Ball and R. B. Jones, editors, *Proceedings of CAV'06*, volume 4144 of *LNCS*, pages 315–328, 2006.

16 Julian Gutierrez. Logics and bisimulation games for concurrency, causality and conflict. In L. de Alfaro, editor, *Proceedings of FoSSaCS'09*, volume 5504 of *LNCS*, pages 48–62. Springer, 2009.

17 Julian Gutierrez. *On bisimulation and model-checking for concurrent systems with partial order semantics*. PhD thesis, University of Edinburgh, 2011.

18 Julian Gutierrez and Juilian C. Bradfield. Model-checking games for fixpoint logics with partial order models. In M. Bravetti and G. Zavattaro, editors, *Proceedings of CONCUR'09*, volume 5710 of *LNCS*, pages 354–368. Springer, 2009.

19 Lalita Jategaonkar and Albert R. Meyer. Deciding true concurrency equivalences on safe, finite nets. *Theoretical Computer Science*, 154(1):107–143, 1996.

20 Alan Jeffrey and James Riely. On thin air reads towards an event structures model of relaxed memory. In M. Grohe, E. Koskinen, and N. Shankar, editors, *Proceedings of LICS'16*, pages 759–767. ACM, 2016.

21 Marcin Jurdzinski, Mogens Nielsen, and Jirí Srba. Undecidability of domino games and hhp-bisimilarity. *Information and Computation*, 184(2):343–368, 2003.

22 Emanuel Kieronski. Results on the guarded fragment with equivalence or transitive relations. In C.-H. Luke Ong, editor, *Computer Science Logic*, volume 3634 of *LNCS*, pages 309–324. Springer, 2005.

23 P. Madhusudan. Model-checking trace event structures. In *Proceedings of LICS 2013*, pages 371–380. IEEE Computer Society, 2003.

24 Ugo Montanari and M. Pistore. Minimal transition systems for history-preserving bisimulation. In R. Reischuk and M. Morvan, editors, *Proceedings of STACS'97*, volume 1200 of *LNCS*, pages 413–425. Springer, 1997.

25 Mogens Nielsen and Christian Clausen. Games and logics for a noninterleaving bisimulation. *Nordic Journal of Computing*, 2(2):221–249, 1995.

26 Martin Otto. Two variable first-order logic over ordered domains. *Journal of Symbolic Logic*, 66:685–702, 1999.

27 Tommaso Padoan. Relating some logics for true concurrency. In Alessandro Aldini and Marco Bernardo, editors, *Proceedings of ICTCS '18*, volume 2243 of *CEUR Workshop Proceedings*, pages 242–253. CEUR-WS.org, 2018.

28 Tommaso Padoan. *Tableaux, Automata and Games for True Concurrency Properties*. PhD thesis, University of Padua, 2019.

29 Wojciech Penczek. Branching time and partial order in temporal logics. In *Time and Logic: A Computational Approach*, pages 179–228. UCL Press, 1995.

30 I. Phillips and I. Ulidowski. Event identifier logic. *Mathematical Structures in Computer Science*, 24(2):1–51, 2014. doi:10.1017/S0960129513000510.

31 Jean Pichon-Pharabod and Peter Sewell. A concurrency semantics for relaxed atomics that permits optimisation and avoids thin-air executions. In R. Bodík and R. Majumdar, editors, *Proceedings of POPL'16*, pages 622–633. ACM, 2016.

32 S. Pinchinat, F. Laroussinie, and Ph. Schnoebelen. Logical characterization of truly concurrent bisimulation. Technical Report 114, LIFIA-IMAG, Grenoble, 1994.

33 Christian Prisacariu. Higher dimensional modal logic. *CoRR*, abs/1405.4100, 2014. arXiv:1405.4100.

34 Alexander M. Rabinovich and Boris A. Trakhtenbrot. Behaviour structures and nets. *Fundamenta Informaticae*, 11:357–404, 1988.

35 Wieslaw Szwast and Lidia Tendera. The guarded fragment with transitive guards. *Annals of Pure and Applied Logic*, 128(1):227–276, 2004. doi:10.1016/j.apal.2004.01.003.

36 P. S. Thiagarajan. Regular event structures and finite Petri nets: A conjecture. In W. Brauer, H. Ehrig, J. Karhumäki, and A. Salomaa, editors, *Formal and Natural Computing – Essays Dedicated to Grzegorz Rozenberg [on occasion of his 60th birthday]*, volume 2300 of *LNCS*, pages 244–256. Springer, 2002.

37 Robert J. van Glabbeek and Ursula Goltz. Refinement of actions and equivalence notions for concurrent systems. *Acta Informatica*, 37(4/5):229–327, 2001.

38 Walter Vogler. Deciding history preserving bisimilarity. In J. Albert, B. Monien, and M. Rodríguez-Artalejo, editors, *Proceedings of ICALP'91*, volume 510 of *LNCS*, pages 495–505. Springer, 1991.

39 Hao Wang. Proving theorems by pattern recognition – I. *Communications of the ACM*, 3(4):220–234, 1960. doi:10.1145/367177.367224.

40 Hao Wang. Proving theorems by battern recognition – II. *The Bell System Technical Journal*, 40(1):1–41, 1961. doi:10.1002/j.1538-7305.1961.tb03975.x.

41 Glynn Winskel. Event Structures. In W. Brauer, W. Reisig, and G. Rozenberg, editors, *Petri Nets: Applications and Relationships to Other Models of Concurrency*, volume 255 of *LNCS*, pages 325–392. Springer, 1987.

42 Glynn Winskel. Events, causality and symmetry. *Computer Journal*, 54(1):42–57, 2011.