Automata for True Concurrency Properties

Paolo Baldan, Tommaso Padoan

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

Abstract. We present an automata-theoretic framework for the model checking of true concurrency properties. These are specified in a fixpoint logic, corresponding to history-preserving bisimilarity, capable of describing events in computations and their dependencies. The models of the logic are event structures or any formalism which can be given a causal semantics, like Petri nets. Given a formula and an event structure satisfying suitable regularity conditions we show how to construct a parity tree automaton whose language is non-empty if and only if the event structure satisfies the formula. The automaton, due to the nature of event structure models, is usually infinite. We discuss how it can be quotiented to an equivalent finite automaton, where emptiness can be checked effectively. In order to show the applicability of the approach, we discuss how it instantiates to finite safe Petri nets. As a proof of concept we provide a model checking tool implementing the technique.

1 Introduction

Behavioural logics with the corresponding verification techniques are a cornerstone of automated verification. For concurrent and distributed systems, so called true concurrent models can be an appropriate choice, since they describe not only the possible steps in the evolution of the system but also their causal dependencies. A widely used foundational model in this class 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. A survey on the use of such causal models can be found in [2]. Recently they have been used in the study of concurrency in weak memory models [3,4], for process mining and differencing [5], in the study of atomicity [6] and of information flow [7] properties.

Operational models can be abstracted by considering true concurrent equivalences that range from hereditary history preserving bisimilarity to the coarser pomset and step equivalences (see, e.g., [8]) and behavioural logics expressing causal properties (see, e.g., [9,10,11,12,13,14] for a necessarily partial list and [15,16,17,18,19] for some related verification techniques).

Event-based logics have been recently introduced [20,21], capable of uniformly characterising the equivalences in the true concurrent spectrum. Their formulae include variables which are bound to events in computations and describe their dependencies. While the relation between operational models, be-

havioural equivalences and event-based true concurrent logics is well understood, the corresponding model checking problem has received limited attention.

We focus on the logic referred to as $\mathcal{L}_{hp}$ in [20], corresponding to a classical equivalence in the spectrum, i.e., history preserving (hp-)bisimilarity [22,23,24].

Decidability of model checking 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 [25]. In a recent paper [26] we proved the decidability of the problem for the alternation free fragment of the logic $\mathcal{L}_{hp}$ over a class of event structures satisfying a suitable regularity condition [27] referred to as strong regularity. The proof relies on a tableau-based model checking procedure. Despite the infiniteness of the model, a suitable stop condition can be identified, ensuring that a successful finite tableau can be generated if and only if the formula is satisfied by the model.

Besides the limitation to the alternation free fragment of $\mathcal{L}_{hp}$, a shortcoming of the approach is that a direct implementation of the procedure can be extremely inefficient. Roughly speaking, the problem is that in the search of a successful tableau, branches which are, in some sense, equivalent are explored several times.

In this paper we devise an automata-theoretic technique, in the style of [28], for model checking $\mathcal{L}_{hp}$ that works for the full logic, without constraints on the alternation depth. Besides providing an alternative approach for model-checking $\mathcal{L}_{hp}$, amenable of a more efficient implementation, this generalises the decidability result of [26] to the full logic $\mathcal{L}_{hp}$. Given a formula in $\mathcal{L}_{hp}$ and a strongly regular event structure, the procedure generates a parity tree automaton. Satisfiability is reduced to emptiness in the sense that the event structure satisfies the formula if and only if the automaton accepts a non-empty language.

The result is not directly usable for practical purposes since the automaton is infinite for any non-trivial event structure. However an equivalence on states can be defined such that the quotiented automaton accepts the same language as the original one. Whenever such equivalence is of finite index the quotiented automaton is finite, so that satisfaction of the formula can be checked effectively on the quotient. We show that for all strongly regular event structures a canonical equivalence always exists that is of finite index.

The procedure is developed abstractly on event structures. A concrete algorithm on some formalism requires the effectiveness of the chosen equivalence on states. We develop a concrete instantiation of the algorithm on finite safe Petri nets. It is implemented in a tool, wishfully called *True concurrency workbench* (TCWB), written in Haskell. Roughly, the search of an accepting run in the automaton can be seen as an optimisation of the procedure for building a successful tableau in [26] where the graph structure underlying the automaton helps in the reuse of the information discovered. Some tests reveal that the TCWB is way more efficient than the direct implementation of the tableau-based procedure (which could not manage most of the examples in the TCWB repository).

The rest of the paper is structured as follows. In § 2 we review event structures, strong regularity and the logic $\mathcal{L}_{hp}$ of interest in the paper. In § 3 we introduce (infinite state) parity tree automata and we show how the model checking

problem for $\mathcal{L}_{hp}$ on strongly regular PES can be reduced to the non-emptiness of the language of such automata. In § 4 we discuss the instantiation of the approach to Petri nets. Finally, in § 5 we discuss some related work and outline directions of future research. Due to space limitations, proofs are only sketched.

2 Event Structures and True Concurrent Logic

We introduce prime event structures [1] and the subclass of strongly regular event structures on which our model checking approach will be developed. Then we present the logic for true concurrency of interest in the paper.

2.1 Prime Event Structures and Regularity

Throughout the paper $\mathbb{E}$ is a fixed countable set of events, Λ a finite set of labels ranged over by $\mathsf{a}, \mathsf{b}, \mathsf{c} \ldots$ and $\lambda : \mathbb{E} \to \Lambda$ a labelling function.

Definition 1 (prime event structure). *A (Λ-labelled)* prime event structure *(*PES*) is a tuple $\mathcal{E} = \langle E, \leq, \# \rangle$, where $E \subseteq \mathbb{E}$ is the set of* events *and $\leq$, $\#$ are binary relations on E, called* causality *and* conflict *respectively, such that: 1. $\leq$ is a partial order and $\lceil e \rceil = \{e' \in E \mid e' \leq e\}$ is finite for all $e \in E$; 2. $\#$ is irreflexive, symmetric and inherited along $\leq$, i.e., for all $e, e', e'' \in E$, if $e \# e' \leq e''$ then $e \# e''$.*

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

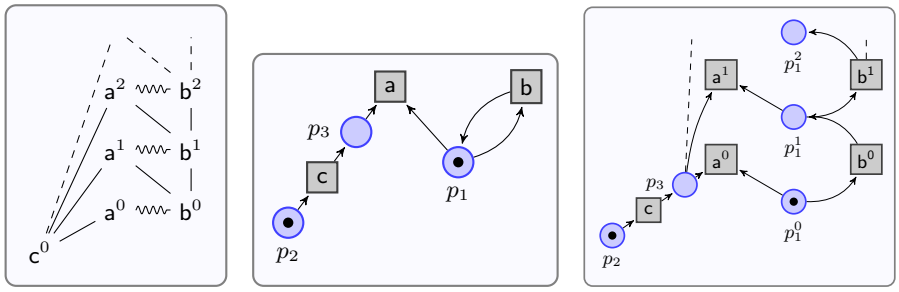
In the following, we will assume that the components of a PES $\mathcal{E}$ are named as in the definition above, possibly with subscripts. The concept of concurrent computation for PESs is captured by the notion of configuration.

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 $e \in C$). We denote by $\mathcal{C}(\mathcal{E})$ the set of configurations of $\mathcal{E}$.*

The evolution of a PES can be represented by a transition system over configurations, with the empty configuration as initial state.

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 $en(C) = \{e \in E \mid C \xrightarrow{e} C'\}$. 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. In addition, they report its label $\lambda(e)$, a subset of causes X and a set of events $Y \subseteq C$ concurrent with e. When



(a) $\mathcal{E}_{\mathcal{N}}$ (b) $\mathcal{N}$ (c) $\mathcal{U}(\mathcal{N})$

Fig. 1. (a) A PES $\mathcal{E}_{\mathcal{N}}$ associated with the net $\mathcal{N}$ in (b) via its unfolding (c).

X or Y are empty they are normally often, i.e., e.g., we write $C \xrightarrow{X<e}_{\lambda(e)} C'$ for $C \xrightarrow{\emptyset<e}_{\lambda(e)} C'$ and $C \xrightarrow{e}_{\lambda(e)} C'$ for $C \xrightarrow{\emptyset,\overline{\emptyset}<e}_{\lambda(e)} C'$.

The PES modelling a non-trivial system is normally infinite. We will work on a subclass identified by finitarity requirements on the possible substructures.

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 restriction of causality and conflict of $\mathcal{E}$. Intuitively, it represents the PES that remains to be executed after the computation expressed by C. Given $C \in \mathcal{C}(\mathcal{E})$ and $X \subseteq C$, we denote by $\mathcal{E}[C] \cup X$ the PES obtained from $\mathcal{E}[C]$ by adding the events in X with the causal dependencies they had in the original PES $\mathcal{E}$.

Definition 5 (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 [26] is obtained from the notion of regularity in [27], 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. We will later show in § 4 that the PESs associated with finite safe Petri nets, i.e., the regular trace PESs [27], are strongly regular.

A simple PES is depicted in Fig. 1a. 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 $\mathcal{E}_{\mathcal{N}}$, the events a^0 and b^0, labelled by a and b, respectively, are in conflict. Event c^0 causes the events a^i and it is concurrent with b^i for all $i \in \mathbb{N}$. It is an infinite PES associated with the Petri net $\mathcal{N}$ in Fig. 1b in a way that will be discussed in § 4.1, hence it is strongly regular by Corollary 1. It has five (equivalence classes of) residuals extended with an event from the past $\mathcal{E}_{\mathcal{N}}[\{\mathsf{b}^0\}] \cup \{\mathsf{b}^0\}$, $\mathcal{E}_{\mathcal{N}}[\{\mathsf{c}^0, \mathsf{b}^0\}] \cup \{\mathsf{b}^0\}$, $\mathcal{E}_{\mathcal{N}}[\{\mathsf{c}^0, \mathsf{a}^0\}] \cup \{\mathsf{c}^0\}$, $\mathcal{E}_{\mathcal{N}}[\{\mathsf{c}^0, \mathsf{a}^0\}] \cup \{\mathsf{a}^0\}$, and $\mathcal{E}_{\mathcal{N}}[\{\mathsf{c}^0, \mathsf{b}^0, \mathsf{a}^1\}] \cup \{\mathsf{b}^0\}$.

2.2 True Concurrent Logic

The logic of interest for this paper, originally defined in [20], is a Hennessy-Milner style logic that allows one to specify the dependencies (causality and concurrency) between events in computation.

Logic 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 a corresponding boldface letter $\mathbf{x}$ and, abusing the notation, tuples will be often used as sets. The logic includes diamond and box modalities. The formula $\langle\!\langle \mathbf{x}, \overline{\mathbf{y}} < \mathsf{a}\, z \rangle\!\rangle \varphi$ holds in a configuration when an a-labelled event e is enabled which causally depends on the events bound to $\mathbf{x}$ and is concurrent with those in $\mathbf{y}$. Event e is executed and then the formula φ must hold, with e bound to variable z. Dually, $[\![\mathbf{x}, \overline{\mathbf{y}} < \mathsf{a}\, z]\!] \varphi$ is satisfied when all a-labelled events causally dependent on $\mathbf{x}$ and concurrent with $\mathbf{y}$ bring to a configuration where φ holds.

For dealing with fixpoint operators we fix a denumerable set $\mathcal{X}^a$ of *abstract propositions*, ranged over by $X, Y, \ldots$. 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 (syntax). *The syntax of $\mathcal{L}_{hp}$ over the sets of event variables Var, abstract propositions $\mathcal{X}^a$ and labels Λ is defined as follows:*

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

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

For example, the formula $\varphi_1 = \langle\!\langle \mathsf{c}\, x \rangle\!\rangle (\langle\!\langle x < \mathsf{a}\, y \rangle\!\rangle \mathsf{T} \wedge \langle\!\langle \overline{x} < \mathsf{b}\, z \rangle\!\rangle \mathsf{T})$ 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. It is satisfied by $\mathcal{E}_{\mathcal{N}}$ in Fig. 1a. Instead $\varphi_2 = \langle\!\langle \mathsf{c}\, x \rangle\!\rangle (\langle\!\langle \overline{x} < \mathsf{a}\, y \rangle\!\rangle \mathsf{T} \wedge \langle\!\langle \overline{x} < \mathsf{b}\, z \rangle\!\rangle \mathsf{T})$ requiring both events to be concurrent would be false. Moving to infinite computations, consider $\varphi_3 = [\![\mathsf{b}\, x]\!] \nu Z(x).\langle\!\langle \mathsf{c}\, z \rangle\!\rangle \langle\!\langle \overline{z} < \mathsf{b}\, y \rangle\!\rangle \mathsf{T} \wedge [\![x < \mathsf{b}\, y]\!] Z(y)$, expressing that all non-empty causal chains of b-labelled events reach a state where it is possible to execute two concurrent events labelled c and b, respectively. Then φ_3 holds in $\mathcal{E}_{\mathcal{N}}$. Another formula satisfied by $\mathcal{E}_{\mathcal{N}}$ is $\varphi_4 = \langle\!\langle \mathsf{c}\, x \rangle\!\rangle \langle\!\langle \overline{x} < \mathsf{b}\, y \rangle\!\rangle \nu X(x, y).\langle\!\langle y, \overline{x} < \mathsf{b}\, z \rangle\!\rangle X(x, z)$ requiring the existence of an infinite causal chain of b-labelled events, concurrent with a c-labelled event.

The logic $\mathcal{L}_{hp}$ is interpreted over PESs. The satisfaction of a formula is defined with respect to a configuration C and a (total) function $\eta : Var \to E$, called

an *environment*, that binds free variables in φ 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 $\mathcal{C}(\mathcal{E}) \times Env_{\mathcal{E}}$. The semantics of $\mathcal{L}_{hp}$ also depends on a *proposition environment* $\pi : \mathcal{X} \to 2^{\mathcal{C}(\mathcal{E}) \times Env_{\mathcal{E}}}$ which provides an interpretation for propositions. In order to ensure that the semantics of a formula only depends on the events associated with its free variables and is independent on the naming of the variables, it is required that if $(C, \eta) \in \pi(X(\mathbf{x}))$ and $\eta'(\mathbf{y}) = \eta(\mathbf{x})$ pointwise, then $(C, \eta') \in \pi(X(\mathbf{y}))$. We denote by $PEnv_{\mathcal{E}}$ the set of proposition environments, ranged over by π.

We can now give the semantics of logic $\mathcal{L}_{hp}$. Given an event environment η and an event e we write $\eta[x \mapsto e]$ for 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.

Definition 7 (semantics). *Let $\mathcal{E}$ be a* PES. *The denotation of a formula φ in $\mathcal{L}_{hp}$ is given by the function $\{\!|\cdot|\!\}^{\mathcal{E}} : \mathcal{L}_{hp} \to PEnv_{\mathcal{E}} \to 2^{\mathcal{C}(\mathcal{E}) \times Env_{\mathcal{E}}}$ defined inductively as follows, where we write $\{\!|\varphi|\!\}^{\mathcal{E}}_{\pi}$ instead of $\{\!|\varphi|\!\}^{\mathcal{E}}(\pi)$:*

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

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

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

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

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

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 defined by $f_{\varphi, Z(\mathbf{x}), \pi}(S) = \{\!|\varphi|\!\}^{\mathcal{E}}_{\pi[Z(\mathbf{x}) \mapsto S]}$ and $gfp(f_{\varphi, Z(\mathbf{x}), \pi})$ (resp. $lfp(f_{\varphi, Z(\mathbf{x}), \pi})$) denotes the corresponding greatest (resp. least) fixpoint. We say that a PES *$\mathcal{E}$ satisfies a formula φ and write $\mathcal{E} \models \varphi$ if $(\emptyset, \eta) \in \{\!|\varphi|\!\}^{\mathcal{E}}_{\pi}$ for all environments η and π.*

The semantics of boolean operators 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 causally depends 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, once executed, it produces a new configuration $C' = C \cup \{e\}$ which, paired with the environment $\eta' = \eta[z \mapsto e]$, satisfies the formula φ. Dually, $[\![\mathbf{x}, \overline{\mathbf{y}} < \mathsf{a}\, z]\!] \varphi$ holds when all a-labelled events executable from C, caused by $\mathbf{x}$ and concurrent with $\mathbf{y}$ bring to a configuration where φ is satisfied.

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

3 Automata-based Model Checker

We introduce nondeterministic parity tree automata and we show how the model checking problem for $\mathcal{L}_{hp}$ on strongly regular PESs can be reduced to the non-emptiness of the language of such automata. The automaton naturally generated

from a PES and a formula has an infinite number of states. We discuss how the automaton can be quotiented to a finite one accepting the same language and thus potentially useful for model checking purposes.

3.1 Infinite Parity Tree Automata

Automata on infinite trees revealed to be a powerful tool to various problems in the setting of branching temporal logics. Here we focus on nondeterministic parity tree automata [29], with some (slightly) non-standard features. We work on k-trees (rather than on binary trees), a choice that will simplify the presentation, and we allow for possibly infinite state automata.

When automata are used for model checking purposes it is standard to restrict to unlabelled trees. A *k-bounded branching tree* or *k-tree*, for short, is a subset $\mathcal{T} \subseteq [1,k]^\star$, such that

1. $\mathcal{T}$ is prefix closed, i.e., if $wv \in \mathcal{T}$ then $w \in \mathcal{T}$
2. $w1 \in \mathcal{T}$ for all $w \in \mathcal{T}$
3. for all $i \in [2,k]$ if $wi \in \mathcal{T}$ then $w(i-1) \in \mathcal{T}$.

Elements of $\mathcal{T}$ are the nodes of the tree. The empty string ϵ corresponds to the root. A string of the form wi corresponds to the i-th child of w. Hence by (2) each branch is infinite and by (3) the presence of the i-th child implies the presence of the j-th children for $j \leq i$.

Definition 8 (nondeterministic parity automaton). *A k-bounded nondeterministic parity tree automaton (NPA) is a tuple $\mathcal{A} = \langle Q, \rightarrow, q_0, \mathcal{F}\rangle$ where Q is a set of states, $\rightarrow \subseteq Q \times \bigcup_{i=1}^{k} Q^k$ is the* transition relation*, $q_0 \in Q$ is the* initial state*, and $\mathcal{F} = (F_0, \ldots, F_h)$ is the* acceptance condition*, where $F_0, \ldots, F_h \subseteq Q$ are mutually disjoint subsets of states.*

Transitions are written as $q \rightarrow (q_1, \ldots, q_m)$ instead of $(q, (q_1, \ldots, q_m)) \in \rightarrow$.

Given a k-tree $\mathcal{T}$, a *run* of $\mathcal{A}$ on $\mathcal{T}$ is a labelling of $\mathcal{T}$ over the states $r : \mathcal{T} \rightarrow Q$ consistent with the transition relation, i.e., such that $r(\epsilon) = q_0$ and for all $u \in \mathcal{T}$, with m children, there is a transition $r(u) \rightarrow (r(u1), \ldots, r(um))$ in $\mathcal{A}$. A *path* in the run r is an infinite sequence of states $p = (q_0, q_1, \ldots)$ labelling a complete path from the root in the tree. It is called *accepting* if there exists an even number $l \in [0,h]$ such that the set $\{j \mid q_j \in F_l\}$ is infinite and the set $\{j \mid q_j \in \bigcup_{l<i\leq h} F_i\}$ is finite. The run r is *accepting* if all paths are accepting.

Definition 9 (language of an NPA). *Let $\mathcal{A}$ be an NPA. The language of $\mathcal{A}$, denoted by $L(\mathcal{A})$, consists of the trees $\mathcal{T}$ which admit an accepting run.*

Observe that for a k-bounded NPA, the language $L(\mathcal{A})$ is a set of k-trees.

The possibility of having an infinite number of states and the associated acceptance condition are somehow non-standard. However, it is easy to see that whenever an NPA is finite, the acceptance condition coincides with the standard one requiring a single state with maximal even priority to occur infinitely often.

Since NPAs are nondeterministic, different runs (possibly infinitely many) can exist for the same input tree. Still, the non-emptiness problem, also for our k-ary variant, is decidable when the number of states is finite (and solvable by a corresponding parity game [30]).

3.2 Infinite NPAs for Model Checking

We show how, given a PES and a closed formula in $\mathcal{L}_{hp}$, we can build an NPA in a way that, for strongly regular PESs, the satisfaction of φ in $\mathcal{E}$ reduces to the non-emptiness of the automaton language. The construction is inspired by that in [28] for the mu-calculus.

The acceptance condition for the automaton will refer to the fixpoint alternation in the formulae of $\mathcal{L}_{hp}$. We adapt a definition from [28]. A fixpoint formula $\alpha X(\mathbf{y}).\varphi'$, for $\alpha \in \{\nu, \mu\}$, is called an α-formula. Hereafter α ranges over $\{\nu, \mu\}$. Given an α-formula $\varphi = \alpha X(\mathbf{y}).\varphi'$, we say that a subformula ψ of φ is a *direct active subformula*, written $\psi \sqsubseteq_d \varphi$, if the abstract proposition X appears free in ψ. The transitive closure of $\sqsubseteq_d$ is a partial order and when $\psi \sqsubseteq_d^* \varphi$ we say that ψ is an *active subformula* of φ. We denote by $\mathsf{sf}(\varphi)$ the set of subformulae of a formula φ and by $\mathsf{sf}_\alpha(\varphi)$ the set of active α-subformulae.

The *alternation depth* of a formula φ in $\mathcal{L}_{hp}$, written $\mathsf{ad}(\varphi)$, is defined, for a ν-formula φ, as $\mathsf{ad}(\varphi) = \max\{1 + \mathsf{ad}(\psi) \mid \psi \in \mathsf{sf}_\mu(\varphi)\}$ and dually, for a μ-formula φ, as $\mathsf{ad}(\varphi) = \max\{1 + \mathsf{ad}(\psi) \mid \psi \in \mathsf{sf}_\nu(\varphi)\}$. For any other formula φ, $\mathsf{ad}(\varphi) = max\{\mathsf{ad}(\psi) \mid \psi \in \mathsf{sf}(\varphi) \setminus \{\varphi\}\}$. It is intended that $\max \emptyset = 0$. E.g., by the first clause above, the alternation depth of $\nu X(\mathbf{x}).\, \varphi$ is 0 in absence of active μ-subformulae.

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.

Hereafter, if X and X' are abstract propositions quantified in α-subformulae $\alpha X(\mathbf{x}).\, \varphi$ and $\alpha' X'(\mathbf{x}').\, \varphi'$, we will write $\mathsf{ad}(X)$ for $\mathsf{ad}(\alpha X(\mathbf{x}).\, \varphi)$ and $X \sqsubseteq_d X'$ for $\alpha X(\mathbf{x}).\, \varphi \sqsubseteq_d \alpha' X'(\mathbf{x}').\, \varphi'$. Moreover, given a PES $\mathcal{E}$, for a pair $(C, \eta) \in \mathcal{C}(\mathcal{E}) \times Env_{\mathcal{E}}$ and variables $\mathbf{x}$, $\mathbf{y}$, z, we define $(\mathbf{x}, \overline{\mathbf{y}} < \mathsf{a}z)$-successors of (C, η), as

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

We can now illustrate the construction of the NPA for a formula and a PES.

Definition 10 (NPA for a formula). *Let $\mathcal{E}$ be a bounded* PES *and let $\varphi \in \mathcal{L}_{hp}$ be a closed formula. The NPA for $\mathcal{E}$ and φ is $\mathcal{A}_{\mathcal{E},\varphi} = \langle Q, \to, q_0, \mathcal{F}\rangle$ defined as follows. The set of states $Q \subseteq \mathcal{C}(\mathcal{E}) \times Env_{\mathcal{E}} \times \mathsf{sf}(\varphi)$ is $Q = \{(C, \eta, \psi) \mid \eta(fv(\psi)) \subseteq C\}$. The initial state $q_0 = (\emptyset, \eta, \varphi)$, for some chosen $\eta \in Env_{\mathcal{E}}$. The transition relation is defined, for any state $q = (C, \eta, \psi) \in Q$, by:*

- *if $\psi = \mathsf{T}$ or $\psi = \mathsf{F}$, then $q \to (q)$;*
- *if $\psi = \psi_1 \wedge \psi_2$, then $q \to (q_1, q_2)$ where $q_i = (C, \eta, \psi_i)$, $i \in \{1, 2\}$;*
- *if $\psi = \psi_1 \vee \psi_2$, then $q \to (q_1)$ and $q \to (q_2)$ where $q_i = (C, \eta, \psi_i)$, $i \in \{1, 2\}$;*

– *if* $\psi = [\![\mathbf{x}, \overline{\mathbf{y}} < \mathsf{a}\, z]\!]\psi'$ *and* $\mathsf{Succ}^{\mathbf{x},\overline{\mathbf{y}}<\mathsf{a}z}(C,\eta) = \{(C_1,\eta_1),\ldots,(C_n,\eta_n)\} \neq \emptyset$ *then* $q \to (q_1,\ldots,q_n)$ *where* $q_i = (C_i,\eta_i,\psi')$ *for* $i \in [1,n]$*, otherwise* $q \to (q)$*;*
– *if* $\psi = \langle\!|\mathbf{x}, \overline{\mathbf{y}} < \mathsf{a}\, z|\!\rangle\psi'$ *and* $\mathsf{Succ}^{\mathbf{x},\overline{\mathbf{y}}<\mathsf{a}z}(C,\eta) = \{(C_1,\eta_1),\ldots,(C_n,\eta_n)\} \neq \emptyset$ *then* $q \to (q_i)$ *where* $q_i = (C_i,\eta_i,\psi')$ *for* $i \in [1,n]$*, otherwise* $q \to (q)$*;*
– *if* $\psi = \alpha X(\mathbf{x}).\psi'$ *then* $q \to (q')$ *where* $q' = (C,\eta,X(\mathbf{x}))$*;*
– *if* $\psi = X(\mathbf{y})$ *and* $\psi' \in \mathsf{sf}(\varphi)$ *is the unique subformula such that* $\psi' = \alpha X(\mathbf{x}).\psi''$ *then* $q \to (q')$ *where* $q' = (C,\eta[\mathbf{x}\mapsto\eta(\mathbf{y})],\psi'')$*.*

The acceptance condition is $\mathcal{F} = (F_0,\ldots,F_h)$ *where* $h = \mathsf{ad}(\varphi)+1$ *and the* F_i *are as follows. Consider* $A_0,\ldots,A_h \subseteq \mathsf{sf}(\varphi)$ *such that for* $i \in [0,h]$*, if* i *is even (odd) then* A_i *contains exactly all propositions quantified in* ν*-subformulae (*μ*-subformulae) with alternation depth* i *or* $i-1$*. Then* $F_0 = (\mathcal{C}(\mathcal{E}) \times Env_{\mathcal{E}} \times (A_0 \cup \{\mathsf{T}\})) \cup B$ *where* $B = \{(C,\eta,[\![\mathbf{x},\overline{\mathbf{y}} < \mathsf{a}\, z]\!]\psi) \mid \mathsf{Succ}^{\mathbf{x},\overline{\mathbf{y}}<\mathsf{a}z}(C,\eta) = \emptyset\}$ *is the set of all subformulae of* φ *in a context where they are trivially true, and* $F_i = \mathcal{C}(\mathcal{E}) \times Env_{\mathcal{E}} \times A_i$*, for* $i \in [1,h]$*.*

States of $\mathcal{A}_{\mathcal{E},\varphi}$ are triples (C,η,φ) consisting of a configuration C, an environment η and a subformula ψ of the original formula φ. The intuition is that a transition reduces the satisfaction of a formula in a state to that of subformulae in possibly updated states. It can just decompose the formula, as it happens for $\wedge$ or $\vee$, check the satisfaction of a modal operator, thus changing the state consequently, or unfold a fixpoint.

The automaton $\mathcal{A}_{\mathcal{E},\varphi}$ is bounded but normally infinite (whenever the PES $\mathcal{E}$ is infinite and the formula φ includes some non-trivial fixpoint).

We next show that for a strongly regular PES the satisfaction of the formula φ on the PES $\mathcal{E}$ reduces to the non-emptiness of the language of $\mathcal{A}_{\mathcal{E},\varphi}$.

Theorem 1 (model checking via non-emptiness). *Let* $\mathcal{E}$ *be a strongly regular* PES *and let* $\check{\varphi}$ *be a closed formula in* $\mathcal{L}_{hp}$*. Then* $L(\mathcal{A}_{\mathcal{E},\check{\varphi}}) \neq \emptyset$ *iff* $\mathcal{E} \models \check{\varphi}$*.*

We next provide an outline of the proof. A basic ingredient is an equivalence that can be defined on the NPA. As a first step we introduce a generalised notion of residual in which the relation with some selected events in the past is kept.

Definition 11 (pointed residual). *Given a* PES $\mathcal{E}$ *and a set* X*, a* X-pointed configuration *is a pair* $\langle C,\zeta\rangle$ *where* $C \in \mathcal{C}(\mathcal{E})$ *and* $\zeta : X \to C$ *is a function. We say that the* X*-pointed configurations* $\langle C,\zeta\rangle$*,* $\langle C',\zeta'\rangle$ *have isomorphic pointed residuals, written* $\mathcal{E}[\langle C,\zeta\rangle] \approx \mathcal{E}[\langle C',\zeta'\rangle]$ *if there is an isomorphism of* PESs $\iota : \mathcal{E}[C] \to \mathcal{E}[C']$ *such that for all* $x \in X$*,* $e \in \mathcal{E}[C]$ *we have* $\zeta(x) \leq e$ *iff* $\zeta'(x) \leq \iota(e)$*.*

Then two states are deemed equivalent if they involve the same subformula (up to renaming of the event variables) and the configurations, pointed by the free variables in the formulae, have isomorphic residuals. This resembles the notion of contextualised equivalence used on tableau judgments in [26].

Definition 12 (future equivalence). *Let* $\mathcal{E}$ *be a* PES*,* φ *be a formula and let* $q_i = (C_i,\eta_i,\psi_i)$*,* $i \in \{1,2\}$ *be two states of the NPA* $\mathcal{A}_{\mathcal{E},\varphi}$*. We say that* q_1 *and* q_2 *are* future equivalent*, written* $q_1 \approx_f q_2$*, if there exists a formula* ψ *and substitutions* $\sigma_i : fv(\psi) \to fv(\psi_i)$ *such that* $\psi\sigma_i = \psi_i$*, for* $i \in \{1,2\}$*, and the* $fv(\psi)$*-pointed configurations* $\langle C_i,\eta_i \circ \sigma_i\rangle$ *have isomorphic pointed residuals.*

It can be shown that, given $q_i = (C_i, \eta_i, \psi_i)$, $i \in \{1, 2\}$ as above, for all proposition environments π (satisfying a technical property of saturation) we have that $(C_1, \eta_1) \in \{\!|\psi_1|\!\}^{\mathcal{E}}_{\pi}$ if and only if $(C_2, \eta_2) \in \{\!|\psi_2|\!\}^{\mathcal{E}}_{\pi}$. Additionally, using strong regularity, one can prove that the the semantics of fixpoint formulae is properly captured by finite approximants and that equivalence $\approx_f$ is of finite index. These are fundamental building bricks in the proof of Theorem 1 which, roughly, proceeds as follows.

Assume that the the language $L(\mathcal{A}_{\mathcal{E},\varphi}) \neq \emptyset$. Then there is an accepting run r over some k-tree $\mathcal{T}$. Since φ is finite, in each infinite path there are infinitely many states $q_{i_h} = (C_{i_h}, \eta_{i_h}, \psi_{i_h})$ where ψ_{i_h} is the same subformula, up to renaming. Since $\approx_f$ is of finite index, infinitely many such states are equivalent. Then one deduces that, for some h, the subformula ψ_{i_h} is satisfied in (C_{i_h}, η_{i_h}). For fixpoint subformulae, this requires to show that, since the run is accepting, the subformula of maximal alternation depth that repeats infinitely often is a ν-formula and use the fact that, as mentioned before, its semantics can be finitely approximated. Then, by a form of backward soundness of the transitions, we get that all the nodes, including the root, contain formulae which are satisfied.

For the converse implication, assume that $\mathcal{E} \models \varphi$. Starting from the initial state $q_0 = (\emptyset, \eta, \varphi)$ where the formula is satisfied, and using the automaton transitions, we can build a k-tree $\mathcal{T}$ and a run where for each state (C', η', ψ) the subformula ψ is satisfied in (C', η') and such run can be proved to be accepting.

3.3 Quotienting the Automaton

In order to have an effective procedure for checking the satisfaction of a formula we need to build a suitable quotient of the NPA, with respect to an equivalence which preserves emptiness. A simple but important observation is that it is sufficient to require that the equivalence is a bisimulation in the following sense. An analogous notion is studied in [31] in the setting of nondeterministic tree automata over finite trees.

Definition 13 (bisimulation). *Given an NPA $\mathcal{A}$, a symmetric relation $R \subseteq Q \times Q$ over the set of states is a bisimulation if for all $(q, q') \in R$*

1. *for all $i \in [0, h]$, $q \in F_i \iff q' \in F_i$;*
2. *if $q \to (q_1, \ldots, q_m)$ then $q' \to (q'_1, \ldots, q'_m)$ with $(q_i, q'_i) \in R$ for $i \in [1, m]$.*

Given an NPA $\mathcal{A}$ and an equivalence $\equiv$ on the set of states which is a bisimulation, we define the quotient as $\mathcal{A}_{/\equiv} = \langle Q_{/\equiv}, \to_{/\equiv}, [q_0]_{\equiv}, \mathcal{F}_{/\equiv} \rangle$ where $[q]_{\equiv} \to_{/\equiv} ([q_1]_{\equiv}, \ldots, [q_m]_{\equiv})$ if $q \to (q_1, \ldots, q_m)$ and $\mathcal{F}_{/\equiv} = (F_{0/\equiv}, \ldots, F_{h/\equiv})$. An NPA and its quotient accept exactly the same language.

Theorem 2 (language preservation). *Let $\mathcal{A}$ be an NPA and let $\equiv$ be an equivalence on the set of states which is a bisimulation. Then $L(\mathcal{A}_{/\equiv}) = L(\mathcal{A})$.*

When $\equiv$ is of finite index, the quotient $\mathcal{A}_{\mathcal{E},\varphi_{/\equiv}}$ is finite and, exploiting Theorems 1 and 2, we can verify whether $\mathcal{E} \models \varphi$ by checking the emptiness of the

language accepted by $\mathcal{A}_{\mathcal{E},\varphi/\equiv}$. Clearly a concrete algorithm will not first generate the infinite state NPA and then take the quotient, but it rather performs the quotient on the fly: whenever a new state would be equivalent to one already generated, the transition loops back to the existing state.

Whenever $\mathcal{E}$ is strongly regular, the future equivalence on states (see Definition 12) provides a bisimulation equivalence of finite index over $\mathcal{A}_{\mathcal{E},\varphi}$.

Lemma 1 ($\approx_f$ is a bisimulation). *Let $\mathcal{E}$ be a strongly regular* PES *and let φ be a closed formula in $\mathcal{L}_{hp}$. Then the future equivalence $\approx_f$ on $\mathcal{A}_{\mathcal{E},\varphi}$ is a bisimulation and it is of finite index.*

An obstacle towards the use of the quotiented NPA for model checking purposes is the fact that the future equivalence could be hard to compute (or even undecidable). In order to make the construction effective we need a decidable bisimulation equivalence on the NPA and the effectiveness of the set of successors of a state. This is further discussed in the next section.

4 Model Checking Petri Nets

We show how the model checking approach outlined before can be instantiated on finite safe Petri nets, a classical model of concurrency and distribution [32], by identifying a suitable effective bisimulation equivalence on the NPA.

4.1 Petri Nets and their Event Structure Semantics

A *Petri net* is a tuple $\mathcal{N} = (P, T, F, M_0)$ where P, T are disjoint sets of *places* and *transitions*, respectively, $F : (P \times T) \cup (T \times P) \to \{0, 1\}$ is the *flow function*, and M_0 is the initial marking, i.e., the initial state of the net. We assume that the set of transitions is a subset of a fixed set $\mathbb{T}$ with a labelling $\lambda_N : \mathbb{T} \to \Lambda$.

A *marking* of $\mathcal{N}$ is a function $M : P \to \mathbb{N}$, indicating for each place the number of tokens in the place. A transition $t \in T$ is *enabled* at a marking M if $M(p) \geq F(p, t)$ for all $p \in P$. In this case it can be *fired* leading to a new marking M' defined by $M'(p) = M(p) + F(t, p) - F(p, t)$ for all places $p \in P$. This is written $M[t\rangle M'$. We denote by $\mathcal{R}(\mathcal{N})$ the set of markings reachable in $\mathcal{N}$ via a sequence of firings starting from the initial marking. We say that a marking M is *coverable* if there exists $M' \in \mathcal{R}(\mathcal{N})$ such that $M \leq M'$, pointwise. A net $\mathcal{N}$ is *safe* if for every reachable marking $M \in \mathcal{R}(\mathcal{N})$ and all $p \in P$ we have $M(p) \leq 1$. Hereafter we will consider only safe nets. Hence markings will be often confused with the corresponding subset of places $\{p \mid M(p) = 1\} \subseteq P$. For $x \in P \cup T$ the *pre-set* and *post-set* are defined ${}^\bullet x = \{y \in P \cup T \mid F(y, x) = 1\}$ and $x^\bullet = \{y \in P \cup T \mid F(x, y) = 1\}$ respectively.

An example of Petri net can be found in Fig. 1b. Graphically places and transitions are drawn as circles and rectangles, respectively, while the flow function is rendered by means of directed arcs connecting places and transitions. Markings are represented by inserting tokens (black dots) in the corresponding places.

The concurrent behaviour of a Petri net can be represented by its unfolding $\mathcal{U}(\mathcal{N})$, an acyclic net constructed inductively starting from the initial marking of $\mathcal{N}$ and then adding, at each step, an occurrence of each enabled transition.

Definition 14 (unfolding). *Let $\mathcal{N} = (P, T, F, m_0)$ be a safe net. Define the net $U^{(0)} = (P^{(0)}, T^{(0)}, F^{(0)})$ as $T^{(0)} = \emptyset$, $P^{(0)} = \{(p, \bot) \mid p \in m_0\}$ and $F^{(0)} = \emptyset$, where $\bot$ is an element not belonging to P, T or F. The unfolding is the least net $\mathcal{U}(\mathcal{N}) = (P^{(\omega)}, T^{(\omega)}, F^{(\omega)})$ containing $U^{(0)}$ and such that*

- *if $t \in T$, the set of places $X \subseteq P^{(\omega)}$ is coverable and $\pi_1(X) = {}^\bullet t$, then $e = (t, X) \in T^{(\omega)}$;*
- *for any $e = (t, X) \in T^{(\omega)}$, the set $Z = \{(p, e) \mid p \in \pi_1(e)^\bullet\} \subseteq P^{(\omega)}$ where $\pi_1(u, v) = u$; moreover ${}^\bullet e = X$ and $e^\bullet = Z$.*

Places and transitions in the unfolding represent tokens and firing of transitions, respectively, of the original net. The projection π_1 over the first component maps places and transitions of the unfolding to the corresponding items of the original net $\mathcal{N}$. The initial marking is implicitly identified as the set of minimal places. For historical reasons transitions and places in the unfolding are also called *events* and *conditions*, respectively.

One can define *causality* $\leq_N$ over the unfolding as the transitive closure of the flow relation. *Conflict* is the relation $e \# e'$ if ${}^\bullet e \cap {}^\bullet e' \neq \emptyset$, inherited along causality. The events $T^{(\omega)}$ of the unfolding of a finite safe net, endowed with causality and conflict, form a PES, denoted $\mathcal{E}(\mathcal{N})$. The transitions of a configuration $C \in \mathcal{C}(\mathcal{E}(\mathcal{N}))$ can be fired in any order compatible with causality, producing a marking $C^\circ = (P^{(0)} \cup \bigcup_{t \in C} t^\bullet) \setminus (\bigcup_{t \in C} {}^\bullet t)$ in $\mathcal{U}(\mathcal{N})$; in turn, this corresponds to a reachable marking of $\mathcal{N}$ given by $\mathsf{M}(C) = \pi_1(C^\circ)$. As an example, the unfolding $\mathcal{U}(\mathcal{N})$ of the running example net $\mathcal{N}$ and the corresponding PES can be found in Figs. 1c and 1a.

4.2 Automata Model Checking for Petri Nets

The PES associated with a safe Petri net is known to be regular [27]. We next prove that it is also strongly regular and thus we can apply the theory developed so far for model checking $\mathcal{L}_{hp}$ over safe Petri nets.

Let $\mathcal{N} = \langle S, T, F, M_0 \rangle$ be a safe Petri net. A basic observation is that the residual of the PES $\mathcal{E}(\mathcal{N})$ with respect to a configuration $C \in \mathcal{C}(\mathcal{E}(\mathcal{N}))$ is uniquely determined by the marking produced by C. This correspondence can be extended to pointed configurations by considering markings which additionally record, for the events of interest in the past, the places in the marking which are caused by such events. This motivates the definition below.

Definition 15 (pointed marking). *Let $\mathcal{N} = \langle S, T, F, M_0 \rangle$ be a safe Petri net. Given a set X, a X-*pointed marking *is a pair $\langle M, r \rangle$ with $r : X \to 2^M$.*

A X-pointed configuration $\langle C, \zeta \rangle$ induces an X-pointed marking $\mathsf{M}(\langle C, \zeta \rangle) = \langle \mathsf{M}(C), r \rangle$ where $r(x) = \{\pi_1(b) \mid b \in C^\circ \ \wedge \ \zeta(x) < b\}$. Pointed configurations producing the same pointed marking have isomorphic pointed residuals.

Proposition 1 (pointed markings vs residuals). *Let $\mathcal{N} = \langle S, T, F, M_0 \rangle$ be a safe Petri net. Given a set X and two X-pointed configurations $\langle C_1, \zeta_1 \rangle$, $\langle C_2, \zeta_2 \rangle$ in $\mathcal{U}(\mathcal{N})$, if $\mathsf{M}(\langle C_1, \zeta_1 \rangle) = \mathsf{M}(\langle C_2, \zeta_2 \rangle)$ then $\mathcal{E}(\mathcal{N})[\langle C_1, \zeta_1 \rangle] \approx \mathcal{E}(\mathcal{N})[\langle C_2, \zeta_2 \rangle]$.*

By the previous result the PES associated with a finite safe Petri net is strongly regular. Indeed, the number of residuals of X-pointed configurations, up to isomorphism, by Proposition 1, is smaller than the number of X-pointed markings, which is clearly finite since the net is safe.

Corollary 1 (strong regularity). *Let $\mathcal{N}$ be finite safe Petri net. Then the corresponding* PES *$\mathcal{E}(\mathcal{N})$ is strongly regular.*

In order to instantiate the model checking framework to finite safe Petri nets, the idea is to take an equivalence over the infinite NPA by abstracting the (pointed) configurations associated with its states to pointed markings.

Definition 16 (pointed-marking equivalence on NPA). *Let $\mathcal{N}$ be a finite safe Petri net and let φ be a closed formula in $\mathcal{L}_{hp}$. Two states q_1, q_2 in the NPA $\mathcal{A}_{\mathcal{E}(\mathcal{N}),\varphi}$ are* pointed-marking equivalent*, written $q_1 \approx_m q_2$, if $q_i = \langle C_i, \eta_i, \psi \rangle$, $i \in \{1, 2\}$, for some $\psi \in \mathsf{sf}(\varphi)$ and $\mathsf{M}(\langle C_1, \eta_{1|fv(\psi)} \rangle) = \mathsf{M}(\langle C_2, \eta_{2|fv(\psi)} \rangle)$.*

Using Proposition 1 we can immediately prove that $\approx_m$ refines $\approx_f$. Moreover we can show that $\approx_m$ is a bisimulation in the sense of Definition 13.

Proposition 2 (marking equivalence is a bisimulation). *Let $\mathcal{N}$ be a finite safe Petri net and let φ be a closed formula in $\mathcal{L}_{hp}$. The equivalence $\approx_m$ on the automaton $\mathcal{A}_{\mathcal{E}(\mathcal{N}),\varphi}$ is a bisimulation and it is of finite index.*

Relying on Propositions 1 and 2 we provide an explicit construction of the quotient automaton $\mathcal{A}_{\mathcal{E}(\mathcal{N}),\varphi/\approx_m}$. We introduce a convenient notation for transitions between pointed markings. Given the variables $\mathbf{x}$, $\mathbf{y}$, a set X such that $\mathbf{x} \cup \mathbf{y} \subseteq X$ and an X-pointed marking $\langle M, r \rangle$, we write $\langle M, r \rangle \xrightarrow{\mathbf{x}, \overline{\mathbf{y}} < t}_{\mathsf{a}, z} \langle M', r' \rangle$ if $M[t\rangle M'$, $\lambda_N(t) = \mathsf{a}$, for all $x \in \mathbf{x}$ we have $r(x) \cap {}^\bullet t \neq \emptyset$ and for all $y \in \mathbf{y}$ it holds $r(y) \cap {}^\bullet t = \emptyset$ and r' is defined by $r'(z) = t^\bullet$ and $r'(w) = (r(w) \cap M') \cup \{s \mid r(w) \cap {}^\bullet t \neq \emptyset \ \wedge \ s \in t^\bullet\}$, for $w \neq z$. In words, from the pointed marking $\langle M, r \rangle$ transition t is fired and "pointed" by variable z. Transition t is required to consume tokens caused by $\mathbf{x}$ and not to consume tokens caused by $\mathbf{y}$, in order to be itself caused by $\mathbf{x}$ and independent from $\mathbf{y}$. After the firing, variables which were causes of some $p \in {}^\bullet t$ become causes of the places in $t^\bullet$ and, clearly, z causes $t^\bullet$.

Construction 1 (quotient NPA). *Let $\mathcal{N}$ be a finite safe Petri net and let $\varphi \in \mathcal{L}_{hp}$ be a closed formula. The quotient NPA $\mathcal{A}_{\mathcal{E}(\mathcal{N}),\varphi/\approx_m}$ is defined as follows. The set of states $Q = \{(M, r, \psi) \mid M \in \mathcal{R}(\mathcal{N}) \ \wedge \ r : fv(\psi) \to 2^M \ \wedge \ \psi \in \mathsf{sf}(\varphi)\}$. The initial state $q_0 = (M_0, \emptyset, \varphi)$. The transition relation is defined, for any state $q = (M, r, \psi) \in Q$, by:*

- *if $\psi = \mathsf{T}$ or $\psi = \mathsf{F}$, then $q \to (q)$*
- *if $\psi = \psi_1 \wedge \psi_2$, then $q \to (q_1, q_2)$ where $q_i = (M, r, \psi_i)$, $i \in \{1, 2\}$*

– *if* $\psi = \psi_1 \vee \psi_2$, *then* $q \to (q_1)$ *and* $q \to (q_2)$ *where* $q_i = (M, r, \psi_i)$, $i \in \{1, 2\}$
– *if* $\psi = [\![\mathbf{x}, \overline{\mathbf{y}} < \mathsf{a}\, z]\!]\psi'$, *let* $S = \{(M', r'_{|fv(\psi')}) \mid \langle M, r\rangle \xrightarrow{\mathbf{x},\overline{\mathbf{y}} < t}_{a,z} \langle M', r'\rangle\}$; *if* $S = \{(M_1, r_1), \ldots, (M_n, r_n)\} \neq \emptyset$ *then* $q \to (q_1, \ldots, q_n)$ *where* $q_i = (M_i, r_i, \psi')$ *for* $i \in [1, n]$, *otherwise* $q \to (q)$;
– *if* $\psi = \langle\!\langle \mathbf{x}, \overline{\mathbf{y}} < \mathsf{a}\, z \rangle\!\rangle \psi'$, *let* $S = \{(M', r'_{|fv(\psi')}) \mid \langle M, r\rangle \xrightarrow{\mathbf{x},\overline{\mathbf{y}} < t}_{a,z} \langle M', r'\rangle\}$; *if* $S = \{(M_1, r_1), \ldots, (M_n, r_n)\} \neq \emptyset$ *then* $q \to (q_i)$ *where* $q_i = (M_i, r_i, \psi')$ *for* $i \in [1, n]$, *otherwise* $q \to (q)$;
– *if* $\psi = \alpha X(\mathbf{x}).\psi'$ *then* $q \to (q')$ *where* $q' = (M, r, X(\mathbf{x}))$;
– *if* $\psi = X(\mathbf{y})$ *and* $\psi' \in \mathsf{sf}(\varphi)$ *is the subformula such that* $\psi' = \alpha X(\mathbf{x}).\psi''$ *then* $q \to (q')$ *where* $q' = (M, r[\mathbf{x} \mapsto r(\mathbf{y})], \psi'')$.

The acceptance condition is as in Definition 10.

4.3 A Prototype Tool

The algorithm for model checking Petri nets outlined before is implemented in the prototype tool TCWB (*True Concurrency Workbench*) [33], written in Haskell. The tool inputs a safe Petri net $\mathcal{N}$ and a closed formula φ of $\mathcal{L}_{hp}$ and outputs the truth value of the formula on the initial marking of $\mathcal{N}$. The algorithm builds the quotient NPA $\mathcal{A}_{\mathcal{E}(\mathcal{N}),\varphi\,/\approx_m}$ "on demand", i.e., the states of the automaton are generated when they are explored in the search of an accepting run. A path is recognised as successful when it includes a loop where a $\sqsubseteq_d^*$-maximal subformula is T, a $[\![\,]\!]$-subformula or a ν-subformula. In this way only the fragment of $\mathcal{A}_{\mathcal{E}(\mathcal{N}),\varphi\,/\approx_m}$ relevant to decide the satisfaction of φ is built.

Given a net $\mathcal{N} = (P, T, F, M_0)$ and a formula φ, the number of states in the quotient automaton $\mathcal{A}_{\mathcal{E}(\mathcal{N}),\varphi\,/\approx_m}$ can be bounded as follows. Recall that a state consists of a triple (M, r, ψ) where $\psi \in \mathsf{sf}(\varphi)$, M is a reachable marking and $r : fv(\psi) \to 2^M$ is a function. This leads to an upper bound $O(|\mathsf{sf}(\varphi)| \cdot |\mathcal{R}(\mathcal{N})| \cdot 2^{|P| \cdot v})$, where $v = max\{|fv(\psi)| : \psi \in \mathsf{sf}(\varphi)\}$ is the largest number of event variables appearing free in a subformula of φ. In turn, since $|\mathcal{R}(\mathcal{N})| \leq 2^{|P|}$, this is bounded by $O(|\mathsf{sf}(\varphi)| \cdot 2^{|P| \cdot (v+1)})$. The size of the automaton is thus exponential in the size of the net and linear in the size of the formula. Moving from the interleaving fragment of the logic (where $v = 0$) to formulae capable of expressing true concurrent properties thus causes an exponential blow up. However, note that the worst case scenario requires all transitions to be related by causality and concurrency to all places in any possible way, something that should be quite unlikely in practice. Indeed, despite the fact that the tool is very preliminary and more tweaks and optimisations could improve its efficiency, for the practical tests we performed the execution time seems to be typically well below than the theoretical worst case upper bound.

5 Conclusions

We introduced an automata-theoretic framework for the model checking of the logic for true concurrency $\mathcal{L}_{hp}$, representing the logical counterpart of a classical

true concurrent equivalence, i.e., history preserving bisimilarity. The approach is developed abstractly for strongly regular PESs, that include regular trace PESs. A concrete model-checking procedure requires the identification of an effective bisimulation equivalence for the construction of the quotient automaton. We showed how this can be done for finite safe Petri nets. The technique is implemented in a proof-of-concept tool.

We proved that the class of regular trace PESs is included in that of strongly regular PESs which in turn is included in the class of regular PESs. The precise relation of strongly regular PESs with the other two classes is still unclear and interesting in view of [34] that recently showed that regular trace PESs are strictly included in regular PESs, disproving Thiagarajan's conjecture.

Several other papers deal with model checking for logics on event structures. In [35] a technique is proposed for model checking a CTL-style logic with modalities for immediate causality and conflict on a subclass of PESs. The logic is quite different from ours as formulae are satisfied by single events, the idea being that an event, with its causes, represents the local state of a component. The procedure involves the construction of a finite representation of the PES associated with a program which has some conceptual relation with our quotienting phase. In [19] the author shows that first order logic and Monadic Trace Logic (MTL), a restricted form of monadic second order (MSO) logic are decidable on regular trace event structures. The possibility of directly observing conflicts in MTL and thus of distinguishing behaviourally equivalent PESs (e.g., the PESs consisting of a single or two conflicting copies of an event), and the presence in $\mathcal{L}_{hp}$ of propositions which are non-monadic with respect to event variables, make these logics not immediate to compare. Still, a deeper investigation is definitively worth to pursue, especially in view of the fact that, in the propositional case, the mu-calculus corresponds to the bisimulation invariant fragment of MSO logic [36].

The work summarised in [18] develops a game theoretic approach for model-checking a concurrent logic over partial order models. It has been observed in [20] that such logic is incomparable to $\mathcal{L}_{hp}$. Preliminary investigations shows that our model-checking framework could be adapted to such a logic and, more generally, to a logic joining the expressive power of the two. Moreover, further exploring the potentialities of a game theoretic approach in our setting represents an interesting venue of further research.

Compared to our previous work [26], we extended the range of the technique to the full logic $\mathcal{L}_{hp}$, without limitations concerning the alternation depth of formulae. Relaxing the restriction to strongly regular PESs, instead, appears to be quite problematic unless one is willing to deal with transfinite runs which, however, would be of very limited practical interest.

The tool is still very preliminary. As suggested by its (wishful) name (inspired by the classical Edinburgh Concurrency Workbench [37]) we would like to bring the TCWB to a more mature stage, working on optimisations and adding an interface that gives access to a richer set of commands.

Acknowledgements. We are grateful to Perdita Stevens for insightful hints and pointers to the literature 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. Winskel, G.: Events, causality and symmetry. Computer Journal **54**(1) (2011) 42–57
3. Pichon-Pharabod, J., Sewell, P.: A concurrency semantics for relaxed atomics that permits optimisation and avoids thin-air executions. In Bodík, R., Majumdar, R., eds.: Proceedings of POPL'16, ACM (2016) 622–633
4. Jeffrey, A., Riely, J.: On thin air reads towards an event structures model of relaxed memory. In Grohe, M., Koskinen, E., Shankar, N., eds.: Proceedings of LICS'16, ACM (2016) 759–767
5. Dumas, M., García-Bañuelos, L.: Process mining reloaded: Event structures as a unified representation of process models and event logs. In Devillers, R.R., Valmari, A., eds.: Petri Nets 2015. Volume 9115 of LNCS., Springer (2015) 33–48
6. Farzan, A., Madhusudan, P.: Causal atomicity. In: Proceedings of CAV'06. Volume 4144 of LNCS. (2006) 315–328
7. Baldan, P., Carraro, A.: A causal view on non-intereference. Fundamenta Informaticae **140**(1) (2015) 1–38
8. van Glabbeek, R., Goltz, U.: Refinement of actions and equivalence notions for concurrent systems. Acta Informatica **37**(4/5) (2001) 229–327
9. De Nicola, R., Ferrari, G.: Observational logics and concurrency models. In Nori, K.V., Madhavan, C.E.V., eds.: Proceedings of FST-TCS'90. Volume 472 of LNCS., Springer (1990) 301–315
10. 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)
11. Pinchinat, S., Laroussinie, F., Schnoebelen, P.: Logical characterization of truly concurrent bisimulation. Technical Report 114, LIFIA-IMAG, Grenoble (1994)
12. Penczek, W.: Branching time and partial order in temporal logics. In: Time and Logic: A Computational Approach, UCL Press (1995) 179–228
13. Nielsen, M., Clausen, C.: Games and logics for a noninterleaving bisimulation. Nordic Journal of Computing **2**(2) (1995) 221–249
14. Bradfield, J., Fröschle, S.: Independence-friendly modal logic and true concurrency. Nordic Journal of Computing **9**(1) (2002) 102–117
15. Alur, R., Peled, D., Penczek, W.: Model-checking of causality properties. In: Proceedings of LICS'95, IEEE Computer Society (1995) 90–100
16. Gutierrez, J., Bradfield, J.C.: Model-checking games for fixpoint logics with partial order models. In Bravetti, M., Zavattaro, G., eds.: Proceedings of CONCUR'09. Volume 5710 of LNCS., Springer (2009) 354–368
17. Gutierrez, J.: Logics and bisimulation games for concurrency, causality and conflict. In de Alfaro, L., ed.: Proceedings of FoSSaCS'09. Volume 5504 of LNCS., Springer (2009) 48–62
18. Gutierrez, J.: On bisimulation and model-checking for concurrent systems with partial order semantics. PhD thesis, University of Edinburgh (2011)
19. Madhusudan, P.: Model-checking trace event structures. In: Proceedings of LICS 2013, IEEE Computer Society (2003) 371–380
20. Baldan, P., Crafa, S.: A logic for true concurrency. Journal of the ACM **61**(4) (2014) 24:1–24:36

21. Phillips, I., Ulidowski, I.: Event identifier logic. Mathematical Structures in Computer Science **24**(2) (2014) 1–51
22. Best, E., Devillers, R., Kiehn, A., Pomello, L.: Fully concurrent bisimulation. Acta Informatica **28** (1991) 231–261
23. Rabinovich, A.M., Trakhtenbrot, B.A.: Behaviour structures and nets. Fundamenta Informaticae **11** (1988) 357–404
24. 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 Workshop. Volume 354 of LNCS., Heidelberg, DE, Springer (1988) 438–466
25. Jurdzinski, M., Nielsen, M., Srba, J.: Undecidability of domino games and hhp-bisimilarity. Information and Computation **184**(2) (2003) 343–368
26. Baldan, P., Padoan, T.: Local model checking in a logic for true concurrency. In Esparza, J., Murawski, A.S., eds.: Proceedings of FoSSaCS'17. Volume 10203 of LNCS., Springer (2017) 407–423
27. Thiagarajan, P.S.: Regular event structures and finite Petri nets: A conjecture. In Brauer, W., Ehrig, H., Karhumäki, J., Salomaa, A., eds.: Formal and Natural Computing - Essays Dedicated to Grzegorz Rozenberg [on occasion of his 60th birthday]. Volume 2300 of LNCS., Springer (2002) 244–256
28. Emerson, E.A., Jutla, C.S., Sistla, A.P.: On model checking for the μ-calculus and its fragments. Theoretical Computer Science **258**(1-2) (2001) 491–522
29. Mostowski, A.W.: Regular expressions for infinite trees and a standard form of automata. In Skowron, A., ed.: Proceedings of Computation Theory: Fifth Symposium 1984. Volume 208 of LNCS., Springer (1985) 157–168
30. Klauck, H.: Algorithms for parity games. In Grädel, E., Thomas, W., Wilke, T., eds.: Automata, Logics, and Infinite Games: A Guide to Current Research. Volume 2500., Springer (2002)
31. Abdulla, P.A., Kaati, L., Högberg, J.: Bisimulation minimization of tree automata. In Ibarra, O.H., Yen, H.C., eds.: Proceedings of CIAA'06. Volume 4094., Springer (2006) 173–185
32. Petri, C.: Kommunikation mit Automaten. Schriften des Institutes für Instrumentelle Matematik, Bonn. (1962)
33. Padoan, T.: True concurrency workbench. Available at `http://github.com/tpadoan/TCWB`
34. Chalopin, J., Chepoi, V.: A counterexample to Thiagarajan's conjecture on regular event structures. In Chatzigiannakis, I., Indyk, P., Kuhn, F., Muscholl, A., eds.: Proceedings of ICALP'17. Volume 80 of LIPIcs., Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik (2017) 101:1–101:14
35. Penczek, W.: Model-checking for a subclass of event structures. In Brinksma, E., ed.: Proceedings of TACAS'97. Volume 1217 of LNCS., Springer (1997) 145–164
36. Janin, D., Walukiewicz, I.: On the expressive completeness of the propositional mu-calculus with respect to monadic second order logic. In Montanari, U., Sassone, V., eds.: Proceedings of CONCUR'96, Springer (1996) 263–277
37. Stevens, P., Stirling, C.: Practical model-checking using games. In Steffen, B., ed.: Proceedings of TACAS'98, Springer (1998) 85–101