Hyper Partial Order Logic

Béatrice Bérard, Stefan Haar, Loïc Hélouët

Hyper Partial Order Logic

B. Bérard
Sorbonne Université, CNRS, Laboratoire d'Informatique de Paris 6, LIP6, F-75005 Paris
Beatrice.Berard@lip6.fr
https://pages.lip6.fr/Beatrice.Berard/

S. Haar
INRIA and LSV, ENS Paris-Saclay and CNRS, Université Paris-Saclay, France
stefan.haar@inria.fr
http://www.lsv.fr/ haar/

L. Hélouët
Univ Rennes, Inria, CNRS, IRISA
loic.helouet@inria.fr
http://people.rennes.inria.fr/Loic.Helouet

Abstract

We define HyPOL, a local hyper logic for partial order models, expressing properties of sets of runs. These properties depict shapes of causal dependencies in sets of partially ordered executions, with similarity relations defined as isomorphisms of past observations. Unsurprisingly, since comparison of projections are included, satisfiability of this logic is undecidable. We then address model checking of HyPOL and show that, already for safe Petri nets, the problem is undecidable. Fortunately, sensible restrictions of observations and nets allow us to bring back model checking of HyPOL to a decidable problem, namely model checking of MSO on graphs of bounded treewidth.

2012 ACM Subject Classification Theory of computation → Logic and verification

Keywords and phrases Partial orders, logic, hyper-logic

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2018.NN

1 Introduction

Hyperproperties. A way to address information security in systems is to guarantee various information flow properties. Examples of such properties are non-interference [17] (an attacker of a system cannot obtain confidential information from its observation of the system), or opacity of secrets [2] (an attacker cannot decide whether the system is in some particular secret configuration). For a long time since the seminal work of [17] introducing non-interference, security properties have been characterized as equivalences between partially observed behaviors of systems. This idea was later formalized [22] as combinations of language closure properties, the so-called "basic security predicates". We refer to [27] for a survey on language based information flow properties. More recently, logics with path equivalences [1] encompassing indistinguishability among partially observed executions have been proposed as a generic framework to define security conditions. Security properties are now frequently called hyperproperties [10, 9], i.e. properties of sets of runs.

Most proposals address verification questions in an interleaved setting, ignoring concurrency aspects. For instance, non-interference properties were considered for Petri nets [7], but still with techniques relying on interleaved interpretation of behaviors. Recently, [6] showed how to characterize some non-interference properties that cannot be handled in an interleaved model. This result is interesting, as it shows that even if complexity gains are not straightforward, considering causal dependences in systems leads to characterize types of

attacks of a system that cannot be characterized in an interleaved setting.

Local logics. We focus here on *local logics*, that account for causal dependencies and concurrency in behaviors of models. Several variants of local logic have been proposed : TLC^-, LD_0, PDL, $LPOC$, or even MSO. The first one, proposed by [25], is a logic tailored for Message Sequence Charts (MSCs). The logic features propositions, a next and an until operator and is interpreted over causal paths of MSCs. Model checking TLC^- is decidable for families of partial orders generated by High-level Message Sequence Charts (HMSCs). It is linear in the size of the considered HMSC, but exponential in the size of the formula.

The logic LD_0 [24] addresses properties of causal paths in partial orders. It resembles LTL in that its atomic propositions are attached to events, but it follows causal paths rather than linearizations, and is equipped with successor/predecessor relations.

An extension of TLC^- called Propositional Dynamic logic (PDL), which also subsumes LD_0, is given in [8] to express properties of Communicating Finite State Machines (CFSM). This logic is divided into path formulas and local formulas. Path formulas make it possible to navigate forward or backward in partially ordered executions via two relations: One that indicates whether an event f is the next executed event after e on the same process, and one that indicates whether a pair (e, f) forms a message. At each event along a followed path, truth of a local formula can be checked. Local formulas are used to check whether some atomic proposition holds at a given event, or whether some path formula holds at an event together with another PDL subformula. In general, verification of PDL for CFSM is undecidable, but checking whether some B-bounded execution of a CFSM (in which buffer contents can remain of size smaller than B) satisfies a PDL formula is PSPACE-complete. This result extends to HMSC specifications, which executions are naturally bounded. Another approach to study properties of partial orders generated by system executions is to express them directly as MSO properties. As MSO verification can easily be undecidable for some families of graphs, decidability is proved for families of partial orders generated by Message Sequence Charts in [20]. The result is obtained thanks to the particular shape of orders generated by MSCs that are "layered". Similarly, [21] considers restrictions in executions of CFSMs that have to synchronize frequently.

LPOC [16] is a logic for partially ordered computations. It describes the shape of partial orders, and not only of their causal paths. In addition to standard local operators, the logic has the ability to require existence of a particular partial order pattern in the causal past of an event. It was used as a specification formalism for diagnosis purposes, but without restriction, satisfiability of an LPOC formula is undecidable.

Contributions. We propose a framework unifying path equivalence logics, hyperproperties and partial order approaches. The logic borrows ingredients from LPOC [16]: in particular, it expresses existence of a *pattern* in a partial order, rather than on a causal path. It also borrows the idea of comparing executions up to observation, as proposed in $CTL_{\equiv}$, one of the branching logics with path equivalence proposed in [1]. Events in a pair of executions are considered as equivalent if the (partial) observations of their causal pasts are isomorphic. One of the artifacts used by [1] to obtain decidability of $CTL_{\equiv}$ is to require equivalence to hold only among events located at the *same depth* in executions. We do not use such an interpretation of equivalence, and rather exhibit sufficient conditions on behaviors of systems, that are almost a layeredness property [20], to obtain decidability.

We first define a partial order logic called Hyper Partial Order Logic (HyPOL for short). While we show undecidability for the satisfiability of this logic, we address model checking on true concurrency model, and start with Labeled Safe Petri Nets (LSPN). The universe of all behaviors of an LSPN can be defined as the set of processes of its complete unfolding [23].

Unsurprisingly, model checking HyPOL on runs of LSPNs is again undecidable. We then consider sensible assumptions on nets and projections saying that behaviors of a Petri net cannot remain unobserved for an arbitrary long time, and that equivalences necessarily link events whose common past is "not too old". We consider the unfolding of an LSPN as a graph connecting events and conditions via a successor relation. Isomorphism of causal pasts of events can be encoded as an additional relation on this unfolding graph. With these restrictions on nets and observations, model checking HyPOL can be brought back to verification of MSO on a graph generated by an hyperedge replacement grammar [18]. As MSO is decidable for such graphs [12], this yields decidability of HyPOL model checking for this subclass of nets and observations.

Outline. We introduce basic notations in Section 2. In Section 3, we define the logic HyPOL and prove undecidability of satisfiability. In Section 4, we show undecidability of HyPOL model checking on sets of processes of safe Petri nets, while decidability is proved in Section 5 for a subclass. Due to lack of space, proofs are provided in appendix.

2 Preliminaries

▶ **Definition 1.** A *labeled partial order (LPO)* over alphabet Σ is a triple $O = (E, \leq, \lambda)$ where E is a set of events, $\leq \subseteq E \times E$ is a partial ordering, i.e. a reflexive, transitive, antisymmetric relation, and $\lambda : E \to 2^{\Sigma}$ is a function associating with each event a set of labels from Σ.

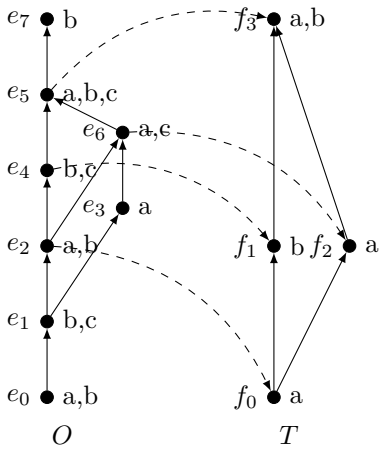

Figure 1 A partial order O over events $\{e_0, e_1, e_2, e_3, e_4, e_5, e_6, e_7\}$, a template T with events $\{f_0, f_1, f_2, f_3\}$, and a mapping (dashed arrows) witnessing that O matches T.

We denote by $\mathcal{LPO}(\Sigma)$ the set of labeled partial orders over Σ. For $O = (E, \leq, \lambda)$, we denote by $\max(O) = \{e \in E \mid \nexists f \neq e, e \leq f\}$ the set of its *maximal events*, and by $\min(O) = \{e \in E \mid \nexists f \neq e, f \leq e\}$ the set of its *minimal events*. The *covering* relation of O is a relation $< \subseteq E \times E$ such that $e < f$ iff $e \leq f$, $e \neq f$ and $\forall e' : (e \leq e' \leq f) \Rightarrow (e' \in \{e, f\})$. A *causal path* of O is a sequence of events $e_1.e_2 \dots e_n$ such that $e_i < e_{i+1}$. If $e \in E$, the *ideal* of e is the set $\downarrow e = \{f \mid f \leq e\}$ and its *ending section* is the set $\uparrow e = \{f \mid e \leq f\}$. The arrows and relations may be indexed by the order in case of ambiguity. A set $H \subseteq E$ of events is *downward closed* iff $H = \bigcup_{e \in H} \downarrow e$, and *upward closed* iff $H = \bigcup_{e \in H} \uparrow e$.

▶ **Definition 2.** The *restriction* of $O = (E, \leq, \lambda)$ to a subset $H \subseteq E$ is the LPO $O_{|H} = (H, \leq_{|H}, \lambda_{|H})$ where $\leq_{|H} = \leq \cap (H \times H)$ and $\lambda_{|H}$ is the restriction of λ to H. The *projection* of O on a subset of labels $\Sigma' \subseteq \Sigma$ is the restriction of O to events that carry labels in Σ'.

▶ **Definition 3.** Two partial orders $O = (E, \leq, \lambda)$ and $O' = (E', \leq', \lambda')$ over Σ are *isomorphic* (written $O \equiv O'$) iff there exists a bijective function $h : E \to E'$ such that $e \leq e' \iff h(e) \leq' h(e')$ and $\lambda(e) = \lambda'(h(e))$.

Note that two *discrete* LPOs O and O' are isomorphic iff their coverings are isomorphic.

▶ **Definition 4.** Let $O = (E, \leq, \lambda)$ and $T = (E_T, \leq_T, \lambda_T)$ be partial orders over Σ. Then O *matches* T iff there exists $H \subseteq E$ and a bijective mapping $h : H \to E_T$ such that

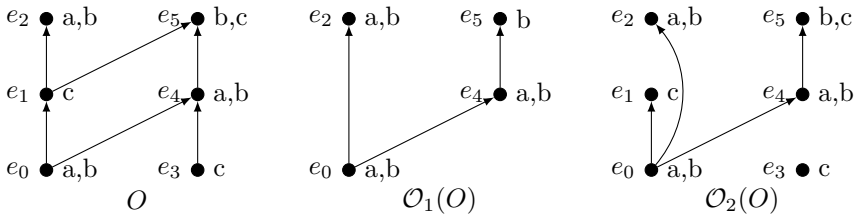

Figure 2 A partial order O, its projection $\mathcal{O}_1(O)$ on events that carry label a or b, and its restriction $\mathcal{O}_2(O)$ to causal dependencies from any event carrying label a to other events.

$\lambda_T(h(e)) \subseteq \lambda(e)$, and $e <_T e'$ implies $h^{-1}(e) < h^{-1}(e')$. The partial order T is called a *template* and we say that h is *witnessing* the matching.

In the sequel, we constrain the mapping witnessing a matching, using the notion of *anchored matching*. We say that there exists an anchored matching of template T at event e in O and f in T iff O matches T, and there exists a mapping $h_{e,f}$ witnessing this matching such that $h_{e,f}(e) = f$. In the example shown in Figure 1, the order O matches template T: the mapping h (depicted by dashed arrows) is defined by $h(e_2) = f_0$, $h(e_4) = f_1$, $h(e_6) = f_2$, $h(e_5) = f_3$. It satisfies: $\lambda_T(f_0) \subseteq \lambda(e_2)$, $\lambda_T(f_1) \subseteq \lambda(e_4)$, $\lambda_T(f_2) \subseteq \lambda(e_6)$, $\lambda_T(f_3) \subseteq \lambda(e_5)$.

An *observation function* is a mapping $\mathcal{O} : \mathcal{LPO}(\Sigma) \to \mathcal{LPO}(\Sigma')$, representing the visible part of the system. In what follows, we focus on observation functions that are the identity function id (i.e., the function such that $id(O) = O$), relabelings, and various restrictions of orders, for instance associating with $O = (E, \leq, \lambda)$ the order $O_{|F}$ for some $F \subseteq E$.

With a slight abuse, if $O = (E, \leq, \lambda)$ and $F \subseteq E$, we write $\mathcal{O}(F)$ for the corresponding subset of events of $\mathcal{O}(O)$. With observation functions like those described above, either an event is kept by observation (but it can be relabeled) or deleted. When event $e \in E$ has an image in $\mathcal{O}(E)$, we denote this image by $\mathcal{O}(e)$.

Consider the example of Figure 2. The partial order O contains events labeled by atomic propositions a, b, c. Let observation $\mathcal{O}_1$ be the projection of orders on events carrying a proposition in $\{a, b\}$. Such a projection can be used to indicate which actions are observed by a particular user. Now, consider observation $\mathcal{O}_2$, that restricts an order to causal dependencies in $\leq \cap \{(e, f) \mid a \in \lambda(e)\}$. This kind of observation can encode the fact that a particular user observing the execution of a system is not able to know if some events are causally related or not. Last, we can combine projections and order restriction: the observation defined by $\mathcal{O}_3(O) = \mathcal{O}_1(\mathcal{O}_2(O))$ describes what would be visible to a user of the system that logs events tagged with propositions a and b, and can only know dependencies from events tagged by a. For the order O in Figure 2, $\mathcal{O}_3(O) = \mathcal{O}_1(O)$.

3 Hyper Partial Order Logic

We are now ready to define HyPOL, a hyperproperty partial order logic. HyPOL is designed to express properties of partially observed sets of executions described by LPOs in $\mathcal{LPO}(\Sigma)$.

3.1 Syntax and semantics

We consider a set A of atomic propositions, a finite set $\mathcal{T}$ of templates labeled over A, and a finite set $\mathcal{O}bs$ of observation functions producing LPOs over A. We assume that $\Sigma \subseteq A$ but, since event labeling can be modified by observations, it is not always the case that $A = \Sigma$. The syntax of HyPOL is given by:

$$\phi ::= true \mid match(\mathcal{O}, T, f) \mid EX_{D,\mathcal{O}}\ \phi \mid EX_{\equiv,\mathcal{O}}\ \phi \mid \phi_1\ EU_{D,\mathcal{O}}\ \phi_2 \mid EG_{D,\mathcal{O}}\ \phi \mid \neg\, \phi \mid \phi_1 \vee \phi_2$$

where $D \subseteq A$, $T \in \mathcal{T}$, f is an event of T, and $\mathcal{O} \in \mathcal{O}bs$ an observation function.

A formula is *equivalence-free* iff it does not use the $EX_{\equiv,\mathcal{O}}$ operator. To reduce the number of primitives in our logic, we address labeling of events via templates. For $D \subseteq A$, we define a template T_D composed of a single event f_D labeled by all propositions in D. In particular, when $D = \{a\}$ for some proposition $a \in A$, we write T_a instead of $T_{\{a\}}$ and f_a instead of $f_{\{a\}}$. When template T_a is matched at some event e in an order O under observation $\mathcal{O}$, this means that the image of e by $\mathcal{O}$ carries proposition a.

We define derived operators (with $D \subseteq A$):

$$\begin{array}{lcl} \lambda_{\notin D} & ::= & \bigwedge_{a\in D} \neg match(id, T_a, f_a) \\ \lambda_{=D} & ::= & match(id, T_D, f_D) \wedge \lambda_{\notin A\setminus D} \\ EF_{D,\mathcal{O}}\ \phi & ::= & true\ EU_{D,\mathcal{O}}\phi \end{array} \qquad \begin{array}{lcl} AG_{D,\mathcal{O}}\ \phi & ::= & \neg EF_{D,\mathcal{O}}\neg\phi \\ AX_{D,\mathcal{O}} & ::= & \neg EX_{D,\mathcal{O}}\neg\phi \\ AX_{\equiv,\mathcal{O}} & ::= & \neg EX_{\equiv,\mathcal{O}}\neg\phi \end{array}$$

The semantics of HyPOL formulas is defined over a set $\mathcal{W} \subseteq \mathcal{L}PO(\Sigma)$ of orders, for $O=(E,\leq,\lambda)\in\mathcal{W}$ and $e \in E$. Letting $\lambda_{\mathcal{O}}$ be the labeling of $\mathcal{O}(O)$ and $<_{\mathcal{O}}$ its covering, we say that $O \in \mathcal{W}$ *satisfies* ϕ at event e (denoted by $O, e \models \phi$) if formula ϕ is satisfied when starting its evaluation from event e in order O:

- $O, e \models true$ for every event $e \in E$;
- $O, e \models \neg\phi$ iff $O, e \not\models \phi$ and $O, e \models \phi_1 \vee \phi_2$ iff $O, e \models \phi_1$ or $O, e \models \phi_2$;
- $O, e \models match(\mathcal{O}, T, f)$ if and only if f is an event of T, e has image e' in $\mathcal{O}(\downarrow e)$, and $\mathcal{O}(\downarrow e)$ matches T with at least a witness mapping $h_{e',f}$ associating f with e';
- $O, e \models EX_{D,\mathcal{O}}\ \phi$ iff $\exists f \in E$, e has image $e' \in \mathcal{O}(\uparrow e)$, f has image $f' \in \mathcal{O}(\uparrow f)$, $e' <_{\mathcal{O}} f'$, such that $\lambda_{\mathcal{O}}(e') \cap D \neq \emptyset$ and $O, f \models \phi$;
- $O, e \models EX_{\equiv,\mathcal{O}}\phi$ iff there exists $O' \in \mathcal{W}$ and $e' \neq e \in O'$ such that $\mathcal{O}(\downarrow_O e) \equiv \mathcal{O}(\downarrow_{O'} e')$ and $O', e' \models \phi$;
- $O, e \models \phi_1\ EU_{D,\mathcal{O}}\ \phi_2$ iff there exists an event $f \in E$ such that $O, f \models \phi_2$, and a finite set of events $e'_1, e'_2, \ldots e'_k \in \mathcal{O}(O)$ such that
  - $e'_1 <_{\mathcal{O}} e'_2 <_{\mathcal{O}} \cdots <_{\mathcal{O}} e'_k$, $e'_1 = \mathcal{O}(e)$ and $e'_k = \mathcal{O}(f)$,
  - $\forall i \in 2..k-1$, e'_i is the image of some event $e_i \in E$ by $\mathcal{O}$, $\lambda_{\mathcal{O}}(e'_i) \cap D \neq \emptyset$ and $O, e_i \models \phi_1$;
- $O, e \models EG_{D,\mathcal{O}}\phi$ iff
  - either there exists an infinite sequence of events $(e_i)_{i\geq 1}$ in E such that $e = e_1$, every e_i has an image e'_i in $\mathcal{O}(O)$, and $\forall i \geq 1$, $e'_i <_{\mathcal{O}} e'_{i+1}$, $\lambda(e'_i) \cap D \neq \emptyset$ and $O, e_i \models \phi$, or
  - there exists a finite set of events $e_1, \ldots e_k \in E$ such that $e = e_1$, for every $i \in 1..k$, e_i has an image e'_i by $\mathcal{O}$ with $e'_1 <_{\mathcal{O}} e'_2 <_{\mathcal{O}} \cdots <_{\mathcal{O}} e'_k$, $\lambda_{\mathcal{O}}(e'_i) \cap D \neq \emptyset$, $O, e_i \models \phi$, and $e'_k \in \max(\mathcal{O}(O))$.

In particular, $O, e \models match(id, T_a, f_a)$ iff e carries label a in order O, i.e. $a \in \lambda(e)$. Intuitively, formulas of the form $O, e \models EG_{D,\mathcal{O}}\ \phi$, $O, e \models \phi_1\ EU_{D,\mathcal{O}}\ \phi_2$, and $O, e \models EX_{D,\mathcal{O}}\ \phi$ describe properties of causal paths in orders, and have the standard interpretation seen for instance in LTL for words. Observation $\mathcal{O}$ is used to select successive events along a path, and set D performs an additional filtering among possible next events, by requiring the next considered event in a path to carry a label in D. The definition $O, e \models EX_{\equiv,\mathcal{O}}\ \phi$ requires existence of another order $O' \in \mathcal{W}$ and of an event $e' \in E_{O'}$ such that $e' \neq e$, but nothing forces O' and O to be different orders. Hence, e and e' can be distinct events from the same order that cannot be distinguished by observing their causal past.

An order O *satisfies* ϕ, denoted by $O \models \phi$, iff there exists $e \in \min(O)$ such that $O, e \models \phi$. The set of orders $\mathcal{W}$ satisfies ϕ iff every LPO $O \in \mathcal{W}$ satisfies ϕ. Last, ϕ is *satisfiable* iff

there exists a set of LPOs $\mathcal{W}$ such that $\mathcal{W} \models \phi$. Unsurprisingly, HyPOL is very powerful and satisfiability is undecidable on LPOs:

▶ **Theorem 5.** *Satisfiability of a HyPOL formula is undecidable.*

Proof Sketch. The proof is a reduction of Post's Correspondence Problem (PCP): given an instance I of PCP, we build a HyPOL formula ϕ_I such that I has a solution iff ϕ_I is satisfiable. (See Appendix 6.1 for details). ◀

3.2 An example: Causal Non-Interference

We begin with a small example showing that, in the context of concurrent models, languages are not discriminative enough. In Figure 3, the set $\mathcal{W} = \{O_1, O_2\}$ represents behaviors of a concurrent system, where h labels a non observable secret action, while events with labels a and b can be observed by an attacker. In a language-based setting, an attacker only observes the linearizations $a.b$ and $b.a$ of these orders. Hence it is not possible to deduce whether h has occurred or not. On the other hand, if causal dependencies are considered, observing that a precedes b reveals the occurrence of h, thus leaking the information that h occurred.

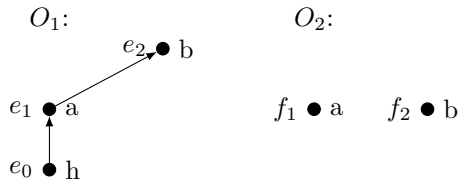

Figure 3 Two orders where observing linearizations is not enough to leak information.

For a more general example showing the discriminating power of HyPOL, consider non-interference. In the setting proposed by [17], a system is non-interferent if users cannot infer that classified actions have occurred only from observation of the system, i.e. execution of a classified event does not affect what a user can see or do. Such situations occur in a distributed system which can be accessed by two kinds of users: those with a high accreditation level and low-level users, that have limited access to operations and observations of the system. We suppose that high-level users can perform classified actions, the occurrences of which shall not be detected by low-level users. In a standard setting for non-interference properties, this situation is modeled by associating with each event occurring in the system a particular operation name. Let Σ be the set of all these names, with Σ_{high} the subset of confidential ones and $\Sigma_{low} = \Sigma \setminus \Sigma_{high}$ containing those which can be observed by low-level users. Observation $\mathcal{O}_{low}$ projects orders on events that carry at least one label in Σ_{low}. We can define a causal non-interference property with HyPOL as follows:

$$\phi_{CNI} ::= AG_{\Sigma,id}\left(\lambda_{\in\Sigma_{high}} \vee Pred_h \implies EX_{\equiv,\mathcal{O}_{low}}(\lambda_{\notin\Sigma_{high}} \wedge \neg Pred_h)\right)$$

where $\lambda_{\in\Sigma_{high}}$ stands for $\neg\lambda_{\notin\Sigma_{high}}$, $Pred_h ::= \bigvee_{a\in\Sigma} match(\mathcal{O}_{\mathrm{h},a}, T_{\mathrm{h}\leq a}, f)$, and $T_{\mathrm{h}\leq a}$ is the template containing a pair of events f_{h}, f such that $f_{\mathrm{h}} \leq f$, f_{h} carries proposition h, f carries proposition a and $\mathcal{O}_{\mathrm{h},a}$ is the observation that projects orders on $\Sigma_{high} \cup \{a\}$ and relabels events representing confidential operations with h.

Intuitively, satisfying $Pred_h$ means that a confidential operation occurred in the causal past of an event. Hence, an order O satisfies ϕ_{CNI} if, for every high-level event e in O, there exists an order O' and an event $e' \in O'$ such that $e \neq e'$, no high-level operation has occurred in the causal past of e', and a low level user cannot distinguish e from e' (i.e., $\mathcal{O}_{low}(\downarrow e) \equiv \mathcal{O}_{low}(\downarrow e')$). A system is (causally) non-interferent iff every order generated by this system satisfies ϕ_{CNI}, i.e. every order that contains a confidential operation cannot be distinguished from other orders that do not contain confidential operations. Note that $\mathcal{O}_{low}(O)$ is a partial order, hence ϕ_{CNI} uses the discriminating power of causal dependencies.

4 Model-checking HyPOL

We address the question of model checking of HyPOL formulas for a model for which at least reachability is decidable. As a starting point, we choose *labeled safe Petri nets (LSPN)*.

▶ **Definition 6.** A *Petri net* is a tuple $\mathcal{N} = (P, T, F, M_0)$ where P is a set of places, T is a set of transitions with $P \cap T = \emptyset$, $F \subseteq P \times T \cup T \times P$ is the flow relation, and $M_0 \in \mathbb{N}^P$ is the initial marking.

A net is *labeled* if it is equipped with a (not necessarily injective) mapping $\lambda : T \to \Sigma$ labeling the transitions. A *marking* is a multiset $M \in \mathbb{N}^P$. For $x \in P \cup T$, we define its *preset* by ${}^\bullet x = \{y \mid (y, x) \in F\}$ and its *postset* by $x^\bullet = \{y \mid (x, y) \in F\}$. The interleaved semantics of Petri nets can be defined as a (possibly infinite) transition system $LTS(\mathcal{N})$ where states are markings, the initial state is M_0, and the transition relation is defined by: $M \xrightarrow{t} M'$, iff (i) $M(p) \geq 1$ for all $p \in {}^\bullet t$, in which case transition t is said *firable* from M and (ii) $M' = (M \setminus {}^\bullet t) \uplus t^\bullet$ is the new marking reached by firing t. We write $M_0 \xrightarrow{*} M$ iff there exists a sequence of transition firings reaching M from M_0. The set of reachable markings is denoted by $Reach(\mathcal{N}) = \{M \mid M_0 \xrightarrow{*} M\}$.

We henceforth consider only safe Petri nets, where $Reach(\mathcal{N})$ is a subset of $\{0, 1\}^P$; we also assume that all transitions have at least one pre- and one post-place, i.e. $\forall\, t \in T :\ |{}^\bullet t| \geq 1 \leq |t^\bullet|$. Let us recall standard vocabulary and notations for nets (we borrow definitions from [15]). Two nodes $x, y \in P \cup T$ are in *causal relation* iff xF^*y. Transitions t and t' are in *immediate (structural) conflict* iff $t \neq t'$ and ${}^\bullet t \cap {}^\bullet t' \neq \emptyset$. Nodes $x, x' \in T \cup P$ are in *conflict*, written $x \# x'$, iff there exist $t, t' \in T$ in immediate conflict such that tF^*x and $t'F^*x'$. A subset C of $T \cup P$ is *conflict free* if for all $x, x' \in C$, $\neg(x \# x')$.

▶ **Definition 7.** An *occurrence net* is a Petri net $ON = (B, E, F, Cut_0)$ where the elements of B are called *conditions* and those of E *events*, and $Cut_0 \subseteq B$ such that:

- ON is acyclic, and hence $< \stackrel{\text{def}}{=} F^+$ and $\leq \stackrel{\text{def}}{=} F^*$ are strict and weak partial orders;
- $\forall e \in E :\ \neg(e \# e)$ (no event is in conflict with itself);
- $\forall b \in B, |{}^\bullet b| \leq 1$ (every condition has a unique predecessor);
- ON is finitary: for all $x \in E \cup B$, the set $Past(x) \stackrel{\text{def}}{=} \{y \mid y \leq x\}$ is finite; and
- Cut_0 contains exactly the $<$-minimal nodes of ON.

Nodes x and y are in *concurrency relation*, denoted $x \mid\mid y$, if neither $x < y, x > y$ nor $x \# y$ holds. Note that *every* occurrence net is safe, and that occurrence net ON is *conflict free* iff for every $b \in B$, one has $|b^\bullet| \leq 1$.

▶ **Definition 8.** A *prefix* of an occurrence net $ON = (B, E, F, Cut_0)$ is an event set $R \subseteq E$ that is *downward closed*, i.e. such that $e \in R$ and $e' < e$ together imply $e' \in R$. A prefix $C \subseteq E$ is a *configuration* iff it is *conflict free.*

▶ **Definition 9.** Given a net $\mathcal{N} = (P, T, F, M_0)$, and an occurrence net $ON = (B, E, \hat{F}, Cut_0)$, a *homomorphism* is a map $\mu : E \cup B \to T \cup P$ such that:

- $\mu(B) \subseteq P$ and $\mu(E) \subseteq T$,
- for all $e \in E$, the restriction of μ to ${}^\bullet e$ is a bijection from ${}^\bullet e$ to ${}^\bullet \mu(e)$, and the restriction of μ to $e^\bullet$ is a bijection from $e^\bullet$ to $\mu(e)^\bullet$, and
- $\mu(Cut_0) = \{p \in P \mid M_0(p) = 1\}$

The "unfolding" semantics of a safe labeled Petri net yields a labeled occurrence net.

▶ **Definition 10** (Unfolding). A *branching process* of a labeled Petri net $\mathcal{N} = (P, T, F, M_0, \lambda)$ is a triple $BR = (ON, \mu, \lambda')$ where $ON = (B, E, \hat{F}, Cut_0)$ is an occurrence net, μ is a homomorphism and $\forall e \in E, \lambda'(e) = \lambda(\mu(e))$. A *process* of a net $\mathcal{N}$ is a branching process of $\mathcal{N}$ such that for every condition $b \in B$, $|b^\bullet| \leq 1$, or equivalently, such that E is a configuration. If $BR_1 = (B_1, E_1, \hat{F}_1, Cut_0, \mu_1, \lambda'_1)$ and $BR_2 = (B_2, E_2, \hat{F}_2, Cut_0, \mu_2, \lambda'_2)$ are two branching processes of $\mathcal{N}$, BR_1 is a *prefix* of BR_2 iff $E_1 \subseteq E_2$, and $\hat{F}_1, \mu_1, \lambda'_1$ are the respective restrictions of $\hat{F}_2, \mu_2, \lambda'_2$ to B_1 and E_1. The *unfolding* of $\mathcal{N}$, denoted by $\mathcal{U}(\mathcal{N})$, is the maximal branching process w.r.t. the prefix relation.

Appendix 6.2 gives an algorithm for constructing the unfolding of a labeled safe Petri net. With every process $BR = (ON, \mu, \lambda)$ contained in $\mathcal{U}(\mathcal{N})$, with $ON = (B, E, F, Cut_0)$, is associated an LPO $\mathcal{O}rd(BR) = (E, \leq, \lambda)$. Note that events in such LPOs are labeled by a singleton (transition label), which is a sub-case of the LPOs defined in Section 2. We define $PR(\mathcal{N})$, the set of processes - up to isomorphism - that can be built from $\mathcal{N}$. Given a HyPOL formula ϕ, we say that $\mathcal{N}$ satisfies ϕ iff $\mathcal{O}rd(PR(\mathcal{N})) \models \phi$.

▶ **Theorem 11.** *The HyPOL model checking problem for safe Petri nets is undecidable.*

Proof (Sketch). We reuse the encoding of PCP from the proof of Theorem 5, and build a safe Petri net whose behaviors (processes) are exactly concatenations of the templates used in the HyPOL formula ϕ_I associated with an instance I of PCP (see Appendix 6.3). ◀

5 Decidability

The reason for the undecidability results above is that projections give a huge expressive power to HyPOL. Indeed, the difference in depth of equivalent events can be arbitrary large, and labeling allows for the design of a pair of growing sequences of letters w_1, w_2 where w_1 is always a prefix of w_2, yielding a non-terminating instance of PCP. We show in this section that one can recover decidability when restricting to Petri nets in which the difference in the depth of equivalent events is bounded.

Since the set of processes of a safe Petri net can be depicted in a compact way by its unfolding (as recalled in Section 4), a natural question is whether validity of a HyPOL formula expressing hyperproperties of the processes of a safe Petri net $\mathcal{N}$ can be rewritten as a property of its unfolding $\mathcal{U}(\mathcal{N})$. We first prove that this unfolding can be seen as a graph and defined as the production of a Hyperedge Replacement Grammar (HRG) [18].

▶ **Proposition** 1. Let $\mathcal{N}$ be a safe labeled Petri net. Then, there exists a hyperedge replacement grammar $\mathcal{G}_\mathcal{N}$ that generates $\mathcal{U}(\mathcal{N})$.

We detail the construction of $\mathcal{G}_\mathcal{N}$ in Appendix 6.4. Note that $\mathcal{G}_\mathcal{N}$ does not define a semantics of $\mathcal{N}$ via application of one rewriting rule per transition firing, as proposed in [3, 4], but rather *builds* the unfolding. The grammar $\mathcal{G}_\mathcal{N}$ starts from an axiom Ax. Denoting by $\mathcal{G}^\omega_\mathcal{N}(Ax)$ the (unique) graph generated from Ax, we have $\mathcal{G}^\omega_\mathcal{N}(Ax) = \mathcal{U}(\mathcal{N})$. $\mathcal{G}_\mathcal{N}$ exhibits a certain form of regularity, but this is not yet sufficient to check HyPOL formulas, nor to express HyPOL properties in terms of properties of $\mathcal{G}_\mathcal{N}$. Indeed, the graphical representation of $\mathcal{U}(\mathcal{N})$ does not address equivalences. We adapt the idea of [1], and represent isomorphism of causal pasts of events w.r.t. an observation function as a new relation connecting events. In other words, we augment $\mathcal{U}(\mathcal{N})$ with additional edges connecting equivalent events.

▶ **Definition 12** (Execution Graph). Given a set of observation functions $\mathcal{O}_1, \ldots, \mathcal{O}_k$, the *execution graph* of $\mathcal{N}$ is the graph $G_{\mathcal{U}(\mathcal{N})} = (E \cup B, \longrightarrow, \lambda)$, where E and B are the sets of events and conditions in $\mathcal{U}(\mathcal{N})$, and $\longrightarrow \subseteq (E \times \{0\} \times B) \cup (B \times \{0\} \times E) \cup (E \times \{1, \ldots k\} \times E)$

is the relation defined by: $(e,0,b) \in \longrightarrow$ iff $e \in {}^\bullet b$ in $\mathcal{U}$, $(b,0,e) \in \longrightarrow$ iff $b \in {}^\bullet e$ in $\mathcal{U}$, and $(e,i,e') \in \longrightarrow$ for $1 \leq i \leq k$ iff $e \neq e'$ and $\mathcal{O}_i(\downarrow e) \equiv \mathcal{O}_i(\downarrow e')$.

We write $e \xrightarrow{i} e'$ for $(e,i,e') \in \longrightarrow$. So far, we have simply recast ordering and equivalence of events into a graph setting, but this translation does not change decidability of hyperproperties. Even if the unfolding $\mathcal{U}(\mathcal{N})$ can be generated by an HRG, this is not the case for $G_{\mathcal{U}(\mathcal{N})}$. Indeed, to produce edges, hyperarcs of an HRG need to memorize nodes that will be at the origin or destination of an edge in future productions of the grammar. In particular, for $G_{\mathcal{U}(\mathcal{N})}$, this means that hyperarcs of any HRG producing this graph have to memorize a list of events that will be declared as equivalent to some event (w.r.t. a particular observation $\mathcal{O}_i$) generated in future rewritings.

▶ **Proposition** 2. There exist labeled safe Petri nets and observation functions whose execution graphs are not of bounded treewidth, and cannot be represented by an hyperedge replacement grammar.

Proof (Sketch). We exhibit a net, and an observation function whose execution graph contains grid minors of arbitrary sizes. It is well known [26] that a family of graphs FG has bounded treewidth iff there exists a constant m such that no graph $G \in FG$ has a minor isomorphic to the $m \times m$ grid and that HRGs can only generate graphs of bounded treewidth (see for instance [11]). See appendix 6.5 for a complete proof. ◀

▶ **Definition 13.** Let $ON = (B, E, F, Cut_0)$ be an occurrence net. The *height* of en event e or condition b in ON is the function $\mathcal{H} : B \cup E \to \mathbb{N}$ be defined recursively by

$$\begin{aligned} \forall\, b \in Cut_0 :\ \mathcal{H}(b) &\stackrel{\text{def}}{=} 1 \\ \forall\, x \in B \cup E :\ \mathcal{H}(x) &\stackrel{\text{def}}{=} 1 + \max\{\mathcal{H}(y) \mid y \in {}^\bullet x\}\,. \end{aligned}$$

By extension, the height $\mathcal{H}(A)$ for a set $A \subseteq (B \cup E)$ is given by $\mathcal{H}(\emptyset) = 0$ and $\mathcal{H}(A) \stackrel{\text{def}}{=} \sup_{x \in A} \mathcal{H}(x)$. Now, define the *distance* $\mathsf{dist} : (B \cup E) \times (B \cup E) \to \mathbb{N}$ by

$$\begin{aligned} \mathcal{H}_\sqcap(e,e') &\stackrel{\text{def}}{=} \mathcal{H}\left(\downarrow e \cap \downarrow e'\right) \\ \mathsf{dist}(e,e') &\stackrel{\text{def}}{=} \max\left(\mathcal{H}\left(e\right), \mathcal{H}\left(e'\right)\right) - \mathcal{H}_\sqcap(e,e'). \end{aligned}$$

Intuitively, $dist(e,e')$ measures the maximal number of edges between e, e' and their common past. This distance dist defines a pseudometric. Using this notion of distance, we can define the *K-Ball* of an event e in the unfolding $\mathcal{U}(\mathcal{N})$ as the set of nodes in $\mathcal{U}(\mathcal{N})$ that are at distance at most K from e. Formally, $Ball_K(e) = \{n \in \mathcal{U}(\mathcal{N}) \mid \mathsf{dist}(n,e) \leq K\}$. In the rest of the paper, we consider classes of unfoldings where two events can only be equivalent w.r.t. any observation $\mathcal{O}_i$ if they are in the *K-Ball* of one another.

An important remark is that even for a safe Petri net $\mathcal{N}$, given an integer $K \in \mathbb{N}$, the *K-Ball* of an event e may not be finite. Furthermore, the graph $(E \cup B, \xrightarrow{0})$ depicting the unfolding $\mathcal{U}(\mathcal{N})$ without equivalence edges is always a graph of finite incoming degree, but this is not necessarily the case for $G_{\mathcal{U}(\mathcal{N})}$. In the rest of the paper, we will see that HyPOL formulas can be encoded as MSO properties of $G_{\mathcal{U}(\mathcal{N})}$. The reason for undecidability of HyPOL is hence the nature of execution graphs, that cannot be generated in general by context free graph grammars, are not of bounded treewidth,... nor enjoy any of the properties that usually make MSO decidable. We can recover decidability with some restrictions. Let $\downarrow_K e = \downarrow e \cap Ball_K(e)$ denote the $K-$bounded past of e.

▶ **Definition 14.** Let $\mathcal{N}$ be a safe Petri net, and $\mathcal{O}_i$ be an observation function. $\mathcal{N}$ is $K-$layered w.r.t. $\mathcal{O}_i$ iff $\forall e, e' \in \mathcal{U}(\mathcal{N})$:

- there is a bound $S_K \in \mathbb{N}$ such that $|Ball_K(e)| \leq S_K$;
- $\mathsf{dist}(e, e') > K$ implies $e \not\equiv e'$;
- $\mathsf{dist}(e, e') \leq K$ implies one can compute $H = \{f_1, \dots f_m\} \subseteq \downarrow_K e \cup \downarrow_K e'$ such that, letting $F_{e,e'} = \bigcup_{i \in 1..m} \downarrow f_i$ and $\hat{F}_{e,e'} = F_{e,e'} \setminus H$,

$$e \equiv_i e' \text{ iff } \mathcal{O}_i(\downarrow e \setminus \hat{F}_{e,e'}) \equiv_i \mathcal{O}_i(\downarrow e' \setminus \hat{F}_{e,e'}).$$

In the sequel, we assume that observation functions $\mathcal{O}_1, \dots \mathcal{O}_k$ are given, and we say that a safe Petri net $\mathcal{N}$ is $K-$layered iff it is $K-$layered for every $\mathcal{O}_i$. Intuitively, a Petri net is $K-$layered w.r.t. observation $\mathcal{O}_i$ iff one can decide equivalence of a pair of events e, e' w.r.t. $\mathcal{O}_i$ from their $K-$bounded past.

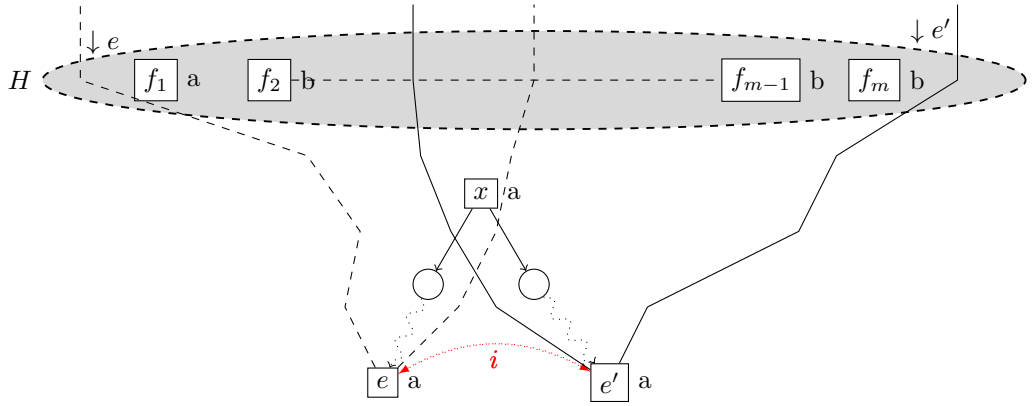

Figure 4 Equivalence w.r.t. $\mathcal{O}_i$ in the unfolding of a $K-$layered Petri net

▶ **Proposition 3**. Let $\mathcal{N}$ be a $K-$layered safe Petri net. Then, one can effectively compute an hyperedge replacement grammar $\mathcal{G}_{K,\mathcal{N}}$ that recognizes the execution graph $G_{\mathcal{U}(\mathcal{N})}$.

Proof (sketch). First, one can notice that in the unfolding of a $K-$layered safe Petri net, for every observation $\mathcal{O}_i$, every event e has a bounded number of events connected to it via relation $\xrightarrow{i}$. This is due to the fact that this set is contained in its finite *K-Ball*. The hyperedge replacement grammar $\mathcal{G}_{K,\mathcal{N}}$ starts from an axiom representing a complete finite prefix of the unfolding of $\mathcal{N}$ with hyperarcs. Its hyperarcs represent possible extensions of this prefix from its maximal markings. Rules of $\mathcal{G}_{K,\mathcal{N}}$ are of the form $r = (h_{t,lab}, HG_{t,lab})$ where $h_{t,lab}$ contains all conditions and events appearing in the *K-Balls* of the next occurrence of a transition t that can be appended after a maximal marking, and lab is a labeling providing sufficient information to know the ordering among events and a part of their common past. $HG_{t,lab}$ is an hypergraph containing the newly generated occurrences of events and conditions in the execution graph, the flow relation among them, and connects equivalent events (contained in the events of $h_{t,lab}$ and $HG_{t,lab}$) and creating one hyperarc per new maximal marking. Appendix 6.7 gives a complete construction of this grammar. ◀

We now show that model checking HyPOL on $K-$layered execution graphs can be brought back to verification of an equivalent MSO property. But the first question to address is decidability of MSO on execution graphs. An MSO formula uses the following syntax:

$$\phi \quad ::= \quad lab_a(x) \mid edge(x, y) \mid edge_i(x, y) \mid x = y \mid x \in X \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid \exists x, \phi \mid \exists X, \phi$$

where $x, y, ...$ are first order variables representing vertices in a graph, and $X, Y, ...$ are second order variables representing sets of vertices in a graph. In execution graphs, first order variables will represent events or conditions, and an edge the flow relation or isomorphism.

An interpretation $\mathcal{I}$ of an MSO formula ϕ over a graph G is an assignment of nodes of G to first order variables used in ϕ and of subsets of nodes of G to second order variables. An MSO formula ϕ holds for G under interpretation $\mathcal{I}$ iff replacing variables in ϕ by their interpretation yields a tautology. A graph satisfies formula ϕ iff there exists an interpretation $\mathcal{I}$ such that ϕ holds for G under $\mathcal{I}$. Classes of graphs with decidable MSO theory have been considered for a long time (see for instance [11] for a complete monograph on this topic). As MSO is decidable for context free graphs such as the graphs generated by HRGs ([12], Corollary 4.10), we immediately have the following property:

▶ **Corollary 15.** *MSO is decidable on execution graphs of K–layered labeled safe Petri nets.*

Note that the decidability highlighted in corollary 15 does not necessarily hold outside the class of $K-$layered nets. As shown in Proposition 2, execution graphs of safe Petri nets may contain grids minors of arbitrary sizes and hence in general do not have a bounded treewidth [26]. MSO is also undecidable in general for execution graphs: one can use a safe Petri net whose unfolding is a binary tree and an observation that implements the "same level" relation on this tree. It is well known that MSO is undecidable on this graph [28]. We will use MSO to address decidability of HyPOL, by converting formulas to MSO, and in particular equivalences into $\xrightarrow{i}$ relations among events.

▶ **Proposition 4**. Let ϕ be a HyPOL formula. Then there exists an MSO formula ψ such that $\mathcal{N} \models \phi$ iff $G_{\mathcal{U}(\mathcal{N})} \models \psi$.

Proof (sketch). We first encode in MSO a $succ(e, e')$ relation that relates pairs of events such that $e^\bullet \cap {}^\bullet e \neq \emptyset$. Then, causal precedence $\leq$ in an order can be encoded with MSO. A property of the form $x \models EX_{\equiv,\mathcal{O}_i}\ \phi$ asks existence of an edge $x \xrightarrow{i} y$ where y satisfies the MSO translation of ϕ. Until operations are described as properties of chains of events, that can again be encoded with MSO, and patterns embedding are MSO properties checking existence of some subgraph. A complete translation is given in Appendix 6.6. ◀

Proposition 4 holds for any net $\mathcal{N}$ and its execution graph $G_{\mathcal{U}(\mathcal{N})}$. However, in general, $G_{\mathcal{U}(\mathcal{N})}$ is not of bounded treewidth. One can always choose an integer K, and build a context free graph grammar $\mathcal{G}_{K,\mathcal{N}}$ as proposed in Proposition 3, but in general, the graph generated by $\mathcal{G}_{K,\mathcal{N}}$ is only a subgraph of $G_{\mathcal{U}(\mathcal{N})}$, where some $\xrightarrow{i}$ edges are missing. This is not surprising: in non-layered nets, the sizes of equivalence classes in $G_{\mathcal{U}(\mathcal{N})}$ need not be finite. If $\mathcal{N}$ is $K-$layered, the graph generated by $\mathcal{G}_{K,\mathcal{N}}$ and $G_{\mathcal{U}(\mathcal{N})}$ are equivalent. Further, isomorphism is one of the building block of HyPOL, but in general cannot be expressed in MSO. The translation from HyPOL to MSO applies to any HyPOLformula for any type of net and observation. We know that MSO is decidable for HRGs [12, 19]. So, in general, $G_{\mathcal{U}(\mathcal{N})}$ is not the production of an HRG. Altogether, these remarks give the following corollaries:

▶ **Corollary 16.** *It is undecidable whether the execution graph of a net $\mathcal{N}$ satisfies an MSO formula.*

▶ **Corollary 17.** *Model checking equivalence-free HyPOL properties on labeled safe Petri nets is decidable.*

▶ **Corollary 18.** *HyPOL model checking is decidable for $K-$layered safe Petri nets.*

K–layeredness is a semantic property, that should hold on the possibly infinite unfolding of a net. However, some syntactic classes of nets meet the conditions needed to layer equivalences. In the following, we only consider observations that are projections. Slightly abusing our notations, for a transition t we will denote by $\mathcal{O}_i(t)$ the LPO obtained by applying observation $\mathcal{O}_i$ to the LPO O_t that contains a single event e with $\lambda(e) = \lambda(t)$.

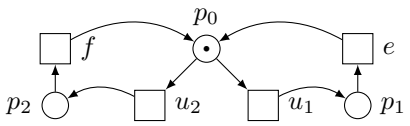

Figure 5 A net $\mathcal{N}_1$. Observation $\mathcal{O}_a$ projects LPOs on events labeled a. $\mathcal{N}_1$ is not observable: $\mathcal{O}_a$ cannot distinguish behaviors in $u_1.e.(u_1.e + u_2.f)^k$ from those in $u_2.f.(u_1.e + u_2.f)^k$

▶ **Definition 19.** Let $\mathcal{N}$ be a safe Petri net. Two transitions t, t' are *independent* iff there is no link from t to t' in the flow relation of $\mathcal{N}$. We will say that $\mathcal{N}$ is *observable* iff,

i) for every observation $\mathcal{O}_i$, and every cyclic behavior $t_1 \dots t_n$ of $LTS(\mathcal{N})$, $\mathcal{O}_i(t_1 \dots t_n) \neq \emptyset$,

ii) For every reachable marking M of $\mathcal{N}$, every observation $\mathcal{O}_i$ and every pair of conflicting transitions t_1, t_2 enabled in M, there exists a bound k_c such that for every pair of path $\rho = t_1.t_{1,1} \dots t_{1,p}$ and $\rho_2 = t_2.t_{2,1} \dots t_{2,q}$, if $p > k_c$ or $q > k_c$ then $\mathcal{O}_i(O_{\rho_1}) \neq \mathcal{O}_i(O_{\rho_2})$, where O_{ρ_1} (resp O_{ρ_2}) is the process of $\mathcal{N}$ obtained by successively appending $t_1, t_{1,1}, \dots$ (resp. $t_2, t_{2,1}, \dots$) to M_0.

iii) for every observation $\mathcal{O}_i$ and every cyclic behavior $M \xrightarrow{\rho} M$ of $LTS(\mathcal{N})$ with $\rho = t_1 \dots t_n$ and such that $t_1 \dots t_n$ can be partitioned into sets $T_1, T_2, \dots T_k$ of independent transitions $\forall j, j' \in 1..k$, there exists $t_j \in T_j$ and $t_{j'} \in T_{j'}$ such that $\mathcal{O}_i(t_j) \neq \mathcal{O}_i(t_{j'})$.

Condition *i*) forbids cyclic behaviors that cannot be observed. This is a sensible restriction often required for diagnosis (where it is called *convergence*, as in [5]). It guarantees that an event cannot be equivalent to an arbitrary number of predecessors. Condition *ii*) indicates that each branch of a choice in the net is eventually visible by each observation after a bounded duration. Condition *iii*) says that parallel sequences of transition cannot grow up to an arbitrary size without becoming distinguishable by all observations.

▶ **Proposition** 5. Let $\mathcal{N} = (P, T, F, M_0, \lambda)$ be a safe labeled observable Petri net for observations $\mathcal{O}_1, \dots, \mathcal{O}_k$. Then $\mathcal{N}$ is $K-$layered, for some $K \leq \max(2.k_c, 3.|T|)$

▶ **Corollary 20.** *HyPOL model-checking is decidable for observable safe Petri nets.*

6 Conclusion

HyPOL is a local logic for hyperproperties of partially observed set of labeled partial orders. It is powerful enough to express properties such as non-interference in distributed systems. This logic follows the same line as local logics such as TLC^- or LD_0, as it depicts shapes of causal chains in partially ordered computations. In addition, it is possible to check whether some finite behavior has occurred in the past, and a new modal operator is introduced to move from an event in an LPO to another equivalent event in another LPO. Unsurprisingly, such a powerful logic is undecidable, even for simple models such as safe labeled Petri nets. However, upon some restrictions, one can bring back verification of HyPOL formulas to verification of MSO properties on unfoldings of nets decorated with additional edges that simulate equivalences. The restrictions forbid nets with infinite unobservable runs, and assume bounds on the depth of indistinguishable suffixes. In this context, equivalence of runs only depends on a bounded future and past of each event, and decorated unfoldings have bounded treewidth. So far, we do not know whether K–layeredness is decidable for a fixed K. Another interesting question is existence of a bound K such that a net $\mathcal{N}$ is K–layered. We strongly believe that some restrictions used in observable nets can be relaxed, or adapted to consider larger classes of nets for which decorated unfoldings are of bounded treewidth or split-width [13]. A natural question that follows is whether theses classes of nets have sensible and decidable syntactic characterizations.

References

1 R. Alur, P. Cerný, and S. Chaudhuri. Model checking on trees with path equivalences. In *TACAS 2007*, volume 4424 of *LNCS*, pages 664–678. Springer, 2007.

2 E. Badouel, M.A. Bednarczyk, A.M. Borzyszkowski, B. Caillaud, and P. Darondeau. Concurrent secrets. *Discrete Event Dynamic Systems*, 17(4):425–446, 2007.

3 P. Baldan, T. Chatain, S. Haar, and B. König. Unfolding-based diagnosis of systems with an evolving topology. In *19th International Conference on Concurrency Theory (CONCUR'08)*, volume 5201 of *LNCS*, pages 203–217. Springer, 2008. URL: `http://www.lsv.ens-cachan.fr/Publis/PAPERS/PDF/BCHK-concur08.pdf`.

4 P. Baldan, T. Chatain, S. Haar, and B. König. Unfolding-based diagnosis of systems with an evolving topology. *Information and Computation*, 208(10):1169–1192, October 2010. URL: `http://www.lsv.ens-cachan.fr/Publis/PAPERS/PDF/BCHK-icomp10.pdf`, `doi:10.1016/j.ic.2009.11.009`.

5 B. Bérard, S. Haar, S. Schmitz, and S. Schwoon. The complexity of diagnosability and opacity verification for petri nets. In *PETRI NETS'17*, volume 10258 of *LNCS*, pages 200–220. Springer, 2017.

6 B. Bérard, L. Hélouët, and J. Mullins. Non-interference in partial order models. *ACM Trans. Embedded Comput. Syst.*, 16(2):44:1–44:34, 2017.

7 E. Best, P. Darondeau, and R. Gorrieri. On the decidability of non interference over unbounded Petri nets. In *Proc. of SecCo*, volume 51 of *EPTCS*, pages 16–33, 2010.

8 B. Bollig, D. Kuske, and I. Meinecke. Propositional dynamic logic for message-passing systems. *Logical Methods in Computer Science*, 6(3), 2010.

9 M.R. Clarkson, B. Finkbeiner, M. Koleini, K.K. Micinski, M.N. Rabe, and C. Sánchez. Temporal logics for hyperproperties. In *POST*, pages 265–284, 2014.

10 M.R. Clarkson and F.B. Schneider. Hyperproperties. *Journal of Computer Security*, 18(6):1157–1210, 2010.

11 B. Courcelle and J. Engelfriet. *Graph Structure and Monadic Second-Order Logic, a language theoretic approach*. Cambridge Univ. Press, 2012.

12 Bruno Courcelle. The monadic second-order logic of graphs. i. recognizable sets of finite graphs. *Inf. Comput.*, 85(1):12–75, 1990.

13 A. Cyriac, P. Gastin, and K.N. Narayan Kumar. MSO decidability of multi-pushdown systems via split-width. In *CONCUR 2012*, volume 7454 of *LNCS*, pages 547–561, 2012.

14 J. Engelfriet. Branching processes of Petri nets. *Acta Inf.*, 28(6):575–591, 1991.

15 J. Esparza, S. Römer, and W. Vogler. An improvement of McMillan's unfolding algorithm. *Formal Methods in System Design*, 20(3):285–310, 2002.

16 T. Gazagnaire, L. Hélouët, and S. S. Yang. Logic-based diagnosis for distributed systems. *Perspectives in Concurrency, a festschrift for P.S. Thiagarajan*, pages 482–505, 2009.

17 J.A. Goguen and J. Meseguer. Security policies and security models. In *Proc. of IEEE Symposium on Security and Privacy*, pages 11–20, 1982.

18 A. Habel. *Hyperedge Replacement: Grammars and Languages*, volume 643 of *LNCS*. Springer, 1992.

19 C. Lautemann. Tree automata, tree decomposition and hyperedge replacement. In *Graph-Grammars and Their Application to Computer Science, 4th International Workshop*, volume 532 of *LNCS*, pages 520–537. Springer, 1990.

20 P. Madhusudan and B. Meenakshi. Beyond message sequence graphs. In *FST TCS'01: Foundations of Software Technology and Theoretical Computer Science*, volume 2245 of *LNCS*, pages 256–267. Springer, 2001.

21 P. Madhusudan, P.S. Thiagarajan, and S. Yang. The MSO theory of connectedly communicating processes. In *FSTTCS'05*, volume 3821 of *LNCS*, pages 201–212. Springer, 2005.

22 H. Mantel. Possibilistic definitions of security - an assembly kit. In *Proc. of the 13th IEEE Computer Security Foundations Workshop, (CSFW'00)*, pages 185–199, 2000.

23 K.L. McMillan. A technique of state space search based on unfolding. *Formal Methods in System Design*, 6(1):45–65, 1995.

24 B. Meenakshi and R. Ramanujam. Reasoning about layered message passing systems. *Computer Languages, Systems & Structures*, 30(3-4):171–206, 2004.

25 D.A. Peled. Specification and verification of message sequence charts. In *FORTE/PSTV'00*, volume 183 of *IFIP Conference Proceedings*, pages 139–154. Kluwer, 2000.

26 N. Robertson and P.D. Seymour. Graph minors. x. obstructions to tree-decomposition. *J. Comb. Theory, Ser. B*, 52(2):153–190, 1991.

27 A. Sabelfeld and A.C. Myers. Language-based information-flow security. *IEEE Journal on Selected Areas in Communications*, 21(1):5–19, 2003.

28 A. Spelten, W. Thomas, and S. Winter. Trees over infinite structures and path logics with synchronization. In *13th International Workshop on Verification of Infinite-State Systems, INFINITY 2011*, volume 73 of *EPTCS*, pages 20–34, 2011.

Appendix

6.1 Proof of Theorem 5

Theorem 5: *Satisfiability of a HyPOL formula is undecidable.*

Proof. The proof consists of a reduction of the Post Correspondence Problem (PCP). Recall that an instance I of PCP is a sequence $(x_1, y_1), \ldots, (x_n, y_n)$ of n pairs of words over some alphabet. A (non trivial) solution of size k is a (non empty) sequence of indices $\sigma = i_1 \ldots i_k$ such that $x_{i_1} \ldots x_{i_k} = y_{i_1} \ldots y_{i_k}$. If the alphabet contains at least two letters, PCP is undecidable for $n \geq 7$. Moreover, we can assume that for all $1 \leq i \leq n$, $x_i \neq y_i$ (otherwise the problem can be trivially decided with a solution of size $k = 1$).

Given an instance I, we build a formula ϕ_I of HyPOL such that ϕ_I is satisfiable if and only if I has a (non trivial) solution.

Let I be the sequence $(x_1, y_1), \ldots, (x_n, y_n)$ of words over alphabet A. We write $z = z(1) \ldots z(\ell)$ where $\ell = |z|$ is the length of word z, with $\ell_i = |x_i|$ and $h_i = |y_i|$, $1 \leq i \leq n$ and we consider the family of templates T_i, $1 \leq i \leq n$, as depicted in Figure 6. The set of events of T_i is $E_i = \{x_i, y_i, s_i, e_i\} \cup \{x_{i,j} \mid 1 \leq j \leq \ell_i\} \cup \{y_{i,j} \mid 1 \leq j \leq h_i\}$ and labels are in $P_i = A \cup \{\sharp, \text{start}_i, \text{end}_i\}$. We set $Ind = \{start_i, end_i, 1 \leq i \leq n\}$, $S = \{start_i, 1 \leq i \leq n\}$ and the global set of labels is $P = \cup_{i=1}^{n} P_i$. Intuitively, a solution $\sigma = i_1 \ldots i_k$ will be described by the sequence of templates $T_{i_1} \ldots T_{i_k}$.

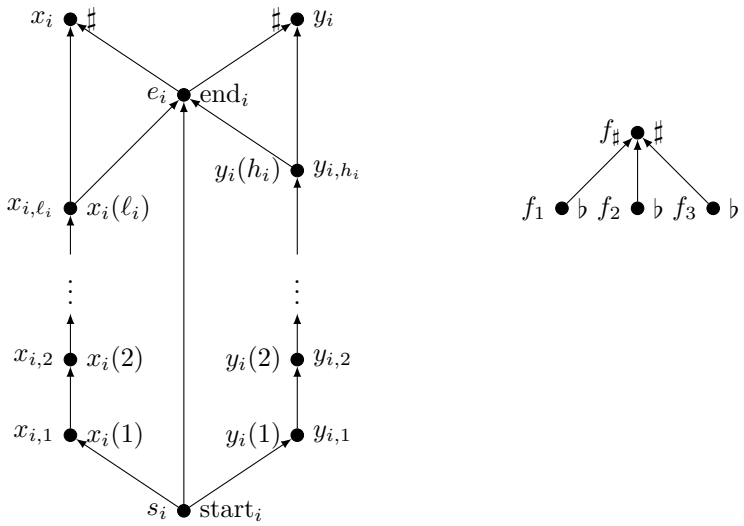

Figure 6 Templates T_i and $T_\sharp$.

To detect that a solution ends with an event labeled by $\sharp$, we define the formula $stop ::= \lambda_{=\{\sharp\}} \wedge \neg EX_{P,id} true$. We also need to express that any event with label $\sharp$ has at most two predecessors:

$$two\text{-}pred\sharp ::= AG_{P,id}(\lambda_{=\{\sharp\}} \implies \neg match(\mathcal{O}_\sharp, T_\sharp, f_\sharp))$$

where $T_\sharp$ is the pattern depicted on Figure 6 right and $\mathcal{O}_\sharp$ keeps any event with label $\sharp$ unchanged and relabels all other events with $\flat$. Now, if $\mathcal{O}_S$ denotes the projection on S, keeping only events with labels in S, a solution is decribed by:

$$IsSeqIndex ::= EG_{S,\mathcal{O}_S}(\vee_{i=1}^{n} HoldsT_i) \wedge EF_{P,id} stop$$

where $HoldsT_i ::= match(id, T_i, s_i)$.

Finally, we consider the subset $\mathcal{W}$ of orders of $\mathcal{L}PO(P)$ where all labels are singletons. Note that this condition can be ensured by the formula $Sing ::= AG_{P,id}(\vee_{p\in P}\lambda_{=\{p\}})$. For an order $O = (E, \leq, \lambda) \in \mathcal{W}$, we write $E = E_A \cup E_\sharp \cup E_{ind}$ as a disjoint union with $E_A = E \cap \lambda^{-1}(A)$, $E_\sharp = E \cap \lambda^{-1}(\{\sharp\})$ and $E_{ind} = E \cap \lambda^{-1}(Ind)$. We define the observation function $\mathcal{O}_{sol}$ over $\mathcal{W}$ by keeping all events and restricting $\leq$ to $(E \times E) \setminus ((E_A \times E_{ind}) \cup (E_{ind} \times E_A))$, thus removing the order between letters and indices.

The formula ϕ_I is then defined by :

$$\phi_I ::= \textit{two-pred}\sharp \wedge \textit{IsSeqIndex} \wedge (\textit{stop} \implies EX_{\equiv,\mathcal{O}_{sol}} \textit{true}),$$

where the last sub-formula means that from some final $\sharp$, it will not be possible to distinguish between paths with labels from the x_i's and those with labels from the y_i's.

Then, there is an order O in $\mathcal{W}$ satisfying ϕ_I if and only if I has a non trivial solution. ◀

6.2 An algorithm to build an unfolding of a safe Petri net

Although the construction is rather standard since [14], we give here, for the sake of completeness, a procedure to build an unfolding $\mathcal{U}(\mathcal{N})$ of an SLPN $\mathcal{N}$. We first define the notion of co-set and cut. A *co-set* of a branching process $BR = (ON, \mu, \lambda)$ with $ON = (B, E, \hat{F}, Cut_0)$ is a set of conditions that are pairwise concurrent. A maximal co-set (w.r.t. set inclusion) is called a *cut*. Finite configurations, cuts and markings are related as follows. If C is a configuration of a branching process $BR = (ON, \mu, \lambda')$, then we can define the co-set $Cut(C) = (Min(ON) \cup C^\bullet) \setminus {}^\bullet C$. The set of places in $Cut(C)$ represents the marking reached after firing transitions in $\mu(C)$ in an order compatible with the ordering prescribed by ON.

The construction of an unfolding of a net $\mathcal{N} = (P, T, F, M_0)$ consists in iteratively extending an initial branching process of $\mathcal{N}$. For convenience, we assume a dummy event $\perp$, whose postset fills all places of M_0. A *condition* of a branching process built by unfolding $\mathcal{N}$ is of the form $b = (e, p)$ where $p \in P$ is such that $\mu(b) = p$ and e is the (unique) input event of the condition b. Similarly, events are of the form $e = (X, t)$ where X is a set of conditions (and more precisely a co-set) and t the transition such that $\mu(e) = t$. One can notice that with these definitions of events and conditions, the flow relation in an unfolding is implicit : for an event $e = (X, t)$ and a condition $b = (e', p)$, $b \in {}^\bullet e$ iff $b \in X$, and $e \in {}^\bullet b$ iff $e' = e$. A *possible extension* of a branching process BR is an event (X, t), where $t \in T$ and X is a co-set such that $\mu(X) = {}^\bullet t$ and which does not belong to BR.

The initial branching process of the unfolding algorithm is $BR_0 = (ON_0, \mu_0, \lambda_0)$, where $ON_0 = (B_0, E_0, F_0)$, $B_0 = \{(\perp, p) \mid M_0(p) = 1\}$, $E_0 = \emptyset$, $F_0 = \{(\perp, b) \mid b \in B_0\}$, $\mu_0((\perp, p)) = p$. The following steps are then iterated to produce $BR_{i+1} = (B_{i+1}, E_{i+1}, F_{i+1}, \mu_{i+1}, \lambda_{i+1})$ from $BR_i = (B_i, E_i, F_i, \mu_i, \lambda_i)$:

1) find the set PE of possible extensions of BR_i;
2) if PE is not empty, choose a particular event $e = (X, t)$;
3) $E_{i+1} = E_i \cup \{e\}$
   $B_{i+1} = B_i \cup X'$ with $X' = \{(e, p) \mid p \in t^\bullet\}$
   $F_{i+1} = F_i \cup (X \times \{e\}) \cup (\{e\} \times X')$
   μ_{i+1} extends μ_i by $\mu_{i+1}(e) = t$ and for any $b = (e, p) \in X'$, $\mu_{i+1}(b) = p$
   λ_{i+1} extends λ_i by $\lambda_{i+1}(e) = \lambda(t)$.

6.3 Proof of Theorem 11

Theorem 11: *The HyPOL model checking problem for labeled safe Petri nets is undecidable.*

Proof. We reuse the encoding of PCP from the proof of Theorem 5, and build a labeled safe Petri net whose behaviors (processes) are exactly concatenations of the templates used in the formula ϕ_I associated with an instance I of PCP. For such an instance $I = (x_1, y_1), \ldots, (x_n, y_n)$, we compute a safe (labeled) Petri net $\mathcal{N}_I$ such that $\mathcal{N}_I$ satisfies ϕ_I iff there is a solution for I. We create an initially marked place p_0, and for $i \in 1..n$ a place p_{start_i}, a transition $t_{start,i}$ labeled by $start_i$, a pair of transitions $t_{end_i,1}, t_{end_i,2}$ labeled by end_i and a pair of transitions $t_{\sharp_i,1}, t_{\sharp_i,2}$ labeled by $\sharp$. Then, for every word $x_i = x_{i,1} \ldots x_{i,\ell_i}$ we create a set of places $p_{x_i,0}, \ldots p_{x_i,\ell_i}$ and a set of transitions $t_{x_i,1}, \ldots, t_{x_i,\ell_i}$, respectively labeled by the letters $x_{i,1}, \ldots, x_{i,\ell_i}$. We repeat this operation for each y_i, creating places $p_{y_i,0}, \ldots p_{y_i,h_i}$ and transitions $t_{y_i,1}, \ldots, t_{y_i,h_i}$. Last, we create places $p_{end_i,1}, p_{end_i,2}$.

The flow relation is the following, for each $i \in 1..n$:

- ${}^\bullet t_{start,i} = \{p_0\}$, $t_{start,i}{}^\bullet = \{p_{x_i,0}, p_{y_i,0}, p_{start_i}\}$, ${}^\bullet t_{end_i,1} = {}^\bullet t_{end_i,2} = \{p_{x_i,\ell_i}, p_{y_i,h_i}\}$, $t_{end_i,1}{}^\bullet = \{p_{end_i,1}, p_{end_i,2}\}$, $t_{end_i,2}{}^\bullet = \{p_0\}$;
- ${}^\bullet t_{\sharp_i,1} = \{p_{end_i,1}\}$; $t_{\sharp_i,1}{}^\bullet = \emptyset$, ${}^\bullet t_{\sharp_i,2} = \{p_{end_i,2}\}$, $t_{\sharp_i,2}{}^\bullet = \emptyset$;
- For all $j \in 1..\ell_i$, $k \in 1..h_i$, ${}^\bullet t_{x_i,j} = p_{x_i,j-1}$, $t_{x_i,j}{}^\bullet = p_{x_i,j}$, ${}^\bullet t_{y_i,k} = p_{y_i,k-1}$, $t_{y_i,k}{}^\bullet = p_{y_i,k}$.

Figure 7 shows a part of $\mathcal{N}_I$ where the first pair of I is $(x_1 = aab, y_1 = ab)$. We observe that at any time, tokens circulate only in one of the subparts (corresponding to some $i \in 1..n$) of the net located between place p_0 and transitions $t_{end_i,2}$, $t_{\sharp_i,1}$ and $t_{\sharp_i,2}$. Transition $t_{end_i,2}$ represents the addition of a PCP domino that is not the last one, while $t_{end_i,1}$ corresponds to the last PCP domino (since no other event can occur after firing $t_{sharp,1}$ and $t_{\sharp,2}$). Clearly, processes of $\mathcal{N}_I$ have the shape of concatenations of PCP words encoded with the templates in the proof of Theorem 5. Thus, $\mathcal{N}_I$ satisfies ϕ_I iff there is a solution for instance I of PCP, and model checking HyPOL on safe Petri nets is undecidable. ◀

6.4 Construction of an hyperarc replacement grammar for $\mathcal{U}(\mathcal{N})$

We show how to build a graph grammar that generates the unfolding $\mathcal{U}(\mathcal{N})$. This allows us to prove proposition 1.

Proposition 1. *Let $\mathcal{N}$ be a safe labeled Petri net. Then, there exists a hyperedge replacement grammar $\mathcal{G}_\mathcal{N}$ that generates $\mathcal{U}(\mathcal{N})$.*

▶ **Definition 21.** A *hyperarc* is a pair (l, V), where l is a label, and $V \subseteq \mathbb{N}$ is an ordered set of vertices. A *hypergraph* is a triple (V, E, H) where V is a set of vertices, E a set of edges, and H a set of hyperarcs. A *hyperedge replacement grammar* (HRG) is defined as a pair $\mathcal{G} = (Ax, \mathcal{R})$, where Ax is a hypergraph called the axiom of the grammar and $\mathcal{R}$ is a set of rules. A grammar *rule* is a pair (L, R) where L, the left part of the rule is a hyperarc, and R, the right part of the rule is a hypergraph that contains all vertices of L.

Let $G = (V, E, H)$ be a hypergraph and $h = (l_h, V_h) \in H$ a hyperarc. Let $r = (L, R)$ be a rule where $L = (l, X)$ is a hyperarc with label $l = l_h$ and the same number of vertices as V_h, and $R = (V_R, E_R, H_R)$. The application of rule r to G simply replaces hyperarc h in G by the right part R. More formally, application of r produces a hypergraph $G' = (V', E', H')$ with $V' = V \uplus (\alpha(V_R) \setminus X)$, $E' = E \uplus \alpha(E_R)$ and $H' = H \setminus \{h\} \uplus \alpha(H_R)$, where $\alpha : \mathbb{N} \to \mathbb{N}$ is a map that associates with the j^{th} vertex of X the identity of the j^{th} vertex in V_h, and associates with vertices of $V_R \setminus X_R$ a fresh identity that does not appear in V. We denote by

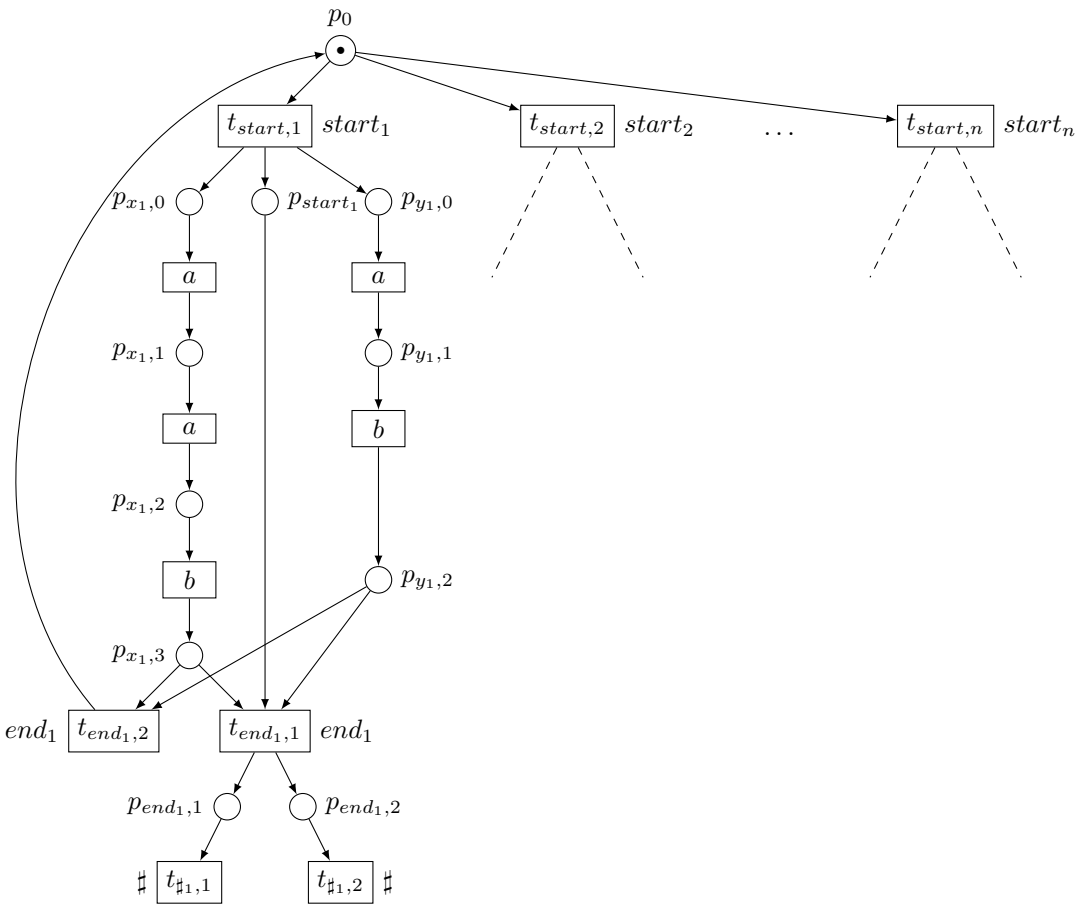

Figure 7 A safe labeled Petri net where processes encode trials of PCP solutions.

$G \xrightarrow{r} G'$ this rewriting step, and by $\mathcal{G}^\omega(G)$ the (possibly infinite) limit graph obtained by application of rules of grammar $\mathcal{G}$ on G.

Let $\mathcal{N} = (P, T, F, M_0, \lambda)$ be a safe labeled Petri net. We fix an arbitrary order $<_P$ on places. Given a marking M, and a set of integers $1 \dots |M|$, we denote by $index(p, M) \in 1 \dots |M|$ the rank of place p in the sequence of integers representing marked places in M. Similarly, given a marking M and a list of integers representing this marking, we denote by $place(i)$ the place represented by index i.

We have seen in section 6.2 an algorithm to build inductively an unfolding of a safe Petri net $\mathcal{N}$. This unfolding can be infinite, but exhibits a regular structure. Furthermore, many verification algorithms addressing reachability of coverability questions work on a structure called a *complete finite prefix*. A complete finite prefix is built inductively as an unfolding, but stops within a finite number of steps, according to some criterion, that forbids the addition of events fulfilling some properties. A stopping criterion frequently met is the reachability criterion: it forbids a possible extension if adding the considered event produces a configuration that ends in a marking that was already visited in the branching process [23]. These events are called *cut-off events*. The principle of the HRG construction described hereafter is to build a complete finite prefix of net $\mathcal{N}$, to find the markings that can be reached when appending cut-off events to maximal configurations of the prefix. We then use these markings as hyperarcs, and the part of the prefix occurring after the marking as the right part of a grammar rule.

Let us first recall some definitions borrowed from [23]. Let $ON = (B, E, F)$ be an occurrence net and let S be a configuration of ON. We denote by $S^\bullet$ the set of all places that are maximal w.r.t. to this configuration, i.e. the set X of all places such that $\forall p \in X, \forall e \in S, p \notin {}^\bullet e$ and $\forall p \in X, \forall e \in E \setminus S, p \notin e^\bullet$. Let μ be a homomorphism from ON to $\mathcal{N}$. The final state of a configuration $\mathcal{F}(S)$ is the marking $\mu(S^\bullet)$. The local configuration of an event e is the set $\downarrow e$.

Let BR be a branching process. A possible extension e is a *cut-off event* (w.r.t. the reachability criterion) iff there exists another event e' such that $\mathcal{F}(\downarrow e^\bullet) = \mathcal{F}(\downarrow e'^\bullet)$, and $|e'^\bullet| < |e^\bullet|$.

Now, the algorithm to compute a complete finite prefix is the following:

0) Start from the initial branching process BR_0
1) find the set PE of possible extensions of BR_i, i.e. the fresh pairs (X, t) such that X is a co-set of BR and $\mu_i(X) = {}^\bullet t$;
2) Compute $NE = \{pe \in PE \mid pe \text{ is not a cut-off event}\}$
3) while NE is not empty,
4) choose a particular event $e = (X, t)$ in NE
5) $E_{i+1} = E_i \cup \{e\}$
   $B_{i+1} = B_i \cup X'$ with $X' = \{(e, p) \mid p \in t^\bullet\}$
   $F_{i+1} = F_i \cup (X \times \{e\}) \cup (\{e\} \times X')$
   μ_{i+1} extends μ_i by $\mu_{i+1}(e) = t$ and for any $b = (e, p) \in X'$, $\mu_{i+1}(b) = p$.
   λ_{i+1} extends λ_i by $\lambda_{i+1}(e) = \lambda(t)$.
6) compute the set PE of possible extensions of BR_{i+1};
7) Compute $NE = \{pe \in PE \mid pe \text{ is not a cut-off event}\}$
8) endwhile

It is well known (see for instance [23]) that:

- the construction of a complete finite prefix w.r.t. the reachability criterion terminates,

- all cuts of the prefix (and in fact even all those of the unfolding) correspond via μ to a reachable marking, and
- conversely, all reachable markings of an unfolded net are represented by at least one cut in the prefix.

Let us call $CFP(\mathcal{N})$ the complete finite prefix thus built; then for every reachable marking M of $\mathcal{N}$, there exists a configuration S of $CFP(\mathcal{N})$ such that $\mathcal{F}(S^\bullet) = M$.

We can now detail the construction of a HRG that generates the unfolding of $\mathcal{N}$. We first build $CFP(\mathcal{N})$ using the algorithm above. Then, we compute the set PE of possible extensions in $CFP(\mathcal{N})$, and add these possible extensions to $CFP(\mathcal{N})$. Let $BR_{CFP,PE}$ be the branching process obtained by adding these events, and let $S_1, \dots S_k$ be the maximal configurations of $BR_{CFP,PE}$. For every S_i there exists at least one configuration S'_i of $CFP(\mathcal{N})$ such that $\mathcal{F}(S_i{}^\bullet) = \mathcal{F}(S'_i{}^\bullet)$. Note that for the reachability cut-off criterion, there can be more than one configuration of this form. We can choose arbitrarily one of them, for instance the configuration with the minimal number of events. For such a configuration S'_i we denote by $\uparrow_{BR_{CFP,PE}} S'_i$ the restriction of $BR_{CFP,PE}$ to events and conditions that are descendants of $S'_i{}^\bullet$.

We build the grammar $\mathcal{G}_\mathcal{N} = (Ax, \mathcal{R})$ as follows. We set $Ax = (N_0, H_0)$ where $N_0 = BR_{CFP,PE}$ and $H_0 = \{(l_i, X_i) \mid S_i \text{ is a maximal configuration of } BR_{CFP,PE}\}$ where each X_i is an ordered set of vertices containing all conditions in $S_i{}^\bullet$ (we can order vertices according to $<_P$ and according to the place $\mu(b)$ represented by each condition b in X_i).

Then, for every maximal configuration S_i in $BR_{CFP,PE}$, we create a rule $r_i = (L_i, R_i)$ where L_i is a hyperarc $L_i = (l_i, 1 \dots |S_i{}^\bullet|)$, and $R_i = (V_i, E_i, H_i)$, where (V_i, E_i) is a copy of $\uparrow_{BR_{CFP,PE}} S'_i$, in which conditions in S'_i are numbered $1 \dots |S'_i{}^\bullet|$. Last, H_i is the set of hyperarcs of the form $h = (l_i, X_i)$, where X_i is a set of conditions contained in $E_i \cap BR_{CFP,PE}$.

One can notice that $\mathcal{G}_\mathcal{N}$ may have up to $2^{|P|}$ rules. We can show that $\mathcal{G}^\omega_\mathcal{N}(Ax) = \mathcal{U}(\mathcal{N})$.

6.5 Proof of Proposition 2

Proposition 2. *There exist labeled safe Petri nets and observation functions whose execution graphs are not of bounded treewidth, and cannot be represented by a hyperedge replacement grammar.*

Proof. This proposition is proved by exhibiting a simple example, whose execution graph contains grid minors of arbitrary sizes.

A *grid* of size $m \times m$ is a graph which vertices are words from $\{0,1\}^n$ with $n \in 0..m$ and such that for any such word w, there is an edge from w to $w.1$ and from w to $w.0$. A *minor* of a graph $G = (V, E)$ is a graph G' obtained by either removing vertices, removing edges, or collapsing vertices that are connected. A famous theorem by Robertson and Seymour [26] says that a family of graphs FG has bounded treewidth if and only if there exists a constant m such that no graph $G \in FG$ has a minor isomorphic to the $m \times m$ grid. It is also known that Hyperedge Replacement Grammars can only generate graphs of bounded treewidth (see for instance [11]).

Let us consider the example net $\mathcal{N}$ of Figure 8, labeled by $\lambda(e) = \lambda(f) = a$, $\lambda(u_1) = \lambda(u_2) = b$ and an observation function $\mathcal{O}_{e,f}$ that projects processes of this net on events labeled by a. Notice that all processes generated by these nets provide a total ordering on events (the execution graph of $\mathcal{N}$ is hence a tree). Within this tree, all events labeled a located at the same depth are equivalent w.r.t. $\mathcal{O}_{e,f}$.

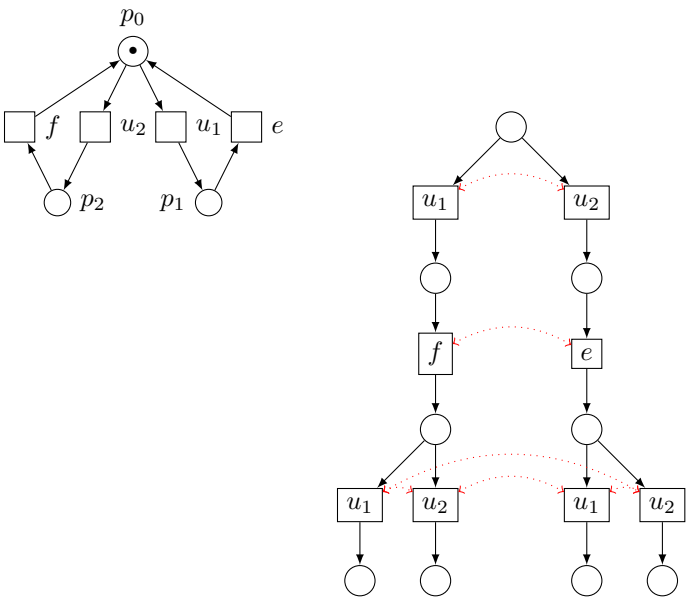

Figure 8 A simple labeled safe Petri net

Clearly, the execution graph of $\mathcal{N}$ has the graph of Figure 9 below as minor (it suffices to collapse conditions with their predecessors, and to keep only nodes corresponding to transitions u_1 and e if they are successors of an occurrence of e. Last, ordering vertices according to lexicographical order $e < u_1 < f < u_2$ we keep only equivalence edges that go from right to left.

One can clearly see from this picture that a grid of arbitrary size n can be created by first removing all events at depth $\leq 2 \cdot n$, then collapsing every occurrence of u_1, u_2 with its predecessor, and then removing the leftmost part of the grid to have only red chains of at most n event. As a consequence, the execution graph of $\mathcal{N}$ has all grids of size $n \times n$ as minors. Following the results of Robertson & Seymour, it is then not of bounded treewidth. It is well known that context-free hyperedge replacement grammars and equational graphs are alternative definitions for families of graphs [11]. So, an HRG can only generate graphs of bounded treewidth. Hence, there is no context free HRG generating the execution graph for the example considered in Figure 8. ◀

6.6 Proof of Proposition 4

Proposition 4. *Let ϕ be a HyPOL formula. Then there exists an MSO formula ψ such that $\mathcal{N} \models \phi$ iff $G_{\mathcal{U}(\mathcal{N})} \models \psi$*

Proof. Without leaving MSO, we can define a particular labeling to differentiate events and conditions in $G_{\mathcal{U}(\mathcal{N})}$: We write $Cond(x)$ for the predicate that holds for every condition and $Event(x)$ for the predicate that holds on all events.

We first define some basic formulas, holding at some node of $G_{\mathcal{U}(\mathcal{N})}$:

- *true* holds for every element of $G_{\mathcal{U}(\mathcal{N})}$;
- $Lab(x) \cap D \neq \emptyset$ is equivalent to the formula $\bigvee_{d \in D} lab_d(x)$;
- $Event(x)$ holds under any interpretation that assigns an event of $G_{\mathcal{U}(\mathcal{N})}$ to x;

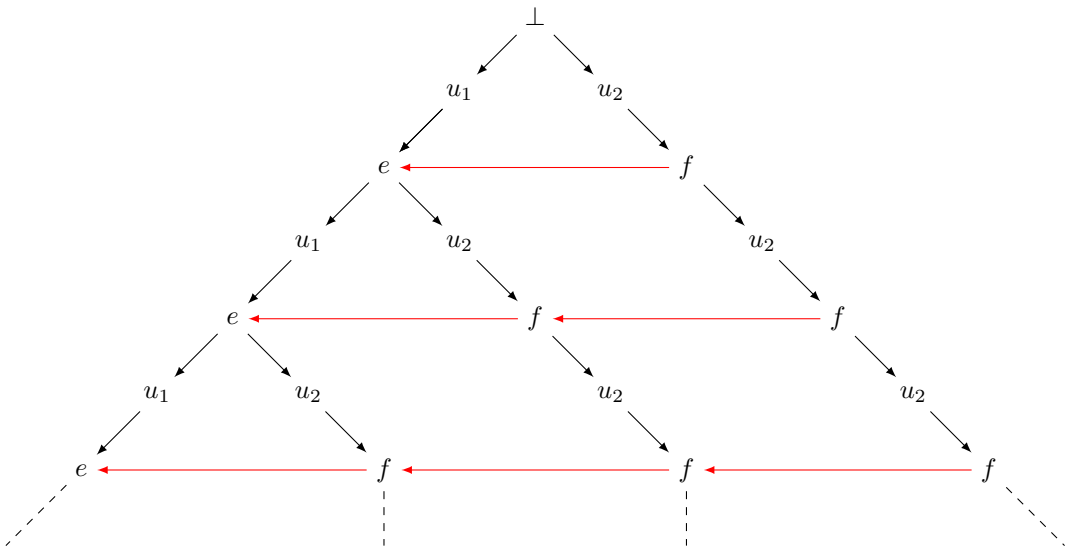

Figure 9 A part of minor of the execution graph for the net of Figure 8 with observation function $\mathcal{O}_{e,f}$ that projects processes on events labeled by a.

- $Cond(x)$ holds under any interpretation that assigns a condition of $G_{\mathcal{U}(\mathcal{N})}$ to x;
- $edge(x, y)$ holds under an interpretation that assigns a condition b to x and an event e to y, and such that $b \in {}^\bullet e$, or an event e to x and a condition b to y such that $b \in e^\bullet$;
- $edge_i(x, y)$ holds under any interpretation $\mathcal{I}$ that assigns events $\mathcal{I}(x)$ and $\mathcal{I}(y)$ of $G_{\mathcal{U}(\mathcal{N})}$ to x and y and such that $\mathcal{I}(x) \xrightarrow{i} \mathcal{I}(y)$.

From these building blocks, we can define more advanced expressions.

- $succ(x, y)$ is a formula that holds under an interpretation $\mathcal{I}$ such that $e = \mathcal{I}(x)$ is an event, $f = \mathcal{I}(y)$ is an event, and the pair of events e, f is in immediate successor relation in $G_{\mathcal{U}(\mathcal{N})}$. Formally, this is written as:
$succ(x, y) ::= \exists z, Event(x) \wedge Event(y) \wedge Cond(z) \wedge edge(x, z) \wedge edge(z, y)$.
- $isMinimal(x, X)$ is a formula that holds under an interpretation that maps variable x to an event, X to a set of nodes of $G_{\mathcal{U}(\mathcal{N})}$, and such that $\mathcal{I}(x)$ is minimal in X with respect to the causal ordering of $G_{\mathcal{U}(\mathcal{N})}$. Formally, we write:
$isMinimal(x, X) ::= x \in X \wedge Event(x) \wedge \nexists y \in X, succ(y, x)$.
- $isMaximal(x, X)$ is similar to the previous formula, and requires $\mathcal{I}(x)$ to be maximal in X. It is defined as: $isMaximal(x, X) ::= x \in X \wedge Event(x) \wedge \nexists y \in X, succ(x, y)$
- $isAChain(x, X)$ is a formula that holds for any interpretation $\mathcal{I}$ in which X is a chain (a totally ordered sequence of events w.r.t. the successor relation) starting from x. It is formulated as follows:

$$\begin{array}{rl} & isMinimal(x, X) \wedge \forall y \in X, \\ isAChain(x, X) ::= & (isMinimal(y, X) \Longrightarrow x = y) \wedge \\ & (\exists z \in X, succ(y, z)) \Longrightarrow (\nexists z' \in X, z \neq z' \wedge succ(y, z')) \end{array}$$

- $x \leq y$ can be defined as the formula:

$$\begin{array}{rl} & Event(x) \wedge Event(y) \wedge \exists X, x \in X \wedge y \in X \\ x \leq y ::= & \wedge \forall z \in X, succ(z, z') \Longrightarrow z' \in X \\ & \wedge \forall u \in X, \nexists u', succ(u', u) \Longrightarrow u = x \end{array}$$

More intuitively, this formula says that $\mathcal{I}(X)$ is the set of all successors of $\mathcal{I}(x)$ in $G_{\mathcal{U}(\mathcal{N})}$, and it contains $\mathcal{I}(y)$. This is a standard formula frequently used when addressing properties of partially ordered sets.

- $x < y$ (covering) is defined by $x < y ::= x \leq y \wedge \nexists z, z \neq x, z \neq y, x \leq z \wedge z \leq y$.
- Let $\mathcal{O}$ be a particular observation erasing events that do not carry a label from a particular subset D, and restrict covering of the obtained order to pairs of events carrying specific pairs of labels in $R \subseteq \Sigma \times \Sigma$. Then one can define $x <_{\mathcal{O}} y$ as the formula stating that the labels attached to x and y are contained in D, that $(lab(x), lab(y)) \in R$, that there exists a path from x to y such that every intermediate event visited between x and y carries a label that does not belong to D. This type of construction applies for all kind of labeling-based projection and order restriction.
  More formally :

$$x <_{\mathcal{O}} y \quad ::= \quad \begin{array}{l} Event(x) \wedge Event(y) \wedge Lab(x) \cap D \neq \emptyset \wedge Lab(y) \cap D \neq \emptyset \\ \wedge\, x \leq y \\ \wedge\, \forall z, x < z \wedge z < y \Longrightarrow Lab(z) \cap D = \emptyset \\ \wedge \bigvee\limits_{(a,b) \in R} lab_a(x) \wedge lab_b(y) \end{array}$$

We are now ready to transform HyPOL formulas into MSO formulas. For every hypol formula ϕ we will build inductively an MSO formula ψ. The inductive construction will use fresh first order variables $x, y, ...$ and second order variables $X, Y, ...$ at every induction step. Further, as HyPOL formulas should hold at a particular event, we will design ψ with a particular free variable x depicting the event at which ψ must hold. For every HyPOL formula ϕ, letting ψ be the MSO formula obtained by translation of ϕ into MSO, for every order O in $\mathcal{O}rd(PR(\mathcal{N}))$ and every event $e \in E_O$, $O, e \models \phi$ if and only if ψ holds in $G_{\mathcal{U}(\mathcal{N})}$ under an interpretation that assigns e to x. We hence define $\psi = MSO(\phi, x, C)$ where C is a context listing variable names already used, x is a free variable in ψ, that appears in C, and ψ is an MSO formula over x and fresh variable names not used in C. For a given HyPOL formula ϕ, we build inductively $\psi = MSO(\phi, x, C)$ as follows:

- if $\phi = true$ then $MSO(\phi, x, C) = true$ for any variable x and context C;
- if $\phi = \neg\phi'$ then $MSO(\phi, x, C) = \neg(MSO(\phi', x, C))$;
- if $\phi = \phi_1 \wedge \phi_2$ then $MSO(\phi, x, C) = MSO(\phi_1, x, C) \wedge MSO(\phi_2, x, C)$;
- if $\phi = EX_{D,\mathcal{O}}\ \phi'$ then $MSO(\phi, x, C) = \exists y, x \leq_{\mathcal{O}} y \wedge MSO(\phi, y, C')$ where y is a fresh variable name (w.r.t. C and to the set $C_{x \leq_{\mathcal{O}} y}$ of variables used to encode subformula $x \leq_{\mathcal{O}} y$) and $C' = C \cup \{y\} \cup C_{x \leq_{\mathcal{O}} y}$;
- if $\phi = match(\mathcal{O}, T, f)$ where $T = (E, <_T, \lambda_T)$, with $E = \{f\} \cup \{e_1, e_{|E|-1}\}$ then
  $MSO(\phi, x, C) = \exists x_1, \ldots x_{|E|-1}, \quad \bigwedge\limits_{(f,e_i) \in <_T} x <_{\mathcal{O}} x_i \wedge \bigwedge\limits_{(e_i,e_j) \in <_T} x_j <_{\mathcal{O}} x_i \wedge \bigwedge\limits_{i \in 1..|E|-1} Lab(x_i) \supseteq \lambda_T(x_i)$
  where $x_1, \ldots x_{|E|-1}$ are fresh variable names (w.r.t. C);
- if $\phi = EX_{\equiv,\mathcal{O}_i}\ \phi'$ then
  $MSO(\phi, X, C)) = \exists y, edge_i(x, y) \wedge MSO(\phi', y, C')$
  where y is a fresh variable name (w.r.t. C) and $C' = C \cup \{y\}$;
- if $\phi = \phi_1\ EU_{D,\mathcal{O}}\ \phi_2$ then
  $MSO(\phi, x, C) = \exists X, \quad isAChain(x, X) \wedge \forall y \in X, \exists y', y <_{\mathcal{O}} y' \Longrightarrow MSO(\phi_1, y, C') \wedge \nexists y', y <_{\mathcal{O}} y' \Longrightarrow MSO(\phi_2, y, C')$

where y, y', X are fresh variable names (w.r.t. C and to the sets $C_{y,y'}$ and C_{chain} of variables used to encode respectively formulas $y <_{\mathcal{O}} y'$ and $isAChain(x, X)$) and $C' = C \cup \{y\} \cup C_{y,y'} \cup C_{chain}$;

- if $\phi = EG_{D,\mathcal{O}}\, \phi'$ then $MSO(\phi, x, C) = \exists y, Event(y) \wedge x <_{\mathcal{O}} y \wedge MSO(\phi', y, C')$ where y is a fresh variable name (w.r.t. C) and $C' = C \cup \{y\}$.

We have assumed that the unfolding of $\mathcal{N}$ has a unique starting event denoted by $\perp$ and carrying label $\perp$. We can prove by induction on the length of HyPOL formulas that $\mathcal{N} \models \phi$ iff $G_{\mathcal{U}(\mathcal{N}} \models \exists s, x, lab_{\perp}(s) \wedge succ(s, x) \wedge MSO(\phi, x, \{s, x\})$. ◀

6.7 Proof of proposition 3

Proposition 3. *Let $\mathcal{N}$ be a $K-$layered safe Petri net. Then, one can effectively compute a hyperedge replacement grammar $\mathcal{G}_{K,\mathcal{N}}$ that recognizes the execution graph $G_{\mathcal{U}(\mathcal{N})}$.*

Proof. We reuse the construction of the graph grammar $\mathcal{G}_{\mathcal{N}}$ in the proof of Proposition 1. Recall that each rule of this grammar is of the form $r = (h, HG)$ where $h = (l, p_1 \dots p_k)$ is a hyperarc, given as a list $p_1 \dots p_k$ of vertices representing conditions, with a label l depicting a marking and the names of places attached to vertices. Also recall that places are ordered according to an arbitrary ordering $<_P$. Moreover, HG is a a hypergraph containing vertices of h, a set of vertices representing new events and conditions generated by the rule, edges between these vertices representing the flow relation in the unfolding, and a list of hyperarcs representing cuts that will be obtained after the firing of events in configurations of HG.

Grammar $\mathcal{G}_{\mathcal{N}}$ is sufficient to build $\mathcal{U}(\mathcal{N})$, i.e. it generates events, conditions, and the flow relation in $\mathcal{U}(\mathcal{N})$. Now, to be able to draw edges in relation $\xrightarrow{i}$ between events that have isomorphic pasts w.r.t. observation $\mathcal{O}_i$, hyperarcs need to memorize events that may belong to the *K-Ball* of any event that will appear in future rewritings in addition to conditions in cuts reached after a rewriting step. Hyperarcs will hence represent a set of conditions, a set of events, but also information on the relation and distance among memorized events.

Preliminary notations. Slightly overloading the definition of hyperarc proposed for the construction of $\mathcal{G}_{\mathcal{N}}$, we will now consider that a hyperarc is of the form $(l, p_1.p_2 \dots p_n.e_1.e_2 \dots e_{n'})$, i.e. a hyperarc is still a list of vertices, but they represent ordered lists of conditions and events. Note that the number of places and events in the hyperarc is known, as well as their types (the place occurrence and the transition occurrence they represent) so the partition between conditions and events in the vertices list is clear. This information can be encoded in label l.

Let HG be a hypergraph, that contains hyperarcs of the form $(l, p_1, \dots p_n.e_1 \dots e_{n'})$, such that the grammar for the unfolding contains a rule to expand unfolding from marking $p_1, \dots p_n$. Let us assume that an event $e = (X, t)$ is appended to the unfolding by this rule. To decide whether an equivalence edge for observation $\mathcal{O}_i$ must be written between e and an event $e_i \in \{e_1 \dots e_{n'}\}$, additional information will be memorized in the label l: First, the relation among events appearing in the hyperarc. Second, their distance (up to a certain bound), and last, the shape of their common past (again up to a bounded distance).

For each pair of event e_i, e_j in a hyperarc, one can easily memorize whether e_i, e_j are causally related, conflicting, or concurrent. We define a function $rel : E \times E \to \{cr, cf, con, none\}$, with the convention that $rel(e_i, e_j) = cr$ if $e_i \leq e_j$ in $\mathcal{U}(\mathcal{N})$, $rel(e_i, e_j) = cf$ if e_i and e_j are conflicting events, $rel(e_i, e_j) = con$ if e_i and e_j are concurrent events, and $rel(e_i, e_j) = none$ otherwise.

We assume K fixed, and for each pair of events (e, f) in a hyperarc, we define $\mathsf{dist}_{\leq K}(e, f)$ as $\mathsf{dist}_{\leq K}(e, f) = \mathsf{dist}(e, f)$ if $\mathsf{dist}(e, f) \leq K$ and $\mathsf{dist}_{\leq K}(e, f) = \infty$ otherwise.

Process graphs. To give a representation of common pasts of events, we define *process graphs* as graphs of the form $G = (V, \longrightarrow, \alpha)$ representing processes of a Petri net, where $V = \{1 \dots n\}$ is a finite set of integers, $\longrightarrow \subseteq V \times V$, and $\alpha : V \times P \cup T$ associates a place or transition name to each vertex in V. We denote by $PG(P,T)$ the set of all such process graphs. We will use process graphs to memorize a canonical representation of the (bounded) past of a set of events.

Now, we can define a partial function $\mathit{diff}(e_i, e_j, K) : E \times E \times \mathbb{N} \to PG(P,T)$, such that $\mathit{diff}(e_i, e_j, K)$ is defined only for pairs of events such that $d_K(e_i, e_j) < \infty$. The process graph $\mathit{diff}(e_i, e_j, K)$ is isomorphic to the part of the unfolding defined by $({}^\bullet(Past(e_i) \setminus Past(e_j)))^\bullet$. One can notice that with the implicit flow relation due to events construction ($(b,e) \in F$ if $e = (X,t)$ and $b \in X$ and $(e,b) \in F$) if $b = (e,p)$, every vertex in an unfolding has a finite set of predecessors, and $\mathit{diff}(e_i, e_j, K)$ is always a finite graph. Given an occurrence net ON, a sequence of events $e_1, \dots e_n$, and a pair of events e_i, e_j at distance at most K, we can compute $P_K(e_i, e_j)$ the set of conditions and events that appear in the past of e_i and not in the past of e_j at distance smaller than K. Let $P_K(e_1 \dots e_n)$ denote the union of all $P_K(e_i, e_j)$ for all pairs of events e_i, e_j in $e_1, \dots, e_n$. Note that this is not yet a canonical representation. However, one can easily attach a canonical identity (an integer) to each event or condition in $P_K(e_1 \dots e_n)$. Considering an arbitrary ordering on places, transitions, events $e_1, \dots e_n$ and pairs of events, one can define a backward DFS exploration starting from an event e_i in $P_K(e_i, e_j)$, and associate as canonical identity to a condition or event in $P_K(e_1 \dots e_n)$ the integer indicating the order of discovery during the exploration. Let $can(v_i)$ be the function associating to a condition or event its canonical number. Then, $\mathit{diff}(e_i, e_j, K) = (V \longrightarrow, \alpha)$ is the process graph obtained by first computing $P_K(e_i, e_j) = (B, E, F, Cut_0)$, and then defining $V = can(B) \cup can(E)$, $(n, n') \in \longrightarrow$ iff $\exists (b,e)$ in F such that $can(b) = n$ and $can(e) = n'$ or $\exists (e,b)$ in F such that $can(e) = n$ and $can(b) = n'$. Last, we define $\alpha(n) = p$ if $can(b) = n$ for some condition $b = (e,p)$ and $\alpha(n) = t$ if $can(e) = n$ for some event $e = (X,t)$. As we define $\mathit{diff}(e_i, e_j, K)$ for events that are at distance at most K, this process is always finite. Notice that we are not interested in processes themselves, but in their general shape (that will be sufficient to decide isomorphism in unfoldings of layered nets).

Rule construction. We are now ready to define the construction of a rule from the grammar that generates $\mathcal{U}(\mathcal{N})$. The principle is as follows. Consider a hyperarc $h = (l, p_1 ... p_k . e_1 .. e_n)$ of our new grammar. We assume that this hyperarc was correctly built, i.e. the label l attached to the hyperarc is such that $p_1 \dots p_k$ can be distingushed from $e_1 \dots e_n$, it associates a place name with each p_i (i.e. one can recover the marking M_i associated to cut $p_1 \dots p_k$), a transition name to each e_i, and functions rel and diff.

We use the rule $r_i = (L_i, R_i)$ of grammar $\mathcal{G}_\mathcal{N}$ with hyperarc $L_i = (l_i, p_1 ... p_k)$ such that the marking described by L_i is the same as the marking described by h. This rule is unique, and as our new rule only adds events to cuts already used in $\mathcal{G}_\mathcal{N}$, it exists. As the execution graph does not add new events nor conditions to the unfolding of $\mathcal{N}$ we can safely use the right part R_i of rule r_i to add events, conditions, and the flow relation to an already built part of $G_\mathcal{U}(\mathcal{N})$ containing hyperarc h. Let V_{R_i} denote the set of events appended by application of rewriting rule r_i, B_{R_i} the set of conditions and E_{R_i} the set of edges. For every pair of events e, f in the set of events $V_{R_i} \cup \{e_1 .. e_n\}$ we can decide whether $e \xrightarrow{i} f$ as follows:

Suppose $e, f \in e_1 \dots e_n$: then, every equivalence arc that needed to be appended is already drawn, and no additional edge needs to be appended between e and f. Suppose $e \in e_1 \dots e_n$ and $f \in V_{R_i}$. Let $P_f = \{e_1 \dots e_n\} \cap \downarrow f$. Then, one can compute an occurrence net $ON_i = (B_i, X_i, F_i)$ where X_i is the set of events appearing in $\bigcup_{x \in P_f} \mathit{diff}(x, e) \cup \bigcup_{x \in P_f} \mathit{diff}(e, x)$, $B_i = {}^\bullet(X_i) \cup (X_i)^\bullet$. Then, for every x in X_i, one can decide if the distance between x and

f is greater than K, and if it is smaller compute it: if $dist(x,f) > K$ for every $x \in P_f$ then $dist(e,f) > K$. As $\mathcal{N}$ is $K-$layered there is no edge of the form $\xrightarrow{i}$ between e and f. If $\mathsf{dist}(x,e) \leq K$, then $\mathsf{dist}(e,f)$ can also be computed. Assume that there exists $x \in P_f$ such that $e \in \downarrow x$. Then $Past(e) \cap Past(f) = Past(e)$ and $\mathsf{dist}(e,f) = \max_{x \in P_f}(\mathsf{dist}(e,x) + \mathsf{dist}(x,f))$. As the two values $\mathsf{dist}(x,e)$ and $\mathsf{dist}(x,f)$ are known, one can easily check whether $\mathsf{dist}(e,f) \leq K$ and if this is the case, compute the frontier H as the set of predecessors of the maximal places in $(\downarrow e)^\bullet$, compute $\hat{F}_{e,f}$ and as $\mathcal{N}$ is $K-$layered check that $\mathcal{O}_i(\downarrow e \backslash \hat{F}_{e,f}) \equiv \mathcal{O}_i(\downarrow f \backslash \hat{F}_{e,f})$ (in ON_i).

Now, assume that for every $x \in P_f$, $e \notin \downarrow x$. Then, for every $x \in P_f$, e is either in conflict or concurrent with x. For every node v_n in $Past_K(x)$, the past of x at distance at most K either the distance $\mathsf{dist}(v_n,e)$ between v_n and e is already greater than K, or we know precisely the distance $\mathsf{dist}(v_n,e) \leq K$. The distance between e and f is hence $\mathsf{dist}(e,f) = \max_{x \in P_f, v_n \in Past_K(x)}(\mathsf{dist}(v_n,f) + \mathsf{dist}(v_n,e))$. Like for the case $e \leq x$ we can compute H, $\hat{F}_{e,f}$ from ON_i, and check $\mathcal{O}_i(\downarrow e \setminus \hat{F}_{e,f}) \equiv \mathcal{O}_i(\downarrow f \setminus \hat{F}_{e,f})$

We do not detail the case where e, f are newly generated events, that is similar to the former situation where $e \in e_1 \dot{e}_n$ and $f \in V_{R_i}$, with the slight differences that distances have to consider predecessors of both e and f. Hence, starting from an set of conditions $l_1, \dots p_k$ and a set of events in an hyperarc, one can generate the occurrence net ON that contains conditions $p_1, \dots p_k$ in hyperarc, plus additional events and conditions that are obtained by application of a rewriting of the form (L,R) defined in the construction of grammar $\mathcal{G}_\mathcal{N}$, and augment it with equivalence edges.

Generating new hyperarcs. Let us now explain how new hyperarcs are generated. Recall that a hyperarc rewriting by a rule of grammar $\mathcal{G}_{K,\mathcal{N}}$ is a rewriting of a left part of the form $L_i = (l_i, p_1 \dots p_k . e_1 \dots e_n)$ into a right part $HG_i = (G_i, H_i)$, where G_i is a graph containing an occurrence net $ON_i = (V_i, E_i)$ generated by unfolding from marking $p_1 \dots p_k$ using a rule of grammar $\mathcal{G}_\mathcal{N}$ for unfolding (see the grammar construction in section 6.4), and augmented with equivalence edges, and H_i is a set of newly generated hyperarcs. Let us first detail the contents of the hypergraph HG_i. The added conditions and events are conditions and events added by some rewriting rule $r = (L,R)$ of $\mathcal{G}_\mathcal{N}$ with $L = (l, B)$ and $R = (ON, H)$ and such that $B = b_1 \dots b_k$ represents the same marking as $p_1 \dots p_k$, and ON_i (i.e. G_i without equivalence edges) is equal to ON. Note that L is not the full hyperarc L_i, as events and additional information attached to their distance and common past is missing. However, places $p_1 \dots p_k$ suffice to write the needed events and conditions in ON_i, as this additional information is only used to detect whether a pair of events is connected by an equivalence edge. We use a slight shortcut, and call $h_1, \dots h_q$ the hyperarcs obtained by application of rule $r = (L,R)$ from $\mathcal{G}_\mathcal{N}$ to a hypergraph that contains a hyperarc of the form $L_i = (l_i, p_1 \dots p_k . e_1 \dots e_n)$. These hyperarcs represent the maximal places in maximal configurations obtained by unfolding a complete prefix once more. We will use them as a base to design hyperarcs of the form $L_j = (l_j, p_1 \dots p_{k_j} . e_1 \dots e_{n_j})$ and hence complete the construction of rules for $\mathcal{G}_{K,\mathcal{N}}$. As seen before, a hyperarc only needs the information about events at distance $\leq K$ from events produced in the future (and the finite list of considered events to be able to build equivalence edges). That is, for each hyperarc $h_j \in \{h_1, \dots h_q\}$ obtained by application of rule r from $p_1 \dots p_k$, we need to compute:

- $F_{\leq K, h_j}$ the set of events that can be at distance $\leq K$ from events appended in the future when rewriting h_i,
- l_i, the label that associates with places and events enough information to know the type of each node, the relations among them, and the differences since their common causal

past.

Let G_i be the graph generated by rewriting, h_j be a set of maximal conditions in G_i. Let $Place(h_j)$ denote the list of places appearing in h_j, and let t be a transition such that ${}^\bullet t \subseteq Place(h_j)$. Then, a rewriting of a hyperarc with conditions h_j will append to G_i all events of the form $e = (X, t)$ where X is a subset of h_j. Note furthermore that for every event f in G_i, if the distance $\mathsf{dist}(x, f)$ between f and all predecessors x of a condition in h_j is already greater than K then $\mathsf{dist}(e, f) \geq K$. Conversely, if the minimal distance between a predecessor x of a condition in h_j and f is equal to $m \leq K$, then the distance between e and f is $m + 2$ (one needs to cross two additional edges to go from f to e via x). If we repeat this operation for every event e that can be an immediate successor of conditions in h_j, then we can build $F_{\leq K,h_j}$ the set of events that can be at distance $\leq K$ form events that will appear in the future. Then, the causal/conflict/concurrency relation among events in $F_{\leq K,h_j}$ is built from the causal dependences in ON_i and from the existing information in the original rewritten hyperarc $L_i = (l_i, p_1 \dots p_k.e_1 \dots e_n)$. As in the construction of $\mathit{diff}(e, f, K)$ we can find a unique way to number conditions and events in G_i, compute $\mathit{diff}(e, f, K)$ for every pair of events at distance at most K in $F_{\leq K,h_j}$, and integrate this information as a part of a label l_j designed for this hyperarc. We hence have a hyperarc $L_j = (l_j, h_j.F_{\leq K,h_j})$. We repeat this operation for every hyperarc in $\{h_1, \dots h_q\}$. To summarize, we have produced a rule of the form (L_i, R_i) where $L_i = (l_i, p_1 \dots p_k.e_1 \dots e_n)$, and R_i is a hypergraph of the form (G_i, H_i) where G_i is the graph mentioned above obtained by unfolding and decoration with equivalences, and $H_i = \{L_j \mid \exists h_j \in G_{\mathcal{N}}(L_i)\}$ is the set of hyperarcs obtained by adding events and information to hyperarcs of $\mathcal{G}_{\mathcal{N}}$ that rewrite $p_1 \dots p_k$ and are of the form $L_j = (l_j, h_j.F_{\leq K,h_j})$. One can notice that for a given hyperarc of $\mathcal{G}_{\mathcal{N}}$, i.e. for a given marking $M = p_1 \dots p_k$, there is only a bounded number of events that can fire from M. So the *K-Ball* of this set of events is finite, and the set of K predecessors of this union of balls is finite too.

For every newly produced hyperarc, we can reproduce this unfolding and decoration operation to produce new hyperarcs. We compute an axiom as for $\mathcal{G}_{\mathcal{N}}$: we compute a complete finite prefix, decorate it with equivalence edges, and take as hyperarcs the maximal cuts with additional events. We then proceed inductively from each produced hyperarc to build a grammar that generates $G_{\mathcal{U}(\mathcal{N})}$. As the set of hyperarcs in $\mathcal{G}_{\mathcal{N}}$ is finite, as the *K-Ball* of events is finite, and as the set of $\mathit{diff}(e, f, K)$ that can appear for events at distance at most K is also finite, the inductive construction stops.

◀

6.8 Proofs of corollaries 16, 17 and 18

Corollary 16 *It is undecidable whether the execution graph of a net $\mathcal{N}$ satisfies an MSO formula.*

Proof. Assume that MSO is decidable. Then, for each instance I of a PCP, one can build the net $\mathcal{N}_I$ and the formula ϕ_I describing solutions of instance I of the PCP, as proposed in the proof of Theorem 11. As every HyPOL formula can be translated into an MSO formula, checking whether $\mathcal{N}_I \models \phi_I$ is equivalent to checking whether the MSO formula $MSO(\phi_I)$ is satisfied by the execution graph $G_{\mathcal{U}(\mathcal{N}_I)}$. As the PCP is undecidable, this question is undecidable too. ◀

Corollary 17 *Model checking equivalence-free HyPOL properties on labeled safe Petri nets is decidable.*

Proof. Model checking an equivalence-free HyPOL property ϕ of a net $\mathcal{N}$ amounts to model checking individually property ϕ on every process. One can express in MSO that a pair of events is not in conflict, so ϕ can be verified as MSO property $MSO(\phi)$ on the graph grammar $\mathcal{G}_{\mathcal{N}}$ that generates $\mathcal{U}(\mathcal{N})$. ◀

Corollary 18 *HyPOL model checking is decidable for $K-$layered safe Petri nets.*

Proof. MSO is decidable for Hyperedge Replacement Grammars [12, 19]. From proposition 4, we know that we can transform an HyPOL formula ϕ on processes of $\mathcal{N}$ into an MSO formula $MSO(\phi)$, and that $\mathcal{N} \models \phi$ iff $G_{\mathcal{U}(\mathcal{N})} \models MSO(\phi)$. Similarly, if $\mathcal{N}$ is $K-$layered for some K, then one can compute a graph grammar $\mathcal{G}_{K,\mathcal{N}}$ that recognizes $G_{\mathcal{U}(\mathcal{N})}$. ◀

6.9 Proofs for observable nets

Proposition 5: *Let $\mathcal{N} = (P, T, F, M_0, \lambda)$ be a safe labeled observable Petri net for observations $\mathcal{O}_1, \ldots, \mathcal{O}_k$. Then $\mathcal{N}$ is $K-$layered, for some $K \leq \max(2.k_c, 3.|T|)$.*

Proof. Let $\mathcal{N}$ be an observable net, with set of transition T and set of places. One can first notice that in a process of $\mathcal{N}$, if a pair of events e, f is connected by a causal chain of length greater than $|T|$ event f can always be differentiated from event e (and vice versa). Indeed, let $e < x_1 < \ldots x_k < f$ with $k > |T|$, where $x_1, \ldots x_k$ are events. We obviously have $\downarrow e \subseteq \downarrow f$. Furthermore, as $k > |T|$, $\uparrow e \cap \downarrow f$ contains a cyclic behavior, and in particular, as $\mathcal{N}$ is observable, for every observation $\mathcal{O}_i$, $\mathcal{O}_i(e < x_1 < \ldots x_k < f)$ contains at least one observable event. Hence we have that $\mathcal{O}_i(\downarrow e) \neq \mathcal{O}_i(\downarrow f)$. So, an event f in $\mathcal{U}(\mathcal{N})$ is never equivalent to an event e that is located in its causal past at a distance greater that $2.|T|$.

Now, let us consider a pair of events e, f such that e and f are in conflict. Then, there exists a pair of executions ρ_1, ρ_2 that end respectively with e and f, and such that $\rho_1 = \rho.\rho'_1$ and $\rho_2 = \rho.\rho'_2$, i.e. ρ_1 and ρ_2 share a common prefix ρ. We have that $O_\rho \subseteq O_{\rho_1} \cap O_{\rho_2}$. Let us assume that $\rho'_1 > k_c$ and $\rho'_2 > k_c$. Then, as $\mathcal{N}$ is observable, for every observation $\mathcal{O}_i$, $\mathcal{O}_i(O_{\rho_1}) \neq \mathcal{O}_i(O_{\rho_2})$. For a pair of conflicting events, there exists a set of conditions $b_1, \ldots b_k$ that are maximal (w.r.t. to the flow relation) in $\downarrow e \cap \downarrow f$, and such that $\mathcal{H}(e)$ is the maximal length of a path of $\mathcal{U}(\mathcal{N})$:

- 1) from B_0 to e that does not pass through $\downarrow e \cap \downarrow f$ and hence through $b_1, \ldots, b_k$,
- 2) from B_0 to e that passes through $\downarrow e \cap \downarrow f$ and hence through $b_1, \ldots, b_k$,

the length of a path of type 1 is at most $2.(|\rho| + |\rho_1|)$. One can notice that a part of the events listed in ρ are either in $\downarrow e$ and hence must also appear in $\downarrow f$, or are concurrent with e and must also be concurrent with f. So the length of path of type 1 leading from B_0 to e and from B_0 to f differ by at most $2.k_c$. Now let us consider paths of type 2. These paths contain events that belong to $\downarrow e \cap \downarrow f$ and also appear in ρ, events that belong to $\downarrow e \cap \downarrow f$ and appear in ρ'_1 (as ρ_1 and ρ_2 are sequential behaviors, on may still find events in $\downarrow e \cap \downarrow f$ that belong both to ρ_1 and ρ_2). Again, the length of such paths is at most $2.(|\rho| + |\rho_1|)$. Hence, e and f are at a distance at most $2.k_c$ in $\mathcal{U}(\mathcal{N})$. So, an event f in $\mathcal{U}(\mathcal{N})$ is never equivalent to another event e located at conflict distance greater than $2.k_c$ in $\mathcal{U}(\mathcal{N})$.

Last, consider a pair of concurrent events $e = (X_e, t_1), f = (X_f, t_2)$. Clearly, e and f are occurrences of events that belong to independent sets of transitions T_e, T_f. As $\mathcal{N}$ is observable, for every observation $\mathcal{O}_i$, if a cycle ρ containing transitions of T_e and T_f exists, then there is also a pair of transitions t_e, t_f appearing in ρ such that $\mathcal{O}_i(t_e) \neq \mathcal{O}_i(t_f)$. Let T_ρ denote the set of transitions appearing in such cycle ρ. If the distance between e and f is greater than $2.|T|$, then there exists a cyclic behavior of the net. This cyclic behavior

may contain only occurrences of transitions in T_e, only occurrences of transitions in T_f or occurrences of both. Let us assume that the cycle ρ contains occurrences of T_e and T_f. Then as $\mathcal{N}$ is observable, there exists an event $x_e \in\downarrow e$ and an event $x_f \in\downarrow f$ such that $\mathcal{O}_i(x_e) \neq \mathcal{O}_i(x_f)$, and hence e and f cannot be considered as equivalent by $\mathcal{O}_i$. Assume that the cycle contains only occurrences of transitions from T_e. Then, as $\mathcal{N}$ is observable w.r.t. each observation $\mathcal{O}_i$, this cycle contains at least one occurrence of a transition that is observable by $\mathcal{O}_i$. Hence, after two occurrences of cycle ρ, the next occurrence of transition t_1 has a causal past that cannot be equivalent to the causal past of f. Considering cycles that contain only occurrences of transitions in T_f is symmetric. Hence, for every observation $\mathcal{O}_i$, if an event f that is at a concurrency distance greater that $3.|T|$ from an event e, then $\mathcal{O}_i(\downarrow e) \neq \mathcal{O}_i(\downarrow f)$.

It is straightforward that for every $e \in \mathcal{U}(\mathcal{N})$ and for every K, $Ball_K(e)$ is of bounded size, as every event has at most $|P|$ predecessors and $|P|$ successors, and every condition has only one predecessor ans at most $|T|$ successors.

It now remains to prove existence of a frontier H and of a bound K such that H is contained in the K-causal past of e and f, and such that $e \equiv_i f$ iff $\downarrow e \setminus \hat{F}_{e,f} \equiv_i \downarrow f \setminus F_{e,f} \setminus H$.

First, let us assume that e and f are not equivalent. Then, it is sufficient to remember at most $|T|$ events in their causal past if e and f are causally related to be able to differentiate them, and to take as frontier the minimal event in this bounded set. Similarly, if e and f are conflicting events, it is sufficient to remember k_c events in their causal past to differentiate the observation of their past. Last, if e and f are concurrent events, considering $3.|T|$ events in their past suffices to notice that they are not equivalent.

Conversely, assume that a pair of events e, f is equivalent w.r.t observation $\mathcal{O}_i$. Then, these events are at a distance d smaller than $\max(2.k_c, 3.|T|)$. If e, f are causally related, and $e \leq f$ then it suffices to remember the maximal events of $\downarrow e$ as frontier (this frontier is finite). Then, checking that $\downarrow e \equiv_i \downarrow f$ amounts to checking that $(\downarrow f \cap Ball_d(f)) \setminus (\downarrow e \cap Ball_d(f))$ is an empty observation. Now, assume that e and f are conflicting events. As e and f are at distance at most $2.k_c$, they share a common past, whose events are located at distance at most k_c. It is then sufficient to take as frontier the maximal event in $\downarrow e \cap \downarrow f$, which are contained in the $2.k_c$-*Ball* of e and f and then check that $\downarrow e \setminus \hat{F}_{e,f} \equiv_i \downarrow f \setminus F_{e,f} \setminus H$. A similar reasoning holds for concurrent events. Hence, It suffices to set $K = \max(2.k_c, 3.|T|)$ and to consider the nature of pairs of events to find an appropriate frontier allowing to check equivalence of any pair of events located at distance smaller than K.

◀

Corollary 20: HyPOL is decidable for observable safe Petri nets.

Proof. The proof directly follows from corollary 18 and Proposition 5: as an observable net $\mathcal{N}$ is $K-$layered, for $K = \max(2.k_c, 3.|T|)$, it suffices to build the grammar $\mathcal{G}_{K,\mathcal{N}}$ that generates the execution graph $G_{\mathcal{U}}(\mathcal{N})$ for a $K = 3.|T|$. Then, for every HyPOL formula, one can compute an equivalent MSO formula, and verify that this property is satisfied on $\mathcal{G}_{K,\mathcal{N}}$. ◀