Non-interference in partial order models

BÉATRICE BÉRARD, Sorbonne Universités, UPMC Univ. Paris 06, CNRS UMR 7606, LIP6
LOÏC HÉLOUËT, INRIA Rennes
JOHN MULLINS, École Polytechnique de Montréal

Non-interference (NI) is a property of systems stating that confidential actions should not cause effects observable by unauthorized users. Several variants of NI have been studied for many types of models, but rarely for true concurrency or unbounded models. This work investigates NI for High-level Message Sequence Charts (HMSC), a scenario language for the description of distributed systems, based on composition of partial orders. We first propose a general definition of security properties in terms of equivalence among observations of behaviors. Observations are naturally captured by partial order automata, a formalism that generalizes HMSCs and permits to assemble partial orders. We show that equivalence or inclusion properties for HMSCs (hence for partial order automata) are undecidable, which means in particular that NI is undecidable for HMSCs. We hence consider decidable subclasses of partial order automata and HMSCs. Finally, we define weaker local properties, describing situations where a system is attacked by a single agent, and show that *local NI* is decidable. We then refine local NI to a finer notion of *causal NI* that emphasizes causal dependencies between confidential actions and observations, and extend it to causal NI with (selective) declassification of confidential events. Checking whether a system satisfies local and causal NI and their declassified variants are PSPACE-complete problems.

CCS Concepts: • **Software and its engineering** → **Software verification;**

Additional Key Words and Phrases: Security, non-interference, partial orders, verification

ACM Reference Format:
Béatrice Bérard, Loïc Hélouët, and John Mullins, 2016. Non-interference in partial order models. *ACM Trans. Embedd. Comput. Syst.* V, N, Article A (YYYY), 31 pages.
DOI: 0000001.0000001

1. INTRODUCTION

Context. *Non-interference* (NI) has been introduced to characterize the absence of harmful information flow in a system. It ensures that confidential actions of a system can not produce any effect visible by a public observer. The original notion of non-interference in [Goguen and Meseguer 1982] was expressed in terms of language equivalence for deterministic Mealy machines with confidential input and public output. Since then, several variants of *information flow properties* (IFP) have extended NI to non-deterministic models (transition systems, process algebra, Petri nets,...) and finer notions of observation (simple trace observation, deadlock or branching detection,....) to describe the various observational powers of an attacker. For a given system S, NI is usually defined as: $\pi_V(\llbracket S \setminus C \rrbracket) \approx \pi_V(\llbracket S \rrbracket)$ where $\approx$ denotes some behavioural system equivalence (language equivalence, bisimulation, ...), $\llbracket S \rrbracket$, the semantics of S, π_V, the projection on a subset V of visible actions of the system, and $S \setminus C$, the model S from which all confidential actions from C are pruned. *Intransitive non-interference* (INI) relaxes NI to handle possible *declassification* of confidential ac-

This work is an extended version of [Bérard et al. 2015]. It was partially done while John Mullins was visiting the LIP6, Université Pierre & Marie Curie. John Mullins is supported by the NSERC Discovery Individual grant No. 13321 (Government of Canada), the FQRNT Team grant No. 167440 (Quebec's Government) and the CFQCU France-Quebec Cooperative grant No. 167671 (Quebec's Government).

Author's addresses: B. Bérard, Sorbonne Universités, UPMC Univ. Paris 06, CNRS UMR 7606, LIP6, 75005, Paris, France; L. Hélouët, INRIA Rennes Campus de Beaulieu, 35041 Rennes Cedex, France; J. Mullins, Dept. of Computer & Software Eng., École Polytechnique de Montréal, P.O. Box 6079, Station Centre-ville, Montreal (QC) Canada, H3C 3A7.

DOI: 0000001.0000001

tions. It ensures that confidential actions of a system cannot produce any effect visible by a public observer unless they are declassified, causing so a harmless information flow. This issue has been addressed in [Rushby 1992], by comparing observations of visible actions in runs of a system (hence including runs containing non-declassified confidential actions), and observations of visible actions in runs of the same system that only contain confidential actions that are declassified afterwards. Most IFPs have been expressed as combinations of *basic security predicates* (BSPs) [Mantel 2000; 2001; D'Souza et al. 2011] or as a behavioral equivalence under observation contexts [Focardi and Gorrieri 2001]. A systematic presentation of IFPs can be found, *e.g.*, in [Mantel 2000; 2001; Focardi and Gorrieri 2001].

Concurrency issues. Despite the fact that IFPs are always informally expressed in term of causality *i.e.*, confidential activity should not cause observable effects on the public behavior, they are almost always formalized in terms of interleaving semantics [Busi and Gorrieri 2009; Gorrieri and Vernali 2011; Best et al. 2010; Best and Darondeau 2012] and hence, do not consider true concurrency or causality. This is clearly a lack in the formalization of IFPs for several reasons. First, from an algorithmic point of view, it is usually inefficient to compute a set of linearizations to address a problem that can be solved on an equivalent partial order representation. Second, from a practical point of view, an attacker of a system may gain more information if he knows that some confidential action has occurred recently in its causal past. Indeed, transactions in a distributed system can leave many traces (visited websites, cookies,...) on machines which are not *a priori* committed to protect confidential actions of third parties. At the best of our knowledge, [Baldan and Carraro 2014] is the first to address NI in a true concurrency setting: they characterized NI for Petri nets as a syntactic property of their unfoldings. However, the technique addresses only safe nets.

Unbounded models. Very few results address IFPs for unbounded models. BSPs and NI are proved undecidable for pushdown systems, but decidability was obtained for small subclasses of context-free languages [D'Souza et al. 2011]. Decidability of a bisimulation-based strengthened version of NI called *non-deducibility on composition* (NDC) for unbounded Petri nets is proved in [Best et al. 2010]. A system satisfies NDC if observation of its visible actions remains indistinguishable from the observation of the system interacting with *any* environment. This result was extended in [Best and Darondeau 2012] to INI with selective declassification (INISD).

Contribution. This work considers IFPs for an unbounded true concurrency model, namely *High-level Message Sequence Charts* (HMSCs). This model, standardized by the ITU [ITU-T 2011], is well accepted to represent executions of distributed systems, where security problems are of primary concern. We first define a class of IFPs on HMSCs, as an inclusion relation on observations, following [Focardi and Gorrieri 2001; D'Souza et al. 2011] and [Bérard and Mullins 2014]. To keep IFPs within a true concurrency setting, observations of HMSCs are defined as partial orders. We define a new model called *partial order automata* (POA), that is powerful enough to recognize infinite sets of partial orders, and in particular observations of HMSCs. Unsurprisingly, most of IFPs and the simple NI property are undecidable for HMSCs. As a consequence, inclusion of partial order automata languages is undecidable. We then characterize decidable subclasses of the problem: inclusion of sets of orders generated by POA becomes decidable when the depicted behaviors do not allow observed processes to race each other. This is for instance the case when a POA describes an observation of visible events located on a single process. This also applies when the observed HMSC is *locally synchronized* meaning that within any iterated behavior, all processes synchronize at each iteration. We discuss the meaning of NI in a context where causal dependencies among event occurrences are considered. This leads to a new notion called *causal interference* for HMSCs. Causal interference detects interference as soon as an attacker can observe occurrences of confidential actions from visible events, and furthermore, one of the observed events causally depends on the confidential one. We finally relax causal interference in the context of declassification. We introduce *intransitive causal non-interference* that considers observable causal dependencies among confidential and visible events as safe, as soon as a declassification occurs in between. We show that all local variants of these problems are PSPACE-complete.

Outline. The basic models and definitions used in this paper are defined in Section 2. Observations, inclusion problems and non-interference are introduced in Section 3 for a single scenario and in Section 4 for HMSCs, where NI is proved undecidable. Section 4 introduces partial order automata as a way to recognizes observations of HMSCs. We identify subclasses of HMSCs and POAs where inclusion problems becomes decidable in Section 5. Then we consider local variants in Section 6 and extend this framework to declassification in Section 7 Section 8, we compare this work with some related approaches, and draw several research directions. Due to lack of space, several proofs are omitted or simply sketched, but can be found in an extended version at `hal.inria.fr/XXXXXX`.

2. PRELIMINARIES

In this section, we recall definitions of automata, partial orders and High-level Message Sequence Charts (HMSCs), with their associated languages. Message Sequence Charts (MSCs) are formal representations of distributed executions, *i.e.*, chronograms, that are frequently used to depict the behavior of a set of asynchronous communicating processes. This simple graphical representation emphasizes on messages and localization of actions, with partial order semantics (see illustration in Figure 1 Section 3).

The model of HMSCs, standardized by the ITU [ITU-T 2011], was proposed to describe more elaborate behaviors of distributed systems, for instance those of communication protocols, by combining MSCs. An example is given in Figure 3 of Section 6. HMSCs are used to describe sets of typical scenarios in distributed systems, and then serve as requirements. They can also be used as input to generate code skeletons for distributed systems. Hence, an information leak that appears in these early requirements is likely to be a feature of the final system. It is then interesting to find these leaks at early design stages. Another interesting point with HMSCs is their expressive power: they define behaviors of systems with asynchronous communications, which are not necessarily finite state systems and can not be captured by finite automata. They are also uncomparable with Petri nets. Answering interference questions for HMSCs provides security techniques for a whole class of infinite systems that can not be modeled with other formalisms.

2.1. Finite automata and partial orders

Let Σ be a finite alphabet. A word over Σ is a sequence $w = a_1 a_2 \dots a_n$ of letters from Σ, and Σ^* denotes the set of finite words over Σ, with ε the empty word. A *language* is a subset L of Σ^*. Given a relation $R \subseteq E \times E$ on some set E, we denote by R^* the transitive and reflexive closure of R. A partial order on E is a reflexive, transitive, and acyclic relation. Let f_1 and f_2 be two functions over disjoint domains $Dom(f_1)$ and $Dom(f_2)$. Then, $f_1 \cup f_2$ denotes the function defined on $Dom(f_1) \cup Dom(f_2)$, that associates $f_1(x)$ with every $x \in Dom(f_1)$ and $f_2(x)$ with every $x \in Dom(f_2)$.

A *Finite Automaton* over alphabet Σ is a tuple $\mathcal{A} = (S, \delta, s_0, F)$, where S is a finite set of states, $s_0 \in S$ is the initial state, $F \subseteq S$ is a set of accepting states, and $\delta \subseteq S \times \Sigma \times S$ is a transition relation. A word $w = a_1 \dots a_n \in \Sigma^*$, is accepted by $\mathcal{A}$ if there exists a sequence of transitions $(s_0, a_1, s_1)(s_1, a_2, s_2) \dots (s_{n-1}, a_n, s_n)$ such that $s_n \in F$. It is well known that finite automata accept *regular languages*.

A *Labeled Partial Order* (LPO) over alphabet Σ is a triple $(E, \leq, \alpha)$ where $(E, \leq)$ is a partially ordered set (poset) and $\alpha : E \to \Sigma$ is a labeling of E by letters of Σ. The set of all LPOs over alphabet Σ is denoted by *LPO*(Σ). For a subset of events $E' \subseteq E$, the set of *predecessors* of E' is $\downarrow(E') = \{f \in E \mid f \leq e \text{ for some } e \in E'\}$ and the set of *successors* of E' is $\uparrow(E') = \{f \in E \mid e \leq f \text{ for some } e \in E'\}$. The set E' is *downward closed* if $\downarrow(E') = E'$, and *upward closed* if $\uparrow(E') = E'$. A *linear extension* of LPO $(E, \leq, \alpha)$ with n events is a sequence $r = e_1 e_2 \dots e_n$ of all events of E such that for every $j > k$, $e_j \not\leq e_k$.

Let $O_1 = (E_1, \leq_1, \alpha_1)$ and $O_2 = (E_2, \leq_2, \alpha_2)$ be two LPOs over Σ. We write $O_1 \sqsubseteq O_2$ if O_1 is a *prefix* of O_2: there exists an injective mapping $h : E_1 \to E_2$ such that $\alpha_2(h(e)) = \alpha_1(e)$ for all $e \in E_1$, $h(E_1)$ is downward closed, and $e_1 \leq_1 f_1$ iff $h(e_1) \leq_2 h(f_1)$. Moreover, O_1 is isomorphic to O_2, denoted by $O_1 \equiv O_2$, if $O_1 \sqsubseteq O_2$ and $O_2 \sqsubseteq O_1$. A set of partial orders Y *contains* another set of partial orders X, denoted by $X \subseteq Y$, if for every $x \in X$, there exists $y \in Y$ such that $x \equiv y$. We will write $X \equiv Y$ if $X \subseteq Y$ and $Y \subseteq X$. We say that X embeds into Y, denoted $X \sqsubseteq Y$ iff for every $x \in X$, there exists $y \in Y$ such that $x \sqsubseteq y$. Given a LPO $O = (E, \leq, \alpha)$, the *covering* of O is a triple $(E, \prec, \alpha)$ where $\prec$ is the transitive and reflexive reduction of $\leq$, *i.e.*, the smallest subset of $E \times E$ such that $\prec^* = \leq$. Since two orders are isomorphic iff their coverings are isomorphic, we often consider covering relations instead of orders in the rest of the paper.

2.2. High Level Message Sequence Charts

Definition 2.1 (*MSC*). A *Message Sequence Chart* over finite sets $\mathbb{P}$ of processes, $\mathbb{M}$ of messages and finite alphabet A, is a tuple $M = (E, (\leq_p)_{p \in \mathbb{P}}, \alpha, \mu, \phi)$, where:

- E is a finite set of *events*, partitioned as $E = E_S \uplus E_R \uplus E_I$, according to the type of event considered: message sending, reception, or internal *atomic action*;
- $\phi : E \rightarrow \mathbb{P}$ is a mapping associating with each event the process that executes it. Hence, the sets $E_p = \phi^{-1}(\{p\})$ for $p \in \mathbb{P}$, also form a partition of E;
- For each $p \in \mathbb{P}$, the relation $\leq_p \subseteq E_p \times E_p$ is a total ordering on events located on process p;
- $\mu \subseteq E_S \times E_R$ is a relation symbolizing message exchanges, such that if $(e, f) \in \mu$ with $e \in E_p$ and $f \in E_q$, then $p \neq q$. Furthermore, it induces a bijection from E_S onto E_R, so with a slight abuse of notation, $(e, f) \in \mu$ is also written as $f = \mu(e)$. The relation $\leq_M = (\bigcup_{p \in \mathcal{P}} \leq_p \cup \mu)^*$ is a partial order on E;
- α is a mapping from E to $\Sigma \subseteq (\mathbb{P} \times \{!, ?\} \times \mathbb{P} \times \mathbb{M}) \cup (\mathbb{P} \times A)$ and from μ to $\mathbb{M}$, associating a label with each event, and a message $\alpha(e, f)$ in $\mathbb{M}$ with each pair $(e, f) \in \mu$. The labeling is consistent with μ: if $f = \mu(e)$, with associated message $\alpha(e, f) = m$, sent by process p to process q, then $\alpha(e)$ is written as $p!q(m)$ and $\alpha(f)$ as $q?p(m)$. If e is an internal action a located on process p, then $\alpha(e)$ is of the form $p(a)$. The labeling is extended by morphism over E^*.

The definition above implies that the triple $(E, \leq_M, \alpha)$ is a LPO over Σ, hence all notions related to posets apply to MSCs. When clear from the context, we simply write $\leq$ instead of $\leq_M$. We denote by $\mathcal{M}sc(\mathbb{P}, \mathbb{M}, A)$ the set of all MSCs over the sets $\mathbb{P}$ of processes, $\mathbb{M}$ of messages, and alphabet A. Given a subset E' of E, the *restriction* of M to E', denoted by $M_{|E'}$, is associated with the LPO $\big(E', \leq_M \cap (E' \times E'), \alpha_{|E'}\big)$ and we denote by $M \setminus E'$ the restriction of M to $E \setminus E'$.

Definition 2.2. A *linearization* of MSC M is a word $w \in \Sigma^*$ such that there exists a linear extension r of M with $w = \alpha(r)$. The *language* $\mathcal{L}(M)$ of M, is the set of linearizations of M.

The language of an MSC is hence defined over alphabet $\Sigma = \{p!q(m) \mid p, q \in \mathbb{P} \wedge m \in \mathbb{M}\} \cup \{p?q(m) \mid p, q \in \mathbb{P},\ m \in \mathbb{M}\} \cup \{p(a) \mid p \in \mathbb{P},\ a \in A\}$. To design more elaborate behaviors, including choices and iterations, MSCs are composed. A key ingredient is sequential composition, that assembles MSCs processwise to form larger MSCs.

Definition 2.3. Let $M_1 = (E_1, (\leq_{1,p})_{p \in \mathbb{P}}, \alpha_1, \mu_1, \phi_1)$ and $M_2 = (E_2, (\leq_{2,p})_{p \in \mathbb{P}}, \alpha_2, \mu_2, \phi_2)$ be two MSCs defined over disjoint sets of events. The *sequential composition* of M_1 and M_2, denoted by $M_1 \circ M_2$ is the MSC $M_1 \circ M_2 = (E_1 \cup E_2, (\leq_{1 \circ 2, p})_{p \in \mathbb{P}}, \alpha_1 \cup \alpha_2, \mu_1 \cup \mu_2, \phi_1 \cup \phi_2)$, where $\leq_{1 \circ 2, p} = \big(\leq_{1,p} \cup \leq_{2,p} \cup\, \phi_1^{-1}(\{p\}) \times \phi_2^{-1}(\{p\})\big)^*$.

This (associative) operation, also called concatenation, can be extended to n MSCs. For a set $\mathcal{M}$ of MSCs, we denote by $\mathcal{M}^{\circ *}$ the set of all MSCs obtained by concatenation of (a finite number of) MSCs in $\mathcal{M}$. It is used to give a semantics to higher level constructs, namely HMSCs. Roughly speaking, an HMSC is a finite automaton where transitions are labeled by MSCs. It produces a **set of MSCs** obtained by concatenating MSCs that appear along paths.

Definition 2.4 (*HMSC*). A *High-level MSC* (HMSC) is a tuple $H = (N, \rightarrow, \mathcal{M}, n_0, F)$, where N is a set of nodes, $\mathcal{M}$ is a finite set of MSCs, $\rightarrow \subseteq N \times \mathcal{M} \times N$ is a transition relation, $n_0 \in N$ is the initial node, and F is a set of accepting nodes.

As for any kind of automaton, paths and languages can be defined for HMSCs. A *path* of H is a sequence $\rho = t_1 t_2 \dots t_k$ such that for each $i \in \{1, \dots, k\}$, $t_i = (n_i, M_i, n'_i)$ is a transition in $\rightarrow$, with $n'_i = n_{i+1}$ for each $i \leq k-1$. The path ρ is a cycle if $n'_k = n_1$. It is *accepting* if it starts from node n_0 (*i.e.*, $n_1 = n_0$), and it terminates in a node of F ($n'_k \in F$).

Definition 2.5. Let $\rho = t_1 t_2 \dots t_k$ be a path of an HMSC H. The MSC associated with ρ is $M_\rho = h_1(M_1) \circ h_2(M_2) \cdots \circ h_k(M_k)$ where each h_i is an isomorphism that guarantees $\forall j \neq i, h_i(E_i) \cap h_j(E_j) = \emptyset$.

More intuitively, the MSC associated with a path is obtained by concatenating MSCs encountered along this path after renaming the events to obtain disjoint sets of events. To simplify notation, we often drop the isomorphisms used to rename events, writing simply $M_\rho = M_1 \circ M_2 \circ \cdots \circ M_k$.

With this automaton structure and the sequential composition of MSCs, an HMSC H defines a set of *accepting paths*, denoted by $\mathcal{P}_H$, a set of MSCs $\mathcal{F}_H = \{M_\rho \mid \rho \in \mathcal{P}_H\}$, and a linearization language $\mathcal{L}(H) = \bigcup_{M \in \mathcal{F}_H} \mathcal{L}(M)$.

It is well known that the linearization language of an HMSC is not necessarily regular, but rather a closure of a regular language under partial commutation, which yields many undecidability results (see for instance [Muscholl and Peled 1999; Caillaud et al. 2000]). This does not immediately mean that all IFPs are undecidable for HMSCs: Indeed, several classes of HMSCs with decidable properties have been identified and we later define non-trivial and meaningful subclasses of HMSCs and observations for which IFPs become decidable. In particular, the *locally synchronized* HMSCs defined below have regular linearization languages [Alur and Yannakakis 1999]:

Definition 2.6. The *communication graph* of an MSC $M = (E, (\leq_p)_{p \in \mathbb{P}}, \alpha, \mu, \phi)$ is the graph $(\mathbb{P}, \rightarrow)$ where $(p, q) \in \rightarrow$ if there exists a pair of events $(e, f) \in \mu$ such that $\phi(e) = p$ and $\phi(f) = q$. An HMSC H is said *locally synchronized* iff for every cycle ρ of H, the communication graph of M_ρ is strongly connected.

3. OBSERVATION AND NON-INTERFERENCE FOR MSCs

The power of an external observer can be described by an observation function, mapping every behavior of a system to some observables. In [Mantel 2000; 2001; D'Souza et al. 2011], observation functions are seen as some particular kind of language theoretic operations (projection, morphism, insertion, deletion of letters,...), and in [Bérard and Mullins 2014], they are defined as combinations of rational operations (transductions, intersections, unions of languages).

In a distributed context, visible events can originate from different processes. In a distributed and asynchronous setting, the date at which an event is observed provides a linear ordering on observed events. However, the linear ordering provided by events observation date does not necessarily correspond to an actual execution: two concurrent processes may execute events concurrently, or conversely, there might be some causal depndecies among among observed events. This information on actions dependencies might be available to observers: If the system is equipped with vector clocks (vectors maintained by each process to count the known number of events that other processes have executed, as proposed in [Mattern 1988]), one can also record causal dependencies. Hence, the natural and realistic notion of observation for distributed computations is a labeled partial order, where events that are not causally dependant are considered concurrent.

3.1. Observations for MSCs

Definition 3.1. An observation function is a mapping from $\mathcal{M}sc(\mathbb{P}, \mathbb{M}, A)$ to $LPO(B)$ for some alphabet B.

From this definition, any mapping from MSCs to LPOs can be called an observation. However, some observation functions are natural when considering IFPs. As proposed in [Mantel 2001] with the notion of *views*, the alphabet labeling events that occur during an execution of a system can be partitioned as $\Sigma = V \uplus C \uplus N$ with visible, confidential and internal (neutral) labels. Actions with labels in V can be observed while actions labeled in C are confidential and should be hidden. Internal actions have labels in N and are not observable *a priori*, but need not be kept secret. Subsequently, depending on their labels, events are also called visible, confidential, or internal events.

Various observation functions can be defined from such a partition. The most natural ones are restrictions to visible events, and pruning of confidential actions, which are standard operations in language based non-interference literature, but need to be precisely defined in a partial order setting. Let $M = (E, (\leq_p)_{p\in\mathbb{P}}, \alpha, \mu, \phi)$ be an MSC with labeling alphabet Σ. We consider the following observation functions:

- **identity:** the identity $id(M) = M$ outputs the same LPO as the executed MSC;
- **Restriction:** $\mathcal{O}^V(M)$ is the LPO obtained by restriction of M to $E \cap \alpha^{-1}(V)$. Intuitively, $\mathcal{O}^V(M)$ represents the visible events and their causal dependencies that one may observe during the complete execution of M; Note that restriction to $\alpha^{-1}(V)$ suffices, as $\leq$ is transitive.
- **Pruning:** $\mathcal{O}^V_{\setminus C}(M) = \mathcal{O}^V(M \setminus \uparrow(\alpha^{-1}(C)))$ is a function that prunes out the future of confidential events from M, leaving only the visible events and their causal dependencies, observed when no confidential event, nor their future, are executed within M;
- **Localization:** $\mathcal{O}^p(M) = \mathcal{O}^V(M_{|E_p})$, for a given process $p \in \mathbb{P}$, is the observation of visible events of M restricted to those events located on process p. Note that $\mathcal{O}^p(M)$ is a total order. In a distributed setting, $\mathcal{O}^p(M)$ is particularly interesting, as it represents the point of view of a single process $p \in \mathbb{P}$, considered as the attacker of the system. We hence assume no restriction on the set of events that can be executed and observed by p, and let $V = \Sigma_p = \alpha(E_p)$ when using $\mathcal{O}^p$.

3.2. Non-interference for MSCs

As noticed by [D'Souza et al. 2011] in a language setting, information flow properties of a system S are usually defined as compositions of atomic propositions of the form $op_1(S) \subseteq op_2(S)$. Changing the observation functions op_1, op_2 (or the partition of Σ) leads to a variety of such atomic properties. Information flow properties of MSCs can be defined similarly.

Definition 3.2. Let $\mathcal{O}_1, \mathcal{O}_2$ be two observation functions over $\mathcal{M}sc(\mathbb{P}, \mathbb{M}, A)$. An MSC M *satisfies* the *inclusion property* for $\mathcal{O}_1, \mathcal{O}_2$, written $\sqsubseteq_{\mathcal{O}_1,\mathcal{O}_2}(M)$, if $\mathcal{O}_1(M) \sqsubseteq \mathcal{O}_2(M)$.

Very often, interference is informally described as causal dependencies between confidential actions and observable ones, but formalized in terms of languages comparison, *i.e.*, with interleaved representations that miss information on concurrency and causality. For a single MSC M, the notion of non-interference can be defined as a comparison of partial orders:

Definition 3.3. An MSC M over labeling alphabet $\Sigma = C \uplus V \uplus N$ is *non-interferent* if $\mathcal{O}^V(M) \equiv \mathcal{O}^V_{\setminus C}(M)$. Otherwise M is said *interferent*.

We now show that interference in a single MSC can be defined in terms of causal dependencies from confidential events (in C) to visible ones (in V). We then show in Section 3.3 that checking existence of such dependencies can be performed via coloring of events. For a single MSC, comparing observations $\mathcal{O}^V$ and $\mathcal{O}^V_{\setminus C}$ defined above suffices to highlight dependencies between confidential and visible actions. Hence, interference in a *single* MSC can be defined through causality:

PROPOSITION 3.4. *Let M be an MSC over $\Sigma = C \uplus V \uplus N$ and set of events E. Then, M is interferent if and only if there are two events e, f such that $\alpha(e) \in C$, $\alpha(f) \in V$, and $e \leq f$.*

This result can be used to define interference in terms of a property of a coloring of MSCs, and to show that this coloring is compositional.

3.3. Interference detection by coloring

The relation between causal dependencies and interference calls for a graphical interpretation of interference in MSCs, represented as a propagation of a black token inherited from confidential actions along causal dependencies. Intuitively, any confidential action and successors of actions marked with a black token are also marked with a black token and every process containing a black action is also marked as black. Though the black/white coloring of MSCs is not essential to prove interference, it will be used later to detect information flows in HMSCs.

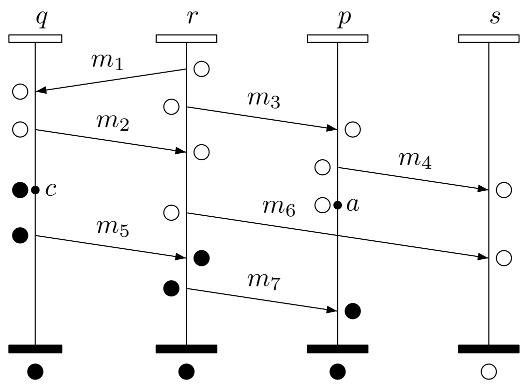

Fig. 1. An MSC M_{bw} tagged with black and white tokens

Definition 3.5 (*MSC and process coloring*). Let M be an MSC over alphabet $\Sigma = C \uplus V \uplus N$ and set of events E. An event $e \in E$ is *black* if $\alpha(\uparrow (e)) \cap C \neq \emptyset$, and *white* otherwise. A process $p \in \mathbb{P}$ is *black after* M (resp. *white after* M) if there exists a black event located on p (resp. no black event on p).

Intuitively, a black process can detect occurrences of confidential events, as it executes events that are causal consequences of confidential events. Clearly, an MSC is non-interferent if and only if it does not contain visible black events. Figure 1 shows a coloring of an MSC M_{bw} in black and white. The alphabet of confidential actions is $C = \{q(c)\}$ and contains the label of the atomic action c executed by process q. We attach a black token to every black event and a white token to other events. Similarly, we indicate with a black/white token below process lines whether a process has met a black token during its execution. In this example, process p can detect occurrences of c (it is black after M_{bw}), but process s cannot.

Deciding if an MSC is interferent, or equivalently if it contains a visible black event then consists in finding a path from a confidential event to a visible one in an acyclic graph where events are seen as vertices and pairs of events (e, f) in $(\cup_{p\in\mathcal{P}} \leq_p) \cup \mu$ as edges. Since an event has at most two immediate successors, the graph to consider has at most $n = |E_M|$ vertices and $2n$ edges. Hence, coloring of MSCs and interference detection can be performed in linear time as a graph exploration starting from confidential events. We now show that deciding the black/white status of a process along a sequence of MSCs of arbitrary size can be performed with bounded memory.

PROPOSITION 3.6. *Let M_1, M_2 be two MSCs with labels in $\Sigma = C \uplus V \uplus N$. Then, process $p \in \mathbb{P}$ is black after $M_1 \circ M_2$ iff it is black after M_1, or it is black after M_2, or there exists a process q black after M_1 and a pair of events $e \leq f$ in M_2 such that e is located on q and f is located on p.*

This important property means that it is sufficient to remember the black/white status of each process after concatenation $M_1 \circ \cdots \circ M_k$ along a path of an HMSC to compute the status of process p after concatenation $M_1 \circ \cdots \circ M_k \circ M_{k+1}$.

4. OBSERVATIONS ON HMSCs AS PARTIAL ORDER AUTOMATA

In this section, we first discuss extending observation functions from MSCs to HMSCs and show that the inclusion problem as well as non-interference are undecidable for HMSCs. We also remark that some definitions of observation functions on HMSCs involve assembling partial orders obtained from the MSCs encountered along paths. This suggests the definition of Partial Order Automata

(POA) that are finite automata where transitions are labeled by LPOs. To increase the expressive power of this model, we introduce various ways of assembling the partial orders appearing along paths through composition operators and selection functions. The main purpose of this section is to present the material needed in Section 5 where we prove that non-interference is decidable for the subclass of locally synchronized HMSCs. This result is obtained by (1) building two partial order automata $\mathcal{A}_{H,\mathcal{O}_1}$ and $\mathcal{A}_{H,\mathcal{O}_2}$ associated with a locally synchronized HMSC H, respectively accepting observations $\mathcal{O}_1(H)$ and $\mathcal{O}_2(H)$ and (2) proving that in this case, the inclusion problem $\mathcal{O}_1(H) \subseteq \mathcal{O}_2(H)$ is decidable. In this section, we mainly identify sufficient conditions on the observation functions to achieve point (1) above, while decidability is proved in the next section.

4.1. Extending observations to HMSCs

In order to extend an observation $\mathcal{O}$ to an HMSC H, a first way consists in applying $\mathcal{O}$ to all MSCs in $\mathcal{F}_H$, defining $\mathcal{O}(H) = \{\mathcal{O}(M) \mid M \in \mathcal{F}_H\}$. In particular : $\mathcal{O}^{V,\circ}(H) = \{\mathcal{O}^V(M) \mid M \in \mathcal{F}_H\}$, $\mathcal{O}^{V,\circ}_{\backslash C}(H) = \{\mathcal{O}^V_{\backslash C}(M) \mid M \in \mathcal{F}_H\}$, and $\mathcal{O}^{p,\circ}(H) = \{\mathcal{O}^p(M) \mid M \in \mathcal{F}_H\}$. Extending Definitions 3.2 and 3.3 to HMSCs, we have:

Definition 4.1. An HMSC H satisfies the *inclusion* problem for $\mathcal{O}_1, \mathcal{O}_2$ (written $\sqsubseteq_{\mathcal{O}1,\mathcal{O}_2}(H)$) if $\mathcal{O}_1(H) \subseteq \mathcal{O}_2(H)$. It is *non-interferent* if $\mathcal{O}^{V,\circ}_{\backslash C}(H) \equiv \mathcal{O}^{V,\circ}(H)$.

Unfortunately, the observation functions above do not take into account the structure of the HMSC generating $\mathcal{F}_H$, and furthermore, they are not necessarily compositional. In general, an observation function $\mathcal{O}$ is not a morphism with respect to the concatenation, that is, $\mathcal{O}(M_1 \circ M_2) \neq \mathcal{O}(M_1) \circ \mathcal{O}(M_2)$. This drawback was already observed in [Genest et al. 2003] for projections of MSCs: in general, $\mathcal{O}^V(M_1 \circ M_2) \neq \mathcal{O}^V(M_1) \circ \mathcal{O}^V(M_2)$. Hence, checking inclusion for HMSCs may require to consider properties of complete sequences of MSCs as a whole, raising algorithmic difficulties, or even undecidability. Other ways to extend observations to HMSCs, are to assemble observations of MSCs piecewise, following the automaton structure of HMSCs, or to forbid MSCs containing confidential events:

$\mathcal{O}^{V,\bullet}(H) = \{\mathcal{O}^V(M_1) \circ \cdots \circ \mathcal{O}^V(M_k) \mid M_1 \circ \cdots \circ M_k \in \mathcal{F}_H\}$,
$\mathcal{O}^{V,\bullet}_{\backslash C}(H) = \{\mathcal{O}^V(M_1 \circ \cdots \circ M_k) \mid M_1 \circ \cdots \circ M_k \in \mathcal{F}_H \wedge \forall i, \alpha(E_i) \cap C = \emptyset\}$,
$\mathcal{O}^{p,\bullet}(H) = \{\mathcal{O}^p(M_1) \circ \cdots \circ \mathcal{O}^p(M_k) \mid M_1 \circ \cdots \circ M_k \in \mathcal{F}_H\}$,

where concatenation of LPOs is performed processwise like for MSCs. The observation $\mathcal{O}^{V,\bullet}_{\backslash C}(H)$ is of particular interest, as it describes observations of MSCs in $\mathcal{F}_H$ that do not contain MSCs with confidential events. Also note that, since $\mathcal{O}^p(M)$ is a total order, $\mathcal{O}^p$ satisfies the morphism property, which implies $\mathcal{O}^{p,\circ}(H) = \mathcal{O}^{p,\bullet}(H)$.

Even when a projection of an HMSC is an HMSC language (*i.e.*, a language recognizable by an HMSC), equivalence, inclusion or emptiness of intersection are undecidable. HMSC languages are not always regular and the observation of an HMSC needs not be regular either. In fact, due to the close relationship between HMSCs and Mazurkiewicz traces, most properties requiring to compare languages or partial order families are undecidable for HMSCs ([Caillaud et al. 2000; Muscholl and Peled 1999; 2000]). So, given two HMSCs H_1 and H_2, one can not decide if $\mathcal{L}(H_1) \subseteq \mathcal{L}(H_2)$, nor if $\mathcal{F}_{H_1} \subseteq \mathcal{F}_{H_2}$. This yields the following result:

THEOREM 4.2. *The inclusion problem $\sqsubseteq_{\mathcal{O}_1,\mathcal{O}_2}(H)$ is undecidable for H an HMSC and $\mathcal{O}_1$ and $\mathcal{O}_2$ two observation functions.*

PROOF. The proof is a reduction from the inclusion problem for partial order families generated by HMSCs. For two HMSCs H_1 and H_2, the question of whether $\mathcal{F}_{H_1} \subseteq \mathcal{F}_{H_2}$ is undecidable.

Let $H_1 = (N_1, \rightarrow_1, \mathcal{M}_1, n_{0,1}, F_1)$ and $H_2 = (N_2, \rightarrow_2, \mathcal{M}_2, n_{0,2}, F_2)$ be two HMSCs, defined over an alphabet of visible actions V, and with a set $\mathbb{P}$ containing at least two processes. We build an HMSC H, that behaves like H_1 or H_2 if a confidential action can occur, and like H_2 otherwise, and choose observation functions $\mathcal{O}_1 = \mathcal{O}^V, \mathcal{O}_2 = \mathcal{O}^V_{\setminus C}$. Then inclusion $\sqsubseteq_{\mathcal{O}_1,\mathcal{O}_2}(H)$ holds iff $\mathcal{F}_{H_1} \subseteq \mathcal{F}_{H_2}$.

Let c be a new confidential action and $P_c \notin \mathbb{P}$ a new process. We define M_c as the MSC containing the single atomic action c on process P_c, as illustrated on Figure 2 (middle). The new HMSC $H = (N_1 \uplus N_2, \rightarrow, \mathcal{M}, n_{0,2}, F_1 \uplus F_2)$ is defined over alphabet $\Sigma' = V \cup C$, where $C = \{P_c(c)\}$, as follows: $\mathcal{M} = \mathcal{M}_1 \uplus \mathcal{M}_2 \uplus \{M_c\}$ and $\rightarrow = \rightarrow_1 \uplus \rightarrow_2 \uplus \{(n_{0,2}, M_c, n_{0,1})\}$, as illustrated on the left part of Figure 2. Choosing $\mathcal{O}_1 = \mathcal{O}^{V,\circ}$ and $\mathcal{O}_2 = \mathcal{O}^{V,\circ}_{\setminus C}$, we have $\mathcal{O}_2(H) = \mathcal{F}_{H_2}$ and $\mathcal{O}_1(H) = \mathcal{F}_{H_1} \cup \mathcal{F}_{H_2}$. Thus $\sqsubseteq_{\mathcal{O}_1,\mathcal{O}_2}(H)$ if and only if $\mathcal{F}_{H_1} \subseteq \mathcal{F}_{H_2}$, which concludes the proof. $\square$

Note that undecidability is not due to a particular choice of observation function: a similar proof is obtained for $\mathcal{O}_1 = \mathcal{O}^{V,\circ}$ or $\mathcal{O}_1 = \mathcal{O}^{V,\bullet}$ and $\mathcal{O}_2 = \mathcal{O}^{V,\bullet}_{\setminus C}$, by replacing M_c by an MSC M'_c in which process P_c sends a message to all other processes after performing action c, as depicted on the right of Figure 2.

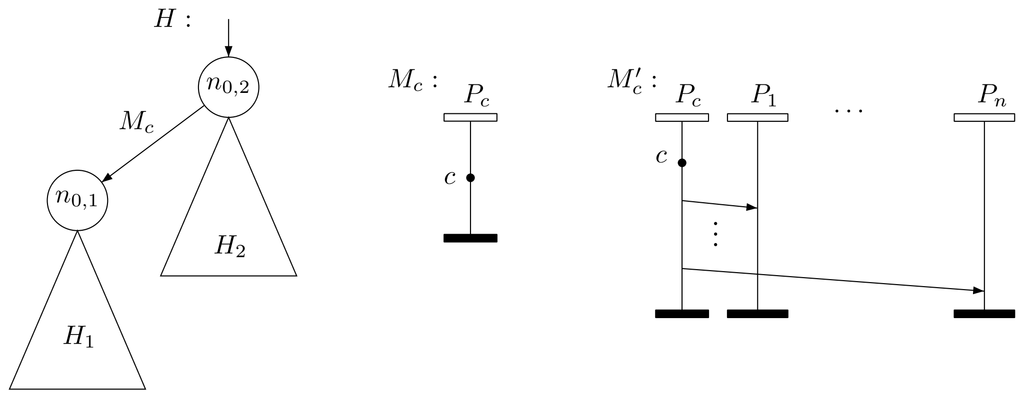

Fig. 2. Non-interference in HMSCs as an inclusion problem

This result extends to non-interference properties:

COROLLARY 4.3. *Non-interference for HMSCs is undecidable.*

PROOF. Consider the example HMSC in Figure 2, and the proof of theorem 4.2. The chosen partial order automata are HMSCs, and the observation functions are $\mathcal{O}^V$ and $\mathcal{O}^V_{\setminus C}$. If an algorithm answers the interference question, then it can be used to check isomorphism of $\mathcal{F}_{H_1}$ and $\mathcal{F}_{H_2}$ for any pair of HMSC H_1, H_2. Thus, the interference problem for HMSCs is undecidable. $\square$

4.2. Partial order automata

While HMSCs assemble finite MSCs to produce larger MSCs, i.e. particular LPOs, inclusion and interference properties do not compare MSCs but *observations of MSCs*. As mentioned above, projections of HMSCs are not in general HMSCs [Genest et al. 2003], hence observations of HMSCs are not HMSCs either. To compare the orderings (or their coverings) obtained by observation of a set of MSCs, we need more general structures. We propose in this section a model called *Partial Order Automata*, that assemble partial orders (or their coverings). Partial order automata are automata labeled by finite orders, and at each transition, the way to assemble the labeling order depends on a glueing operator attached to this transition. This model is more general than HMSCs, where the glueing operator is the same (sequential composition) for every transition.

Definition 4.4 (*Composition operator*). A *composition operator* for partial orders is an operator $\otimes : LPO(\Sigma) \times 2^{\mathbb{E}} \times LPO(\Sigma) \rightarrow LPO(\Sigma)$, where $\mathbb{E}$ is a set of events, that computes a partial order from a pair of partial orders O_1, O_2 and a subset of identified events Mem_1 from O_1. The result is denoted by $(O_1, \mathit{Mem}_1) \otimes O_2$. In practice, the operation is performed from coverings of O_1 and O_2 and produces a covering of the result.

A *selection function* is a function Π associating with a partial order $O = (E, \leq, \alpha)$ a subset of events $E' \subseteq E$. A *selection function* Π is *monotonic* if, for every pair of orders $O_1 \sqsubseteq O_2$, with $O_1 = (E_1, \leq_1, \alpha_1)$ and $O_2 = (h(E_1) \uplus E_2, \leq_2, \alpha_2)$ for the injective mapping h between the sets of events, then $\Pi(O_2) \subseteq h(\Pi(O_1)) \uplus E_2$. Function Π is a finite memory function if there exists $K \in \mathbb{N}$ such that $|\Pi(O)| \leq K$ for every LPO.

Selection functions are used to memorize events of interest during the construction of a covering relation by a partial order automaton. Intuitively, given two coverings $O_1 = (E_1, \prec_1, \alpha_1)$ and $O_2 = (E_2, \prec_2, \alpha_2)$, and a memorized subset of events Mem_1 in E_1 then $O = (O_1, \mathit{Mem}_1) \otimes O_2$ is a covering relation $O = (E, \prec, \alpha)$ where $E = E_1 \uplus E_2$, $\alpha = \alpha_1 \cup \alpha_2$, and $\prec$ is a covering relation that contains $\prec_1 \cup \prec_2$ and such that $\prec \setminus (\prec_1 \cup \prec_2) \subseteq \mathit{Mem}_1 \times E_2$ (the operator only glues events from the selection and events from the newly added order). Let us consider a monotonic selection function Π, and sequence of composition operations. Slightly abusing our notation, we write $O_1 \otimes_1 O_2$ instead of $(O_1, \Pi(O_1)) \otimes_1 O_2$, and similarly for sequences of compositions, we write $O = O_1 \otimes_1 O_2 \otimes_2 \cdots \otimes_{k-1} O_k$, and leave the selection process implicit. For monotonic selection functions, remembering previously memorized events suffice to compute a new memory. We can hence safely write $\Pi(\mathit{Mem} \otimes O)$ to define the set of events memorized after concatenation of O to an order with memory *Mem*.

In the rest of the paper, we consider composition operators that assemble multiple copies from a finite set of orders, *i.e.*, compositions of the form $O = O_1 \otimes_1 O_2 \otimes_2 \cdots \otimes_{k-1} O_k$ where each O_i is a copy from a finite set of LPOs $\mathcal{L}$, and $\otimes_1, \cdots \otimes_{k-1}$ are composition operators. To distinguish multiple copies of an order and of its events, we denote by $L^{(j)}$ the j^{th} occurrence of order $L = (E_L, \leq_L, \alpha_L) \in \mathcal{L}$, and by $e^{(j)}$ the j^{th} occurrence of some event $e \in E_L$.

Example 4.5. An example of selection function is the function, denoted by *MaxEvt*, that selects the last occurrence of each event of each order in $\mathcal{L}$. For a sequence $O = O_1 \otimes_1 O_2 \otimes_2 \cdots \otimes_{k-1} O_k$ and $L \in \mathcal{L}$, then $\mathit{MaxEvt}(O) = \cup_{L \in \mathcal{L}} \{e^{(j)} \mid e \in E_L \wedge |O|_L = j\}$, where $|O|_L$ is the number of occurrences of L in O. One can notice that *MaxEvt* is monotonic, and returns a finite set of events regardless of the size of the considered sequence of compositions.

Definition 4.6. A Partial Order Automaton (POA) is a tuple $\mathcal{A} = (Q, \longrightarrow, \mathcal{L}, q_0, F, \mathit{OPS}, \Lambda, \Pi)$ where Q is a finite set of states, $q_0 \in Q$ is the initial state, $F \subseteq Q$ is a set of final states, $\mathcal{L}$ is a finite set of LPOs, $\longrightarrow \subseteq Q \times \mathcal{L} \times Q$ is a set of transitions, Λ is a mapping associating with each transition an operator from some set *OPS*, and Π is a monotonic selection function. The transition relation is deterministic: for each $L \in \mathcal{L}$, and each $q \in Q$, there is at most one $q' \in Q$ such that $(q, L, q') \in \longrightarrow$.

For every path $\rho = q_0 \xrightarrow{O_1} q_1 \xrightarrow{O_2} q_2 \ldots q_{k-1} \xrightarrow{O_k} q_k$ of $\mathcal{A}$, one can compute an LPO O^ρ assembled as $O_\rho = O_1\, \Lambda(q_1, O_2, q_2)\, O_2 \ldots \Lambda(q_{k-1}, O_k, q_k)\, O_k$. For readability, we often omit the specific operators used to assemble orders, and simply write $O_\rho = O_1 \otimes O_2 \cdots \otimes O_k$. For two events e and f, we write $e \leq_\rho f$ when e precedes f in the partial order O_ρ. The partial order language of a POA $\mathcal{A}$ is the set of orders obtained by assembling orders along accepting paths of $\mathcal{A}$, and is denoted by $\mathcal{F}_\mathcal{A}$. The linearization language of $\mathcal{A}$ is the set of linearizations of orders in $\mathcal{F}_\mathcal{A}$.

First note that deterministic finite automata are particular cases of POA where each order labeling a transition is reduced to a single event, and the only operator involved is the standard concatenation on words. As mentioned above, HMSCs can also be seen as particular cases of partial order automata: one simply needs to relabel transitions with the partial order associated with the corresponding MSCs, use as selection function a function that memorizes the last event on each process, and as unique operator the operator that connects for each process this last event to the next occurrence of the minimal event on the same process. In the other way around, observation functions can be applied to POAs in a similar way as done for HMSCs: an observation function $\mathcal{O}$ can be applied on a partial order and for a POA $\mathcal{A}$, $\mathcal{O}(\mathcal{A}) = \{\mathcal{O}(L) \mid L \in \mathcal{F}_{\mathcal{A}}\}$. According to these remarks, the undecidability results for HMSCs immediately extend to POA:

PROPOSITION 4.7. *Let $\mathcal{A}$, $\mathcal{A}_1$, $\mathcal{A}_2$ be POA, and let $\mathcal{O}_1, \mathcal{O}_2$ be two observation functions. Then the inclusion problems $\mathcal{F}_{\mathcal{A}_1} \subseteq \mathcal{F}_{\mathcal{A}_2}$ and $\mathcal{O}_1(\mathcal{A}) \subseteq \mathcal{O}_2(\mathcal{A})$ are undecidable.*

4.3. Threaded and locally synchronized partial order automata

Most of formal properties of HMSCs are undecidable, and NI is not an exception. However, decidable subclasses of HMSCs have been identified. Locally synchronized HMSCs have regular linearization languages [Alur and Yannakakis 1999]. Hence inclusion of a regular language, or comparison of HMSC linearizations are decidable problems for locally synchronized HMSCs. It is then reasonable to consider a similar approach for partial order automata and identify subclasses on which comparison of covering relations is decidable. One of the factors that yields decidability in HMSCs is very often the fact that orderings are organized as processes. We can not have similar notions of processes in partial order automata, that only assemble occurrences of labeled events. However, we can use the fact that orders in $\mathcal{F}_{\mathcal{A}}$ are generated as compositions from a finite set of patterns to characterize subclasses of partial order automata.

Definition 4.8 (*Threaded POA*). A partial order automaton is *threaded* if for every path ρ of $\mathcal{A}$ containing at least two occurrences of some order $O = (E, \leq, \alpha)$ and every event $e \in E$, we have $e^{(i)} \prec_{\rho} e^{(i+1)}$ for any two consecutive occurrences of O.

One can notice that the composition mechanism of HMSCs immediately grants threaded partial order automata, as MSCs are composed processwise, and hence two successive occurrences of the same event in two occurrences of an MSC are necessarily ordered.

THEOREM 4.9. *Given a partial order automaton $\mathcal{A}$ with selection function MaxEvt, one can decide if $\mathcal{A}$ is threaded. Furthermore this problem is in co-NP.*

PROOF. The proof of this theorem is obtained by showing that the property can be checked on finite sequences. All path containing consecutive occurrences can be obtained by insertion of elementary cycles in acyclic path. We call elementary sequences the elementary cycles of $\mathcal{A}$ augmented with a transition that adds an occurrence of an order at the end of the cycle to allow considering consecutive occurrences of events. Then if two occurrences of an event in an elementary sequence are ordered, then a causal chain also exists between these two occurrences of an event in the order generated for a path obtained by insertion of a cycle in this elementary sequence. Hence, it suffices to consider elementary sequences to chech whether a POA is threaded. □

Being threaded is not a sufficient condition for a POA to define a regular language. We inspire from the class of locally synchronized HMSCs to define an appropriate syntactic class of POAs that have regular languages. Locally synchronized HMSCs rely on properties of communication among processes in cycles. POAs do not possess this notion of process, but in threaded POAs, ordering among events of the same kind replace this total ordering among events. We hence rely on properties of a *commutation graph* (instead of communication graphs in HMSCs) to define *locally synchronized* POAs.

Definition 4.10. Let $O = (E, \leq, \alpha)$ be a partial order, and ρ be a path such that $O = O_{\rho}$. The commutation graph of ρ is a graph $CG(\rho) = (E, \rightsquigarrow)$ where $(e, f) \in \rightsquigarrow$ iff

- $e \leq_{\rho} f$ (e precedes f in O_{ρ}),
- $e^{(1)} \leq_{\rho.\rho} f^{(2)}$ (the first occurrence of e precedes the next occurrence of f).

A POA is *locally synchronized* iff it is threaded, and for every of its cycles ρ, $CG(\rho)$ is strongly connected.

THEOREM 4.11. *Let $\mathcal{A}$ be a threaded partial order automaton with a selection function Π that memorizes a bounded number K of events. Then one can effectively decide if $\mathcal{A}$ is locally synchronized.*

PROOF. (Sketch) We can prove that existence of disconnected communication graphs can be proved on cycles than contain at most one occurrence of each transition. Indeed, for a selected pair of events e, f in such cycles, as considered automata are threaded, then insertion of another elementary cycles simply extends the lenght of causal chains and does not change connectedness of $e^{(1)}, f^{(1)}, e^{(2)}, f^{(2)}$. It then suffices to detect these cycles, build their commutation graph and check that these graphs are connected. □

4.4. Finitely decomposable observation functions

Definition 4.12. An observation function $\mathcal{O}$ is *decomposable* w.r.t. a set of MSCs $\mathcal{M}$ iff there exists a finite set of operators $OPS = \{\otimes_1, \ldots, \otimes_k\}$ and a function $\Psi : \mathcal{M}^{\circ *} \rightarrow OPS$ such that for every pair of MSCs M_1, M_2 in $\mathcal{M}^{\circ *}$, $\mathcal{O}(M_1 \circ M_2) = \mathcal{O}(M_1) \otimes_{\Psi(M_1)} \mathcal{O}(M_2)$.

Decomposability of an observation function w.r.t. to a set of MSCs guarantees that the set of operations needed to assemble the observations of two MSCs and obtain the observation of the concatenation of these MSC is finite, and that the operation to apply only depends on the order observed so far. This is a first step towards some form of compositionality for observations. This is however not sufficient to build incrementally an observation of a HMSC, as one may still need unbounded memory to assemble two observations.

Definition 4.13. An observation function $\mathcal{O}$ is *finitely decomposable* iff it is decomposable, and

(1) there exists a bound $c \in \mathbb{N}$ such that for every sequence of MSCs $M_1, \ldots, M_k$, and $\forall j \in 1..k$, denoting by $|<_{i..j}|$ the size of the covering of $\mathcal{O}(M_i \circ M_j)$, we have $|<_{1..k} \setminus (<_{1..j} \cup <_{j..k}| < c$
(2) there exists a bound m and a selection function Π such that for every sequence of MSCs, $M_1, \ldots, M_k$, for every $1 < i < j < k$,
   (a) Ψ is *regular*, *i.e.*, there exists a deterministic finite state machine $\mathcal{B}_\psi$ that reads sequences of MSCs and associates an operator with every state.
   (b) $\Pi(\mathcal{O}(M_1) \ldots \mathcal{O}(M_i))$ is a set Mem_i of size at most m containing events in $\mathcal{O}(M_1) \ldots \mathcal{O}(M_i)$
   (c) $<_{i,i+1} \subseteq Mem_i \times E_{i+1}$ (the memorized events are sufficient to build the ordering from events in $O_1 ... O_i$ to events in O_{i+1},
   (d) $\Pi(\mathcal{O}(M_1) \ldots \mathcal{O}(M_j))$ is a set Mem_j of size at most m containing m events in $\mathcal{O}(M_1) \ldots \mathcal{O}(M_j)$ such that $Mem_j \cap O_1 \ldots O_i \subseteq Mem_i$

Intuitively, for finitely decomposable observation functions with memorization function Π, the memorization function recalls only a bounded number of events that need to be used later along observation of a sequence, and the computation of the memory is compositional, in the sense that is removes useless events from memory at previous step, and adds new events that will be used later.

THEOREM 4.14. *For every HMSC $H = (N, \longrightarrow, \mathcal{M}, n_0, F)$, and every finitely decomposable observation function $\mathcal{O}$ (w.r.t. $\mathcal{M}$), one can build a POA $\mathcal{A}_{H,\mathcal{O}}$ that recognizes $\mathcal{O}(H)$.*

PROOF. For a given HMSC $H = (N, \longrightarrow, \mathcal{M}, n_0, F)$ we build the finite partial order automaton $\mathcal{A}_{H,\mathcal{O}} = (Q, \longrightarrow', \mathcal{O}(\mathcal{M}), q_0, F', \Lambda, \Pi)$. We define $Q = \{n_0\} \cup N \times OPS$, where OPS is the set of operators used by the finitely decomposable observation function $\mathcal{O}$. We set $\longrightarrow'$ as the set of triples of the form $((n, op), \mathcal{O}(M), (n', op'))$ such that $(n, M, n') \in \longrightarrow$ and there exists a path $\rho = n_0 \xrightarrow{M_1} n_1 \cdots \xrightarrow{M_k} n$ of H such that $\Psi(M_1 \ldots M_k) = op$ and $\Psi(M_1 \ldots M_k.M) = op'$. As Ψ is regular, $\longrightarrow'$ is finite and can be built inductively. Last, $\Lambda : Q \times \mathcal{O}(\mathcal{M}) \times Q \rightarrow OPS$ associates operator op to every transition $t = ((n, op), \mathcal{O}(M), (n', op')) \in \longrightarrow'$, and $\Lambda((n_0, O_i, n')) = id$, that is an observation starting from the initial node of the HMSC simply copies the observation of the first MSC recognized from the initial node of H. □

PROPOSITION 4.15. *For every HMSC H, observation functions $\mathcal{O}^V, \mathcal{O}^V_{\setminus C}, \mathcal{O}^p$ are finitely decomposable, with bounds $m \leq |\mathbb{P}|^2$ and $c \leq |\mathbb{P}|^3$.*

PROOF. (Sketch) We build this proof on the result of [Genest et al. 2003], that show that projections of HMSCs can be recognized by finite partial order automata that recognize projections of HMSCs. These automata memorize events that can still have a successor in the projected covering relation (whence a finite memory), and use a single composition operator that connects the projection of a newly observed MSC to memorized events (whence finite decomposability of the function that associates an operator to sequences of MSCs). As the set of events to memorize is always finite, as shown in [Genest et al. 2003], one can design a POA that recognizes $\mathcal{O}^V(H)$. The proof for $\mathcal{O}^V_{\setminus C}(H)$ and $\mathcal{O}^p(H)$ is similar. □

A consequence of this proposition is that one can build partial order automata that generate $\mathcal{O}^V(H), \mathcal{O}^V_{\setminus C}(H), \mathcal{O}^p(H)$. One can also notice that these automata are threaded, since with the composition operators used, the n^{th} event on a process necessarily precedes the $n+1^{th}$ event. Now, this does not mean that inclusion properties are decidable. We have to consider subclasses of partial order automata, and then check that observations fall into these subclasses.

5. INTERFERENCE DETECTION ALGORITHMS

To check an inclusion problem for an HMSC H, and subsequently check non-interference, one needs to compare runs in $\mathcal{A}_{\mathcal{O}_1,H}$ and $\mathcal{A}_{\mathcal{O}_2,H}$ for two suitable observation functions $\mathcal{O}_1, \mathcal{O}_2$. A run ρ_2 of $\mathcal{A}_{\mathcal{O}_2,H}$ is *compatible* with a run ρ_1 of $\mathcal{A}_{\mathcal{O}_1,H}$ if O_{ρ_1} is a prefix of O_{ρ_2} (*i.e.*, one can find a matching function h sending events of O_{ρ_1} onto O_{ρ_2}, as in the definition of prefix in section 2). One shall notice that there can be several runs of $\mathcal{A}_{\mathcal{O}_2,H}$ (possibly an infinite number of them) that are compatible with some run ρ_1. However, as soon as partial order automata are threaded, we can give a finite representation for sets of runs that comply with a finite order.

5.1. Minimal explanations and unfoldings of POA

Definition 5.1 (*Minimal explanations*). Let O, O' be LPOs, let *Mem* be a subset of events of O', and let $\mathcal{A}$ be a threaded POA with finite memory function Π and let q be a state of $\mathcal{A}$. The set of *minimal explanations* of $\mathcal{A}$ compatible with O starting from O', *Mem*, q is the set of all shortest path $\rho = (q, O_1, q_1) \dots (q_{k-1}, O_k, q_k)$ of $\mathcal{A}$, starting from q such that $O \sqsubseteq O' \otimes O_1 \otimes \cdots \otimes O_k$.

Considering shortest path is one essential requirement to have a finite representation for the set of paths of $\mathcal{A}$ that have a particular order O as prefix. We hence do not consider paths of the from $\rho.\rho'$ but only path ρ if O is already a prefix of O_ρ. This is however not sufficient to obtain a finite representation of paths containing O: a set of minimal explanations can still be infinite. Indeed, consider an order O with only two events $a \leq a'$. Then O could be a prefix of any order of the form $O_1(\otimes O_2)^k \otimes O_3$ where O_1 contains a, O_3 contains a', and O_2 only events that are not causally related to occurrences of a or a'. However, such iterations can be handled. We reuse ideas from [Hélouët et al. 2014] where a finite unfolding of an HMSC is built to perform diagnosis from a partial order observation, and an abstraction technique introduced in [Alur and Yannakakis 1999] to represent finitely sequences of MSCs that are partially executed. Let us first build a finite representation for this set of paths.

Starting from POA $\mathcal{A} = (Q, \longrightarrow, \mathcal{L}, q_0, F, \Lambda, \Pi)$ and LPO O, we build inductively a POA $\mathcal{B}$, which states and transitions are obtained by unfolding $\mathcal{A}$, and remembering after each transition the part of O that is a prefix of a path ending in this state, and the memorized events. States are hence of the form $(q, Mem_q, E_q,)$, where q is a state of $\mathcal{A}$, Mem_q is a description of memorized events (a subset containing events from Mem - the initial memory contents- and newly generated events), E_q is a subset of events of E_O. There is a transition from $(q, Mem_q, E_q,)$ to (q', Mem'_q, E'_q) labeled by O_i iff there exists a transition (q, O_i, q') in $\mathcal{A}$, and

- $E_q \neq E_O$ (O was not already regognized)

- $Mem'_q = \Pi(Mem_q \otimes O_i)$, where $\otimes = \Lambda(q, O_i, q')$,
- E'_q is the maximal subset of events of E_O that contains E_q and such that $O_{|(E'_q \setminus E_q) \cup Mem_q} \sqsubseteq Mem_q \otimes O_i$.

Intuitively, appending O_i to already built paths allows to embed a larger part of O in the recognized order. We define $\Lambda'((q, Mem, E), O_i, (q', Mem', E')) = \Lambda(q, O_i, q')$ and $\Pi' = \Pi$. During this construction, we hence consider loops that do not change the recognized part of O, nor the memory contents. States of the form (q, Mem_q, E_O) have no successor (O is a prefix of orders generated along all paths ending in this state) and are called final states. The construction can be performed inductively and stops when no new state is discovered. If the memory selection function of $\mathcal{A}$ memorizes only a finite number of events, and if $\mathcal{A}$ is threaded (which guarantees that the set of paths of $\mathcal{A}$ to explore to find the next occurrence of some action is bounded), then this construction terminates, and for every $O' \in \mathcal{F}_\mathcal{B}$, $O \sqsubseteq O'$.

We can then extract from $\mathcal{B}$ a finite set of sequential representations for the minimal explanations of O as follows: it is the set of acyclic path from q_0 to a final state, decorated with connected components for which transitions do not change the memory contents nor the part of O discovered so far. We call these transitions silent transitions: They are labeled by orders with events that might appear in larger orders containing O, but are not yet necessary to find a path ρ such that $O \sqsubseteq O_\rho$. Similarly, one can find minimal explanations from any state q starting with an already recognized order O' and memory contents *Mem*.

As $\mathcal{A}$ is threaded, for every transition $t = ((q, Mem, E), O_i, (q', Mem', E'))$, one can find which events of O_i are used to witness embedding of $O_{|(E'_q \setminus E_q) \cup Mem_q}$ into $Mem \otimes O_i$ (*i.e.*, are used to build a matching from O to $O' \otimes O_1 \dots O_i$). Once $\mathcal{B}$ is computed, we can compute a partial map $h_\mathcal{B}()$ that associates with an event along each transition (q, O_i, q') the corresponding event in E_O to which it corresponds. We say that an event e is marked if $h_\mathcal{B}(e)$ is defined, and denote by *Marked*$(t) \subseteq E_{O_i}$ the set of events marked in a transition $t = ((q, Mem, E), O_i, (q', Mem', E'))$. Note that it is not always the case that *Marked*$(t) = E_{O_i}$: when the transition reaches a final state of $\mathcal{B}$, a suffix of O_i may not be used to witness an embedding of O. Similarly, O_i can contain unmatched events located on parallel threads that will never be used to witness embedding of O along current path. We say that transition t is incompletely marked if *Marked*$(t) \neq E_{O_i}$.

5.2. Checking inclusion for POA

Now, let $\mathcal{A}_1$ and $\mathcal{A}_2$ be two partial order automata. Let $O = O_1 \otimes O_2 \otimes \dots O_n$ be an order generated along a path ρ_1 of $\mathcal{A}_1$, and let $\mathcal{B}$ be the minimal unfolding of $\mathcal{A}_2$ starting from $q_{2,0}$ (the initial state of $\mathcal{A}_2$) with empty order and memory. Let $\rho_{2,1}, \dots \rho_{2,k}$ be explanations provided by $\mathcal{B}$ ending on final states $q_1, \dots q_k$. These explanations $\rho_{2,1}, \dots \rho_{2,k}$ are paths decorated with silent connected components, hence they are partial order automata. Let $h_1, \dots, h_k$ be the mappings associated with transitions in $\rho_{2,1}, \dots \rho_{2,k}$, that links every event along path $\rho_{2,i}$ to the corresponding event of O, and let *Mrk*$_1, \dots,$ *Mrk*$_k$ denote the set of marked events along each path and let $\mathcal{B}_i$ denote the partial order automaton obtained by adding to $\rho_{2,i}$ all transitions of $\mathcal{A}$ that are accessible from q_i. Then, $O \otimes O_{n+1}$ is a prefix of some order generated by $\mathcal{A}_2$ iff it is a prefix of an order generated by one of the $\mathcal{B}_i$'s. Hence, $O \otimes O_{n+1}$ is a prefix of some order generated along a path ρ_2 of $\mathcal{A}_2$ if ρ_2 can be decomposed as $\rho_2 = \alpha_1 \beta_1 \alpha_2 \dots \alpha_k.\gamma$, where $O \sqsubseteq O_{\alpha_1 \dots \alpha_k}$, $\alpha_1 \dots \alpha_k$ is a path of $\mathcal{A}$ ending in some state q_α with memory Mem_α, β_i are finite sequences of transitions allowed between α_i and α_{i+1},and γ is a finite sequence of transitions obtained from an unfolding of $\mathcal{A}_2$ to check inclusion of O_{n+1} starting from state q with memory Mem_α, and from the restriction of order $O_{\alpha_1 \beta_1 \alpha_2 \dots \alpha_k}$ to unmarked events. However, the converse operation is more interesting. Starting from a finite path embedding an order O, and an explanation ρ (a sequence of transition with silent connected components attached to some states) as we add only a finite number of events and as $\mathcal{A}_2$ is threaded, one can hence compute all possible explanations for $O \otimes O_{n+1}$ by choosing adequate $\beta_i s$ in connected components of ρ and then computing possible explanations γ.

As proposed in [Alur and Yannakakis 1999] we can go further, and memorize only subsequences of each path that contain incomplete transitions, and the connection between these subsequences. Let $\mathcal{B}_i$ be the automaton associated with an explanation as above, and suppose that it starts with a single transition $t = (q, O_1, q')$ (*i.e.*, its initial state is not attached a silent strongly component) such that events are all marked. Then $O \otimes O_{n+1}$ is a prefix of some order generated by $\mathcal{A}_2$ iff $O \otimes O_{n+1} \setminus h_i(E_{O_1})$ is a prefix of $\mathcal{B}_i \setminus \{t\}$ with initial state q'. Hence, one can safely forget initial transitions which are all marked. Last, $O \otimes O_{n+1}$ is a prefix of some order generated by $\mathcal{A}_2$ iff O_{n+1} is a prefix of the projection of some O_ρ where ρ is a path of some $\mathcal{B}_i$ on its unmarked events. This means that one can simply memorize incompletely marked transitions, silent connected components, final states of all $\mathcal{B}_i s$ and still check that appending a particular order O_{n+1} preserves the inclusion proved so far. Starting from an explanation $\rho_{2,i}$ we denote by $Prune(\rho_{2,i})$ the sequence of incompletely marked transitions and connected components obtained from $\rho_{2,i}$. As extension of an explanation only uses connected components or appends orders at the end of the explanation, one can compute a new explanation from a pruned explanation. For an explanation ρ proving that an order O is a prefix of some order of $\mathcal{A}_2$, we denote by $Succ(\rho, O_n)$ the set of explanations obtained this way for $O \otimes O_n$.

This immediately gives the idea of the following algorithm to compare two partial order automata $\mathcal{A}_1$ and $\mathcal{A}_2$. The algorithm follows paths of $\mathcal{A}_1$, by remembering a set of selected events in memory and the last state visited in $\mathcal{A}_1$, and on the other side, it maintains a set of pruned explanations of $\mathcal{A}_2$ that are compatible with the followed paths. As at each step of the construction, when choosing a new transition $t = (q, O_n, q')$ from $\mathcal{A}_1$, *i.e.*, extending some path ρ_1, we ensure that $O_{\rho_1.t}$ is a prefix of an order for at least one explanation. Note however that pruning does not guarantee finiteness of the memorized information in general.

As already mentioned, the set of configuration in $Xplore$ can grow arbitrarily, and nothing guarantees that the algorithm terminates in general. However, the class of locally synchronized partial order automata allow only regular sets of linearizations, and describe behaviors in which one can not iterate a behavior (or equivalently recognize a part of a prefix within a cycle) without terminating the preceding occurrence of this cycle.

THEOREM 5.2. *If $\mathcal{A}_1$ and $\mathcal{A}_2$ are locally synchronized, then the order inclusion algorithm terminates.*

PROOF. Suppose that at some stage, an explanation ρ_2 obtained when recognizing an order $O_{\rho,1}$ contains more than $|\mathcal{A}|$ states from which some cycle β coming from a silent connected component can be appended. In other words, ρ_2 is an explanation for paths of the form $\rho = \rho_{2,1}.\rho_{2,2}.\dots.\rho_{2,k}$, where occurrence of a cycle can be inserted between each pair $\rho_{2,i}$, $\rho_{2,i+1}$. As it is of size greater that $\mathcal{A}$, then ρ necessarily contains a cycle. As inserting β is optional to explain $O_{\rho,1}$ we know that all events that append the contents of this cycle result in appending an order that is completely concurrent with O_ρ. However, if β commutes with elements of a path of size greater than $|\mathcal{A}|$, then $\mathcal{A}$ is not locally synchronized. So, all possible insertion of cycles in an acyclic explanation can occur between transitions of a path located in a suffix of this path of size at most $|\mathcal{A}|$. Hence, there are less than $|\mathcal{A}|$ cycles in any path of $\mathcal{B}$. Now in a given path ρ of $\mathcal{A}_2$, if the i^{th} occurrence of an event in a particular order O_j is marked, then the preceding occurrences are also marked. As we do not add to pruned path sequences that end with unmarked order (otherwise these sequences would not be minimal) in a path of size greater than $|\mathcal{A}|^2.max$ where max is the size of the largest order in $\mathcal{A}_2$, there is necessarily a sequence of transitions carrying only marked events. Hence, for every path of $\mathcal{A}_1$ followed, there is only a finite number of corresponding shapes for subsequences describing matching paths of $\mathcal{A}_2$, and the algorithm terminates. □

PROPOSITION 5.3. *If H is a locally synchronized HMSC, then $\mathcal{A}_{H,\mathcal{O}^V}$ and $\mathcal{A}_{H,\mathcal{O}^V_{\setminus C}}$ are locally synchronized partial order automata.*

ALGORITHM 1: Checking inclusion

Input: Two partial order automata, $\mathcal{A}_1$, $\mathcal{A}_2$
Output: true if $\mathcal{F}_{\mathcal{A}_1} \subseteq \mathcal{F}_{\mathcal{A}_2}$, false otherwise
Visited=$\emptyset$;
// Configurations remember a path of $\mathcal{A}_1$ and several compatible paths
// of $\mathcal{A}_2$ with information on how events of $\mathcal{A}_2$ are used for embedding
$X_0 := \big((q_0^1, \varepsilon), \emptyset\big)$; // We start from the initial node of each automaton.
$Xplore := \{X_0\}$;
while $XPlore \neq \emptyset$ **do**
  Select $X = ((q_1, Mem_1), Exp = \{E_1, \dots, E_k\}))$ in Xplore ;
  // choose a particular configuration : a node of $\mathcal{A}_1$, and a partial description of all paths of $\mathcal{A}_2$ compatible with the chosen run of $\mathcal{A}_1$
  $Visited := Visited \cup X$;
  $Xplore := Xplore \setminus \{X\}$;
  for $(q_1, O_1, q_1') \in \longrightarrow_1$ **do**
    // for every transition leaving q_1 in $\mathcal{A}_1$
    $Mem_1' := \Pi(Mem_1, O_1)$;
    $Exp' := \{Succ(E_i, O_1) \mid E_i \in Exp\}$;
    // keep pruned explanations that embed the previously recognized order plus O_1
    if $Exp' = \emptyset$ *or* q_1' *final state and* $\epsilon \notin Exp'$ **then**
      // Trying to append order O_1 and showing it is still a prefix
      // of some path of $\mathcal{A}_2$ failed. So, we found path of $\mathcal{A}_1$ that
      // generates an order that is not a prefix of an order of $\mathcal{F}_{\mathcal{A}_2}$
      return false;
    end
    else
      // Continue exploration from $(q1', Mem_1')$ and explanations found
      $Xplore = Xplore \cup \{((q_1', Mem_1') \times Prune(Exp'))\}$;
    end
  end
end
return true;

PROOF. If H is locally synchronized, then for any cycle ρ, and pair of events e, f in M_ρ, we have $e^{(1)} \leq f^{(2)}$ and $f^{(1)} \leq e^{(2)}$ in $M_\rho \circ M_\rho$. As $\mathcal{O}^V$ is simply a projection, for any pair of events with labels in V, $e^{(1)}, f^{(1)}, e^{(2)}, f^{(2)}$ are ordered similarly. Hence $\mathcal{A}_{H,\mathcal{O}^V}$ is locally synchronized. Not every cycle of H becomes a cycle of $\mathcal{A}_{H,\mathcal{O}^V_{\setminus C}}$, as $\mathcal{O}^V_{\setminus C}$ may force to remove more events on transitions than a simple projection. However, cycles of $\mathcal{A}_{H,\mathcal{O}^V_{\setminus C}}$ are also obtained from cycles of H and labeled by projections, hence $\mathcal{A}_{H,\mathcal{O}^V_{\setminus C}}$ is also locally synchronized. □

We then have the obvious following corollary:

COROLLARY 5.4. *Non-interference is decidable for locally synchronized HMSCs.*

6. LOCAL AND CAUSAL NON INTERFERENCE

We now turn to other types of decidable classes, related to regularity. Indeed, inclusion problems become decidable as soon as one can recast the order comparison problem in a regular setting. It is however undecidable whether an HMSC or a partial order automaton has a regular behavior, and one has to rely on syntactic subclasses of the models such as locally-synchronized HMSCs/POAs as above to obtain decidability. We show in this section that several HMSC observation functions describing the discriminating power of a *single process* always define sets of orders that can be recognized by finite automata, that form a subclass of partial order automata. In this restricted setting, it is then possible to decide whether a process $p \in \mathbb{P}$ can detect occurrences of confidential actions. As HMSCs explicitly specify distribution of actions on processes, exhibiting the behavior of a fixed process within an HMSC specification is an easy task. In this section, we show that this *local* setting allows for the definition of two decidable notions of non-interference.

6.1. Local interference

Considering the attacker of a system as a single process $p \in \mathbb{P}$, with action labels in some alphabet $\Sigma_p = \alpha(E_p)$, we should assume that process p does not execute confidential actions, that is $C \cap \Sigma_p = \emptyset$. In a similar way, the observation power of a single process should be restricted to its own events, hence we can safely set $V = \Sigma_p$. The definition of non-interference (Def. 3.3) proposed in section 3 can accommodate this particular partition of the alphabet. From now on, we consider this restricted form of non-interference, and call it *local non-interference*.

For a single MSC, it is then defined as satisfaction of two inclusion problems, with $\mathcal{O}^V_{\setminus C}$ and $\mathcal{O}^p$ as observation functions. This property can be verified by checking whether $\uparrow(\alpha^{-1}(C)) \cap E_p = \emptyset$ that is checking if no causal consequence of a confidential action is located on process p. Recast in the setting of MSC coloring, this amounts to checking that p is not marked with a black token. As explained in Section 3.3, this can be performed in linear time. We can now look at local interference for HMSCs:

Definition 6.1. Let H be an HMSC over a set of processes $\mathbb{P}$, with alphabet $\Sigma = V \uplus C \uplus N$, such that $\Sigma = \uplus_{p \in \mathbb{P}} \Sigma_p$ with $V = \Sigma_p$. Then H is *locally non-interferent* w.r.t. process $p \in \mathbb{P}$ if $\mathcal{O}^{V,\bullet}_{\setminus C}(H) \equiv \mathcal{O}^{V,\circ}(H)$.

Intuitively, local interference holds when an observer can not distinguish in $\mathcal{F}_H$ behaviors that are concatenations of MSCs containing no confidential event, and other behaviors.

PROPOSITION 6.2. *For every HMSC H, one can build a (partial order) automaton $\mathcal{A}_{H,\mathcal{O}^p}$ that recognizes $\mathcal{O}^p(H)$. If $V = \Sigma_p$, then one can build (partial order) automata $\mathcal{A}_{H,\mathcal{O}^{V,\bullet}_{\setminus C}}$ and $\mathcal{A}_{H,\mathcal{O}^{V,\circ}}$ that recognize respectively $\mathcal{O}^{V,\bullet}_{\setminus C}(H)$ and $\mathcal{O}^{V,\circ}(H)$.*

PROOF. (Sketch) For any H, we can build a finite automaton $\mathcal{A}_p(H)$ that recognizes (linearizations of) projections of all MSCs in $\mathcal{F}_H$ on p. As concatenation of MSCs imposes a total order on events of the same process, these projections are concatenations of finite sequences of events (local projections of MSCs along transitions of H). Hence $\mathcal{A}_p(H)$ has transitions using labels of events located on process p, and just needs to remember the transition of H that is recognized (the current MSC under execution), and a bounded integer symbolizing the last event of the current MSC executed by p. Similarly, we can design an HMSC $H_{\setminus C}$ where transitions are labeled by MSCs that do not contain confidential events, and hence an automaton $\mathcal{A}'_p(H)$ that accepts only projections on p of sequences of MSCs with only visible (white) events. Hence $\mathcal{A}'_p(H)$ recognizes $\mathcal{O}^{V,\bullet}_{\setminus C}(H)$. Last, if $V = \Sigma_p$, then $\mathcal{O}^{V,\circ}(H) = \mathcal{O}^p(H)$. $\square$

It should be noted that $\mathcal{A}_{H,\mathcal{O}^p} = \mathcal{A}_p(H)$ and $\mathcal{A}_{H,\mathcal{O}^{V,\bullet}_{\setminus C}} = \mathcal{A}'_p(H)$ (and hence also $\mathcal{A}_{H,\mathcal{O}^{V,\circ}}$) are finite automata. Recast in the context of partial order automata, they are locally synchronized, have finite memory functions (that remember only the last event appended), and a unique composition operator $\circ$ based on the concatenation of sequences of events on process p.

COROLLARY 6.3. *The problem of deciding local interference of an HMSC H with respect to a given process $p \in \mathbb{P}$ is PSPACE-complete.*

PROOF. (Sketch) With the results of proposition 6.2, it remains to the languages of two automata (whence the complexity in PSPACE). For the hardness part, we can also show that any regular language inclusion problem can be encoded as a local interference problem. □

Local interference is decidable, and describes a situation where a process can discover that the running execution of the system contains *or will contain* a confidential action. However, local interference does not distinguish between a situation where an observation is a causal consequence of some confidential action and a situation where observation and confidential action highlighted by the interference are concurrent. This drawback also occurs in standard language-based interference settings, where causality is represented as interleaving, and one can not decide whether in a word $c.v$ actions c (confidential) and v (visible) are concurrent or not.

6.2. Causal interference

We first give a concrete example showing that interference is much more dangerous when the confidential event that is detected lays within the causal past of some observation. Nowadays, a lot of attention is devoted to privacy. However, it is well known that users spread a lot of information to visited sites when browsing the web. This information is not always local information (cookies, cache, etc.) that can be erased by users if needed. It can also be information stored elsewhere on the web: logs, forms, etc.. When observation of a causal consequence of a confidential action (Mr X has bought a book on commercial site Y) by an attacker indicates that a confidential operation has occurred, this may also mean that classified information might be available at some vulnerable site (the credit card details of X are stored somewhere on Y's website). Hence, characterizing interference where confidential actions and observations are causally related, is important.

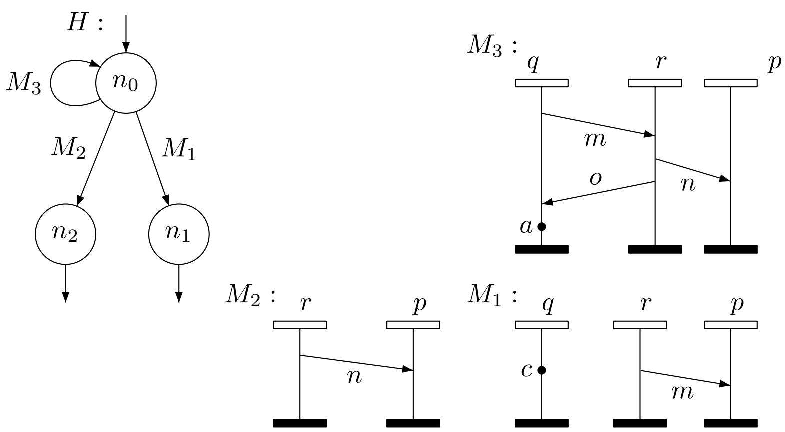

Fig. 3. An interferent HMSC

Example 6.4. On the HMSC depicted in Figure 3, the projection of MSCs recognized by H on p is the language $(?n)^*.(?m+?n)$, and every MSC with projection on p in $(?n)^*.?m$ is the projection of a concatenation of several occurrences of M_3, followed by one occurrence of M_1, which contains a confidential event. According to definition 6.1, this HMSC is locally interferent. However, when observing arrival of message m, process p can deduce that it is currently executing *a behavior in which a confidential action occurs,* but not that this action *has already occurred*.

This means in particular that NI does not always characterize a cause to effect relation among hidden actions and observation. To overcome this weakness of language-based information flow characterizations, the notion of NDC (Non-Deducibility on Composition) was proposed to detect when confidential actions *cause* observable effects. Formally, NDC says that a system S composed with any process R (that enables/forbids confidential events) is observationally equivalent to S.

In the rest of this section, we propose a decidable notion of *causal interference* (still with respect to a fixed attacker $p \in \mathbb{P}$). It emphasizes on causal dependencies between confidential and visible actions of the system. Bearing in mind that a black event located on process p is a consequence of a confidential event, we show that causal dependencies can be highlighted in terms of an observation function built using the black/white tokens attached to events and processes within an MSC. We want to check if a process p can detect whether some confidential action has occurred in the causal past of its observed events. In other words, we have to check whether all projection on p of an execution of H that contains a black event, only have equivalent projections that do not contain black events.

Definition 6.5. For an HMSC H and a process $p \in \mathbb{P}$, H is *causally non-interferent* with respect to p if for every MSC M in $\mathcal{F}_H$ such that M contains a black event on process p, there exists another MSC M' in $\mathcal{F}_H$ such that:

- M' contains no black event on process p, and
- $\mathcal{O}^p(M) = \mathcal{O}^p(M')$

Causal non-interference is weaker that NDC: it compares the observations of an HMSC with the observations that are still possible whitout confidential events. NDC compares a behavior of a specification with a specification controlled by a process R, in which some confidential events can be allowed.

THEOREM 6.6. *For a fixed set of processes $\mathbb{P}$, deciding causal non-interference of an HMSC H with respect to a process $p \in \mathbb{P}$ is PSPACE-complete.*

We prove this theorem in several steps. We first use the result of Proposition 3.6, *i.e.*, the fact that black/white coloring of processes at the end of a sequence of concatenated MSCs can be done by remembering the status of processes after each MSC. This property holds for MSCs built along paths of HMSCs, and is used (in Proposition 6.7) to build HMSCs that recognize MSCs that belong to $\mathcal{F}_H$ and after which a fixed process is black (or similarly remains white). These HMSCs contain nodes of H, but remember for each node n whether processes are black or white after an MSC built along a path ending in n. Then causal interference will then be reduced to an inclusion problem of finite automata that recognize sequences of actions along a process.

PROPOSITION 6.7. *Let H be an HMSC, $p \in \mathbb{P}$, and $\Sigma = C \uplus V \uplus N$. Then, one can build:*

- *an HMSC $H^{B,p}$ that recognizes MSCs from $\mathcal{F}_H$ after which p is a black process.*
- *an HMSC $H^{W,p}$ that recognizes MSCs from $\mathcal{F}_H$ after which p is a white process.*

of sizes in $O(|H|.2^{|\mathbb{P}|})$.

PROOF. (Sketch) The nodes of the HMSCs built in the proof memorize a node of the original HMSC, to which is added information on the color of each process: according to Proposition 3.6, this is the only information needed to remember the color of all processes in an MSC M_ρ assembled along a path ρ of H. Accepting nodes require p to be black in $H^{B,p}$, and white in $H^{W,p}$. $\square$

We are now ready to prove theorem 6.6:

PROOF. (of theorem 6.6) Following the construction of $H^{B,p}$ or $H^{W,p}$, we can define automata $\mathcal{A}_p^{B,p}$ and $\mathcal{A}_p^{W,p}$ that recognize the projections of $H^{B,p}$ or $H^{W,p}$ on process p. Let us denote by $\mathcal{O}^{B,p}(H) = \{\mathcal{O}^p(M) \mid M \in \mathcal{F}_H \wedge p \text{ is black after } M\}$ the observation function that returns the projection and by $\mathcal{O}^{W,p}(H) = \{\mathcal{O}^p(M) \mid M \in \mathcal{F}_H \wedge p \text{ is white after } M\}$. Clearly, we have $\mathcal{L}(\mathcal{A}_p^{B,p}) = \mathcal{O}^p(H^{B,p}) = \mathcal{O}^{B,p}(H)$ and $\mathcal{L}(\mathcal{A}_p^{W,p}) = \mathcal{O}^p(H^{W,p}) = \mathcal{O}^{W,p}(H)$, so $\mathcal{O}^{B,p}(H)$ and $\mathcal{O}^{W,p}(H)$ are recognized by finite automata.

Deciding causal interference of H with respect to $p \in \mathbb{P}$ consists in deciding the inclusion problem $\sqsubseteq_{\mathcal{O}^{B,p},\mathcal{O}^{W,p}}$ for H, that is checking whether $\mathcal{L}(\mathcal{A}_p^{B,p}) \subseteq \mathcal{L}(\mathcal{A}_p^{W,p})$. Clearly, if H is of size n, then $H^{B,p}$ and $H^{W,p}$ are of size in $O(n.2^{|\mathbb{P}|})$, and so are $\mathcal{A}_p^{B,p}$ and $\mathcal{A}_p^{W,p}$. Then, checking inclusion of $\mathcal{L}(\mathcal{A}^{B,p})$ into $\mathcal{L}(\mathcal{A}_p^{B,p})$ is equivalent to checking $\mathcal{L}(\mathcal{A}_p^{B,p}) \cap \mathcal{L}(\overline{\mathcal{A}^{B,p}}) = \emptyset$. Emptiness of regular language is an NLOGSPACE problem, but the size of the automaton that recognizes the intersection is in $O(n.2^{|\mathbb{P}|}.2^{n.2^{|\mathbb{P}|}})$, that is inclusion can be performed with space in $O(log(n) + |\mathbb{P}| + n.2^{|\mathbb{P}|})$. For a fixed set of processes, the space needed to check causal interferences is hence polynomial in the size of the input HMSC.

Like for local non-interference, the hardness result can be proved by polynomial encoding of a regular language inclusion problem. Given two regular languages L_1, L_2, one first designs two HMSCs H_1, H_2 with initial nodes n_0^1, n_0^2 such that $\mathcal{O}^p(H_i) = L_i$, for $i \in \{1, 2\}$. Then, using again the construction illustrated in Figure 2, we consider MSC M'_c that contains one confidential event on some fresh process $P_c \notin \mathbb{P}$, followed by messages from P_c to all processes in $\mathbb{P}$, and at last, an HMSC H that contains all transitions and accepting nodes of H_1, H_2, an initial node n_0 and an additional transition $t_1 = (n_0, M'_c, n_0^1)$. Any path of H starting with transition t_1 generates an MSC in which p is black, and whose projection on p is in L_1. Other paths that do not start with t_1 generate MSCs from $\mathcal{F}_{H_2}$, and in particular MSCs in which p is white and whose projection on p is in L_2. Hence, H is causally interferent with respect to p if and only if $L_1 \subseteq L_2$. $\square$

Causal interference can be checked in $O(log(|H|) + |\mathbb{P}| + |H|.2^{|\mathbb{P}|})$. It is polynomial in space in the size of the HMSC, and exponential in the number of processes, but HMSC specifications are usually defined for small sets of processes. Also remark that reusing the construction of $H^{W,p}$, we can easily design an automaton recognizing $\mathcal{O}_{\setminus C}^{V,\circ}(H)$ as soon as $V = \Sigma_p$.

7. DECLASSIFICATION

Non-interference considers confidential information as secrets that should remain undisclosed along all runs of a system. This point of view is too strict to be of practical interest: In many cases, confidentiality of a secret action has a limited duration and secrets can be downgraded. Consider the following example: a user wants to buy an item online, and pays by sending his credit card information. Everything from this transaction between the online shop and the buyer (even if encryption is used) should remain secret. Within this setting, all payment steps should be considered confidential, and flow from these actions to observable events should be prevented. However, if a buyer uses a one time credit card (i.e. a virtual credit card number generated on request that can be used only once for a transaction), then all information on the card is valueless as soon as the payment is completed. Hence, after completing the transaction, learning that a payment occurred is harmless and the sequence of interactions implementing a secured online payment need not be kept secret. This declassification possibility was first proposed as *conditional interference* by [Goguen and Meseguer 1982] and later defined in [Rushby 1992] as intransitive interference. Intransitive non interference (INI) can be formulated as follows: for any run of the system containing a confidential action that is not downgraded subsequently, there is a run with no classified action (all confidential actions are downgraded) which is equivalent from the observer's point of view.

Usually, INI is defined using a pruning function that removes from a run all confidential actions that are not declassified, and compares observations of pruned and normal runs (see [Gorrieri and Vernali 2011] for a definition of INI for transition systems). From now on, we assume that the alphabet $\Sigma = C \uplus V \uplus N$ contains a particular subset $D \subseteq V \uplus N$ of declassification events. Intuitively, declassification events downgrade all their confidential causal predecessors.

Definition 7.1. Let M be an MSC. An event $e \in E_M$ is *classified* if it is a confidential event ($\alpha(e) \in C$), it has an observable successor $v \in V$ and it is not declassified before v, i.e. there exists no d such that $e \leq d \leq v$ and $\alpha(d) \in D$. We denote by $Clas(M)$ the set of classified events of M.

The observation function $\mathcal{O}^V_{\setminus C,D}$ is defined by $\mathcal{O}^V_{\setminus C,D}(M) = \mathcal{O}^V(M \setminus Clas(M))$. An MSC M is *intransitively non-interferent* (INI) iff $\mathcal{O}^V_{\setminus C,D}(M) = \mathcal{O}^V(M)$.

We can characterize INI in a single MSC M as a property depending on the causal order in M and on the sets of confidential, declassification, and observable events.

PROPOSITION 7.2. *An MSC M is intransitively non-interferent w.r.t. an alphabet $\Sigma = C \uplus V \uplus N$ and a set of declassification letters D iff for every pair of events $c \leq v$ such that $\alpha(c) \in C$ and $\alpha(v) \in V$, we have $(\uparrow(c) \cap \downarrow(v)) \cap \alpha^{-1}(D) \neq \emptyset$.*

This proposition means that a declassification must occur between every confidential event and a causally related visible event. We now define observation functions for HMSCs and propose a definition of intransitive non interference for HMSCs. We define $\mathcal{O}^V_{II,D}(H) = \{\mathcal{O}^V(M) \mid M \text{ is not } \mathit{INI}\}$ and $\mathcal{O}^V_{INI,D}(H) = \{\mathcal{O}^V(M) \mid M \text{ is } \mathit{INI}\}$. We follow the definition of [Gorrieri and Vernali 2011] to define INI for HMSCs. An HMSC is INI if for every intransitively interferent (II for short) MSC M in $\mathcal{F}_H$, there exists another MSC M' in $\mathcal{F}_H$ such that M' that is INI and such that $\mathcal{O}^V(M') = \mathcal{O}^V(M)$.

Definition 7.3. An HMSC is intransitively non-interferent w.r.t. a declassification alphabet D if $\mathcal{O}^V_{INI,D}(H) = \mathcal{O}^V(H)$.

Obviously, $\mathcal{O}^V_{INI,D}(H) \subseteq \mathcal{O}^V(H)$, so proving INI boils down to proving $\mathcal{O}^V(H) \subseteq \mathcal{O}^V_{INI,D}(H)$. Note that all II MSCs are also interferent, and that checking non-interference amounts to checking INI with $D = \emptyset$. This remark extends to HMSCs: all intransitively interferent HMSCs are also causally interferent, and checking causal interference amount to checking INI with $D = \emptyset$. We then establish the following result:

THEOREM 7.4. *INI for HMSCs is undecidable. For a fixed set of processes, if $V \subseteq \Sigma_p$, then INI is PSPACE-complete.*

We prove the decidability part of this theorem in three steps detailed below. We first show that INI can be decided for a sequence of MSCs by remembering only the shape of causal chains originating from confidential events instead of the whole sequence. We then show that one can design an HMSC H^{II} that recognizes II MSCs of $\mathcal{F}_H$, and similarly an HMSC H^{INI} that recognizes INI MSCs of $\mathcal{F}_H$. An immediate consequence is that $\mathcal{O}^V_{INI,D}(H)$ can be recognized by a finite automaton if $V \subseteq \Sigma_p$. A second consequence is that checking INI is PSPACE-complete. Let us first show that INI can be decided in a compositional way.

PROPOSITION 7.5. *Let M_1, M_2 be two MSCs. Then, $M_1 \circ M_2$ is INI if and only if M_1 and M_2 are INI, and for each pair of events $c \in M_1$, $v \in M_2$ such that $\alpha(c) \in C$, $\alpha(v) \in V$, and $c \leq_{1\circ 2} v$, there exists a process q, with*

- *$c \leq f$, where f is the maximal event on process q in M_1,*
- *$f' \leq v$, where f' is the minimal event on q in M_2,*

and an event d such that $\alpha(d) \in D$, and $c \leq d \leq f$ or $f' \leq d \leq v$.

This proposition can be intuitively seen as a property of causal chains. A causal chain from c to v is a sequence of events $c \leq e_1 \leq \dots e_n \leq v$. We say that a chain from c to v is declassified if $\alpha(e_i) \in D$ for some $i \in 1..n$. Then an MSC is INI if for any pair (c, v) of confidential/visible events such that $c \leq v$ there exists at least one declassified causal chain from c to v. If so, the confidential event c is guaranteed to be declassified by the occurrence of some declassifying action **before** the execution of v occurs.

A causal chain from c to v in $M_1 \circ M_2$ can be decomposed into a chain from c to the maximal event f on a process q in M_1, a causal ordering from f to a minimal event f' located on process q in M_2 coming from the sequential composition of M_1 and M_2, and then a causal chain from the minimal event f' on q to v. However, one does not need to know precisely the contents of M_1 to decide whether $M_1 \circ M_2$ is INI. It suffices to remember for each process p the confidential events of M_1 that are not yet declassified and are predecessors of the maximal event executed by process p in M_1.

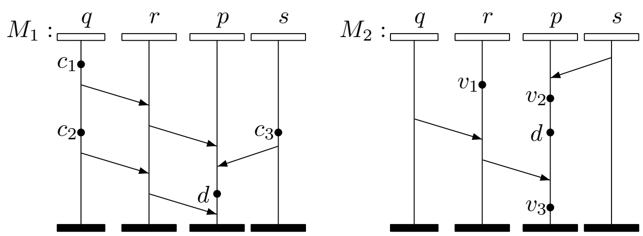

Fig. 4. An example of non INI sequence of MSCs

On the example depicted in Figure 4, MSC M_1 (left) contains three confidential actions c_1, c_2, c_3, and a declassification operation d. On the right, MSC M_2 contains three visible actions v_1, v_2, v_3, and a declassification operation d. All other events belong to $\alpha^{-1}(N)$. Both MSCs are INI, since no observation depends on a confidential action in M_1 or in M_2. However, in the concatenation $M_1 \circ M_2$, execution of v_1 or v_2 reveals the occurrence of c_2. Also note that c_1 is declassified by the first occurrence of d in M_1. This example is particularly interesting, as it shows that in order to abstract an arbitrarily long execution, it is not sufficient to remember a boolean value indicating whether there exists a not yet declassified action on a process, as two confidential events can be declassified via different ways. Indeed, some confidential actions could be declassified for a process while some others could not, even when located on the same process.

We can characterize II MSCs in a set $\mathcal{F}_H$ by remembering finite sets of shapes of causal chains. In order to define these shapes, let M be an MSC, let c be a confidential event in M. We define a function $cl(c, M) : \mathbb{P} \longrightarrow \{\perp, +, \top\}$ such that $cl(c, M)(p) = \perp$ if there exists no causal chain from c to an event located on p, $cl(c, M)(p) = +$ if there exists a causal chain from c to a maximal event f located on p, and $(\uparrow c \cap \downarrow f) \cap \alpha^{-1}(D) = \emptyset$, and $cl(c, M)(p) = \top$ otherwise. This function classifies processes according to the existence and classification degree (declassified or not) of causal chains between the confidential event c and the last event seen on each process. For a set $\mathbb{P}$ of processes, any such map $cl(c, M)$ can have at most $3^{|\mathbb{P}|}$ distinct values. Let $\mathcal{C}l = \{\perp, +, \top\}^{\mathbb{P}}$ denote the set of all maps. By proposition 7.5, $M_1 \circ M_2$ is not INI if M_1 or M_2 is not INI, or if there exists $c \in M_1$ and $v \in M_2$ such that:

- there exists a process p such that $cl(c, M_1)(p) = +$, and an event f located on p in M_2, such that no causal chain from f to v is declassified.
- for every process q such that $cl(c, M_1)(q) = \top$ there exists no event $f \leq v$ located on q in M_2, and v is not located on q.

One can furthermore compute $cl(c, M_1 \circ M_2 \circ \cdots \circ M_k)(p)$ incrementally with finite memory:
$\mathbf{cl(c, M_1 \circ M_2)(p)} = \perp$ if $cl(c, M_1)(p) = \perp$, and if there exists no pair of events $e \leq f$ in M_2 with f is located of p, and $cl(c, M_1)(\phi(e)) \neq \perp$.
$\mathbf{cl(c, M_1 \circ M_2)(p)} = +$ if $cl(c, M_1)(p) \in \{\perp, +\}$, there exists a process q such that $cl(c, M_1)(q) = +$, and a pair of events $e \leq f$ in M_2 such that e is minimal on q, f is maximal on process p, and furthermore, no causal chain from e to f is declassified, and for every process $q' \neq q$, if $cl(c, M_1)(q') = +$, then no declassified causal chain from an event on q' to f exists in M_2, if $cl(c, M_1)(q') = \top$ then no causal chain from an event on q' to f exists in M_2.

$\mathbf{cl(c, M_1 \circ M_2)(p)} = \top$ if $cl(c, M_1)(p) = \top$, or

- there exist a process q such that $cl(c, M_1)(q) = +$ and a declassified chain from an event e located on process q to an event f located on process p, or

- there exist a process q such that $cl(c, M_1)(q) = \top$, and a causal chain from an event e located on process q to an event f located on process p.

Last, $cl(c, M_1 \circ M_2)(p) = \bot$ if $cl(c, M_1)(p) = \bot$ and M_2 does not contain a pair of events $e \leq f$ such that e is located on q with $cl(c, M_1)(q) \neq \bot$, and f is located on p.

Now, if M_1 contains two confidential events c_1, c_2 such that $cl(c_1, M_1) = cl(c_2, M_1)$, then $cl(c_1, M_1 \circ M_2) = cl(c_2, M_1 \circ M_2)$. It means that to detect interferences, one does not have to remember events, but only the shape of causal relations (existing, declassified or not) from confidential events to their successors on each process. Furthermore, at most $3^{|\mathbb{P}|}$ distinct shapes can appear in an MSC, so one can check INI along arbitrarily long sequences of MSCs with finite memory.

PROPOSITION 7.6. *Let H be an HMSC, with labeling alphabet Σ and set D of declassification letters. Then, one can build an HMSC H^{II} generating all II MSCs in $\mathcal{F}_H$ and an HMSC H^{INI} generating all INI MSCs in $\mathcal{F}_H$, with sizes at most $2.|H|.2^{3^{|\mathbb{P}|}}$*

PROOF. (Sketch) We build HMSC H^{II} as follows: a state (n, b, X) of H^{II} memorizes a node n of H, a boolean b indicating whether an interference has been detected, and a set $X = \{cl_1, \ldots cl_\ell) \subseteq \mathcal{C}l$, where each cl_i is a function from $\mathbb{P}$ to $\{\bot, +, \top\}$ that memorizes the shape of causal chains from a confidential event to maximal events on processes. H^{II} follows transitions of H, and updates cl_i's. For each new confidential event c occurring in a transition labeled by an MSC M, a new function $cl(c, M)$ is added to memorized shapes in X. As soon as an interference is detected, b is set to true. Accepting states of H^{II} are of the form (n, b, X) where n is accepting in H, and b is true. H^{INI} is built similarly, but with accepting states of the form (n, b, X) with n accepting in H and b false. $\square$

We are now ready to prove Theorem 7.4:

PROOF. (of Theorem 7.4) Undecidability is easily obtained from undecidability of causal interference, and by setting $D = \emptyset$. Let us now consider the decidability part, with $V \subseteq \Sigma_p$. Following the proof of proposition 7.6, one can build an automaton $\mathcal{A}_p(H^{INI})$ of size at most $2.|H|.2^{3^{|\mathbb{P}|}}$ that recognizes $\mathcal{O}^V(H^{INI})$. One can easily prove that when $V \subseteq \Sigma_p$, we have $\mathcal{O}^V(H^{INI}) = \mathcal{O}^V_{INI,D}(H)$, and hence $\mathcal{L}(\mathcal{A}_p(H^{INI})) = \mathcal{O}^V_{INI,D}(H)$, i.e. $\mathcal{O}^V_{INI,D}(H)$ is recognized by a finite automaton.

From proposition 6.2, we can build an automaton $\mathcal{A}_p(H)$ of size in $O(k.H)$, where k is the maximal number of events in an MSC of H, that recognizes $\mathcal{O}^V(H)$. Then it is sufficient to check whether $\mathcal{L}(\mathcal{A}_p(H)) \subseteq \mathcal{L}(\mathcal{A}_p(H^{INI}))$ to decide if H is intransitively interferent, which is again an inclusion problem that can be checked in space in $O(2.|H|.2^{3^{|\mathbb{P}|}})$. Hardness is proved by showing a polynomial reduction from a language inclusion problem to an INI problem with $D = \emptyset$. $\square$

The declassification setting can be refined to consider selective declassification. Following the definition of [Best and Darondeau 2012], in addition to the declassification alphabet D, we define a map $h : D \to 2^C$, where $h(\alpha_d)$ defines the labels of confidential events that an action with label α_d declassifies. Definition 7.1 easily adapts to this setting, simply by requiring that a causal chain from a confidential event c to a visible event v is declassified by an event d such that $\alpha(c) \in h(\alpha(d))$. We then say that an event c is classified if it is a confidential event ($\alpha(c) \in C$), it has an observable successor v, and it is not declassified by one of the actions that can declassify it, that is, $\alpha(c) \notin h\left(\alpha(\uparrow(c) \cap \downarrow(v)) \cap D\right)$. INI with selective declassification (INISD) adapts the definitions of INI to consider declassification without changing observations. Like for standard declassification, we can build an HMSC that recognizes INISD MSCs of $\mathcal{F}_H$. The only change w.r.t. INI is that one has to remember in the HMSC construction the label of confidential events from which chains originate, yielding automata of sizes in $2.|H|.2^{|C|.3^{|\mathbb{P}|}}$. If $V \subseteq \Sigma_p$, then $\mathcal{O}^V_{II,D}$ and $\mathcal{O}^V_{INI,D}$ are recognized by finite automata. We hence have:

COROLLARY 7.7. *INISD is undecidable for HMSCs. For a fixed set of processes, it is PSPACE-complete when $V \subseteq \Sigma_p$.*

8. RELATED WORK AND CONCLUSION

Related work. Non-interference was seldom studied for scenario formalisms. A former work considers non-interference for Triggered Message Sequence Charts [Ray et al. 2004]. The interference property is defined in terms of comparison of ready sets (sets of actions that are fireable after a given sequence of actions w). However, this work mainly considers finite scenarios, and does not address decidability and complexity issues.

A first work considering non-interference for true concurrency models appears in [Busi and Gorrieri 2009]. The authors consider interference for elementary nets (*i.e.*, nets where firing rules allow places to contain at most one token). They characterize *causal places*, where firing a high-level transition causally precedes the firing of a low-level one and *conflict places*, where firing a high-level transition inhibits the firing of a low-level one. Reachability of causal or conflict places is shown equivalent to BNDC (Bisimulation-based NDC, the variant of Non-Interference using bisimulation instead of language equality). In [Gorrieri and Vernali 2011], the notion of intransitive non-interference from [Rushby 1992] is revisited for transition systems, and non-interference with downgraders is considered for elementary nets. A structural characterization is given in terms of reachable causal and conflict places. As in [Busi and Gorrieri 2009], causal and conflict places are characterized in terms of possible fireable sequences of transitions, hence considering the interleaving semantics of the net.

Darondeau *et al.* [Best et al. 2010] study (B)NDC and INI for **unbounded** labeled Petri nets, and extend their results to selective declassification in [Best and Darondeau 2012]. They obtain decidability of these properties for injectively labeled nets by a very clever exploitation of specific decidability results for language inclusion, which is undecidable for general Petri nets languages. The characterization relies on sequences of transitions, and not on causal properties of nets.

A contrario, Baldan et al [Baldan and Carraro 2014] emphasize the fact that characterizing BNDC in terms of structural conditions expressing causality or conflict between high and low-level transitions, is a way to provide efficient algorithms to check interference. They propose a definition of complete unfolding w.r.t. non-interference, and reduce BNDC for safe nets to checking that a complete unfolding is weak-conflict and weak-causal place free. Weak causal places characterize dependencies and conflicts between high and low transitions. Their results show that interference can be detected in concurrent models without relying on interleaving semantics. They only hold for safe nets, *i.e.*, for finite state systems.

Conclusion. We proposed a partial order framework for information flow properties analysis, that relies on comparisons of sets of partial orders depicting observations of execution of systems. We proved that inclusion of observed orders and non-interference is undecidable in general. To alleviate this problem, we proposed partial order automata, as a model to recognize observations of executions. We then identified subclasses of partial order automata for which inclusion of languages is decidable. Non-interference of locally synchronized HMSCs falls into this decidable setting, and is hence decidable. A different approach to obtain decidability in this partial order framework is to restrict the kind of observation functions that can be used. This is a sensible approach, as it amounts to restrict the power of attackers. If one considers that visible events are observed by a single process of the system, most of observation functions applied to HMSCs define regular languages. As a consequence, several notions of local non-interference and their extensions with declassification, are decidable. We show that local versions of non-interference are PSPACE-complete problems, and give decision procedures that never compute the interleaving semantics of the original HMSC.

So far, partial order automata are mainly used as an intermediate technicality to prove decidability of non-interference for locally-synchronized HMSCs when several processes can observe the system. However, this model is more general than HMSCs. A possible refinement of the landscape is to consider decidability of interference for partial order automata that generate sets of orders with non-regular linearization languages. We conjecture that decidability of inclusion can be generalized to some subclasses of non-regular partial order automata, some classes of graph grammars, or more generally to subclasses of models with bounded-split width [Aiswarya et al. 2014].

Another line of research is to consider security issues when an attacker can interact with the system in order to gain information (active interference), or when he can get information on the current configuration of the system (state-based interference). Extending definitions of information flows in HMSCs to quantify the amount of information disclosure by mean of measures (e.g. probability measure, average number of bits leaked per action,...) is also a challenging task.

REFERENCES

C. Aiswarya, P. Gastin, and K. Narayan Kumar. 2014. Verifying Communicating Multi-pushdown Systems via Split-Width. In *Proc. of 12th Int. Symp. on Automated Technology for Verification and Analysis, ATVA 2014 (LNCS)*, F. Cassez and J.-F. Raskin (Eds.), Vol. 8837. Springer, 1–17.

R. Alur and M. Yannakakis. 1999. Model Checking of Message Sequence Charts. In *Proc. of 10th Int. Conf. on Concurrency Theory, CONCUR '99 (LNCS)*, J. C. M. Baeten and S. Mauw (Eds.), Vol. 1664. Springer, 114–129.

P. Baldan and A. Carraro. 2014. Non-interference by Unfolding. In *35th Int. Conf. on Application and Theory of Petri Nets and Concurrency, PETRI NETS (LNCS)*, G. Ciardo and E. Kindler (Eds.), Vol. 8489. Springer, 190–209.

B. Bérard, L. Hélouët, and J. Mullins. 2015. Non-interference in Partial Order Models. In *Proc. of 15th Int. Conf. on Application of Concurrency to System Design, ACSD 2015*. IEEE Computer Society, 80–89.

E. Best and P. Darondeau. 2012. Deciding Selective Declassification of Petri Nets. In *Proc. of 1st Int. Cong. on Principles of Security and Trust, POST 2012 (LNCS)*, P. Degano and J. D. Guttman (Eds.), Vol. 7215. Springer, 290–308.

E. Best, P. Darondeau, and R. Gorrieri. 2010. On the Decidability of Non Interference over Unbounded Petri Nets. In *Proc. of SecCo (EPTCS)*, Vol. 51. 16–33.

B. Bérard and J. Mullins. 2014. Verification of Information Flow Properties under Rational Observation. In *Proc. of AVOCS 2014, ECEASST 70*.

N. Busi and R. Gorrieri. 2009. Structural non-interference in elementary and trace nets. *Mathematical Structures in Computer Science* 19, 6 (2009), 1065–1090.

B. Caillaud, P. Darondeau, L. Hélouët, and G. Lesventes. 2000. *HMSCs en tant que spécifications partielles et leurs complétions dans les réseaux de Petri*. RR-3970. INRIA.

D. D'Souza, R. Holla, K.R. Raghavendra, and B. Sprick. 2011. Model-checking trace-based information flow properties. *Journal of Computer Security* 19, 1 (2011), 101–138.

R. Focardi and R. Gorrieri. 2001. Classification of Security Properties (Part I: Information Flow). In *Foundations of Security Analysis and Design (LNCS)*, Vol. 2171. Springer-Vale, 331–396.

B. Genest, L. Hélouët, and A. Muscholl. 2003. High-Level Message Sequence Charts and Projections. In *Proc. of 14th Int. Conf. on Concurrency Theory, CONCUR'03 (LNCS)*, R. M. Amadio and D. Lugiez (Eds.), Vol. 2761. Springer, 308–322.

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

R. Gorrieri and M. Vernali. 2011. On Intransitive Non-interference in Some Models of Concurrency. In *FOSAD VI Tutorial Lectures (LNCS)*, Vol. 6858. 125–151.

L. Hélouët, H. Marchand, B. Genest, and T. Gazagnaire. 2014. Diagnosis from scenarios. *Discrete Event Dynamic Systems* 24, 4 (2014), 353–415.

ITU-T. 2011. *Z.120 : Message Sequence Charts (MSC)*. Technical Report. International Telecommunication Union.

H. Mantel. 2000. Possibilistic Definitions of Security - An Assembly Kit. In *Proc. of the 13th IEEE Computer Security Foundations Workshop, (CSFW'00)*. 185–199.

H. Mantel. 2001. Information Flow Control and Applications - Bridging a Gap. In *Proc. of FME 2001 (LNCS)*, Vol. 2021. 153–172.

F. Mattern. 1988. Time and global states of distributed systems. *in Proc. Int. Workshop on Parallel and Distributed Algorithms, Bonas, France, North Holland* (1988), 215–226.

A. Muscholl and D. Peled. 1999. Message Sequence Graphs and Decision Problems on Mazurkiewicz Traces. In *Proc. of 24th Int. Conf. on Mathematical Foundations of Computer Science, MFCS (LNCS)*, M. Kutylowski, L. Pacholski, and T. Wierzbicki (Eds.), Vol. 1672. Springer, 81–91.

A. Muscholl and D. Peled. 2000. Analyzing Message Sequence Charts. In *Proc. of 2nd Workshop on SDL and MSC, SAM 2000*, E. Sherratt (Ed.). VERIMAG, IRISA, SDL Forum, 3–17.

A. Ray, B. Sengupta, and R. Cleaveland. 2004. Secure Requirements Elicitation Through Triggered Message Sequence Charts. In *Proc. of 1st Int. Conf. on Distributed Computing and Internet Technology, ICDCIT 2004 (LNCS)*, R. K. Ghosh and H. Mohanty (Eds.), Vol. 3347. Springer, 273–282.

J. Rushby. 1992. *Noninterference, Transitivity, and Channel-control Security Policies*. Technical Report CSL-92-02. SRI International.

Received XX ; revised YY ; accepted ZZ

A. APPENDIX: PROOFS

This appendix details missing proofs for propositions in the text. Proofs for propositions 3.6, 7.2 and 7.5 are straightforward consequences of definitions 3.5 and 7.1, and are not provided.

A.1. Proof of Proposition 3.4

Proposition 3.4. Let M be an MSC with labeling alphabet $\Sigma = C \uplus V \uplus N$ and set of events E. Then, M is *interferent* if and only if there are two events e, f such that $\alpha(e) \in C$, $\alpha(f) \in V$, and $e \leq f$.

Proof. We prove this lemma by showing the two directions of the implication.

First, let us suppose that there exists no pair of events e, f in M such that $\alpha(e) \in C$, $\alpha(f) \in V$, and $e \leq f$. If there is no event $e \in E$ such that in $\alpha(e) \in C$, then $M \setminus \uparrow (\alpha^{-1}(C)) = M$, and hence $\mathcal{O}^V(M) \equiv \mathcal{O}^V_{\setminus C}(M)$. If some confidential events exist in M, i.e., $\alpha^{-1}(C) \neq \emptyset$, then for each $e \in \alpha^{-1}(C)$, as there is no visible f causally dependent form e, we have $\uparrow (e) \cap \alpha^{-1}(V) = \emptyset$. In this case, $M \setminus \uparrow (\alpha^{-1}(C)) = M \setminus \alpha^{-1}(C)$. This means that $\mathcal{O}^V(M) \equiv \mathcal{O}^V_{\setminus C}(M)$, which yields the result.

$ii)$ Let us now prove the converse direction. Suppose that there exists a pair of events $e \leq f$ such that e is a confidential event, and f is a visible one. Then, e is not maximal, and hence $\mathcal{O}^V_{\setminus C}(M)$ does not contain f. As $\mathcal{O}^V(M)$ contains all observable events of M we can not have $\mathcal{O}^V(M) \sqsubseteq \mathcal{O}^V_{\setminus C}(M)$. This immediately implies that $\mathcal{O}^V_{\setminus C}(M) \not\equiv \mathcal{O}^V(M)$. □

A.2. Proof of Theorem 4.9

Theorem 4.9. Given a partial order automaton $\mathcal{A}$ with selection function *MaxEvt*, one can decide if $\mathcal{A}$ is threaded. Furthermore this problem is in co-NP.

Proof. A path satisfying the property of Def. 4.8 is said to be threaded. We first show that is it sufficient to consider elementary cycles of $\mathcal{A}$ extended by one transition to decide whether a POA is threaded. Consider an accessible cycle $\rho = q \xrightarrow{O_1} q_1 \dots \xrightarrow{O_k} q$ of $\mathcal{A}$ and the path $\rho' = \rho.(q, O_1, q_1)$ that extends ρ with one single transition. We call such a path an elementary sequence. Obviously, if there is an event e in O_1 such that $e^{(1)} \not\leq e^{(2)}$, then any path of $\mathcal{A}$ that ends with ρ' is not threaded, and hence $\mathcal{A}$ is not threaded.

Conversely, suppose that all elementary sequences of $\mathcal{A}$ are threaded, but that one can find a path ρ of $\mathcal{A}$ that is not threaded. That is, ρ is of the form $\rho_1.(q, O_1, q').\rho_2.(q, O_1, q')\rho_3$, and is such that some occurrence $e^{(i)}$ in the i^{th} occurrence of O_1 does not precede the $(i+1)^{th}$ occurrence of e in the $(i+1)^{th}$ occurrence of O_1. Clearly, the sequence of transitions $(q, O_1, q').\rho_2.(q, O_1, q')$ is not an elementary sequence, otherwise one would have $e^{(i)} \leq e^{(i+1)}$. However, $(q, O_1, q').\rho_2.(q, O_1, q')$ is obtained by insertion of elementary cycles in an elementary sequence $\rho_{el} = (q, O_1, q').\rho'_{el}.(q, O_1, q')$ starting and ending with transition (q, O_1, q'). In ρ, we have $e^{(1)} \leq e^{(2)}$, that is, there exists a causal chain $e^{(1)} \prec f_1 \prec \dots f_k \prec g_1 \prec \dots \prec g_{k'} \prec h_1 \prec \dots \prec h_{k''} \prec e^{(2)}$, where $f_1, \dots, f_k$ are events of $O_1^{(1)}$, $g_1, \dots, g_{k'}$ are events of $O_{\rho'_{el}}$, and $h_1, \dots, h_{k''}$ are events of $O_1^{(2)}$. Consider insertion of an elementary sequence by replacing transition (q_s, O_2, q_{s+1}) by an elementary sequence $\rho_s = (q_s, O_2, q_{s+1}) \dots (q_s, O_2, q_{s+1})$ in ρ'_{el}. Then if there exists no event of O_2 in the causal chain from $e^{(1)}$ to $e^{(2)}$, then the causal chain is preserved by replacement of one transition by this elementary sequence. Now, supposing that O_2 contains a set of events $g_t \prec \dots g_{t'}$ of the causal chain, we still have $e^{(1)} \leq g_t^{(1)}$ in O_{ρ_s}. Similarly, we have $g_t^{(1)} \leq g_{t'}^{(1)}$, and due to properties of elementary sequences, we have $g_{t'}^{(1)} \leq g_{t'}^{(2)}$. Now, as the selection function recalls last occurrences of events, and uses the same operators that depend only on chosen transitions, we will have $g_{t'}^{(2)} \leq h_1 \leq \dots e^{(2)}$. Similar reasoning holds when inserting several occurrences of elementary sequences between two occurrences of O_1. As all paths containing two consecutive occurrences of some order can only be obtained by such insertions, this allows to conclude that ρ is threaded, which contradicts our starting hypothesis. Hence $\mathcal{A}$ is threaded iff all its elementary sequences are threaded.

Now, let us consider the complexity part. Finding an acyclic path ρ containing twice transitions (q_1, O, q_2) can be done non-deterministically in polynomial time, by choosing non-deterministically a path starting from q_2, and stopping as soon as some transition was already encountered, or when reaching the second occurrence of transition (q_1, O, q_2). Followed path are of length at most $|\longrightarrow|$. Appending an order to an existing one and maintaining a set of selected events can be done in polynomial time, as it suffices to add a bounded number of elements (events and covering relation). Denoting by m the maximal size of an order in $\mathcal{L}$, each step hence adds at most m events and $|\mathcal{L}| \times m^2$ elements to the covering relation built so far. For a chosen event e, one can maintain during construction of the order a set S of at most $|\mathcal{L}|.m$ events that are both in the set of events kept by the selection function, and successors of e. Then, if one ends with a second occurrence of (q_1, O, q_2), it is easy to check that $e^{(2)}$ is a successor of some event of S. ◻

A.3. Proof of Theorem 4.11

Theorem 4.11 Let $\mathcal{A}$ be a threaded partial order automaton with a selection function Π that memorizes a bounded number K of events. Then one can effectively decide if $\mathcal{A}$ is locally synchronized.

Proof. Let us consider a path $\rho_1.\rho_2$ such that ρ_2 is a cycle, and let e, f be a pair of events in O_{ρ_2}. The commutation graph of O_{ρ_2} is strongly connected iff for every pair of events e, f in O_{ρ_2} there exists a chain from $e^{(1)}$ to $f^{(1)}$ or from $e^{(1)}$ to $f^{(2)}$, and another chain from $f^{(1)}$ to $e^{(1)}$ or from $f^{(1)}$ to $e^{(2)}$. Following the same argumentation as in the proof of theorem **??**, as $\mathcal{A}$ is threaded, insertion of new occurrences of existing orders in a path can only extend the length of existing causal chains, but cannot "break" causal chains: if $e \leq f$ in $\rho_2.\rho 2'.\rho_2.\rho 2'$, then $e \leq f$ in $\rho_2.\rho_3.\rho 2'.\rho_2.\rho_3.\rho 2'$ for any cycle ρ_3. It is hence sufficient to consider occurrences of cycles that contain at most one occurrence of each transition to check existence of disconnected communication graphs.

The algorithm follows a path of $\mathcal{A}$. At some point, it non-deterministically chooses an acyclic path and checks if its communication graph is strongly connected. ◻

ALGORITHM 2: Checking local synchronization

Input: Partial Order Automaton $\mathcal{A}$
Output: Non-deterministically return true if $\mathcal{A}$ is locally synchronized, stops otherwise
randomly pick up a number d between 1 and $|A|.2^K$
`// By considering runs on this length, we have visited all possible memory configurations and transitions`
Mem $= \emptyset$;
$cs = n_0$;
`// current state`
while $d > 0$ **do**
  randomly select a transition (cs, O, n) and compute $\Pi($*Mem* $\otimes_{\Lambda(cs,O,n')} O)$;
  cs=n;
  d=d-1;
end
randomly select an **acyclic** sequence of transition $\rho = (cs, O_i, n_1) \ldots (n_k, O_k, n)$;
Compute the communication graphs G of O_ρ
if G *is not strongly connected* **then**
  return $\mathcal{A}$ *is not locally synchronized*
end
STOP;

A.4. Proof of Proposition 4.15

Proposition 4.15. For every HMSC H, observation functions $\mathcal{O}^V, \mathcal{O}^V_{\setminus C}, \mathcal{O}^p$ are finitely decomposable, with bounds $m \leq |\mathbb{P}|^2$ and $c \leq |\mathbb{P}|^3$.

Proof. Let us first consider $\mathcal{O}^V$. For every pair of MSCs M_1, M_2, we know that the set of events in $O = \mathcal{O}^V(M_1 \circ M_2) = (E_O, \prec_O, \lambda_O)$ is exactly the union of the the set of events in $\mathcal{O}^V(M_1)$ and $\mathcal{O}^V(M_2)$. We can reuse the result in [Genest et al. 2003] on projections of HMSCs. For an event in e in $\mathcal{O}(M_1)$, let us call $F(e) = \{\phi(f) \mid \exists f, e \leq_1 f\}$ the set of processes that contain a successor event for e. Let us define $Dead(e) = \bigcup_{e \prec_O e'} F(e')$, and $LiveF(e) = F(e) \setminus Dead(e)$. Intuitively, $LiveF(e)$ represents the set of processes that may contain an observable successor of e appended after M_1. We call $ToCheck(M_1) = \{e \in \mathcal{O}(M_1) \mid LiveF(e) \neq \emptyset\}$ the events of M_1 that can still have a immediate successor in $\mathcal{O}^V(M_1 \circ M_2)$ for an arbitrary M_2. It was shown in [Genest et al. 2003] that for sequences of MSCs ρ of arbitrary size it is sufficient to remember $ToCheck(M_\rho)$ and $LiveF(e)$ for every e in $ToCheck(M_\rho)$ to be able to build $\mathcal{O}^V(M_\rho \circ M_2)$ for any M_2. This work also shows that for every ρ, $ToCheck(M_\rho)$ is of size smaller than $|\mathbb{P}|^2$. Then, from this information, building $\mathcal{O}^V(M_\rho \circ M_2)$ simply consists in assembling $\mathcal{O}^V(M_\rho)$ and $\mathcal{O}^V(M_2)$ as a union of both coverings, to which causalities from every event e in $Tocheck(M^\rho)$ to minimal events of $\mathcal{O}^V(M_2)$ located on $LiveF(e)$. We refer readers to [Genest et al. 2003] for details and proof of correctness of the construction. One can see that the composition operation used to assemble observations of MSCs is the same, and needs only to memorize a finite number of events $e_1, \ldots, e_k$ with $k \leq |\mathbb{P}|^2$. However, there is an additional information carried by $LiveF(e)$, that is used to decide whether pairs of events in $ToCheck(M_\rho) \times \mathcal{O}^V(M_2)$ should be in $\prec$: One needs only to remember a set of processes attached to each event e_i by map $LiveF()$. This can be encoded by a map $h : 1..|\mathbb{P}|^2 \rightarrow 2^P$, and there is only a finite number of such maps. Let us denote by $\mathcal{H}$ the set of such maps, and for a particular map, let us denote by $\otimes_h$ the operator that glues two observations, considering that $LiveF(e_i) = h(e_i)$. For a given path ρ let us denote by $LiveF_\rho()$ the map $LiveF()$ computed from M_ρ. We can associate with every sequence of MSCs $\rho = M_1 \ldots M_k$ the finite memory $Tocheck(M_\rho)$ and the composition operator $\otimes_h$ where $h = LiveF_\rho()$, which immediately defines the functions Ψ and Π. As $Tocheck(M_\rho)$ is finite, each concatenation adds only a bounded number of causal dependencies, hence $\mathcal{O}^V$ is finitely decomposable. The bounds for m and c come from the size of $Tocheck(M_\rho)$.

The proof easily extends to $\mathcal{O}^p$ as no assumption is done on the set of observable events. We will however prove later that one does not need the whole power of partial order automata to consider $\mathcal{O}^p$.

Now, for observation $\mathcal{O}^V_{\setminus C}$, we can build a similar proof. However, at concatenation time, one can only append events that are not causal consequences of confidential events to already built observations. Using the notion of black/white processes and events, this means that concatenation operators can only add observable events that are white on processes that are also white. We have shown that maintaining the status of processes along a sequence of MSCs is memoryless 3.6. We can hence relay on a set of operators of the form *OPS* $= \{\otimes_{h,BW}\}$ where h is the set of functions defined for $\mathcal{O}^V$ and BW is the black/white status of processes, and build Ψ and Π accordingly. □

A.5. Proof of Proposition 6.2

Proposition 6.2. For every HMSC H, one can build a (partial order) automaton $\mathcal{A}_{H,\mathcal{O}^p}$ that recognizes $\mathcal{O}^p(H)$. If $\mathcal{O} = \Sigma_p$, then one can build a (partial order) automata $\mathcal{A}_{H,\mathcal{O}^{V,\bullet}_{\setminus C}}$ and $\mathcal{A}_{H,\mathcal{O}^{V,\circ}}$ that recognize respectively $\mathcal{O}^{V,\bullet}_{\setminus C}(H)$ and $\mathcal{O}^{V,\circ}(H)$.

Proof Let $H = (N, \rightarrow, \mathcal{M}, n_0, F)$, and let us build a finite state automaton $\mathcal{A}_p(H)$ recognizing $\mathcal{O}^p(H)$ or equivalently $\mathcal{O}^{V,\circ}(H)$ when $\mathcal{O} = \Sigma_p$.

Let k be the maximal size of a projection of a MSC in $\mathcal{M}$. The automaton $\mathcal{A}_p(H)$ is defined by $\mathcal{A}_p = (N \times \{0,\ldots,k-1\}, \delta, (n_0,0), F \times \{0\}, \Lambda, \Pi)$. We let $\Lambda(n,O,n') = \circ$, i.e. we use standard sequential composition for every transition. We let $\Pi(O_1 \ldots O_k) = max(O_k)$ (Π remembers the last event in the last order appended. Let (n,M,n') be a transition in H. The observation $\mathcal{O}^p(M)$ is a possibly empty partial order over Σ_p. If $\mathcal{O}^p(M)$ is an empty order, then δ contains the transition $((n,0),\varepsilon,(n',0))$. If $\mathcal{O}^p(M) = a_1 \ldots a_q$ (with $q \leq k$), then δ contains the transitions $((n,i-1),a_i,(n,i))$ for each $i \in \{1,..,q-1\}$, and $((n,q-1),a_q,(n',0))$.

An easy induction shows that for every path $(n_0,M_1,n_1)\ldots(n_{\ell-1},M_\ell,n_\ell)$, such that the projection of each M_i on p is a word $w_i = a_{i,1}\ldots a_{i,q_i}$ there exists a path $(n_0,0) \xrightarrow{a_{1,1}} (n_0,1) \xrightarrow{a_{1,2}} \ldots \xrightarrow{a_{\ell,q_\ell}} (n_\ell,0)$, and conversely. Furthermore, if n_ℓ is an accepting state of H, then $(n_\ell,0)$ is an accepting state of $\mathcal{A}_p$. Hence, $\mathcal{A}_p$ recognizes $\mathcal{O}^p(H)$. The size of $\mathcal{A}_p(H)$ is in $O(|N|.k)$.

Let us now show that one can design an automaton that recognizes $\mathcal{O}^{V;\bullet}_{\setminus C}(H)$ for any HMSC H. Let us first recall the definition of $\mathcal{O}^{V;\bullet}_{\setminus C}(H)$. We have $\mathcal{O}^{V;\bullet}_{\setminus C}(H) = \{\mathcal{O}^V(M_1 \circ \cdots \circ M_k) \mid M_1 \circ \cdots \circ M_k \in \mathcal{F}_H \wedge \forall i, \alpha(E_i) \cap C = \emptyset\}$. Let us now design a new HMSC $H_{\setminus C} = (N, \rightarrow_{\setminus C}, \mathcal{M}, n_0, F)$ such that $(n,M,n') \in \rightarrow_{\setminus C}$ iff $(n,M,n') \in \rightarrow$ and $\alpha(E_M) \cap C = \emptyset$. Clearly, $\mathcal{F}_{H_{\setminus C}}$ is the set of MSCs generated by H that do not contain actions from C, and $H_{\setminus C}$ is also an HMSC. We have $\mathcal{O}^p(H_{\setminus C}) = \mathcal{O}^{V;\bullet}_{\setminus C}(H)$, and hence we can apply the technique above to design an automaton $\mathcal{A}'_p(H) = \mathcal{A}_p(H_{\setminus C})$ of size in $O(|N|.k)$ that recognizes $\mathcal{O}^{V;\bullet}_{\setminus C}(H)$. Hence $\mathcal{O}^p(H)$ and $\mathcal{O}^{V;\bullet}_{\setminus C}$ can be recognized by partial order automaton. □

A.6. Proof of Corollary 6.3

Corollary 6.3. The problem of deciding local interference of an HMSC H with respect to a given process $p \in \mathbb{P}$ is PSPACE-complete.

Proof From proposition 6.2, for any HMSC H and any process p, we can design an automaton $\mathcal{A}_p(H)$ that recognizes $\mathcal{O}^p((H)$, and an automaton $\mathcal{A}'_p(H)$ that recognizes $\mathcal{O}^{V;\bullet}_{\setminus C}(H)$. One can notice that $\mathcal{A}_p(H)$ and $\mathcal{A}'_p(H)$ are standard finite state automata that recognize sequences of events. Hence inclusion of $\mathcal{O}^p((H)$ in $\mathcal{O}^{V;\bullet}_{\setminus C}(H)$ with these automata is a regular language inclusion problem.

Note that these automata are of size linear in the size of H. One can notice that $\mathcal{L}(\mathcal{A}'_p(H)) \subseteq \mathcal{L}(\mathcal{A}_p(H))$. So, checking local non-interference of an HMSC H amounts to a single inclusion problem $\sqsubseteq_{\mathcal{O}^p,\mathcal{O}^{V;\bullet}_{\setminus C}}$ for HMSC H, i.e checking that $\mathcal{L}(\mathcal{A}_p(H)) \subseteq \mathcal{L}(\mathcal{A}'_p(H))$. Language inclusion for finite automata is a well-known PSPACE-complete problem, hence checking local non-interference is in PSPACE.

For the hardness part, let $\mathcal{A} = (Q_A, \delta_A, q_{0A}, F_A)$ and $\mathcal{B} = (Q_B, \delta_B, q_{0B}, F_B)$ be two automata over alphabet Σ, with disjoint set of states. Similarly to Figure 2, we design a HMSC $H = (Q_A \uplus Q_B, \rightarrow, q_{0B}, F_A \uplus F_B)$ over a set of processes $\{p_1, p_2, P_c\}$ and alphabet $\Sigma \cup \{c\}$, with $V = \Sigma$ and $C = \{c\}$, such that $\rightarrow$ contains:

— a transition (q_{0B}, M_h, q_{0A}) in which M_c is an MSC with a single atomic confidential action located on process P_c (like in Figure 2),
— for each $(q,a,q') \in \delta_A \cup \delta_B$, a transition (q, M_a, q') where M_a is a MSC containing a single message m_a from p_2 to p_1.

Then the language inclusion problem $\mathcal{L}(A) \subseteq \mathcal{L}(B)$ can be reduced in polynomial time to local non-interference of H with respect to process p_1. Hence, local non-interference is PSPACE-complete. □

A.7. Proof of Proposition 6.7

Proposition 6.7. Let H be an HMSC, $p \in \mathbb{P}$, and $\Sigma = C \uplus V \uplus N$. Then, one can build:

— an HMSC $H^{B,p}$ that recognizes MSCs from $\mathcal{F}_H$ after which p is a black process.
— an HMSC $H^{W,p}$ that recognizes MSCs from $\mathcal{F}_H$ after which p is a white process.

of size in $O(|H|.2^{|\mathbb{P}|})$.

Proof. We build $H^{B,p} = (N^{B,p}, \rightarrow^{B,p}, \mathcal{M}, n_0^{B,p}, F^{B,p})$ as follows:

— $N^{B,p} \subseteq N \times 2^{\mathbb{P}}$ is a set of nodes. In a pair (n, P), n denotes a node of H, and P a subset of black processes. We set $n_0^{BW} = (n_0, \emptyset)$.
— the set of transitions and nodes of $H^{B,p}$ is built inductively as follows: from a node (n, P), if there exists a transition (n, M, n') in H, we add (n', P') to $N^{B,p}$, with $P' = P \cup \{p \in \mathbb{P} \mid \exists e \leq_M f \wedge \phi(f) = p \wedge \phi(e) \in P\} \cup \{p \in \mathbb{P} \mid \exists e \leq f, \alpha(e) \in C \wedge \phi(f) = p\}$, and we add transition $\big((n, P), M, (n', P')\big)$ to $\rightarrow^{B,p}$
— $F^{B,p} = F \times \{P \in 2^{\mathbb{P}} \mid p \in P\}$ is the set of accepting nodes. A path of $H^{B,p}$ is accepting if it ends after recognizing an MSC $M \in \mathcal{F}_H$ such that p is black after M.

Building $H^{W,p} = (N^{W,p}, \rightarrow^{W,p}, \mathcal{M}, n_0^{W,p}, F^{W,p})$ can be done in a similar way, but setting $F^{W,p} = F \times \{P \in 2^{\mathbb{P}} \mid p \notin P\}$.

The status of a process is built progressively along transitions in a path. Following proposition 3.6, the process part of a node in $H^{B,p}$ or $H^{W,p}$ faithfully encodes the status of a process in the MSCs generated by sequences of transitions ending in this node. Hence, $H^{B,p}$ (resp. $H^{W,p}$) recognize MSCs of $\mathcal{F}_H$ after which p is black (resp. white).

As the nodes of these HMSCs belong to $N \times 2^{\mathbb{P}}$, the size of $H^{B,p}$ or $H^{W,p}$ is in $O(|H|.2^{|\mathbb{P}|})$. ☐

A.8. Proof of Proposition 7.6

Proposition 7.6. Let H be an HMSC, Σ an alphabet and D be a set of declassification letters. Then, one can build

— an HMSC H^{II} that generates the set of II MSCs in $\mathcal{F}_H$.
— an HMSC H^{INI} that generates the set of INI MSCs in $\mathcal{F}_H$.

that are of sizes at most $2.|H|.2^{3^{|\mathbb{P}|}}$

Proof. We first show how $H^{II} = (N^{II}, \rightarrow^{II}, \mathcal{M}, n_0^{II}, F^{II})$ is computed, then we show that H^{II} recognizes intransitively interferent MSC generated by H. We first define the following functions. A map $cl : \mathbb{P} \rightarrow \{+, \perp, \top\}$ represents existing causal dependencies from a confidential event to maximal event of processes, plus gives information on whether a causal chain ending on a process declassifies this confidential event. We denote by $CL(M)$ the set of functions that are computed starting from all confidential events. Note that if M contains no confidential event, then $CL(M) = \emptyset$. Given two MSCs M_1, M_2, we have seen that $M_1 \circ M_2$ is intransitively interferent if M_1 is II, or M_2 is II, or there exists $cl \in CL(M_1)$ such that $cl(p) = +$ and M_2 contains a chain from an event located on process to an observable event v such that there exists no chain from an event on process q to v with $cl(q) = \top$. Hence, knowing if M_1 is *II* or not, and $CL(M_1)$, one can decide whether $M_1 \circ M_2$ is *II*. We denote by *II*(CL, M) the predicate that is true when a set of maps CL representing causal chains and declassification in an MSC M' allows to prove that $M' \circ M$ contains an intransitive interference.

The crux is then to be able to maintain $CL(M_1 \circ \cdots \circ M_k)$ and the II information along path of H. For a given map cl and an MSC M, we define the map $Update(cl, M)$ as follows:

We have $Update(cl, M)(p) = \perp$ if $cl(p) = \perp$, and if there exists no pair of events $e \leq f$ in M_2 with f is located of p, and $cl(\phi(e)) \neq \perp$.

We have $Update(cl, M)(p) = +$ if $cl(p) \in \{\perp, +\}$, and there exists a process q such that $cl(q) = +$, and pair of events $e \leq f$ in M_2 such that e is minimal on q, f is maximal on process p, and furthermore, no causal chain from e to f is declassified, and for every process $q' \neq q$, if $cl(q') = +$, then no declassified causal chain from an event on q' to f exists in M_2, if $cl(q') = \top$ then no causal chain from an event on q' to f exists in M_2.

We have $Update(cl, M)(p) = \top$ if

— $cl(p) = \top$, or
— there exist a process q such that $cl(q) = +$ and a declassified chain from an event e located on process q to an event f located on process p, or
— there exist a process q such that $cl(q) = \top$, and a causal chain from an event e located on process q to an event f located on process p.

The map updating function extends to sets of maps the obvious way : $Update(X, M) = \bigcup_{cl \in X} Update(cl, M)$.

We are now ready to define $H^{II} = (N^{II}, \rightarrow^{II}, \mathcal{M}, n_0^{II}, F^{II})$. We have:

— $N^{II} \subseteq N \times \{tt, ff\} \times 2^{\mathcal{C}l}$ is a set of nodes that are reachable from n_0^{II}. Each node of N^{II} is hence a triple of the form (n, b, X), where n is a node of H, b is a boolean that indicates if *II* has already been discovered, and X is a set of maps depicting (declassified) causal chains from confidential events in the sequence $M_1 \circ \ldots M_k$ read so far along transitions of H and ending at node n. We set $n_0^{II} = (n_0, ff, \emptyset)$.
— We define the transitions relation as follows. We have $((n, b, X), M, (n', b', X')) \in \rightarrow^{II}$ iff
  — $(n, M, n') \in \rightarrow$ (the transition exists in H),
  — $b' = b \vee \bigvee_{cl \in X} II(cl, M) \wedge M$ is *II*, that is if *II* was detected before, then the concatenated MSCs remain *II*, and otherwise become *II* if M is *II*, or one of the maps depicting chains starting from a confidential events in the formerly assembled MSC $M_1 \circ \cdots \circ M_k$ witnesses an *II* in $M_1 \circ \cdots \circ M_k \circ M$.
  — $F^{II} = F \times \{tt\} \times 2^{\mathcal{C}l}$
  — $X' = Update(X, M) \cup CL(M)$. The representation of chains originating from confidential events is updated to consider chains of M and their declassifications, and new observable events may occur in M, starting new chains and potential new witnesses for *II* MSCs.

Obviously, all MSCs generated by H^{II} belong to $\mathcal{F}_H$, as $\rightarrow^{II}$ always agrees with $\rightarrow$. Furthermore, due to compositionality of cl computation, updating of a chain cl can be done incrementally while concatenating MSCs without remembering the whole sequence. Now, it suffices to remember once the shape of causal chains from observables actions to maximal events on processes (the maps cl) to detect II. One needs not differentiate similar occurrences of maps computed for chains originating from distinct observable events. Hence, updating of sets of causal chains representation suffices to represent all classified chains in a sequence of MSCs recognizes between n_0 and the current node, and hence to detect all occurrences of intransitive interferences. we can conclude that all MSCs recognized by H^{II} contain an intransitive interference.

The HMSC $H^{INI} = (N^{INI}, \rightarrow^{INI}, \mathcal{M}, n_0^{INI}, F^{INI})$ can be built with the same nodes and transition functions, but with final sates $F^{INI} = F \times \{ff\} \times 2^{\mathcal{C}l}$. The sizes of H^{II} and H^{INI} are at most $2.|H|.2^{3^{|\mathbb{P}|}}$.

$\square$