Available online at www.sciencedirect.com

ScienceDirect

Electronic Notes in Theoretical Computer Science

Electronic Notes in Theoretical Computer Science 180 (2007) 35–53

www.elsevier.com/locate/entcs

Non-Interference Control Synthesis for Security Timed Automata[1]

Guillaume Gardey* John Mullins** Olivier H. Roux*[2]

* *IRCCyN/CNRS*
BP 92101 1 rue de la Noë 44321 Nantes Cedex 3 France.

** *École Polytechnique de Montréal*
P.O. Box 6079, Station Centre-ville, Montreal (Quebec), Canada, H3C 3A7.

Abstract

In this paper, the problem of synthesizing controllers that ensures non interference for multilevel security dense timed discrete event systems modeled by an extension of Timed Automata, is addressed for the first time. We first discuss a notion of non interference for dense real-time systems that refines notions existing in the literature and investigate decidability issues raised by the verification problem for dense time properties. We then prove the decidability of the problem of synthesis of the timed controller for some of these timed non interference properties, providing so a symbolic method to synthesize a controller that ensures them.

Keywords: Non-interference, control synthesis, verification, decidability, Timed Automata.

1 Introduction

Non interference. Nowadays computing environments allow users to employ programs that are sent or fetched from different sites to achieve their goal, either in private or in an organization. Such programs may be run as a code to do simple calculation task or as interactive communicating programs doing IO operations or communications. Sometimes they deal with secret information such as a private data of a user or as classified data of an organization. Similar situations may occur in any computing environments where multiple users share common computing resources. One of the basic concerns in such context is to ensure programs not to leak sensitive data to a third party, either maliciously or inadvertently. This is one of the key aspects of the security concerns, that is often called *secrecy*. The *information*

[1] Work supported by an NSERC grant (Canadian Government) and by the ACI grant (French Government) Cortos (Control and Observation of Real-Time Open Systems)

[2] Currently at the École Polytechnique de Montréal

doi:10.1016/j.entcs.2005.05.046

flow analysis addresses this concern by clarifying conditions when a flow of information in a program is safe (i.e. high-level information never flows into low-level channels). These conditions named *non interference* properties, capture any causal dependency between high-level actions and low-level behavior. Their characterization has appeared rapidly out of the scope of the safety-liveness classification of system properties achieved by the system verification community during the last twenty five years. Also, in recent years, verification of information flow security has become an emergent field of research in computer science with a success story in its application to the analysis of cryptographic protocols where numerous uniform and concise characterizations of information flow security properties (*e.g.* confidentiality, authentication, non-repudiation or anonymity) in terms of non-interference have been proposed.

Timed non interference verification problem. The growing importance of verification for real-time systems, leads naturally to the next question of whether proof techniques developed in the untimed setting can be generalized for timed systems in order to be able to capture, besides the logical information flows, also the *time dependent* interference, *e.g.* timing covert channel [8]. Some untimed bisimulation-based non interference properties for information flow studied in [9] have been reformulated in [17] in a discrete time setting. In [5], some state-based and trace-based non interference properties have been introduced in a dense time setting using timed automata.

Timed non interference control problem. A natural generalization of security verification is *control of security* which is useful in the context of automated security system design. The problem here is not to verify that the system meets a given security policy, but to *control* the system in such a way that the security policy is met. In this framework, a system, often called a plant, is usually viewed as *open* and interacting with a "hostile" environment. The problem then is to synthesize a controller such that no matter how the environment behaves, the controlled plan satisfies the given security policy. The controller can control only a subset of actions of the plant, referred to as the controllable actions while the non-controllable actions represent the environment actions. In a real-time framework, the plant is modeled using discrete or continuous clocks. Recent researches have presented symbolic control synthesis algorithms for Timed Automata [14,2,7].

Organization of the paper and contributions. In this paper, we complete the dense time picture for non interference started in [5] by reformulating in this setting some untimed bisimulation-based non interference properties for information flow studied in [9] and by introducing a cosimulation-based notion of timed non interference (section 3). Also, we investigate decidability issues raised by the problem of their verification as real-time requirements of finite-state systems. We then focus (section 4) on the state-based and cosimulation-based timed non interference properties and prove the decidability of the problem of synthesis of their timed controller. The main result of the paper is a symbolic method to synthesize a

controller that ensures these properties. Finally, an example (section 5) is given to illustrate this method for the cosimulation-based timed non interference property. In the next section we give a short presentation of the notion of Timed Automata used in this paper.

2 Timed Automata

In this section we recall the definitions of Timed Automata (section 2.2) and some of their constructors. But first, some preliminaries about Timed Transition Systems and their behavior are given in section 2.1 in order to express semantics of Timed Automata.

2.1 Timed Transition Systems

Definition 2.1 [Timed Transition Systems] A *timed transition system (TTS)* over the set of actions Σ is a tuple $S = (Q, Q_0, \Sigma, \longrightarrow)$ where Q is a set of states, $Q_0 \subseteq Q$ is the set of initial states, Σ is a finite set of actions disjoint from $\mathbb{R}_{\geq 0}$, $\longrightarrow \subseteq Q \times (\Sigma \cup \mathbb{R}_{\geq 0}) \times Q$ is a set of edges. We also write $q \xrightarrow{e} q'$ for $(q, e, q') \in \longrightarrow$.

Definition 2.2 [Timed Similation] Let $S_1 = (Q_1, Q_0^1, \Sigma, \longrightarrow_1)$ and $S_2 = (Q_2, Q_0^2, \Sigma, \longrightarrow_2)$ be two TTS and $\sqsubseteq$ be a binary relation over $Q_1 \times Q_2$. We write $s \sqsubseteq s'$ for $(s, s') \in \sqsubseteq$. $\sqsubseteq$ is a *strong (timed) simulation relation* of S_1 by S_2 if: 1) if $s_1 \in Q_0^1$ there is some $s_2 \in Q_0^2$ s.t. $s_1 \sqsubseteq s_2$; 2) if $s_1 \xrightarrow{d}_1 s_1'$ with $d \in \mathbb{R}_{\geq 0}$ and $s_1 \sqsubseteq s_2$ then $s_2 \xrightarrow{d}_2 s_2'$ for some s_2', and $s_1' \sqsubseteq s_2'$; 3) if $s_1 \xrightarrow{a}_1 s_1'$ with $a \in \Sigma$ and $s_1 \sqsubseteq s_2$ then $s_2 \xrightarrow{a}_2 s_2'$ and $s_1' \sqsubseteq s_2'$.
A TTS S_2 *strongly simulates* S_1 if there is a strong (timed) simulation relation of S_1 by S_2. We write $S_1 \sqsubseteq_{\mathcal{S}} S_2$ in this case.

When there is a strong simulation relation $\sqsubseteq$ of S_1 by S_2 and also a strong simulation $\sqsubseteq'$ of S_2 by S_1, we have a *strong (timed) cosimulation relation*. When $\sqsubseteq' = \sqsubseteq^{-1}$, we have a *strong (timed) bisimulation relation* and we write $S_1 \approx_{\mathcal{S}} S_2$.

Let $S = (Q, Q_0, \Sigma^\varepsilon, \longrightarrow)$ be a TTS. We now use the ε-abstract TTS $S^\varepsilon = (Q, Q_0^\varepsilon, \Sigma, \longrightarrow_\varepsilon)$ by:

- $q \xrightarrow{d}_\varepsilon q'$ with $d \geq \mathbb{R}_{\geq 0}$ iff there is a run $\rho = q \xrightarrow{*} q'$ with $\mathit{Untimed}(\rho) = \varepsilon$ and $\mathit{Duration}(\rho) = d$,
- $q \xrightarrow{a}_\varepsilon q'$ with $a \in \Sigma$ iff there is a run $\rho = q \xrightarrow{*} q'$ with $\mathit{Untimed}(\rho) = a$ and $\mathit{Duration}(\rho) = 0$,
- $Q_0^\varepsilon = \{q \mid \exists q' \in Q_0 \mid q' \xrightarrow{*} q \text{ and } \mathit{Duration}(\rho) = 0 \wedge \mathit{Untimed}(\rho) = \varepsilon\}$.

Definition 2.3 [Weak Timed Simulation] Let $S_1 = (Q_1, Q_0^1, \Sigma_\varepsilon, \longrightarrow_1)$ and $S_2 = (Q_2, Q_0^2, \Sigma_\varepsilon, \longrightarrow_2)$ be two TTS and $\sqsubseteq$ be a binary relation over $Q_1 \times Q_2$. $\sqsubseteq$ is a *weak (timed) simulation relation* of S_1 by S_2 if it is a strong timed simulation relation of S_1^ε by S_2^ε. A TTS S_2 *weakly simulates* S_1 if there is a weak (timed) simulation relation of S_1 by S_2. We write $S_1 \sqsubseteq_{\mathcal{W}} S_2$ in this case.

When there is a weak simulation relation $\sqsubseteq$ of S_1 by S_2 and also a weak simulation $\sqsubseteq'$ of S_2 by S_1, we have a *weak (timed) cosimulation relation*. When $\sqsubseteq'=\sqsubseteq^{-1}$, we have a *weak (timed) bisimulation relation* and we write $S_1 \approx_{\mathcal{W}} S_2$.

Remark 2.4 Remark that if $S_1 \sqsubseteq_{\mathcal{S}} S_2$ then $S_1 \sqsubseteq_{\mathcal{W}} S_2$ and if $S_1 \sqsubseteq_{\mathcal{W}} S_2$ then $\mathcal{L}(S_1) \subseteq \mathcal{L}(S_2)$. In particular, weak timed bisimulation refines weak timed cosimulation that refines timed language (or trace) equivalence.

Finally, we introduced a definition of a TTS induced by a set of states which is informally the restriction of the TTS to a given set of states.

Definition 2.5 [TTS induced by a set of states] Let $\mathcal{S} = (Q, Q_0, \Sigma, \rightarrow)$ a TTS and $Y \subseteq Q$. The TTS *induced by* Y *on* $\mathcal{S}$ is the TTS $\mathcal{S}^Y = (Y, Q_0 \cap Y, \Sigma, \rightarrow_i)$ where $\rightarrow_i$ is defined by: $(q, \bullet, q') \in \rightarrow_i \Leftrightarrow q \in Y \wedge q' \in Y \wedge (q, \bullet, q') \in \rightarrow$

2.2 Timed Automata

Timed Automata (TA) were introduced by Alur & Dill [3] and have since been extensively studied. This model is an extension of finite automata with (dense time) *clocks* and enables one to specify real-time systems.

Definition 2.6 [Timed Automaton] A *Timed Automaton* $\mathcal{A}$ is a tuple $(L, l_0, X, \Sigma^\varepsilon, E, Inv)$ where: L is a finite set of *locations*; $l_0 \in L$ is the *initial location*; X is a finite set of positive real-valued *clocks*; $\Sigma^\varepsilon = \Sigma \cup \{\varepsilon\}$ is a finite set of *actions* and ε is the *silent* action; $E \subseteq L \times \mathcal{C}(X) \times \Sigma^\varepsilon \times 2^X \times L$ is a finite set of *edges*, $e = \langle l, \gamma, a, R, l' \rangle \in E$ represents an edge from the location l to the location l' with the guard γ, the label a and the reset set $R \subseteq X$; $Inv \in \mathcal{C}(X)^L$ assigns an *invariant* to any location. We restrict the invariants to conjuncts of terms of the form $x \preceq r$ for $x \in X$ and $r \in \mathbb{N}$ and $\preceq \in \{<, \leq\}$.

Definition 2.7 [Semantics of a Timed Automaton] The semantics of a Timed Automaton $\mathcal{A} = (L, l_0, X, \Sigma^\varepsilon, E, Inv)$ is a timed transition system $S^{\mathcal{A}} = (Q, q_0, \Sigma^\varepsilon, \rightarrow)$ with $Q = L \times (\mathbb{R}_{\geq 0})^X$, $q_0 = (l_0, \mathbf{0})$ is the initial state and $\rightarrow$ is defined by:

$$(l, v) \xrightarrow{a} (l', v') \quad \textit{iff} \quad \mathbf{Fired}(l, \gamma, a, R, l') \in E \textit{ s.t. } \begin{cases} \gamma(v) = \mathtt{tt}, \\ v' = v[R \mapsto 0] \\ Inv(l')(v') = \mathtt{tt} \end{cases}$$

$$(l, v) \xrightarrow{t} (l', v') \quad \textit{iff} \quad \begin{cases} l = l' \qquad v' = v + t \quad \textit{and} \\ \forall\, 0 \leq t' \leq t,\ Inv(l)(v + t') = \mathtt{tt} \end{cases}$$

We denote $Q^{\mathcal{A}}$, for the set of states of $\mathcal{A}$. A run ρ of $\mathcal{A}$ is an initial run of $S^{\mathcal{A}}$. The timed language accepted by $\mathcal{A}$ is $\mathcal{L}(\mathcal{A}) = \mathcal{L}(S^{\mathcal{A}})$.

State space abstraction

The analysis of a TA is based on the exploration of a finite graph, the *region graph*, where the nodes are *symbolic states* i.e. an equivalence classes of clock values. The state space is built by analyzing successors of the initial region (forward

analysis) [3]. Actually efficient forward (and backward) algorithms using regions do not code regions but zones, a finite convex union of regions because regions suffer of a combinatorial explosions and are quite uneasy to manipulate [14].

Constructors

Lets finally introduce some constructors over Timed Automata in order to express timed information flow in concurrent timed systems.

To describe a system as a parallel composition of Timed Automata, we use the classical composition notion based on a *synchronization function à la* Arnold-Nivat. Let $X = \{x_1, \cdots, x_n\}$ be a set of clocks, $\mathcal{A}_1$, ..., $\mathcal{A}_n$ be n Timed Automata with $\mathcal{A}_i = (N_i, l_{i,0}, X, \Sigma, E_i, Inv_i)$. A *synchronization function* f is a partial function from $(\Sigma \cup \{\bullet\})^n \hookrightarrow \Sigma$ where $\bullet$ is a special symbol used when an automaton is not involved in a step of the global system. We denote by $(\mathcal{A}_1 | \ldots | \mathcal{A}_n)_f$ the parallel composition of the $\mathcal{A}_i$'s w.r.t. f. The configurations of $(\mathcal{A}_1 | \ldots | \mathcal{A}_n)_f$ are pairs $(\mathbf{l}, \mathbf{v})$ with $\mathbf{l} = (l_1, \ldots, l_n) \in N_1 \times \ldots \times N_n$ and $\mathbf{v} = (v_1, \cdots, v_n)$ where each v_i is the value of the clock $x_i \in X$.

Definition 2.8 [Hiding Timed Automata] Let $\mathcal{A} = (L, l_0, X, \Sigma^\varepsilon, E, Inv)$ be a TA and $\Gamma \subseteq \Sigma$. We define the Γ-hiding TA $\mathcal{A}_{/\Gamma} = (L, l_0, X, (\Sigma \setminus \Gamma)^\varepsilon, E_{/\Gamma}, Inv)$ (with hided Γ-transitions) by $\langle l, \gamma, a, R, l' \rangle \in E_{/\Gamma}$ iff (1) $a \in (\Sigma \setminus \Gamma)^\varepsilon$ and $\langle l, \gamma, a, R, l' \rangle \in E$ or (2) $a = \epsilon$ and there is a transition $\langle l, \gamma, b, R, l' \rangle \in E$ with $b \in \Gamma$.

Transitions labeled with Γ are replaced by ϵ transitions, simulating so the hiding of information.

Definition 2.9 [Restriction Timed Automata] Let $\mathcal{A} = (L, l_0, X, \Sigma^\varepsilon, E, Inv)$ be a TA and $\Gamma \subseteq \Sigma$. We define the Γ-restriction TA $\mathcal{A}_{\setminus\Gamma} = (L, l_0, X, (\Sigma \setminus \Gamma)^\varepsilon, E_{\setminus\Gamma}, Inv)$ (without Γ-transitions) by $\langle l, \gamma, a, R, l' \rangle \in E_{\setminus\Gamma}$ iff $a \in (\Sigma \setminus \Gamma)^\varepsilon$ and $\langle l, \gamma, a, R, l' \rangle \in E$.

All transitions labeled with Γ are cut off the Timed Automaton.

3 Timed Non Interference for Security Timed Automata

In this section, we reformulate some non interference properties for information flow analysis in dense time discrete event systems previously studied in an untimed [9] or discrete time [17] setting. So we refine the notion of timed non interference for Timed Automata introduced in [5]. We start with *Strong Non-deterministic Non Interference* (SNNI). The basic idea of SNNI is that the public behavior of a system is not affected by the effects of its private behavior. We define this property on different behavior notions in section 3.3: trace equivalence, weak cosimulation, weak bisimulation and reachability equivalence. We then classify these properties among them and address decidability issues raised by their verification. A timed security model based on Timed Automata is presented in section 3.2.

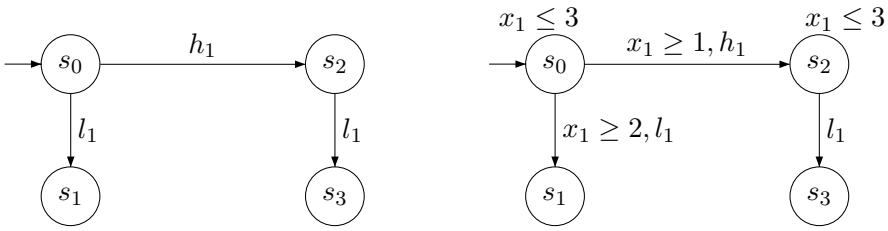

Fig. 1. Non-interfering automaton and interfering Timed Automaton

3.1 An introductory example

Let start by considering informally how information flow prohibition arises in interacting transition systems, taking a simple example. Consider first the transition system depicted on the left-hand side of the Figure 1. Suppose we attach secrecy levels to each action, for example $h_1 \in \Sigma_{priv}$ and $l_1 \in \Sigma_{pub}$. Intuitively this means that we wish interaction at h_1 to be secret, while interaction at l_1 may be known by a wider public: any private action may interact at h_1 and l_1, while a public action may interact only at l_1. None of the public observers *i.e.* observers allowed to observe (or react) only the public actions, has the possibility to know whether an interaction with h_1 happened or not. Otherwise stated, no causal dependency from private behavior may be inferred by any public observer. However adding timing constraints on this transition system could introduce prohibited information flow from the private level to the public one. As an illustration suppose that l_1 cannot occurs from s_0 before 2 time units as depicted by the Timed Automaton on the right-hand side of the Figure 1, then any public observer has the possibility to know whether an interaction with h_1 happened or not in the eventuality of l_1 occurring from s_2 before 2 time units *i.e.* there is a correlation between some private behavior and some public observation.

3.2 A Timed Security Model

We are looking at extended Timed Automata to model processes and computations of computing entities interacting at different trust levels.

Definition 3.1 [Security Timed Automaton] *A* Security Timed Automaton *is a Timed Automaton whose the set of visible actions* Σ *is partitioned in 2 sets* Σ_{pub}, Σ_{priv}.

The set Σ_{priv} represents the actions of a high level user to which secrecy can be attached. Σ_{low} is the set of actions of low level users.

3.3 Timed Information Flow Properties

The introductory example discussed in the section 3.1 motivates the following definitions.

3.3.1 Timed Strong Non-deterministic Non Interference

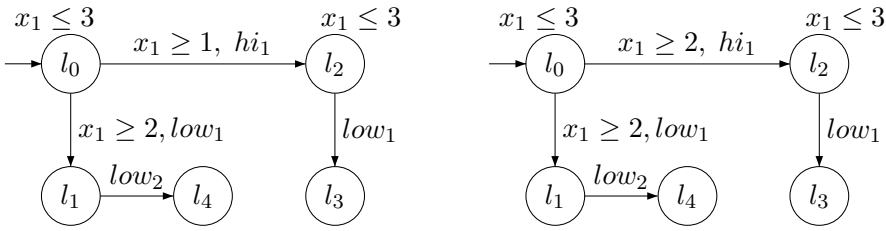

Fig. 2. Timed CSNNI : an interfering and a non-interfering TA

SNNI has been first proposed by Focardi [9] as a trace-based generalization of non interference for concurrent systems.

Definition 3.2 [Timed SNNI] *Let $\mathcal{A}$, a Security Timed Automaton. $\mathcal{A}$ satisfies* timed strong non-deterministic non interference *(timed SNNI) if and only if*

$$\mathcal{L}(\mathcal{A}_{/\Sigma_{priv}}) = \mathcal{L}(\mathcal{A}_{\setminus\Sigma_{priv}})$$

This definition stands that a timed system is timed SNNI *iff* a low level user can only observe timed words of $\mathcal{A}_{\setminus\Sigma_{priv}}$. It is then impossible for a low level user observing the timed words of the system to deduce information on the high level.

3.3.2 Timed Cosimulation-based SNNI

We now give a weak timed cosimulation non interference definition.

Definition 3.3 [Timed Cosimulation-based SNNI] *Let $\mathcal{A}$, a Security Timed Automaton. $\mathcal{A}$ satisfies* timed cosimulation non-deterministic non interference *(timed CSNNI) if and only if*

$$S^{\mathcal{A}_{/\Sigma_{priv}}} \sqsubseteq_{\mathcal{W}} S^{\mathcal{A}}$$

Consider the Timed Automaton depicted on the right hand side of the Figure 2. It is easy to see that it is timed CSNNI. Indeed, the capability of a public observer to interact at low_1 is not correlated to the occurrence of hi_1 while such a correlation exists for the Timed Automaton depicted on the left hand side. Any public interaction at low_1 for $1 \leq x_1 < 2$ is correlated to the occurrence of hi_1.

The next result presents a characterization of timed cosimulation non-deterministic non interference (timed CSNNI) in terms of cosimulation and will often be used as an alternative to Def. 3.3 in the sequel:

Theorem 3.4 *A Security Timed Automaton $\mathcal{A}$ satisfies timed cosimulation non-deterministic non interference (timed CSNNI) iff*

$$S^{\mathcal{A}_{/\Sigma_{priv}}} \sqsubseteq_{\mathcal{W}} S^{\mathcal{A}_{\setminus\Sigma_{priv}}} \tag{1}$$

$$S^{\mathcal{A}_{\setminus\Sigma_{priv}}} \sqsubseteq'_{\mathcal{W}} S^{\mathcal{A}_{/\Sigma_{priv}}} \tag{2}$$

Proof. Equation 1 is obtained from definition 3.3: as $\mathcal{A}_{/\Sigma_{priv}}$ is defined over Σ_{pub} we have $S^{\mathcal{A}_{/\Sigma_{priv}}} \sqsubseteq_{\mathcal{W}} S^{\mathcal{A}} \Leftrightarrow S^{\mathcal{A}_{/\Sigma_{priv}}} \sqsubseteq_{\mathcal{W}} S^{\mathcal{A}_{\setminus\Sigma_{priv}}}$. Equation 2 is obtained directly

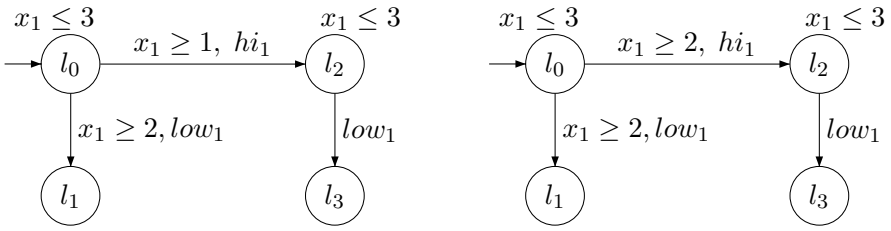

Fig. 3. Timed BSNNI : an interfering and a non-interfering TA

from definitions of Timed Automata, hiding and restriction: for any TA over $\Sigma = \Sigma_{priv} \cup \Sigma_{pub}$, we have $S^{\mathcal{A}\setminus\Sigma_{priv}} \sqsubseteq'_{\mathcal{W}} S^{\mathcal{A}/\Sigma_{priv}}$. Furthermore $\sqsubseteq'_{\mathcal{W}}$ is not necessarily equal to $\sqsubseteq^{-1}_{\mathcal{W}}$. □

3.3.3 Timed Bisimulation-based SNNI

We now give a timed version of the bisimulation-based definition of strong non-deterministic non interference proposed in [9]. Actually, any bisimulation-based information flow property presented in [9] could be recast in a similar manner. Fig. 2 and Fig. 3 illustrates differences between CSNNI and BSNNI.

Definition 3.5 [Timed Bisimulation-based SNNI] *Let $\mathcal{A}$, a Security Timed Automaton. $\mathcal{A}$ satisfies* timed bisimulation-based strong non-deterministic non interference *(timed BSNNI) if and only if*

$$S^{\mathcal{A}/\Sigma_{priv}} \approx_{\mathcal{W}} S^{\mathcal{A}\setminus\Sigma_{priv}}$$

The figure 3 represents an interfering and a non-interfering Timed Automata. The one at the left hand side is interfering since any low level interaction at low_1 for $1 \leq x_1 < 2$ is correlated to the occurrence of hi_1.

3.3.4 Timed State NNI

In [5], authors propose a decidable notion of timed non deterministic non-interference based on states. We reformulate it as follows :

Definition 3.6 [Timed State NNI] Let $\mathcal{A}$ be a Security Timed Automaton over Σ ($\Sigma = \Sigma_{pub} \cup \Sigma_{priv}$). $\mathcal{A}$ is said to be Timed State Non Deterministic Non Interfering (timed StNNI) if and only if:

$$Q^{\mathcal{A}/\Sigma_{priv}} = Q^{\mathcal{A}\setminus\Sigma_{priv}} \text{ i.e. } Q^{\mathcal{A}} = Q^{\mathcal{A}\setminus\Sigma_{priv}}$$

Let us note that as $Q^{\mathcal{A}\setminus\Sigma_{priv}} \subseteq Q^{\mathcal{A}}$, then timed StNNI is equivalent to $Q^{\mathcal{A}/\Sigma_{priv}} \subseteq Q^{\mathcal{A}\setminus\Sigma_{priv}}$ i.e. $Q^{\mathcal{A}} \subseteq Q^{\mathcal{A}\setminus\Sigma_{priv}}$.

3.4 Ordering relation among properties

Theorem 3.7 (Timed non interference classification)

$$\textit{timed-BSNNI} \Rightarrow \textit{timed-CSNNI} \Rightarrow \textit{timed-SNNI} \quad (3)$$

$$\textit{timed-SNNI} \not\Rightarrow \textit{timed-CSNNI} \not\Rightarrow \textit{timed-BSNNI} \quad (4)$$

Proof. Equation 3 follows directly Remark 2.4.

Concerning the strict implications 4, we present here a counterexample for the last implication. Let us consider the Timed Automaton depicted on the right hand side of the Figure 2. It is timed-CSSNI, however is not timed BSNNI since a public observer after interacting with low_1 has now the capability to detect the deadlock correlated to the occurrence of hi_1.

□

Remark 3.8 There is no ordering between Timed StNNI and any of the previous non interference definitions presented here.

Remark 3.9 It can be proved that 0-trace-non-interference introduced in [5] is equivalent to Timed StNNI.

3.5 Decidability results

Theorem 3.10 (Timed SNNI decidability) *Timed SNNI is undecidable.*

Proof. We prove that the universality problem for TA can be reduced to a timed SNNI problem. Let $\mathcal{A}$ be a TA over Σ. The Security Timed Automaton $\mathcal{A}_f$ is constructed from $\mathcal{A}$ by adding a new edge labeled with $h \notin \Sigma$ from the initial locality of $\mathcal{A}$ to a new locality ℓ_u and for each action $a \in \Sigma$, a loop over ℓ_u with label a. We set $\Sigma_{pub} = \Sigma$ and $\Sigma_{priv} = \{h\}$. Clearly, $\mathcal{A}$ accepts universal timed language iff $\mathcal{A}_f$ is timed trace non-interfering. As universality problem is known to be undecidable for TA, it follows that timed SNNI problem is also undecidable. □

Theorem 3.11 (Timed BSNNI, CSNNI and StNNI decidability) *timed BSNNI, timed CSNNI and timed StNNI are decidable.*

Proof. Simulation [1] and bisimulation [12] are decidable for TA and then timed BSNNI and timed CSNNI are decidable. As reachability [3] is decidable for TA, timed StNNI is also decidable. □

4 Timed Non-Interference Controllability

4.1 Introduction

Recent researches have presented symbolic control synthesis algorithms for Timed Automata [20,14,2]. The aim of such algorithms is to design an agent (also named controller or supervisor) that restricts the behavior of the initial system so that a property of interest is modeled by the controlled system also called closed-loop system.

In such frameworks, the set of actions of the model is classically partitioned into controllable and uncontrollable actions. Controllable actions, corresponding to system actions, can be disabled, delayed or forced by the controller whereas uncontrollable actions, denoting actions from environment, cannot be restricted. In order to address symbolic control synthesis for Security Timed Automata, the set of actions Σ is here, more or less arbitrarily, partitioned in two sets Σ_{pub}, Σ_{priv}. Σ_{pub} is the set of uncontrollable actions and Σ_{priv} is the set of controllable actions. Actually, the choice for partitioning will depends on what is considered as intruder's actions.

We formalize the controller of a Security Timed Automaton in the following definition:

Definition 4.1 [Controller] Let $\mathcal{A} = (L, l_0, X, \Sigma, E, Inv)$ be a Security Timed Automaton over Σ ($\Sigma = \Sigma_{pub} \cup \Sigma_{priv}$). A controller $\mathcal{C} = (L^C, l_0^C, X, \Sigma, E^C, Inv^C)$ for $\mathcal{A}$ is a Timed Automaton over Σ such that $\mathcal{A}_C = (\mathcal{A}|\mathcal{C})_{f_c}$ where $\mathcal{A}_C = (L_C, (l_0, l_0^C), X, \Sigma, E_C, Inv_C)$ with $L_C \in L \times L^C$ and f_c is a control synchronization function defined by $f_c(a, a) = a$.

The controller must not restrict uncontrollable behavior in the following sense: Let $(l_\mathcal{A}, l_\mathcal{C})$ a locality of $\mathcal{A}_\mathcal{C}$,

(i) $\forall \langle l_\mathcal{A}, \gamma, a, R, l'_\mathcal{A}\rangle \in E$ s.t. $a \in \Sigma_{pub}$, there is an edge $\langle (l_\mathcal{A}, l_\mathcal{C}), \gamma, a, R, (l'_\mathcal{A}, l'_\mathcal{C})\rangle \in E_C$,

(ii) if $\nexists \langle (l_\mathcal{A}, l_\mathcal{C}), \gamma, a, R, (l'_\mathcal{A}, l'_\mathcal{C})\rangle \in E_C$ with $a \in \Sigma_{priv}$ then $Inv_C(l_\mathcal{A}, l_\mathcal{C}) = Inv(l_\mathcal{A})$

The first condition claims that the controller cannot prevent the firing of uncontrollable edges. The rationale behind the second condition is the following:

One cannot impose the firing of an uncontrollable transition but on the other hand if from a state, it is possible to fire both an uncontrollable transition and a controllable transition, then one can force the firing of the controllable transition by a temporal constraint as long as the uncontrollable transition is not fired.

In the timed framework, the cornerstone to solutions of control problems is the *controllable predecessors operator* denoted $\pi(X)$ [14,2,19].

Definition 4.2 [Controllable predecessors] Let $X \subseteq Q$ be a set of states of a Security Timed Automaton S. The set of controllable predecessors of X is defined by:

$$\begin{aligned}\pi(X) = \{q \in Q \mid ((\exists \delta \in \mathbb{R}_{\geq 0}\ \exists a \in \Sigma\ \exists q' \in X,\ q \xrightarrow{\delta}\xrightarrow{a} q') \\ \vee\ (\exists \delta \in \mathbb{R}_{\geq 0}\ \exists q' \in X,\ q \xrightarrow{\delta} q')) \\ \wedge\ \forall \delta_u \in \mathbb{R}_{\geq 0} \text{ if } \exists a_u \in \Sigma_{low}, q'_u \notin X, q \xrightarrow{\delta_u}\xrightarrow{a_u} q'_u \\ \text{then } \exists \delta_c < \delta_u\ a_c \in \Sigma_{priv}, \exists q'_c \in X\ q \xrightarrow{\delta_c}\xrightarrow{a_c} q'_c\}\end{aligned}$$

Informally, q is a controllable predecessor of X iff:

- a state q' of X is reachable by time elapsing and firing of a transition,

- for any uncontrollable transition diverging from X, the controller can take a decision at an earlier date to constraint the system in X.

4.2 Timed Non-interference control problems

In this section we define and propose solutions for some timed non-interference control problem. We prove the decidability of control problems of *timed-StNNI* and *timed-CSNNI* by giving algorithms for the synthesis of controller guaranteeing these two properties. As well as the safety control problem for Timed Automata, the TTS solution given by the algorithm can be both interpreted as the system in closed-loop and the controller. For all these problems, the controller that forbids any controllable actions is a solution but not necessarily the most permissive.

4.2.1 Timed-StNNI Control Problem

First, we consider the *Timed State Non-Interference Control Problem.*

Definition 4.3 [Timed-StNNI Control Problem] Let $\mathcal{A}$ be a Security Timed Automaton over Σ ($\Sigma = \Sigma_{pub} \cup \Sigma_{priv}$). A StNNI controller $\mathcal{C}$ for $\mathcal{A}$ (according to def. 4.1) is a Timed Automaton over Σ such that :

$$Q^{((\mathcal{A}|\mathcal{C})_{fc})_{/\Sigma_{priv}}} = Q^{((\mathcal{A}|\mathcal{C})_{fc})\setminus\Sigma_{priv}}$$

$$\text{i.e. } Q^{(\mathcal{A}|\mathcal{C})_{fc}} = Q^{((\mathcal{A}|\mathcal{C})_{fc})\setminus\Sigma_{priv}}$$

Let us consider the solution of the safety control problem on $\mathcal{A}$ that is to find a controller C such that $Q^{(A|C)_{fc}} \subseteq Q^{\mathcal{A}\setminus\Sigma_{priv}}$.

Proposition 4.4 *If C is the solution of the control problem $Q^{(A|C)_{fc}} \subseteq Q^{\mathcal{A}\setminus\Sigma_{priv}}$, C is a solution of timed-StNNI if and only if:*

$$\forall q \in Q^{(A|C)_{fc}} \exists \rho = \tau_1 \cdot low_1 \ldots \tau_n \cdot low_n \in \mathcal{S}^{(A|C)_{fc}}, low_i \in \Sigma_{pub} \text{ such that } q_0 \xrightarrow{\rho} q$$

Proof. Let C the solution of the safety control problem $Q^{(A|C)_{fc}} \subseteq Q^{\mathcal{A}\setminus\Sigma_{priv}}$.

If C is the solution of the timed-StNNI problem then $Q^{(A|C)_{fc}} \subseteq Q^{((A|C)_{fc})\setminus\Sigma_{priv}}$. Since the property holds for all states of $Q^{((A|C)_{fc})\setminus\Sigma_{priv}}$, it also holds for any state of $Q^{(A|C)_{fc}}$.

The converse is straightforward. □

This characterization gives rise to a decision procedure for the timed-StNNI control problem:

Theorem 4.5 (Timed-StNNI Control Problem Decidability) *The timed-StNNI control problem is decidable and there exists a state-based controller $\mathcal{C}$ such that $(\mathcal{A}|\mathcal{C})_{f_c}$ is timed-StNNI.*

Proof. The main idea is to use a classical controllable predecessors algorithm extended with a step that ensures that all states computed are reachable from the initial states by a sequence of low level actions (Σ_{pub}), that is states of $((\mathcal{A}|\mathcal{C})_{f_c})_{\setminus\Sigma_{priv}}$. Since the controller can restrict the behaviors of the system, this step ensures that computed states belong to the set of states of $((\mathcal{A}|\mathcal{C})_{f_c})_{\setminus\Sigma_{priv}}$.

In order to detail, let us consider the region graph of $\mathcal{A}$, $\mathcal{R}^*$ [3].

$\mathcal{R}^*$ has a finite number of regions.The iterative process given by $X_0 = Q^{\mathcal{A}_{\setminus\Sigma_{priv}}}$ and $X_{i+1} = X_i \cap \pi(X_i) \cap \overrightarrow{Post_{\Sigma_{pub}}}^*(\pi(X_i))$ will converge after finitely many steps for TA. Let the greatest fixed point obtained be denoted by X^*.

The TTS induced by X^* on $\mathcal{A}$ is timed-StNNI.

It is straightforward that $\forall i X_i \subseteq Q^{\mathcal{A}_{\setminus\Sigma_{priv}}}$ and that the TTS verifies the condition of the proposition 4.4

□

4.2.2 Timed-CSNNI Control Problem

In this section, we will prove that timed-CSNNI is decidable and a controller is computable. Intuitively, the computation method we propose consists first in computing the (controllable as well as uncontrollable) non interfering behavior of the STA $\mathcal{A}$ that is to say the intersection of behavior of $\mathcal{A}$ with the behavior of $\mathcal{A}_{\setminus\Sigma_{priv}}$ given by the product $(\mathcal{A}|\mathcal{A}_{\setminus\Sigma_{priv}})_f$ synchronized on actions of Σ_{pub} and free on actions of Σ_{priv}. The controllable behavior guaranteeing the non-interference property is then extracted from this result.

Definition 4.6 [Timed-CSNNI Control Problem] Let $\mathcal{A}$ be a Security Timed Automaton over $\Sigma = \Sigma_{pub} \cup \Sigma_{priv}$. A CSSNI controller $\mathcal{C}$ for $\mathcal{A}$ (according to def. 4.1) is a timed automaton over Σ such that:

$$\mathcal{S}^{(\mathcal{A}|\mathcal{C})_{/\Sigma_{priv}}} \sqsubseteq_{\mathcal{W}} \mathcal{S}^{(\mathcal{A}|\mathcal{C})_{\setminus\Sigma_{priv}}}$$

Let us now define the simulation relation we will use to prove that the timed-CSNNI problem is decidable :

Definition 4.7 [Simulation relation] Let $q = (l_1, l_2, v) \in Q^{(\mathcal{A}|\mathcal{C})_{/\Sigma_{priv}}}$ and $q' = (l'_1, l'_2, v') \in Q^{(\mathcal{A}|\mathcal{C})_{\setminus\Sigma_{priv}}}$: $q \sqsubseteq q' \Leftrightarrow l_2 = l'_2 \wedge v = v'$

Since $\mathcal{S}^{(\mathcal{A}|\mathcal{C})_{\setminus\Sigma_{priv}}} \sqsubseteq \mathcal{S}^{\mathcal{A}_{\setminus\Sigma_{priv}}}$ (the controller restrict the behavior of $\mathcal{A}$), we will also note $q \sqsubseteq q'$ for any $q = (l_1, l_2, v) \in Q^{(\mathcal{A}|\mathcal{C})_{\setminus\Sigma_{priv}}}$ and $q' = (l'_2, v') \in Q^{\mathcal{A}_{\setminus\Sigma_{priv}}}$ if and only if $l_2 = l'_2$ and $v = v'$.

Definition 4.8 [Unfolding of $\mathcal{A}$ over $(\mathcal{A}|\mathcal{A}_{\setminus\Sigma_{priv}})_f$] Let $\mathcal{A} = (L, \ell_0, X, \Sigma, E, Inv)$ be a TA over $\Sigma = \Sigma_{pub} \cup \Sigma_{priv}$. Let $\mathcal{A}' = (L', (l'_0, l'_0), X, \Sigma, E', Inv'))$ with $L' \in L \times L$ be a TA defined by $\mathcal{A}' = (\mathcal{A}|\mathcal{A}_{\setminus\Sigma_{priv}})_f$ where $f(a, \bullet) = a$, if $a \in \Sigma_{priv}$ and $f(a, a) = a$ $a \in \Sigma_{pub}$.

The TA $\mathcal{A}'' = (L'', (l_0, l_0), X, \Sigma, E'', Inv'')$ with $L'' \in (L \times L) \cup \{bad\}$ is the unfolding of $\mathcal{A}$ over $\mathcal{A}'$ iff :

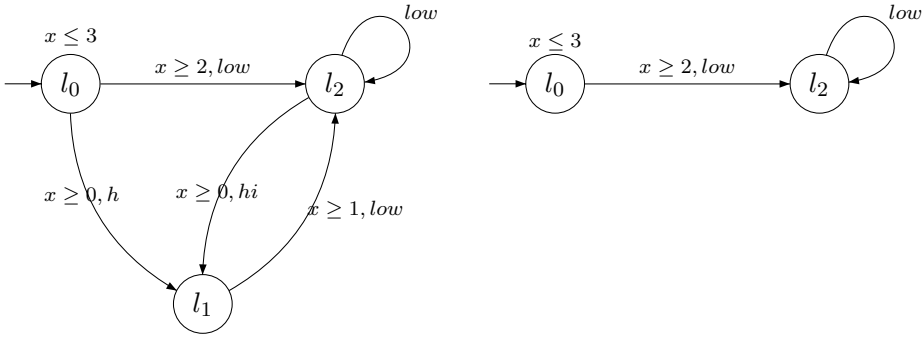

Fig. 4. $\mathcal{A}$ and $\mathcal{A}_{\setminus \Sigma_{priv}}$

(i) $L'' = L' \cup \{bad\}$

(ii) $\forall e' = \langle (l_i, l_j), \gamma', a, R', (l'_i, l'_j) \rangle \in E'$ such that $e = \langle l_i, \gamma, a, R, l'_i \rangle \in E$

$$\text{we have } \langle (l_i, l_j), \gamma, a, R, (l'_i, l'_j) \rangle \in E'',$$

(iii) $\forall e = \langle l, \gamma, a, R, l' \rangle \in E$ such that $a \in \Sigma_{pub}$, $\forall (l, l_i) \in L'$ and $(l', l_j) \in L'$

$$\text{we have } \langle (l, l_i), \gamma, a, R, (l', l_j) \rangle \in E'',$$

(iv) $\forall e = \langle l, \gamma, a, R, l' \rangle \in E$ such that $a \in \Sigma_{pub}$, $\forall (l, l_i) \in L'$ and $(l', l_j) \notin L'$

$$\text{we have } \langle (l, l_i), \gamma, a, R, (bad) \rangle \in E'',$$

(v) $\forall l = (l_i, l_j) \in L'', Inv(l) = Inv(l_i)$

The second and the fifth requirements relax the constraints inherited from $\mathcal{A}_{\setminus \Sigma_{priv}}$. The third and the fourth requirements add edges over public actions removed by $(\mathcal{A}|\mathcal{A}_{\setminus \Sigma_{priv}})_f$. We use the location *bad* as destination of edges leading to location not in L'.

Remark 4.9 $\mathcal{A}'$ is easily seen to be Timed-CSNNI (see figure 5). By synchronizing $\mathcal{A}$ with $\mathcal{A}_{\setminus \Sigma_{priv}}$, we isolate all non-interfering behaviors of A. We will then use $\mathcal{A}'$ to compute a controller for $\mathcal{A}$.

Remark 4.10 Unfolding of $\mathcal{A}$ over $\mathcal{A}'$ has the same discrete structure as $\mathcal{A}'$ but extended with the behavior of $\mathcal{A}$. This Timed Automaton will be then used to isolate states that are timed-CSNNI and compute the controller ensuring the timed-CSNNI property.

Definition 4.11 [States of $\mathcal{A}''$ similar to states of $\mathcal{A}_{\setminus \Sigma_{priv}}$] Let us consider the Timed Automaton $\mathcal{A}''$ and the set of states of $\mathcal{A}'$: $Q^{\mathcal{A}'} \subseteq Q^{\mathcal{A}''}$. We note $SiSt(\mathcal{A}')$ the set of states defined by:

$$SiSt(\mathcal{A}') = \{q \in Q^{\mathcal{A}'} \mid \text{ if } \exists low \in \Sigma_{pub} \; st. \; q \xrightarrow{low} \in E'' \text{ then } \exists q' \in Q^{\mathcal{A}_{\setminus \Sigma_{priv}}} \; st. \; q \sqsubseteq q' \wedge q' \xrightarrow{low} \in E^{\mathcal{A}_{\setminus \Sigma_{priv}}}\}$$

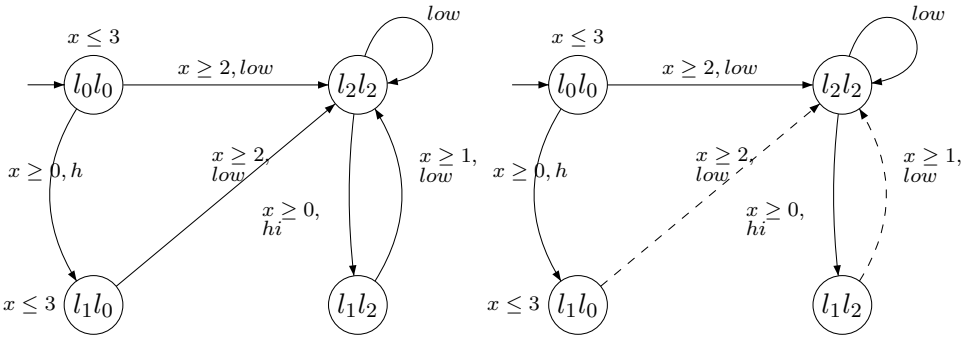

Fig. 5. $\mathcal{A}' = (\mathcal{A}|\mathcal{A}_{\setminus \Sigma_{priv}})_f$ and A''

$SiSt(\mathcal{A}')$ contains all states that are weakly similar to a state of $\mathcal{A}_{\setminus \Sigma_{priv}}$. That is, for every state $q = (l_1, l_2, v)$ such that the transition low is possible in the TTS defined by $\mathcal{A}''$, there exists also such a transition $(l_2, v) \xrightarrow{low}$ in the TTS defined by $\mathcal{A}_{\setminus \Sigma_{priv}}$. Considering the example of the figure 5, $(l_1, l_0, x = 1) \notin SiSt(\mathcal{A}')$ since $(l_0, x = 1) \not\xrightarrow{low}$ for $\mathcal{A}_{\setminus \Sigma_{priv}}$.

The sketch of the method to find a controller is then the following:

(i) Compute $\mathcal{A}'$ (This is timed-CSSNI) and the unfolding $\mathcal{A}''$ of $\mathcal{A}$.

(ii) Compute the largest controllable subset of $SiSt(\mathcal{A}'))$ preserving the timed-CSNNI property, that is states of $\mathcal{S}^{(\mathcal{A}|\mathcal{C})_{/\Sigma_{priv}}}$ weakly timed similar to a state of $\mathcal{S}^{(\mathcal{A}|\mathcal{C})_{\setminus \Sigma_{priv}}}$. Informally, at each iteration, a set of controllable safe states is computed and then refined to match the timed-CSNNI property.

More in detail, let us consider the algorithm 1 where $R_{\setminus \Sigma_{priv}}(X) = \{q \in X \mid \exists q_0 \in Q_0, \exists (\tau_i) \in (\mathbb{R}_{\geq 0})^n,\ \exists (low_i) \in (\Sigma_{pub})^n,\ q_0 \xrightarrow{\tau_1} q_1 \xrightarrow{low_1} q'_1 \cdots \xrightarrow{low_n} q'_n = q \wedge \forall i,\ q_i, q'_i \in X\}$ is the set of states that are reachable only by elapsing of time or low level actions (Σ_{pub}) such that all intermediary states remain in X, $Pred_{\Sigma^*_{priv}}(X) = \{q \in Q \mid \exists q' \in X, \exists \rho \in \Sigma^*_{priv}\ q \xrightarrow{\rho} q'\}$ is the set of predecessors of X by a sequence of discrete controllable transitions (Σ_{priv}), $Pred_{low}(X) = \{q \in Q \mid \exists q' \in X\ q \xrightarrow{low} q', low \in \Sigma_{pub}\}$, is the set of discrete predecessors of X by an uncontrollable transition, $Pred^Y_{\Sigma_{priv}}(X) = \{q \in Y \mid q \xrightarrow{hi_1} q_1 \xrightarrow{hi_2} q_2 \ldots \xrightarrow{h_n} q_n \wedge q, q_1, \ldots q_{n-1} \in Y, q_n \in Y\ \wedge \forall i,\ hi_i \in \Sigma_{priv}\}$ is the set of predecessors of X by a sequence of discrete controllable transitions such that all intermediary states remain in Y, $\pi(X)$ is the classical controllable predecessor operator and $Sim(X, Y) = \{q \in X \mid \exists q' \in Y q \sqsubseteq q'\}$ is the set of states X similar to Y.

Algorithm 1 (*Alg*) *We note Alg the following fix-point algorithm:*

1: $X_0 \leftarrow SiSt(\mathcal{A}')$
2: ***repeat***
3: $\quad X_{i+1} \leftarrow \pi(X_i)$

4: $X_{i+1} \leftarrow Sim(X_{i+1}, R_{\setminus \Sigma_{priv}}(X_{i+1}))$
5: ***for all*** $l \in \Sigma_{pub}$ ***do***
6: $Y \leftarrow Pred_{\Sigma^*_{priv}}(Pred_{low}(X_{i+1}))$
7: $Z \leftarrow Pred^{X_{i+1}}_{\Sigma^*_{priv}}(Pred^{X_{i+1}}_{low}(X_{i+1}))$
8: $X' \leftarrow X_{i+1} \setminus (Z \setminus Y)$
9: ***until*** $X' = X_i$

Proposition 4.12 *If Alg converges, we note X^* the solution. The TTS induced by X^* on $\mathcal{S}_{\mathcal{A}''}$ is CSNNI.*

Proof. We have to prove that $(\mathcal{S}^{X^*}_{A''})_{/\Sigma_{priv}} \sqsubseteq_{\mathcal{W}} (\mathcal{S}^{X^*}_{A''})_{\setminus \Sigma_{priv}}$.

Let $q \in Q^{\mathcal{S}^{X^*}_{A''}}$. According to line 4 of the algorithm, $\exists q' \in Q^{\mathcal{S}^{X^*}_{A''} \setminus \Sigma_{priv}}$ such that $q \sqsubseteq q'$. It there exists a firing sequence $q \xrightarrow{hi_1 \ldots hi_n} s \xrightarrow{low_1} r$ then $q \sqsubseteq s$ (and so $q' \sqsubseteq s$) and $\exists r' \in Q^{\mathcal{S}^{X^*}_{A''} \setminus \Sigma_{priv}}$ such that $r \sqsubseteq r'$ (line 4). We also have with lines 5–8, $s, r \in X^*$. Since $s \xrightarrow{low_1} r$ and $q' \sqsubseteq s$, $q' \xrightarrow{low_1} r'$. So $q' \xrightarrow{low_1} r'$ and $r \sqsubseteq r'$. Consequently, q and q' fulfill the weak timed simulation (by replacing all hi_i by ϵ).

The proof would be similar for continuous transitions. □

Proposition 4.13 *The algorithm Alg terminates.*

Proof. The convergence of the algorithm is ensured by the *region graph* $\mathcal{R}^*$ [3]: the family sequence (X_i) is monotone ($X_{i+1} \subseteq X_i$) over a finite domain $\mathcal{R}^*$. □

As a corollary of propositions 4.12 and 4.13, we get the following result:

Theorem 4.14 (Timed-CSNNI Control Problem Decidability) *Timed-CSNNI is decidable and a controller is computable.*

Indeed, since the synchronization function (f_c) is total, the TTS solution given by the algorithm can be both interpreted as the system in closed-loop and the controller for the system (as for safety control problem on TA). The complete method is illustrated on a simple example in the next section.

5 CSNNI Control Problem - A simple example

Let us consider the Security Timed Automaton $\mathcal{A}$ of figure 6 and the CSNNI control problem on $\mathcal{A}$.

Following the method proposed in section 4.2.2, we compute $\mathcal{A}' = (\mathcal{A}|\mathcal{C})_{f_c}$. $\mathcal{A}'$ is represented in figure 7. We are then able to compute $Q^{\mathcal{A}'}$ (table 1).

Given $Q^{\mathcal{A}'}$ and $\mathcal{A}''$ we can then compute $SiSt(\mathcal{A}')$(table 2). For instance, at state $(l_2, l_0, x_1 = 1)$ the firing of low_1 is possible for $\mathcal{A}''$ while it is not possible for the state $(l_0, x_1 = 1)$ in $\mathcal{A}'$. So this state must be discarded. The aim of the controller will be to prevent this state to be reachable.

The solution of the algorithm 1 is the set of states of table 3 which can be transformed into the Timed Automaton of the figure 9.

Localities	**Clock Space**
$l_0 l_0$	$x_1 \leq 3$
$l_1 l_1$	$x_1 \geq 2$
$l_4 l_4$	$x_1 \geq 2$
$l_2 l_0$	$1 \leq x_1 \leq 3$
$l_3 l_1$	$2 \leq x_1$
$l_5 l_1$	$1 \leq x_1$

Table 1
$Q^{\mathcal{A}'}$

Localities	**Clock Space**
$l_0 l_0$	$x_1 \leq 3$
$l_1 l_1$	$x_1 \geq 2$
$l_4 l_4$	$x_1 \geq 2$
$l_2 l_0$	$2 \leq x_1 \leq 3$
$l_3 l_1$	$1 \leq x_1$
$l_5 l_1$	$1 \leq x_1$
bad	$\emptyset$

Table 2
$SiSt(\mathcal{A}')$

Localities	**Clock Space**
$l_0 l_0$	$x_1 \leq 3$
$l_1 l_1$	$x_1 \geq 2$
$l_4 l_4$	$x_1 \geq 2$
$l_2 l_0$	$2 \leq x_1 \leq 3$
$l_3 l_1$	$2 \leq x_1$
$l_5 l_1$	$\emptyset$
bad	$\emptyset$

Table 3
Solution of control algorithm 1

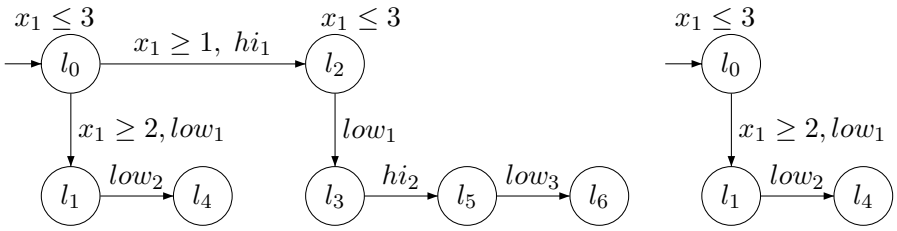

Fig. 6. Control example :$\mathcal{A}$ and $\mathcal{A}_{\setminus \Sigma_{priv}}$

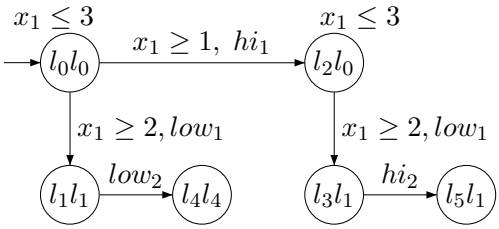

Fig. 7. Control example : $\mathcal{A}'$

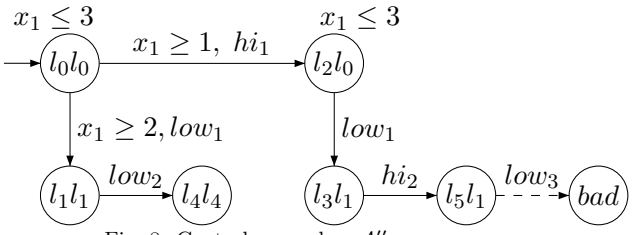

Fig. 8. Control example : $\mathcal{A}''$

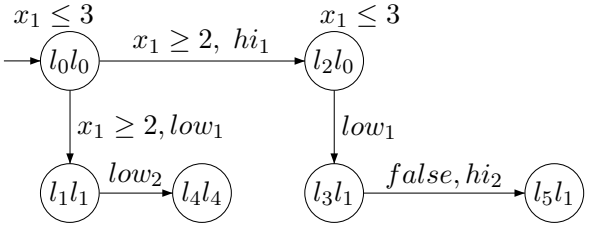

Fig. 9. Solution of the CSNNI control problem on $\mathcal{A}$

6 Conclusion and future work

In this paper we have reformulated SNNI in dense time setting for four semantics of Security Timed Automata and addressed the decidability issues of their associate verification problem. We have also addressed, for the first time, the control synthesis problem for Timed CSNNI and Timed StSNNI and, in each case, provided algorithms computing an associated controller.

In the following we would like to highlight four specific aspects we intend to focus our attention on, in view of further developments: Timed BSNNI controller, verification and controller synthesis for Timed BNDC, timed control with partial

observability and Timed Intransitive Non Interference.

Control synthesis techniques for Timed BSNNI. Of course it remains to extend control synthesis to Timed BSNNI to complete the picture depicted in this paper.

Proof and control synthesis techniques for Timed BNDC. In [9] it is proposed *Bisimulation-based nondeducibility on Composition* (BNDC) as the most natural untimed information flow property: a system $\mathcal{S}$ satisfies BNDC if for any private level user Π, the public behavior of $\mathcal{S}$ is not affected by its interaction with Π. A next issue is the extension to a dense time reformulation of BNDC in our theory together with related proof and control synthesis techniques.

Timed non interference control synthesis with partial observability. Depending on the nature of the plant, the non-controllable actions could be observable (*full observability*) or only a proper subset (*partial observability*) may be observable by the controller. In the timed setting, partial observability assumption applies not only to uncontrollable actions but also to the clocks of the security system.

Timed intransitive non interference control synthesis. Non interference ensures absence of information flow. But absence of information flow is rarely very interesting as a description of confidentiality, as many practical applications are intended to preserve confidentiality, but nonetheless leak information. Also, another important issue is the extension to dense time reformulation of *Intransitive Non Interference* (INI). This term refers to information flow properties required of systems like downgraders in which it may be legitimate for information to flow indirectly between two users but not directly. A clever and complete development of INI can be found in [18]. It is formulated in terms of purging and based on Moore and Mealy machines. Purging involves applying a function to the history of the system up to which removes all those parts that should not influence what a given agent sees. This purge-based definition of INI has been also characterized in [11] in terms of observability in the context of Discrete Event Systems (DES) as introduced by Lin and Wonham [13]. Recently Hadj-Alouane, Lin and Yeddes [10] in a personal communication presented a preliminary work in view of an extension of this purge-based definition of INI to dense time in the context of Timed Automata and it has to be stressed that nor decidability of Timed INI or Timed INI control synthesis are addressed in this draft. Moreover all the approaches to INI mentioned above are limited to deterministic systems and thus are not applicable to distributed systems. However nondeterministic generalizations for untimed INI have been proposed in [16,15,4,6].

References

[1] Aceto, L., A. Ingólfsdóttir, M. L. Pedersen and J. Poulsen, *Characteristic formulae for timed automata*, RAIRO - Theoretical Informatics and Applications **34** (2000), pp. 565–584.

[2] Altisen, K. and S. Tripakis, *Tools for controller synthesis of timed systems*, in: *RT-TOOLS'02*, 2002.

[3] Alur, R. and D. Dill, *A theory of timed automata*, Theoretical Computer Science **126** (1994), pp. 183–235.

[4] Backes, M. and B. Pfitzmann, *Intransitive non-interference for cryptographic pusposes*, in: *Proc. of the IEEE Symp. on Security and Privacy*, 140–152, 2003.

[5] Barbuti, R. and L. Tesei, *A decidable notion of timed non-interference*, Fundamenta Informaticae **54** (2003), pp. 137–150.

[6] Bossi, A., C. Piazza and S. R. S. Rossi, *Modelling downgrading in information flow security*, in: *17th IEEE Computer Security Foundations Workshopsp (CSFW'2004)*, 2004, pp. 187–.

[7] D'Souza, D. and P. Madhusudan, *Timed control synthesis for external specifications*, in: *Proc. STACS '02*, number 2285 in LNCS, 2002.

[8] Felten, E. W. and M. A. Schneider, *Timing attacks on web privacy*, in: *CCS '00: Proceedings of the 7th ACM conference on Computer and communications security* (2000), pp. 25–32.

[9] Focardi, R. and R. Gorrieri, *Classification of security properties (part I: Information flow)*, in: R. Focardi and R. Gorrieri, editors, *Foundations of Security Analysis and Design I: FOSAD 2000 Tutorial Lectures*, Lecture Notes in Computer Science **2171** (2001), pp. 331–396.

[10] Hadj-Alouane, N., F. Lin and M. Yeddes, *Timed intransitive non-interference*, Personal communication (2004), pp. 1–4.

[11] Hadj-Alouane, N. B., S. Lafrance, F. Lin, J. Mullins and M. Yeddes, *Characterizing intransitive non-interference in security policies with observability*, IEEE Trans. on Automatic Control (2004).

[12] Laroussinie, F., K. G. Larsen and C. Weise, *From timed automata to logic – and back*, in: J. Wiedermann and P. Hájek, editors, *Proceedings of the 20th International Symposium on Mathematical Fundations of Computer Science (MFCS'95)*, Lecture Notes in Computer Science **969** (1995), pp. 27–41.
URL http://www.lsv.ens-cachan.fr/Publis/PAPERS/PS/LLW-mfcs95.ps.gz

[13] Lin, F. and W. M. Wonham, *On observability of discrete-event systems*, Information Sciences **44** (1988), pp. 173–198.

[14] Maler, O., A. Pnueli and J. Sifakis, *On the synthesis of discrete controllers for timed systems*, in: E. Mayr and C. Puech, editors, *Proc. STACS '95*, number 900 in LNCS (1995), pp. 229–242.

[15] Mantel, H., *Information flow control flow and applications – bridging a gap*, in: *Proc. of the International Symposium of Formal Methods Europe (FME'01)*, 153–172, 2001.

[16] Mullins, J., *Nondeterministic admissible interference* **6** (2000), pp. 1054–1070.
URL http://www.jucs.org/jucs_6_11/

[17] R. Focardi, R. G. and F. Martinelli, *Real-time information flow analysis*, IEEE Journal on Selected Areas in Communications **21** (2003), pp. 20–35.

[18] Rushby, J., *Noninterference, transitivity and channel-control security policies*, Technical Report CSL-92-02, SRI International, Menlo Park CA, USA (1992).

[19] Tripakis, S., "The analysis of timed systems in practice," Ph.D. thesis, Université Joseph Fourier (1998).

[20] Wong-Toi, H. and G. Hoffmann, *The control of dense real-time dicrete event systems*, in: *Proc. 30th IEEE Conf. Decision and Control*, 1527–1528, 1991.