A Notion of Non-Interference for Timed Automata

(Extended Abstract)

Roberto Barbuti[†], Nicoletta De Francesco[‡], Antonella Santone[‡], Luca Tesei[†]

[†]Dipartimento di Informatica, Università di Pisa
Corso Italia, 40, 56125 Pisa - Italy
e-mail: {barbuti, tesei}@di.unipi.it
fax: +39-050887226

[‡]Dipartimento di Ingegneria dell'Informazione, Università di Pisa
Via Diotisalvi, 2, 56126 Pisa - Italy
e-mail: {nicoletta.defrancesco, antonella.santone}@iet.unipi.it
fax: +39-050568522

Abstract

The non-interference property of concurrent systems is a security property concerning the flow of information among different levels of security of the system. If we suppose that the behavior of the system is divided into a high security behavior and low security behavior, we say that the system respects the non-interference property if the low-level behavior is not affected by the high-level one. In this paper we discuss a notion of non-interference for real-time systems, where the time is a crucial parameter. We represent these systems by timed automata. The non-interference notion we define depends on a number n representing a minimum delay between high-level actions such that the system meets the above desired behavior. Thus our non-interference notion is parametric with respect to time delays between high-level actions.

1 Introduction

The non-interference property [6] of concurrent systems is a security property concerning the flow of information among different levels of security of the system. Let us suppose for simplicity that the security levels are two: *high* and *low*. If we suppose that the behavior of the system is divided into a high security behavior and low security behavior, we say that the system respects the non-interference property if the low-level behavior is not affected by the high-level one. In other words, if a system P acts in an environment where low-level and high-level users are present, P is secure if the behaviors P offers to the low-level users are the same as though the high-level users had never communicated anything.

In this paper we discuss a notion of non-interference for real-time systems, where the time is a crucial parameter. We represent these systems by timed automata [1], which are a well-known formalism to describe time constraints on the events of systems.

Consider a timed automaton T that models a speed-dependent real-time system like an airplane control system. It has to control a lot of basic events and has to respond with basic actions in order to maintain the flight stability. These actions and events may be considered low-level actions and are always activated. When the pilot decides, for example, to turn right, he uses the cloche and this may be considered as a high-level action. Thus the system receives high-level events separated by certain delays; *it must respond to them and must continue to catch and manage basic events*. When this happens we say that high-level actions delays magnitude does not affect the basic behavior of the system.

The non-interference notion we define depends on a number n representing a minimum delay between high-level actions such that the system meets the above desired behavior. Thus our non-interference notion is parametric with respect to time delays between high-level actions.

Section 2 recalls timed automata, Section 3 defines non-interference for timed automata, Section 4 shows an example, and Section 5 discusses the results and future work.

2 Timed Automata

In this section we recall the definition of timed automata [1]. In the following, $\mathcal{R}$ is the set of real numbers and $\mathcal{R}^+$ the set of non-negative real numbers. A *clock* takes values from $\mathcal{R}^+$. Given a set $\mathcal{X}$ of clocks, a *clock valuation* over $\mathcal{X}$ is a function assigning a non-negative real number to every clock. The set of valuations of $\mathcal{X}$, denoted $\mathcal{V}_\mathcal{X}$, is the set of total function from $\mathcal{X}$ to $\mathcal{R}^+$. Given $\nu \in \mathcal{V}_\mathcal{X}$ and $\delta \in \mathcal{R}^+$, with $\nu + \delta$ we denote the valuation that maps each clock $x \in \mathcal{X}$ into $\nu(x) + \delta$.

Given a set $\mathcal{X}$ of clocks, a *reset* γ is a subset of $\mathcal{X}$. The set of all resets of $\mathcal{X}$ is denoted by $\Gamma_\mathcal{X}$. Given a valuation $\nu \in \mathcal{V}_\mathcal{X}$ and a reset γ, with $\nu \backslash \gamma$ we denote the valuation

$$\nu\backslash\gamma(x) = \begin{cases} 0 & \text{if } x \in \gamma \\ \nu(x) & \text{if } x \notin \gamma \end{cases}$$

Given a set $\mathcal{X}$ of clocks, the set $\Psi_{\mathcal{X}}$ of *clock constraints* over $\mathcal{X}$ are defined by the following grammar:

$$\psi ::= \mathit{true} \mid \mathit{false} \mid \psi \wedge \psi \mid \psi \vee \psi \mid \neg\psi \mid x\#t \mid x - y\#t$$

where $x, y \in \mathcal{X}$, $t \in \mathcal{R}^+$, and $\#$ is a binary operator in $\{<, >, \leq, \geq, =\}$. Clock constraints are evaluated over clock valuations. The satisfaction by a valuation $\nu \in \mathcal{V}_{\mathcal{X}}$ of the clock constraint $\psi \in \Psi_{\mathcal{X}}$ is denoted by $\nu \models \psi$.

Definition 1 (Timed automaton) *A timed automaton T is a tuple $(Q, \Sigma, \mathcal{E}, I, R, \mathcal{X})$, where: Q is a finite set of states, Σ is a finite alphabet of actions, $\mathcal{E}$ is a finite set of edges, $I \subseteq Q$ is the set of initial states, $R \subseteq Q$ is the set of repeated states, $\mathcal{X}$ is a finite set of clocks. Each edge $e \in \mathcal{E}$ is a tuple in $Q \times \Psi_{\mathcal{X}} \times \Gamma_{\mathcal{X}} \times \Sigma \times Q$.*

If $e = (q, \psi, \gamma, \sigma, q')$ is an edge, q is the source*, q' is the* target*, ψ is the* constraint*, σ is the* label*, γ is the* reset*.*

The semantics of a timed automaton T is an infinite transition system $\mathcal{S}(T) = (S, \rightarrow)$, where S is a set of states and $\rightarrow$ is the transition relation. The states S of $\mathcal{S}(T)$ are pairs (q, ν), where $q \in Q$ is a state of T, and ν is a valuation. An initial state of $\mathcal{S}(T)$ is a state (q, ν), where $q \in I$ is an initial state of T and ν is the valuation which assigns 0 to every clock in $\mathcal{X}$. At any state q, given a valuation ν, T can stay idle or it can perform an action labeling an outgoing edge e. The rules to derive the transitions of $\mathcal{S}(T)$ are the following:

$$1.\ \frac{\delta \in \mathcal{R}^+}{(q, \nu) \xrightarrow{\delta} (q, \nu + \delta)} \qquad 2.\ \frac{(q, \psi, \gamma, \sigma, q') \in \mathcal{E}, \nu \models \psi}{(q, \nu) \xrightarrow{\sigma} (q', \nu\backslash\gamma)}$$

Rule 1. represents the case in which T stays idle in a state and the time passes, while Rule 2. corresponds to the occurrence of an action.

Definition 2 (run, action sequence) *Given a timed automaton $T = (Q, \Sigma, \mathcal{E}, I, R, \mathcal{X})$, a* run *of the automaton is an infinite sequence of states and transitions of $\mathcal{S}(T)$ $s_0 \xrightarrow{l_0} s_1 \xrightarrow{l_1} \ldots$ where*

- *$s_0 = (q, \nu)$ where $q \in I$ and $\nu(x) = 0$ for every $x \in \mathcal{X}$*
- *a state $q \in R$ exists such that q occurs infinitely often in the pairs of the sequence $\{s_i\}$*

Note that, given a run $s_0 \xrightarrow{l_0} s_1 \xrightarrow{l_1} \ldots$, for each i, $l_i \in (\Sigma \cup \mathcal{R}^+)$. Let r be a run.

- *The* time sequence *$\overline{t_j}$ of the time elapsed from state s_0 to state s_j in r is defined as follows:*

$$t_0 = 0$$

$$t_{i+1} = t_i + \begin{cases} 0 & \textit{if } l_i \in \Sigma \\ l_i & \textit{otherwise} \end{cases}$$

- *The* event sequence *of the events occurring during r, including the elapsed times, is defined as follows:* $(l_0, t_0)(l_1, t_1)\ldots$

- *The* action sequence *of r is the projection of the event sequence of r on the pairs* $\{(l,t) | l \in \Sigma\}$

Definition 3 (timed word, timed language) *Let Σ be an alphabet. A* timed word *over Σ is an infinite sequence $\overline{(\sigma, t)} = (\sigma_0, t_0)(\sigma_1, t_1)\ldots$, where $\sigma_i \in \Sigma$, $t_i \in \mathcal{R}^+$ and $t_i \leq t_i + 1$, for all $i = 0, 1, \ldots$.*

A timed language *over Σ is a subset of the set of all timed words over Σ.*

Definition 4 (acceptance) *Given a timed automaton $T = (Q, \Sigma, \mathcal{E}, I, R, \mathcal{X})$, a timed word w over Σ is* accepted *by T iff there exists a run r of T such that $w = v$, where v is the action sequence of r. The set of timed words accepted by T is called the* accepted language *of T and is denoted by $\mathcal{L}(T)$.*

Note that we use the Büchi acceptance condition for the runs [1]. Moreover, we say that two timed automata T_1 and T_2 are *equivalent* if $\mathcal{L}(T_1) = \mathcal{L}(T_2)$ and we write $T_1 \approx T_2$

Automata with ϵ edges are defined in the same way, but with a special action, the non-observable action ϵ, belonging to Σ. Thus some transitions of the semantic automaton $\mathcal{S}(T)$ can be labeled by ϵ, and they are called ϵ transitions. When defining the accepted language for an automaton with ϵ edges, we must consider the action sequences as the the projection of the event sequences on the pairs $\{(l,t) | l \in \Sigma - \{\epsilon\}\}$.

The design of complex systems can be simplified by modeling subsystems with different timed automata and considering, for the whole system, the product of them. The product operation is a syntactic operation between timed automata.

Definition 5 (Product) *Let $T_1 = (Q_1, \Sigma_1, \mathcal{E}_1, I_1, R_1, \mathcal{X}_1)$ and $T_2 = (Q_2, \Sigma_2, \mathcal{E}_2, I_2, R_2, \mathcal{X}_2)$ be two timed automata with $\mathcal{X}_1 \cap \mathcal{X}_2 = \emptyset$. The product of T_1 and T_2, denoted by $T_1 \parallel T_2$, is the following timed automaton:*

$$T_1 || T_2 = \langle Q_1 \times Q_2, \Sigma_1 \cup \Sigma_2, \mathcal{E}, I_1 \times I_2, R_1 \times R_2, \mathcal{X}_1 \cup \mathcal{X}_2 \rangle$$

where $\mathcal{E}$ is defined by:

1. **Synchronization actions**

   $\forall \sigma \in \Sigma_1 \cap \Sigma_2, \forall (q_1, \psi_1, \gamma_1, \sigma, q_1') \in \mathcal{E}_1, \forall (q_2, \psi_2, \gamma_2, \sigma, q_2') \in \mathcal{E}_2$

   $\mathcal{E}$ contains $((q_1, q_2), \psi_1 \wedge \psi_2, \gamma_1 \cup \gamma_2, \sigma, (q_1', q_2'))$

2. T_1 **actions**
$\forall \sigma \in \Sigma_1 \backslash \Sigma_2, \forall (q, \psi, \gamma, \sigma, q') \in \mathcal{E}_1, \forall s \in Q_2$
$\mathcal{E}$ *contains* $((q,s), \psi, \gamma, \sigma, (q', s)$

3. T_2 **actions**
$\forall \sigma \in \Sigma_2 \backslash \Sigma_1, \forall (q, \psi, \gamma, \sigma, q') \in \mathcal{E}_2, \forall s \in Q_1$
$\mathcal{E}$ *contains* $((s,q), \psi, \gamma, \sigma, (s, q'))$

Thus, the product automaton behavior is the interleaving of the components behaviors where actions with the same name are executed synchronously.

3 Non-Interference for Timed Automata

Non-interference for concurrent non-real-time systems has been modeled in process algebras using bisimulation equivalence (see, for example, [8, 3, 4, 7]): if the actions of a system P are divided into high-level and low-level ones, *the system respects the non-interference property if its behavior in absence of high-level actions is equivalent to its behavior, observed on low-level actions, when high-level actions occur*. To compare the two behaviors we check equivalence of two processes obtained from P: in the first one high-level actions are forbidden, in the second one they are hidden. Different notions of bisimulation equivalence are used to model different notions of non-interference. These have been reformulated in [5] in a real-time setting using a discrete time process algebra.

In this paper we use timed automata approach to real-time systems modeling to define a new notion of non-interference. This is based on high-level actions delays magnitude and on equivalence of timed automata. Given a natural number n, we say that *high-level actions do not interfere with the system, considering a minimum delay n, if the system behavior in absence of high-level actions is equivalent to the system behavior, observed on low-level actions, when high-level actions can occur, but the delay between any two of them is greater than or equal to n*. Thus, if the environment of the system does not offer high-level events separated by less than n time units and the property holds, there is no way for low-level users to detect any high-level activity. The main improvement respect to the untimed notion of non interference is that time is observable and the property captures those systems in which the time delay between high-actions *cannot* be used to construct illegal information flows from high-level to low-level users.

Let T be a timed automaton over the alphabet Σ. We suppose that Σ is partitioned into two disjoint sets of actions H and L: H is the set of high-level actions, while L is the set of low-level ones.

To observe the behavior of an automaton T in absence of high-level actions, we can compose T in parallel with an automaton, from now on called $Inhib_H$, that does not allow the execution of high-level actions.

The automaton $Inhib_H$ is shown in Figure 1, where the arc represents a set of edges, one for each action in H, with the same constraint and reset. In

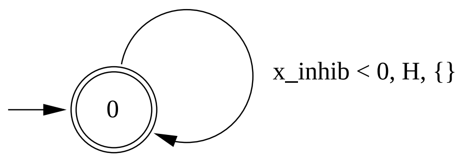

Figure 1: $Inhib_H$

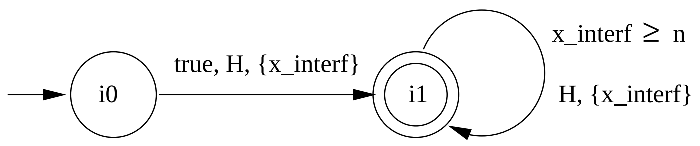

Figure 2: The structure of $Interf_H^n$

the product $T||Inhib_H$, where high-level actions are the only synchronization actions, the component T cannot have a transition labeled by $\sigma \in H$ because his partner in synchronization, $Inhib_H$, never performs high-level actions. Thus only low-level actions are executed.

We have that $\mathcal{L}(T||Inhib_H)$ contains all basic behaviors of T, i.e. all timed words obtained by runs in which high-level actions do not occur.

Consider the automaton $Interf_H^n$ in Figure 2. As in the automaton $Inhib_H$, each arc represents a set of edges, one for each action in H. This automaton allows the execution of high-level actions only when separated by at least n time units. Given an automaton T, the product of T with $Interf_H^n$, $T||Interf_H^n$, allows to observe the set of all behaviors of T where high-level actions occur at times separated by an interval whose length is greater than, or equal to a given minimum delay n.

Before giving the definition of n-non-interference, we need the following operation, which, applied to an automaton T, gives an automaton having the same behaviors of T, but high-level actions are considered non-observable.

Definition 6 (hiding of high-level actions) *Let T be a timed automaton over an alphabet $\Sigma = (H, L)$. We denote by T/H the automaton obtained by T by replacing each edge $e = (q, \psi, \gamma, \sigma, q')$ of T, with $\sigma \in H$, with $e' = (q, \psi, \gamma, \epsilon, q')$.*

Now we can define formally the intuitive notion of non-interference given above.

Definition 7 (n-non-interference) *Let T be a timed automaton over an alphabet $\Sigma = (H, L)$, and $n \in I\!N$. High-level actions do not interfere in T, with a minimum delay n, if and only if*

$$(T||Interf_H^n)/H \;\approx\; T||Inhib_H \tag{1}$$

The definition above requires that T has a structure such that basic behaviors do not change whenever high-level actions are recognized in successive steps separated by at least n time units.

3.1 Timed languages characterization

In this section we characterize the notion of non-interference defined in the previous section using timed languages. First, consider the restriction of a timed language to low-level actions.

Definition 8 (restriction to low-level actions) *Let I be a set of timed words over an alphabet $\Sigma = (H, L)$. $I|_L$ is the subset of I where all elements are timed words on L:*

$$I|_L = \{\overline{(\sigma, t)} \in I \mid \forall (\sigma_i, t_i) \in \overline{(\sigma, t)}.\ \sigma_i \in L\}$$

The following proposition holds:

Proposition 1 *Let T be a timed automaton over an alphabet $\Sigma = (H, L)$. Then $\mathcal{L}(T)|_L = \mathcal{L}(T \| Inhib_H)$.*

The restriction to consider only high level actions separated by at least n time units is defined as follows.

Definition 9 (n-delay restriction) *Let I be a set of timed words over an alphabet $\Sigma = (H, L)$. Let n be a natural number. I^n_H is the subset of I containing all timed words in I such that the delay between any two actions in H is at least n:*

$$I^n_H = \{\overline{(\sigma, t)} \in I \mid \forall (\sigma_i, t_i), (\sigma_j, t_j) \in \overline{(\sigma, t)}.\ i \neq j \wedge \sigma_i, \sigma_j \in H \Rightarrow |t_i - t_j| \geq n\}$$

Proposition 2 *Let T be a timed automaton over an alphabet $\Sigma = (H, L)$. Then $\mathcal{L}(T)^n_H = \mathcal{L}(T \| Interf^n_H)$.*

The hiding of high-level actions is expressed as follows.

Definition 10 (hiding of high-level actions) *Let I be a set of timed words over an alphabet $\Sigma = (H, L)$. For every $\omega = \overline{(\sigma, t)}$ in I there is a correspondent element ω' in I/H such that ω' is the projection of the sequence ω on the pairs $\{(\sigma, t) | \sigma \in L\}$:*

$$I/H = \left\{ \omega' \;\middle|\; \begin{array}{l} \omega = \overline{(\sigma, t)} \in I \textit{ and } \omega' \textit{ is the projection of the sequence } \omega \\ \textit{on the pairs } \{(\sigma, t) | \sigma \in L\}\ \} \end{array} \right\}$$

Proposition 3 *Let T be a timed automaton over an alphabet $\Sigma = (H, L)$. Then $\mathcal{L}(T)/H = \mathcal{L}(T/H)$.*

The following proposition states that the above characterization correctly expresses n-non-interference.

Proposition 4 *Let T be a timed automaton over an alphabet $\Sigma = (H, L)$, and $n \in I\!N$.*

$$T \textit{ is } n\textit{-non interfering} \quad \textit{iff} \quad \mathcal{L}(T)^n_H / H = \mathcal{L}(T)|_L$$

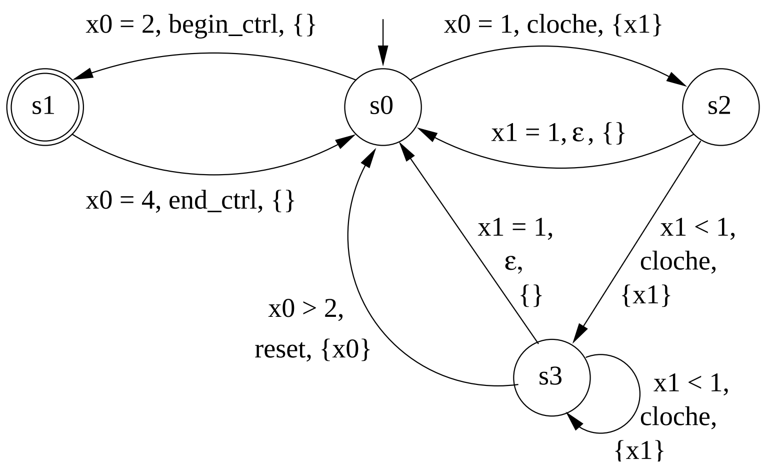

Figure 3: Automaton T: a simplified airplane control

4 An example

In this section we consider a simple airplane control T (Figure 3) and study its non-interference properties. The system periodically executes, at predefinite instants, a set of operations to control flight stability. The control operations begin with action *begin_ctrl* and end with action *end_ctrl*. State *s1* is an abstraction for the control operations, which require 2 time units to be completed. One time unit before entering each control cycle the system can catch an input action from the pilot; in this simple case we consider only a single input action *cloche* we imagine modeling cloche movements. When *cloche* occurs, the system handles it and then continues to manage control actions. Let *begin_ctrl* and *end_ctrl* be low-level actions and *cloche, reset* be the high-level actions.

At first consider the behavior of T when the high-level action *cloche* is disabled (i.e. the moves of $T \parallel Inhib_H$); this consists in a simple cycle between states *s0* and *s1* where transitions are separated by exactly 2 time units.

Thus $\mathcal{L}(T \parallel Inhib_H)$ contains a single timed word

$$(begin\_ctrl, 2)(end\_ctrl, 4) \cdots (begin\_ctrl, 2 + 4i)(end\_ctrl, 4(i + 1)) \cdots \qquad (2)$$

Consider, now, the general behavior of the system, i.e. all actions are enabled with no restrictions. When *cloche* occurs, the system moves to state *s2*. When the system is in *s2* it is handling the *cloche* action. This operation requires one time unit, and then the system returns to the initial state. While being in *s2*, the system can catch other *cloche* actions going to state *s3*. If the time needed to handle them exceeds the time interval between two basic control cycles, it is necessary to postpone the beginning of the successive control cycle (action *reset*).

Intuitively, the *cloche* actions interfere with the basic system behavior only if they are too close to each other. In this case their handling could require a delay such that the times in which the basic control cycle is performed will be modified (the *reset* action is executed). To see this formally with our automaton approach to non-interference consider a natural number n and the automaton $T \parallel Interf_H^n$. If $n = 0$, the *reset* action could be executed and consequently the system in which high-level actions can occur without restrictions on their relative delays does not satisfy our non-interference definition. If, instead, $n \geq 1$, only one *cloche* action occurs between two successive control cycles, and it is managed without affecting the basic behavior. Thus, for each $n \geq 1$, $(T \parallel Interf_H^n)/H$ generates the single timed word (2) which in turn is the unique timed word generated by $T \parallel Inhib_H$. From Proposition 4 we conclude that high-level actions do not interfere in T with a minimum delay $n \geq 1$. On the other hand, if $n = 0$ there is interference.

5 Discussion

In order that the theory we have developed can become useful in practice, we must check the equivalence of automata. Checking equivalence of two timed automata implies checking inclusion in both directions of the respective recognized languages. It is well known that for general timed automata language inclusion is undecidable. However, the language inclusion problem can be solved if we use deterministic automata [1] or event-clock automata [2]. To make the method effective also in the general case, we are following two directions. On one side we are studying weaker notions of non-interference which are checkable on general timed automata. On the other side, we are investigating sufficient conditions on the structure of the automata allowing deciding language inclusion required for non-interference checking. This direction is promising since the automata that have to be compared have a very similar structure derived from the initial automaton T. We plan to develop a checkable sufficient condition to n-non-interference requiring a reasonable structure for automaton T.

References

[1] R. Alur, D.L, Dill. A Theory of Timed Automata. *Theoretical Computer Science*, 126, 183–235, 1994.

[2] Alur, R., Fix, L. and Henzinger, T.A. Event-Clock Automata: A Determinizable Class of Timed Automata. *Theoretical Computer Science*, 204, 1997.

[3] R. Focardi, R. Gorrieri. Automatic Compositional Verification of Some Security Properties. In *Proceedings of Second International Workshop on Tools and Algorithms for Construction and Analysis of Systems (TACAS'96)*. Lecture Notes in Computer Science 1055, 167-186, 1996.

[4] R. Focardi, R. Gorrieri. The Compositional Security Checker: A Tool for the Verification of Information Flow Security Properties. *IEEE Transactions on Software Engineering*, 23(9): 550-571, 1997.

[5] R. Focardi, R. Gorrieri and F. Martinelli Information Flow in a Discrete-Time Process Algebra In Proceedings of 13th IEEE Computer Security Foundations Workshop (CSFW'00), (P. Syverson ed.), IEEE press, Cambridge, England, July 2000.

[6] J.A. Goguen, J. Meseguer. Security Policy and Security Models. In *Proceedings of the 1982 IEEE Symposium on Security and Privacy*, pp. 11-20. IEEE Computer Society Press, 11-20, 1982.

[7] P. Y. A. Ryan, S. A. Schneider. Process Algebra and Non-Interference. In *Proceedings of 12th Computer Security Foundations Workshop (CSFW'99).*

[8] A.W. Roscoe, J.C.P. Woodcock, L. Wulf. Non-Interference Through Determinism. *Journal of Computer Security*, 4(1), 1996.