In *Proceedings of 13th IEEE Computer Security Foundations Worhshop* (CSFW'00), IEEE Press, Cambridge (UK), July 2000.

Information Flow Analysis in a Discrete-Time Process Algebra *

Riccardo Focardi
Dipartimento di Informatica, Università Ca' Foscari di Venezia, Italy.
e_mail: focardi@dsi.unive.it

Roberto Gorrieri
Dipartimento di Scienze dell'Informazione, Università di Bologna, Italy.
e_mail: gorrieri@cs.unibo.it

Fabio Martinelli
Istituto per le Applicazioni Telematiche C.N.R., Pisa, Italy.
e_mail: Fabio.Martinelli@iat.cnr.it

Abstract

Some of the non interference properties studied in [4, 6, 18] for information flow analysis in computer systems, notably $BNDC$, are reformulated here in a real-time setting. This is done by enhancing the Security Process Algebra of [6, 10] with some extra constructs to model real-time systems (in a discrete time setting); and then by studying the natural extensions of those properties in this enriched setting. We prove essentially the same results known for the untimed case: ordering relation among properties, compositionality aspects, partial model checking techniques. Finally, we illustrate a case study of a system that presents no information flows when analyzed without considering timing constraints. But, when the specification is refined with time, some interesting information flows are detected.

1 Introduction

Information flow analysis is considered the main technique for studying security in computer systems. Many different properties have been proposed in the literature to capture the intuitive idea of information flow. Some years ago, the authors have made an effort to classify most of them once formulated in a uniform framework, namely a CCS-like process algebra where actions are partitioned into two different levels of confidentiality (see [5, 6]). The resulting process algebra (called *Security Process Algebra*, SPA for short) can express only the *nondeterministic* (sometimes called possibilistic) behaviour of a system. This is enough to model the functionality of systems and hence to be used as a basis for capturing *logical* information flows, e.g., covert channels. As a result of the classification effort of [5, 6], two of the authors proposed $BNDC$ as the most natural security property: if a SPA specification satisfies $BNDC$, then no information flow is possible from high level users to low level ones. The acronym $BNDC$ stands for Bisimulation Non Deducibility on Compositions: roughly, a specification E is $BNDC$ if for all high level users Π, the low view of the behaviour of E interacting with Π is bisimulation equivalent [19] to the low view of the behaviour of E in isolation; hence, from the point of view of a low level observer, the system with or without the interaction with the high level user Π are indistinguishable. $BNDC$, as well as its related approximation properties $BSNNI$ and $SBSNNI$, are all non interference like properties. [1]

In this paper we want to extend the theory of [4, 5, 6, 16, 18] to a real-time setting in order to be able to capture, besides logical information flows, also *time dependent* information flows, e.g., timing covert channels. In principle, this can be done:

- by taking a real-time version of SPA, following one of the many approaches proposed in the literature to extend process algebras with real-time features

* Work partially supported by MURST Progetto "Certificazione automatica di programmi mediante interpretazione astratta" and Progetto "TOSCA"; CNR Progetto "Modelli e Metodi per la Matematica e l'Ingegneria"; CSP Progetto "ISA: Isp Secured trAnsactions".

[1] The long acronyms $BNDC$, $BSNNI$ (Bisimulation Strong Nondeterministic Non-Interference) and $SBSNNI$ (Strong BSNNI), certainly less than optimal, are kept here only for homogeneity with the papers [5, 6] where they have been introduced.

(see, e.g., [2, 14, 22, 23, 28]), and then

- by adapting the non interference properties of interest to this enriched setting.

Ideally, a more concrete specification might show up new information flows. Indeed, even if one starts from an untimed specification that does not show logical information flows, after having specified its timing requirements, it may be the case that some time dependent information flows are now possible by exploiting the timing of actions to send indirect information from high users to low users.

The first problem we have to face is the real-time extension of SPA. As our aim is a feasibility study, we have decided to follow a simple approach, where time is discrete, actions are durationless and there is one special $tick$ action to represent the elapsing of time. These are the features of the so-called *fictitious clock* approach of, e.g., [2, 14, 28] . A global clock is supposed to be updated whenever all the processes of the system agree on this, by globally synchronizing on action $tick$. All the other actions are assumed to take no time. This is reasonable if we choose a time unit such that the actual time of an action is negligible w.r.t. the time unit. Hence, the computation proceeds in lock-steps: between the two global synchronizations on action $tick$ (that represent the elapsing of one time unit), all the processes proceeds asynchronously by performing durationless actions. Moreover, another feature of the *Timed SPA* (*tSPA* for short) is the so-called *maximal progress assumption* (e.g. see [14, 28]) according to which $tick$ actions have lower priority w.r.t. internal τ actions: a communication (or an internal activity τ) prevents time from occurring, hence the clock synchronization on $tick$ takes place only when all local processes have completed the execution of all the possible communications in that round. Finally, *tSPA* offers a construct, called the *idling* operator, for delaying the execution of the currently executable actions of its argument process. Once one has specified the system in SPA, timing constraints can be added to the specification by simply giving time duration to actions. For instance, if $a.E$ is a SPA process and we assume that action a lasts two time units, we can represent this in *tSPA* as $a.tick.tick.E'$, where E' is obtained from E by applying this transformation to all actions, according to their assumed duration.

The second problem is to adapt the security properties studied for $SPA$to the new real-time setting of *tSPA*. As the untimed non interference like properties of [5, 6] are defined w.r.t. weak bisimulation, we have first of all to define a suitable bisimulation-based equivalence notion for *tSPA*. Then, the definition of the timed version of $BNDC$ (called $tBNDC$ for short) is obvious.

It is worthwhile noticing that, even if E can be $BNDC$, E' may be not $tBNDC$; this is the case when the addition of concrete, implementation-oriented information to the specification may offer new, time dependent information flows that cannot be revealed by the more abstract, purely nondeterministic SPA specification. An example in Section 4 illustrates this issue.

Like $BNDC$, also $tBNDC$ is difficult to check in practice because it contains a universal quantification over all possible interacting high level users. It turns out that the timed version of the approximations of $BNDC$ (namely *tBSNNI* and *tSBSNNI*) are also reasonable approximations for $tBNDC$. Finally, we prove that the techniques developed in [15, 16, 18] for analyzing $BNDC$-like properties in a logical manner can be extended to cope with real-time as considered here. As a side consequence, we obtain that some particular $tBNDC$-like properties are all decidable over finite state *tSPA* processes.

The paper is organized as follows. Section 2 introduces *tSPA* and the timed weak bisimulation. Section 3 is devoted to discuss the non interference properties $tBNDC$, *tBSNNI* and *tSBSNNI*; in particular, it is proved that *tSBSNNI* $\subseteq$ $tBNDC$ $\subseteq$ *tBSNNI* and that *tSBSNNI* is compositional (while $tBNDC$ is not). In Section 4 a simple case study is presented. Section 5 reports the partial model checking technique, that offers a compositional analysis approach also for $tBNDC$. Finally, some conclusive remarks are given in Section 6. Proofs are omitted for lack of space (please see the full paper [7]).

2 A simple model for describing real-time systems

In this section we introduce the Timed Security Process Algebra (*tSPA* , for short), based on the *Security Process Algebra* (*SPA*, for short) used in [5, 6] for the description of security properties in an untimed setting. In SPA, quantitative time aspects between events cannot be expressed, only qualitative temporal orderings. Thus, we extend the SPA language with operators that permit to express the elapsing of time (by using several concepts from [14]).

First we formally introduce the syntax and semantics of our timed language *tSPA*. We have a set $\mathcal{L}$, ranged over by l, of visible actions. $\mathcal{L}$ is $I \cup O$ where $I = \{a, b, c \dots\}$ is the set of input action and $O = \{\overline{a}, \overline{b}, \overline{c}, \dots\}$ of output actions. A special action τ models an internal computation, i.e. it is not visible by an external observer. We also have a complementation function $(\overline{\cdot}) : \mathcal{L} \mapsto \mathcal{L}$, such that $\forall l \in \mathcal{L} : \overline{\overline{l}} = l$ To reflect different levels of secrecy the set $\mathcal{L}$ of visible actions is partitioned into two sets $ActH$ (or simply H) and $ActL$, closed by complementation function. Let $tick$ be the special action used to model time elapsing and let $Act = \mathcal{L} \cup \{\tau\} \cup \{tick\}$, ranged over by $\alpha, \beta, \ldots$, while $\mathcal{L} \cup \{\tau\}$ is ranged over by $a, b \ldots$.

We now describe the syntax for $tSPA$ terms:

$$E ::= \mathbf{0} \mid \alpha.E \mid E_1 + E_2 \mid E_1 \| E_2 \mid E \backslash F \mid \iota(E) \mid Z$$

where $\alpha \in Act$, $F \subseteq \mathcal{L}$ and Z is a process constant that must be associated with a definition $Z \doteq E$. (As usual we assume that constants are guarded, i.e. they must be in the scope of some prefix operator $\alpha.E'$.) We call $Proc^t$ the set of all the $tSPA$ terms.

The formal behaviour of Timed SPA terms is described by means of the *labelled transition system* (LTS, for short) $< Proc^t, Act, \{\xrightarrow{\alpha}\}_{\alpha \in Act} >$, where $\xrightarrow{\alpha}_{\alpha \in Act}$ is the least relation between Timed SPA terms induced by axioms and inference rules of Figure 1.

We give an informal overview of $tSPA$ operators.

- $\mathbf{0}$ is a process that does nothing;
- $\alpha.E$ is a process that can perform an α action and then behaves as E; in particular $tick.E$ represents a process willing to let one time unit pass;
- $E_1 + E_2$ (*choice*) represents the non deterministic choice between the two processes E_1 and E_2; when both are able to perform a *tick* action then the choice can perform this actions and reach a configuration where both the derivatives of the summands can still be chosen;
- $E_1 \| E_2$ (*parallel*) is the parallel composition of processes that can proceed in an asynchronous way but they must synchronize on complementary actions to make a communication, represented by an internal action τ. This is the core operator for time: Both components must agree on performing a *tick* action, but this can be done only if no communications are possible. This enforces the so called *maximal communication progress* assumption, i.e. when a communication is possible, then it must start immediately.
- $E \backslash F$ is the process E when actions in $F \cup \overline{F}$ are prevented;
- $\iota(E)$ (*idling*) allows the process E to wait indefinitely. At every instant of time if the process E performs an action l then the whole system proceeds in this state, while dropping the idling operator. The first two rules enforce the so called *time determinacy* that is a central property of timed process algebras, namely if $E \xrightarrow{tick} E'$ and $E \xrightarrow{tick} E''$ then $E' = E''$. Moreover, the first rule is used to enforce the *maximal progress* assumption, that sometimes is defined as if $E \xrightarrow{tick}$ then $E \not\xrightarrow{\tau}$ (see [2]). Hence, in the first rule, the check on the τ action is needed in order to prevent the generation of a $tick$ action when also a τ action is possible. Roughly, the internal or τ actions have precedence on the elapsing of time.

Let us consider the following relations between SPA terms: $E \stackrel{\tau}{\Longrightarrow} E'$ (or $E \Longrightarrow E'$) if $E \xrightarrow{\tau}_* E'$, and for $\alpha \neq \tau$, $E \stackrel{\alpha}{\Longrightarrow} E'$ if $E \xrightarrow{\tau}_* \xrightarrow{\alpha} \xrightarrow{\tau}_* E'$ (where $\xrightarrow{\tau}_*$ is the reflexive and transitive closure of the $\xrightarrow{\tau}$ relation). Let $Der(E)$ be the set of derivatives of E, i.e. the set of processes that can be reached through the transition relations. Let $Sort(E) = \{\alpha | \alpha \in Act, \exists E' \in Der(E), E' \xrightarrow{\alpha}\}$.

First of all we state two peculiarities of timed SPA. The first one is *time determinacy* and the second one is the *maximal progress* assumption (see [2]).

Lemma 1 *For every tSPA process E we have:*

1. *If* $E \xrightarrow{tick} E'$ *and* $E \xrightarrow{tick} E''$ *then* $E' = E''$;
2. *If* $E \xrightarrow{tick}$ *then* $E \not\xrightarrow{\tau}$. ■

Here we give two simple examples to clarify some aspects of this language.

Example 1 *Consider a simple system, with a sender P and a receiver Q. When P starts, it emits an action start and then tries to communicate on the channel com. The receiver Q waits for a communication on the channel com and then emits the action stop. Hence, in the standard SPA (the untimed part), we could write such a system in this way:*

$$P = \overline{start}.\overline{com}.\mathbf{0}$$
$$Q = com.\overline{stop}.\mathbf{0}$$
$$Sys = (P \| Q) \setminus \{com\}$$

We restrict the composition on the channel com in order to force the communication (synchronization) on this channel.

In this way we are not able to say how much time the communication between P and Q takes. Let us model the same situation in timed SPA. First of all, even though we assume that actions are atomic, i.e. they take no time, we can still model activities which involve the elapsing of time. For example, suppose that a communication on com takes two units of time, then we can refine the system Sys in the following way:

$$P_1 = \overline{start}.\overline{com}.tick.tick.\mathbf{0}$$
$$Q_1 = com.tick.tick.\overline{stop}.\mathbf{0}$$
$$Sys_1 = (P_1 \| Q_1) \setminus \{com\}$$

PREFIXING:

$$\frac{}{\alpha.E \xrightarrow{\alpha} E}$$

CHOICE:

$$\frac{E_1 \xrightarrow{a} E_1'}{E_1 + E_2 \xrightarrow{a} E_1'} \quad \frac{E_2 \xrightarrow{a} E_2'}{E_1 + E_2 \xrightarrow{a} E_2'} \quad \frac{E_1 \xrightarrow{tick} E_1' \quad E_2 \xrightarrow{tick} E_2'}{E_1 + E_2 \xrightarrow{tick} E_1' + E_2'}$$

PARALLEL:

$$\frac{E_1 \xrightarrow{a} E_1'}{E_1 \| E_2 \xrightarrow{a} E_1' \| E_2} \quad \frac{E_2 \xrightarrow{a} E_2'}{E_1 \| E_2 \xrightarrow{a} E_1 \| E_2} \quad \frac{E_1 \xrightarrow{l} E_1' \quad E_2 \xrightarrow{\bar{l}} E_2'}{E_1 \| E_2 \xrightarrow{\tau} E_1' \| E_2'}$$

$$\frac{E_1 \xrightarrow{tick} E_1' \quad E_2 \xrightarrow{tick} E_2' \quad \forall l \in \mathcal{L} \quad \neg(E_1 \xrightarrow{l} \wedge E_2 \xrightarrow{\bar{l}})}{E_1 \| E_2 \xrightarrow{tick} E_1' \| E_2'}$$

CONSTANT:

$$\frac{Z \doteq E \quad E \xrightarrow{\alpha} E'}{Z \xrightarrow{\alpha} E'}$$

RESTRICTION:

$$\frac{E_1 \xrightarrow{\alpha} E_1'}{E_1 \backslash F \xrightarrow{\alpha} E_1' \backslash F} (\alpha \notin F \cup \overline{F})$$

IDLING:

$$\frac{E \not\xrightarrow{tick} \quad E \not\xrightarrow{\tau}}{\iota(E) \xrightarrow{tick} \iota(E)} \quad \frac{E \xrightarrow{tick} E'}{\iota(E) \xrightarrow{tick} \iota(E')} \quad \frac{E \xrightarrow{a} E'}{\iota(E) \xrightarrow{a} E'}$$

Figure 1. Operational semantics for timed SPA, where l ranges over $\mathcal{L}$, a ranges over $\mathcal{L} \cup \{\tau\}$ and α over Act.

In general if we assume that the action a takes n units of time we can substitute every prefix operator $a.P'$ in a process P by $a.tick^n.P'$, where $tick^n$ simply denotes the concatenation of n $tick$ actions. Thus, by looking at the operational semantics we note that the system Sys_1 can (only) perform the following sequence of actions:

$$Sys_1 \xrightarrow{\overline{start}} \xrightarrow{\tau} \xrightarrow{tick} \xrightarrow{tick} \xrightarrow{\overline{stop}}$$

As expected the communication takes two units of time. ■

We can model time-out in our language as follows:

Example 2 *Suppose we want to describe a process which is willing to perform an action a for at most a certain amount of time, say 3 time units. It could be defined as:*

$$Time\_out = tick.tick.tick.\tau.\mathbf{0} + \iota(a.\mathbf{0})$$

By looking at the semantics rules, we note that process $Time\_out$ can perform at most 3 tick actions and during this computation it can always choose to perform the action a. Moreover, after the execution of the three $ticks$, $Time\_out$ cannot wait longer since it has to choose between stopping ($\tau.\mathbf{0}$) or executing immediately a. ■

The equivalence relation we consider is the timed version of observation equivalence (also called weak bisimulation [20]). This equivalence permits to abstract to same extent from the internal behaviour of the systems, represented by the internal actions.

Definition 1 *A relation $\mathcal{R} \subseteq Proc^t \times Proc^t$ is a timed weak simulation iff for every $(p, q) \in \mathcal{R}$ we have:*

- *if $p \xrightarrow{a} p'$ then there exists q' s.t. $q \overset{a}{\Longrightarrow} q'$ and $(p', q') \in \mathcal{R}$,*
- *if $p \xrightarrow{tick} p'$ then there exists q' s.t. $q \overset{tick}{\Longrightarrow} q'$ and $(p', q') \in \mathcal{R}$.* ■

A timed weak bisimulation is a relation $\mathcal{R}$ s.t. both $\mathcal{R}$ and $\mathcal{R}^{-1}$ are timed weak simulations. We represent with $\approx_t$ the union of all the timed weak bisimulations.

In order to prove that two processes P, Q are timed weak bisimilar ($P \approx_t Q$), we only need to provide a timed weak bisimulation that contains them. The process $Sys'_1 = \overline{start}.tick.tick.\overline{stop}.\mathbf{0}$ is timed weakly bisimilar to Sys_1. As a matter of fact, the following is a weak bisimulation which contains them:

$$
\begin{array}{l}
\{(Sys_1, Sys'_1),\\
((\overline{com}.tick.tick.\mathbf{0} \| com.tick.tick.\overline{stop}.\mathbf{0}) \setminus \{com\},\\
\quad tick.tick.\overline{stop}.\mathbf{0}),\\
((tick.tick.\mathbf{0} \| tick.tick.\overline{stop}.\mathbf{0}) \setminus \{com\},\\
\quad tick.tick.\overline{stop}.\mathbf{0}),\\
((tick.\mathbf{0} \| tick.\overline{stop}.\mathbf{0}) \setminus \{com\}, tick.\overline{stop}.\mathbf{0}),\\
((\mathbf{0} \| \overline{stop}.\mathbf{0}) \setminus \{com\}, \overline{stop}.\mathbf{0}),\\
((\mathbf{0} \| \mathbf{0}) \setminus \{com\}, \mathbf{0})\}
\end{array}
$$

An interesting feature of $\approx_t$ is that given two finite-state processes, we can decide in polynomial time whether they belong to $\approx_t$ or not.

3 Security properties in a real-time setting

In this section we present some information flow security properties. In particular, we recast the *non interference* theory proposed in [5, 6] in a real time framework. Hence, we are also able now to detect time effects of the activity of potential enemies.

The central property is the so called *Non deducibility on Composition* (NDC, for short). Its underlying idea is that the system behaviour must be invariant w.r.t. the composition with every high user. This means that the low level users cannot tell anything about the high level activity since, for them, the system is always the same. Indeed, there is no possibility of establishing a communication (i.e. sending information).

Generally, the NDC idea can be represented as follows:

$$\forall \Pi \in High\ users\ E \mid \Pi \equiv E \ \ w.r.t. \quad Low\ users \tag{1}$$

where $\mid$ stands for the composition operator and $\equiv$ for the equivalence relation. This property can be instantiated by assuming different notions of composition and equivalence. Let $\mathcal{E}_H = \{\Pi \mid Sort(\Pi) \subseteq H \cup \overline{H} \cup \{\tau\}\}$ be the set of High users. In terms of SPA parallel composition operator and *bisimulation* equivalence [20] (denoted with $\approx$), we have:

Definition 2 $E \in BNDC$ iff $\forall \Pi \in \mathcal{E}_H : (E \| \Pi) \backslash H \approx E \backslash H$. ■

It is worthwhile noticing that the underlying idea of NDC is very natural for describing security properties. Recently, a generalization of this idea ($GNDC$, for short [10]) has been shown to be a general method for describing properties of cryptographic protocols. Indeed, several existing security properties (e.g. authentication, message authentication, secrecy, non-repudiation, etc...) can be easily encoded in this schema [8, 9, 10]. This opens the way to apply tools for the analysis of Non-Interference also to the analysis of security protocols [3, 6].

3.1 Timed Bisimulation Non Deducibility on Compositions

We restate the $BNDC$ security property in the Timed SPA language, where also timing aspect of systems may be analyzed. The most important property is *timed* $BNDC$ (*tBNDC*, for short) which is the natural extension of $BNDC$ with time. Before formally introducing it, we need to discuss briefly on the nature of the possible High users. We need the following definition of *weakly time aliveness* for processes.

Definition 3 *A process E is directly weakly time alive* iff $E \stackrel{tick}{\Longrightarrow}$. *A process E is weakly time alive* iff *for all $E' \in Der(E)$, we have E' is directly weakly time alive.* ■

It is worthwhile noticing that the above property is preserved by the parallel composition.

In the *tSPA* model, when defining NDC like properties, we cannot consider all high processes for the interaction with the systems. Indeed, we must restrict ourselves to *weakly time alive* processes that can perform only action in $ActH \cup \{\tau, tick\}$. We call $\mathcal{E}^t_H$ the set of such processes.

The reason is the following: a process, that is not directly weakly time alive, may prevent time from elapsing in parallel with every system E. Indeed, in a compound process, time can pass iff both components let it pass. Hence, a high user which is not weakly time alive, could block the time flow and we certainly want to avoid this unrealistic possibility.

The *tBNDC* property in Timed SPA can be thus defined as follows:

Definition 4 *Let $E \in Proc^t$. Then, $E \in$ tBNDC if and only if* $\forall \Pi \in \mathcal{E}^t_H$ *we have* $(E \| \Pi) \backslash H \approx_t E \backslash H$. ■

Due to the presence of the universal quantification *tBNDC* is not very easy to be checked. Thus, by following the line of [5, 6], in the next two sections, we define two other properties, namely *tBSNNI* and *tSBSNNI*. These approximate *tBNDC* from above and below, respectively, and correspond to *BSNNI* and *SBSNNI* as defined in [5, 6].

3.2 *tBSNNI* property

Roughly, E is *tBSNNI* if the system $E\backslash H$, where no high level activity is allowed, behaves like system E that has all the high level actions *hidden* (i.e., transformed into internal τ actions).

We first introduce the *hiding* operator E/H. In the untimed case this is defined by means of the following rules:

$$\frac{E_1 \xrightarrow{\alpha} E_1'}{E_1/F \xrightarrow{\alpha} E_1'/F}(\alpha \notin F \cup \overline{F}) \qquad \frac{E \xrightarrow{f} E' \quad f \in F \cup \overline{F}}{E/F \xrightarrow{\tau} E'/F} \tag{2}$$

We show an interesting property of this operator in the untimed *SPA* language. Consider the Top_H process in *SPA*, i.e. the process that can repeatedly perform every action in $H \cup \overline{H}$. It is defined as:

$$Top_H = \sum_{a \in H \cup \overline{H}} a.Top_H$$

We can see that $(E \| Top_H) \setminus H \approx_t E/H$ for every process E of *SPA*. Actually, this could be considered as a good defining equation for the hiding operator.

Due to the maximal progress assumption, the previous equality does not hold in the timed language, even considering the appropriate process $Top_H^{tick} = \iota(Top_H)$ that can also let the time pass arbitrarily.

Indeed, consider the process $E = \iota(h)$. By using the previous rules for hiding and considering $H = \{h\}$, we obtain that $E/H \xrightarrow{\tau}$ and, at the same time, $E/H \xrightarrow{tick}$ which contradicts the maximal progress assumption. Note that the process $(E\|Top_H^{tick})\backslash H$ cannot perform a $tick$ action.

Thus we need to slightly modify the semantics of the hiding operator E/F as follows:

$$\frac{E_1 \xrightarrow{\alpha} E_1'}{E_1/F \xrightarrow{\alpha} E_1'/F}(\alpha \notin \{F \cup \overline{F} \cup \{tick\}\}) \qquad \frac{E \xrightarrow{f} E' \quad f \in F \cup \overline{F}}{E/F \xrightarrow{\tau} E'/F} \qquad \frac{E \xrightarrow{tick} E' \quad \forall f \in F \cup \overline{F}\ E \not\xrightarrow{f}}{E/F \xrightarrow{tick} E'/F} \tag{3}$$

With these rules, we avoid the possibility that the process E/F performs a *tick* action if at the same time it can perform an action in $F \cup \overline{F}$. Now, we can prove that for every process E its composition with Top_H^{tick} (restricted on the high actions) is equivalent to E/H, i.e., that hiding corresponds to enabling every high level action.

Proposition 1 *For every $tSPA$ process E we have that $(E\|Top_H^{tick}) \setminus H \approx_t E/H$.* ■

Property *tBSNNI* can be now defined as follows:

Definition 5 $E \in$ *tBSNNI* iff $E\backslash H \approx_t E/H$. ■

As remarked, this is an approximation of *tBNDC*. Indeed, *tBNDC* is at least as strong as *tBSNNI*:

Proposition 2 *tBNDC* $\subseteq$ *tBSNNI*

Proof: We observe that Top_H belongs to $\mathcal{E}_H^t$ and so the thesis follows. ■

By using an example similar to the one of [5], we can also prove that the previous inclusion is strict. Consider the following process $E = l.\mathbf{0} + h.h.l.\mathbf{0}$. This process is *tBSNNI* since $E\backslash H = (l.\mathbf{0} + h.h.l.\mathbf{0})\backslash H \approx_t l.\mathbf{0}$ and $E/H \approx_t l.\mathbf{0} + \tau.\tau.l.\mathbf{0} \approx_t l.\mathbf{0}$. But, if we consider the process $\Pi = \iota(\overline{h}.\iota(\mathbf{0}))$ we get that $(E\|\Pi)\backslash H \approx_t l.\mathbf{0} + \tau.\mathbf{0} \not\approx_t l.\mathbf{0} \approx_t E\backslash H$.

3.3 *tSBSNNI* property

An approximation from below for *tBNDC* can be defined as follows:

Definition 6 $E \in$ *tSBSNNI* iff $\forall E' \in Der(E) : E' \in tBSNNI$. ■

This requires that the system is *tBSNNI* in every derivative. By following the proof technique of [6] we can prove that *tSBSNNI* is stronger than *tBNDC*:

Proposition 3 *tSBSNNI* $\subseteq$ *tBNDC*. ■

By using an example similar to the one in [5], we can prove that the previous inclusion is strict.

Proposition 4 *Process $E = l.(\tau.\mathbf{0} + \tau.l.\mathbf{0}) + l.\mathbf{0} + l.h.l.\mathbf{0}$ is tBNDC but not tSBSNNI.*

Proof: For lack of space we omit the proof that $E \in$ *tBNDC*. For $E \notin tSBSNNI$ consider the following derivative $E' = h.l.\mathbf{0}$. This is not *tBSNNI* because $\mathbf{0} \approx_t E'\backslash H \not\approx_t E'/H \approx_t \tau.l.\mathbf{0}$. ■

3.4 Compositionality of properties

One of the most interesting features of *SBSNNI* is that it is compositional. Here we prove that also the timed extension of this property has this nice feature.

Proposition 5 *If $E, F \in tSBSNNI$ then $E\|F \in tSBSNNI$.* ■

By checking *tSBSNNI* over the parallel subcomponents of a system (as done in [6]), it is thus possible to avoid the so called *state-explosion* problem (see [1]) caused by the interleaving of all the possible executions of parallel processes.

Unfortunately for *tBNDC*, as for *BNDC* (see Prop. 1 of [18]), we have the following negative result:

Proposition 6 *tBNDC is not compositional.* ■

4 An example

We consider an example of a process that manages, following a mutual exclusion policy, the accesses to a shared variable (see also Example 3 in [18]). It implements the *no-write-down no-read-up* policy (see [11]): High users can only read the variable and low users can only write it. Hence, the information should only flow from low level users to high level users. However, it could be possible to construct some *covert channels*, i.e., indirect ways of communicating information. We want to check this using the *BNDC*/*tBNDC* properties.

A very first description of the manager may be as follows:

$$C \quad = \quad req_H.\overline{read_H}.C + req_L.write_L.C$$

where the set H of high actions is $\{req_H, read_H\}$ and actions $req_H, \overline{read_H}, req_L$ and $write_L$ represent the (abstractions of) high level read requests, high level reads, low level write requests and low level writes, respectively. In particular an access request can be done by either a high level user or a low level one. After such request the corresponding access is executed and finally the monitor process returns to its initial state C.

We are not modelling time in this first specification. We can thus try to verify the *BNDC* property. Unfortunately, it is easy to see that it is not satisfied since the high level process $X = \overline{req_H}.\mathbf{0}$ can block the monitor process and this is revealed by bisimulation equivalence. It is possible to prove that these potential high level deadlocks may be exploited for constructing covert channels, as reported in [4]. The idea is to consider two instances of the non *BNDC* system, and to alternatively block/unblock them in order to transmit 0 or 1, respectively.

This problem can be resolved by introducing a timeout mechanism which unblock the system if the high level reading is not performed. Again, we start modelling the system without considering time explicitly:

$$C_1 = req_H.(\tau.C_1 + \overline{read_H}.C_1) + req_L.write_L.C_1$$

Now it can be formally verified that this system is *BNDC* (since it is actually *SBSNNI*). Indeed the logical time-out represented by the τ alternative has resolved the deadlock problem. But, if we refine the model by introducing $tick$ actions, it is possible to see that there could be some timing covert channels. For example, suppose to refine $req_L, req_H, write_L$ and the τ (which models time out) by adding one $tick$ for every such action. We also allow the system to idle for $read_H$ actions obtaining the following system:

$$\begin{aligned} C_2 \quad &= \quad req_H.tick.(tick.\tau.C_2 + \iota(\overline{read_H}.C_2)) \\ &\qquad + req_L.tick.write_L.tick.C_2 \end{aligned}$$

This system is not *tBNDC*, in fact consider the following high processes:

$$\begin{aligned} X_1 \quad &= \quad \iota(\overline{req_H}.\iota(read_H.\iota(\mathbf{0}))) \\ X_2 \quad &= \quad \iota(\overline{req_H}.tick.\iota(read_H.\iota(\mathbf{0}))) \end{aligned}$$

It is now easy to see that $(C_2 \| X_1)\backslash H \stackrel{tick}{\Longrightarrow}\stackrel{req_L}{\Longrightarrow}$ and $(C_2 \| X_2)\backslash H \stackrel{tick}{\Longrightarrow}\stackrel{tick}{\Longrightarrow}\stackrel{req_L}{\Longrightarrow}$, while $C_2\backslash H$ is not able to simulate these sequences.

The problem is that the high users can decide how much time to spend in the critical section. Hence, low users can detect the presence of high level activity and even worst they can detect how much time this activity takes. In this way high level users have a lot of possibilities to transmit some information.

For example, by assuming that the time-out is set to n units of time, then if an high user spends $k \leq n$ time units in the critical section then this could be considered as a way of transmitting the integer k and so on.

We solve these problems as follows: First, we impose a discipline of accessing to the variable that forces high users to spend a fixed amount of time in the critical section; next, we show how to mask the high level activity to low level users.

Hence, to tackle the first point, we differently model the time out as:

$$\begin{aligned} Timer \; &= \; go.\overline{stop}.Timer \\ C_3 \; &= \; req_H.\overline{go}.(stop.C_3 + \overline{read_H}.stop.C_3) \\ &\qquad + req_L.write_L.C_3 \\ Manager \; &= \; (Timer \| C_3)\backslash\{go, stop\} \end{aligned}$$

We added a process $Timer$ which is indeed an abstraction of a timer: When a high level request is accepted, the timer is started by means of an action go and then the read must be performed before the $stop$ action is issued. The $Manager$ process can be proved to be *BNDC* (in particular we can check through the CoSeC tool [6] that it is *SBSNNI* thus *BNDC*). Furthermore we can prove that $Manager$ is weak bisimilar to C_1.

However it is certainly important to see if it still generates timing covert channels. As before, we refine the specification as follows (we set the timer to one unit of time):

$$\begin{array}{rcl} Timer_t & = & \iota(go.tick.\iota(\overline{stop}.Timer)) \\ C_t & = & req_H.tick.\overline{go}.(\iota(stop.C_t) \\ & & +\ \iota(\overline{read_H}.\iota(stop.C_t))) \\ & & +\ req_L.tick.write_L.tick.C_t \\ Manager_t & = & (Timer_t \| C_t)\backslash\{go, stop\} \end{array}$$

Now, high users must spend exactly one unit of time in the critical section and then they must leave it. Indeed, the manager in parallel with X_1 has no more the computation $\stackrel{tick}{\Longrightarrow}\stackrel{req_L}{\Longrightarrow}$. But it still have the following one:

$$(Manager_t \| X_1)\backslash H \stackrel{tick}{\Longrightarrow}\stackrel{tick}{\Longrightarrow}\stackrel{req_L}{\Longrightarrow}$$

As a matter of fact, this sequence cannot be performed by the process $Manager_t\backslash H$. Thus, we find that this system is not $tBNDC$. It just represents the fact that time elapses when a high level access is performed. This is clearly observable by low level users that have to wait until the high level request is terminated before obtaining access to the variable.

This represents a typical situation that can be exploited to build a timing covert channel, where the unavailability of a shared resource for a certain amount of time (in this example, the amount at which the timer is set, i.e., one unit) can be encoded as 0 or 1 in order to transmit data.

We need to mask the high level activity to low level users. A solution to this problem could be to split the execution time of the $Manager_t$ process in two slots of fixed length. Then, in one slot only the high level users are served and, similarly, in the other one only the low level accesses are performed. This makes it impossible to discover if a high level user is doing something since the same fixed amount of time is dedicated to high users, even if they are doing nothing.

Here, for simplicity, we consider an abstraction of this mechanism. We just introduce a busy-waiting cycle which non deterministically stops the process $Manager_t$ for exactly two ticks. This is an abstraction of the possibility of allocating a slot to high users which, however, will do nothing in such slot. Now, the low level users cannot distinguish between high level accesses and these busy-waiting cycles, thus deducing nothing about high level activity.

The new system is then specified as follows:

$$\begin{array}{rcl} Timer & = & \iota(go.tick.\iota(\overline{stop}.Timer)) \\ C' & = & req_H.tick.\overline{go}.(\iota(stop.C') \\ & & +\ \iota(\overline{read_H}.\iota(stop.C'))) \\ & & +\ req_L.tick.write_L.tick.C' \\ & & +\ \tau.tick.tick.C' \\ Manager'_t & = & (Timer \| C')\backslash\{go, stop\} \end{array}$$

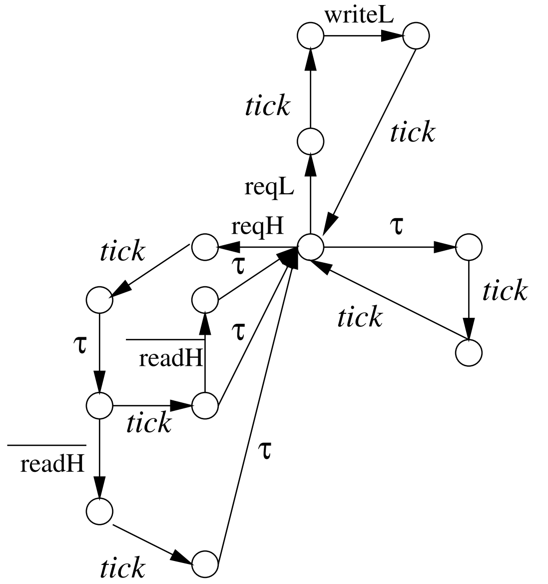

Figure 2. The behaviour of the process $Manager'_t$, represented by a Labeled Transition System.

where a branch $\tau.tick.tick.C'$ has been added in C (now C').

Remark 1 *It is worthwhile noticing that, in principle, we could mask the high level activity in the system C_2 as we did in the system $Manager'_t$. As a matter of fact, by adding two cycles, i.e. $\tau.tick.C_2$ and $\tau.tick.tick.C_2$ to C_2 we obtain that the resulting system is $tBNDC$. However, this solution is less than optimal when we consider a time-out with hundred of time units. Furthermore, we feel that process $Manager'_t$ is a more realistic implementation.*

We can directly prove that $Manager'_t \in tBNDC$. The idea is to build a timed weak bisimulation that contains the infinite number of pairs:

$$(Manager'_t \| \Pi\backslash H, Manager'_t\backslash H)$$

for every high user $\Pi \in \mathcal{E}^t_H$. This is one of the advantages of considering timed (weak) bisimulation as equivalence relation between our systems. Indeed, this kind of equivalence has a nice proof technique: We can prove that two processes are equivalent by simply providing a bisimulation relation that contains them.

In our specific case, this is very interesting since at the same time we are able to prove an infinite number of equivalences (like $BNDC$ membership requires)!

Consider Figure 3: The dashed lines represent pairs of a function f from derivatives of $Manager'_t$ to deriva-

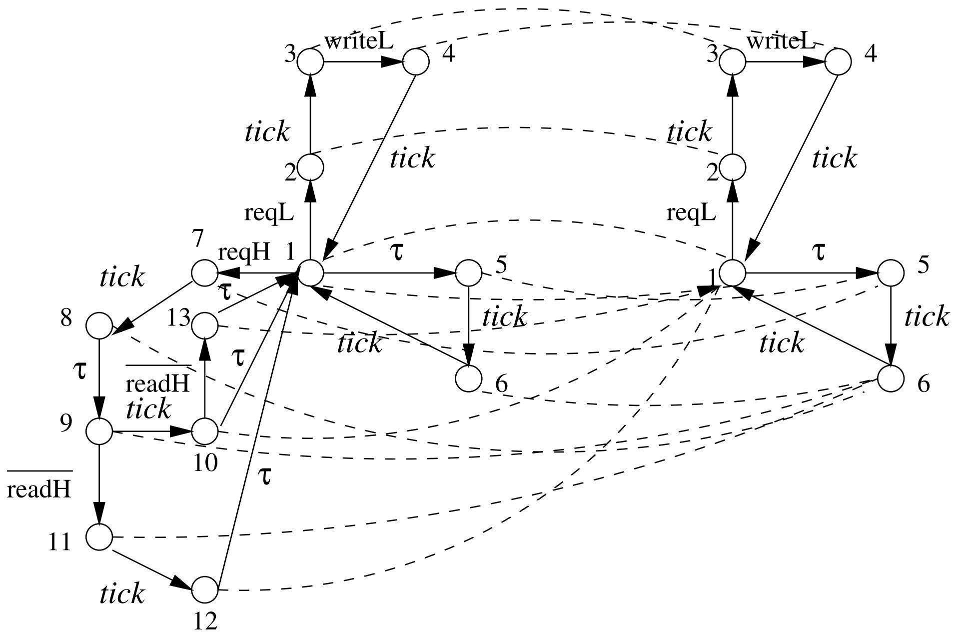

Figure 3. A graphical explanation of $tBNDC$ membership of $Manager'_t$.

tives of $Manager'_t\backslash H$. For the sake of simplicity, in Figure 3 we use numbers to represent derivatives instead of terms. Hence we have the following relation:

$$\begin{aligned}\mathcal{R} \;=\; & \{(E'\|\Pi\backslash H, E'') \mid (E', E'') \in f, \\ & \Pi \in \mathcal{E}^t_H, E' \in Der(Manager'_t)\}.\end{aligned}$$

The proof that $\mathcal{R}$ is a timed weak bisimulation follows by inspection of the possible cases. As example, we show that if $(9\|\Pi\backslash H, 6) \in \mathcal{R}$ we have the following possibilities to consider:

- if $9\|\Pi\backslash H \xrightarrow{tick} 10\|\Pi'\backslash H$, hence $\Pi \xrightarrow{tick} \Pi'$ and $\Pi \stackrel{\overline{read_H}}{\not\longrightarrow}$. In this case $6 \xrightarrow{tick} 1$ and $(10\|\Pi'\backslash H, 1) \in \mathcal{R}$.
- if $9\|\Pi\backslash H \xrightarrow{\tau} 11\|\Pi'\backslash H$, then $\Pi \xrightarrow{read_H} \Pi'$, but since $(11\|\Pi'\backslash H, 6) \in \mathcal{R}$ we have completed.
- if $9\|\Pi\backslash H \xrightarrow{\tau} 9\|\Pi'\backslash H$, then $\Pi \xrightarrow{\tau} \Pi'$ but also in this case we have $(9\|\Pi'\backslash H, 6) \in \mathcal{R}$.
- if $6 \xrightarrow{tick} 1$, we have two possibilities, if $\Pi \xrightarrow{read_H} \Pi'$ then $9\|\Pi\backslash H \xrightarrow{\tau} 11\|\Pi'\backslash H$. Now, since Π' must be weakly alive, we have $11\|\Pi'\backslash H \stackrel{\tau}{\Longrightarrow} 11\|\Pi'_1 \xrightarrow{tick} 12\|\Pi''\backslash H$, and $(12\|\Pi''\backslash H, 1) \in \mathcal{R}$. The other possibility is that $\Pi \stackrel{read_H}{\not\longrightarrow}$ and also in this case since Π must be weakly alive, we have $9\|\Pi\backslash H \stackrel{\tau}{\Longrightarrow} 9\|\Pi' \xrightarrow{tick} 10\|\Pi''\backslash H$ and $(10\|\Pi''\backslash H, 1) \in \mathcal{R}$.

5 Partial model checking

In [15, 16, 17, 18] a generalization of the NDC idea for the definition of security properties has been proposed and studied within a logical framework. The idea is to express security properties as follows:

$$\forall \quad \text{hostile environment } \Pi \qquad S \mid \Pi \models \phi \qquad (4)$$

where S is the system under investigation, $\mid$ is a parallel composition operator, ϕ is a logical formula and $\models$ is the truth relation. (Essentially we write $E \models \phi$ whenever the process E satisfies the property denoted by the formula ϕ.) Thus, if we assume that ϕ expresses a desired property (e.g. the system has no leaks of secrets, etc..), then the above property ensures that this property is granted by the system against every possible enemy (intruder, environment etc..), that may try to interfere, modify or capture some information from the system S.

For analyzing similar properties we still have the problem of the universal quantification on every hostile enemy. But, we noted that the above problem is somewhat similar to well studied problems in logic, i.e. validity problems. In these problems, we study whether every model satisfies a certain formula. Actually, the situation is slightly different, since we quantify over an unbounded number of processes, but we check if the composition of these processes together a process S satisfies a certain formula.

The key idea applied in [16] to tackle verification problems like (4) is to develop and suitably apply partial model checking techniques [1] in order to reduce them to validity problems. These techniques permit us to project the property that a compound system must satisfy into a property that one of the two components must enjoy. In our case we can indeed check if the property obtained by projecting the property ϕ on the second process is satisfied by every hostile process.

By using this approach, it has been possible to analyze $BNDC$-like properties [18], secrecy [17] and authentication (see [15] where a prototype tool is also presented).

In this section we show the flexibility of the methodology proposed in [16] by applying similar steps to the analysis of NDC-like properties in a real-time setting. We need to tackle the following points:

1. *Reducing tBNDC-like properties to properties in the form (4).* This point can be solved by resorting to the concept of characteristic formula for the timed equivalence. A formula ϕ is characteristic for a process P and a relation $\mathcal{R}$ whenever $P\mathcal{R}Q$ iff $Q \models \phi$. Hence, it is possible to express relationships between processes in a logical way (for certain relationships). Furthermore, we can note that our "timed equivalence" is similar observational equivalence, and hence we can use the characteristic formula for this equivalence already studied in [24, 21].

2. *Reducing properties of the form (4) to validity problems.* This point is solved by using the results in [16] which extend the theory of partial model checking [1] to deal also with operators whose semantics rules use negative premises (e.g., the timed parallel operator).

3. *Studying the decidability of the validity problem for the formulas obtained after the partial model checking with respect to specific classes of models.* This is the most difficult step. The models for the formulas of our logic are actually LTSs. But, we have that some (finite-state) LTS is not the semantics of a $tSPA$ process. (This does not happen with SPA processes.) We overcame this problem by considering a bigger set of High users, and hence by analyzing slightly stronger properties than $tBNDC$.

In the following, we introduce the logical language for expressing the properties of the systems, the partial model checking techniques for real time systems and their use in the analysis of *non interference* properties.

5.1 Equational $\mu-$calculus.

As a logic for expressing properties of processes we use the equational $\mu-$calculus, a variant of $\mu-$calculus, which is very suitable for partial model checking (see [1]).

Let a be in Act and X be a variable ranging over a finite set of variables $Vars$. Equational $\mu-$calculus is based on fixpoint equations that permit to define recursively the properties of systems. A minimal (maximal) fixpoint equation is $X =_\mu A$ ($X =_\nu A$), where A is an assertion, i.e. a simple modal formula without recursion operators. The syntax of the assertions (A) and of the lists of (D) equations is given by the following grammar:

$$\begin{array}{ll} A := & \mid X \mid \mathbf{T} \mid \mathbf{F} \mid X_1 \wedge X_2 \mid X_1 \vee X_2 \mid \\ & \mid \langle a \rangle X \mid [a]X \\ D ::= & X =_\nu \ A\ D \ \mid X =_\mu \ A\ D \ \mid \ \epsilon. \end{array}$$

Roughly, the semantics $[\![D]\!]$ of the list of equations D is the solution of the system of equations corresponding to D. [2] This solution is basically a function which assigns a set of processes to each variable on the left-hand of the equations in D. We denote the value of the variable X with $[\![D]\!](X)$. We will write $P \models D \downarrow X$ as a notation for $P \in [\![D]\!](X)$.

We give only an intuitive idea of the operators of this logic. For example we may expresses the fact that a process P is able to perform an a action, by requiring that $P \models \langle a \rangle \mathbf{T}$ (possibility operator). We can also require that after performing an action a the process P must satisfy a property ψ, by $P \models [a]\psi$ (necessity operators). The equation $Y =_\mu \langle a \rangle Y \vee \phi$ expresses the fact that through a finite sequence of actions a it is possible to reach a state which enjoys ϕ. The equation $Y =_\nu [a]Y \wedge \phi$ asserts that in every state reachable by means of a sequence of a actions the formula ϕ holds.

5.2 Partial model checking for the timed parallel operator.

Partial model checking technique has been developed for the compositional analysis of concurrent systems (processes). (Here we follow [1].)

The intuitive idea is the following: Suppose to have a system $P\|Q$ which is the parallel composition of two processes P, Q, and we wonder whether this process satisfies a formula ϕ or not. Now, it is possible to find a reduced formula $\phi_{//P}$ s.t.

$$P\|Q \models \phi \text{ iff } Q \models \phi_{//P}$$

[2] See Appendix or [1] for the formal semantics of the equational $\mu-$calculus.

$$
\begin{array}{rcl}
(D\!\downarrow X)//t & = & (D//t)\!\downarrow X_t \\
\epsilon//t & = & \epsilon \\
(X =_\sigma AD)//t & = & ((X_s =_\sigma A//s)_{s\in Der(E)})(D)//t \\
X//t & = & X_t \\
\langle a\rangle A//s & = & \langle a\rangle(A//s) \vee \bigvee_{s\xrightarrow{a}s'} A//s', \quad \text{if } a\neq\tau, a\neq tick \\
\langle\tau\rangle A//s & = & \langle\tau\rangle(A//s) \vee \bigvee_{s\xrightarrow{\tau}s'} A//s' \vee \bigvee_{s\xrightarrow{a}s'} \langle\overline{a}\rangle(A//s') \\
\langle tick\rangle A//s & = & \begin{cases} \langle tick\rangle A//s' \wedge \wedge_{a:s\xrightarrow{a}}[\overline{a}]\mathbf{F} & s\xrightarrow{tick}s' \\ \mathbf{F} & otherwise \end{cases} \\
[a]A//s & = & [a](A//\mathrm{s}) \wedge \bigwedge_{s\xrightarrow{a}s'} A//s', \quad \text{if } a\neq\tau, a\neq tick \\
[\tau]A//s & = & [\tau](A//\mathrm{s}) \wedge \bigwedge_{s\xrightarrow{\tau}s'} A//s' \wedge \bigwedge_{s\xrightarrow{a}s'} [\overline{a}](A\,//s') \\
[tick]A//s & = & \begin{cases} [tick]A//s' \vee \vee_{a:s\xrightarrow{a}}\langle\overline{a}\rangle\mathbf{T} & s\xrightarrow{tick}s' \\ \mathbf{T} & otherwise \end{cases} \\
A_1\wedge A_2//s & = & (A_1//s)\wedge(A_2//s) \\
A_1\vee A_2//s & = & (A_1//s)\vee(A_2//s) \\
\mathbf{T}//s & = & \mathbf{T} \\
\mathbf{F}//s & = & \mathbf{F}
\end{array}
$$

Table 1. Partial evaluation function for parallel operator $\|$ of timed SPA.

In this way, we have reduced the previous problem to a hopefully smaller one. In general, these techniques are applied when both P and Q are defined. In our case, the second process, which represents the attacker is not effectively defined. Hence, by means of this technique we can project the property that must be verified by the composition of two processes in a property that must be verified by only one. In our case, given a system we can find the property that the attackers must satisfy in order to perform a successful attack on the system.

It is very interesting to point out that partial model checking functions can be inferred automatically from the operational semantics rules of the language, when this is represented as SOS rules (see [1]) and the logical language is the equational $\mu-$calculus.

Unfortunately, *tSPA* semantics contains also rules with negative premises (e.g., parallel composition), and those techniques cannot be directly applied.

But in [16] a slight extension to the theory of Andersen has been proposed in such a way that is also possible to deal with semantics rules with negative premises (actually with a subset of GSOS rules). Thus, by applying the results of [16], we can automatically derive the partial evaluation function $D//E$ for the timed parallel operator and the equational $\mu-$calculus. This is given in Tab. 1, where E is a *tSPA* process and D an equational definition. Thus, we can state:

Lemma 2 *Given two processes E_1, E_2 (E_1 finite-state) and an equational specification $D\!\downarrow\!X$ we have:*

$$E_1\|E_2 \models (D\!\downarrow X) \text{ iff } E_2 \models (D\!\downarrow X)//E_1. \qquad \blacksquare$$

Example 3 *Let us see an example of partial evaluation. Consider the process $P = \iota(h.\mathbf{0})$. We want to find the necessary and sufficient conditions on a process X s.t.*

$$P\|X \models \langle tick\rangle\mathbf{T}.$$

By applying the rules in Table 1, we find out that $\langle tick\rangle\mathbf{T}//P$ is equal to $(\langle tick\rangle\mathbf{T}) \wedge [\overline{h}]\mathbf{F}$. Roughly, it requires that X is able to do a tick action, but no $\overline{h}$ actions. Indeed, for the maximal progress assumption*, the tick action is not executable if X can communicate with P through h.* $\blacksquare$

5.3 Analyzing NDC-like properties.

First of all we can parameterize the definition of *Non Deducibility on Composition* w.r.t. the set S of *high* users that are composed with the system when it is checked (as done in [18]). In this way we obtain more flexibility for expressing security properties.

Definition 7 $E \in tBNDC^S$ *if and only if* $\forall\Pi \in S$: $(E\|\Pi)\backslash H \approx_t E\backslash H$. $\blacksquare$

Under certain constraints on the set S we can provide a method for reducing the verification of $BNDC^S$ membership to a validity problem in equational $\mu-$calculus.

Informally, the solution is as follows. By using the characteristic formula for $\approx_t$ of $E\backslash H$ (see Appendix),

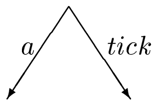

Figure 4. Example of LTS that does not corresponds to the semantics of a process E.

we obtain the following characterization: [3]

$$E \in tBNDC^S \quad \text{iff} \quad \forall \Pi \in S : (E \| \Pi) \backslash H \models \phi_{\approx_t, E\backslash H}$$

Now, we can apply the partial evaluation techniques: First, for the restriction operator (see [1]) and then for the timed parallel operator (see Tab. 1) to the formula $\phi_{\approx_t, E\backslash H}$ by getting the formula ϕ'. Then the previous problem is equivalent to check that every process in S satisfies ϕ'. Indeed, the behaviour of E has been evaluated and encoded in the formula ϕ'. Thus:

$$E \in tBNDC^S \quad \text{iff} \quad \forall \Pi \in S : \Pi \models \phi'$$

We expect to have decidability results only if we restrict ourselves to finite-state systems. Let $fs = \{E | Der(E) \text{ is finite}\}$ be the set of *finite state* processes. We also require that the set $\mathcal{L}$ of visible actions is finite.

If the membership in S can be defined by a formula ϕ'' then we obtain that the previous problem is equivalent to

$$\forall \Pi : \quad \Pi \models \phi'' \implies \phi'$$

Unfortunately, if we consider S as $\mathcal{E}^t_H$ then it is not so easy to restrict to consider only LTSs which are the semantics of processes in $\mathcal{E}^t_H$.

These LTSs enjoy several properties that could be handled, in particular *time determinacy* and maximal progress. In fact, we can use $tick-$deterministic equational $\mu-$calculus, in order to deal with the transitions systems denoted by the term of the *tSPA*, instead of simple $\mu-$calculus (as we did in [18]). The validity problem for this logic is decidable (see [26]). Moreover we can express with a formula the maximal progress property (since we consider a finite set of visible actions). But, the semantics for the choice operator imposes a sort of time persistency that seems not easily characterizable. As an example, it seems not possible to find a process E s.t. its associated LTS is the one of Figure 4. We avoid this problem by considering the choice operator of high users as the standard CCS one. (This is obtained by dropping the third rule for choice in Figure 1, and by considering a ranging over Act.) Then it is easy to see that for every finite-state LTS we can find a high user whose semantics is this LTS. Hence we consider properties that are stronger than $tBNDC$.[4]

We call fs^* the set of finite-state processes, which are deterministic, enjoy the maximal progress assumption, are weakly time alive and whose choice operator has the new semantics. (Actually, we still suppose to restrict ourselves to consider $tick$-deterministic processes, even though with this choice operator is possible to generate LTS which does not enjoy this property.)

We can prove the decidability of $tBNDC$-like properties when we only consider finite state processes fs^*.

Proposition 7 *Property tBNDC$^{fs^*}$ is decidable for all finite state processes E.* ■

5.4 An example

Now, we show the flexibility of the specification format (4) and of the partial model checking techniques we developed. In certain situations, we could be interested to specify and analyze also weaker properties than $tBNDC$. For example, we could be interested to show that a system E composed with a set of high processes S simply does not present deadlocks, or else that is always able to produce a certain action. Let us re-consider the process $E = l.\mathbf{0} + h.h.l.\mathbf{0}$. Actions l and h could represent the allocation of the same shared resource by a low level user and a high level user, respectively. Assume also that the second execution of h or l corresponds to the deallocation of such a resource. For example, $h.h.l$ represents the fact that after the resource has been allocated and deallocated by an high level user, then it can be reallocated by a low level one. Consider now the following equational definition D (please note that Y is a variable here):

$$Y =_\nu [\tau]Y \wedge Z, \quad Z =_\mu \langle\tau\rangle Z \vee \langle l \rangle \mathbf{T}$$

It asserts that a process, whatever state it reaches by means of τ actions, is always able to move (through a finite sequence of τ actions) into a state where an action l can be performed. In other words, it requires that the resource is always allocable (after some internal transitions) by low level users. If we want that this is true even in the presence of some hostile processes we can just require that $\forall \Pi \in S (E \| \Pi) \setminus H \models D$, where $H = \{\overline{h}, h\}$. As for the study of $tBNDC$-like properties we can apply partial evaluation for both the parallel and restriction operators and (after several trivial simplifications,

[3] Actually, this is true only if we consider finite-state processes. The same holds for weak bisimulation in CCS.

[4] However, we feel that is possible to suitably change the validity procedure of [26] in order to consider only LTSs which are the semantics of some *tSPA* term. We leave the proof of this conjecture as a future work.

see the full paper [7]) we obtain that $D//\setminus H//E$ is:

$$\begin{array}{rcl} Y_E & =_\nu & [\tau]Y_E \wedge [\overline{h}]Y_{h.l.\mathbf{0}} \\ Y_{h.l.\mathbf{0}} & =_\nu & [\tau]Y_{h.l.\mathbf{0}} \wedge Z_{h.l.\mathbf{0}} \\ Z_{h.l.\mathbf{0}} & =_\mu & \langle\tau\rangle Z_{h.l.\mathbf{0}} \vee \langle\overline{h}\rangle\mathbf{T} \end{array}$$

which, roughly, expresses that after performing a τ action, the system reaches a configuration s.t. if it performs a $\overline{h}$ action then it is able to perform a finite sequence of τ followed by $\overline{h}$. It basically requires that when an high level user allocates a resource, then it must eventually deallocate it. As a matter of fact, consider the set of processes S which after performing an action $\overline{h}$ are obliged to issue another action $\overline{h}$. Then, it is easy to see (and can be formally proved, e.g. by using the proof systems of [16]) that these processes satisfy the equational specification $D//\setminus H//E$.

This requirement, obtained through partial model checking, allows us to find a way of "correcting" bad (or insecure) high level behaviours. For example, we could impose the following execution policy for high users: Every high user, when executed in the system E, is equipped with a preemptive process $C = Top^{tick}_{\overline{h}}$, which is able to deallocate resources previously allocated by high level users. In order to see if this process is able to force a correct behaviour of high level user it is now sufficient to check that

$$\forall \Pi \qquad (E\|(C\|\Pi))\setminus H \models D \tag{5}$$

Now, we already know that processes $(C\|\Pi)$ must enjoy properties $D//\setminus H//E$. Thus, we may work in a compositional way by simply building the reduced formula $(D//\setminus H//E)//C$ which expresses the properties the processes Π must enjoy in order to have (5). Now, it is easy to see that $(D//\setminus H//E)//C = \mathbf{T}$. This means that if the high users Π are encapsulated together with the process C, the low level users will always be able to execute l, i.e., to allocate the resource. In other words no (potentially dangerous) deadlock can be caused by encapsulated high users.

6 Conclusions and future research

We have shown the flexibility of the theory of [5, 6, 18] by extending it to a discrete time process algebra. Even if the real-time setting is rather simple, we think that similar work can be done also for other, more sophisticated approaches to real-time. Moreover, other aspects of system behaviour can be included in the specification, e.g., probability (see [12, 13]). We plan to extend the approach also to a probabilistic process algebra and then to stochastic, as well as real-time probabilistic process algebras. We expect that $BNDC$ can be formulated also in these richer settings with little effort. The reason of considering more concrete models is that they give a more detailed, closer to the implementation, description of a system, that may then reveal indirect information flows that are not possible in more abstract specifications of the same system. We think that the results reported here are encouraging and justify our expectations.

The idea of $BNDC$ has been already successfully applied to the verification and analysis of cryptographic protocols [3, 6, 8, 9, 10]. We plan to extend such techniques using the real-time model proposed in this paper. This should allow us to easily capture also time-dependent protocol properties.

Acknowledgements

We would like to thanks the anonymous referees for their helpful comments and suggestions.

References

[1] H. R. Andersen. Partial model checking (extended abstract). In *Proc. of LICS'95*, pages 398–407. IEEE Computer Society Press, 1995.

[2] F. Corradini, D. D'Ortenzio, and P. Inverardi. On the relationships among four timed process algebras. *Fundamenta Informaticae*, 34:1–19, 1999.

[3] A. Durante, R. Focardi, and R. Gorrieri. CVS: A compiler for the analysis of cryptographic protocols. In *Proceedings of CSFW'99*, pages 203–212. IEEE press, 1999.

[4] R. Focardi. *Analysis and Automatic Detection of Information Flows in Systems and Networks*. PhD thesis, Department of Computer Science, University of Bologna, 1998.

[5] R. Focardi and R. Gorrieri. A classification of security properties. *Journal of Computer Security*, 3(1):5–33, 1995.

[6] R. Focardi and R. Gorrieri. The compositional security checker: A tool for the verification of information flow security properties. *IEEE Trans. Sof. Eng.*, 27:550–571, 1997.

[7] R. Focardi, R. Gorrieri, and F. Martinelli. Information flow analysis in a discrete-time process algebra. Technical report, Istituto per le Applicazioni Telematiche, 2000.

[8] R. Focardi, R. Gorrieri, and F. Martinelli. Message authentication through non interference. To appear in proceedings of International Conference on Algebraic Methodology And Software Technology (AMAST 2000), May 2000.

[9] R. Focardi, R. Gorrieri, and F. Martinelli. Secrecy in security protocols as non interference. In *Proceedings of DERA/RHUL Workshop on Secure Architectures and Information Flow (S. Schneider and P. Ryan ed.)*. Elsevier ENTCS, Volume 32, 2000.

[10] R. Focardi and F. Martinelli. A uniform approach for the definition of security properties. In *Proceedings of World Congress on Formal Methods (FM'99)*, pages 794–813. Springer, LNCS 1708, 1999.

[11] J. Goguen and J. Meseguer. Security policy and security models. In *Proc. of the 1982 Symposium on Security and Privacy*, pages 11–20. IEEE Computer Society Press, 1982.

[12] J. W. Gray, III. Toward a mathematical foundation for information flow security. In *Proceedings of the 1991 IEEE Computer Society Symposium on Security and Privacy*, pages 21–34. IEEE Computer Society Press, 1991.

[13] J. W. Gray, III and P. F. Syverson. A logical approach to multilevel security of probabilistic systems. In *Proceedings of the 1992 IEEE Computer Society Symposium on Security and Privacy*, pages 164–176. IEEE Computer Society Press, 1992.

[14] M. Hennessy and T. Regan. A temporal process algebra. *Information and Computation*, 117:221–239, 1995.

[15] D. Marchignoli and F. Martinelli. Automatic verification of cryptographic protocols through compositional analysis techniques. In *Proceedings of the International Conference on Tools and Algorithms for the Construction and the Analysis of Systems*, volume 1576 of *LNCS*, 1999.

[16] F. Martinelli. *Formal Methods for the Analysis of Open Systems with Applications to Security Properties*. PhD thesis, University of Siena, Dec. 1998.

[17] F. Martinelli. Languages for description and analysis of authentication protocols. In *Proceedings of 6th Italian Conference on Theoretical Computer Science*. World Scientific, 1998.

[18] F. Martinelli. Partial model checking and theorem proving for ensuring security properties. In *Proc. of CSFW'98*, pages 44–52. IEEE Computer Society Press, 1998.

[19] R. Milner. *Communication and Concurrency*. Prentice Hall, 1989.

[20] R. Milner. Operational and algebraic semantics of concurrent processes. In J. van Leewen, editor, *Handbook of Theoretical Computer Science*, volume B: Formal Models and Semantics, chapter 19, pages 1201–1242. The MIT Press, New York, N.Y., 1990.

[21] M. Muller-Olm. Derivation of characteristic formulae. *Electronic Notes in Theoretical Computer Science*, 18, 1998.

[22] X. Nicollin and J. Sifakis. An overview and synthesis on timed process algebras. In J. W. de Bakker, C. Huizing, W. P. de Roever, and G. Rozenberg, editors, *Proceedings of Real-Time: Theory in Practice*, volume 600 of *Lecture Notes in Computer Science*, pages 526–548, 1992.

[23] S. Schneider. *Concurrent and Real-time Systems: the CSP Approach*. J. Wiley & sons, Inc., 1999.

[24] B. Steffen and A. Ingólfsdóttir. Characteristic formulae for processes with divergence. *Information and Computation*, 110(1):149–163, 1994.

[25] C. Stirling. Modal and temporal logics for processes. In *Logics for Concurrency: Structures versus Automata*, volume 1043 of *LNCS*, pages 149–237, 1996.

[26] R. S. Street and E. A. Emerson. The propositional $\mu-$calculus is elementary. In *Eleventh International Colloquium in Automata, Languages and Programming*, volume 172 of *LNCS*, pages 465–472, 1984.

[27] R. S. Street and E. A. Emerson. An automata theoretic procedure for the propositional $\mu-$calculus. *Information and Computation*, 81(3):249–264, 1989.

[28] I. Ulidowski and S. Yuen. Extending process languages with time. In *Proceedings of the 6th International Conference on Algebraic Methodology and Software Technology*, volume 1349 of *Lecture Notes in Computer Science*, pages 524–538, 1997.

A Appendix

A.1 Semantics of equational $\mu-$calculus

In this section we give the formal semantics of equational $\mu-$calculus (see [1]).

It is assumed that variables appear only once on the left-hand sides of the equations of the list, the set of these variables will be denoted as $Def(D)$. Let $\langle S, A, \{\xrightarrow{a}\}_{a\in A}\rangle$ be an LTS extended with ρ an environment that assigns subsets of S to the variables that appear in the assertions of D, but which are not in $Def(D)$.

The semantics $[\![A]\!]'_\rho$ of an assertion A is the following:

$$\begin{aligned}
[\![T]\!]'_\rho &= S \\
[\![F]\!]'_\rho &= \emptyset \\
[\![X]\!]'_\rho &= \rho(X) \\
[\![A_1 \wedge A_2]\!]'_\rho &= [\![A_1]\!]'_\rho \cap [\![A_2]\!]'_\rho \\
[\![A_1 \vee A_2]\!]'_\rho &= [\![A_1]\!]'_\rho \cup [\![A_2]\!]'_\rho \\
[\![\langle a\rangle A]\!]'_\rho &= \{s \mid \exists s' : s \xrightarrow{a} s' \text{ and } s' \in [\![A]\!]'_\rho\} \\
[\![[a]A]\!]'_\rho &= \{s \mid \forall s' : s \xrightarrow{a} s' \text{ implies } s' \in [\![A]\!]'_\rho\}
\end{aligned}$$

The semantics of a list of equations $D, [\![D]\!]_\rho$ is an environment that assigns subsets of S to variables in $Def(D)$. A list of equations is closed if every variable that appears in the assertions of the list is in $Def(D)$. We use $\sqcup$ to represent union of disjoint environments. Let σ be in $\{\mu, \nu\}$, then $\sigma U.f(U)$ represents the σ fixpoint of the function f in one variable U.

$$\begin{aligned}
[\![\epsilon]\!]_\rho &= [] \\
[\![X =_\sigma AD']\!]_\rho &= [\![D']\!]_{(\rho\sqcup[U'\setminus X])} \sqcup [U'/X]
\end{aligned}$$

where

$$U' = \sigma U.[\![A]\!]'_{(\rho\sqcup[U/X]\sqcup\rho'(U))}, \rho'(U) = [\![D']\!]_{(\rho\sqcup[U/X])}$$

It informally says that *the solution to* $(X =_\sigma A)D$ *is the* σ *fixpoint solution* U' *of* $[\![A]\!]$ *where the solution to the rest of the list of equations* D *is used as environment*.

A.2 Characteristic formula

Given a finite state process E, we present below the definition of a formula ϕ_E that is characteristic (w.r.t. timed *weak* bisimulation) for this process E (see [21]). Let $\langle\langle\tau\rangle\rangle\phi$ be a short notation for $\mu X.\langle\tau\rangle X \vee \phi$, $[[\tau]]\phi$ for $\nu X.[\tau]X \wedge \phi$, where X is not free in ϕ. (The semantics of μ and ν can be found in [27].) Let $\langle\langle l\rangle\rangle\phi, [[l]]\phi, (l \in \mathcal{L} \cup \{tick\})$ be respectively $\langle\langle\tau\rangle\rangle\langle l\rangle\langle\langle\tau\rangle\rangle\phi, [[\tau]][l][[\tau]]\phi$. It can be shown that these derived modalities can be equivalently expressed in an equational form. Let us see the definition of the characteristic formula:

Definition 8 *Given a finite state process E, its characteristic formula (with respect to timed weak bisimulation) $D_E \downarrow X_E$ is defined by the following equations for every $E' \in Der(E)$, $a \in Act$:*

$$X_{E'} =_\nu (\bigwedge_{a,E'':E' \xrightarrow{a} E''} \langle\langle a\rangle\rangle X_{E''}) \wedge$$
$$\wedge (\bigwedge_a ([a](\bigvee_{E'':E' \stackrel{a}{\Longrightarrow} E''} X_{E''}))) \qquad \blacksquare$$

Intuitively, for every state of the process there is a variable that encodes the capabilities of that state. Following [21, 24, 25] if ϕ_{E_2} is characteristic for E_2 (w.r.t. $\approx_t$) then:

Lemma 3

1. *If $E_1 \approx_t E_2$ then $E_1 \models \phi_{E_2}$.*
2. *If $E_1 \models \phi_{E_2}$ and E_1 is finite state then $E_1 \approx_t E_2$.* $\blacksquare$