Real-Time Information Flow Analysis

Riccardo Focardi, Roberto Gorrieri, and Fabio Martinelli

***Abstract*—In previous work, we have studied some noninterference properties for information flow analysis in computer systems on classic (possibilistic) labeled transition systems. In this paper, some of these properties, notably bisimulation-based nondeducibility on compositions (BNDC), are reformulated in a real-time setting. This is done by first enhancing the Security Process Algebra proposed by two of the authors with some extra constructs to model real-time systems (in a discrete time setting), and then by studying the natural extension of these 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 the approach through two case studies, where in both cases the untimed specification is secure, while the timed specification may show up interesting timing covert channels.**

***Index Terms*—Bisimulation, information flow security, partial mode checking, process algebra, real-time systems, timing covert channels.**

I. Introduction

INFORMATION flow analysis is considered one of the main techniques for studying confidentiality in computer systems. The aim is to prevent any possible flow from the confidential (*high*) level to the public (*low*) one. Intuitively, an information flow from high to low is present when the behavior of the low part of the system is affected by changes in the behavior of the high part of the system. Many different formalizations have been proposed in the literature *To capture this intuitive idea of information flow*. Some years ago, in [8] and [10], we have made an effort to classify most of them within the uniform framework of security process algebra (SPA), a CCS-like [27] calculus where actions are partitioned into two different levels of confidentiality. SPA can express only the *nondeterministic* (sometimes called possibilistic) behavior 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., logical covert channels.

As a result of the classification effort of [8]–[10], two of the authors proposed bisimulation-based nondeducibility on compositions (BNDC) as the most natural information flow property: a system E is BNDC if for every high-level (possibly hostile) user Π, the low view of the behavior of E is not modified by the presence of Π. Technically, BNDC requires that E in parallel with Π is bisimulation equivalent to E in isolation, from the low-level point-of-view. More recent research [12], [14] has shown the key role of BNDC as a general property suitable also for modeling and analyzing various security properties of cryptographic protocols. BNDC, as well as its related approximation properties bisimulation-based strong nondeterministic noninterference (BSSNI) and strong BSNNI (SBNNI)[1] are all noninterference like properties: checking the absence of information flow is reduced to verifying that the high-level activity may not interfere with the low-level behavior of the system.

In this paper, we extend the theory of [8]–[10], and related proof techniques studied in [23] and [24], 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. We will do this by:

- defining a real-time version of SPA, called timed SPA (tSPA), following one of the many approaches proposed in the literature to extend process algebras with real-time features (see, e.g., [5], [19], [29], [31], [37]), and then by
- adapting the noninterference properties of interest to this enriched setting.

Note that a more concrete specification might show up new information flows. Even if one starts from an untimed specification that shows no logical information flows, after having specified its timing requirements, some time dependent information flows may become possible, e.g., if a high user is able to modify the low-level response time of the system, this could be used as a code for transmitting information from the high level to the low one.

The first problem we have to face is the definition of tSPA. 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., [5], [19], and [37]. 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 with respect to 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 proceed asynchronously by performing durationless actions.

Another feature of tSPA is the so-called *maximal progress assumption* (e.g., see [19], [37]) according to which *tick* actions have lower priority with respect to internal τ actions: communications and internal τ moves prevent time from occurring,

Manuscript received April 4, 2002; revised August 20, 2002. This work was supported in part by MIUR project "Metodi Formali per la Sicurezza" (MEFISTO), by MIUR project "Strumenti, ambienti ed applicazioni innovative per la società dell'informazione," by CSP with the project "SeTAPS," and in part by EU under Contract IST-2001-32617 "Models and Types for Security in Mobile Distributed Systems" (MyThS).

R. Focardi is with the Dipartimento di Informatica, Università Ca' Foscari di Venezia, I-30173 Mestre, Italy (e-mail: focardi@dsi.unive.it).

R. Gorrieri is with the Dipartimento di Scienze dell'Informazione, Università di Bologna, I-40127 Bologna, Italy (e-mail: gorrieri@cs.unibo.it).

F. Martinelli is with the Istituto di Informatica e Telematica, National Research Council (C.N.R.), Pisa, Italy (e-mail: Fabio.Martinelli@iit.cnr.it).

Digital Object Identifier 10.1109/JSAC.2002.806122

[1]The long acronyms BNDC, BSNNI and SBSNNI, certainly less than optimal, are kept here for homogeneity with the papers [8], [9] where they have been introduced.

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 $a.\mathrm{tick}.\mathrm{tick}.E'$, 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. The untimed noninterference like properties advocated in [8] and [9] are based on the notion of weak bisimulation. To adapt them to the new real-time setting, we only need to define a suitable *timed bisimulation* equivalence notion for tSPA, e.g., the definition of the timed version of BNDC (tBNDC) is derived by using timed bisimulation equivalence in place of weak bisimulation.

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. Two examples in Section IV illustrate 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 [23] and [24] for analyzing BNDC-like properties in a logical manner can be extended to cope with real-time tSPA processes. 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 II introduces tSPA and timed weak bisimulation. Section III is devoted to the noninterference properties tBNDC, tBSNNI, and tSBSNNI; in particular, we prove that $\mathrm{tSBSNNI} \subseteq \mathrm{tBNDC} \subseteq \mathrm{tBSNNI}$ and that tSBSNNI is compositional (while tBNDC is not). In Section IV, two case studies are presented. Section V extends the partial model checking technique of [23], [24] to tBNDC. Finally, some conclusive remarks are given in Section VI. The proofs of some propositions are postponed to the Appendix.

II. Simple Real-Time Model

In this section, we introduce the tSPA, a real-time extension of the SPA, that was used in [8], [9] for the description of security properties in an untimed setting. In SPA, it is only possible to express qualitative temporal orderings among events, while quantitative time aspects cannot be expressed. Thus, we extend the SPA language with operators that permit to express the elapsing of time, following the approach of [19].

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 $I = \{a, b, c \ldots, \}$ is the set of input actions and $O = \{\overline{a}, \overline{b}, \overline{c}, \ldots\}$ 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 $(\bar{\ }) : \mathcal{L} \mapsto \mathcal{L}$, 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. Let *tick* be the special action used to model time elapsing and let $\mathrm{Act} = \mathcal{L} \cup \{\tau\} \cup \{\mathrm{tick}\}$, ranged over by $\alpha, \beta, \ldots$, while $\mathcal{L} \cup \{\tau\}$ is, with abuse of notation, ranged over by $a, b, \ldots$. We now describe the syntax for tSPA terms

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

where $\alpha \in \mathrm{Act}$, $L \subseteq \mathcal{L}$ and Z is a process constant that must be associated with a definition $Z \doteq E$. As usual, we assume that constants are *guarded* [27], i.e., they must be in the scope of some prefix operator $\alpha.E'$. The set of tSPA processes, i.e., of terms with guarded constants, is denoted with Proc^t, ranged over by $E, F, P, Q, \ldots$ We will often use some usual syntactic simplifications, e.g., omission of trailing $\mathbf{0}$'s, as well as omission of brackets on restriction on a single action.

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 $\mathrm{tick}.E$ represents a process willing to let one time unit pass.
- $E_1 + E_2$ (*choice*) represents the nondeterministic choice between the two processes E_1 and E_2; when both are able to perform a *tick* action, then $E_1 + E_2$ can perform this action and reach a configuration where both summand derivatives can still be chosen.
- $E_1 \parallel E_2$ (*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 cannot be delayed.
- $E\backslash L$ (*restriction*) is the process E when actions in $L \cup \overline{L}$ are prevented.
- $\iota(E)$ (*idling*) allows process E to wait indefinitely. At every instant of time, if process E performs an action l, then the whole system proceeds in this state, while dropping the idling operator.

The formal semantics of tSPA processes is described by the *labeled transition system* (LTS) $\langle \mathrm{Proc}^t, \mathrm{Act}, \{\xrightarrow{\alpha}\}_{\alpha \in \mathrm{Act}} \rangle$, where $\xrightarrow{\alpha}_{\alpha \in \mathrm{Act}}$ is the least relation between tSPA processes induced by the axioms and the inference rules of Fig. 1. Such a relation is well-defined even if negative premises[2] occur in the rules for the parallel operator and for the idling operator, because the relation is *strictly stratifiable* [18].

Let us consider the following relations between tSPA terms: $E \stackrel{\tau}{\Longrightarrow} E'$ (or $E \stackrel{\tau}{\Longrightarrow} E'$) if $E \xrightarrow{\tau}{}^{*} E'$ (where $\xrightarrow{\tau}{}^{*}$ is the reflexive and transitive closure of the $\xrightarrow{\tau}$ relation); for $\alpha \neq \tau$, $E \stackrel{\alpha}{\Longrightarrow} E'$ if $E \stackrel{\tau}{\Longrightarrow} \xrightarrow{\alpha} \stackrel{\tau}{\Longrightarrow} E'$. Let $\mathrm{Der}(E)$ be the set of derivatives of E, i.e., the set of processes that can be reached through

[2] A negative premise is a premise of a rule of the form $E \not\xrightarrow{\alpha}$, meaning that there is no E' such that $E \xrightarrow{\alpha} E'$.

PREFIXING:

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

CHOICE:

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

PARALLEL:

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

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

CONSTANT:

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

RESTRICTION:

$$\frac{E_1 \xrightarrow{\alpha} E_1'}{E_1 \backslash L \xrightarrow{\alpha} E_1' \backslash L} (\alpha \notin L \cup \bar{L})$$

IDLING:

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

Fig. 1. Operational semantics for timed SPA.

the transition relations. Let $\mathrm{Sort}(E)$ be the set of actions in *Act* syntactically occurring in E.

First of all, we state two peculiarities of tSPA. The first one is *time determinacy* which requires that the time elapsing never moves a process to two different states (i.e., it is *tick*-deterministic); the second one is the *maximal progress* assumption, requiring that internal τ actions have precedence on the elapsing of time. The proof of the following lemma can be easily given by inspecting the operational rules. In particular, the first two rules of the idling operator are the key rules enforcing time determinacy, while the rule for *tick* synchronization and the first rule of the idling operator are the key rules enforcing maximal progress.

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

1) if $E \xrightarrow{\mathrm{tick}} E'$ and $E \xrightarrow{\mathrm{tick}} E''$ then $E' = E''$;
2) if $E \xrightarrow{\mathrm{tick}}$ then $E \not\xrightarrow{\tau}$. ■

Three simple examples follow to clarify some aspects of *tSPA*. The first one is a timed communication between two processes, the second one is a timeout construct and the third one shows two peculiar timed processes.

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

$$E = \overline{\mathrm{start}}.\overline{\mathrm{com}}.\mathbf{0} \quad F = \mathrm{com}.\overline{\mathrm{stop}}.\mathbf{0}$$
$$\mathrm{Sys} = (E \| F) \backslash \mathrm{com}.$$

We restrict the composition on the channel *com* in order to force the synchronization on this channel. In this way, we are not able to say how much time the communication between E and F takes. Let us model the same situation in tSPA. First of all, even though we assume that actions 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:

$$E_1 = \overline{\mathrm{start}}.\overline{\mathrm{com}}.\mathrm{tick}.\mathrm{tick}.\mathbf{0}$$
$$F_1 = \mathrm{com}.\mathrm{tick}.\mathrm{tick}.\overline{\mathrm{stop}}.\mathbf{0}$$
$$\mathrm{Sys}_1 = (E_1 \| F_1) \backslash \mathrm{com}.$$

In general, if we assume that action a takes n units of time we can replace every subterm $a.P'$ in a process P by $a.\mathrm{tick}^n.P'$, 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 perform only the following sequence of actions:

$$\mathrm{Sys}_1 \xrightarrow{\overline{\mathrm{start}}} \xrightarrow{\tau} \xrightarrow{\mathrm{tick}} \xrightarrow{\mathrm{tick}} \xrightarrow{\overline{\mathrm{stop}}}.$$

As expected the communication takes two time units. ■

Example 2: We can easily model timeout constructs in tSPA. Assume $n_1 \leq n_2$ and define the process

$$\mathrm{Time\_out}(n_1, n_2, A, B) = \mathrm{tick}^{n_1}.\iota(A) + \mathrm{tick}^{n_2}.\tau.B.$$

$\mathrm{Time\_out}(n_1, n_2, A, B)$ first performs a sequence of n_1 tick actions; then, the system may perform $n_2 - n_1$ tick actions, unless A resolves the choice by performing an action; instead if A does nothing, after n_2 time units, via the execution of a τ, the process is forced to act as B. ■

Example 3: In tSPA there are processes, such as $\mathbf{0}$, that do not allow time to proceed, i.e., they have no reachable *tick* labeled transitions. Process $\mathbf{0}$, in particular, is both functionally terminated and time blocked. Moreover, as the operational rule for parallel composition forces a global synchronization on *tick* actions, the effect of composing a process P with $\mathbf{0}$ is to prevent

P from letting time pass. In other words, $\mathbf{0}$ acts as a time annihilator for its parallel context. On the contrary, $\iota(\mathbf{0})$ is process that, even if functionally terminated, let time to proceed indefinitely. Hence, $\iota(\mathbf{0})$ acts as a neutral element for parallel composition, as it does not influence the possible executions of processes composed in parallel with it. ■

In the following, it is useful to consider the class of processes that do allow time to proceed, the so-called *weakly time alive* processes.

Definition 1: A process E is directly weakly time alive iff $E \stackrel{\mathrm{tick}}{\Longrightarrow}$, while it is weakly time alive iff for all $E' \in \mathrm{Der}(E)$, we have E' is directly weakly time alive. ■

Since $E \stackrel{\alpha}{\longrightarrow} E'$ implies $\mathrm{Der}(E') \subseteq \mathrm{Der}(E)$, it directly follows that if E is weakly time alive, then any derivate E' of E is weakly time alive as well. Moreover, it is worthwhile noticing that the above property is preserved by the parallel composition.

As any other process algebra, also tSPA comes equipped with suitable notions of behavioral equivalences, that equate processes if they are indistinguishable by external observers, according to some assumptions on the observational power of the observers. The behavioral relations we consider are the timed versions of trace equivalence and weak bisimulation [27], respectively. These equivalences permit, in a different way, to abstract to some extent from the internal behavior of the systems, represented by the invisible τ actions.[3]

Definition 2: Let $E, F \in \mathrm{Proc}^t$. The set of timed traces for E is $Tr(E) = \{\gamma \in (\mathcal{L} \cup \{\mathrm{tick}\})^* \mid \exists \mathrm{E}'.\mathrm{E} \stackrel{\gamma}{\Longrightarrow} \mathrm{E}'\}$. The timed trace preorder, denoted with $\leq_t$, is defined as follows: $E \leq_t F$ iff $Tr(E) \subseteq Tr(F)$. E and F are *trace equivalent*, denoted with $E \sim_t F$, if $Tr(E) = Tr(F)$. ■

Definition 3: A relation $\mathcal{R} \subseteq \mathrm{Proc}^t \times \mathrm{Proc}^t$ is a timed weak simulation iff for every $(E, F) \in \mathcal{R}$, we have

- if $E \stackrel{a}{\longrightarrow} E'$, then there exists F' s.t. $F \stackrel{a}{\Longrightarrow} F'$ and $(E', F') \in \mathcal{R}$;
- if $E \stackrel{\mathrm{tick}}{\longrightarrow} E'$, then there exists F' s.t. $F \stackrel{\mathrm{tick}}{\Longrightarrow} F'$ and $(E', F') \in \mathcal{R}$.

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. ■

As widely agreed upon in concurrency theory, trace equivalence is not always adequate for reactive systems, as it forgets about the points of choice and so it is not able to detect potential deadlocks. Moreover, bisimulation is equipped with a very powerful proof technique; indeed, in order to prove that two processes E, F are timed weakly bisimilar ($E \approx_t F$), we only need to provide a timed weak bisimulation that contains them. For instance, we can prove that process $\mathrm{Sys}'_1 = \overline{\mathrm{start}}.\mathrm{tick}.\mathrm{tick}.\overline{\mathrm{stop}}.\mathbf{0}$ is timed weakly bisimilar to Sys_1 by checking that the relation at the bottom of the page is a weak simulation for them. An interesting feature of $\approx_t$ is that given two finite-state tSPA processes, we can decide in polynomial time whether they belong to $\approx_t$ or not.

[3]Other equivalences are in between trace and bisimulation semantics. We do not intend to discuss here their relative merits. The interested reader can consult [15] for an overview (in the untimed setting).

III. Security Properties in a Real-Time Setting

In this section, we present some information flow security properties. In particular, we recast the *noninterference* theory proposed in [8], [9] 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 nondeducibility on composition (NDC). Its underlying idea is that the system behavior must be invariant with respect to 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:

$$\forall\, \Pi \in \text{High users}\ E \mid \Pi \equiv E \text{ with respect to Low users}$$

where $\mid$ stands for the composition operator and $\equiv$ for an equivalence relation. This property can be instantiated by assuming different notions of composition and equivalence. Let $\mathcal{E}_H = \{\Pi \mid \mathrm{Sort}(\Pi) \subseteq \mathrm{Act}H \cup \{\tau\}\}$ be the set of High users. In terms of SPA parallel composition operator and *bisimulation* equivalence [27] (denoted with $\approx$), we have the following (recall that H is a shorthand notation for $\mathrm{Act}H$).

Definition 4: $E \in \mathrm{BNDC}$ if and only if $\forall\, \Pi \in \mathcal{E}_H$, we have $(E \| \Pi) \backslash H \approx E \backslash H$. ■

It is worthwhile noticing that the underlying idea of NDC is very natural for describing security properties. In the last few years, a generalization of this idea (GNDC [14]) 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, nonrepudiation, etc.) can be easily encoded in this scheme [12]–[14]. This opens the way to apply tools for the analysis of nonInterference also to the analysis of security protocols [6], [9]. We will come back to this issue in the conclusions.

$$\{(\mathrm{Sys}, \mathrm{Sys}'_1),$$
$$((\overline{\mathrm{com}}.\mathrm{tick}.\mathrm{tick}.\mathbf{0} \| \mathrm{com}.\mathrm{tick}.\mathrm{tick}.\overline{\mathrm{stop}}.\mathbf{0}) \backslash \mathrm{com}, \mathrm{tick}.\mathrm{tick}.\overline{\mathrm{stop}}.\mathbf{0})$$
$$((\mathrm{tick}.\mathrm{tick}.\mathbf{0} \| \mathrm{tick}.\mathrm{tick}.\overline{\mathrm{stop}}.\mathbf{0}) \backslash \mathrm{com}, \mathrm{tick}.\mathrm{tick}.\overline{\mathrm{stop}}.\mathbf{0})$$
$$((\mathrm{tick}.\mathbf{0} \| \mathrm{tick}.\overline{\mathrm{stop}}.\mathbf{0}) \backslash \mathrm{com}, \mathrm{tick}.\overline{\mathrm{stop}}.\mathbf{0})$$
$$((\mathbf{0} \| \overline{\mathrm{stop}}.\mathbf{0}) \backslash \mathrm{com}, \overline{\mathrm{stop}}.\mathbf{0})$$
$$((\mathbf{0} \| \mathbf{0}) \backslash \mathrm{com}, \mathbf{0})\}.$$

A. Timed Trace-Based Security Properties

We restate some information flow security properties, as studied in [8]–[10], in the tSPA language.

1) Timed Nondeducibility on Compositions: The most interesting property is timed NDC (tNDC) which is the natural extension of NDC to a timed setting. Before formally introducing it, we need to discuss briefly on the nature of the admissible High users. Contrary to SPA, in the tSPA model, 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 $\mathrm{Act}H \cup \{\tau, \mathrm{tick}\}$. We call $\mathcal{E}_H^t$ the set of such processes. The reason is the following: A process Π that is not weakly time alive may prevent time from elapsing when composed in parallel with some system E. Indeed, in a compound process, time can pass iff all components let it pass. Hence, a high user which is not weakly time alive could block the time flow also for the low users and we certainly want to avoid this unrealistic (and undesirable) possibility. The tNDC property in tSPA can be, thus, defined as follows.

Definition 5: $E \in \mathrm{tNDC}$ if and only if $\forall\, \Pi \in \mathcal{E}_H^t$, we have $(E\|\Pi)\backslash H \sim_t E\backslash H$. ■

Due to the presence of the universal quantification, tNDC is not very easy to check. For SPA, we proved in [8] and [9] that NDC is equivalent to SNNI, an easierly checkable property discussed in the next section. We will show that such an equivalence result does not hold for tSPA; more precisely, tNDC is stronger than tSNNI.

2) Timed SNNI: Intuitively, E is strong nondeterministic noninterference (SNNI) if $E\backslash H$, where no high-level activity is allowed, behaves like system E where all the high-level activities are *hidden* (i.e., transformed into internal τ actions). To express this second system, we have to introduce the *hiding* operator E/L, where L is an arbitrary subset of $\mathcal{L}$. In the untimed case, it is defined by means of the following rules:

$$\frac{E_1 \xrightarrow{\alpha} E_1'}{E_1 L \xrightarrow{\alpha} E_1'/L}(\alpha \notin L \cup \overline{L}) \qquad \frac{E \xrightarrow{l} E' l \in L \cup \overline{L}}{E/L \xrightarrow{\tau} E'/L}. \quad (1)$$

Now we are ready to define the property for SPA as follows: $E \in SNNI$ if and only if, we have $E\backslash H \sim E/H$, where $\sim$ is trace equivalence.

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 *ActH*. It is defined as

$$\mathrm{Top}_H = \sum_{a \in \mathrm{Act}H} a.\mathrm{Top}_H$$

(where $\sum$ is the n-ary generalization of the binary $+$ operator). We can see that $(E\|\mathrm{Top}_H)\backslash H \sim E/H$ for every process E of *SPA*. Roughly, hiding is the same as enabling every high action. Actually, this could be considered as a good defining equation for the hiding operator.

Due to the maximal progress assumption, the above equality for the hiding operator does not hold in the timed language, even if considering the appropriate process

$$\mathrm{Top}_H^{\mathrm{tick}} = \sum_{a \in \mathrm{Act}H} \iota\left(a.\mathrm{Top}_H^{\mathrm{tick}}\right)$$

that can also let time pass arbitrarily. Indeed, consider process $E = \iota(h)$. By using the rules in (1) for hiding and considering $L = \{h\}$, we obtain that $E/L \xrightarrow{\tau}$ and, at the same time, $E/L \xrightarrow{\mathrm{tick}}$. On the contrary, $(E\|\mathrm{Top}_H^{\mathrm{tick}})\backslash L$ cannot perform *tick*, because the operational rules for the parallel operator correctly implements the maximal progress assumption. Thus we need to slightly modify the operational rules for the hiding operator E/L as follows:

$$\frac{E_1 \xrightarrow{\alpha} E_1'}{E_1/L \xrightarrow{\alpha} E_1'/L}(\alpha \notin \{L \cup \overline{L} \cup \{\mathrm{tick}\}\})$$
$$\frac{E \xrightarrow{l} E' \quad l \in L \cup \overline{L}}{E/L \xrightarrow{\tau} E'/L} \qquad \frac{E \xrightarrow{\mathrm{tick}} E' \quad \forall\, l \in L \cup \overline{L} \quad E \not\xrightarrow{\alpha}}{E/L \xrightarrow{\mathrm{tick}} E'/L}. \quad (2)$$

With these new rules, we avoid the possibility that process E/L performs a *tick* action if, at the same time, it can perform an action in $L \cup \overline{L}$, hence preventing the problem outlined above. Now, the following proposition states that for every process E its composition with $\mathrm{Top}_H^{\mathrm{tick}}$ (restricted on the high actions) is equivalent to E/H even for the finer timed bisimulation equivalence, i.e., that hiding corresponds to enabling every high-level action.

Proposition 1: For every tSPA process E, we have that $(E\|\mathrm{Top}_H^{\mathrm{tick}})\backslash H \approx_t E/H$. ■

Property tSNNI requires intuitively that process E, where high-level actions are forbidden, has the same low observable behavior of E where the high-level actions can be freely executed. This intuition can be now formally stated as follows.

Definition 6: $E \in \mathrm{tSNNI}$ iff $E\backslash H \sim_t E/H$. ■

As previously announced, this is a weaker approximation of tNDC. Indeed, tNDC is at least as strong as tSNNI.

Proposition 2: $\mathrm{tNDC} \subseteq \mathrm{tSNNI}$.

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

Example 4: To see that the previous inclusion is strict consider the timed process $E = \iota(l.\mathbf{0}) + \iota(h.\iota(h.\iota(l.\mathbf{0})))$. Process E is tSNNI since $Tr(E\backslash H) = \{\mathrm{tick}^n l \mid n \geq 0\} = Tr(E/H)$. But, if we consider the process $\Pi = \iota(\overline{h}.\iota(\mathbf{0}))$ we get that $(E\|\Pi)\backslash H$ cannot perform any trace beginning with *tick* and ending with the low action l. ■

In SPA, properties SNNI and NDC do coincide because of following two results: first, Top_H is the top element of the trace preorder (as well as $\mathbf{0}$ is the least element); second, the trace preorder is a precongruence with respect to parallel composition (cf. [8]). On the contrary, for *tSPA* we still have that $\mathrm{Top}_H^{\mathrm{tick}}$ is the top element of the timed trace preorder, but such a preorder is not a precongruence, as illustrated by the following example.

Example 5: Let us consider $E_1 = \iota(l)$ and $E_2 = \iota(l) + \iota(h)$. Clearly, $E_1 \leq_t E_2$. Now, consider $\Pi = \iota(\overline{h})$. It is easy to see that $Tr(E_1\|\Pi) = \{\mathrm{tick}^n l\overline{h} | n \geq 0\} \cup \{\mathrm{tick}^n \overline{h} l | n \geq 0\}$, while $Tr(E_2\|\Pi)$ is composed only of the four traces $\{l\overline{h}, h\overline{h}, \overline{h}l, \overline{h}h\}$, because the maximal progress assumption does not let time proceed as a synchronization on h is possible. The same two processes above show that the timed trace preorder is not a precongruence for the hiding operator too. ■

Finally, we want to emphasize that, even if tNDC seems quite a reasonable noninterference property, we actually need to move toward its counterpart tBNDC based on the finer

notion of equivalence of weak bisimulation. The main reasons are the following.

- As real-time systems are reactive systems, it is fundamental to be able to observe the choice points; for instance, we would like to distinguish the two low-level processes $l.(l_1 + l_2)$ and $l.l_1 + l.l_2$ as only for the former process there is a state where both l_1 and l_2 are executable.
- From a technical point of view, bisimulation is equipped with simple proof techniques; this is also true when considering tBNDC where we will be able to prove that a system is tBNDC just by providing a suitable bisimulation relation (see the proof of Proposition 5).

B. Timed Bisimulation-Based Security Properties

Here, we simply rephrase the properties above to the timed bisimulation setting. Besides tBNDC and tBSNNI, we will also introduce strong tBSNNI (tSBSNNI) as a stronger, more easily checkable, approximation of tBNDC.

1) tBNDC and tBSNNI: By replacing timed trace equivalence by timed weak bisimulation in Definition 5, we get the definition of tBNDC.

Definition 7: $E \in \mathrm{tBNDC}$ if and only if $\forall\, \Pi \in \mathcal{E}_H^t$ 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 check. In Section V, we directly face this problem by using partial model checking techniques. Here, instead, we address the problem by defining suitable approximations of tBNDC, following the line of [8] and [9]. 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 [8]–[10].

Property tBSNNI requires intuitively that process E where high-level actions are forbidden has the same low observable behavior of E where the high-level actions can be freely executed. This intuition can be now formally stated as follows.

Definition 8: $E \in \mathrm{tBSNNI}$ iff $E\backslash H \approx_t E/H$. ■

As previously announced, this is an approximation of tBNDC, as it is at least as strong as tBSNNI.

Proposition 3: $\mathrm{tBNDC} \subseteq \mathrm{tBSNNI}$.

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

By using an example similar to one in [8], we can also prove that the previous inclusion is strict. Consider the following (untimed) 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 $(E||\Pi)\backslash H \approx_t l.\mathbf{0} + \tau.\mathbf{0} \not\approx_t l.\mathbf{0} \approx_t E\backslash H$.

Finally, as an example of a process that is tNDC but not tBNDC, let us simply consider the following (untimed) process $E = l + h$. E is tNDC because, whatever high-level process Π one considers, the traces of $(E||\Pi)\backslash H$ and $E\backslash H$ are only $\{\epsilon, l\}$. On the contrary, when using bisimulation and by choosing $\Pi = \overline{h}, (E||\Pi)\backslash H \xrightarrow{\tau} (\mathbf{0}||\mathbf{0})\backslash H$, but no bisimilar state can be reached from $E\backslash H$.

2) tSBSNNI Property: An approximation for tBNDC that is stronger than tBNDC can be defined as follows.

Definition 9: $E \in \mathrm{tSBSNNI}$ iff $\forall\, E' \in \mathrm{Der}(E)$ we have $E' \in \mathrm{tBSNNI}$. ■

This requires that the system is tBSNNI in every derivative. Since $E \xrightarrow{\alpha} E'$ implies $\mathrm{Der}(E') \subseteq \mathrm{Der}(E)$, it directly follows the following.

Lemma 2: Let E be tSBSNNI. Then for all $E' \in \mathrm{Der}(E)$, E' is also tSBSNNI.

We prove the following two propositions (as usual, proofs in the Appendix), the first stating that tSBSNNI is at least as strong as tBNDC, the second offering a distinguishing element.

Proposition 4: $\mathrm{tSBSNNI} \subseteq \mathrm{tBNDC}$. ■

Proposition 5: The (untimed) process $E = l.(\tau.\mathbf{0} + \tau.l.\mathbf{0}) + l.h.l.\mathbf{0}$ is tBNDC but not tSBSNNI. ■

C. Compositionality of Properties

One of the most interesting features of SBSNNI is that it is compositional. The proposition below states that also the timed extension of this property has this nice feature.

Proposition 6: Whenever $E, F \in \mathrm{tSBSNNI}$, then also $E||F \in \mathrm{tSBSNNI}$. ■

By checking tSBSNNI over the parallel subcomponents of a system (as done in [9]), it is thus possible to alleviate the so-called *state-explosion* problem caused by the interleaving of all the possible executions of parallel processes. As a matter of fact, when checking $E||F$, one first checks E and F separately; if the check is successful, then $E||F$ is secure. However, it might be the case that E or F are not tSBSNNI, while $E||F$ is so; in such a case, the compositionality property does not help. Unfortunately, for tBNDC, as for BNDC (see [24, Prop. 1]), we have the following negative result.

Proposition 7: tBNDC is not compositional. ■

D. Timed Versus Untimed

Consider a timed process which enjoys a timed information flow security property. Then, if we ignore the observation of timing information (i.e., the *tick* actions are considered as τ actions), then what we obtain is a process that is secure in the untimed setting.

To be more formal, we define an additional operator $[U]$ of tSPA, which transforms *tick* actions into internal, or externally invisible, τ actions

$$\frac{E \xrightarrow{a} E'}{E[U] \xrightarrow{a} E'[U]} \qquad \frac{E \xrightarrow{\mathrm{tick}} E'}{E[U] \xrightarrow{\tau} E'[U]}.$$

For each tSPA process E, we have that $E[U]$ is not able to perform *tick* actions. Thus, its semantics could be originated by one SPA process. With the notation E is tX-secure (X-secure) we mean that $E \in tX (E \in X)$, where X ranges over BSNNI, BNDC, and SBSNNI. The following result trivially holds.

Lemma 3: If $E \in \mathrm{Proc}^t$ is tX-secure, then $E[U]$ is X-secure, where X ranges over BSNNI, BNDC, and SBSNNI. ■

IV. Two Case Studies

In this section, we will present two examples of processes that are secure, when the specification is untimed, but that are insecure only when some timing information is added to the specification. The first example is a very simple case of a manager

that implements a *no write-down/no read-up* policy on an object, that can only be read by high-level users and only be written by low-level users. The second case study is about an attack on web privacy. In our tSPA approach, we formalize this case study, originally introduced in [7].

A. Object Manager

We consider a process that has to manage, following a mutual exclusion policy, the accesses to a shared variable (see also [24, Ex. 3]). It implements the *no-write-down/no-read-up* policy (see [16]) as follows: 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. Nevertheless, 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 preliminary, untimed description of the manager is the following:

$$C = \mathrm{req}_H.\overline{\mathrm{read}_H}.C + \mathrm{req}_L.\mathrm{write}_L.C$$

where the set H of high actions is $\{\mathrm{req}_H, \mathrm{read}_H\}$ and actions $\mathrm{req}_H, \overline{\mathrm{read}_H}, \mathrm{req}_L$, and write_L represent the (abstractions of) high-level read requests, high-level read operations, low-level write requests, and low-level write operations, respectively. In particular, an access request can be done by either a high-level user (for reading) or by a low level one (for writing). After such a request the corresponding access operation is performed and finally the monitor process returns to its initial state C. In this first specification, we are abstracting from time; hence, we can try to verify the manager by checking the BNDC property. It is easy to see that C is not BNDC since the high-level process $\Pi = \overline{\mathrm{req}_H}.\mathbf{0}$ 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 [10]. This problem can be solved by introducing a (logical) timeout mechanism which unblocks the system if the high-level reading is not performed. Hence, the improved, still untimed, specification is the process C_1 below

$$C_1 = \mathrm{req}_H.(\tau.C_1 + \overline{\mathrm{read}_H}.C_1) + \mathrm{req}_L.\mathrm{write}_L.C_1.$$

Now it can be formally verified that this system is *BNDC*, since it is actually SBSNNI and such a verification can be automated by using the CoSeC tool [9]. Indeed, the (logical) timeout has resolved the deadlock problem but, if we refine the model by introducing *tick* actions, it is possible to have some timing covert channels. For example, suppose to refine $\mathrm{req}_L, \mathrm{req}_H, \mathrm{write}_L$, and the τ (which models timeout) by adding one *tick* for every such action. We also allow the system to idle for read_H actions (hence, giving implicitly priority to the high request of reading with respect to time), obtaining the following system.

$$C_2 = \mathrm{req}_H.\mathrm{tick}.(\mathrm{tick}.\tau.C_2 + \iota(\overline{\mathrm{read}_H}.C_2)) + \mathrm{req}_L.\mathrm{tick}.\mathrm{write}_L.\mathrm{tick}.C_2.$$

This system is not tBNDC; as a matter of fact, consider the following high process:

$$\Pi_1 = \iota(\overline{\mathrm{req}_H}.\iota(\mathrm{read}_H.\iota(\mathbf{0})))$$
$$\Pi_2 = \iota(\overline{\mathrm{req}_H}.\mathrm{tick}.\iota(\mathrm{read}_H.\iota(\mathbf{0}))).$$

It is now easy to see that $(C_2||\Pi_1)\backslash H \stackrel{\mathrm{tick}}{\Longrightarrow}\stackrel{\mathrm{req}_L}{\Longrightarrow}$ and $(C_2||\Pi_2)\backslash H \stackrel{\mathrm{tick}}{\Longrightarrow}\stackrel{\mathrm{tick}}{\Longrightarrow}\stackrel{\mathrm{req}_L}{\Longrightarrow}$, 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 by observing the passage of time, 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, e.g., by assuming that the timeout is set to n units of time, if a 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.

We solve these problems as follows. First, we impose a discipline of accessing the variable that forces high users to spend a fixed amount of time in the critical section; next, we show how to masquerade the high-level activity to low-level users. Hence, to tackle the first point, we differently model the timeout as

$$\begin{aligned}\mathrm{Timer} =& \mathrm{go}.\overline{\mathrm{stop}}.\mathrm{Timer}\\ C_3 =& \mathrm{req}_H.\overline{\mathrm{go}}.(\mathrm{stop}.C_3 + \overline{\mathrm{read}_H}.\mathrm{stop}.C_3)\\ &+ \mathrm{req}_L.\mathrm{write}_L.C_3\\ \mathrm{Manager} =& (\mathrm{Timer}||C_3)\backslash\{go, \mathrm{stop}\}.\end{aligned}$$

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 weakly bisimilar to C_1; therefore, also the *Manager* is SBSNNI, hence BNDC. However, it is certainly important to see if, when timing information is added, it still generates timing covert channels. As before, we refine the specification as follows (we set the timer to one unit of time):

$$\begin{aligned}\mathrm{Timer}_t =& \iota(\mathrm{go}.\mathrm{tick}.\iota(\overline{\mathrm{stop}}.\mathrm{Timer}))\\ C_t =& \mathrm{req}_H.\mathrm{tick}.\overline{\mathrm{go}}.(\iota(\mathrm{stop}.C_t)\\ &+ \iota(\overline{\mathrm{read}_H}.\iota(\mathrm{stop}.C_t)))\\ &+ \mathrm{req}_L.\mathrm{tick}.\mathrm{write}_L.\mathrm{tick}.C_t\\ \mathrm{Manager}_t =& (\mathrm{Timer}_t||C_t)\backslash\{\mathrm{go}, \mathrm{stop}\}.\end{aligned}$$

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 Π_1 has no more the computation $\stackrel{\mathrm{tick}}{\Longrightarrow}\stackrel{\mathrm{req}_L}{\Longrightarrow}$, but it still has the following one:

$$(\mathrm{Manager}_t||\Pi_1)\backslash H \stackrel{\mathrm{tick}}{\Longrightarrow}\stackrel{\mathrm{tick}}{\Longrightarrow}\stackrel{\mathrm{req}_L}{\Longrightarrow}$$

which cannot be performed by process $\mathrm{Manager}_t\backslash H$; thus, this system is not tBNDC. Time elapses when a high-level access is performed and this is observable by low-level users that have to wait until the high-level request is terminated before obtaining access to the variable. This is 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 zero or one in order to transmit data. As anticipated above, in order to obtain a secure manager, the second step is to mask the high-level activity to low-level users. A solution to this problem could be to split the execution time of the $\mathrm{Manager}_t$ into two slots of fixed length. Then, in one slot only the high-level users are served

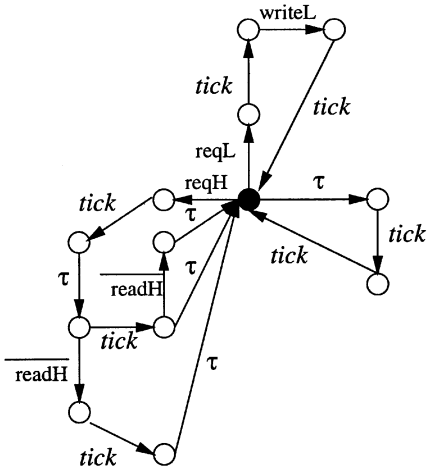

Fig. 2. The behavior of the process $\mathrm{Manager}'_t$, represented by a labeled transition system.

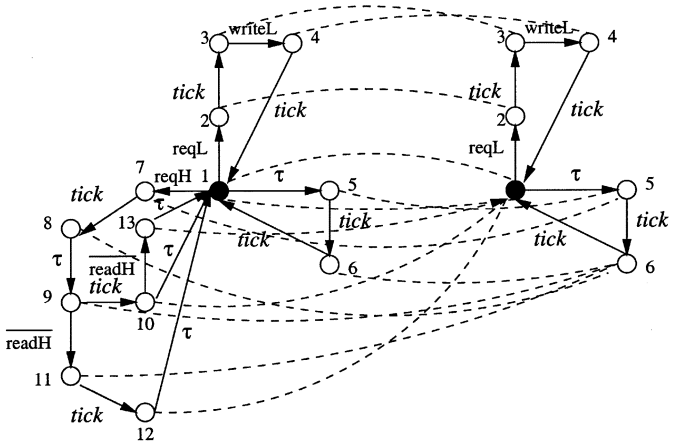

Fig. 3. A graphical explanation of *tBNDC* membership of $\mathrm{Manager}'_t$.

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 nondeterministically stops the process $\mathrm{Manager}_t$ 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 a 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 $\mathrm{Manager}'_t$ is, then, specified as follows:

$$\begin{aligned}\mathrm{Timer}_t =& \iota(\mathrm{go}.\mathrm{tick}.\iota(\overline{\mathrm{stop}}.\mathrm{Timer}))\\ C' =& \mathrm{req}_H.\mathrm{tick}.\overline{\mathrm{go}}.(\iota(\mathrm{stop}.C')\\ &+ \iota(\overline{\mathrm{read}_H}.\iota(\mathrm{stop}.C')))\\ &+ \mathrm{req}_L.\mathrm{tick}.\mathrm{write}_L.\mathrm{tick}.C'\\ &+ \tau.\mathrm{tick}.\mathrm{tick}.C'\\ \mathrm{Manager}'_t =& (\mathrm{Timer}_t \| C')\backslash\{\mathrm{go},\mathrm{stop}\}\end{aligned}$$

where a branch $\tau.\mathrm{tick}.\mathrm{tick}.C'$ has been added in C_t (now C'). See Fig. 2 for the LTS of $\mathrm{Manager}'_t$.

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

Remark 2: The technique of masquerading high-level activities has a drawback: there is no guarantee of liveness. As a matter of fact, $\mathrm{Manager}'_t$ can be forever engaged in the masquerading branch $\tau.\mathrm{tick}.\mathrm{tick}.C'$, hence, in a (timed) livelock. Moreover, it may be not obvious how to use this technique when further implementation details are added to the specification. For instance, if each branch of system $\mathrm{Manager}'_t$ is taken according to some probability value, then, one may show that a probabilistic covert channel is still possible. Preliminary results on a possible extension of the noninterference theory of [8] and [10] to a probabilistic setting is reported in [1].

We can directly prove that $\mathrm{Manager}'_t \in tBNDC$. The idea is to build a timed weak bisimulation that contains

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

for every high user $\Pi \in \mathcal{E}^t_H$. This is one of the advantages of considering timed (weak) bisimulation as an 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 (as BNDC membership requires)! Consider Fig. 3. For the sake of simplicity, in Fig. 3, we use natural numbers to represent derivatives instead of terms. The dashed lines represent pairs of a function f from derivatives of $\mathrm{Manager}'_t$ to derivatives of $\mathrm{Manager}'_t\backslash H$; e.g., $f(9) = 6$. Hence, we have the following relation $\mathcal{R}$:

$$\{(E\|\Pi\backslash H, f(E)) \mid \Pi \in \mathcal{E}^t_H, E \in \mathrm{Der}(\mathrm{Manager}'_t)\}.$$

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

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

B. Cache Attack on Web Privacy

We formalize in our framework the example reported in [7]. The attack compromises the privacy of user's web-browsing histories by allowing a malicious web site to determine whether or not the user has recently visited some other, unrelated, web page w. A Java applet is embedded in the malicious web site and, when a user surfs on it, the applet is downloaded and run by the user's browser. The applet first performs a request to access web page w, and then it performs a new request to the originating malicious site. So, assuming that the network delay is somehow constant in a short period of time (this assumption is perhaps not too realistic), the malicious site can measure the time elapsed between the two requests which it receives from the user, and, if such a time is under a certain bound, it infers that w was in the cache of the browser of the user, thus implying that w has been recently visited by the user. This shows how some information about the cache of web browsers might be leaked to the external world. In particular, it could be possible to detect whether or not a web browser has recently accessed a particular web page.

In this context, the high-level component is the cache. The low-level processes must not deduce anything about the cache. Thus, the low-level view of the system must be the same for whatever cache is present. Unfortunately, this is not the case. Consider the following description of the system under investigation:

$$\begin{aligned} B &= \iota(\overline{r_e}.\iota(a_e.\mathrm{Time\_out}(1,2,\mathrm{Cache},\mathrm{Web}))) \\ \mathrm{Cache} &= \mathrm{cached}_w.B \\ \mathrm{Web} &= \mathrm{Time\_out}(100,250,B,B). \end{aligned}$$

System B works as follows. We assume that there is an applet in the system which requests the web page. The system allows the applet to perform a request (denoted by the action r_e) to the original malicious site e, which may answer, by performing action a_e. Then, the applet requests a particular web page w, i.e., the web page it wants to know whether or not it is in the cache. We have two different possibilities depending on the presence of this page in the cache. If so, the system may again perform a request to the site e, in at most two units of time; otherwise, the system will access the web and the new request to the web site e cannot be performed before 100 units of time. Therefore, we can simply check that the cache may be exploited to leak some information since the low-level view of the system depends on the status of the cache.

Let $H = \{\mathrm{cached}_w\}$. We can show that B is not tBSNNI, i.e., $B\backslash H \not\approx_t B/H$. In particular, note that $B/H \stackrel{\overline{r_e}}{\Longrightarrow} \stackrel{a_e}{\Longrightarrow} \stackrel{\mathrm{tick}}{\Longrightarrow} \stackrel{\overline{r_e}}{\Longrightarrow}$, while $B\backslash H$ is not able to perform such a computation. As a matter of fact, $B\backslash H$ represents a system where no cached_w action is possible. This models an empty cache. While B/H represents a system whose cache is accessible, in particular page w, as $\overline{\mathrm{cached}_w}$ is a high action. Summing up, we have that B is not tBSNNI and consequently that B is not tBNDC.

V. Partial Model Checking

In [23]–[26], 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:

$$\forall \text{ hostile environment } \Pi \quad E \mid \Pi \models \phi \tag{3}$$

where E 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 information leaks, etc.), then the above property ensures that this property is granted by the system against every possible enemy (malicious user, environment etc.), that may try to interfere, modify or capture some information from the system E.

For analyzing similar properties we still have the problem of the universal quantification on every hostile environment. But, we note 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 with a process E satisfies a certain formula.

The key idea applied in [23] to tackle verification problems like (3) is to develop and suitably apply partial model checking techniques [3] 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.[4] 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 noninterference properties such as BNDC-like ones and also properties of cryptographic protocols e.g., secrecy [24], [25] and authentication [22].

In this section, we show the flexibility of the methodology proposed in [23] and [25] 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* (3). 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 $P\mathcal{R}Q$, $Q \models \phi$. Hence, it is possible to express relationships between processes in a logical way (for certain relationships). Furthermore, we can note that timed weak bisimulation is similar to observational equivalence and, hence, we can apply the results for characteristic formulas of observational equivalence developed in [28], [33].
2) *Reducing properties of the form* (3) *to validity problems*. This point may be solved by suitably exploiting the results in [23] which extend the theory of partial model checking [3] to deal also with operators whose semantics rules use negative premises (e.g., the timed parallel operator).

[4]Sometimes partial model checking is also called partial evaluation since there has been the evaluation of a part of the system.

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 any 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 *noninterference* properties.

A. Equational μ—Calculus

As a logic for expressing properties of processes we use the equational μ—calculus, a variant of μ—calculus, which is very suitable for partial model checking (see [3]). Let a be in *Act* and X be a variable ranging over a finite set of variables *Vars*. Equational μ—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:

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

Roughly, the semantics $[\![D]\!]$ of the list of equations D is the solution of the system of equations corresponding to D.[5] 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 $E \models D \downarrow X$ as a notation for $E \in [\![D]\!](X)$. We give only an intuitive idea of the operators of this logic. For example we may expresses the fact that a process E is able to perform an a action, by requiring that $E \models \langle a \rangle \mathbf{T}$ (possibility operator). We can also require that after performing an action a the process E must satisfy a property ψ, by $E \models [a]\psi$ (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.

B. Partial Model Checking for the Timed Parallel Operator

Partial model checking techniques have been developed for the compositional analysis of concurrent systems (processes). (Here, we follow [3].) The intuitive idea is the following: Suppose to have a system $E||F$ which is the parallel composition of two processes E, F, and we wonder whether this process satisfies a formula ϕ or not. Now, it is possible to find a reduced formula $\phi_{//E}$ s.t.

$$E||F \models \phi \text{ iff } F \models \phi_{//E}.$$

[5]See Appendix or [3] for the formal semantics of the equational μ—calculus.

TABLE I
PARTIAL EVALUATION FUNCTION FOR PARALLEL OPERATOR $\|$ OF TIMED *SPA*

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

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. Note that the reduced formula $\phi_{//E}$ only depends on the formula ϕ and the process E. No information is required on the process F. In our scenario, F represents the possible enemies. Thus, given a certain system E, we can find the property that the enemies 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 these are represented in the structured operational semantics (SOS) style (see [3]) and the logical language is the equational μ—calculus. Unfortunately, tSPA semantics contains also rules with negative premises (e.g., parallel composition), and those techniques cannot be directly applied. But in [23] a slight extension to the theory of Andersen has been proposed in such a way that is also possible to deal with semantic rules with negative premises (actually with a subset of GSOS rules). Thus, by suitably applying the results of [23], we can automatically derive the partial evaluation function $D//E$ for the timed parallel operator and the equational μ—calculus. This is given in Table I, where E is a tSPA process and D an equational definition. Thus, we can state the following.

Lemma 4: Given two processes E_1, E_2 (E_1 finite-state) and an equational specification $D \downarrow X$, we have

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

■

Example 6: Let us see an example of partial evaluation. Consider the process $E = \iota(h.\mathbf{0})$. We want to find the necessary and sufficient conditions on a process X s.t.

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

By applying the rules in Table I, we find out that $\langle \mathrm{tick} \rangle \mathbf{T}//E$ is equal to $(\langle \mathrm{tick} \rangle \mathbf{T}) \wedge [\overline{h}]\mathbf{F}$. 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 E through h. ■

C. Analyzing BNDC-Like Properties

First of all, it is useful to parameterize the definition of NDC with respect to the set S of *high* users that are composed with the system when it is checked (as done in [24]). Indeed, 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 μ—calculus.

Definition 10: $E \in \mathrm{tBNDC}^S$ if and only if $\forall\, \Pi \in S : (E\|\Pi)\backslash H \approx_t E\backslash H$. ■

By using the characteristic formula for $\approx_t$ of $E\backslash H$ (see Appendix), we obtain the following characterization[6]:

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

Now, we can apply the partial evaluation function for the restriction operator (see [3]) and then the one for timed parallel operator (see Table I) to the formula $\phi_{\approx_t, E\backslash H}$ by getting a formula ϕ'. Then the previous problem is equivalent to checking that every process in S satisfies ϕ'. Indeed, the behavior of E has been evaluated and encoded in the formula ϕ'. Thus

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

We expect to have decidability results only if we restrict ourselves to finite-state systems. Let $fs = \{E|\mathrm{Der}(E) \text{ is finite}\}$ 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 b

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

Unfortunately, if we consider S as $\mathcal{E}_H^t$, then it is not so easy to restrict to consider only LTSs which are the semantics of processes in $\mathcal{E}_H^t$. These LTSs enjoy several properties that can be handled, in particular *time determinacy* and maximal progress. In fact, we can use *tick*—deterministic equational μ—calculus [23], in order to deal with the transition systems generated by the semantics of *tSPA* terms, instead of simple μ—calculus. The validity problem for this logic may be shown to be decidable by using the same proof techniques of [35]. 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 Fig. 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 Fig. 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,

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

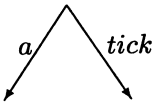

Fig. 4. Example of LTS that does not corresponds to the semantics of a process E.

we consider properties that are stronger than tBNDC.[7] We call fs^* the set of finite-state processes, which are *tick*-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 it is possible to generate LTSs which do not enjoy this property).

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

Proposition 8: Property tBNDC^{fs^*} is decidable for all finite-state processes E. ■

D. An Example

Now, we show the flexibility of the specification format (3) 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 reconsider the process $E = l.\mathbf{0} + h.h.l.\mathbf{0}$. Consider 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, in whatever state it reaches by means of τ actions, is still able to move in a state where an action l can be performed, possibly after a finite sequence of τ actions. Hence, we would like to study properties like

$$\forall\, \Pi \in S \quad (E\|\Pi)\backslash H \models D \tag{4}$$

where $H = \{\overline{h}, h\}$. As for the study of tBNDC-like properties, we can apply the partial evaluation for the parallel operator (see Table I) and restriction one (see [3] and [23]) and after several logical simplifications (reported in the Appendix), we obtain that $D//\backslash H//E$ is

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

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 $\overline{h}$. The information obtained through partial model checking can also be used to enforce a security policy which prevents a system from having certain information leaks (see [26]).

[7]However, we feel that is possible to suitably change the validity procedure of [35] 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.

Suppose to have the following execution policy for high users: Every high user, when executed in the system E, is equipped with a control process $C = \mathrm{Top}_{\bar{h}}^{\mathrm{tick}}$. Hence, we want to study problems like

$$\forall\, \Pi \quad (E||(C||\Pi))\backslash H \models D. \tag{5}$$

Now, we already know which properties processes $(C||\Pi)$ must enjoy, these are exactly $D//\backslash H//E$. Thus, we may work in a compositional way and we simply build the reduced formula $(D//\backslash H//E)//C$ which expresses the properties the processes Π must enjoy in order to have (5). Now, it is easy to see that $(D//\backslash H//E)//C = \mathbf{T}$. This means that for every high user Π, if we encapsulated it together with the process C, then (5) is satisfied.

VI. Conclusions and Future Research

We have shown the flexibility of the noninterference theory of [8]–[10] and [24], 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. As a future work, we plan to extend the CoSeC tool to manage tSPA specifications, by defining a mapping from tSPA to the timed language implemented in the concurrency workbench [4].

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. Moreover, other aspects of system behavior can be included in the specification, e.g., probability (see [17]). Preliminary results about the extension of the noninterference theory to a probabilistic process algebra are reported in [1].

Related literature includes [20] and [32]. In [32], a CSP based process algebra, extended with a special event to mark the passage of time, is used for the analysis of cryptographic protocols. The semantic model is rather similar (even if maximal progress is not assumed); we think that, in the line of [30], also the noninterference theory developed in this paper can be adapted to that setting. Similarly, the idea in [32] of using a real-time calculus for the analysis of time-dependent properties of cryptographic protocols can be adapted also to our setting, in the line of [12] and [14]. In [20], the dense-time model of timed automata [2] is used as a basis for studying information flow properties, based on trace semantics. The information flow properties studied there seem to be less restrictive than tBNDC since they deal on the recognition of given patterns on the execution of high and low activities.

Appendix

A. Semantics of Equational μ—Calculus

In this section, we give the formal semantics of equational μ—calculus (see [3]). 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 $\mathrm{Def}(D)$. Let $\langle S, A, \{\stackrel{a}{\longrightarrow}\}_{a\in A}\rangle$ 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 $\mathrm{Def}(D)$. The semantics $[\![A]\!]'_\rho$ of an assertion A is the following:

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

The semantics of a list of equations D, $[\![D]\!]_\rho$ is an environment that assigns subsets of S to variables in $\mathrm{Def}(D)$. A list of equations is closed if every variable that appears in the assertions of the list is in $\mathrm{Def}(D)$. 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

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

where

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

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*.

B. Characteristic Formula

Given a finite-state process E, we present below the definition of a formula ϕ_E that is characteristic (with respect to timed *weak* bisimulation) for this process E (see [28]). Let $\langle\langle\tau\rangle\rangle\phi$ be a short notation for $\mu X.\langle\tau\rangle X \vee \phi$, where X is not free in ϕ. (The semantics of μ can be found in [36]). Let $\langle\langle l\rangle\rangle\phi, (l \in \mathcal{L} \cup \{\mathrm{tick}\})$ be $\langle\langle\tau\rangle\rangle\langle l\rangle\langle\langle\tau\rangle\rangle\phi$. 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 11: 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 $E' \in \mathrm{Der}(E)$, $a \in \mathrm{Act}$:

$$X_{E'} =_\nu \left(\bigwedge_{a, E'' : E' \stackrel{a}{\longrightarrow} E''} \langle\langle a\rangle\rangle X_{E''} \right) \wedge \left(\bigwedge_a \left([a] \left(\bigvee_{E'' : E' \stackrel{a}{\Longrightarrow} E''} X_{E''} \right) \right) \right).$$

■

Intuitively, for every state of the process there is a variable that encodes the capabilities of that state. Following [28], [33], and [34] if ϕ_{E_2} is characteristic for E_2 (with respect to $\approx_t$) then

Lemma 5:

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 $E_1 \approx_t E_2$.

■

C. Technical Proofs

Proposition 1: For every process E, we have $(E\|\mathrm{Top}_H^{\mathrm{tick}})\backslash H \approx_t E/H$.

Proof: For the sake of readability we use *Top* instead of $\mathrm{Top}_H^{\mathrm{tick}}$. We show that the following relation is a timed weak bisimulation:

$$\mathcal{R} = \{((E\|\mathrm{Top})\backslash H, E/H) \mid E \in \mathrm{Proc}^t\}.$$

We analyze only the most interesting case (*tick* move). Assume that $((E\|\mathrm{Top})\backslash H, E/H) \in \mathcal{R}$. Then

- $(E\|\mathrm{Top})\backslash H \xrightarrow{\mathrm{tick}} (E'\|\mathrm{Top})\backslash H$, it means that $E \xrightarrow{\mathrm{tick}} E'$ and, due to the *maximal progress* assumption, for no action h in $H \cup \overline{H}$, we have $E \xrightarrow{h}$. But, in this case also $E/H \xrightarrow{\mathrm{tick}} E'/H$ and $((E'\|\mathrm{Top})\backslash H, E'/H) \in \mathcal{R}$.
- $E/H \xrightarrow{\mathrm{tick}} E'/H$, it means that $E \xrightarrow{\mathrm{tick}} E'$ and for no action h in $H \cup \overline{H}$, we have $E \xrightarrow{h}$. So, since *Top* is always capable to let one unit of time pass, i.e., $\mathrm{Top} \xrightarrow{\mathrm{tick}} \mathrm{Top}$, we have also $(E\|\mathrm{Top})\backslash H \xrightarrow{\mathrm{tick}} (E'\|\mathrm{Top})\backslash H$ and $((E'\|\mathrm{Top})\backslash H, E'/H) \in \mathcal{R}$. ∎

Proposition 4: $\mathrm{tSBSNNI} \subseteq \mathrm{tBNDC}$.

Proof: Let E be a process satisfying tSBSNNI. The proof that E is also tBNDC is performed by showing that the following relation is a timed weak bisimulation (up to)

$$\mathcal{R} = \{(E'\backslash H, E'\|\Pi\backslash H) \mid E' \in \mathrm{Der}(E), \quad \Pi \in \mathcal{E}_H^t\}.$$

Then, the thesis follows because $E \in \mathrm{Der}(E)$, hence, the pairs $(E\backslash H, E\|\Pi\backslash H) \in \mathcal{R}$ for all $\Pi \in \mathcal{E}_H^t$. The proof proceeds by inspection of possible cases. Note that if $E' \in \mathrm{Der}(E)$ and $E' \xrightarrow{a} E''$ then $E'' \in \mathrm{Der}(E)$ and so this condition of the relation $\mathcal{R}$ is always satisfied. Let us consider the pair $(E'\backslash H, E'\|\Pi\backslash H) \in \mathcal{R}$. Let us first consider the moves of $E'\backslash H$.

- If $E'\backslash H \xrightarrow{a} E''\backslash H$ $(a \neq \mathrm{tick})$, then $E'\|\Pi\backslash H \xrightarrow{a} E''\|\Pi\backslash H$, and also $(E''\backslash H, E''\|\Pi\backslash H) \in \mathcal{R}$.
- If $E'\backslash H \xrightarrow{\mathrm{tick}} E''\backslash H$, then necessarily $E' \xrightarrow{\mathrm{tick}}$ and consequently, by Lemma 1 $E' \overset{\mathrm{tick}}{\not\longrightarrow}$ (as well as for $E'\backslash H$). The following *fact* holds: for whatever sequence of high actions $\alpha_1, \ldots, \alpha_n$ s.t. $E' \xRightarrow{\alpha_1} E_1 \ldots E_{n-1} \xRightarrow{\alpha_n} E_n$, we have $E'/H \xRightarrow{\tau} E_n/H$ and $E_n/H \approx_t E'/H \approx_t E'\backslash H \approx_t E_n\backslash H$. Since, $E'\backslash H \xrightarrow{\mathrm{tick}}$ and $E'\backslash H \approx_t E'/H$, we have that there exists a sequence of high actions $\alpha_1, \ldots, \alpha_n$ s.t. $E' \xRightarrow{\alpha_1} E_1 \ldots E_{n-1} \xRightarrow{\alpha_n} E_n$, $E_n \xrightarrow{\alpha}$ for no action $h \in H$ and $E_n/H \xrightarrow{\mathrm{tick}} \xRightarrow{\tau} E'''/H$, with $E'''/H \approx_t E'''\backslash H \approx_t E''\backslash H$. Thus, if we consider $(E'\|\Pi)\backslash H$, then, we may have a sequence $(E'\|\Pi)\backslash H \xRightarrow{\tau} (E_i\|\Pi')\backslash H$, with $1 \leq i \leq n$ s.t. if $E_i \xrightarrow{h}$ then $\Pi' \overset{\overline{h}}{\not\longrightarrow}$. Now, from the previous *fact*, it follows that $E_i\backslash H \approx_t E_i/H \approx_t E'\backslash H$ and so $E_i\backslash H \xRightarrow{\mathrm{tick}} E^{iv}\backslash H$, with $E^{iv}\backslash H \approx_t E''\backslash H$, since $E'\backslash H \xrightarrow{\mathrm{tick}} E''\backslash H$. This means that $(E'\|\Pi)\backslash H \xRightarrow{\mathrm{tick}} (E^{iv}\|\Pi'')\backslash H$. Since $E''\backslash H \approx_t E^{iv}\backslash H \mathcal{R} (E^{iv}\|\Pi'')\backslash H$, the thesis follows.

Let us now consider the moves of $E'\|\Pi\backslash H$. (We show only the most interesting cases.)

- If $E'\|\Pi\backslash H \xrightarrow{\tau} E''\|\Pi'\backslash H$, with $E' \xrightarrow{h} E''$ and $\Pi \xrightarrow{\overline{h}} \Pi'$, then $E'/H \xrightarrow{\tau} E''/H$. As $E' \in \mathrm{tBSNNI}$ we have that $E'\backslash H \approx_t E'/H$; hence, to match the τ transition above, there exists E_1 such that $E'\backslash H \xRightarrow{\tau} E_1\backslash H$ with $E_1\backslash H \approx_t E''/H$; moreover, also $E'' \in \mathrm{tBSNNI}$, hence, $E''/H \approx_t E''\backslash H$. Summing up, we have that $(E''\backslash H, E''\|\Pi\backslash H) \in \mathcal{R}$ up to weak bisimulation.
- If $E'\|\Pi\backslash H \xrightarrow{\mathrm{tick}} E''\|\Pi'\backslash H, E' \xrightarrow{\mathrm{tick}} E''$ and $\Pi \xrightarrow{\mathrm{tick}} \Pi'$, then $E'\backslash H \xrightarrow{\mathrm{tick}} E''\backslash H$ and $(E''\backslash H, E''\|\Pi'\backslash H) \in \mathcal{R}$. ∎

Proposition 5: The process $E_1 = l.(\tau.\mathbf{0} + \tau.l.\mathbf{0}) + l.h.l.\mathbf{0}$ is in tBNDC, but not in tSBSNNI.

Proof: In order to prove that $E \notin \mathrm{tSBSNNI}$, consider the following derivative: $E' = h.l.\mathbf{0}$. This is not tBSNNI because $\mathbf{0} \approx_t E'\backslash H \not\approx_t E'/H \approx_t \tau.l.\mathbf{0}$.

In order to prove that E is in tBNDC, we need only to prove that the relation $\mathcal{R}$ given below is a timed weak bisimulation. Let us first split the set $\mathcal{E}_H^t$ into three disjoint sets, depending on their ability to perform action h. Set S^1 is formed by the processes that cannot perform $\overline{h}$ weakly, i.e., $\Pi \overset{\overline{h}}{\not\Longrightarrow}$. Set S^2 contains exactly those processes that have τ-derivatives in set S^1 (hence, unable to perform $\overline{h}$), as well as τ-derivatives that can perform $\overline{h}$; formally: $\Pi \Longrightarrow \Pi'$ and $\Pi' \in S^1$ and $\Pi \Longrightarrow \Pi''$ and $\Pi'' \xRightarrow{\overline{h}}$. Let S^3 be $\mathcal{E}_H^t \setminus (S^1 \cup S^2)$, hence containing the processes that can never reach silently a state where $\overline{h}$ cannot be performed weakly.

Relation $\mathcal{R}$ is defined as follows:

$$\begin{aligned}\mathcal{R} = &\{((E_1\|\Pi) \setminus H, E_1\backslash H) \mid \Pi \in \mathcal{E}_H^t\} \\ &\cup \{((E\|\Pi) \setminus H, E\backslash H) \mid \Pi \in \mathcal{E}_H^t, \\ &\qquad E \in \{\mathbf{0}, l, \tau + \tau.l\})\} \\ &\cup \{((\mathbf{0}\|\Pi) \setminus H, h.l \setminus H) \mid \Pi \in \mathcal{E}_H^t\} \\ &\cup \{((h.l\|\Pi) \setminus H, \mathbf{0} \setminus H) \mid \Pi \in S^1\} \\ &\cup \{((h.l\|\Pi) \setminus H, (\tau + \tau.l) \setminus H) \mid \Pi \in S^2\} \\ &\cup \{((h.l\|\Pi) \setminus H, l \setminus H) \mid \Pi \in S^3\}.\end{aligned}$$

Now, let us see that the above is a timed weak bisimulation. Consider first the pair $((E_1\|\Pi) \setminus H, E_1\backslash H)$, and let us start with the moves from $(E_1\|\Pi) \setminus H$.

- If $(E_1\|\Pi) \setminus H \xrightarrow{l} (\tau + \tau.l\|\Pi) \setminus H$, then also $E_1 \setminus H \xrightarrow{l} (\tau + \tau.l)\backslash H$ and $((\tau + \tau.l\|\Pi)\backslash H, (\tau + \tau.l)\backslash H) \in \mathcal{R}$ (second group).
- If $(E_1\|\Pi) \setminus H \xrightarrow{l} (h.l\|\Pi) \setminus H$, then, we have three possible disjoint cases:
  - $\Pi \in S^1$. Then $E_1\backslash H \Longrightarrow \mathbf{0}\backslash H$ and $((h.l\|\Pi) \setminus H, \mathbf{0}\backslash H) \in \mathcal{R}$,
  - $\Pi \in S^2$. Then $E_1\backslash H \Longrightarrow (\tau + \tau.l) \setminus H$ and $((h.l\|\Pi) \setminus H, (\tau + \tau.l)\backslash H) \in \mathcal{R}$.
  - $\Pi \in S^3$. Then $E_1\backslash H \Longrightarrow l \setminus H$ and $((h.l\|\Pi) \setminus H, l\backslash H) \in \mathcal{R}$.

When considering the actions performed by $E_1 \setminus H$ it is more easy to prove the membership in $\mathcal{R}$ of the respective derivatives. The only interesting case is when $E_1 \setminus H \xrightarrow{l} h.l \setminus H$; in such a case $(E_1\|\Pi)\backslash H \Longrightarrow (\mathbf{0}\|\Pi)\backslash H$, with the pair $((\mathbf{0}\|\Pi)\backslash H, h.l.\mathbf{0}\backslash H) \in \mathcal{R}$.

The proof for the pairs in the second, third, and fourth groups are omitted because they are trivial.

Next, we consider the fifth group of pairs of the form: $((h.l||\Pi) \setminus H, (\tau + \tau.l) \setminus H)$ with Π in S^2. We have the following three subcases.

- If $(h.l||\Pi)\setminus H \xrightarrow{\tau}(l||\Pi')\setminus H$, then $(\tau+\tau.l)\setminus H \stackrel{\tau}{\Longrightarrow} l\setminus H$ and the pair $((l||\Pi) \setminus H, l \setminus H) \in \mathcal{R}$ (second group).
- If $(\tau + \tau.l) \setminus H \xrightarrow{\tau} \mathbf{0} \setminus H$ then, since $\Pi \in S^2$, we have that $\Pi \Longrightarrow \Pi'$ with $\Pi' \in S^1$ and so $((h.l||\Pi) \setminus H \stackrel{\tau}{\Longrightarrow}(h.l||\Pi')\setminus H$ and $((h.l||\Pi')\setminus H, \mathbf{0}\setminus H) \in \mathcal{R}$ (fourth group).
- If $(\tau+\tau.l)\setminus H \xrightarrow{\tau} l\setminus H$ then, since $\Pi \in S^2$, we have that $\Pi \Longrightarrow \Pi'$ with $\Pi' \stackrel{\overline{h}}{\Longrightarrow}$ and so $(h.l||\Pi) \setminus H \stackrel{\tau}{\Longrightarrow}(h.l||\Pi') \setminus H$ and the pair $((h.l||\Pi') \setminus H, l \setminus H) \in \mathcal{R}$ (sixth group).

Finally, we consider pairs like $((h.l||\Pi) \setminus H, l \setminus H)$ with $\Pi \in S_3$.

- If $(h.l||\Pi)\setminus H \xrightarrow{\tau}(h.l||\Pi')\setminus H$, then also $l \setminus H \Longrightarrow l \setminus H$ and $((h.l||\Pi') \setminus H, l \setminus H) \in \mathcal{R}$. Similarly, if $(h.l||\Pi) \setminus H \xrightarrow{\tau}(l||\Pi') \setminus H$ due to a synchronization, then also $l \setminus H \Longrightarrow l \setminus H$ and $((l||\Pi') \setminus H, l \setminus H) \in \mathcal{R}$ (second group).
- If $l \setminus H \xrightarrow{l} \mathbf{0} \setminus H$, then also $(h.l||\Pi) \setminus H \stackrel{l}{\Longrightarrow}(\mathbf{0}||\Pi') \setminus H$ and $((\mathbf{0}||\Pi') \setminus H, \mathbf{0} \setminus H) \in \mathcal{R}$ (second group).

■

Proposition 8: tBNDC^{fs^*} is decidable for all finite-state processes E.

Proof: Similar to the proof of [24, Prop. 4]. In particular, since we deal with *tick*-deterministic processes, we use *tick*-deterministic μ—calculus (see [23] and [35]). Moreover, Lemmas 4 and 5 are used, together with a simple logical characterization of weakly time alive processes and maximal progress.

Let $(D_{E\setminus H} \downarrow X_{E\setminus H})$ be the characteristic formula (up to timed weak bisimulation) for $E\setminus H$, then: $E \in \mathrm{tBNDC}^{fs^*}$ if and only if $\forall\, \Pi \in fs^*$, we have $(E||\Pi)\setminus H \models (D_{E\setminus H} \downarrow X_{E\setminus H})$. Thus

$$\begin{aligned} & E \in tBNDC^{fs^*} \\ \Longrightarrow & \forall\, \Pi \in fs^* : (E||\Pi)\setminus H \approx_t E\setminus H \\ \Longrightarrow_{5.i} & \forall\, \Pi \in fs^* : (E||\Pi)\setminus H \models (D_{E\setminus H}\downarrow X_{E\setminus H}). \end{aligned}$$

On the other hand

$$\begin{aligned} & E \notin tBNDC^{fs^*} \\ \Longrightarrow & \exists \Pi \in fs^* : (E||\Pi)\setminus H \not\approx_t E\setminus H \\ \Longrightarrow & \exists \Pi \in fs^* : ((E||\Pi)\setminus H \text{ is finite state}) \\ & \qquad \wedge (E||\Pi)\setminus H \not\approx_t E\setminus H \\ \Longrightarrow_{5.ii} & \exists \Pi \in fs^* : (E||\Pi)\setminus H \not\models (D_{E\setminus H} \downarrow X_{E\setminus H}) \\ \Longrightarrow & \neg(\forall\, \Pi \in fs^* : (E||\Pi)\setminus H \models (D_{E\setminus H} \downarrow X_{E\setminus H})). \end{aligned}$$

So by using the partial model checking for the parallel operator (Lemma 4) and the partial evaluation for restriction ([3], [23]), we have

$$\begin{aligned} & E \in \mathrm{tBNDC}^{fs^*} \text{ iff} \\ & \forall\, \Pi \in fs^*\ \Pi \models ((D_{E\setminus H}\downarrow X_{E\setminus H})//\setminus L)//E. \end{aligned}$$

Now, applying the translation tr from equational μ—calculus to standard one (see [23]), we obtain a closed formula $\phi_E = tr(((D_{E\setminus H} \downarrow X_{E\setminus H})//\setminus L)//E)$ of μ—calculus. To reduce the problem to a validity problem on *tick*-deterministic μ—calculus, we use the formula ϕ''

$$\nu X.([H \cup \{\tau\} \cup \{\mathrm{tick}\}]X \wedge ([\mathcal{L}\setminus H]F) \wedge \langle\langle \mathrm{tick}\rangle\rangle \wedge(\langle \mathrm{tick}\rangle \Longrightarrow [\tau]\mathbf{F}))$$

which characterizes the high processes which are weakly time alive and enjoy the maximal progress assumption.

We can now reduce the decision problem of BNDC^{fs^*} membership to a validity problem on μ—calculus. Now we have $E \in \mathrm{tBNDC}^{fs^*}$ iff $\phi'' \Longrightarrow \phi_E$ (i.e., $\neg\phi'' \vee \phi_E$) is valid. Indeed

$$\begin{aligned} & E \notin \mathrm{tBNDC}^{fs^*} \\ \Longrightarrow & \exists \Pi \in fs^*\Pi \not\models \phi_E \\ \Longrightarrow & \phi'' \Longrightarrow \phi_E \text{ is not valid.} \end{aligned}$$

On the other hand

$$\begin{aligned} & \phi'' \Longrightarrow \phi_E \text{is not valid} \\ \Longrightarrow & \exists \Pi \models \phi'' \text{ and } \Pi \not\models \phi_E \\ \Longrightarrow & \exists \Pi \in fs^*\ \Pi \not\models \phi_E \\ & (\mu - \text{calculus enjoys the finite model property}) \\ \Longrightarrow & E \notin \mathrm{tBNDC}^{fs^*}. \end{aligned}$$

Since on *tick*-deterministic μ—calculus the validity problem is decidable [35], then the result follows. Notice that in [23] a sound axiomatization of *tick*-deterministic μ—calculus is given. Hence, we can use this axiomatization in oder to prove the validity of formulas instead of using the validity procedure. ■

D. Compositionality Results

Lemma 6: $(E|F)/H \approx_t (E/H)|(F/H)$.

Proof: The following is a timed weak bisimulation:

$$\mathcal{R} = \{(E||F/H, E/H||F/H) \mid E, F \in \mathrm{Proc}^t\}.$$

We show only two interesting cases

- $E||F/H \xrightarrow{\tau} E'||F'/H$ since $E \xrightarrow{h} E'$ and $F \xrightarrow{\overline{h}} F'$ then also $E/H||F/H \xrightarrow{\tau} \xrightarrow{\tau} E'/H||F'H$ and $(E'||F'/H, E'/H||F'/H) \in \mathcal{R}$.
- $E||F/H \xrightarrow{\mathrm{tick}} E'||F'H$ since $E \xrightarrow{\mathrm{tick}} E'$ and $F \xrightarrow{\mathrm{tick}} F'$, this means that neither E nor F can perform actions in $H \cup \overline{H}$. Hence, we have $E/H \xrightarrow{\mathrm{tick}} E'/H$ and $F/H \xrightarrow{\mathrm{tick}} F'/H$ and $E/H||F/H \xrightarrow{\mathrm{tick}} E'/H||F'/H$ and $(E'||F'/H, E'/H||F'/H) \in \mathcal{R}$.

■

Moreover, we have to note that if we consider two processes E and F which are timed weakly bisimilar, then $\approx_t$ is preserved by parallel composition.

Lemma 7: If $E \approx_t F$ then for every R we have $E||R \approx_t F||R$.

Proof: The following is a timed weak bisimulation:

$$\mathcal{R} = \{(E||R, F||R) \mid E \approx_t F, R \in \mathrm{Proc}^t\}.$$

We show only the interesting case of the *tick* action.

- If $E||R \xrightarrow{\mathrm{tick}} E'||R'$, then $E \xrightarrow{\mathrm{tick}} E'$ and $R \xrightarrow{\mathrm{tick}} R'$. Now, since $E \approx_t F$ there exist G, G', F' s.t. $F \stackrel{\tau}{\Longrightarrow} G \xrightarrow{\mathrm{tick}} G' \stackrel{\tau}{\Longrightarrow} F'$ and $E' \approx_t F'$. This implies

that there exists E'' s.t. $E \stackrel{\tau}{\Longrightarrow} E''$ and $E'' \approx_t G$. But, since we are considering processes which enjoy the maximal progress assumption (see Lem. 1) and $E \stackrel{\text{tick}}{\longrightarrow}$ it must be that $E'' = E$. Hence, $E \approx_t G$. Since also $G \stackrel{\text{tick}}{\longrightarrow}$, we get that for every action $l \in \mathcal{L}$, we have that $E \stackrel{l}{\longrightarrow}$ iff $G \stackrel{l}{\longrightarrow}$. This implies that $G||R \stackrel{\text{tick}}{\longrightarrow} G'||R' \stackrel{\tau}{\Longrightarrow} F'||R'$, and finally, we get $(E'||R', F'||R') \in \mathcal{R}$. ■

Proposition 6 : If $P, Q \in$ tSBSNNI, then $P||Q \in$ tSBSNNI.

Proof: We have to prove that if P and Q are in tSBSNNI, then for all of their derivatives P' and Q', we have that $P'||Q' \in$ tBSNNI, i.e., that $P'||Q'/H \approx_t P'||Q'\backslash H$. Note that by Lemma 2, also all of the derivatives of a tSBSNNI process are in tSBSNNI. Hence, we can actually obtain the result by showing that the following relation is a timed weak bisimulation:

$$\mathcal{R} = \{(E||F/H, E||F\backslash H) \mid E, F \in \text{tSBSNNI}\}.$$

We prove this fact by inspection of possible cases. Assume $(E||F/H, E||F\backslash H) \in \mathcal{R}$. Let us consider the possible moves from $E||F/H$. (As usual, we consider only the most interesting cases).

- $E||F/H \stackrel{\tau}{\longrightarrow} E'||F'/H$, then if $E \stackrel{a}{\longrightarrow} E'$, $F \stackrel{a}{\longrightarrow} F'$ with $a \in$ Act we have $E||F\backslash H \stackrel{\tau}{\longrightarrow} E'||F'\backslash H$ and $(E'||F'/H, E'||F'\backslash H) \in \mathcal{R}$.
- If $E||F/H \stackrel{\tau}{\longrightarrow} E'||F/H$, with $E \stackrel{h}{\longrightarrow} E'$, then since $E/H \approx_t E\backslash H$ there exists E_1 s.t. $E\backslash H \stackrel{\tau}{\Longrightarrow} E_1\backslash H$ and $E_1\backslash H \approx_t E'/H \approx_t E_1/H$, the last equality holding because E_1 is tSBSNNI. Hence, we also have $E||F\backslash H \stackrel{\tau}{\Longrightarrow} E_1||F\backslash H$ and we can now prove that $E'||F/H \approx_t \mathcal{R} \approx_t E_1||F\backslash H$. It is sufficient to note that $(E_1||F/H, E_1||F\backslash H) \in \mathcal{R}$ and that, by Lemma 7, we have

$$\begin{aligned} E'||F/H \approx_t & E'/H||F/H \approx_t E_1/H||F/H \approx_t \\ & \approx_t E_1||F/H \mathcal{R} E_1||F\backslash H. \end{aligned}$$

  So, up to bisimulation, we get $(E'||F/H, E'||F\backslash H) \in \mathcal{R}$.
- If $E||F/H \stackrel{\text{tick}}{\longrightarrow} E'||F'/H$, then $E \stackrel{\text{tick}}{\longrightarrow} E'$ and $F \stackrel{\text{tick}}{\longrightarrow} F'$. Since $E, F \in$ tSBSNNI, it must hold for all $h \in H$ that $E \stackrel{h}{\not\longrightarrow}$, and the same is also true for F. Hence, $E||F\backslash H \stackrel{\text{tick}}{\longrightarrow} E'||F'\backslash H$ and $(E'||F'/H, E'||F'\backslash H) \in \mathcal{R}$.

Let us now consider the moves from $E||F\backslash H$.

- If $E||F\backslash H \stackrel{\text{tick}}{\longrightarrow} E'||F'\backslash H$, we have that $E \stackrel{\text{tick}}{\longrightarrow} E'$ and $F \stackrel{\text{tick}}{\longrightarrow} F'$. Moreover, we get $E\backslash H \stackrel{\text{tick}}{\longrightarrow} E'\backslash H$ and $F\backslash H \stackrel{\text{tick}}{\longrightarrow} F'\backslash H$. Since $E \in$ tSBSNNI we have $E\backslash H \approx_t E/H$, from which it follows $E/H \stackrel{\tau}{\Longrightarrow} E_1/H \stackrel{\text{tick}}{\longrightarrow} E_1'/H \stackrel{\tau}{\Longrightarrow} E_2/H \approx_t E'\backslash H$.

  Analogously for F/H, we have $F/H \stackrel{\tau}{\Longrightarrow} F_1/H \stackrel{\text{tick}}{\longrightarrow} F_1'/H \stackrel{\tau}{\Longrightarrow} F_2/H \approx_t F'\backslash H$. Finally, we get $E||F/H \approx_t E/H||F/H \stackrel{\tau}{\Longrightarrow} E_1||F_1/H \stackrel{\text{tick}}{\longrightarrow} E_1'/H||F_1'/H \stackrel{\tau}{\Longrightarrow} E_2/H||F_2/H \approx_t E'/H||F'/H \approx_t E'||F'/H$; hence, up to bisimulation, $(E'||F'/H, E'||F'\backslash H) \in \mathcal{R}$.

■

Proposition 7: tBNDC is not compositional.

Proof: Let $E_1 = l.(\tau + \tau.l.\overline{h}) + l.h.l$ and $E_2 = l.(\tau + \tau.l_1) + l.h.l_1$. We want to prove that $E_1, E_2 \in$ tBNDC but $E_1||E_2$ is not in tBNDC. The fact that both processes are tBNDC can be proved similarly as in the proof of Proposition 5. To prove that the composition is not tBNDC, let us consider Π to be $\iota(h.\iota(\mathbf{0}))$. Then, $(E_1||E_2||\Pi)\backslash H$ can perform weakly an action l by reaching the $(l.\overline{h}||E_2||\Pi) \setminus H$. Similarly, $(E_1||E_2)\backslash H \stackrel{l}{\Longrightarrow} (l.\overline{h}||E_2) \setminus H$. Now, $(l.\overline{h}||E_2||\Pi) \setminus H \stackrel{l}{\longrightarrow} C_1 = (l.\overline{h}||h.l_1||\Pi) \setminus H$ and no bisimilar state can be reached from $(l.\overline{h}||E_2) \setminus H$. Indeed, the only two possible moves are:

- $(l.\overline{h}||E_2) \setminus H \stackrel{l}{\longrightarrow} C_2 = (l.\overline{h}||\tau + \tau.l_1) \setminus H$, but then $C_2 \stackrel{\tau}{\longrightarrow} C_2' = (l.\overline{h}||\mathbf{0})\backslash H$ from which only l is possible, and there exists a Π (e.g., $\Pi = \overline{h}.\mathbf{0}$), such that no equivalent state can be found from C_1 via a τ move.
- $(l.\overline{h}||E_2) \setminus H \stackrel{l}{\longrightarrow} C_3 = (l.\overline{h}||h.l_1) \setminus H$, but then there exists a Π (e.g., $\Pi = h.\mathbf{0}$) such that C_1 may be unable to perform l_1 (when the synchronization takes place between Π and the left component $l.\overline{h}$), while C_3 will certainly do that. ■

E. An Example With Partial Model Checking

The equational definition D is the following:

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

The equational specification $D// \setminus H//E$ obtained from the partial evaluation is

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

Since we are considering high users, then, we can freely substitute $\langle l\rangle\mathbf{T}$ with $\mathbf{F}$ in the previous equational definition. Indeed such processes are not able to satisfy $\langle l\rangle\mathbf{T}$. Moreover, after several simplifications (e.g., Z_l is equivalent to $\mathbf{T}$ and this implies that also Y_l is equivalent to $\mathbf{T}$), we obtain

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

By partially evaluating C with respect to the previous equational definition we obtain the equational definition $(D// \setminus H//E)//C$

$$\begin{aligned} Y_{E,C} &=_\nu [\tau]Y_{E,C} \wedge [h]Y_{E,C} \wedge [\overline{h}]Y_{E,C} \wedge Y_{h.l.\mathbf{0},C} \\ Y_{h.l.\mathbf{0},C} &=_\nu [\tau]Y_{h.l.\mathbf{0},C} \wedge [h]Y_{h.l.\mathbf{0},C} \wedge Z_{h.l.\mathbf{0},C} \\ Z_{h.l.\mathbf{0},C} &=_\mu \langle\tau\rangle Z_{h.l.\mathbf{0},C} \vee \langle h\rangle Z_{h.l.\mathbf{0},C} \vee \mathbf{T} \end{aligned}$$

which is equivalent to $\mathbf{T}$ since, $Z_{h.l.\mathbf{0},C}$ is trivially equivalent to $\mathbf{T}$ and for every a, $Y =_\nu [a]Y \wedge \mathbf{T}$ is equivalent to $\mathbf{T}$. (Please

note there are existing tools that perform similar reductions, e.g., see [21] and [25]).

ACKNOWLEDGMENT

We would like to thanks the anonymous referees for their useful comments.

REFERENCES

[1] A. Aldini, "Probabilistic information flow in a process algebra," in *Proc. CONCUR'01*, vol. LNCS 2154, 2001, pp. 152–168.
[2] R. Alur and D. L. Dill, "A theory of timed automata," *Theor. Comput. Sci.*, vol. 126, pp. 183–235, 1994.
[3] H. R. Andersen, "Partial model checking (extended abstract)," in *Proc. of LICS'95*, 1995, pp. 398–407.
[4] The Concurrency Workbench-NC. [Online]. Available: http://www.cs.sunysb.edu/~cwb/
[5] F. Corradini, D. D'Ortenzio, and P. Inverardi, "On the relationships among four timed process algebras," *Fundamenta Informaticae*, vol. 34, pp. 1–19, 1999.
[6] A. Durante, R. Focardi, and R. Gorrieri, "A compiler for analysing cryptographic protocols using noninterference," *ACM Trans. Softw. Eng. Methodology (TOSEM)*, vol. 9, pp. 489–530, 2000.
[7] E. Felten and M. Schneider, "Timing attacks on web privacy," in *Proc. 7th ACM Conf. Computer Communications Security*, 2000, pp. 25–32.
[8] R. Focardi and R. Gorrieri, "A classification of security properties," *J. Comput. Security*, vol. 3, pp. 5–33, 1995.
[9] ——, "The compositional security checker: A tool for the verification of information flow security properties," *IEEE Trans. Softw. Eng.*, vol. 27, pp. 550–571, 1997.
[10] ——, "Classification of security properties. Part I: Information flow," in *Foundations of Security Analysis and Design*. New York: Springer-Verlag, 2001, vol. LNCS 2171, pp. 331–396.
[11] R. Focardi, R. Gorrieri, and F. Martinelli, "Information flow analysis in a discrete time process algebra," in *Proc. 13th IEEE Computer Security Foundations Workshop (CSFW13)*, 2000, pp. 170–184.
[12] ——, "Noninterference for the analysis of cryptographic protocols," in *Proc. Int. Colloquium Automata, Languages, Programming (ICALP'00)*, vol. LNCS 1853, 2000, pp. 354–372.
[13] ——, "A comparison of three authentication properties," *Theor. Comput. Sci.*, to be published.
[14] R. Focardi and F. Martinelli, "A uniform approach for the definition of security properties," in *Proc. World Congress Formal Methods (FM'99)*, vol. LNCS 1708, 1999, pp. 794–813.
[15] R. J. van Glabbeek, "The linear time—Branching time spectrum," in *Handbook of Process Algebra*. New York: Elsevier, 2001, ch. 1.
[16] J. Goguen and J. Meseguer, "Security policy and security models," in *Proc. 1982 Symp. Security Privacy*, 1982, pp. 11–20.
[17] J. W. Gray III and P. F. Syverson, "A logical approach to multilevel security of probabilistic systems," in *Proc. 1992 IEEE Computer Society Symp. Security Privacy*, pp. 164–176.
[18] J. F. Groote, "Transition system specifications with negative premises," *Theor. Comput. Sci.*, vol. 118, pp. 263–299, 1993.
[19] M. Hennessy and T. Regan, "A temporal process algebra," *Inform. Comput.*, vol. 117, pp. 221–239, 1995.
[20] R. Lanotte, A. Maggiolo-Schettini, and S. Tini, "Privacy in real-time systems," in *Proc. MTCS'01, Electronic Notes in Theoretical Computer Science*, vol. 52.
[21] J. Lind-Nielsen, "Mudiv: A Program Performing Partial Model Checking," M.S. thesis, Dept. Inform. Technol., Technical Univ. Denmark, Denmark, 1996.
[22] D. Marchignoli and F. Martinelli, "Automatic verification of cryptographic protocols through compositional analysis techniques," in *Proc. Int. Conf. Tools and Algorithms for the Construction and the Analysis of Systems (TACAS'99)*, vol. 1579, Lecture Notes in Computer Science, 1999.
[23] F. Martinelli, "Formal Methods for the Analysis of Open Systems With Applications to Security Properties," Ph.D. dissertation, University of Siena, Italy, 1998.
[24] ——, "Partial model checking and theorem proving for ensuring security properties," in *Proc. CSFW'98*, 1998, pp. 44–52.
[25] ——, "Languages for description and analysis of authentication protocols," in *Proc. 6th Italian Conf. Theoretical Computer Science*, 1998, pp. 306–315.
[26] ——, "Toward automatic synthesis of systems without informations leaks," in *Proc. Workshop on Issues in Security (WITS)*, 2000.
[27] R. Milner, "Operational and algebraic semantics of concurrent processes," in *Handbook of Theoretical Computer Science*, J. van Leewen, Ed. Canbridge, MA: MIT Press, 1990, vol. B, Formal Models and Semantics, ch. 19, pp. 1201–1242.
[28] M. Muller-Olm, "Derivation of characteristic formulae," *Electron. Notes Theor. Comput. Sci.*, vol. 18, 1998.
[29] X. Nicollin and J. Sifakis, "An overview and synthesis on timed process algebras," in *Proc. Real-Time: Theory in Practice*, vol. 600, Lecture Notes in Computer Science, J. W. de Bakker, C. Huizing, W. P. de Roever, G. Rozenberg, and editors, Eds., 1992, pp. 526–548.
[30] P. Y. A. Ryan, "Mathematical models of computer security," in *Foundations of Security Analysis and Design*. New York: Springer-Verlag, 2001, pp. 1–62.
[31] S. Schneider, *Concurrent and Real-Time Systems: The CSP Approach*. New York: Wiley, 1999.
[32] ——, "Analysing time-dependent security properties in CSP using PVS," in *Procs. ESORICS*, vol. LNCS 1895, 2000, pp. 222–237.
[33] B. Steffen and A. Ingólfsdóttir, "Characteristic formulae for processes with divergence," *Inform. Comput.*, vol. 110, no. 1, pp. 149–163, 1994.
[34] C. Stirling, "Modal and temporal logics for processes," in *Logics for Concurrency: Structures Versus Automata*, 1996, vol. LNCS 1043, pp. 149–237.
[35] R. S. Street and E. A. Emerson, "The propositional μ—Calculus is elementary," in *Proc. 11th Int. Colloquium Automata, Languages, Programming*, vol. LNCS 172, 1984, pp. 465–472.
[36] ——, "An automata theoretic procedure for the propositional μ—Calculus," *Inform. Comput.*, vol. 81, pp. 249–264, 1989.
[37] I. Ulidowski and S. Yuen, "Extending process languages with time," in *Proc. 6th Int. Conf. Algebraic Methodology Software Technology*, vol. 1349, Lecture Notes in Computer Science, 1997, pp. 524–538.

Riccardo Focardi received the M.S. and Ph.D. degrees from Bologna University, Bologna, Italy, in 1993 and 1999, respectively.

He is an Assistant Professor of computer science, University of Venice, Italy. His research activity mainly focuses on security, formal methods, static analysis, and verification tools.

Roberto Gorrieri received the M.S. and Ph.D. degrees from Pisa University, Pisa, Italy, in 1986 and 1991, respectively.

He is a Professor of computer science, University of Bologna, Bologna, Italy. He has authored more than 90 papers on formal methods, theory of concurrency, and foundations of security.

Fabio Martinelli received the M.S. degree from Pisa University, Pisa, Italy, in 1994, and the Ph.D. degree from Siena University, Siena, Italy, in 1999.

He is a Researcher in the Information Security Group, Institute of Informatics and Telematics (IIT) of the National Research Council (CNR), Italy. He has authored more than 20 papers on formal methods, foundations of security, and verification tools.