Timed Information Flow Logic for Timed Automata

Ruggero Lanotte[1], Andrea Maggiolo-Schettini[1], and Simone Tini[2]

[1] Dipartimento di Informatica, Università di Pisa, Pisa, Italy
[2] Dipartimento di Scienze CC.FF.MM., Università dell'Insubria, Como, Italy

Abstract. We study the problem of information flow in real-time systems described by Timed Automata. We distinguish between secret and observable actions of automata, we introduce a logic to express information flow properties as properties of the languages accepted by automata, and we give an algorithm to check these properties.

1 Introduction

A requirement of mobile code is that it must guarantee some kind of security to clients executing it. One of the security requirements is the client's *privacy*, i.e. that executing mobile code does not imply leaking of private information.

Several papers (see, among the others, [8–10, 15, 14]) consider *two-level* systems, where the *high level* (or *secret*) behavior is distinguished from the *low level* (or *observable*) one. In these papers, systems respect privacy if there is no *information flow* from the high level to the low level. This means that the secret behavior cannot influence the observable one, or, equivalently, no information on the observable behavior permits to infer information on the secret one.

Our aim is to study the problem of information flow in real-time systems. To this purpose, we consider the framework of Timed Automata [4], which are one of the most studied models for real-time systems. When using this formalism, the behaviors of a system are described by infinite *timed words*, i.e. infinite sequences of pairs (action performed, time of firing). In describing two-level systems, we distinguish between high-level and low-level actions. We formulate a *Timed Information Flow Logic* (TIFL) that permits to describe whether the observation of a sequence of observable actions in certain intervals of time implies that some non observable actions have been or have not been performed. So, TIFL permits to describe behaviors giving rise to information flow.

Automata for the specification of security problems have been used for example in [12, 13]. Introducing specific logics for specific problems of security is known in the literature [1–3]. By these automata and logics one can describe systems with the assumption of discrete time. This restriction is too strong for describing a network environment with real time. Actually, in this case formalisms with explicit time and dense time assumption are necessary. Timed Automata combine simplicity of a model theoretic approach with the request of dense time. Now, logics proposed for describing and proving general properties of Timed Automata

with dense time are undecidable [5], hence the interest for logics tailored to deal with specific problems, which besides making easier to express the properties of interest, enjoy decidability. TIFL permits expressing security properties related to time assumed to be dense, and we prove its decidability.

2 A problem of privacy in the web

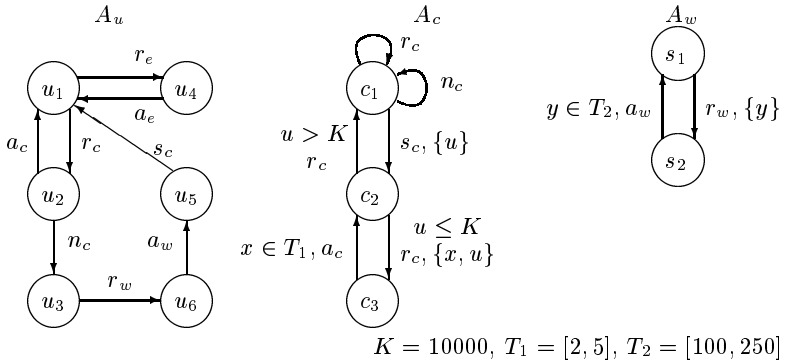

Fig. 1. The web system

Let us assume a user's browser that interacts with its cache, with a given site w, and with other sites which may be treated as one only site e. In Fig. 1 we model this system by Timed Automata. The automaton A_u represents the behavior of the user. This can perform a request r_e to the site e and then it receives the answer a_e. It can perform also a request r_c to the cache to obtain a web page in w. If the requested page is cached, then the cache gives a positive answer a_c. Otherwise, the cache gives a negative answer n_c, the user downloads the page from w (actions r_w and a_w) and, then, the page is cached (action s_c).

Automaton A_w represents the site w. The time elapsed between a request r_w (which resets clock y) and an answer a_w is in the interval $T_2 = [100, 250]$.

Automaton A_c represents the cache. When a page in w is requested by the user (action r_c) and the page is not yet in the cache (state c_1), the cache gives a negative answer (action n_c). In this case the user downloads the page, which is cached (action s_c, which resets clock u). Now, if the page is not requested for a time greater than 10000 the page is removed from the cache (such a deadline is checked by clock u) and the next request r_c causes A_c to reach state c_1. When the page is in the cache (state c_2 is active and $u \leq K$ holds), the time elapsed between a request r_c and an answer a_c is in the interval $T_1 = [2, 5]$.

Now, the only observable actions for e are r_e and a_e, since it cannot observe interactions between the browser and the cache and between the browser and w.

In [7] it is shown that, when the user visits the site e, e can attack the browser and infer whether it has recently visited some web page in w or not, thus violating the privacy of the user. In fact, assume that e contains an applet

that, when executed, causes a request of the page in w and, then, a request to e itself. When the user's browser downloads the page of e, it performs the applet. Therefore, if e receives the original request and the request caused by the applet within 100 units of time, it infers that no communication between the user and w has happened in the meantime, i.e. that the page was in the cache of the user. In fact, the browser takes at least 100 units of time to download a page from w.

3 LH Timed Automata

3.1 Security alphabet and timed words

A *security alphabet* is a pair of two disjoint sets $(Low, High)$, where Low (resp. $High$) contains the *low actions*, (resp. *high actions*), which can be performed by the system and can (resp. cannot) be observed by the external environment. We let l, h and a range over Low, $High$ and $Low \cup High$, respectively.

Given any *time domain* T (non-negative rational numbers, or non-negative real numbers, as examples), we consider (possibly finite) *timed sequences* of the form $(t_1, a_1) \dots (t_n, a_n) \dots$, with $t_i \in T$ and $a_i \in (Low \cup High)$, describing the temporal behavior of a system that performs action a_1 at time t_1 and action a_n precisely t_n units of time after action a_{n-1}. (Note that Alur and Dill describe the same behavior with the sequence $(t_1, a_1) \dots (\sum_{1 \leq i \leq n} t_i, a_n) \dots$.)

For a finite timed sequence $\omega_1 = (t_1, a_1) \dots (t_n, a_n)$ and any timed sequence $\omega_2 = (t'_1, a'_1) \dots (t'_n, a'_n) \dots$, $\omega_1\omega_2$ denotes $(t_1, a_1) \dots (t_n, a_n)(t'_1, a'_1) \dots (t'_n, a'_n) \dots$.

An infinite timed sequence $(t_1, a_1) \dots (t_n, a_n) \dots$ is a *timed word* if it satisfies the *time progress property*, i.e., $\forall t \in T$ there is some index i with $\sum_{h=1}^{i} t_h > t$.

For a timed word $\omega = (t_1, a_1) \dots (t_n, a_n) \dots$, we denote with ω_{Low} the *observable part* of ω, i.e. the (possibly finite) sequence $(t'_1, a_{i_1}) \dots (t'_n, a_{i_n}) \dots$ s.t.:

- $a_{i_1}, \dots, a_{i_n}, \dots \in Low$ and, for each $1 \leq k < i_1$ and $i_j < k < i_{j+1}$, $a_k \in High$
- $t'_1 = \sum_{h=1}^{i_1} t_h$ and, for each $j > 1$, $t'_j = \sum_{h=i_{j-1}+1}^{i_j} t_h$.

3.2 Clock valuations and clock constraints

We assume a set X of variables measuring time, called *clocks*, ranged over by x. They increase uniformly with time when an automaton is in any state.

A *clock valuation* over X is a map $v : X \to T$ from clocks to time values. For a clock valuation v and a time value t, let $v + t$ denote the clock valuation s.t. $(v + t)(x) = v(x) + t$. Moreover, for a subset of clocks $Y \subseteq X$, let $v[Y]$ denote the clock valuation s.t. $v[Y](x) = 0$, if $x \in Y$, and $v[Y](x) = v(x)$, otherwise.

The set of *clock constraints* over X, denoted $\Theta(X)$, is defined by the following grammar, where θ ranges over $\Theta(X)$, $x \in X$, $c \in T$ and $\# \in \{<, \leq, =, \neq, >, \geq\}$:

$$\theta ::= x \,\#\, c \mid \theta \wedge \theta \mid \neg\theta \mid \theta \vee \theta \mid true$$

We write $v \models \theta$ when *the clock valuation v satisfies the clock constraint θ*. More precisely, $v \models x \,\#\, c$ iff $v(x) \,\#\, c$, $v \models \theta_1 \wedge \theta_2$ iff both $v \models \theta_1$ and $v \models \theta_2$, $v \models \theta_1 \vee \theta_2$

iff either $v \models \theta_1$ or $v \models \theta_2$, $v \models \neg\theta$ iff $v \not\models \theta$, and $v \models true$.

3.3 The formalism

Definition 1. *Assume a security alphabet* $(Low, High)$. *An* LH Timed Automaton *is a tuple* $\mathcal{A} = (A_1, \ldots, A_m)$, *where, for each* $1 \leq i \leq m$, $A_i = (Low_i, High_i, Q_i, q_i^0, X_i, \delta_i)$ *is a* sequential automaton, *with:*

- *a security alphabet* $(Low_i, High_i)$ *with* $Low_i \subseteq Low$ *and* $High_i \subseteq High$
- *a finite set of* states Q_i *such that* $Q_1, \ldots, Q_m$ *are pairwise disjoint*
- *an* initial state $q_i^0 \in Q_i$
- *a set of* clocks X_i *such that* $X_1, \ldots, X_m$ *are pairwise disjoint*
- *a set of* transitions $\delta_i \subseteq Q_i \times \Theta(X_i) \times (Low_i \cup High_i) \times 2^{X_i} \times Q_i$.

Intuitively, a transition (q, θ, a, Y, q') in A_i fires when state q is active, the clock valuation of A_i satisfies the clock constraint θ, and action a is performed. In such a case, state q' is entered and the clocks in Y are reset.

A *configuration* of $\mathcal{A}$ is a tuple $s = ((q_1, v_1), \ldots, (q_m, v_m))$ such that, for each $1 \leq i \leq m$, q_i is a state in Q_i and v_i is a clock valuation over clocks X_i.
The *initial configuration* s_0 is the tuple $((q_1^0, v_1^0), \ldots, (q_m^0, v_m^0))$, with q_i^0 the initial state of A_i and with v_i^0 the valuation such that $v_i^0(x) = 0$ for each clock $x \in X_i$.

There is a *step* from $s = ((q_1, v_1), \ldots, (q_m, v_m))$ to $s' = ((q_1', v_1'), \ldots, (q_m', v_m'))$ at time t with action $a \in High \cup Low$, written $s \rightarrow_t^a s'$, iff, for each $1 \leq i \leq m$, either $a \in Low_i \cup High_i$ and there is a transition $(q_i, \theta_i, a, Y_i, q_i') \in \delta_i$ with $v_i + t \models \theta_i$ and $v_i' = (v_i + t)[Y_i]$, or $a \notin Low_i \cup High_i$, $q_i' = q_i$ and $v_i' = v_i + t$.

A *run* r is an infinite sequence of steps $r = s_1 \rightarrow_{t_1}^{a_1} s_2 \ldots s_n \rightarrow_{t_n}^{a_n} s_{n+1} \ldots$. It is *accepting* iff s_1 is the initial configuration (we assume implicitly that all states are *final* according to [4]). The set of the accepted runs is denoted $\mathcal{R}(\mathcal{A})$.

We denote with ω^r the timed word $(t_1, a_1) \ldots (t_n, a_n) \ldots$ described by r.

A timed word ω is *accepted* by $\mathcal{A}$ if there is an accepting run r with $\omega = \omega^r$. The *language of* A is denoted $\mathcal{L}(\mathcal{A})$ and is the set of words accepted by $\mathcal{A}$.

By application of a cartesian product construction, any automaton can be transformed into an equivalent one having only one sequential component.

Proposition 1. *For any automaton* $\mathcal{A}$ *there exists an automaton* $\mathcal{A}'$ *composed by one only sequential component such that* $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}')$.

Sometimes, given the finite timed sequence $\omega' = (t_1, a_1) \ldots (t_n, a_n)$, we denote with $s_1 \stackrel{\omega'}{\rightarrow} s_{n+1}$ the sequence of steps $s_1 \rightarrow_{t_1}^{a_1} \ldots s_n \rightarrow_{t_n}^{a_n} s_{n+1}$.

Moreover, we will write $\rightarrow_{t_1}^{a_1} \ldots \rightarrow_{t_n}^{a_n} \ldots$ and $\stackrel{\omega}{\rightarrow}$ to denote a run, if we are not interested in the states.

Finally, we say that a run r is equal to $\stackrel{\omega}{\rightarrow} r'$ if r performs the finite sequence ω and, then, it coincides with the run r'.

3.4 Region graph

Let us recall the notion of *region graph* of a timed automaton. By Prop.1 we can consider, without loss of generality, automata with only one sequential component. The notion of region graph was introduced in [4] to give an algorithm that

checks whether the language of a given automaton is empty. We exploit the region graph to give an algorithm that checks whether a given automaton satisfies a given TIFL formula. To this purpose, it is sufficient to consider automata with clock constraints permitting only comparisons with integer constants (see [4] or the beginning of Sect. 6). Moreover, we assume that all states of the automata have departing transitions.

Let us consider the equivalence relation $\sim$ over clock valuations containing precisely the pairs (v, v') such that:

- for each clock x, either $\lfloor v(x) \rfloor = \lfloor v'(x) \rfloor$, or both $v(x)$ and $v'(x)$ are greater than c_x, with c_x the largest integer appearing in clock constraints over x.
- for each pair of clocks x and y with $v(x) \leq c_x$ and $v(y) \leq c_y$, $fract(v(x)) \leq fract(v(y))$ iff $fract(v'(x)) \leq fract(v'(y))$ ($fract(\_)$ is the fractional part).
- for each clock x with $v(x) \leq c_x$, $fract(v(x)) = 0$ iff $fract(v'(x)) = 0$.

As proved in [4], $v \sim v'$ implies that, for any $\theta \in \Theta(X)$, $v \models \theta$ iff $v' \models \theta$.

A *clock region* is an equivalence class induced by $\sim$. The set of the clock regions is finite. We denote by $[v]$ the clock region to which v belongs.

A *region* is a pair $(q, [v])$, with q a state and $[v]$ a clock region. The *initial region* is the pair $(q^0, [v^0])$ with q^0 the initial state and $v^0(x) = 0$, for each $x \in X$.

The *region graph* $RG(\mathcal{A})$ is a graph having the regions of $\mathcal{A}$ as set of nodes and having an edge $\langle (q, [v]), a, (q', [v']) \rangle$ if and only if, for some pair of valuations $v \in [v]$ and $v' \in [v']$, $(q, v) \rightarrow_t^a (q', v')$ for some time t.

Let us introduce now the notion of dense set of runs.

Definition 2. *A set of runs $\mathcal{R}$ of an automaton $\mathcal{A}$ is* dense *iff for each configuration (q, v) reachable by some run in $\mathcal{R}$, it holds that also any configuration (q, v') is reachable by some run in $\mathcal{R}$ if the following requirements are satisfied:*

- *for each clock x, it holds that $\lfloor v(x) \rfloor = \lfloor v'(x) \rfloor$ and $\lceil v(x) \rceil = \lceil v'(x) \rceil$,*
- *for each pair of clocks x and y it holds that $fract(v(x)) \# fract(v(y))$ if and only if $fract(v'(x)) \# fract(v'(y))$, where $\# \in \{<, =, >\}$.*

The meaning of a set of runs $\mathcal{R}$ to be dense is that if a configuration (q, v) of $\mathcal{A}$ is reachable by some run in $\mathcal{R}$, then every configuration (q, v') with the same integer part of valuation of clocks and preserving relationships between fractional parts of different clocks is also reachable by some run in $\mathcal{R}$.

Proposition 2. *The set of runs $\mathcal{R}(\mathcal{A})$ of any automaton $\mathcal{A}$ is dense.*

4 Information flow and LH Timed Automata

Let us consider the finite timed sequences $(t_1, l_1) \ldots (t_n, l_n)$ describing finite observable behaviors, with $t_1, \ldots, t_n$ time values and $l_1, \ldots, l_n$ low actions, that are denoted by the following grammar:

$$D ::= (I, L) \mid D_1 + D_2 \mid D_1 D_2 \mid D_1^+$$

where I is an interval with an upper bound and $L \subseteq Low$ is a set of low actions.

Expression (I, L) denotes the set of sequences (t, l) with $t \in I$ and $l \in L$. Regular expressions $D_1 + D_2$, $D_1 D_2$ and D_1^+ denote union, concatenation and iteration, respectively. Formally, the function $\mathcal{D}$ defined below associates a set of finite timed sequences with each regular expression:

$$\mathcal{D}[(I, L)] = \{(t, l) \mid t \in I \text{ and } l \in L\}$$
$$\mathcal{D}[D_1 + D_2] = \mathcal{D}[D_1] \cup \mathcal{D}[D_2]$$
$$\mathcal{D}[D_1 D_2] = \{d_1\, d_2 \mid d_1 \in \mathcal{D}[D_1] \text{ and } d_2 \in \mathcal{D}[D_2]\}$$
$$\mathcal{D}[D_1^+] = \{d_1 \ldots d_k \mid k \geq 1 \text{ and } d_i \in \mathcal{D}[D_1] \text{ for each } 1 \leq i \leq k\}$$

We say that two runs r and r' *have indistinguishable observable start*, written $r \equiv_{os} r'$, iff $(\omega^r)_{Low} = (t, l) \ldots$ and $(\omega^{r'})_{Low} = (t', l') \ldots$ implies $t = t'$ and $l = l'$, i.e. one cannot distinguish r and r' by observing their first observable action.

Now, let us consider the properties over a given run r and a given set of runs $\mathcal{R}$, with $r \in \mathcal{R}$, that are described by the following grammar (where $h \in High$):

$$\pi ::= true \mid h \mid \pi^{\forall} \mid \neg\pi \mid \pi_1 \wedge \pi_2 \mid \pi_1 \vee \pi_2$$

A run r and a set of runs $\mathcal{R}$ with $r \in \mathcal{R}$ satisfy the property h, written $(r, \mathcal{R}) \models h$, iff r reads h before reading any low action, i.e. r starts by reading a sequence of actions $h_1, \ldots, h_n, l$, with $h_1, \ldots, h_n \in High$ and $l \in Low$, and it holds that $h \in \{h_1, \ldots, h_n\}$. Moreover, r and $\mathcal{R}$ satisfy the property $\pi^{\forall}$, written $(r, \mathcal{R}) \models \pi^{\forall}$, iff for each run $r' \in \mathcal{R}$ such that $r' \equiv_{os} r$, it holds that $(r', \mathcal{R}) \models \pi$. The relation $\models$ is inductively defined as follows:

$(r, \mathcal{R}) \models true$
$(r, \mathcal{R}) \models h$ iff $r = \rightarrow_{t_1}^{h_1} \ldots \rightarrow_{t_n}^{h_n} \rightarrow_{t_{n+1}}^{l} \ldots$ implies $h \in \{h_1, \ldots, h_n\}$
$(r, \mathcal{R}) \models \pi^{\forall}$ iff for each run $r' \equiv_{os} r$ in $\mathcal{R}$ it holds that $(r', \mathcal{R}) \models \pi$
$(r, \mathcal{R}) \models \neg\pi$ iff $(r, \mathcal{R}) \not\models \pi$
$(r, \mathcal{R}) \models \pi_1 \wedge \pi_2$ iff both $(r, \mathcal{R}) \models \pi_1$ and $(r, \mathcal{R}) \models \pi_2$
$(r, \mathcal{R}) \models \pi_1 \vee \pi_2$ iff either $(r, \mathcal{R}) \models \pi_1$ or $(r, \mathcal{R}) \models \pi_2$

Relation $(r, \mathcal{R}) \models \pi$ may reveal information flow in the initial part of the runs $\mathcal{R}$. As an example, $(r, \mathcal{R}) \models h^{\forall}$, for some run r with $(\omega^r)_{Low} = (t, l) \ldots$, implies that if we consider any run r' in $\mathcal{R}$ and we observe the first low action l at time t, then we can infer that the high action h has been read before. In Section 5 we prove that the same information on the secret actions of r' can be inferred also if l is observed at any time t' lying in a suitable interval I_t containing t.

Now, let us introduce *Timed Information Flow Logic* (TIFL), whose formulae describe properties over sets of runs and can reveal information flow arising in any part of the runs. The grammar for formulae is the following:

$$\psi ::= \epsilon \mid \pi.\psi \mid D.\psi \mid \neg\psi \mid \psi_1 \wedge \psi_2 \mid \psi_1 \vee \psi_2$$

A set of runs $\mathcal{R}$ satisfies ϵ iff $\mathcal{R}$ is not empty.

A set of runs $\mathcal{R}$ satisfies property $\pi.\psi$ iff ψ is satisfied by the set of runs r in $\mathcal{R}$ such that $(r, \mathcal{R})$ satisfies π. The operator $\pi.\_$ permits to select the runs in $\mathcal{R}$ whose non observable initial behavior satisfies the property expressed by π.

A set of runs $\mathcal{R}$ satisfies property $D.\psi$ iff ψ is satisfied by the set of runs r such that $\stackrel{\omega}{\rightarrow} r \in \mathcal{R}$, $\omega_{Low} = d$ for some $d \in \mathcal{D}[D]$, and ω terminates with a low action. More precisely, the operator $D._{-}$ selects from $\mathcal{R}$ the runs r that follow the performance of a sequence ω whose low part is described by D.

Formally, we say that *a set of runs $\mathcal{R}$ satisfies a property ψ*, written $\mathcal{R} \models \psi$, iff the following requirements are fulfilled:

$$
\begin{array}{lll}
\mathcal{R} \models \epsilon & \text{iff} & \mathcal{R} \neq \emptyset \\
\mathcal{R} \models \pi.\psi & \text{iff} & \{r \in \mathcal{R} \mid (r, \mathcal{R}) \models \pi\} \models \psi \\
\mathcal{R} \models D.\psi & \text{iff} & \{r \mid \stackrel{\omega}{\rightarrow} r \in \mathcal{R}, \omega_{Low} \in \mathcal{D}[D], \text{ and } \omega \text{ terminates} \\
 & & \text{with a low action}\} \models \psi \\
\mathcal{R} \models \neg\psi & \text{iff} & \mathcal{R} \not\models \psi \\
\mathcal{R} \models \psi_1 \wedge \psi_2 & \text{iff} & \text{both } \mathcal{R} \models \psi_1 \text{ and } \mathcal{R} \models \psi_2 \\
\mathcal{R} \models \psi_1 \vee \psi_2 & \text{iff} & \text{either } \mathcal{R} \models \psi_1 \text{ or } \mathcal{R} \models \psi_2
\end{array}
$$

We say that *an automaton $\mathcal{A}$ satisfies a property ψ*, written $\mathcal{A} \models \psi$, iff $\mathcal{R}(\mathcal{A}) \models \psi$.

TIFL formulae permit to describe behaviors giving rise to information flow. As an example, let us assume that the runs of a given automaton satisfy formula $(I_1, \{l_1\}).\neg h_1.(I_2, \{l_2\}).\neg h_2.(I_3, \{l_3\}).\neg\epsilon$, for intervals I_1, I_2, I_3, low actions l_1, l_2, l_3 and high actions h_1, h_2. In such a case there is an information flow, since whenever we observe action l_i in interval I_i, for $1 \leq i \leq 3$, we are sure that either h_1 has been read between l_1 and l_2, or h_2 has been read between l_2 and l_3. In fact, let us consider all runs whose observable sequence of actions begins with $(t_1, l_1)(t_2, l_2)(t_3, l_3)$, with $t_i \in I_i$ for $1 \leq i \leq 3$. If we reject runs that do not read h_1 between l_1 and l_2 and, subsequently, we reject runs that do not read h_2 between l_2 and l_3, then we obtain the empty set of runs.

As another example, let us assume that the runs of a given automaton satisfy $(I_1, \{l_1\}).h^{\forall}.(I_2, \{l_2\}).\epsilon$. In such a case, since the set of runs selected by $(I_1, \{l_1\}).h^{\forall}.(I_2, \{l_2\}).{\_}$ is not empty, there is at least a time value t_2 in interval I_2 such that each run that reads l_1 in interval I_1 and l_2 at time t_2 performs h between l_1 and l_2, which is an information flow.

Besides describing behaviors giving rise to information flow, TIFL permit also to certify that suspect behaviors do not give rise to information flow. Both uses of TIFL are showed below.

Example 1. Let us consider the web system of Section 2. Let us take an arbitrary constant C. The following property holds:

$$([0, C], Low)^{+}.a_w.([0, 100), r_e).\neg\epsilon$$

Operator $([0, C], Low)^{+}._{-}$ creates a set of runs $\mathcal{R}$ from the runs of the automaton. Given any run $\stackrel{\omega}{\rightarrow} r$ of the automaton with ω terminating with a low action in a time $t \in [0, C]$, r is in $\mathcal{R}$. So, we consider a situation in which the automaton has advantaged of a certain number of steps and in $\mathcal{R}$ are described the possible steps that it shall perform. Operator $a_w._{-}$ selects the set $\mathcal{R}'$ of runs in $\mathcal{R}$ where the high action a_w appears before low actions. Now, in the runs in $\mathcal{R}'$ the first low action following a_w appears at least 100 units of time after the low actions

that appear before a_w, since a_w can happen only at least 100 units of time after a request r_w. Therefore, operator $([0,100), r_e).\_$ applied to $\mathcal{R}'$ gives the empty set of runs, which satisfy $\neg\epsilon$.

The property means that if the time elapsed between two communications with the web site e is in the interval $[0,100)$, then there has not been any communication with the web site w in the meantime. This information can be exploited by e to violate the privacy of the user. In fact, if e receives an original request from the user and, within 100 units of time, a request caused by the applet described in Section 2, it infers that no communication between the user and w has happened in the meantime, i.e. that the page was in the cache of the user. So, e is able to infer that the user has recently visited the page of site w.

Property $([0,C], Low)^+.a_w.([0,100), r_e).\neg\epsilon$ shows that we are able to detect behaviors violating privacy. We can also show that we are able to certify that privacy is not violated by a suspect behavior. To this purpose, in Fig.1 let us take $T_1 = T_2$ (T_1, T_2 are the intervals in Fig.1). We can enforce the property

$$([0,C], Low)^+.r_c.((a_c)^\forall \vee (a_w)^\forall).([0,C], Low)^+.\neg\epsilon$$

Operator $([0,C], Low)^+.\_$ gives a set of run $\mathcal{R}$ as above. Operator $r_c.\_$ selects the runs $\mathcal{R}'$ in $\mathcal{R}$ that perform r_c before a low action. Operator $((a_c)^\forall \vee (a_w)^\forall).\_$ selects both sets of runs $\mathcal{R}''$ and $\mathcal{R}'''$ in $\mathcal{R}'$ s.t., if the first low actions is performed at a given time t, then we are sure that a_c, in the case of $\mathcal{R}''$, or a_w, in the case of $\mathcal{R}'''$, is performed. But, since $T_1 = T_2$, both a_c and a_w can be performed before the low action at time t. Therefore, both $\mathcal{R}''$ and $\mathcal{R}'''$ are empty.

This property means that whenever we fix a time $t \in [0,C]$ separating two low actions, if there is a request r_c in the meantime then it holds neither that all runs perform a communication with the cache nor that all runs perform a communication with w. So, e is not able to infer whether the user has accessed the cache or the site w by observing its interaction with the user.

We conclude this section with giving the following result, which follows directly from the definition of relation $\models$.

Proposition 3. *Let $\mathcal{A}$ and $\mathcal{A}'$ be two automata such that $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}')$. For each formula ψ it holds that $\mathcal{A} \models \psi$ iff $\mathcal{A}' \models \psi$.*

5 Some properties of formulae

In this section we give some properties of formulae.

Lemma 1. *Let r, r' be runs in a set $\mathcal{R}$ with $(\omega^r)_{Low} = (t,l)\ldots$, $(\omega^{r'})_{Low} = (t',l')\ldots$, $l' = l$, $t, t' \in (c, c+1)$, with $c \in \mathbb{N}$. If $\mathcal{R}$ is dense, then $(r,\mathcal{R}) \models \pi^\forall$ iff $(r',\mathcal{R}) \models \pi^\forall$.*

Lemma 1 states that information expressed by π on the non observable part of the runs in $\mathcal{R}$ can be inferred (if $(r,\mathcal{R}) \models \pi^\forall$) or not (otherwise) independently of the point in the interval $(c, c+1)$ in which we observe the low action l.

We give now a notion of normal form for TIFL formulae, and we prove that any formula can be reduced to an equivalent normal form. We need, before, some more notions.

For a formula ψ, let us denote with Int_ψ the set of intervals $(c, c+1)$ and $[c, c]$ such that $c \in \mathbb{N}$ and $0 \leq c \leq max\{c' \mid c' \text{ is an upper bound of an interval in } \psi\}$.

Let us define the set Π of the *Pi-formulae* as the least set containing every formula $\pi.\psi$, every formula $\neg\psi$ with $\psi \in \Pi$, every formula $\psi_1 \wedge \psi_2$ with either $\psi_1 \in \Pi$ or $\psi_2 \in \Pi$, and every formula $\psi_1 \vee \psi_2$ with either $\psi_1 \in \Pi$ or $\psi_2 \in \Pi$.

Definition 3. *A TIFL formula ψ is a* normal form *iff:*

- *no subformula $\pi.\psi'$ appearing in ψ is such that $\psi' \in \Pi$*
- *each pair (I, L) appearing in ψ is such that $I \in Int_\psi$.*

Proposition 4. *Each TIFL formula ψ has an equivalent normal form.*

The following result will be exploited to prove the decidability of TIFL.

Given a formula π and an expression D, let op_π and op_D be the operators over sets of runs used to define the relation $\mathcal{R} \models \psi$ and such that:

- $op_\pi(\mathcal{R}) = \{r \in \mathcal{R} \mid (r, \mathcal{R}) \models \pi\}$
- $op_D(\mathcal{R}) = \{r \mid \stackrel{\omega}{\rightarrow} r \in \mathcal{R}, \omega_{Low} \in \mathcal{D}[D],$ and ω ends with a low action$\}$.

The theorem below shows that $op_\pi(\_)$ and $op_D(\_)$ preserves density of runs.

Theorem 1. *Let $op_1, \ldots, op_n$ be operators over sets of runs s.t., for each $1 \leq i \leq n$, $op_i = op_\pi$ for some property π, or $op_i = op_D$ for some expression D. For any automaton $\mathcal{A}$, the set of runs $\mathcal{R}_n = op_n(\ldots(op_1(\mathcal{R}(\mathcal{A})))\ldots)$ is dense.*

6 Decidability

In this section we give a procedure to decide whether a TIFL formula ψ is satisfied by a given automaton $\mathcal{A}$. We assume that all intervals appearing in ψ have naturals as bounds and that all clock constraints appearing in $\mathcal{A}$ permit only comparison with natural constants. This assumption is legal for two reasons:

- for each constant t, $\mathcal{A} \models \psi$ if and only if $\mathcal{A} \cdot t \models \psi \cdot t$, where:
  1) $\mathcal{A} \cdot t$ is obtained by replacing each constant c in clock constraints in $\mathcal{A}$ by constant $c\,t$. As it has been proved in [4], $\mathcal{A} \cdot t$ accepts the timed words $(t_1\,t, a_1)\ldots(t_n\,t, a_n)\ldots$ such that $(t_1, a_1)\ldots(t_n, a_n)\ldots$ is accepted by $\mathcal{A}$;
  2) $\psi \cdot t$ is obtained by replacing each interval I in ψ by $\{t'\,t \mid t' \in I\}$;
- there is a constant t such that $\mathcal{A} \cdot t$ and $\psi \cdot t$ satisfy our assumption.

Since we are dealing with dense intervals, the legality of our assumption does not mean that our real-time setting can be reduced to a discrete setting.

We say that formula ϵ argues on all observable words $(t_1, l_1)\ldots(t_n, l_n)\ldots$, with $t_1, \ldots, t_n \in T$ and $l_1, \ldots, l_n \in Low$. Formula $D.\psi$ argues on the observable words $d\omega$, with $d \in \mathcal{D}[D]$ and ω argued on by ψ. Moreover, $\neg\psi$ and $\pi.\psi$ argue

on the same observable words of ψ. Finally, $\psi_1 \vee \psi_2$ and $\psi_1 \wedge \psi_2$ argue on the observable words that are argued on by both ψ_1 and ψ_2. In general, ψ argues on ω iff ψ considers runs r whose observable part $(\omega^r)_{Low}$ coincides with ω.

First of all, we construct an automaton $\mathcal{A}^\psi$ that accepts precisely the timed words ω whose observable part ω_{Low} is argued on by ψ.

Then, we take the cartesian product of $\mathcal{A}$ and $\mathcal{A}^\psi$, denoted $\mathcal{A} \otimes \mathcal{A}^\psi$, that accepts the language $\mathcal{L}(\mathcal{A}) \cap \mathcal{L}(\mathcal{A}^\psi)$, i.e. the timed words in $\mathcal{L}(\mathcal{A})$ whose observable part is argued on ψ. Therefore, $\mathcal{R}(\mathcal{A})$ satisfies ψ iff $\mathcal{R}(\mathcal{A} \otimes \mathcal{A}^\psi)$ does.

Finally, we visit the region graph of $\mathcal{A} \otimes \mathcal{A}^\psi$ to check whether ψ holds.

6.1 Construction of $\mathcal{A}^\psi$

First of all we construct the automaton $\mathcal{A}^D$ that accepts precisely the finite timed sequences ω such that $\omega_{Low} \in \mathcal{D}[D]$ and ω terminates with a low action. The automaton has form (Q, q_0, δ, X, F), with $F \subset Q$ the set of *final states*.

The automaton $\mathcal{A}^{(I,L)}$ is in Fig. 2. It accepts a possibly empty sequence of high actions followed by a low action in L, which is read in the interval I. The automaton has no transition entering the initial state and no transition departing from the final state q_I. This property is satisfied by any $\mathcal{A}^D$ and permits a construction inductive w.r.t. D.

Note that the final state of $\mathcal{A}^{(I,L)}$ is marked by the interval I.

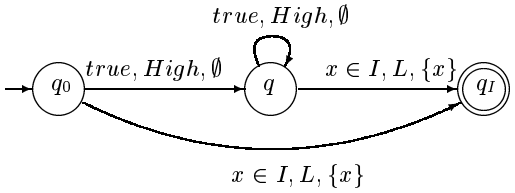

Fig. 2. The automaton $\mathcal{A}^{(I,L)}$

For a set of transitions δ and states q, q', let $\delta[q'/q]$ denote the set of transitions in δ modified by replacing the starting (resp. target) state with q', if the starting (resp. target) state is q. Moreover, for a set of states Q, let $\delta[Q/q]$ denote the set $\{\delta[q'/q] \mid q' \in Q\}$.

For $\mathcal{A}^{D_1} = (Q_1, q_0^1, \delta_1, \{x\}, F_1)$ and $\mathcal{A}^{D_2} = (Q_2, q_0^2, \delta_2, \{x\}, F_2)$, we define:

$$\mathcal{A}^{D_1+D_2} = (Q_1 \cup Q_2 \cup \{q_0\} \setminus \{q_0^1, q_0^2\}, q_0, \delta_1[q_0/q_0^1] \cup \delta_2[q_0/q_0^2], \{x\}, F_1 \cup F_2\})$$

$$\mathcal{A}^{D_1 D_2} = \langle Q_1 \cup Q_2 \setminus \{q_0^2\}, q_0^1, \delta_1 \cup \delta_2[\textstyle\bigcup_{q \in F_1} /q_0^2], \{x\}, F_2 \rangle$$

$$\mathcal{A}^{D_1^+} = \langle Q_1 \cup \{q'\}, q_0^1, \delta_1 \cup \{\langle q, \vartheta, a, Y, q' \rangle \mid \langle q, \vartheta, a, Y, q_f \rangle \in \delta_1 \text{ and } q_f \in F_1\} \cup \{\langle q', \vartheta, a, Y, q \rangle \mid \langle q_0^1, \vartheta, a, Y, q \rangle \in \delta_1\}, \{x\}, F_1 \rangle$$

Automaton $\mathcal{A}^{D_1+D_2}$ chooses between performing a run of $\mathcal{A}^{D_1}$ and performing a run of $\mathcal{A}^{D_2}$. Automaton $\mathcal{A}^{D_1 D_2}$ performs a run of $\mathcal{A}^{D_1}$ followed by a run of $\mathcal{A}^{D_2}$. Finally, automaton $\mathcal{A}^{D_1^+}$ performs a finite sequence of runs of $\mathcal{A}^{D_1}$.

We can define now $\mathcal{A}^\psi$. The automaton $\mathcal{A}^\epsilon$ is defined as follows:

$$\langle \{q_\epsilon\}, q_\epsilon, \{\langle q_\epsilon, true, Low \cup High, \{x\}, q_\epsilon \rangle\}, \{x\}, \{q_\epsilon\} \rangle$$

Given automata $\mathcal{A}^{\psi_1} = (Q_1, q_0^1, \delta_1, \{x\}, F_1)$, $\mathcal{A}^{\psi_2} = (Q_2, q_0^2, \delta_2, \{x\}, F_2)$ and $\mathcal{A}^D = (Q_3, q_0^3, \delta_3, \{x\}, F_3)$, then we define:

$$\mathcal{A}^{\pi.\psi_1} = \mathcal{A}^{\neg\psi_1} = \mathcal{A}^{\psi_1}$$
$$\mathcal{A}^{D.\psi_1} = \langle Q_3 \cup Q_1 \cup \bigcup_{q_f \in F_3} \{q_f^{D.\psi_1}\}, q_0^3, \delta_3 \cup \delta_1[\bigcup_{q_f \in F_3} q_f^{D.\psi_1}/q_0^1], \{x\}, F_1\rangle$$
$$\mathcal{A}^{\psi_1 \vee \psi_2} = \mathcal{A}^{\psi_1 \wedge \psi_2} = (Q_1 \cup Q_2 \cup \{q_0\}, q_0, \delta_1[q_0/q_0^1] \cup \delta_2[q_0/q_0^2], \{x\}, F_1 \cup F_2\})$$

Automaton $\mathcal{A}^\epsilon$ accepts all timed words. Automata $\mathcal{A}^{\pi.\psi_1}$ and $\mathcal{A}^{\neg\psi_1}$ coincide with $\mathcal{A}^{\psi_1}$, since both $\pi.\psi_1$ and $\neg\psi_1$ argue on the same words that are argued on by ψ_1. Automaton $\mathcal{A}^{D.\psi_1}$ performs a run of $\mathcal{A}^D$ followed by a run of $\mathcal{A}^{\psi_1}$. Note that the final states of $\mathcal{A}^D$ are marked by $D.\psi_1$. Both $\mathcal{A}^{\psi_1 \vee \psi_2}$ and $\mathcal{A}^{\psi_1 \wedge \psi_2}$ choose between performing runs of $\mathcal{A}^{\psi_1}$ and performing runs of $\mathcal{A}^{\psi_2}$.

Theorem 2. *Let ψ be a formula. It holds that $\mathcal{A} \models \psi$ iff $\mathcal{A} \otimes \mathcal{A}^\psi \models \psi$.*

6.2 Visit of $\mathcal{A} \otimes \mathcal{A}^\psi$

We say that a region is marked by an interval I or by a formula ψ iff one of the states of the region is.

Given an expression D, a formula ψ, an automaton $\mathcal{A}$ and a set of regions R in $RG(\mathcal{A})$, let $Reach(D, \psi, R, \mathcal{A})$ denote the set of regions in $RG(\mathcal{A})$ that are marked by $D.\psi$ and are reachable from R. The set $Reach(D, \psi, R, \mathcal{A})$ can be computed immediately by visiting $RG(\mathcal{A})$.

Let us define now the algorithm *CheckPi*. It takes a set of regions R of an automaton $\mathcal{A}$, a low actions l, an interval I in I_ψ and a formula π. It returns the set of regions R' such that, given the set of runs $\mathcal{R}$ departing from R, $[s] \in R'$ if and only if there is a run r in $\mathcal{R}$ such that $(r, \mathcal{R}) \models \pi$ and such that r reaches $[s]$ in a time $t \in I$ by reading a sequence of high actions followed by l.

CheckPi (R: set of regions, $\mathcal{A}$: automaton, l: *Low*, I: interval, π: formula): set of regions

1. $to\_be\_visited := \{([s], \emptyset) \mid [s] \in R\}$; $visited{:=}\emptyset$; $A{:=}\emptyset$;
2. **while** $to\_be\_visited \neq \emptyset$ **do**
3. $([s], H)$:=extract($to\_be\_visited$); Add($([s], H), visited$);
4. **for each** transition $\langle [s], l, [s']\rangle$ in $RG(\mathcal{A})$ with $l \in Low$ **do**
5. if $[s']$ is marked by I then $Add(([s'], H), A)$;
6. **for each** transition $\langle [s], h, [s']\rangle$ in $RG(\mathcal{A})$ with $h \in High$ **do**
7. if $([s'], H \cup \{h\}) \notin visited$ then Add($([s'], H \cup \{h\}), to\_be\_visited$);
8. **for each** $([s], H) \in A$ **do**
9. **case** π of
10. $true$ **then** $V(([s], H), \pi) := true$;
11. h **then** $V(([s], H), \pi) := (h \in H)$;
12. $(\pi_1)^\forall$ **then** $V((s, H), \pi) := \bigwedge_{([s'], H') \in A} V(([s'], H'), \pi_1)$;
13. $\neg\pi_1$ **then** $V(([s], H), \pi) :=$ NOT$(V(([s], H), \pi_1))$;
14. $\pi_1 \vee \pi_2$ **then** $V(([s], H), \pi) := (V(([s], H), \pi_1)$ OR $V(([s], H), \pi_2))$;
15. $\pi_1 \wedge \pi_2$ **then** $V(([s], H), \pi) := (V(([s], H), \pi_1)$ AND $V(([s], H), \pi_2))$;
16. **return** $\{[s] \mid ([s], H) \in A$ and $V(([s], H), \pi)\}$.

The algorithm uses pairs of the form $([s], H)$, with $[s]$ a region and H a set of high actions, meaning that $[s]$ is reachable from some region in R by reading precisely the high actions in H. In fact, the algorithm starts with considering the pairs $([s], \emptyset)$ such that $[s] \in R$ (see line 1), and, for each pair $([s], H)$ such that $\langle [s], h, [s'] \rangle$ is in $RG(\mathcal{A})$, the pair $([s'], H \cup \{h\})$ is generated (see lines 6–7).

The set A contains all pairs $([s], H)$ such that the configurations in $[s]$ can be reached by reading a sequence of high actions in H followed by the low action l, in a total time lying in I. In fact, at lines 4–5, for each pair $([s], H)$, if $\langle [s], l, [s'] \rangle$ is in $RG(\mathcal{A})$ and $[s']$ is marked by I, $([s'], H)$ is added to A.

Now, for each pair $([s], H)$ in A, the boolean $V(([s], H), \pi)$ is true iff the configurations in $[s]$ are reached by runs satisfying π. In fact, if $\pi \equiv true$ then $V(([s], H), \pi)$ takes true. If $\pi \equiv h$ then $V(([s], H), \pi)$ takes true iff $h \in H$, i.e. iff h is read to reach configurations in $[s]$. If $\pi \equiv (\pi_1)^{\forall}$ then $V(([s], H), \pi)$ takes true iff $V(([s'], H'), \pi_1)$ takes true for each pair $([s'], H') \in A$, i.e. iff all runs reaching regions $[s']$ with $([s'], H') \in A$ satisfy π_1. Note that we do not distinguish time values in interval I. The reason is immediate if I has the form $[c, c]$. If I has the form $(c, c+1)$ the choice is justified by Lemma 1. (Since $I \in I_\psi$, $I = [c, c]$ or $I = (c, c+1)$.) The cases $\pi \equiv \neg\pi_1$, $\pi \equiv \pi_1 \vee \pi_2$ and $\pi \equiv \pi_1 \wedge \pi_2$ are immediate.

The algorithm returns all regions $[s]$ s.t. $([s], H) \in A$ and $V(([s], H), \pi)$.

The algorithm $CheckPsi(\psi, R, \mathcal{A})$ defined below checks whether runs originating from the regions R of the automaton $\mathcal{A}$ satisfy the formula ψ.

CheckPsi(ψ: formula, R: set of regions, $\mathcal{A}$: automaton): *boolean*

1. **case** ψ of
2. $\quad\epsilon$ **then return** $R \neq \emptyset$;
3. $\quad D.\psi'$ **then return** $CheckPsi(\psi', Reach(D, \psi', R, \mathcal{A}), \mathcal{A})$;
4. $\quad\pi.\psi'$ **then** $R' := \bigcup_{I \in Int_\psi} \bigcup_{l \in Low} ChechPi(R, \mathcal{A}, l, I, \pi)$;
5. $\qquad\qquad$ **return** $CheckPsi(\psi', R', \mathcal{A})$;
6. $\quad\neg\psi_1$ **then return** NOT $(CheckPsi(\psi_1, R, \mathcal{A}))$;
7. $\quad\psi_1 \vee \psi_2$ **then return** $(CheckPsi(\psi_1, R, \mathcal{A})$ OR $CheckPsi(\psi_2, R, \mathcal{A}))$;
8. $\quad\psi_1 \wedge \psi_2$ **then return** $(CheckPsi(\psi_1, R, \mathcal{A})$ AND $CheckPsi(\psi_2, R, \mathcal{A}))$.

If $\psi \equiv \epsilon$ then the algorithm returns *true* iff $R \neq \emptyset$, since all states of automata have departing transitions.

If $\psi \equiv D.\psi'$, we consider the regions $R' = Reach(D, \psi', R, \mathcal{A})$ reachable from R by reading an observable sequence in $\mathcal{D}[D]$, and from which runs that must satisfy ψ' must be performed (this last requirement coincide with asking that the regions are marked by $D.\psi'$). Then, we apply $CheckPsi$ to $\psi', R', \mathcal{A}$.

If $\psi \equiv \pi.\psi'$, we consider the regions R' that are reached by runs satisfying π. These runs are grouped by intervals in Int_ψ in which the regions are reached, and by low actions that are read to reach them. Then, we apply $CheckPsi$ to $\psi', R', \mathcal{A}$. So, given any run $\stackrel{\omega'}{\rightarrow} [s'] \stackrel{\omega''}{\rightarrow}$ from R with $[s'] \in R'$, we forget the portion of the run that performs ω', notwithstanding we should check that the complete run satisfy ψ'. Our choice is correct since ψ is a normal form and, therefore, the forgotten part of the run does not play any rôle in the satisfiability of ψ'.

The cases $\psi \equiv \neg\psi'$, $\psi \equiv \psi_1 \wedge \psi_2$ and $\psi \equiv \psi_1 \vee \psi_2$ are immediate.

Now, to check whether an automaton $\mathcal{A}$ satisfies a formula ψ, it suffices to apply $CheckPsi$ to ψ, the initial region of $\mathcal{A} \otimes \mathcal{A}^{\psi}$, and $\mathcal{A} \otimes \mathcal{A}^{\psi}$. (Note that the termination of the algorithm is immediate.)

Theorem 3. *Let $\mathcal{A}$ be an automaton and ψ be a formula. It holds that:*

$$CheckPsi(\psi, \{(q_0, [v_0])\}, \mathcal{A} \otimes \mathcal{A}^{\psi}) \text{ returns } True \text{ iff } \mathcal{A} \models \psi.$$

7 Discussion and Conclusions

In this section, before drawing some conclusions, we discuss three choices: 1) the choice of introducing TIFL instead of exploiting existing timed temporal logics, like *TCTL* and *TPTL* [5], to express information flow properties; 2) the choice of solving the problem $\mathcal{A} \models \psi$ without exploiting the well known techniques of *reachability* and *emptiness*; 3) the choice of using dense time domains.

There are three good reasons for introducing TIFL. The first is that specifying information flow properties with TIFL is easier, as one expects since TIFL is an ad hoc logic for information flow.

The second reason is that some properties that can be expressed by TIFL formulae can be expressed by TCTL and TPTL formulae that are at least exponential in size w.r.t. the size of TIFL formulae, and that, therefore, are intractable. An example is given by TIFL formula $(l_1, I_1).(h_1 \wedge \ldots \wedge h_n).(l_2, I_2).\epsilon$, which requires that there is a run where $h_1, \ldots, h_n$ appear, in any order, between l_1 and l_2. To express this formula in TPTL and TCTL, we must enumerate each possible sequence of $h_1, \ldots, h_n$, thus having a formula exponential w.r.t. n.

The third reason is that the decidable classes of TCTL and TPTL do not admit punctuality (see [11] and [6]), which is useful in our setting. As an example, let $\mathcal{A}$ be the automaton in Fig.3. We note that $\mathcal{A} \models (l_1, I_1).(\neg h).(l_2, [1,1]).\neg\epsilon$, which means that if l_2 is performed exactly one time unit after l_1, then h has been performed in the meantime. This is a leak of information, which cannot be expressed without considering interval $[1,1]$, i.e. without considering punctuality.

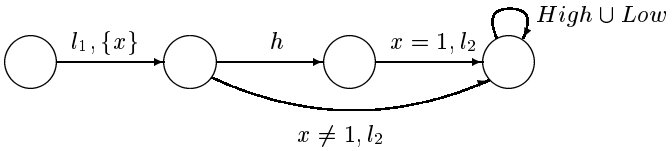

Fig. 3. The rôle of punctuality

Let discuss now reachability and emptiness. Reachability tackles the question whether a given configuration can be reached. In our case, this is not sufficient to know whether a formula is satisfied, for two reasons: 1) The problem $\mathcal{A} \models \psi$ does not depend only on the fact that a given configuration is reached or not, but, in

general, also the behavior of the system before reaching that configuration matters; 2) a word may be recognized by runs which cross different configurations.

The idea of emptiness is to express the negation of a formula ψ with an automaton $\mathcal{A}_{\neg\psi}$, so that $\mathcal{A} \models \psi$ iff $\mathcal{L}(\mathcal{A}) \cap \mathcal{L}(\mathcal{A}_{\neg\psi}) = \emptyset$. As an example, if $\psi = (l_1, I_1).(h).(l_2, I_2).\neg\epsilon$, which requires that h does not appear between l_1 and l_2, then $\mathcal{A}_{\neg\psi}$ is the automaton in Fig.4, whose runs can be combined with the the runs in $\mathcal{A}$ where h appears between l_1 and l_2. Let us take now $\psi = (h_1 \wedge \neg(\neg h_2)^{\forall}).\epsilon$. We have that $\mathcal{A} \models \psi$ iff

$$\{r \in \mathcal{R}(\mathcal{A}) \,|\, (r, \mathcal{R}(\mathcal{A})) \models h_1 \text{ and exists } r' \equiv_{os} r \text{ s.t. } (r', \mathcal{R}(\mathcal{A})) \models h_2\} \neq \emptyset.$$

In this case we cannot apply emptiness, since we are not able to construct $\mathcal{A}_{\neg\psi}$. The reason is that $\mathcal{A}_{\neg\psi}$ cannot accept a run r depending on the fact that h appears or not, as in the case above, since a run r satisfying ψ may read h or not. Here $\mathcal{A}_{\neg\psi}$ should select r depending on what a different run r' does, which is not possible.

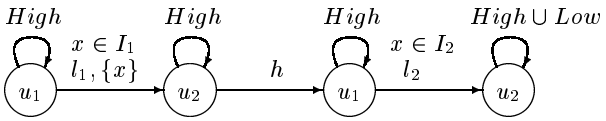

Fig. 4. The automaton $\mathcal{A}_{\neg((l_1,I_1).(h).(l_2,I_2).\neg\epsilon)}$

Finally, let us discuss the choice of dense time. A time domain T is discrete if there is a rational $k \in \mathbb{Q}$ such that $T = \{k \cdot c \,|\, c \in \mathbb{N}\}$. If one assumes discrete time domains for security there is a problem due to determining the value of k. In fact, if we consider dense time domains we are sure that the system respects the reality, which obviously ranges on dense time. In the case of discrete time we must calculate the exact k which gives this certitude. As an example, if we assume dense time then the automaton in Fig.5 satisfies $(l_1, [0,0]).h_1 \wedge h_2.(l_2, [1,1]).\epsilon$, which requires that there is a run where both h_1 and h_2 appear between l_1 and l_2. If we assume discrete time, then the property is satisfied only if we choice a k less than $\frac{1}{3}$, since $(l_1, [0,0]).h_1 \wedge h_2.(l_2, [1,1]).\epsilon$ can be satisfied by a run having three actions in three different instants and in a total time 1. This example shows that the k that must be chosen depends not only on the automaton, but also on the formula. Moreover, the number of the regions, that depends only on the size of the automaton in the case of dense time, depends and also on $\frac{1}{k}$ in the case of discrete time. This means that in the discrete case the number of regions increases with the size of the formula one wants to prove.

Summarizing, we have studied the problem of information flow in real-time systems described by Timed Automata. For this purpose, we have introduced TIFL to express information flow properties as properties of the language accepted by an automaton. We have given an algorithm to check these properties.

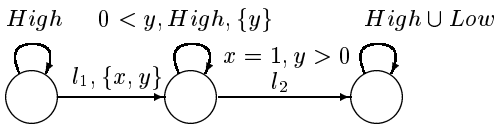

Fig. 5. The importance of k with discrete time

We have motivated the choice of introducing TIFL instead of considering existing time temporal logic, the choice of do not consider reachability and emptiness, and the choice of using dense time.

References

1. M. Abadi, On SDSI's linked local name spaces, J. Computer Security **6**, 1998.
2. M. Abadi, M. Burrows, and R. Needham, A logic of authentication, ACM Trans. Comput. Syst. **8**, 1990.
3. M. Abadi and M.R. Tuttle, A semantics for a logic of authentication, Proc. ACM Symposium on Principles of Distributed Computing, 1991.
4. R. Alur and D.L. Dill, A theory of timed automata, Theoret. Comput. Sci. **126**, 1994.
5. R. Alur and T. Henzinger, Logics and models of real time: A survey, Proc. Real Time: Theory in Practice, Springer LNCS 600, 1992.
6. R. Alur and T. Henzinger, The benefits of relaxing punctuality, J. ACM **43**, 1996.
7. E.W. Felten and M.A. Schneider, Timing attacks on web privacy, Proc. ACM Conference on Computer and Communications Security, 2000.
8. R. Focardi and R. Gorrieri, Automatic compositional verification of some security properties, Proc. TACAS, Springer LNCS 1055, 1996.
9. R. Focardi and R. Gorrieri, Classification of security properties for process algebras, J. Computer Security **3**, 1995.
10. R. Focardi, R. Gorrieri, and F. Martinelli, Information flow analysis in a discrete-time process algebra, Proc. IEEE CSFW, 2000.
11. S. La Torre and M. Napoli, A decidable dense branching-time temporal logic, Proc. FSTTCS, Springer LNCS 1974, 2000.
12. N. Lynch, I/O automaton models and proofs for shared–key communication systems, Proc. IEEE CSFW, 1999.
13. I.S. Moskowitz and O.L. Costich, A classical automata approach to noninterference type problems, Proc. IEEE CSFW, 1992.
14. G. Smith and D. Volpano, Secure information flow in a multi-threaded imperative language, Proc. ACM POPL, 1998.
15. D. Volpano and G. Smith, Confinement properties for programming languages, SIGACT News **29**, 1998.

Appendix

Proof of Prop. 2.
Let us assume that a configuration (q, v) is reached by some run r of $\mathcal{A}$ consisting of n steps. We prove that also any configuration (q, v') satisfying the conditions of Def. 2 is reached by some run of $\mathcal{A}$ consisting of n steps.

The proof is by induction over n.

The case $n = 0$ is immediate, because both (q, v) and (q, v') must be the initial configuration of $\mathcal{A}$.

Let us consider the case $n > 0$.

Let (q_1, v_1) be the configuration that is reached by the first $n-1$ steps of r, and let $(q_1, v_1) \rightarrow_t^a (q, v)$ be the last step of r. Moreover, let $e = \langle q_1, \theta, a, Y, q \rangle$ be the transition fired to perform such a step.

Let us suppose first that $Y = \emptyset$.

¿From the definition of step it holds that $v_1 = v - t$.

Since the time domain T is dense, there exists a time value t' such that $\lfloor t' \rfloor = \lfloor t \rfloor$, $\lceil t' \rceil = \lceil t \rceil$, and, for each clock x, it holds that if $fract(v(x)) \# fract(t)$ then $fract(v'(x)) \# fract(t')$, for any $\# \in \{<, =, >\}$.

Now, first of all we prove that $(q_1, v' - t')$ and $(q_1, v_1) = (q_1, v - t)$ satisfy the hypothesis of Def. 2.

Let us begin with proving that $(q_1, v' - t')$ and (q_1, v_1) satisfy the first condition of Def. 2. We show that for each clock x, $\lfloor (v-t)(x) \rfloor = \lfloor (v'-t')(x) \rfloor$ (the case $\lceil (v-t)(x) \rceil = \lceil (v'-t')(x) \rceil$ is similar). We consider the three cases $fract(t) > fract(v(x))$, $fract(t) = fract(v(x))$ and $fract(t) < fract(v(x))$.

If $fract(t) > fract(v(x))$ then $\lfloor (v-t)(x) \rfloor = \lfloor (v)(x) \rfloor - \lceil t \rceil$. Since t' and v' are such that $\lfloor (v)(x) \rfloor = \lfloor (v)'(x) \rfloor$ and $\lceil t \rceil = \lceil t' \rceil$, it holds that $\lfloor (v-t)(x) \rfloor = \lfloor (v'-t')(x) \rfloor$.

If either $fract(t) = fract(v(x))$ or $fract(t) < fract(v(x))$ then $\lfloor (v-t)(x) \rfloor = \lfloor (v)(x) \rfloor - \lfloor t \rfloor$. Since v' and t' are such that $\lfloor (v)(x) \rfloor = \lfloor (v)'(x) \rfloor$ and $\lfloor t \rfloor = \lfloor t' \rfloor$, it holds that $\lfloor (v-t)(x) \rfloor = \lfloor (v'-t')(x) \rfloor$.

Now, let us show that $(q_1, v' - t')$ and $(q_1, v_1) = (q_1, v - t)$ satisfy also the second condition of Def. 2. We prove only that for each pair of clocks x and y, it holds that $fract((v-t)(x)) < fract((v-t)(y))$ if and only if $fract((v'-t')(x)) < fract((v'-t')(y))$, since the cases $\{=, >\}$ are similar.

We consider the case $fract(t) < fract(v(x)) < fract(v(y))$ and the case $fract(v(x)) < fract(t) < fract(v(y))$. We omit case $fract(v(x)) < fract(v(y)) < fract(t)$, which is similar to the case $fract(t) < fract(v(x)) < fract(v(y))$.

By definitions of v' and t' it holds that $fract(t) < fract(v(x)) < fract(v(y))$ if and only if $fract(t') < fract(v'(x)) < fract(v'(y))$.

Let us observe that for arbitrary $\hat{t}$ and $\hat{v}$, if $fract(\hat{t}) < fract(\hat{v}(x))$, then it holds that $fract((\hat{v} - \hat{t})(x)) = fract((\hat{v})(x)) - fract(\hat{t})$.

So, it holds that $fract((v-t)(x)) < fract((v-t)(y))$ and $fract((v'-t')(x)) < fract((v'-t')(y))$.

Now, let us consider the case $fract(v(x)) < fract(t) < fract(v(y))$. By definitions of v' and t' it holds that $fract(v(x)) < fract(t) < fract(v(y))$ if and only if $fract(v'(x)) < fract(t') < fract(v'(y))$.

Then, it holds that $fract((v-t)(x)) > fract((v-t)(y))$ and $fract((v'-t')(x)) > fract((v'-t')(y))$.

Since $(q_1, v'-t')$ and (q_1, v_1) satisfy the conditions of Def. 2, by inductive hypothesis we infer that $(q_1, v'-t')$ is reachable through a run in $\mathcal{R}(\mathcal{A})$ consisting of $n-1$ steps. To complete the proof it suffices to prove that $(q_1, v'-t') \rightarrow_a^{t'} (q, v')$ is a step. This fact holds since conditions of Def. 2 imply that v and v' are in the same clock region and, therefore, v and v' satisfy the same clock constraints, thus implying that $(q_1, v'-t') \rightarrow_a^{t'} (q, v')$ is a step performing transition $e = \langle q_1, \theta, a, Y, q\rangle$.

Now, we have to consider the case $Y \neq \emptyset$. There exists a valuation v'' such that $v = v''[Y]$. Since (q, v) and (q, v') satisfy the conditions of Def. 2, it holds that $v'(x) = 0$ for each clock $x \in Y$. Therefore, there is a clock valuation v''' such that $v' = v'''[Y]$. Let $(q_1, v_1) = (q_1, v''-t)$ be the configuration that is reached by the first $n-1$ steps of r, and let $(q_1, v''-t) \rightarrow_a^t (q, v)$ be the last step of r. Moreover, let $e = \langle q_1, \theta, a, Y, q\rangle$ be the transition fired to perform such a step. With the arguments of the proof of case $Y = \emptyset$, we can show that there is a time value t' such that configuration $(q_1, v'''-t')$ and $(q_1, v''-t)$ satisfy the conditions of Def. 2. So, by inductive hypothesis $(q_1, v'''-t')$ is reachable by $n-1$ steps. Moreover, v''' and v'' are in the same clock region and, therefore, they satisfy the same constraints, thus implying that $(q_1, v'''-t') \rightarrow_a^{t'} (q, v'''[Y]) = (q, v')$ is a step performing transition e. □

Proof of Lemma 1.
The proof is by contradiction. We study the case $\pi \equiv h$. The others are similar. Without loss of generality, we consider transitions whose clock constraints are conjunctions of basic expressions $x\#c$.

Let $r \in \mathcal{R}$ be a run with $(\omega^r)_{Low} = (t, l)\ldots$. Let us suppose that there exists a run $r' \in \mathcal{R}$ with $(\omega^{r'})_{Low} = (t', l)\ldots$ and t, t' in $(c, c+1)$ such that $(r, \mathcal{R}) \models h^\forall$ and $(r', \mathcal{R}) \not\models h^\forall$. In this case there exists a run $r'' = s_0 \rightarrow_{t_1}^{h_1} s_1 \rightarrow_{t_2}^{h_2} \ldots s_{n-1} \rightarrow_{t_n}^{h_n} s_n \rightarrow_{t_{n+1}}^{l} s_{n+1}$ in $\mathcal{R}$ such that $h \notin \{h_1, \ldots, h_n\}$ and such that $\sum_{i=1}^{n+1} t_i = t'$.

Let e_i be the transition $e_i = \langle q_{i-1}, \theta, a_i, Y_i, q_i\rangle$ that is used by r'' to perform the i^{th} step, for $1 \leq i \leq n+1$ (note that $a_1 = h_1, \ldots, a_n = h_n, a_{n+1} = l$), and let (q_i, v_i) be the configuration that is reached by r'' after the i^{th} step, for $0 \leq i \leq n+1$ (i.e. $s_i = (q_i, v_i)$ for $0 \leq i \leq n+1$).

The clock valuation v_0 can be obtained as one solution of a set $\mathcal{S}$ of solutions of a system of linear inequalities S that is constructed as follows:

- for each clock x such that $fract(v_0(x)) > 0$, we take the inequality $\lfloor v(x)\rfloor < x < \lceil v(x)\rceil$,
- for each clock x such that $fract(v_0(x)) = 0$, we take the inequality $x = v(x)$,

- for each pair of clocks x and y such that $fract(v_0(x)) < fract(v_0(y))$, we take the inequalities $y - x > \lfloor v(y) \rfloor - \lfloor v(x) \rfloor$ and $y - x < \lceil v(y) \rceil - \lfloor v(x) \rfloor$.
- for each pair of clocks x and y such that $fract(v_0(x)) = fract(v_0(y))$, we take the inequality $x - y = \lfloor v(x) \rfloor - \lfloor v(y) \rfloor$.

Note that the representation given above is legal since $\mathcal{R}$ is dense and, therefore, each solution v in $\mathcal{S}$ is such that (s_0, v) is a reachable configuration.

We can determine times $t_1, \ldots, t_n, t_{n+1}$ of the run r'' as the solutions of a set of linear inequalities that is obtained from S by adding the following inequalities:

1. the inequality $t_1 + \ldots + t_i + x \# c$ such that $x \# c$ is a clock constraint of transition e_i, and the clock x is never reset from the first step to the $(i-1)^{th}$ step (i.e. $x \notin Y_1 \cup \ldots \cup Y_{i-1}$)
2. the inequality $t_j + \ldots + t_i \# c$ such that $x \# c$ is a clock constraint of transition e_i, and the clock x is reset in the $(j-1)^{th}$ step, namely $x \in Y_{j-1}$ and $x \notin Y_j \cup \ldots \cup Y_{i-1}$
3. the inequalities $t_1 > 0, \ldots, t_{n+1} > 0$.

The set of solutions of the system defined above is a convex. The vertexes have natural coordinates (this follows from the form of the inequalities and the fact that only integer constants appear in constraints of transitions). Moreover, the function $\sum_{i=1}^{n+1} t_i$ has a natural constant c_1 as inf, because it is in one of the the vertexes. Finally, either $\sum_{i=1}^{j} t_i$ has as sup a natural constant c_2, which is in one of the vertexes, or $sup = \infty$.

Therefore, the runs that perform the same sequence of transitions of r'' can be performed arbitrarily in the interval I such that $inf(I) = c_1$ and $sup(I)$ is either c_2 or ∞, with $c_1, c_2 \in \mathbb{N}$.

We can conclude that there exists a run r''' that takes the same transitions of r'' in time t. Therefore $(r, \mathcal{R}) \not\models h^{\forall}$, which contradicts the hypothesis. □

Proof of Prop. 4
We can transform ψ into a formula satisfying the first condition of Def. 3 by exploiting the following equivalences:

- $\pi_1.\pi_2.\psi \equiv (\pi_1 \wedge \pi_2).\psi$
- $\pi.(\psi_1 \vee \psi_2) \equiv (\pi.\psi_1) \vee (\pi.\psi_2)$
- $\pi.(\psi_1 \wedge \psi_2) \equiv (\pi.\psi_1) \wedge (\pi.\psi_2)$
- $\pi.\neg(\psi) \equiv \neg(\pi.\psi)$

Now, we can transform the formula obtained into a formula satisfying also the second condition of Def. 3, since for each expression (I, L) there are intervals $I_1, \ldots, I_n$ such that $(I, L) = (I_1, L) + \ldots + (I_n, L)$ and $I_1, \ldots I_n \in Int_{\psi}$. □

Proof of Theorem 1
By induction over n.

Base case: If $n = 0$ then the thesis follows directly by Prop. 2, since $\mathcal{R}_n = \mathcal{R}(\mathcal{A})$.

Induction step: Assume that the thesis holds for n and let us prove the thesis for $n+1$. There are two cases: $op_{n+1} = op_\pi$ and $op_{n+1} = op_D$.

Let us consider first the case $op_{n+1} = op_\pi$. We prove that by induction on the structure of π. We can assume, without loss of generality, that the scope of operator $\neg$ is either h or $\pi^\forall$.

- $\pi \equiv h$
  Let (q, v) be any configuration that is reachable by a run $r \in \mathcal{R}_{n+1}$. Let us consider the run $r' \in \mathcal{R}(\mathcal{A})$ and the natural m such that r is obtained from r' by pruning the first m steps. (Prunings are due to the operators $op_1, \ldots, op_n$ of the form op_D.) Let us consider an automaton $\mathcal{A}'$ with states $s_0, s_1, \ldots, s_{m+1}$, with a transition from s_i to s_{i+1} labeled $\langle true, Low \cup High, \emptyset \rangle$, for each $0 \leq i \leq m+1$, with a transition from s_m to s_m labeled $\langle true, Low \cup High \setminus \{h\}, \emptyset \rangle$, with a transition from s_m to s_{m+1} labeled $\langle true, h, \emptyset \rangle$ and with a transition from s_{m+1} to s_{m+1} labeled $\langle true, Low \cup High, \emptyset \rangle$. Let $\mathcal{A} \otimes \mathcal{A}'$ denote the cartesian product of $\mathcal{A}$ and $\mathcal{A}'$. In correspondence with the run r', there is a run r'' in $\mathcal{R}(\mathcal{A} \otimes \mathcal{A}')$ reaching either the configuration $(\{q, s_m\}, v)$ or $(\{q, s_{m+1}\}, v)$. Now, by the inductive hypothesis, $op_n(\ldots(op_1(\mathcal{R}_{\mathcal{A}\otimes\mathcal{A}'})))$ is dense. So, there is a run $\hat{r''}$ in this last set that reaches either the configuration $(\{q, s_m\}, v')$ or $(\{q, s_{m+1}\}, v')$ for each configuration $(\{q, s_m\}, v')$ and $(\{q, s_{m+1}\}, v')$ such that $(\{q, s_m\}, v')$ and $(\{q, s_m\}, v)$ satisfy the conditions of Def. 2 and $(\{q, s_{m+1}\}, v')$ and $(\{q, s_{m+1}\}, v)$ satisfy the conditions of Def. 2. So, we can project $\hat{r''}$ to $\mathcal{A}$, thus obtaining a run $\hat{r} \in \mathcal{R}_n$ reaching configuration (q, v'), with (q, v) and (q, v') satisfying conditions of Def. 2. Moreover, $\hat{r}$ reads h iff r does. So, given (q, v) reachable by the run $r \in \mathcal{R}_{n+1}$, for any (q, v') with (q, v) and (q', v') satisfying conditions of Def. 2, we have been able to prove that (q, v') is reachable by a run in $\mathcal{R}_{n+1}$, thus proving that $\mathcal{R}_{n+1}$ is dense.
- $\pi \equiv \neg h$
  Similar to the previous case.
- $\pi \equiv \pi^\forall$
  We note that $\mathcal{R}_{n+1} = \{r \in \mathcal{R}_n \,|\, (r, \mathcal{R}_n) \models \pi\} = \bigcup_{I \in I_\psi} \mathcal{R}^I_{n+1}$, with $\mathcal{R}^I_{n+1} = \{r \in \mathcal{R}_n \,|\, (r, \mathcal{R}_n) \models \pi, (\omega^r)_{Low} = (t, l) \ldots \text{ and } t \in I\}$. So, it suffices to prove that $\mathcal{R}^I_{n+1}$ is dense for each $I \in I_\psi$. By Lemma 1 and the fact that $\mathcal{R}_n$ is (by inductive hypothesis) dense, it follows that either $\mathcal{R}^I_{n+1} = \emptyset$, or $\mathcal{R}^I_{n+1} = \{r \in \mathcal{R}_n \,|\, (\omega^r)_{Low} = (t, l) \ldots \text{ and } t \in I\}$. In the former case $\mathcal{R}^I_{n+1}$ is immediately dense. In the latter case, let (q, v) be any configuration that is reachable by a run $r \in \mathcal{R}^I_{n+1}$ in time $t \in I$. Let us consider the run $r' \in \mathcal{R}(\mathcal{A})$ and the natural m such that r is obtained from r' by pruning the first m steps. (Prunings are due to the operators $op_1, \ldots, op_n$ of the form op_D.) Let us consider an automaton $\mathcal{A}'$ with states $s_0, s_1, \ldots, s_m$, with a transition from s_i to s_{i+1} labeled $\langle true, Low \cup High, \{x_{\mathcal{A}'}\} \rangle$, for each $0 \leq i \leq m+1$, and with a transition from s_m to s_m labeled $\langle true, Low \cup High, \emptyset \rangle$. In correspondence with the run r', there is a run r'' in $\mathcal{R}(\mathcal{A} \otimes \mathcal{A}')$ reach-

ing the configuration $(\{q, s_m\}, v \cup \{(x_{\mathcal{A}'}, t)\})$. Now, by the inductive hypothesis, $op_n(\ldots(op_1(\mathcal{R}_{\mathcal{A}\otimes\mathcal{A}'})))$ is dense. So, there is a run $\hat{r''}$ in this last set that reaches the configuration $(\{q, s_m\}, v' \cup \{(x_{\mathcal{A}'}, t')\})$ for each configuration $(\{q, s_m\}, v' \cup \{(x_{\mathcal{A}'}, t')\})$ such that $(\{q, s_m\}, v' \cup \{(x_{\mathcal{A}'}, t')\})$ and $(\{q, s_m\}, v \cup \{(x_{\mathcal{A}'}, t)\})$ satisfy the conditions of Def. 2. In particular, it holds that also $t' \in I$. So, we can project $\hat{r''}$ to $\mathcal{A}$, thus obtaining a run $\hat{r} \in \mathcal{R}^I_{n+1}$ reaching configuration (q, v') in a time $t' \in I$, with (q, v) and (q', v') satisfying conditions of Def. 2. This implies the thesis.

- $\pi \equiv \neg(\pi'^{\forall})$:
  similar to the previous case.
- $\pi \equiv \pi_1 \vee \pi_2$ and $\pi \equiv \pi_1 \wedge \pi_2$
  By structural induction and the semantics of π.

Let us consider now the case $op_{n+1} = op_D$. Let (q, v) be any configuration that is reachable by a run $r \in \mathcal{R}_{n+1}$. Let us consider an automaton $\mathcal{A}'$ with states s_0, s_1, with a transition from s_0 to s_1 labeled $\langle true, Low \cup High, \{x_1\}\rangle$, and with a transition from s_1 to s_0 labeled $\langle true, Low \cup High, \{x_0\}\rangle$. Automata $\mathcal{A}$ and $\mathcal{A}\otimes\mathcal{A}'$ are clearly equivalent. By inductive hypothesis, $op_n(\ldots(op_1(\mathcal{R}(\mathcal{A}\otimes\mathcal{A}'))))$ is dense. Let $((q, s_i), v \cup \{(x_i, 0), (x_{i-1 \bmod 2}, t)\})$ be any configuration in $\mathcal{A}\otimes\mathcal{A}'$ corresponding to the configuration (q, v) in $\mathcal{A}$ that is reachable by a run r_1 in $op_{n+1}(\ldots(op_1(\mathcal{R}(\mathcal{A}\otimes\mathcal{A}'))))$. There is a run r'_1 in $op_n(\ldots(op_1(\mathcal{R}(\mathcal{A}\otimes\mathcal{A}'))))$ such that r_1 is obtained from r'_1 by pruning some steps reading a finite observable timed sequence d_1 in $\mathcal{D}[D]$. Since $op_n(\ldots(op_1(\mathcal{R}(\mathcal{A}\otimes\mathcal{A}'))))$ is dense, we are sure that there is a run r'_2 in $op_n(\ldots(op_1(\mathcal{R}(\mathcal{A}\otimes\mathcal{A}'))))$ reaching any configuration $((q, s_i), v' \cup \{(x_i, 0), (x_{i-1 \bmod 2}, t')\})$ with $((q, s_i), v' \cup \{(x_i, 0), (x_{i-1 \bmod 2}, t')\})$ and $((q, s_i), v \cup \{(x_i, 0), (x_{i-1 \bmod 2}, t)\})$ satisfying the conditions of Def. 2. Now, by the proof of Prop. 2, it follows that r'_1 and r'_2 read the same sequence of actions. Let r_2 denote the run obtained from r'_2 by pruning the steps reading the finite observable timed sequence d_2 having the same sequence of actions of d_1. We note that clocks x_0 and x_1 ensure that also $d_2 \in \mathcal{D}[D]$. This implies that $r_2 \in op_{n+1}(\ldots(op_1(\mathcal{R}(\mathcal{A}\otimes\mathcal{A}'))))$. It follows that there is a run in $op_{n+1}(\ldots(op_1(\mathcal{R}(\mathcal{A}))))$ reaching (q, v'). (This run is the projection of r_2 on $\mathcal{A}$.) This implies the thesis. □

Proof of Theorem 2

By the definition of relation $\models$ and the fact that $\mathcal{A}\otimes\mathcal{A}^{\psi}$ recognizes precisely the timed words that are recognized by $\mathcal{A}$ and whose observable part is argued on by ψ. □

Proof of Theorem 3

Since the set of runs originating from the region $(q_0, [v_0])$ coincide with $\mathcal{R}(\mathcal{A})$ and are dense, it suffices to prove that if $\mathcal{R}$ is a dense set of runs starting from

the regions in a set $[\mathcal{R}]$, then it holds that:

$$CheckPsi(\psi, [\mathcal{R}], \mathcal{A} \otimes \mathcal{A}^{\psi}) \text{ returns } True \text{ iff } \mathcal{R} \models \psi.$$

The proof is by induction over ψ.

- $\psi \equiv \epsilon$
  Since $\mathcal{R}$ is empty if and only if the set of starting regions $[\mathcal{R}]$ is empty, the thesis holds.
- $\psi \equiv D.\psi'$
  Let $\mathcal{R}_D$ be the set of runs $\{r \mid \xrightarrow{\omega} r \in \mathcal{R}, \omega_{Low} \in \mathcal{D}[D]$ and ω terminates with a low action$\}$. The semantics of $D.\psi'$ implies that $\mathcal{R} \models D.\psi'$ if and only if $\mathcal{R}_D \models \psi'$.
  By the definition of $CheckPsi$, it holds that $CheckPsi(D.\psi', [\mathcal{R}], \mathcal{A} \otimes \mathcal{A}^{\psi})$ returns $True$ if and only if $CheckPsi(\psi', Reach(D, \psi', [\mathcal{R}], \mathcal{A} \otimes \mathcal{A}^{\psi}), \mathcal{A} \otimes \mathcal{A}^{\psi})$ returns $True$.
  So, we have to prove that $\mathcal{R}_D \models \psi'$ iff $CheckPsi(\psi', Reach(D, \psi', [\mathcal{R}], \mathcal{A} \otimes \mathcal{A}^{\psi}), \mathcal{A} \otimes \mathcal{A}^{\psi})$ returns $True$.
  Now, by the definition of $Reach$, the set of starting regions of $\mathcal{R}_D$ is equal to the set $Reach(D, \psi', [\mathcal{R}], \mathcal{A} \otimes \mathcal{A}^{\psi})$. Moreover, by Theorem 1, the set $\mathcal{R}_D$ is dense. So, $\mathcal{R}_D \models \psi'$ iff $CheckPsi(\psi', Reach(D, \psi', [\mathcal{R}], \mathcal{A} \otimes \mathcal{A}^{\psi}), \mathcal{A} \otimes \mathcal{A}^{\psi})$ returns $True$ holds by the inductive hypothesis.
- $\psi \equiv \pi.\psi'$
  Let $\mathcal{R}_\pi$ be the set of urns $\{r \in \mathcal{R} \mid (r, \mathcal{R}) \models \pi\}$. The semantics of $\pi.\psi'$ implies that $\mathcal{R} \models \pi.\psi'$ iff $\mathcal{R}_\pi \models \psi'$.
  Let $\mathcal{R}_\pi$ be the set $\bigcup_{I \in Int_\psi} \bigcup_{l \in Low} ChechPi([\mathcal{R}], \mathcal{A} \otimes \mathcal{A}^{\psi}, l, I, \pi)$. By the definition of $CheckPsi$, it holds that $CheckPsi(\pi.\psi', [\mathcal{R}], \mathcal{A} \otimes \mathcal{A}^{\psi})$ returns $True$ iff $CheckPsi(\psi', \mathcal{R}_\pi, \mathcal{A} \otimes \mathcal{A}^{\psi})$ returns $True$.
  So, we have to prove that $\mathcal{R}_\pi \models \psi'$ iff $CheckPsi(\psi', \mathcal{R}_\pi, \mathcal{A} \otimes \mathcal{A}^{\psi})$ returns $True$.
  We note that by the definition of algorithm $CheckPi$, the set of regions $\mathcal{R}_\pi$ is equal to the set of starting regions of the set of runs $\bigcup_{I \in I_\psi} \bigcup_{l \in Low} \{r \mid \xrightarrow{\omega} r \in \mathcal{R}_\pi, (\omega)_{Low} = (l, t)$ and $t \in I\}$. Let us denote this last set with $\mathcal{R}'_\pi$. (So, $[\mathcal{R}'_\pi] = \mathcal{R}_\pi$.)
  So, we have to prove that $\mathcal{R}_\pi \models \psi'$ iff $CheckPsi(\psi', [\mathcal{R}'_\pi], \mathcal{A} \otimes \mathcal{A}^{\psi})$ returns $True$. We reason by induction over ψ'. Note that, by Prop. 4, we can assume that $\pi.\psi'$ is in normal form.
  - $\psi' \equiv \epsilon$
    Since $\mathcal{R}_\pi$ is empty if and only if $\mathcal{R}'_\pi$ is empty if and only if the set of starting regions $[\mathcal{R}'_\pi]$ is empty, the thesis holds.
  - $\psi' \equiv D.\psi''$
    Let $\mathcal{R}_{\pi.D}$ be the set $\{r \mid \xrightarrow{\omega} r \in \mathcal{R}_\pi, \omega_{Low} \in \mathcal{D}[D]$ and ω terminates with a low action$\}$. The semantic of $D.\psi''$, implies that $\mathcal{R}_\pi \models D.\psi''$ if and only if $\mathcal{R}_{\pi.D} \models \psi''$.
    By the definition of $CheckPsi$, it holds that $CheckPsi(D.\psi'', [\mathcal{R}'_\pi], \mathcal{A} \otimes \mathcal{A}^{\psi})$ returns $True$ if and only if $CheckPsi(\psi'', Reach(D, \psi'', [\mathcal{R}'_\pi], \mathcal{A} \otimes$

$\mathcal{A}^\psi), \mathcal{A} \otimes \mathcal{A}^\psi)$ returns $True$. By definition of $Reach$, the set of starting regions of $\mathcal{R}_{\pi.D}$ is equal to the set $Reach(D, \psi'', [\mathcal{R}'_\pi], \mathcal{A} \otimes \mathcal{A}^\psi)$. In fact, $Reach$ uses marks to reach the regions by means of D, but the marks do not appear before the starting states of $\mathcal{R}'_\pi$.
Moreover, by Theorem 1, since $\mathcal{R}$ is dense, both $\mathcal{R}_\pi$ and $\mathcal{R}_{\pi.D}$ are dense. Then, the thesis holds by the inductive hypothesis.

- $\psi' \equiv \neg\psi_1$
  The thesis follows immediately by the semantics of $\neg$ and by the inductive hypothesis.
- $\psi' \equiv \psi_1 \vee \psi_2$
  The thesis follows immediately by the semantics of $\vee$ and by the inductive hypothesis.
- $\psi' \equiv \psi_1 \wedge \psi_2$
  The thesis follows immediately by the semantics of $\wedge$ and by the inductive hypothesis.

– $\psi \equiv \neg\psi_1$
The thesis follows immediately by the semantics of $\neg$, by the inductive hypothesis, and by the definition of $CheckPsi(\neg\psi_1)$.

– $\psi \equiv \psi_1 \vee \psi_2$
The thesis follows immediately by the semantics of $\vee$, by the inductive hypothesis, and by the definition of $CheckPsi(\psi_1 \wedge \psi_2)$.

– $\psi \equiv \psi_1 \wedge \psi_2$
The thesis follows immediately by the semantics of $\wedge$, by the inductive hypothesis, and by the definition of $CheckPsi(\psi_1 \vee \psi_2)$.

□