

Timed Abstract Non-interference

Roberto Giacobazzi¹ and Isabella Mastroeni²

¹ Dipartimento di Informatica - Università di Verona - Verona, Italy
roberto.giacobazzi@univr.it

² Department of Computing and Information Sciences - Kansas State University - Manhattan,
Kansas, USA
isabellm@cis.ksu.edu

Abstract. In this paper, we introduce a timed notion of abstract non-interference. This is obtained by considering semantics which observe time elapsed in computations. Timing channels can be modeled in this way either by letting the attacker to observe time as a public variable or reckon the time elapsed by observing the computational traces' length, corresponding to observe the program counter. In the first case abstract non-interference provides a model for abstracting the information about time, namely we can for example consider models of attackers that can observe only intervals of time, or other more abstract properties. In the second case abstract non-interference provides a model for attackers able to observe properties of trace length, e.g., the public memory during the whole computation. We investigate when adding the observation of time does not increase the attacker's power in disclosing confidential information about data. This models the absence of timing channels in language-based security.

Keywords: Abstract interpretation, security, non-interference, timing channels.

1 Introduction

The standard approach to the confidentiality problem, also called *non-interference*, is based on a characterization of attackers that does not impose any observational or complexity restriction on the attackers' power [19]. The notion of abstract non-interference (ANI) has been introduced recently in [10] as a weakening of this notion, modeling weaker attackers having a restricted observation power specified as an abstraction of the concrete domain of computation. In this paper we prove that ANI is an adequate model of non-interference, including a variety of timing channels as a special case.

The problem. Any non-interference problem in language-based security has first to specify what an attacker can observe. Standard denotational semantics is typically not adequate for modeling covert channels such as timing channels, deadlock channels and termination channels. Consider for instance the following simple program fragment:

$$P \stackrel{\text{def}}{=} \mathbf{while} \ h \ \mathbf{do} \ l := l; h := h - 1 \ \mathbf{endw}$$

In this case, the public output is unchanged, independently from the initial value of h . However, if the attacker can measure the time elapsed, then it could understand whether the while-loop is executed or not, disclosing some information about the initial value of h . This means that, in this case, the program is not secure. Standard non-interference does not model this behavior. Also, ANI as specified in [10] does not model this aspect of information leakage, being based on a (time insensitive) denotational semantics.

Main contribution. In this paper we introduce a notion of abstract non-interference which captures timing channels, called *timed abstract non-interference* (TANI). This notion provides the appropriate setting for studying how properties of private data interfere during the execution of the program, with properties of the elapsed time. We will not consider in this paper the converse problem, i.e., how properties of the time elapsed during program execution interfere with properties of data. This because we do not consider here real-time system, where time is relevant to the behavior of programs [3]. We show that timing channels can be modeled as instances of ANI, simply by considering an appropriate, typically more concrete, semantics observing time. The simplest way to include time in the semantics consists in considering the maximal trace semantics of the transition system associated to programs: The trace semantics implicitly embeds the *time* elapsed during computations in the *length* of traces. Another, more explicit, way of including the information about time in the semantics consists in enriching the operational semantics of our language in order to measure the elapsed time: We consider a semantics that *stores*, in a specific variable, the time elapsed during the execution of a program. In this case we consider the time information as a public datum that can be observed by the attacker. Since, the difference between abstract non-interference in [10] and timed abstract non-interference lies only upon the semantics used, it is always possible to characterize the most concrete harmless attacker, observing timing-channels. This gives a measure of the level of security of a program under attack when the time elapsed may represent a critical information. Moreover, time, as well as data, can be abstracted. This is essential in order to model attackers that can observe properties of time elapsed, such as intervals or regular delays such as congruences. It is worth noting that, since we do not consider real-time systems, the only possible timing channels are due to a one-way interaction between data (modeled either as trace length or in the semantics) and time. We model this interaction in the framework of ANI by providing sufficient conditions that avoid timing channels in programs. Indeed, the fact that, in the most general case, an attacker may observe relational dependencies between data and time, may create problems when we want to abstract time. Hence, we provide the conditions characterizing the attackers whose capability of observing time does not increase their power in disclosing confidential information about data. In other words, we characterize in which conditions timed abstract non-interference, where attackers observe time, implies (untimed) abstract non-interference, where the same attackers cannot observe time.

2 Information flows in Language-based Security

Confidential data are considered *private*, labeled with H (high-level of secrecy), while all other data are public, labeled with L (low-level of secrecy) [9]. Non-interference can be naturally expressed by using semantic models of program execution. This idea goes back to Cohen’s work on *strong dependency* [7], which uses denotational semantics for modeling how information can be transmitted among variables during the execution of programs. Therefore non-interference for programs essentially means that “*a variation of confidential (high or private) input does not cause a variation of public (low) output*” [19]. When this happens, we say that the program has only *secure information flows* [5, 7, 9, 16, 22]. This situation has been modeled by considering the denotational (input/output) semantics $\llbracket P \rrbracket$ of the program P . In particular we consider programs where data are typed as private (H) or public (L). Program states in Σ are functions (represented as tuples) mapping variables in the set of values \mathbb{V} . Finite traces on Σ are denoted Σ^+ . If $T \in \{H, L\}$, $n = |\{x \in \text{Var}(P) \mid x : T\}|$, and $v \in \mathbb{V}^n$, we abuse notation by denoting $v \in \mathbb{V}^T$ the fact that v is a possible value for the variables with security type T . Moreover, we assume that any input s , can be seen as a pair (h, l) , where $s^H = h$ is a value for private data and $s^L = l$ is a value for public data. In this case, *non-interference* can be formulated as follows.

$$\boxed{\text{A program } P \text{ is secure if } \forall \text{ input } s, t \ s^L = t^L \Rightarrow (\llbracket P \rrbracket(s))^L = (\llbracket P \rrbracket(t))^L}$$

This problem has been formulated also as a *Partial Equivalence Relation* (PER) [20, 15]. In this case we have that if the input data are equivalent under a given equivalent relation, then also the outputs have to be equivalent. The standard methods for checking non-interference are based on security-type systems and data-flow/control-flow analysis. Type-based approaches are designed in such a way that well-typed programs do not leak secrets. In a security-typed language, a type is inductively associated with program statements in such a way that any statement showing a potential flow disclosing secrets is rejected [21, 24]. Similarly, data-flow/control-flow analysis techniques are devoted to statically discover flows of secret data into public variables [6, 16, 17, 20]. All these approaches are characterized by the way they model attackers (or unauthorized users). As far as timing channels are concerned, they are avoided in the Volpano and Smith type system [23] by adding some restrictions or they can be removed by using the program transformation in [1]. Concerning timed automata, a decidable notion of non-interference has been introduced in [4]. We showed in [13] that this notion can be modeled as a generalization of abstract non-interference.

Abstract Non-Interference. The idea of ANI [10], is that an attacker can observe only some properties, modeled as abstract interpretations of program semantics, of public concrete values. The *model of an attacker*, also called *attacker*, is therefore a pair of abstractions $\langle \eta, \rho \rangle$, with $\eta, \rho \in \text{uco}(\wp(\mathbb{V}^L))^3$, representing what an observer can see

³ $\text{uco}(\wp(\mathbb{V}^L))$ denotes the set of all the upper closure operators on $\wp(\mathbb{V}^L)$.

$$\begin{array}{l}
[\eta]P(\rho) \text{ if } \forall h_1, h_2 \in \mathbb{V}^{\mathbb{H}}, \forall l_1, l_2 \in \mathbb{V}^{\mathbb{L}} . \eta(l_1) = \eta(l_2) \Rightarrow \rho(\llbracket P \rrbracket(h_1, l_1)^{\mathbb{L}}) = \rho(\llbracket P \rrbracket(h_2, l_2)^{\mathbb{L}}) \\
(\eta)P(\phi \rightsquigarrow \rho) \text{ if } \forall h_1, h_2 \in \mathbb{V}^{\mathbb{H}}, \forall l \in \mathbb{V}^{\mathbb{L}} . \rho(\llbracket P \rrbracket(\phi(h_1), \eta(l))^{\mathbb{L}}) = \rho(\llbracket P \rrbracket(\phi(h_2), \eta(l))^{\mathbb{L}})
\end{array}$$

Table 1. Narrow and Abstract Non-Interference.

about, respectively, the input and output of a program. The notion of *narrow (abstract) non-interference* (NNI) represents the first weakening of standard non-interference relatively to a given model of an attacker. When a program P satisfies NNI we write $[\eta]P(\rho)$, see Table 1. The problem with this notion is that it introduces *deceptive flows* [10]. Consider, for instance, $l := l * h^2$, and consider the public input property of being an even number, then we can observe a variation of the output's sign due to the existence of both negative and positive even numbers, revealing flows which does not depend on the private data, here called *deceptive*. In order to avoid deceptive interference we introduce a weaker notion of non-interference, having no deceptive flows, such that, when the attacker is able to observe the property η of public input, and the property ρ of public output, then no information flow concerning the private input is observable from the public output. The idea is to compare the set of all the computations that have the same input property η , instead of the single computations. We call this notion *abstract non-interference* (ANI). When a program P satisfies abstract non-interference we write $(\eta)P(\phi \rightsquigarrow \rho)$, where $\phi \in uco(\wp(\mathbb{V}^{\mathbb{H}}))$, denoting a confidential data property that we want to keep secret (see Table 1). Note that $[id]P(id)$ models exactly (standard) non-interference. Moreover, we have that abstract non-interference is a weakening of both, standard and narrow non-interference: $[id]P(id) \Rightarrow (\eta)P(\phi \rightsquigarrow \rho)$ and $[\eta]P(\rho) \Rightarrow (\eta)P(\phi \rightsquigarrow \rho)$, while standard non-interference is not stronger than narrow one due to deceptive interference. A proof-system has been introduced, in [11], for checking both narrow and abstract non-interference inductively on program's syntax, while in [25] the author derive a type system for enforcing abstract non-interference in a simple λ -calculus. Moreover, in [10], two methods for deriving the most concrete output observation for a program, given the input one, for both narrow and abstract non-interference are provided together with a domain transformer characterizing the most abstract property that should be declassified in order to guarantee abstract non-interference. In [12] we prove that these two construction form an adjunction in the standard framework of abstract interpretation.

Example 1. Consider the properties *Sign* and *Par*, observing, respectively, the sign and the parity of integers, and the program: $P \stackrel{\text{def}}{=} l := l * h^2$. with security typing: $h : \mathbb{H}$ and $l : \mathbb{L}$ and $\mathbb{V} = \mathbb{Z}$. Let us check if $(id)P(id \rightsquigarrow \text{Par})$. Note that $\text{Par}(\llbracket P \rrbracket(2, 1)^{\mathbb{L}}) = \text{Par}(4) = 2\mathbb{Z}$ while $\text{Par}(\llbracket P \rrbracket(3, 1)^{\mathbb{L}}) = \text{Par}(9) = 2\mathbb{Z} + 1$, which are clearly different, therefore in this case $(id)P(id \rightsquigarrow \text{Par})$ doesn't hold. Consider $(id)P(\text{Sign} \rightsquigarrow \text{Par})$. Note that $\text{Par}(\llbracket P \rrbracket(\text{Sign}(2), 1)^{\mathbb{L}}) = \text{Par}(\llbracket P \rrbracket(\text{Sign}(3), 1)^{\mathbb{L}}) = \text{Par}(0+) = \mathbb{Z}$. In this case it is simple to check that $(id)P(\text{Sign} \rightsquigarrow \text{Par})$ holds.

3 The timed semantics for deterministic languages

Consider a simple imperative language, IMP defined by the following syntax: $c ::= \text{nil} \mid x := e \mid c; c \mid \text{while } x \text{ do } c \text{ endw}$, with e denoting expressions evaluated in the set of values \mathbb{V} with standard operations, i.e., if $\mathbb{V} = \mathbb{N}$ then e can be any arithmetical expression. \mathbb{V} can be structured as a flat domain whose bottom element, \perp , denotes the value of undefined variables. We follow Cousot's construction [8], defining semantics, at different levels of abstractions, as the abstract interpretation of the maximal trace semantics of a transition system associated with each well-formed program. In the following, $\widehat{\Sigma}^+$ and $\widehat{\Sigma}^\omega \stackrel{\text{def}}{=} \mathbb{N} \longrightarrow \widehat{\Sigma}$ denote respectively the set of finite nonempty and infinite sequences of symbols in the set $\widehat{\Sigma}$. Given a sequence $\widehat{\sigma} \in \widehat{\Sigma}^\infty \stackrel{\text{def}}{=} \widehat{\Sigma}^+ \cup \widehat{\Sigma}^\omega$, its length is $|\widehat{\sigma}| \in \mathbb{N} \cup \{\omega\}$ and its i -th element is $\widehat{\sigma}_i$. A non-empty finite (infinite) *trace* $\widehat{\sigma} \in \widehat{\Sigma}^\infty$ is a finite (infinite) sequence of program states such that, for all $i < |\widehat{\sigma}|$ we have $\widehat{\sigma}_i \rightarrow \widehat{\sigma}_{i+1}$. The *maximal trace semantics* [8] of a transition system associated with a program P is $\llbracket P \rrbracket^\infty$, denoted $\langle P \rangle$, where if $T \subseteq \widehat{\Sigma}$ is a set of final/blocking states then $\langle P \rangle^{\dot{n}} = \{\widehat{\sigma} \in \widehat{\Sigma}^+ \mid |\widehat{\sigma}| = n, \forall i \in [1, n]. \widehat{\sigma}_{i-1} \rightarrow \widehat{\sigma}_i\}$, $\langle P \rangle^\omega = \{\widehat{\sigma} \in \widehat{\Sigma}^\omega \mid \forall i \in \mathbb{N}. \widehat{\sigma}_i \rightarrow \widehat{\sigma}_{i+1}\}$. We can define $\langle P \rangle^+ = \cup_{n>0} \{\widehat{\sigma} \in \langle P \rangle^{\dot{n}} \mid \widehat{\sigma}_{n-1} \in T\}$, and $\langle P \rangle^n = \langle P \rangle^{\dot{n}} \cap \langle P \rangle^+$. If $\widehat{\sigma} \in \langle P \rangle^+$, then $\widehat{\sigma}_+$ and $\widehat{\sigma}_-$ denote respectively the final and initial state of $\widehat{\sigma}$. The *denotational semantics* $\llbracket P \rrbracket$ associates input/output functions with programs, by modeling non-termination by \perp . This semantics is derived in [8] as an abstract interpretation of the maximal trace semantics: $\alpha^{\mathcal{D}}(X) \stackrel{\text{def}}{=} \lambda \widehat{s} \in \widehat{\Sigma}. \{\widehat{\sigma}_+ \mid \widehat{\sigma} \in X \cap \widehat{\Sigma}^+, \widehat{s} = \widehat{\sigma}_-\} \cup \{\perp \mid \widehat{\sigma} \in X \cap \widehat{\Sigma}^\omega, \widehat{s} = \widehat{\sigma}_-\}$. Note that, in our case, $\alpha^{\mathcal{D}}(X)(\widehat{s})$ is always a singleton. It is well known that we can associate, inductively on its syntax, with each program $P \in \text{IMP}$ a function $\llbracket P \rrbracket$ denoting its input/output relation, such that $\llbracket P \rrbracket \stackrel{\text{def}}{=} \alpha^{\mathcal{D}}(\langle P \rangle)$ [8]. In the following, if $|\text{Var}(P)| = n$, we consider $\Sigma \stackrel{\text{def}}{=} \mathbb{V}^n$ as the set of values for the variables, and $\langle P \rangle$ and $\llbracket P \rrbracket$ will denote the semantics with $\widehat{\Sigma} = \Sigma$.

Consider now a semantics containing the information about the elapsed time. In particular, consider the well-known operational semantics of IMP enhanced with time, described in Table 2, where $s \in \Sigma$, $t \in \mathbb{N}$ and $t_A, t_T \in \mathbb{N}^4$ are constant values denoting respectively the time spent for an assignment and for a test. In this case we suppose that the states in the concrete semantics are $\widehat{\Sigma} \stackrel{\text{def}}{=} \Sigma \times \mathbb{N}$. Namely we suppose that a state is a pair composed by a tuple of values for the variable and by a natural value representing the time elapsed from the beginning of the execution. This operational semantics naturally induces a transition relation on a set of states Σ , denoted \rightarrow , specifying the relation between a state and its possible successors.

This transition system allows us to define the *timed maximal trace semantics* $\langle P \rangle^{+T}$ modeling computations by using traces of states including the information about time. By using the abstraction $\alpha^{\mathcal{D}}$ on this maximal trace semantics, we can obtain a *timed denotational semantics*, denoted by $\llbracket P \rrbracket^{+T} = \alpha^{\mathcal{D}}(\langle P \rangle^{+T})$.

Moreover, we can note that also the (standard) trace semantics, i.e., with $\widehat{\Sigma} = \Sigma$, can be

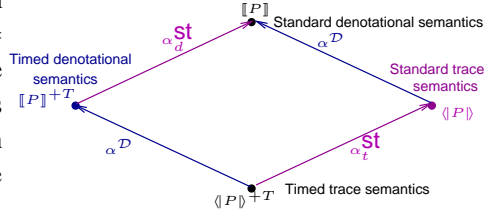
⁴ It would be the same if $t \in \mathbb{R}$

$\langle \mathbf{nil}, \langle s, t \rangle \rangle \rightarrow \langle s, t \rangle \quad \frac{\langle e, \langle s, t \rangle \rangle \rightarrow n \in \mathbb{V}_x}{\langle x := e, \langle s, t \rangle \rangle \rightarrow \langle s[n/x], t + t_A \rangle}$
$\frac{\langle c_0, \langle s, t \rangle \rangle \rightarrow \langle s_0, t_0 \rangle, \langle c_1, \langle s_0, t_0 \rangle \rangle \rightarrow \langle s_1, t_1 \rangle}{\langle c_0; c_1, \langle s, t \rangle \rangle \rightarrow \langle s_1, t_1 \rangle} \quad \frac{\langle x, \langle s, t \rangle \rangle \rightarrow 0}{\langle \mathbf{while } x \mathbf{ do } c \mathbf{ endw}, \langle s, t \rangle \rangle \rightarrow \langle s, t + t_T \rangle}$
$\frac{\langle x, \langle s, t \rangle \rangle \rightarrow n \geq 1, \langle c, \langle s, t \rangle \rangle \rightarrow \langle s_0, t_0 \rangle, \langle \mathbf{while } x \mathbf{ do } c \mathbf{ endw}, \langle s_0, t_0 \rangle \rangle \rightarrow \langle s_1, t_1 \rangle}{\langle \mathbf{while } x \mathbf{ do } c \mathbf{ endw}, \langle s, t \rangle \rangle \rightarrow \langle s_1, t_1 + t_T \rangle}$

Table 2. Operational timed semantics of IMP

seen as an abstraction of the timed maximal trace semantics, i.e., with $\widehat{\Sigma} = \Sigma \times \mathbb{N}$. In particular, let us consider $\alpha^{\text{st}}(\langle s, t \rangle) = s$ and $\alpha_t^{\text{st}}(X) = \bigcup_{\sigma \in X} \langle \alpha^{\text{st}}(\sigma_{\perp}), \dots, \alpha^{\text{st}}(\sigma_{\perp}) \rangle$, where X is a set of traces on $\widehat{\Sigma}$. Then maximal trace semantics is $\llbracket P \rrbracket = \alpha_t^{\text{st}}(\llbracket P \rrbracket^{+T})$.

Moreover, if we define the abstraction α_d^{st} , on the timed denotational semantics, as $\alpha_d^{\text{st}}(f) = \lambda s. \alpha^{\text{st}} \circ f(\langle s, 0 \rangle)$, when $f : \widehat{\Sigma} \rightarrow \wp(\widehat{\Sigma})$, we obtain the (standard) denotational semantics $\llbracket P \rrbracket = \alpha_d^{\text{st}}(\llbracket P \rrbracket^{+T})$. All the semantics, with their abstraction relations, are shown in the Figure on the left.



4 Defining Timed abstract non-interference

One of the most important features of abstract non-interference is that it is parametric on the chosen semantics. This means that we can enrich/change the checked notion of abstract non-interference for imperative languages by simply enriching/changing the considered semantics. For this reason, the idea for making abstract non-interference time sensitive, namely able to detect timing channels, corresponds to considering a more concrete semantics observing time. The first approach consists in considering the maximal trace semantics, instead of the denotational one, since the trace semantics compare the partial results at each step of computation. Indeed, the trace semantics implicitly embeds the *time* elapsed during computations in the *length* of traces since we can suppose that the time is measured as the discrete number of computational steps executed by the system. This observation suggests us that the trace semantics can be used for defining a stronger notion of non-interference that capture also timing channels. Therefore, we assume that we can have a timing channel in presence of an attacker that can count the number of execution steps, which means that the attacker observes time by looking at the program counter. On the other hand, we can also embed the reckon of time in the semantics, by considering time as a public variable observed by any attacker. In this way we can also think of modeling attackers that can only observe properties of time, e.g., intervals or regular delays as congruences.

4.1 Timed Abstract Non-Interference on traces

First of all, consider abstract non-interference, defined in [10] in terms of trace semantics. This means that at *each step* of computation we require that, what an attacker may observe does not depend on private input. This situation is possible whenever we suppose that the attacker is able to observe the public memory modified by the program. Since abstract non-interference is based on the distinction between input and output, the simplest way to extend the two notions is to consider, as output, the results of all the partial computations, and as input only the initial values (namely the initial state). With this assumption we can consider again only two closures, η for the public input, and ρ for the public output⁵.

Consider first narrow abstract non-interference. We can formulate the notion of non-interference by saying that starting from a state with the low property η , then all the possible observations of the states during the computation have the same low property ρ . Therefore, the new notion of narrow non-interference consists simply in abstracting each state of any computational trace. Let us introduce this notion through an example. Consider the standard semantics where states are simply tuples of values for the variables, and consider for example the concrete trace (each state is $\langle h, l \rangle$, with $h : \mathbb{H}$ and $l : \mathbb{L}$): $\langle 3, 1 \rangle \rightarrow \langle 2, 2 \rangle \rightarrow \langle 1, 3 \rangle \rightarrow \langle 0, 4 \rangle \rightarrow \langle 0, 4 \rangle$. Now, suppose to observe the parity of public data, i.e., *Par*, both in input and in output, then intuitively the abstraction, i.e., observation of this trace through the property *Par*, is: $2\mathbb{Z} + 1 \rightarrow 2\mathbb{Z} \rightarrow 2\mathbb{Z} + 1 \rightarrow 2\mathbb{Z} \rightarrow 2\mathbb{Z}$. Formally, given $\sigma \in \langle P \rangle$, we define its abstraction through the observation of ρ , as follows: σ^ρ is such that $\forall i \leq |\sigma| . \sigma_i^\rho = \rho(\sigma_i^L)$. At this point, consider only the terminating trace semantics of P , then we can define the abstract semantics: Let $\rho \in uco(\mathbb{V}^L)$, $X \in \wp(\Sigma^+)$:

$$\langle P \rangle_\rho^\eta = \alpha_\rho^\eta(\langle P \rangle), \quad \alpha_\rho^\eta(X) = \{ s^\eta \delta^\rho \mid s \in \Sigma, s\delta \in X \}$$

This is clearly an abstraction since it is additive by construction. Hence, we can define narrow non-interference for traces as: Let $\eta, \rho \in uco(\wp(\mathbb{V}^L))$ and P a program

$$P \text{ is secure if } \forall h_1, h_2 \in \mathbb{V}^H, \forall l_1, l_2 \in \mathbb{V}^L . \langle P \rangle_\rho^\eta(h_1, l_1) = \langle P \rangle_\rho^\eta(h_2, l_2)$$

In the definition above of narrow non-interference on traces, we compare abstract *observations* of concrete *computations*. This means that in order to define narrow non-interference we keep the concrete semantics and we change its observation. If, instead, we want to define abstract non-interference on traces, then we have to change also the concrete semantics by considering as initial state the set of all the states with the same public input property, namely we consider a symbolic execution of the system. In this case we have to consider a lift of the transition relation to sets: Consider the transition system $\langle \Sigma, \rightarrow \rangle$, we define the lift $\rightarrow \subseteq \wp(\Sigma) \times \wp(\Sigma)$ as follows: $\forall X \in \wp(\Sigma)$

$$X \rightarrow \{ y \in \Sigma \mid \exists x \in X. x \rightarrow y \}$$

⁵ Note that in the most general case we could consider a family of “output” observations.

Consider now the lifted transition system $\langle \wp(\Sigma), \rightarrow \rangle$, and consider the trace semantics obtained from this transition system. Let us denote also this semantics as $\langle P \rangle$, since it is clear from the input (depending on the fact that it is a state or a set of states) which semantics we have to consider. Let us consider the abstract trace semantics $\langle P \rangle_\rho^\eta$, on the lifted transition system, then we define abstract non-interference on traces: Let $\phi \in \text{uco}(\wp(\mathbb{V}^H))$, $\eta, \rho \in \text{uco}(\wp(\mathbb{V}^L))$ and P be a program.

$$P \text{ is secure if } \forall h_1, h_2 \in \mathbb{V}^H, \forall l \in \mathbb{V}^L . \langle P \rangle_\rho^\eta(\phi(h_1), \eta(l))^L = \langle P \rangle_\rho^\eta(\phi(h_2), \eta(l))^L$$

As observed above, the maximal trace semantics contains some discrete information about time, namely it distinguishes, for instance, traces that differ only for the repetition of states. For this reason we can say that it models also the timing channels due to the capability of the attacker of observing the clock of the program. In order to check this kind of non-interference, in the framework proposed in [10], we consider further abstractions of the semantics. The interesting aspect of this extension is that we can apply the transformers defined on abstract non-interference simply by considering the approximation based on *bounded iteration*. Bounded iteration, in fact, proves I/O non-interference by requiring a stronger condition, i.e., it requires that all the partial computations provide the same public output (see [10]).

4.2 Timed abstract non-interference on Timed Semantics

In this section we consider states which contain the information of time, i.e, we explicitly treat time in abstract non-interference. This means that we could use languages where time can interfere in the flow of computation. Suppose that the low inputs are pairs where the first component is a tuple of possible values for low variables, and the second one is the time passed, i.e., $\widehat{l} = \langle l, t \rangle$. We denote by $\widehat{l}^D = l \in \mathbb{V}^L$ the projection on the data component, and $\widehat{l}^T = t \in \mathbb{N}$ the projection on the time component. In the following a state $\widehat{\sigma}$ will be the triple $\langle s^H, s^L, t \rangle$, and the initial states are of the kind $\widehat{\sigma} = \langle s_i, 0 \rangle$. Standard non-interference without timing channels is: Let P be a program

$$P \text{ is secure if } \forall l \in \mathbb{V}^L, t \in \mathbb{N}, \forall h_1, h_2 \in \mathbb{V}^H . (\llbracket P \rrbracket^{+T}(\langle h_1, l, 0 \rangle))^L T = (\llbracket P \rrbracket^{+T}(\langle h_2, l, 0 \rangle))^L T$$

We can define the notions of narrow and abstract timed non-interference. In the following, when a program satisfies timed narrow or abstract non-interference we write respectively $[\eta]P^{+T}(\rho)$ and $(\eta)P^{+T}(\phi \rightsquigarrow \rho)$, and consider $\eta, \rho \in \text{uco}(\wp(\mathbb{V}^L \times \mathbb{N}))$ and $\phi \in \text{uco}(\wp(\mathbb{V}^H))$.

Definition 1. – $P \in \text{IMP}$ is such that $[\eta]P^{+T}(\rho)$ if $\forall h_1, h_2 \in \mathbb{V}^H, \forall \widehat{l}_1, \widehat{l}_2 \in \mathbb{V}^L \times \{0\}$ such that $\eta(\widehat{l}_1)^L = \eta(\widehat{l}_2)^L \Rightarrow \rho(\llbracket P \rrbracket^{+T}(\langle h_1, \widehat{l}_1 \rangle)^L T) = \rho(\llbracket P \rrbracket^{+T}(\langle h_2, \widehat{l}_2 \rangle)^L T)$.
– $P \in \text{IMP}$ satisfies $(\eta)P^{+T}(\phi \rightsquigarrow \rho)$ if $\forall h_1, h_2 \in \mathbb{V}^H, \forall \widehat{l} \in \mathbb{V}^L \times \{0\}$ we have $\rho(\llbracket P \rrbracket^{+T}(\langle \phi(h_1), \eta(\widehat{l})^L \rangle)^L T) = \rho(\llbracket P \rrbracket^{+T}(\langle \phi(h_2), \eta(\widehat{l})^L \rangle)^L T)$.

It is clear that the only difference between these notions and the untimed ones is in the semantics, therefore we can inherit, in a straightforward way, the whole construction made in the previous sections, simply by considering the time as a further public datum. In particular this allows us to derive the most concrete property, about time, that an attacker has to observe in order to be harmless, as we can see in the following example.

Example 2. Let us consider the following example:

$$P \stackrel{\text{def}}{=} h := h \bmod 4; \mathbf{while} \ h \ \mathbf{do} \ l := 2l - l; h := h - 1; \mathbf{endw}$$

with security typing $t = \langle h : \mathbb{H}, l : \mathbb{L} \rangle$ and $\mathbb{V}^{\mathbb{L}} = \mathbb{N}$. Suppose that each state of the trace semantics is $\langle h, l, t \rangle$, and suppose $l \in \mathbb{V}^{\mathbb{L}}$ and $h \in \mathbb{V}^{\mathbb{H}}$, $h \neq 0$:

$$\begin{aligned} \langle 0, l, 0 \rangle &\rightarrow \langle 0, l, t_A \rangle \rightarrow \langle 0, l, t_A + t_T \rangle \\ \langle h, l, 0 \rangle &\rightarrow \langle h \bmod 4, l, t_A \rangle \rightarrow \langle (h \bmod 4) - 1, l, 3t_A + t_T \rangle \\ &\rightarrow \langle 0, l, 2(h \bmod 4)t_A + (h \bmod 4 + 1)t_T \rangle \end{aligned}$$

Therefore, if for example $h \bmod 4 = 2$ then the total time is $4t_A + 3t_T$. This means that the most concrete abstraction of the domain of time that avoids timing channels is the one that have the element $\{t_A + t_T, 2t_A + 2t_T, 4t_A + 3t_T, 6t_A + 4t_T\}$ and abstracts all the other natural numbers in themselves.

5 Timed vs Untimed Abstract Non-Interference...

In this section we compare abstract non-interference defined in terms of standard denotational semantics, with timed non-interference. In particular standard abstract non-interference can be seen as an abstraction of the time component of TANI.

5.1 ...on traces

We observed that the simple use of traces makes abstract non-interference time sensitive. We wonder how a notion of abstract non-interference on traces which is time-insensitive can be obtained as abstraction of TANI. Therefore, we want to derive a trace semantics which is not able to observe the clock, namely we have to make indistinguishable traces that differ only for the repetition of states. This is a well-known notion in literature, called *stuttering* [2]: A semantics is said to be without stuttering if it is insensitive to the repetition of states. Namely we can think of transforming the set of traces that gives semantics to the program by eliminating the stuttering and then we can check non-interference exactly as we have done before. Let X be a property on traces σ . Then X is without stuttering if $\sigma \in X . \sigma = \sigma_0 \sigma_1 \dots \sigma_n \dots$ then $\forall i \geq 0 . \sigma_0 \dots \sigma_i \sigma_i \dots \in X$. It is easy to show that the following abstraction, which characterizes the stuttering properties, is clearly an abstraction of sets of traces. $\langle P \rangle^{\text{stu}} = \alpha^{\text{stu}}(\langle P \rangle_\rho^\eta)$ where

$$\alpha^{\text{stu}}(X) = \{ \langle \sigma_0, \dots, \sigma_n \rangle \mid \exists \delta \in X . \delta = \langle \sigma_0^{k_0}, \dots, \sigma_n^{k_n} \rangle, \forall i . k_i \neq 0, \sigma_i \neq \sigma_{i+1} \}$$

Now we can formalize the abstract non-interference in the following way, obtaining a notion of abstract non-interference on traces unable to detect timing channels.

$$P \text{ is secure for trace-based abstract non-interference if } \forall h_1, h_2 \in \mathbb{V}^H, \forall l \in \mathbb{V}^L . \\ \alpha_\rho^\eta(\llbracket P \rrbracket^{\text{stu}}(\phi(h_1), \eta(l))^L) = \alpha_\rho^\eta(\llbracket P \rrbracket^{\text{stu}}(\phi(h_2), \eta(l))^L).$$

Note that, in this case, as it is also shown in the following example, abstract non-interference is always an abstraction of timed abstract non-interference. Namely the timed notion always stronger than the untimed one.

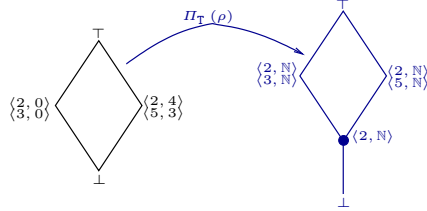
Example 3. Consider the trace semantics of a program P , with states $\langle h, l \rangle$, with $n \in \mathbb{N}$ and $m \in 2\mathbb{N}+1$: $\llbracket P \rrbracket = \{ \langle 0, 2 \rangle \rightarrow \langle 0, 3 \rangle \rightarrow \langle 0, 5 \rangle, \langle n, 2 \rangle \rightarrow \langle 0, 2 \rangle \rightarrow \langle 0, 3 \rangle, \langle n, m \rangle \rightarrow \langle n, m+1 \rangle \}$ Consider $\rho = \text{Par}$. We determine the abstract trace semantics relatively to Par : $\llbracket P \rrbracket_{\text{Par}}^{\text{Par}} = \alpha_{\text{Par}}^{\text{Par}}(\llbracket P \rrbracket) = \{ 2\mathbb{N} \rightarrow 2\mathbb{N}+1 \rightarrow 2\mathbb{N}+1, 2\mathbb{N} \rightarrow 2\mathbb{N} \rightarrow 2\mathbb{N}+1, 2\mathbb{N}+1 \rightarrow 2\mathbb{N} \}$ Hence, when the low input is even, we have interference. If we want to guarantee non-interference knowing that the attacker cannot observe the time elapsed, then we use the stuttering abstraction, obtaining the following semantics, which says that there's not interference, since for each abstract property we have only one possible result. $\llbracket P \rrbracket^{\text{stu}} = \{ \alpha^{\text{stu}}(\llbracket P \rrbracket_{\text{Par}}^{\text{Par}}) = 2\mathbb{N} \rightarrow 2\mathbb{N}+1, 2\mathbb{N}+1 \rightarrow 2\mathbb{N} \}$

5.2 ...on the Timed Semantics

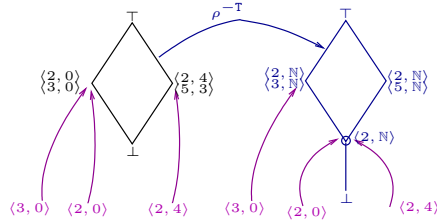
In this section, we study the relation existing between the timed notion of abstract non-interference and the untimed one, when we embed the time information in the semantics. Indeed, starting from closures $\eta, \rho \in \text{uco}(\wp(\mathbb{V}^L))$ without time, the extension to semantics with time is trivially obtained by taking the closure that is not able to observe anything about time, i.e., we interpret a generic closure $\eta \in \text{uco}(\wp(\mathbb{V}^L))$ as the closure that is not able to observe time, therefore that abstracts time to the top: $\eta^{+\text{T}} = \langle \eta, \lambda t. \mathbb{N} \rangle$. It is worth noting that $[\eta]P(\rho) \Leftrightarrow [\eta^{+\text{T}}]P^{+\text{T}}(\rho^{+\text{T}})$. In other words, since time can be treated as an additional public variable, we could see abstract non-interference as timed abstract non-interference where the time variable is abstracted to the top. Unfortunately, if we start from a semantics with time and we want to derive abstract non-interference properties without time, then the relation is not so immediate. Indeed, if time interferes with data we have that the abstraction of time is not straight. Clearly, time cannot interfere with data in the concrete semantics for the simple imperative language we are considering, but it can interfere in the abstract semantics modeling the attacker. In other words, when we consider timed abstract non-interference, the attacker model could observe *relations* between data and time, avoiding the independent abstraction of time. Namely if we abstract time we may lose something about the attacker's observation of data. Therefore, if we start from closure operators on semantics with time, namely from the closures $\eta, \rho \in \text{uco}(\wp(\mathbb{V}^L \times \mathbb{N}))$, we can obtain properties without time in two ways. We can think of erasing the observation of time only in the output, i.e., we abstract away

the information about time, or we can think of collecting all the possible results for every possible time value. In this way, we are able to ignore the information about time also in the input. In the following, we characterize some necessary and sufficient conditions on the attacker model, that indeed make timed abstract non-interference stronger than abstract non-interference, and therefore the two notions comparable.

Abstracting time in the output. Consider the first case, namely we do not observe time in the output. Let us define the projection, of a pair $X \in \wp(\mathbb{V}^L \times \mathbb{N})$, on data or on time in the following way: $\Pi_T(X) \stackrel{\text{def}}{=} \{ \langle x, y \rangle \mid \exists y' \in \mathbb{N}. \langle x, y' \rangle \in X, y \in \mathbb{N} \}$ and $\Pi_D(X) \stackrel{\text{def}}{=} \{ \langle x, y \rangle \mid \exists x' \in \mathbb{N}. \langle x', y \rangle \in X, x \in \mathbb{V}^L \}$. In particular, given a closure $\rho \in \text{uco}(\wp(\mathbb{V}^L \times \mathbb{N}))$, we can apply these abstractions to ρ obtaining for each $X \in \{T, D\}$, $\Pi_X(\rho) \stackrel{\text{def}}{=} \{ \Pi_X(Y) \mid Y \in \rho \}$. It is immediate to show that for each $X \in \{T, D\}$, $\Pi_X \in \text{uco}(\wp(\mathbb{V}^L \times \mathbb{N}))$. It is clear that the set $\Pi_T(\rho)$ is the set of the images of the map $\Pi_T \circ \rho$. Note that, even if both Π_T (or Π_D) and ρ are closures, then their composition may not be a closure, as happens in the picture below, where we have $\langle 2, \mathbb{N} \rangle \notin \Pi_T(\rho)$. In general $\Pi_T \circ \rho$ is not an upper closure operator, so we consider $\rho^{-T} \stackrel{\text{def}}{=} \mathcal{M}(\Pi_T(\rho))$ ⁶



and $\rho^{-D} \stackrel{\text{def}}{=} \mathcal{M}(\Pi_D(\rho))$, which are by definition of \mathcal{M} , closures, i.e., $\eta^{-T} \in \text{uco}(\wp(\mathbb{V}^L))$ and $\eta^{-D} \in \text{uco}(\wp(\mathbb{N}))$. The fact that in general $\rho^{-T} \neq \Pi_T(\rho)$ and $\rho^{-D} \neq \Pi_D(\rho)$ is a problem when we want to compare timed non-interference with abstract non-interference since elements that have the same image in ρ may be different in ρ^{-T} , and viceversa as we can see in the picture below. So, we would like to characterize when the two notions are



comparable. The picture above shows that problems arise when the Moore closure adds new points, namely when $\Pi_T(\rho)$ is not a closure. Therefore, we first want to understand when this is an upper closure operator, namely when $\Pi_T \circ \rho \in \text{uco}(\wp(\mathbb{V}^L \times \mathbb{N}))$. It is well known [18] that, given two closures $\rho, \pi \in \text{uco}(C)$, then we have $\pi \circ \rho \in \text{uco}(C)$ iff $\pi \circ \rho = \rho \circ \pi = \rho \sqcup \pi$. This means that we need a ρ such that $\Pi_T \circ \rho = \rho \circ \Pi_T$.

Theorem 1. *Let $\rho \in \text{uco}(\wp(\mathbb{V}^L \times \mathbb{N}))$, then $\rho^{-T} = \Pi_T \circ \rho \in \text{uco}(\wp(\mathbb{V}^L \times \mathbb{N}))$ iff $\Pi_T \circ \rho \circ \Pi_T = \rho \circ \Pi_T$ iff $\Pi_T \circ \rho \circ \Pi_T = \Pi_T \circ \rho$. Analogously, we have $\rho^{-D} = \Pi_D \circ \rho \in \text{uco}(\wp(\mathbb{V}^L \times \mathbb{N}))$ iff $\Pi_D \circ \rho \circ \Pi_D = \rho \circ \Pi_D$ iff $\Pi_D \circ \rho \circ \Pi_D = \Pi_D \circ \rho$.*

In [14] a method for transforming Π_T in order to make it satisfy $\Pi_T \circ \rho \circ \Pi_T = \rho \circ \Pi_T$ and $\Pi_T \circ \rho \circ \Pi_T = \Pi_T \circ \rho$ is provided. Anyway, in this context we are more interested in modifying ρ in order to guarantee completeness. In particular, let $X \in$

⁶ $\mathcal{M}(X) \stackrel{\text{def}}{=} \{ \wedge S \mid S \subseteq X \}$ is called Moore closure.

$\{T, D\}$, then the following transformations of ρ satisfies forward completeness. Note that $\Pi_x^+(X) \stackrel{\text{def}}{=} \bigcup \{ Y \mid \Pi_x(Y) \subseteq X \}$ is the right adjoint of Π_x :

$$\rho_x^\uparrow(Y) \stackrel{\text{def}}{=} \begin{cases} \Pi_x \circ \rho(Y) & \text{if } Y \in \Pi_x \\ \rho(Y) & \text{otherwise} \end{cases} \quad \rho_x^\downarrow(Y) \stackrel{\text{def}}{=} \begin{cases} \Pi_x^+ \circ \rho(Y) & \text{if } Y \in \Pi_x \\ \rho(Y) & \text{otherwise} \end{cases}$$

Then $\rho_x^\downarrow \sqsubseteq \rho \sqsubseteq \rho_x^\uparrow$. Namely we can always transform the abstractions, used for modeling the attacker, in order to guarantee that the abstraction of time is a complete upper closure operator, i.e., given a generic ρ we always have that $(\rho_x^\downarrow)^{-T}$ and $(\rho_x^\uparrow)^{-T}$ are closure operators. For the timed abstract non-interference this means that, given an attacker's model ρ , we can always find the closest model such that the observation of time does not enrich the attacker capability of observing data.

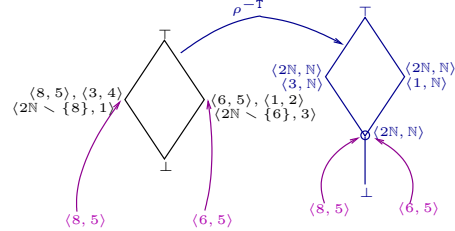
The following theorem says that the semantics for timed abstract non-interference is an abstract interpretation of the one for abstract non-interference with ρ^{-T} iff the output observation ρ commutes with Π_T

Theorem 2. *Let $\eta, \rho \in \text{uco}(\wp(\mathbb{V}^L \times \mathbb{N}))$, then $([\eta]P^{+T}(\rho) \Rightarrow [\eta^{-T}]P^{+T}(\rho^{-T}))$ if and only if we have $(\Pi_T \circ \rho = \rho \circ \Pi_T)$.*

Whenever we observe a relational property between data and time, in timed abstract non-interference we add new *deceptive flows*. Indeed, timed abstract non-interference may fail even if the program is secure and avoids timing channels, as it happens in the following example.

Example 4. Consider the program $P : h := 2; \text{ while } h \text{ do } l := l+2; h := h-1; \text{ endw}$.

In this example, we show that in the timed abstract non-interference we add new deceptive flows due to the possible relation between data and time of the abstract property. Consider a property ρ such that $\rho(\langle 8, 5 \rangle) \neq \rho(\langle 6, 5 \rangle)$ (as depicted in the picture) and $\eta = \langle \text{Par}, \text{id} \rangle$, which observes parity of data and the identity on time. Suppose $t_T = 1$ and $t_A = 0.5$.



Consider the initial low values (data and time) $\langle 4, 0 \rangle$ and $\langle 2, 0 \rangle$, clearly we have $\eta(\langle 4, 0 \rangle) = \langle 2\mathbb{N}, 0 \rangle = \eta(\langle 2, 0 \rangle)$. We have now compute the semantics:

$$\llbracket P \rrbracket(0, \langle 4, 0 \rangle)^{L^T} = \langle 8, 3t_T + 4t_T \rangle = \langle 8, 5 \rangle \text{ and } \llbracket P \rrbracket(0, \langle 2, 0 \rangle)^{L^T} = \langle 6, 5 \rangle$$

At this point, since $\rho(\langle 8, 5 \rangle) \neq \rho(\langle 6, 5 \rangle)$, non-interference is not satisfied, while there aren't timing information flows, namely $\rho^{-T}(\langle 8, 5 \rangle) = \rho^{-T}(\langle 6, 5 \rangle)$.

Note that, the presence of deceptive flows due to the observation of time arise only when there is a relational dependency between data and time.

The following corollary provides a characterization of timing channels in terms of the relation existing between the timed abstract non-interference and the abstract one.

Corollary 1. *If ρ commutes with Π_τ , i.e., $\Pi_\tau \circ \rho = \rho \circ \Pi_\tau$, and $[\eta^{-\tau}]P^{+\tau}(\rho^{-\tau})$ iff $[\eta]P^{+\tau}(\rho)$, then timing channels are impossible in P .*

Abstracting time in the input. As we said above we can think of another way of erasing time, this is also suggested by the fact that we have two possible compositions of a closure with the projection Π_τ : $\Pi_\tau \circ \rho$ and $\rho \circ \Pi_\tau$, which are the same when they are closures. Anyway, their meaning is different, the first compute the property with time and abstract the observation, while the second abstracts time in the input, namely compute the property on the abstracted value, without time. Let $L \subseteq \mathbb{V}^L$, we can define the closure on $\wp(\mathbb{V}^L)$ as $\rho_{-\tau}(L) \stackrel{\text{def}}{=} \downarrow_{\mathbb{D}} \circ \rho \circ \Pi_\tau(\langle L, \mathbb{N} \rangle)$, where $\langle L, \mathbb{N} \rangle \stackrel{\text{def}}{=} \{\langle x, y \rangle \mid x \in L, y \in \mathbb{N}\}$ and the abstraction $\downarrow_{\mathbb{D}}$ is the projection of the tuple on data, i.e., $\downarrow_{\mathbb{D}}(X) \stackrel{\text{def}}{=} \{x \mid \langle x, y \rangle \in X\}$.

In the following, we characterize when the composition $\downarrow_{\mathbb{D}} \circ \rho \circ \Pi_\tau$ is an upper closure operator. This is important in order to derive properties of narrow or abstract non-interference without time in programs where the semantics measures time, and therefore for understanding the relation existing between the notions of abstract non-interference with and without time.

Proposition 1. $\rho_{-\tau} \in \text{uco}(\wp(\mathbb{V}^L))$ iff $\downarrow_{\mathbb{D}} \circ \Pi_\tau \circ \rho \circ \Pi_\tau = \downarrow_{\mathbb{D}} \circ \rho \circ \Pi_\tau$.

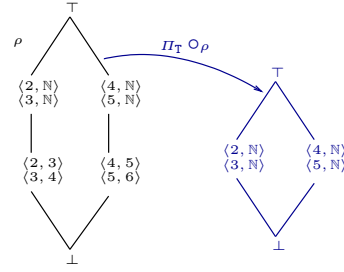
The following theorem says that the semantics for timed abstract non-interference is an abstract interpretation of the one for abstract non-interference with $\rho_{-\tau}$ iff the data projection commutes on elements closed under Π_τ .

Theorem 3. *Let $\eta, \rho \in \text{uco}(\wp(\mathbb{V}^L \times \mathbb{N}))$. Let $\eta_{-\tau}, \rho_{-\tau} \in \text{uco}(\wp(\mathbb{V}^L))$, $\downarrow_{\mathbb{D}} \Pi_\tau \circ \eta = \downarrow_{\mathbb{D}} \circ \Pi_\tau \circ \eta$, then $[\eta]P^{+\tau}(\rho) \Rightarrow [\eta_{-\tau}]P(\rho_{-\tau})$ if and only if $\downarrow_{\mathbb{D}} \circ \rho \circ \Pi_\tau = \downarrow_{\mathbb{D}} \circ \rho \circ \Pi_\tau \circ \rho$.*

Non relational attackers. A sufficient condition, in order to make the non-interference notions comparable, consists in considering only closures defined on $\wp(\mathbb{V}^L) \times \wp(\mathbb{N})$, which are particular closures of $\wp(\mathbb{V}^L \times \mathbb{N})$. Note that, if $\rho \in \text{uco}(\wp(A) \times \wp(B))$, then there always exist two closure ρ_A and ρ_B , such that $\rho(\langle X, Y \rangle) = \langle \rho_A(X), \rho_B(Y) \rangle$. Consider $\rho \in \text{uco}(\wp(\mathbb{V}^L) \times \wp(\mathbb{N}))$, we can obtain the closure $\rho^* \in \text{uco}(\wp(\mathbb{V}^L \times \mathbb{N}))$ as $\rho^* \stackrel{\text{def}}{=} \gamma \circ \rho \circ \alpha$, where $\alpha(X) \stackrel{\text{def}}{=} \langle X \downarrow_{\mathbb{D}}, X \downarrow_{\mathbb{T}} \rangle$ and $\gamma(\langle X, Y \rangle) \stackrel{\text{def}}{=} \{ \langle x, y \rangle \mid x \in X, y \in Y \}$ form a Galois insertion. Note that, in this case there are no more deceptive flows due to the observation of time, since here we cannot observe relations between data and time.

Proposition 2. *Let $\rho \in \text{uco}(\wp(\mathbb{V}^L) \times \wp(\mathbb{N}))$, consider $\rho^* \in \text{uco}(\wp(\mathbb{V}^L \times \mathbb{N}))$ defined $\rho^* \stackrel{\text{def}}{=} \gamma \circ \rho \circ \alpha$, then we have $\Pi_\tau \circ \rho^* = \rho^* \circ \Pi_\tau$ and $\Pi_{\mathbb{D}} \circ \rho^* = \rho^* \circ \Pi_{\mathbb{D}}$.*

Therefore, by Theorem 2 and Theorem 3, this means that in the conditions of the Proposition above, timed abstract non-interference implies abstract non-interference. This is only a sufficient condition, since there are closure operators $\rho \notin \text{uco}(\wp(\mathbb{V}^L) \times \wp(\mathbb{N}))$ such that $\Pi_T \circ \rho = \rho^{-T}$ or $\Pi_D \circ \rho = \rho^{-D}$. As we can see in the picture, $\langle n, \mathbb{N} \rangle \stackrel{\text{def}}{=} \{ \langle n, m \rangle \mid m \in \mathbb{N} \}$. In particular in the example above we can also note that $\Pi_T \circ \rho = \Pi_T \sqcup \rho$.



6 Conclusion

In this paper, we extend abstract non-interference in order to detect timing channels, namely those channels of informations created by the ability of the attacker to observe the time elapsed during computation, obtaining timed abstract non-interference, which embeds the time reckoning into the semantics. Afterwards, we study the relation between this new notion and abstract non-interference defined in [10]. This is an example of how, by changing the semantics, we can change the defined notion of non-interference. In the same way, we would like to define a *probabilistic* abstract non-interference, for checking probabilistic channels, considering also properties of the probabilistic distribution of values.

References

1. J. Agat. Transforming out timing leaks. In *Proc. of the 27th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL '00)*, pages 40–53. ACM-Press, NY, 2000.
2. B. Alpern, A. J. Demers, and F. B. Schneider. Safety without stuttering. *Information Processing Letters*, 23(4):177–180, 1986.
3. R. Alur and D. L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, 1994.
4. R. Barbuti, N. De Francesco, A. Santone, and L. Tesei. A notion of non-interference for timed automata. *Fundamenta Informaticae*, 51:1–11, 2002.
5. D. E. Bell and L. J. LaPadula. Secure computer systems: Mathematical foundations and model. Technical Report M74-244, MITRE Corp. Bedford, MA, 1973.
6. D. Clark, C. Hankin, and S. Hunt. Information flow for algol-like languages. *Computer Languages*, 28(1):3–28, 2002.
7. E. S. Cohen. Information transmission in sequential programs. *Foundations of Secure Computation*, pages 297–335, 1978.
8. P. Cousot. Constructive design of a hierarchy of semantics of a transition system by abstract interpretation. *Theor. Comput. Sci.*, 277(1-2):47,103, 2002.
9. D. E. Denning and P. Denning. Certification of programs for secure information flow. *Communications of the ACM*, 20(7):504–513, 1977.

10. R. Giacobazzi and I. Mastroeni. Abstract non-interference: Parameterizing non-interference by abstract interpretation. In *Proc. of the 31st Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL '04)*, pages 186–197. ACM-Press, NY, 2004.
11. R. Giacobazzi and I. Mastroeni. Proving abstract non-interference. In *Annual Conference of the European Association for Computer Science Logic (CSL'04)*, volume 3210, pages 280–294. Springer-Verlag, 2004.
12. R. Giacobazzi and I. Mastroeni. Adjoining declassification and attack models by abstract interpretation. In *Proc. of the European Symposium on Programming (ESOP'05)*, volume 3444 of *Lecture Notes in Computer Science*, pages 295–310. Springer-Verlag, 2005.
13. R. Giacobazzi and I. Mastroeni. Generalized abstract non-interference for automata. In *In The Third International Workshop "Mathematical Methods, Models and Architectures for Computer Networks Security" (MMM-ACNS'05)*, volume 3685 of *Lecture Notes in Computer Science*, pages 221–234. Springer-Verlag, 2005.
14. R. Giacobazzi, F. Ranzato, and F. Scozzari. Making abstract interpretations complete. *J. of the ACM.*, 47(2):361–416, 2000.
15. S. Hunt and I. Mastroeni. The PER model of abstract non-interference. In *Proc. of The 12th Internat. Static Analysis Symp. (SAS'05)*, volume 3672 of *Lecture Notes in Computer Science*, pages 171–185. Springer-Verlag, 2005.
16. R. Joshi and K. R. M. Leino. A semantic approach to secure information flow. *Science of Computer Programming*, 37:113–138, 2000.
17. P. Laud. Semantics and program analysis of computationally secure information flow. In *In Programming Languages and Systems, 10th European Symp. On Programming, ESOP*, volume 2028 of *Lecture Notes in Computer Science*, pages 77–91. Springer-Verlag, 2001.
18. J. Morgado. Some results on the closure operators of partially ordered sets. *Portug. Math.*, 19(2):101–139, 1960.
19. A. Sabelfeld and A.C. Myers. Language-based information-flow security. *IEEE J. on selected areas in communications*, 21(1):5–19, 2003.
20. A. Sabelfeld and D. Sands. A PER model of secure information flow in sequential programs. *Higher-Order and Symbolic Computation*, 14(1):59–91, 2001.
21. C. Skalka and S. Smith. Static enforcement of security with types. In *ICFP'00*, pages 254–267. ACM Press, New York, 2000.
22. D. Volpano. Safety versus secrecy. In *Proc. of the 6th Static Analysis Symp. (SAS'99)*, volume 1694 of *Lecture Notes in Computer Science*, pages 303–311. Springer-Verlag, 1999.
23. D. Volpano and G. Smith. Probabilistic noninterference in a concurrent language. *Journal of Computer Security*, 7(2,3):231–253, 1999.
24. D. Volpano, G. Smith, and C. Irvine. A sound type system for secure flow analysis. *Journal of Computer Security*, 4(2,3):167–187, 1996.
25. D. Zanardini. Higher-order abstract non-interference. In *Proc. of the Seventh International Conference on Typed Lambda Calculi and Applications (TLCA '05)*, *Lecture Notes in Computer Science*. Springer-Verlag, 2005. To appear.
26. S. Zdancewic and A. C. Myers. Robust declassification. In *Proc. of the IEEE Computer Security Foundations Workshop*, pages 15–23. IEEE Computer Society Press, 2001.