EPTCS 51

Proceedings of the

8th International Workshop on

Security Issues in Concurrency

Paris, France, 30th August 2010

Edited by: Konstantinos Chatzikokolakis and Véronique Cortier

Published: 25th February 2011
DOI: 10.4204/EPTCS.51
ISSN: 2075-2180
Open Publishing Association

Table of Contents

Preface

This volume contains the proceedings of the 8th Workshop on Security Issues in Concurrency (SecCo 2010). The workshop was held in Paris, France on August 30th, 2010, as a satellite workshop of CONCUR'10. Previous editions of this workshop have been organized in Eindhoven (2003), London (2004), San Francisco (2005), Lisbon (2007), Toronto (2008) and Bologna (2009).

The aim of the SecCo workshop series is to cover the gap between the security and the concurrency communities. More precisely, the workshop promotes the exchange of ideas, trying to focus on common interests and stimulating discussions on central research questions. In particular, we called for papers dealing with security issues (such as authentication, integrity, privacy, confidentiality, access control, denial of service, service availability, safety aspects, fault tolerance, trust, language-based security, probabilistic and information theoretic models) in emerging fields like web services, mobile ad-hoc networks, agent-based infrastructures, peer-to-peer systems, context-aware computing, global/ubiquitous/pervasive computing.

We received 4 submissions (an unsually low number for SecCo), including one short paper. However all papers were of good quality; the three long papers were accepted for this volume (one with corrections) and the short one was presented at the workshop. We also had two great invited talks by Jean Goubault-Larrecq and Sjouke Mauw. The reviews have been carried out by the program committee of SecCo'10, which consisted of

- Kostas Chatzikokolakis, (University of Eindhoven, Netherlands; co-chair)
- Véronique Cortier, (LORIA - CNRS, France; co-chair)
- Cas Cremers, (ETH Zurich, Switzerland)
- Jerry den Hartog, (University of Eindhoven, Netherlands)
- Riccardo Focardi, (Universita Ca' Foscari di Venezia, Italy)
- Cédric Fournet, (Microsoft Cambridge, UK)
- Joshua Guttman, (Worcester Polytechnic Institute, USA)
- Jun Pang, (University of Luxembourg, Luxembourg)
- Michael Rusinowitch, (LORIA - INRIA Lorraine, France)
- Steve Schneider, (University of Surrey, UK)

We would like to thank all the persons that contributed to SecCo'10. First of all, the program committee, the invited speakers, the authors and all the participants that attended the workshop. We are also very grateful to the CONCUR'10 organizers, for taking care of all the local organization. We thank the editors of EPTCS (who will publish these proceedings electronically in the EPTCS series).

Eindhoven and Nancy, August 5, 2010

Kostas Chatzikokolakis
Véronique Cortier

K. Chatzikokolakis and V. Cortier (Ed.): 8th International Workshop
on Security Issues in Concurrency (SecCo 2010)
EPTCS 51, 2011, pp. ii–ii, doi:10.4204/EPTCS.51.0

“Logic Wins!”

Jean Goubault-Larrecq
ENS Cachan

Clever algorithm design is sometimes superseded by simple encodings into logic. In particular, we claim that it is particularly simple to encode sound abstractions of security protocols in H1, a decidable fragment of first-order Horn clauses. After reviewing a variant of Nielson, Nielson and Seidl’s work on H1 and the spi-calculus, we describe a verification algorithm designed with the same spirit, and which applies to hardware circuit descriptions written in VHDL. We shall describe the new challenges posed by VHDL, in particular the particular semantics of ’wait’ instructions, and the effect of signal updates and of timeouts.

K. Chatzikokolakis and V. Cortier (Ed.): 8th International Workshop
on Security Issues in Concurrency (SecCo 2010)
EPTCS 51, 2011, pp. iii–iii, doi:10.4204/EPTCS.51.0.1

A Spatial-Epistemic Logic for Reasoning about Security Protocols

Bernardo Toninho
CITI and Faculdade de Ciências e Tecnologia, Universidade Nova de Lisboa
Carnegie Mellon University, Pittsburgh PA, USA
Btoninho@cs.cmu.edu

Luís Caires
CITI and Faculdade de Ciências e Tecnologia, Universidade Nova de Lisboa
Luis.Caires@fct.unl.pt

Reasoning about security properties involves reasoning about *where* the information of a system is located, and how it evolves over time. While most security analysis techniques need to cope with some notions of information locality and knowledge propagation, usually they do not provide a general language for expressing arbitrary properties involving local knowledge and knowledge transfer. Building on this observation, we introduce a framework for security protocol analysis based on dynamic spatial logic specifications. Our computational model is a variant of existing π-calculi, while specifications are expressed in a dynamic spatial logic extended with an epistemic operator. We present the syntax and semantics of the model and logic, and discuss the expressiveness of the approach, showing it complete for passive attackers. We also prove that generic Dolev-Yao attackers may be mechanically determined for any deterministic finite protocol, and discuss how this result may be used to reason about security properties of open systems. We also present a model-checking algorithm for our logic, which has been implemented as an extension to the SLMC system.

1 Introduction

Among the several artifacts in the field of computer security, security protocols are indubitably a fundamental subject of study and research [12, 11]. Security protocols serve a variety of purposes, ranging from secrecy and authentication to forward secrecy and deniable encryption. A common trait of these protocols is their notoriously difficult design, which often leads to unforeseen vulnerabilities.

Therefore, it becomes essential to develop techniques that ensure the correctness of protocols, with respect to some specification of the properties they aim to establish. A wide range of language-based techniques have been proposed to analyze protocols and their correctness, such as type systems, process calculi or static analysis [3, 6, 2] which in many cases result in successful tools [5, 4, 13, 9].

In this paper we propose a framework for protocol analysis based on process calculus models and logic specifications. While the usage of process calculi and logic in this context is not new [14, 8, 2], our approach stems from the fact that many interesting properties of such systems are often a function of what information the several parts of a system may or may not obtain. While other frameworks (e.g., Avispa [4] and Casper [13]) allow one to efficiently verify a wide range of interesting security properties, these are not usually stated in this high-level knowledge oriented approach.

Our contribution consists of a dynamic spatial epistemic logic that allows reasoning about systems (modelled in a variant of the applied π-calculus [2]) at three levels: the *dynamics* of systems and subsystems, the *spatial* arrangement of systems and subsystems, and the *knowledge* (the obtainable information)

K. Chatzikokolakis and V. Cortier (Ed.): 8th International Workshop
on Security Issues in Concurrency (SecCo 2010)
EPTCS 51, 2011, pp. 1–15, doi:10.4204/EPTCS.51.1

```
S(pkp,pkq, sks, pks) = c?(h).select { [pkp = getpk(h)].c!(enc_as(pkp,sks)).S(pkp,pkq) ;
                                      [pkq = getpk(h)].c!(enc_as(pkq, sks)).S(pkp,pkq) };

defproc P(skp,hostQ,pks) = c!(hostQ).c?(m).let pkQ = dec_as(m,pks) in
                            new sK in c!(enc_as(sK,pkQ)).c!(enc(v,sK)).ok!(v);

Q(skq) = c?(m1).let sK = dec_as(m1,skq) in c?(m2).let val = dec(m2,sK) in ok!(val);

defproc Sys = new skp, skq in let pkp = pk(skp) in let pkq = pk(skq)
                              in let hP = host(pkp) in let hQ = host(pkq)
                              in (S(pkp,pkq) | P(skp,hQ) | Q(skq));

World = Sys | Attacker(Sys);

prop pqK = eventually (knows v | knows v | not (knows v))
           and always (2 | not (knows v));

check World |= pqK;

* Process World satisfies the formula pqK *
```

Figure 1: A Motivating Example

of systems and subsystems. The goal is to produce an expressive property language with which we can reason about a protocol by separating it into its different agents (malicious and otherwise), and then reason about the knowledge they can obtain and how it can evolve over time. This enables us to express interesting security properties in a very direct way (eg. agents P and Q can obtain value v, while agents A and S cannot). To clarify our approach, consider the example of Fig. 1.

We have a system `Sys` composed of three processes: P, Q and a key distribution server S. P wishes to exchange a value v with Q. To do so, he requests Q's public key from S, which S emits in a signed message. P then uses it to encrypt a generated symmetric session key and sends the key to Q. Afterwards P will send v encrypted with the session key and terminate. Q will receive the message, decrypt it to obtain v and terminate. We further model the system running with a malicious agent, defined through the primitive `Attacker(Sys)`. This agent consists of a Dolev-Yao attacker which we discuss in Section 4.1.

For this protocol to be correct, it must be the case that the malicious agent interacting with the system can never know v. Consider, however, a slightly stronger property: P and Q want to exchange v securely (with respect to the malicious agent) but they also do not completely trust the server S. They trust it to at least distribute the appropriate keys, but want some assurances that even though S operates according to protocol, it doesn't obtain the value v by observing the data exchanges between P and Q.

This property, while not impossible to state in other frameworks, would usually require some sort of *ad-hoc* modification to the model (e.g, internalizing the server in an attacker, which seems like an indirect strategy at best and may not necessarily yield the correct model). In our framework, the property can be directly stated by combining our epistemic and spatial operators. A formula that reflects such a property is `pqK`: first we state that the system can evolve to a configuration where two of its subsystems (P and Q) know v, but the remaining parts of the system do not. This illustrates the expressiveness of the logic in terms of reasoning about the knowledge of several parts of the complex system. Secondly, we state that throughout *all* executions of the system, a part of it will never know v (`2` indicates that there must be two agents running with the part that does not know v – a precise definition is given in Section 3). By combining spatial reasoning with epistemic reasoning, we can state rich properties of the knowledge of agents (and groups of agents) – both adversaries and principals – within a complex system, and how they can share or restrict that knowledge over time.

While our framework is aimed at reasoning about closed systems, meaningful analyses of security

protocols must necessarily consider attackers. Traditionally, attackers are modelled by an adversarial environment which interacts with the protocol. In our closed system approach, we develop a way internalizing an arbitrary attacker within a closed system by *automatically* (or semi-automatically) deriving a process representation of such an attacker. This representation makes use of special primitives built into the process calculus to greatly simplify the actual modelling of the attacker, to the extent that even if the attacker generation is done semi-automatically, it *never* requires us to actually encode specific attacks. In this work, we show that it is possible to *automatically* derive an attacker (behaving as a Dolev-Yao adversarial environment) for any finite protocol. To fully automate our technique (at the implementation level), further work is needed (as discussed in Section 4.1); the focus of the current paper being essentially on the expressiveness issues. In any case, we already provide tool support for arbitrary passive attackers and for *bounded* Dolev-Yao attackers (where the bound concerns the size of generated messages); this technique can already be used to automatically find attacks, eg., as illustrated in the example of Section 4.2.

The technical contributions of this work are as follows. We develop a process calculus model for security protocols (Section 2.2), inspired in existing π-calculi, supporting explicit modeling of adversarial agents, at an adequate level of abstraction. We introduce a new dynamic spatial epistemic logic (Section 3), oriented for reasoning about spatial distribution of information. We develop a logic-based theory of knowledge deduction (Section 3.2) for our models, proved sound, complete and decidable. This presentation was used in our model-checking algorithms. We discuss attacker representations (Section 4.1), and how it is possible to produce a generic Dolev-Yao attacker for finite protocols. We also show how to model and verify correspondence assertions (Section 4.2) in our framework. Finally, we implemented a model-checking algorithm for the logic as an extension to the SLMC tool, producing the first proof of concept tool aimed at security protocol analysis using spatial logic model checking. The proofs of our technical results are detailed in [17].

2 Process Model

In this section we introduce our process model, starting with some preliminary notions on terms and equational theory and then introducing our process calculus.

2.1 Terms and Equational Theories

Data exchanged by processes is modeled by terms of a term algebra. In order to capture cryptographic operations and data structuring, we will consider term algebras with equational theories (cf. [2]).

We assume an infinite set of variables ranged over by x, y, z, an infinite set of names Λ ranged over by m, n and range over terms with s, t, v. Terms are defined from names and variables by applying function symbols. We thus consider a given term algebra to be defined from a signature Σ and an equational theory E that defines the "semantics" of the function symbols in Σ. An equational theory is a congruence relation defined by a set of equations of form $t = s$.

In certain circumstances, an equational theory may give rise to a set of rewrite rules by orienting each equation to produce the rule $t \to s$, in such a way that two terms are equal modulo E whenever that have a common reduct under rewriting. This is the case of subterm convergent equational theories [1], which are the ones that we will focus on in this work (other equational theories, such as AC theories, can also be applied in this fashion, however with a slightly different formal treatment as detailed in [1]). A subterm convergent system is a convergent rewrite system in which in every rewrite rule the right-hand

side is a proper subterm of the left-hand side. In this paper, we will assume a general rewrite theory $\mathcal{R}$ subject to the conditions above. Given a rewrite rule $t \to s$, we call the outermost function symbol in t a *destructor*, since the application of the rule may open the internal structure of inner terms in t to produce the term s. We classify the remaining function symbols, that never occur as a destructor, as *constructors*. For example, for signature $\Sigma \triangleq \{\mathtt{enc}/2;\mathtt{dec}/2\}$ and equational theory $E \triangleq \{\mathtt{dec}(\mathtt{enc}(x,y),y) = x\}$, $\mathtt{dec}$ is a destructor and $\mathtt{enc}$ a constructor. We range over constructors with f and destructors with δ.

We denote the set of names of a term T by $names(T)$ and the depth of a term as $|T|$ (the depth is the length of the longest path in the tree representation of the term). We state that a term is ground if it does not contain variables. We denote by $=_E$ the usual congruence relation induced by the set of equations E (which can be decided through term rewriting since R is convergent). We write $\mathfrak{F}(\psi)$ for the DY (Dolev-Yao) equational closure of a set of terms ψ, that is, the set of all values (destructor-free terms) generated by terms of ψ through function application, modulo the equational theory. This closure represents all possible information that may be produced from a set of terms while following the rules of the equational theory, which if we interpret a set of terms as a set of messages, is the usual notion of knowledge from the Dolev-Yao model.

Definition 2.1 (Equational Closure) *Given a rewrite theory $\mathcal{R}$, the DY equational closure of a set of terms ψ, noted $\mathfrak{F}(\psi)$, is the least set of terms such that:*

1. $\psi \subseteq \mathfrak{F}(\psi)$
2. $\forall f \in \Sigma$. *if f a constructor and* $t_1,\ldots,t_k \in \mathfrak{F}(\psi)$ *then* $f(t_1,\ldots,t_k) \in \mathfrak{F}(\psi)$
3. $\forall \delta \in \Sigma$. *if δ a destructor and* $t_1,\ldots,t_k \in \mathfrak{F}(\psi)$ *and* $\delta(t_1,\ldots,t_k) \to t'$ *then* $t' \in \mathfrak{F}(\psi)$

When interpreting the DY equational closure of a set of terms as obtainable knowledge, we can state knowledge derivation through term derivation.

Definition 2.2 ***(Knowledge Derivation).*** *Given sets of terms ψ and ϕ, we say that ϕ may be derived from ψ (written $\psi \models \phi$) if and only if $\phi \subseteq \mathfrak{F}(\psi)$.*

The general idea is that one may can derive a piece of information if it can be generated by combining pieces of information using the rules of the equational theory. Given these basic notions relative to terms, equational theories, and knowledge derivation, we may now present our process calculus model.

2.2 Process Calculus

It is known that the high level of abstraction of the π-calculus, convenient from a foundational perspective, is not suitable for modeling cryptographic techniques as necessary for analyzing security protocols.

We therefore adopt an extension to the π-calculus that extends the base values of the language with functional terms (cf. Section 2.1), that can be seen as a fragment of the Applied π-calculus [2]. We choose this calculus over the applied π-calculus mainly for simplicity reasons, not requiring active substitutions nor frames given that our goal is to use our logic to observe terms.

We model cryptographic operations by defining such operations in a term algebra. The calculus is thus aimed at the explicit modeling of agents involved in security protocols, both principals and adversaries. Principals are modeled standardly, using terms to model cryptographic terms. Adversaries are modeled as processes (cf. Section 4.1) using the attacker output prefix - a non-deterministic output of terms that can be generated from known values, which enables reasoning directly about attacker knowledge using our logic.

Definition 2.3 (Processes) *Given a signature Σ, an infinite set of names ranged over by m,n, and an infinite set of variables ranged over by x,y,z, the set of processes (P,Q), of actions α and of terms T are defined in Fig. 2.*

$$
\begin{array}{rcll}
P,Q & ::= & \mathbf{0} & \text{(Null Process)} \\
 & | & P \mid Q & \text{(Parallel Composition)} \\
 & | & (\nu n)P & \text{(Name Restriction)} \\
 & | & \alpha.P & \text{(Action Prefix)} \\
 & | & P + Q & \text{(Choice)} \\
 & | & \text{let } x = T \text{ in } P & \text{(Let Construct)}
\end{array}
\qquad
\begin{array}{rcll}
\alpha & ::= & m(x) & \text{(Input)} \\
 & | & m\langle T\rangle & \text{(Output)} \\
 & | & m\langle *\rangle & \text{(Attacker Output)} \\
 & | & [T_1 = T_2] & \text{(Test)} \\
 & & & \\
T & ::= & n & \text{(Name)} \\
 & | & x & \text{(Variable)} \\
 & | & f(T_1,\ldots,T_a) & \text{(Function)}
\end{array}
$$

Figure 2: Process Calculus Syntax

$$sub(\delta(t_1,\ldots,t_n)) \triangleq sub(t_1)\cup\cdots\cup sub(t_n) \qquad \frac{n \text{ is a name}}{sub(n)\triangleq n} \qquad \frac{x \text{ is a variable}}{sub(x)\triangleq \emptyset}$$

$$\frac{\nexists t_i \text{ with a variable or a destructor}}{sub(f(t_1,\ldots,t_n))\triangleq f(t_1,\ldots,t_n)} \qquad \frac{\exists t_i \text{ with a variable or a destructor}}{sub(f(t_1,\ldots,t_n))\triangleq sub(t_1)\cup\cdots\cup sub(t_n)}$$

Figure 3: Relevant Subterms

Before introducing the semantics of our calculus, we present some definitions that pertain to obtaining the *relevant terms* of a process that are necessary for our semantics.

A destructor function symbol denotes computation at the term level. If such computations are valid (under the equational theory), then the term containing the destructor can be rewritten as one that only has constructors. On the other hand, if such a term cannot be reduced (e.g $\mathtt{dec}(\mathtt{enc}(m,k_1),k_2)$), it has no interesting meaning and has no place being communicated. To obtain the values (destructor free normal forms) of a process, we define a relation $\vdash_k$ that extracts the set of values ψ that occur in a process ($P \vdash_k \psi$). However, some care is needed in the definition of $\vdash_k$ since a term may contain bound names or variables. For instance, in the process $a(x).a\langle\mathtt{dec}(x,k)\rangle.\mathbf{0}$, the term $\mathtt{dec}(x,k)$ is not a proper value since it contains the variable x. In these situations, our extraction has to be such that it will produce a set containing k but not x (nor $\mathtt{dec}(x,k)$). Similarly, when we consider the terms that are to be the object of our attacker output, while it is true that outputting a term containing a variable would be senseless, it is correct to output a term that contains a restricted name, even though the attacker may not be able to use the name in other messages.

To take all this into account, we define a procedure *sub* that extracts the *relevant subterms* (not containing variables or destructors) of a term, and a procedure $\uparrow$ used to eliminate terms with restricted names.

Definition 2.4 (Relevant Subterms) *Given a term M we define the set of its relevant subterms, written $sub(M)$, by the rules of Fig. 3.*

Definition 2.5 (Name Occurrence Term Removal) *We define the removal of terms from a set ψ in which the name x occurs, $\psi\uparrow x$, as:* $\psi\uparrow x \triangleq \{t \mid t\in\psi : x\notin names(t)\}$.

Definition 2.6 (Relevant Term Extraction) *Given a process P, the set ψ of relevant terms of P, written $P\vdash_k\psi$, is defined by the rules of Fig. 4.*

For our attacker output we collect all ground terms that occur in the process, which we denote by $gt(P)$.

The semantics of our calculus are defined standardly, modulo α-conversion of bound names and variables, by a structural congruence relation, labelled transition and reduction, as follows. We denote by $fn(P)$ and $fv(P)$ the set of free-names and free-variables of process P, respectively.

$$\frac{P \vdash_k \varphi \quad Q \vdash_k \psi}{P + Q \vdash_k (\varphi \cup \psi)} \qquad \frac{P \vdash_k \varphi \quad Q \vdash_k \psi}{P \mid Q \vdash_k (\varphi \cup \psi)} \qquad \frac{P \vdash_k \varphi}{n(x).P \vdash_k \varphi} \qquad \frac{P\{n \leftarrow M\} \vdash_k \varphi}{\text{let } n = M \text{ in } P \vdash_k \varphi \cup sub(M)}$$

$$\frac{}{\mathbf{0} \vdash_k \emptyset} \qquad \frac{P \vdash_k \varphi}{x\langle M \rangle.P \vdash_k \varphi \cup sub(M)} \qquad \frac{P \vdash_k \varphi}{(\nu n)P \vdash_k \varphi \uparrow n} \qquad \frac{P \vdash_k \varphi}{[M = N].P \vdash_k \varphi \cup sub(M) \cup sub(N)}$$

Figure 4: Relevant Term Extraction

$n \notin fn(P) \cup fv(P) \Rightarrow P \mid (\nu n)Q \equiv (\nu n)(P \mid Q)$	$P \mid \mathbf{0} \equiv P$
$(\nu n)\mathbf{0} \equiv \mathbf{0}$	$P \mid Q \equiv Q \mid P$
$(\nu n)(\nu m)P \equiv (\nu m)(\nu n)P$	$P \mid (Q \mid R) \equiv (P \mid Q) \mid R$
$M =_E M' \Rightarrow \text{let } x = M \text{ in } P \equiv \text{let } x = M' \text{ in } P$	$P + Q \equiv Q + P$
$M =_E M' \Rightarrow m\langle M \rangle.P \equiv m\langle M' \rangle.P$	$P + (Q + R) \equiv (P + Q) + R$
$M_1 =_E M_1' \Rightarrow [M_1 = M_2].P \equiv [M_1' = M_2].P$	$[M_1 = M_2].P \equiv [M_2 = M_1].P$

Figure 5: Structural Congruence

Definition 2.7 (Structural Congruence) *Structural congruence $\equiv$ is the least congruence relation on processes defined by the rules of Fig. 5.*

We augment the standard structural congruence laws of the π-calculus with rules that equate processes modulo the equality $=_E$ of the equational theory. These laws are essential in our semantics because they allow us to block processes performing actions that use terms that are not values (i.e. terms that contain destructors).

Our semantics, which we now present, capture these *destructor freedom* conditions. If a process is attempting to use a term that contains a destructor, we use structural congruence to rewrite the term destructor-free and reduction proceeds. If the term cannot be rewritten destructor-free, reduction halts. These restrictions ensure that all received terms are actual values, and not some arbitrary erroneous term. Note the semantics of our attacker output, expressed in the *Attacker* rule, that enable the output to emit any message that can be generated by the process, given its ground terms and some fresh values.

Definition 2.8 (Reduction Semantics) *The reduction relation $P \longrightarrow Q$ over closed processes is defined as the least relation closed under the rules of Fig. 6.*

Definition 2.9 (Labelled Transition Semantics) *The labelled transition relation $P \xrightarrow{\alpha} Q$ is the least relation on closed processes closed under the rules of Fig. 7.*

Our labelled semantics is not intended to characterize a complete notion of behavioral equivalence as could be expected, but rather to allow the observation of actions in our logic. Despite not belonging to the scope of this work, we can point out that our labelled semantics do not allow for a complete characterization of behavioral equivalence, in the sense that our rules reveal information in a way that induces a higher discriminative power then that of behavioral equivalence.

3 Logic

Considering it is common to reason about security by reasoning about the knowledge of principals, we explore key aspects of dynamic spatial logics, such as local reasoning, to develop a logic that can reason about epistemic, dynamic and spatial properties of agents.

We propose an extension to a dynamic spatial logic [7] to enable reasoning at the term level. Our extension consists of adding two epistemic modalities: $\mathbb{K}\phi$ denotes the ability of an agent to derive ϕ from its knowledge, and $\mathsf{S}x.A$ allows us to mention values that are only known by an agent (e.g. secrets).

$$\frac{M \text{ is destructor-free}}{\mathit{let}\ x = M\ \mathit{in}\ P \longrightarrow P\{x \leftarrow M\}}(\mathit{Let}) \qquad \frac{M \text{ is destructor-free}}{n\langle M\rangle.P + R \mid n(x).Q + S \longrightarrow P \mid Q\{x \leftarrow M\}}(\mathit{Sync})$$

$$\frac{M_1 \text{ and } M_2 \text{ are destructor-free} \quad M_1 =_E M_2}{[M_1 = M_2].P \longrightarrow P}(\mathit{Test}) \qquad \frac{P \longrightarrow Q}{P \mid R \longrightarrow Q \mid R}(\mathit{Par}) \qquad \frac{P \longrightarrow Q}{(\nu n)P \longrightarrow (\nu n)Q}(\mathit{Scope})$$

$$\frac{P \equiv P' \quad P' \longrightarrow Q' \quad Q' \equiv Q}{P \longrightarrow Q}(\mathit{Cong}) \qquad \frac{M \in \mathfrak{F}(ct(Q) \cup \overline{n}) \quad \overline{n}\ \mathit{fresh}}{c(x).P + R \mid c\langle *\rangle.Q + S \longrightarrow (\nu \overline{n})(P\{x \leftarrow M\} \mid Q)}(\mathit{Attacker})$$

Figure 6: Reduction Semantics

$$\frac{P \longrightarrow Q}{P \xrightarrow{\tau} Q}\ (\text{Tau}) \qquad \frac{M \text{ is destructor-free}}{n\langle M\rangle.P \xrightarrow{n\langle M\rangle} P}\ (\text{Out}) \qquad \frac{M \text{ is destructor-free}}{n(x).P \xrightarrow{n(M)} P}\ (\text{Inp})$$

$$\frac{M \in \mathfrak{F}(gt(P) \cup \overline{s}) \quad \overline{s}\ \mathit{fresh}}{n\langle *\rangle.P \xrightarrow{\nu \overline{s}.n\langle M\rangle} P}\ (\text{AttackerOut}) \qquad \frac{P \xrightarrow{\alpha} Q \qquad \forall n \in \overline{u}\text{: n} \notin \mathit{names}(\alpha)}{(\nu \overline{u})P \xrightarrow{\alpha} (\nu \overline{u})Q}\ (\text{Res})$$

$$\frac{P \xrightarrow{n\langle M\rangle} P' \quad \overline{s} \subseteq \mathit{names}(M) \text{ and } \overline{s} \subseteq \overline{u} \quad \overline{u'} = \overline{u} \setminus \overline{s}}{(\nu \overline{u})P \xrightarrow{\nu \overline{s}.n\langle M\rangle} (\nu \overline{u'})P'}\ (\mathit{BoundOut}) \qquad \frac{P \equiv P' \quad P' \xrightarrow{\alpha} Q' \quad Q' \equiv Q}{P \xrightarrow{\alpha} Q}\ (\mathit{Cong})$$

Figure 7: Labelled Transition Semantics

Our intent is to couple the ability to reason about properties of space and behavior with that of reasoning about derivable information modulo the equational theory. Our notion of knowledge is therefore the ability of an agent to derive terms from the information it possesses.

3.1 Syntax and Semantics

The syntax and semantics of our logic are presented in Fig. 8. We refer to ϕ, ψ as knowledge formulas and ambivalently use ϕ, ψ to denote both knowledge formulas and finite sets of terms. The boolean connectives are standard. $\mathbf{0}$ denotes the empty process; $A \mid B$ denotes a process that can be partitioned in two components, one satisfying A and the other satisfying B; $\mathsf{H}x.A$ allows us to mention restricted names of processes in formulas; $\alpha.A$ denotes a process can perform action α and continue as a process satisfying A; $\Box A$ and $\Diamond A$ denote "always in the future" and "sometime in the future", respectively. $\mathbb{K}\phi$ holds of a process that has the ability to derive the terms denoted by ϕ, that is, the ability to know ϕ; $\mathsf{S}x.A$ holds of a process that satisfies property A that depends on a value that is secret to a process – a term containing a restricted name. It is also useful to define an auxiliary *counting predicate* (written as $\mathbf{n}$, where n is a natural number), that allows us to count the number of sub-processes within a process. For instance, a process consisting of a single thread would satisfy the formula $\mathbf{1}$ defined as $\neg\mathbf{0} \wedge \neg(\neg\mathbf{0} \mid \neg\mathbf{0})$, while a process consisting of two sub-processes would satisfy the formula $\mathbf{2}$ defined as $\neg\mathbf{0} \wedge \neg\mathbf{1} \wedge \neg(\neg\mathbf{0} \mid \neg\mathbf{0} \mid \neg\mathbf{0})$, and so on.

With this logic, we can state properties about the knowledge of agents (and not only adversarial ones) over time, such as "it is never the case that the secret key is known by 3 subsystems":

$$\neg\Diamond \mathsf{H}\, \mathit{key}.(\mathbb{K}\, \mathit{key} \mid \mathbb{K}\, \mathit{key} \mid \mathbb{K}\, \mathit{key})$$

or "it is always the case that 2 agents know the key and one does not": $\Box \mathsf{H}\, \mathit{key}.(\mathbb{K}\, \mathit{key} \mid \mathbb{K}\, \mathit{key} \mid \neg\mathbb{K}\, \mathit{key})$. Since the semantics of our logic blur together processes that are structurally congruent (e.g. $P \mid Q$ and

$$
\begin{array}{lcll}
A,B & ::= & \mathbf{T} & \text{(True)}\\
 & | & \neg A & \text{(Negation)}\\
 & | & A \wedge B & \text{(Conjunction)}\\
 & | & \mathbf{0} & \text{(Void)}\\
 & | & A \mid B & \text{(Composition)}\\
 & | & \mathsf{H}x.A & \text{(Hidden quantification)}\\
 & | & \alpha.A & \text{(Action)}\\
 & | & \Box A & \text{(Always)}\\
 & | & \Diamond A & \text{(Eventually)}\\
 & | & @n & \text{(Free-name Predicate)}\\
 & | & \mathbb{K}\varphi & \text{(Knowledge)}\\
 & | & \mathsf{S}x.A & \text{(Secret quantification)}\\
 & & & \\
\phi,\psi & ::= & \varphi \wedge \psi & \text{(Conjunction)}\\
 & | & t & \text{(Term)}\\
 & | & \top & \text{(True)}
\end{array}
$$

$$
\begin{array}{lcl}
P \models \mathbf{T} & \triangleq & \textit{True}\\
P \models \neg A & \triangleq & \textit{not } P \models A\\
P \models A \wedge B & \triangleq & P \models A \textit{ and } P \models B\\
P \models \mathbf{0} & \triangleq & P \equiv \mathbf{0}\\
P \models A \mid B & \triangleq & \exists Q,R.P \equiv Q \mid R \textit{ and } Q \models A \textit{ and } R \models A\\
P \models \mathsf{H}x.A & \triangleq & \exists Q.P \equiv (\nu n)Q \textit{ and } Q \models A\{x \leftarrow n\}\\
P \models \alpha.A & \triangleq & \exists Q.P \xrightarrow{\alpha} Q \textit{ and } Q \models A\\
P \models \Box A & \triangleq & \forall Q \text{ s.t } P \xrightarrow{\tau^*} Q \textit{ then } Q \models A\\
P \models \Diamond A & \triangleq & \exists Q.P \xrightarrow{\tau^*} Q \textit{ and } Q \models A\\
P \models @n & \triangleq & n \in fn(P)\\
P \models \mathbb{K}\phi & \triangleq & P \vdash_k \psi \textit{ and } \psi \models \phi\\
P \models \mathsf{S}x.A & \triangleq & \exists Q,t.P \equiv (\nu k)Q \textit{ and } Q \models A\{x \leftarrow t\}\\
 & & \textit{and } Q \vdash_k \phi \textit{ such that } t \in \phi\\
 & & \textit{and } k \in names(t)
\end{array}
$$

Figure 8: Logic Syntax and Semantics

$Q \mid P$), we can use the free-name predicate to "tag" specific subsystems and reason about their knowledge explicitly: $\Box\mathsf{H}\,key.(@tag \wedge \mathbb{K}\,key \mid \mathbf{T})$ which denotes "it is always the case that an agent with the free name *tag* knows the key" (this subsumes the need for an indexed knowledge operator such as that in [15]).

Notice how the expressiveness of the logic arises from the ability to combine the three types of modalities: dynamic ($\Box, \Diamond$), spatial (H, $\mid$) and epistemic ($\mathbb{K}$). The dynamic connectives allow us to range over a specific execution or all possible executions, the spatial connectives allow us to mention restricted names (usually used to model keys and nonces) and to refer to subsystems, and the epistemic connectives allow us to analyze derivable terms of a process.

The semantics for $\mathbb{K}\phi$ pose a challenge in the sense that they use the notion of knowledge derivation from Section 2.1. While this definition is adequate from a semantic perspective, it makes use of the DY equational closure of a set which is not stable by reduction of terms, and thus doesn't provide a clear way of algorithmically determining if $\psi \models \phi$. We approach the problem with a purely logical approach and characterize knowledge derivation with a structural proof system for knowledge formulas, unlike the approach of [1].

3.2 Proof System for Knowledge Formulas

Our proof system, formulated as a sequent calculus, is equipped with rules from the equational theory in order to consider the ability to combine terms to generate new information. Each rule of our calculus represents a possible computational step that an agent can perform on terms to produce a new term. Intuitively, if a sequent $\Gamma \vdash \phi$ is derivable, the knowledge formula ϕ is deducible from the knowledge represented by Γ.

Definition 3.1 (Proof System K for Knowledge Formulas) *The sequent calculus formulation of our proof system K for knowledge formulas is defined by the rules of Fig. 9.*

The rules for identity and conjunction are standard. Rule *funRight* states that we are justified in concluding a complex term if we can derive its subterms. Rule *AttLeft* states that all that can be derived from a complex term $f(t_1,\ldots,t_n)$ can also be derived from its subterms; rule *DestrLeft* reflects the equalities of the equational theory: what can be deduced from s can also be deduced from terms equal to s under the equational theory.

$$\frac{}{\Gamma, A \vdash A}(\text{Id}) \qquad \frac{\Gamma, A, B \vdash C}{\Gamma, A \wedge B \vdash C}\ (\wedge\text{: left}) \qquad \frac{\Gamma \vdash A \quad \Gamma \vdash B}{\Gamma \vdash A \wedge B}\ (\wedge\text{: right})$$

For every constructor function symbol f with arity n, such that $f \in \Sigma$:

$$\frac{\Gamma \vdash t_1 \ldots \Gamma \vdash t_n}{\Gamma \vdash f(t_1, \ldots, t_n)}\ (\text{funRight}) \qquad \frac{\Gamma, f(t_1, \ldots, t_n) \vdash C}{\Gamma, t_1, \ldots, t_n \vdash C}\ (\text{AttLeft})$$

For every equation $f(t_1, \ldots, t_n) = s \in E$:

$$\frac{\Gamma, s \vdash C}{\Gamma, f(t_1, \ldots, t_n) \vdash C}\ (\text{DestrLeft})$$

Figure 9: Proof System for Knowledge Formulas

For the sequent calculus K, we establish the results of *soundness*, *completeness* and *decidability*.

Theorem 3.2 (Soundness of K) *Given a set of terms S and a term A, if $S \vdash A$ then $S \models A$.*
Proof: By induction on the derivation of $S \vdash A$ ■

Theorem 3.3 (Completeness of K) *Given a set of terms S and a term A, if $S \models A$ then $S \vdash A$.*

Theorem 3.4 (Decidability of K) *For any set of terms S and term A, $S \vdash A$ is decidable.*

The proofs of completeness and decidability rely on a finite approximation result for the DY equational closure of a set of terms. More concretely, for each finite set of terms S and equational theory, it is possible to build a finite set $b(S)$ from which all terms in the DY equational closure of S may be determined.

Proposition 3.5 (Approximation of $\mathfrak{F}(S)$) *Let S be a finite set of terms. We may construct in polynomial time an approximation to $\mathfrak{F}(S)$, named $b(S)$, a finite set with the following property:*

$$\forall M \in \mathfrak{F}(S), \exists\, C, \overline{t} \in b(S) \text{ such that } M = C[\overline{t}]$$

where $C[-]$ is a functional context solely built out of constructors.
Proof: The finite approximation $b(S)$ is built from the terms of S by interpreting the rewrite rules of the theory as contexts of a bounded size. Therefore, applying function symbols to terms of S up to the bound of the context produces a new term by then applying the rewrite rule. This procedure is iterated, eventually reaching a fix-point, due to the subterm convergency property of the equational theories (the idea is that each time we produce a new term, the term will be smaller then the terms used to generate it). The resulting computable set has the property that defines our approximation [17]. ■

The approximation $b(S)$ is such that all terms of $\mathfrak{F}(S)$ can be built from terms of $b(S)$ just by applying constructors, no longer requiring the equations from the theory. Completeness follows from the fact that our proof system is able to emulate the computation steps required to generate the approximation. Given a set of terms S, $b(S)$ is generated by applying functions to terms of S, applying a rewrite rule to the resulting term and iterating. Thus, our proof system is complete since the computation steps of $b(S)$ may be emulated by the rules of proof system K, and we may then apply function symbols to terms of $b(S)$ to produce terms of $\mathfrak{F}(S)$. The latter is trivial due to *funRight* and *AttLeft*. The former we prove through the following lemma.

Lemma 3.6 ***(Completeness of K i.r.t the Approximation).*** *Given a set of terms S ,if $t \in b(S)$ then $S \vdash t$.*
Proof: Through instances of AttLeft *it is possible to apply functions to terms of S up to the bound of the context used in $b(S)$. Through an instance of* DestrLeft *the corresponding rewrite rule can applied, and through* Id *the new term is derived at the root of the proof tree [17].* ■

To emulate the iteration with the proof system, that is, to perform similar computations with S and the new term, the auxiliary result of reasoning with cuts is required.

Lemma 3.7 ***(Cut Admissibility in K).*** *If* $\Gamma \vdash A$ *and* $\Gamma, A \vdash C$ *then* $\Gamma \vdash C$.
Proof: See [17].

Using Cut, the proof system is able to emulate the iterative procedure by building the previously described proof tree that allows the derivation of a new term, and using the new term as the cut formula. This technique can then be applied to produce any term of $b(S)$, as required. Since the computation of $b(S)$ always terminates, Theorem 3.4 holds.

3.3 Model-Checking

We know that model-checking is decidable for the logic without the new modalities [7], for the class of bounded processes. Therefore, we need only show that our two modalities preserve decidability.

Proposition 3.8 (Decidability of model-checking $\mathbb{K}$) *Let ϕ be a finite set of terms. Checking that $P \models \mathbb{K}\phi$ is decidable.*

The above proposition holds since for any process P it is possible to collect its set of relevant terms ψ ($P \vdash_k \psi$), compute the finite approximation $b(\psi)$ and check that each term in ϕ can be constructed from terms of $b(\psi)$ by application of constructors.

Proposition 3.9 (Decidability of model-checking $\mathsf{S}x.A$) *Checking that $P \models \mathsf{S}x.A$ is decidable.*

Decidability of $\mathsf{S}x.A$ follows from the fact that if $P \equiv (\nu n)Q$, it is possible to collect the set ψ of relevant terms of process Q, pick some term t from ψ that contains the name n and check that $Q \models A\{x \leftarrow t\}$. Given that model-checking the core logic with $\mathbb{K}$ is decidable, it follows that checking $P \models \mathsf{S}x.A$ is decidable and therefore model-checking for our logic is decidable for the class of bounded processes.

Theorem 3.10 (Decidability of Model-Checking) *Checking that $P \models A$ is decidable for the class of bounded processes.*

4 Expressiveness and Extensions

Having presented our framework, we discuss some extensions to our work that can be used to model and analyze systems. In particular, we discuss the representation of attackers and modeling and verification of correspondence assertions [18] in our framework.

4.1 Modeling Attackers

To analyze a security protocol one usually needs to consider how it behaves in any possible environment. While our logic focuses on the analysis of closed systems, it is possible to verify properties of a system in an arbitrary environment, by internalizing an arbitrary attacker in the system. The general idea is that, for any process P, we may determine a process Q (making essential use of the attacker prefix construct) such that $P|Q$ reaches some state whenever P reaches an equivalent state when placed in an arbitrary environment. While the explicit specification of an attacker for a given protocol may not be easy, our approach to represent the attacking environment is quite different and general, and may indeed be used to find attacks (see example in Section 4.2). We can generically model a Dolev-Yao [12] attacker in our framework by considering the number of message exchanges and the communication channels used in a protocol.

$$
\begin{array}{lcl}
A(K) & \triangleq & (\nu K_{ab}, N)\, c\langle \mathtt{enc}(\mathtt{pair}(K_{ab}, N), K)\rangle.c(x).[N-1 = \mathtt{dec}(x, K_{ab})] \\
B(K) & \triangleq & c(x).\mathtt{let}\ K_{ab} = \mathtt{fst}(\mathtt{dec}(x,K))\ N = \mathtt{snd}(\mathtt{dec}(x,K))\ \mathtt{in}\ c\langle \mathtt{enc}(N-1, K_{ab})\rangle \\
Sys & \triangleq & (\nu K)\,(A(K) \mid B(K))
\end{array}
$$

Figure 10: Modeling the Example

$$
\begin{array}{lcl}
Attacker & \triangleq & c(x).c\langle * \rangle.c(y).c\langle * \rangle.mem\langle x,y\rangle \\
World & \triangleq & (Sys \mid Attacker) \\
World & \models & \neg\Diamond \mathsf{H}k.(\mathbf{2} \mid (@mem \wedge \mathbb{K}k))
\end{array}
$$

Figure 11: An Attacker for the Example

Considering an arbitrary protocol modeled as a process, the role of an attacker is to intercept all communications of the principals and be able to inject any message it can produce, given its knowledge at the time, at any point where a principal expects to receive a message (cf. our attacker output). Thus, a Dolev-Yao attacker consists of a process that for all outputs of the protocol performs an input (storing the received message) and for all inputs performs an attacker output. For instance, consider the following protocol, where K is a shared key, N a fresh value and K_{ab} a session key generated by A:

$$
\begin{array}{l}
A \rightarrow B : \{K_{ab}, N\}_K \\
B \rightarrow A : \{N-1\}_{K_{ab}}
\end{array}
$$

In our process model, such a protocol would be represented as done in Fig. 10 (we omit the signature and equational theory). An attacker for this protocol, following our *attacker schema* is presented in Fig. 11. We can then state that it is never the case that the attacker can know one of the keys used in the protocol. While some minor effort of representing an attacker is necessary, we can easily represent a generic attacker for a protocol by following a pre-determined schema.

We currently only consider finite protocols, modeled as processes in our calculus that use a communication channel c as their communication medium (written P_c). We have not pursued infinite protocols as of yet, but we believe it to possible to extend our approach to infinite protocols by defining the attacker as a recursive process with a parallel store (that is used to store the messages of the protocol). To analyze such a system, we would then employ recursive formulas by using the fixpoint operators of the logic.

Our attacker for finite protocols is defined as follows: For each output on c, the attacker performs an input on c (and stores the message). For each input on c, the attacker performs an attacker output.

Definition 4.1 (Attacker Generation Procedure) *Given a process P_c that models a finite protocol, the set S that tracks the attacker memory, an attacker for P can be generated by procedure* Attacker(P,S) *defined in Fig. 12 (x and m are fresh in P and the attacker).*

The generation procedure produces the necessary actions by inspection of the process dynamics: if an output can occur in the process, the attacker intercepts the message and memorizes it; if an input can occur in the process, the attacker injects any message it can produce from its knowledge; in the case where the protocol has no more actions, we represent the attacker memory with an output $m\langle x_1, \ldots, x_n\rangle$, modeling the attacker's memory throughout the protocol run. We thus show how an attacker can be extracted by inspection of the considered protocol. We can show that this attacker is general in our framework, in the sense that it can simulate the behavior expected from an adversarial medium (c.f. Dolev-Yao attacker). Note that this result does not yet fully apply to our tool implementation, as we discuss later in this section.

proc Attacker(P,S) $\equiv$
  if $P \xrightarrow{\alpha} Q \wedge \alpha =$ input on c then $c\langle * \rangle$.Attacker(Q,S) fi
  if $P \xrightarrow{\alpha} Q \wedge \alpha =$ output on c then $c(x)$.Attacker($Q, S \cup x$) fi
  if $P \xrightarrow{\alpha}$ then $m\langle x_1, \ldots, x_n \rangle$ where $x_i \in S$ fi.

Figure 12: Attacker Generation

Definition 4.2 (K-Set) *Given a process P, we define its K-Set K_P, the set of all terms known by P as:* $K_P \triangleq \{t \mid P \models \mathbb{K}\, t\}$

Theorem 4.3 (Monotonicity of K-Sets under Synchronization) *Let P_c and A be a processes such that $P_c \xrightarrow{c\langle M \rangle} P'_c$ and $A \xrightarrow{c(x)} A'$. We have that $K_{A'} \subseteq \mathfrak{F}(K_A \cup M)$.*

We begin with the K-Set of a process, the set of all terms known by the process that we observe in our logic, and we show that the evolution of arbitrary processes' K-Sets through synchronization is monotonic: the resulting process' knowledge will be a subset of the initial process' knowledge, plus any received messages. We state a similar property of our generated attacker's K-Set. Over time, the attacker's K-Set captures all messages exchanged in the protocol.

Theorem 4.4 (Monotonicity of Attacker Storage) *Let P_c and At be processes such that $At =$ Attacker($P_c, \{\}, c$), $P \xrightarrow{c\langle M \rangle} P'$ and $At \xrightarrow{c(x)} At'$. We have that $K_{At'} = \mathfrak{F}(K_{At} \cup M)$.*

Our Attacker Simulation (Lemma 4.5) and Process Knowledge (Lemma 4.6) lemmas provide some insight on the expressiveness of our attacker. Lemma 4.5 shows that a generated attacker, can obtain as much knowledge as an arbitrary process interacting with a finite protocol. Lemma 4.6 states a similar property, regarding the knowledge a finite protocol may obtain while interacting with our attacker.

Lemma 4.5 (Attacker Simulation) *Let P_c and A be processes.*
If $(\nu \overline{n})(P_c \mid A) \longrightarrow (\nu \overline{n})(P'_c \mid A')$ and $At =$ Attacker(P_c, S) with $K_A \subseteq K_{At}$ then $\exists At', S'$ such that $(\nu \overline{n})(P_c \mid At) \xrightarrow{*} (\nu \overline{n})(P'_c \mid At')$ and $At' =$ Attacker($P'_c, S \cup S'$) and $K_{A'} \subseteq K_{At'}$.

Lemma 4.6 (Process Knowledge) *Let P_c, A be processes and ϕ a knowledge formula.*
If $(\nu \overline{n})(P_c \mid A) \xrightarrow{*} (\nu \overline{n})(P'_c \mid A')$ and $P'_c \models \mathbb{K}\phi$ and $At =$ Attacker(P_c, S) with $K_A \subseteq K_{At}$ then $\exists At', S'$ such that $(\nu \overline{n})(P_c \mid At) \xrightarrow{*} (\nu \overline{n})(P'_c \mid At')$ and $P'_c \models \mathbb{K}\phi$ and $At' =$ Attacker($P'_c, S \cup S'$).

Furthermore, from Lemma 4.6 follows that, in our logic, a finite protocol interacting with an arbitrary process is indistinguishable from one interacting with our attacker. Combining these results, we can show that our attacker can behave as one would expect of an adversarial Dolev-Yao agent.

Theorem 4.7 (Preservation of Satisfaction) *Let P_c and A be processes and A any formula. If $(\nu \overline{n})(P_c \mid A) \xrightarrow{*} (\nu \overline{n})(P'_c \mid A')$ and $P'_c \models A$ and $At =$ Attacker(P_c, S) with $K_A \subseteq K_{At}$ then $\exists At', S'$ such that $(\nu \overline{n})(P_c \mid At) \xrightarrow{*} (\nu \overline{n})(P'_c \mid At')$ and $P'_c \models A$ and $At' =$ Attacker($P'_c, S \cup S'$).*

Notice that this result follows from the fact that message size for the attacker output prefix is unbounded. Our implementation currently bounds the generated message, to ensure tractability, and thus sacrificing completeness. However, as shown in [16], it is possible to compute a finite bound on the message size required to find an attack. The implementation of this result we leave for future work. It is anyway important to note that our method is already *sound and complete* for passive attackers, even for the case of non finite processes (eg. we may consider any finite control system, or bounded in the sense of [7]).

```
parameter attacker_depth = 2;

defproc A(k) = new N in c!(enc(N,k)).c?(x).[dec(x,k)=h(N)].end!(h(N));
defproc B(k) = c?(x).(begin!(dec(x,k)) | c!(enc(h(dec(x,k)),k)));
defproc Sys = new k in (A(k) | B(k));
defproc Attacker = c?(v).c!(*).s!(v);
defproc World = (Sys | Attacker);

defprop begin = <begin!> true;
defprop end = <end!> true;
defprop corrsp = always (end => begin);
check World |= corrsp;
Processing...
* Process World satisfies the formula corrsp *
```

Figure 13: Checking Correspondence in a Toy Protocol

```
...
defproc Sys = new k in (c!(k).(A(k) | B(k)));
defproc Attacker = c?(u).c?(v).c!(*).s!(v,u);
defproc World = (Sys | Attacker);
...
check World |= corrsp;
Processing...
* Process World does not satisfy the formula corrsp *
```

Figure 14: Checking Correspondence in a Broken Toy Protocol

4.2 Modeling Correspondence Assertions

Correspondence assertions are a technique for verifying authentication properties in protocols [18]. The idea is that the model of each principal in a protocol is refined with begin/end events, named *correspondence assertions*, at each stage of an authentication procedure. Authentication will be established if, for every run of the protocol, all end events for each stage are preceded by a matching begin event. To illustrate the idea, consider the following protocol:

$$\begin{array}{ll} A \to B : \{N\}_k; & B \text{ asserts the reception of } N \\ B \to A : \{h(N)\}_k; & A \text{ asserts the reception of } h(N) \end{array}$$

Principals A and B share a symmetric key k, N is a fresh value and h is a one-way hash function. When B receives $\{N\}_k$ it asserts the beginning of the run of the protocol. B sends message $\{h(N)\}_k$ so that A can verify the freshness of the run, by comparing the received value with its own hash of N. If the test succeeds, A asserts the reception of $h(N)$ and the end of the run. To check correspondence, one has to check that every run of the protocol, in the presence of an adversary, would be such that A's end assertion is always preceded by B's begin assertion, that is, A only ends if B was involved in the protocol.

Using our framework, we can model correspondence assertions by representing the assertion as an output on a channel that is irrelevant to the protocol, and then observing the existence of such outputs with our logic. For instance, our example could be modeled as done in Fig. 13 (note the `attacker_depth` parameter set to 2 due to the size of the second message). We can also successfully handle the case where we consider a faulty system that leaks k to the attacker (and thus correspondence does not hold), as presented in Fig. 14.

5 Concluding Remarks and Related Work

In this paper we have introduced a dynamic spatial epistemic logic for a variant of the applied π-calculus aimed at reasoning about security protocols. We explore the application of spatial and epistemic reasoning to the several agents involved in a security protocol, be they principals or adversaries. In our work, we can reason about the knowledge of the several agents of a protocol and how it can evolve over

time. Model-checking for the logic is shown to be decidable for an interesting class of processes and cryptographic primitives.

Our framework allows an interesting degree of freedom in the analyses it can perform, not only allowing one to reason directly about knowledge of principals and attackers but also enabling reasoning with correspondence assertions, which is an important addition to the range of available techniques. Moreover, our internalization of attackers, which does not require a complete behavioral specification, is able to accurately emulate the behavior of a Dolev-Yao attacker, enabling reasoning about the dynamics and knowledge of such an attacker.

Finally, the decidability result for our logic allowed us to implement a model-checking algorithm as a proof of concept extension to the SLMC tool. The main difference between the tool and the theory is that our attacker outputs are parametrized with a maximum message size, to bound the state space. This is the main limitation of the current version of our tool, since it does not yet fully capture the expressiveness of our attacker modeling, given that our results employ a more powerful version of the attacker output.

Overall, we have produced an interesting framework for protocol analysis, the first employing dynamic spatial logics. enabling a very natural (yet precise) way of reasoning about security protocols, all the while allowing reasoning with previously established techniques. Note, however, that our tool is merely a proof of concept of the developed framework, not aimed to compete with more mature tools for protocol analysis such as Avispa [4], Scyther [9], Casper [13] or ProVerif [5]. The main point of divergence of our approach and the ones mentioned before is that instead of mainly focusing on a set of built-in properties, we focus on a generic property language (our logic) and explore its expressiveness.

In terms of related logics, Kremer et al. [8] have proposed an epistemic logic for the applied π-calculus. However, their logic lacks the ability to reason about spatial properties, which is a key element in allowing reasoning about individual agents. Their epistemic modalities focus solely on attacker knowledge, not allowing one to state a property such as that of our introductory example where we care about the knowledge of the attacker but also of the agents within the system.

Another closely related logic is Datta et al.'s PCL [10]. PCL is a well established protocol analysis logic that allows one to verify properties of protocols modelled in a CCS style calculus by reasoning about events that occur in traces of the protocol run. While we focus on the combined reasoning about knowledge and spatial distribution of a protocol, PCL is designed to reason about the composition of several protocols and thus its analyses are more sophisticated than ours (reasoning about invariants in the protocol composition interleavings).

Mardare and Priami have also proposed a dynamic epistemic spatial logic [15] without the issues of security in mind. Their logic is hence substantially different from ours, interpreting knowledge as the possibility of observing actions of other processes and not as the ability to know some piece of information. Being based on CCS, such an approach is not suitable for reasoning about the flow of messages within a system, which is one of our main goals.

For future work we wish to further study the problem of attacker representations, aiming at an expressiveness result along the lines of Theorem 4.6 that does not require the attacker to be able to produce a message of an arbitrary size (this should follow from the result of [16]). This result will be key in removing the previously discussed limitation of our tool.

Acknowledgments The first author acknowledges support for this research provided by the Fundação para a Ciência e a Tecnologia through the Carnegie Mellon Portugal Program under Grant SFRH / BD / 33763 / 2009. We thank Mário Pires and Pedro Adão for their interesting comments on some version of this work. We also acknowledge the anonymous reviewers for their comments and suggestions.

References

[1] Martín Abadi & Véronique Cortier (2006): *Deciding knowledge in security protocols under equational theories*. *Theor. Comput. Sci.* 367(1), pp. 2–32.

[2] Martín Abadi & Cédric Fournet (2001): *Mobile values, new names, and secure communication*. In: *POPL '01: Proceedings of the 28th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, ACM, New York, NY, USA, pp. 104–115.

[3] Martín Abadi & Andrew D. Gordon (1997): *A calculus for cryptographic protocols: the spi calculus*. In: *CCS '97: Proceedings of the 4th ACM conference on Computer and communications security*, ACM, New York, NY, USA, pp. 36–47.

[4] A. Armando, David A. Basin, Y. Boichut, Y. Chevalier, L. Compagna, J. Cuéllar, P. H. Drielsma, P.-C. Héam, O. Kouchnarenko, J. Mantovani, S. Mödersheim, D. von Oheimb, M. Rusinowitch, J. Santiago, M. Turuani, L. Viganò & L. Vigneron (2005): *The AVISPA Tool for the Automated Validation of Internet Security Protocols and Applications*. In: *CAV*, pp. 281–285. Available at `http://dx.doi.org/10.1007/11513988_27`.

[5] Bruno Blanchet (2001): *An Efficient Cryptographic Protocol Verifier Based on Prolog Rules*. In: *14th IEEE Computer Security Foundations Workshop (CSFW-14)*, IEEE Computer Society, Cape Breton, Nova Scotia, Canada, pp. 82–96.

[6] Michael Burrows, Martin Abadi & Roger Needham (1990): *A logic of authentication*. *ACM Trans. Comput. Syst.* 8(1), pp. 18–36.

[7] Luis Caires (2004): *Behavioral and Spatial Observations in a Logic for the Pi-Calculus*. *FoSSaCS 2004* .

[8] Rohit Chadha, Stéphanie Delaune & Steve Kremer (2009): *Epistemic Logic for the Applied Pi Calculus*. In: D. Lee, A. Lopes & A. Poetzsch-Heffter, editors: *Proceedings of IFIP FMOODS/FORTE'09*, Lecture Notes in Computer Science, Springer, pp. 182–197.

[9] C.J.F. Cremers (2008): *The Scyther Tool: Verification, Falsification, and Analysis of Security Protocols*. In: *Computer Aided Verification, 20th International Conference, CAV 2008, Princeton, USA, Proc., Lecture Notes in Computer Science* 5123/2008, Springer, pp. 414–418.

[10] Anupam Datta, Ante Derek, John C. Mitchell & Arnab Roy (2007): *Protocol Composition Logic (PCL)*. *Theor. Comput. Sci.* 172(1-3), pp. 311–358.

[11] Dorothy E. Denning & Giovanni Maria Sacco (1981): *Timestamps in key distribution protocols*. *Commun. ACM* 24(8), pp. 533–536.

[12] Danny Dolev & Andrew C. Yao (1981): *On the security of public key protocols*. Technical Report, Stanford University, Stanford, CA, USA.

[13] Gavin Lowe (1998): *Casper: A Compiler for the Analysis of Security Protocols*. In: *Journal of Computer Security*, Society Press, pp. 53–84.

[14] Étienne Lozes & Jules Villard (2008): *A Spatial Equational Logic for the Applied* Π*-Calculus*. In: *CONCUR '08: Proceedings of the 19th international conference on Concurrency Theory*, Springer-Verlag, Berlin, Heidelberg, pp. 387–401.

[15] Radu Mardare & Corrado Priami (2006): *Dynamic Epistemic Spatial Logic*. Technical Report, Center for Computational and Systems Biology, University of Trento.

[16] Michaël Rusinowitch & Mathieu Turuani (2003): *Protocol insecurity with a finite number of sessions and composed keys is NP-complete*. *Theor. Comput. Sci.* 299(1-3), pp. 451–475.

[17] Bernardo Toninho & Luis Caires (2009): *A Spatial-Epistemic Logic and Tool for Reasoning about Security Protocols*. Technical Report, Departamento de Informática, FCT/UNL.

[18] Thomas Y. C. Woo & Simon S. Lam (1993): *A Semantic Model for Authentication Protocols*. In: *SP '93: Proceedings of the 1993 IEEE Symposium on Security and Privacy*, IEEE Computer Society, Washington, DC, USA, p. 178.

On the Decidability of Non Interference over Unbounded Petri Nets

Eike Best
Universität Oldenburg, 26111 Oldenburg, Germany
eike.best@informatik.uni-oldenburg.de

Philippe Darondeau
Inria Rennes - Bretagne Atlantique, Rennes, France
Philippe.Darondeau@inria.fr

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

Non-interference, in transitive or intransitive form, is defined here over unbounded (Place/Transition) Petri nets. The definitions are adaptations of similar, well-accepted definitions introduced earlier in the framework of labelled transition systems [4, 5, 8]. The interpretation of intransitive non-interference which we propose for Petri nets is as follows. A Petri net represents the composition of a controlled and a controller systems, possibly sharing places and transitions. Low transitions represent local actions of the controlled system, high transitions represent local decisions of the controller, and downgrading transitions represent synchronized actions of both components. Intransitive non-interference means the impossibility for the controlled system to follow any local strategy that would force or dodge synchronized actions depending upon the decisions taken by the controller after the last synchronized action. The fact that both language equivalence and bisimulation equivalence are undecidable for unbounded labelled Petri nets might be seen as an indication that non-interference properties based on these equivalences cannot be decided. We prove the opposite, providing results of decidability of non-interference over a representative class of infinite state systems.

1 Introduction

Non-interference has been defined in the literature as an extensional property based on some observational semantics: the high part H (i.e., the secret part) of a system does not interfere with the low part L (i.e., the public part) if whatever is done in H produces *no visible effect* on L. The original notion of non-interference in [6] was defined, using language equivalence, for deterministic automata with outputs. Generalized notions of non-interference were then designed to include (nondeterministic) labelled transition systems and finer notions of observational semantics such as bisimulation (see, e.g., [4,5,13,19–21]). Recently, the problem of defining suitable non-interference properties has been attacked also in the classical model of elementary Petri nets, a special class of Petri nets where places can contain at most one token [1,2]. When it is necessary to declassify information (e.g., when a secret plan has to be made public for realization), the two-level approach (secret/public – H/L) is usually extended with one intermediate level of downgrading (D), so that the high actions that have been performed prior to a declassifying action are made public by this declassifying action. This security policy is known under the name of *intransitive* noninterference [18] (*INI* for short) because the information flow relation is considered not transitive: even if information flows from H to D and from D to L are allowed, direct flows from H to L are forbidden. In [8] intransitive non-interference has been defined for elementary net systems.

The technical goal of this paper is to show the decidability of intransitive non-interference in the extended framework of unbounded (Place/Transition) Petri nets, and this for both definitions based al-

K. Chatzikokolakis and V. Cortier (Ed.): 8th International Workshop
on Security Issues in Concurrency (SecCo 2010)
EPTCS 51, 2011, pp. 16–33, doi:10.4204/EPTCS.51.2

ternatively on language equivalence or on weak bisimulation equivalence. As both equivalences are undecidable for unbounded labelled Petri nets [9] [11], the decidability of intransitive non-interference is not a trivial result. This is however not the first result of this type for infinite-state systems. It was actually shown in [3] that Strong Low Bisimulation and Strong Security which is based on the latter equivalence can be decided for *Parallel While Programs* defined over expressions from decidable first order theories. Decidability is also established in [3] for Strong Dynamic Security that takes both downgrading and upgrading into account. In that work, decidability comes for a large part from the property of Strong Low Bisimulation to envisage implicitly through its recursive definition all possible modifications of the dynamic store by a concurrent context (without any effective definition). In our work, decidability comes also for a large part from the fact that our basic security properties are *NDC* (*NonDeducibility on Composition*) and its bisimulation version *BNDC* [4, 5], hence we envisage implicitly arbitrary concurrent contexts defined by Petri nets with high-level transitions. Now, the results presented in [3] concern language based security whereas our results concern discrete event systems security. As a matter of fact, both settings do not compare: on the one hand, owing to the impossibility of testing places for zero, unbounded Place/Transition nets have less computing power than Parallel Write Programs, but on the other hand they have *labeled* transition semantics whereas Parallel Write Programs have *unlabeled* transition semantics.

Let us now explain the meaning of non-interference in the context of systems and control. In the Ramadge and Wonham approach to supervisory control for safety properties of discrete event systems [16, 17], one considers closed loop systems made of a plant (the system under control) and a controller that may share actions but have disjoint sets of local states. Synchronization on shared actions allows the controller to observe the plant and to disable selected actions of the plant. Actions of the plant may be invisible to the controller, but all actions of the controller are shared with the plant and synchronized. Moreover controllers are deterministic, hence the current state of the controller may be inferred from the past behaviour of the plant. In the present paper, the closed system made of the plant and the controller is modelled by an unbounded Petri net with three levels of transitions L, D and H. A place may count e.g. an unbounded number of clients or goods. Transitions in L represent actions of the plant alone. Transitions in D represent synchronized actions of the plant and the controller. Transitions in H represent actions of the controller alone. Here the controller can check and modify proactively the global state to orient runs towards reaching some set of states or to maximize some profit. Intransitive non-interference means the impossibility for the controlled system, seen as the adversary of the controller, to win by forcing or dodging synchronized actions that depend upon the decisions taken by the controller after the last synchronized action. An example is given in Section 4.

We are mainly interested in intransitive non-interference. Nevertheless, in a large part of the paper, we shall focus on classical non-interference, in order to establish first the technical results in a simpler framework. In Section 2 we recall the basics of labeled transition systems and Petri nets. Section 3 presents the definitions of classical non-interference notions for PT-nets, and proves that both language equivalence and weak bisimulation equivalence based notions of classical non-interference are decidable. Section 4 presents the definition of intransitive non-interference for PT-nets, introduces examples showing the practical significance of this notion in the context of discrete event systems, and provides decidability results extending the results of Section 3. Section 5 reports some conclusive remarks. A short appendix recalls some results on Petri nets and semi-linear sets used in our proofs.

2 Background

2.1 Transition systems and bisimulations

Definition 2.1 (LTS)**.** *A* labeled transition system *over a set of* labels Σ *is a tuple* $\mathscr{T} = (Q, T, q_0)$ *where* Q *is a set of* states, $q_0 \in Q$ *is the initial state, and* $T \subseteq Q \times \Sigma \times Q$ *is a set of* labeled transitions. *An LTS is said to be* deterministic *if* $(q, \sigma, q') \in T$ *and* $(q, \sigma, q'') \in T$ *entail* $q' = q''$.

Definition 2.2 (LTS under partial observation)**.** *A partially observed LTS is an LTS* $\mathscr{T} = (Q, T, q_0)$ *over a set of labels* Σ *which is partitioned into* observable *labels* $\sigma \in \Sigma_o$ *(for convenience, we assume that* $\varepsilon \notin \Sigma_o$*) and* unobservable *labels* $\tau \in \Sigma_{uo}$. *In a partially observed LTS,* $q \to^* q'$ *denotes the least binary relation on states such that* $q \to^* q$ *for all* $q \in Q$, $q \to^* q'$ *for all* $(q, \tau, q') \in T$ *with* $\tau \in \Sigma_{uo}$, *and* $q \to^* q'$ *whenever* $q \to^* q''$ *and* $q'' \to^* q'$ *for some* q''.

Definition 2.3 (Language equivalence)**.** *The* language *of a partially observed LTS is the set of all finite words* $\sigma_1\sigma_2\ldots\sigma_n$ *(including* ε *which corresponds to* $n = 0$*) such that* $q_0 \to^* q_1 \xrightarrow{\sigma_1} q_1' \to^* q_2 \xrightarrow{\sigma_2} q_2' \ldots \to^* q_n \xrightarrow{\sigma_n} q_n'$ *for some adequate sequence of states* q_i *and* q_i'. *Two partially observed LTS's* $\mathscr{T}$ *and* $\mathscr{T}'$ *are* language equivalent *(in notation,* $\mathscr{T} \sim \mathscr{T}'$*) if they have the same language.*

Definition 2.4 (Weak simulation)**.** *Given a set of labels* $\Sigma = \Sigma_o \cup \Sigma_{uo}$ *and two partially observed LTS's* $\mathscr{T}$ *and* $\mathscr{T}'$ *over* Σ, $\mathscr{T}$ *is* weakly simulated *by* $\mathscr{T}'$ *(or* $\mathscr{T}'$ *weakly simulates* $\mathscr{T}$*) if there exists a binary relation* $R \subseteq Q \times Q'$, *called a* weak simulation, *such that* $(q_0, q_0') \in R$ *and the following requirements are satisfied for all* $(q_1, q_1') \in R$, *and for all* $\sigma \in \Sigma_o$ *and* $\tau \in \Sigma_{uo}$*:*

- *if* $q_1 \xrightarrow{\sigma} q_2$ *then* $(\exists q_2') : (q_2, q_2') \in R$ *and* $q_1' \to^* q_1'' \xrightarrow{\sigma} q_2'' \to^* q_2'$,
- *if* $(q_1, \tau, q_2) \in T$ *then* $(\exists q_2') : (q_2, q_2') \in R$ *and* $q_1' \to^* q_2'$.

If $\mathscr{T}$ is simulated by $\mathscr{T}'$, then the language of $\mathscr{T}$ is included in the language of $\mathscr{T}'$.

Definition 2.5 (Weak bisimilarity)**.** *Given a set of labels* $\Sigma = \Sigma_o \cup \Sigma_{uo}$, *two partially observed LTS's* $\mathscr{T} = (Q, T, q_0)$ *and* $\mathscr{T}' = (Q', T', q_0')$ *over* Σ *are* weakly bisimilar *(in notation,* $\mathscr{T} \approx \mathscr{T}'$*) if and only if there exists some binary relation* $R \subseteq Q \times Q'$, *called a* weak bisimulation, *such that* $(q_0, q_0') \in R$ *and both* R *and* R^{-1} *are weak simulations.*

If $\mathscr{T}$ and $\mathscr{T}'$ are weakly bisimilar, then they are language equivalent.

2.2 Place/Transition Petri nets

In order to keep the presentation concise, we omit here the basic definition of Petri nets which may be found in an appendix together with some classical decidability results.

Definition 2.6 (PT-net system)**.** *A* PT-net system $\mathscr{N} = (P, T, F, M_0)$ *is a PT-net with an* initial marking M_0. *The* reachability set $RS(\mathscr{N})$ *of* $\mathscr{N}$ *is the set of all markings that may be reached from* M_0 *by sequences of transitions of the net. The* reachability graph $RG(\mathscr{N})$ *of* $\mathscr{N}$ *is the LTS with the set of states* $[M_0\rangle$ *and the initial state* M_0, *where* $[M_0\rangle = RS(\mathscr{N})$ *and there is a transition from* M *to* M' *labeled with* t *iff* $M[t\rangle M'$. *Given* $\mathscr{N} = (P, T, F, M_0)$, *the* underlying net *is* $\mathscr{U}(\mathscr{N}) = (P, T, F)$. *For convenience, we write* $\mathscr{N} = (\mathscr{U}(\mathscr{N}), M_0)$.

Definition 2.7 (Composition of net systems)**.** *Given two PT-net systems* $\mathscr{N}_1 = (P_1, T_1, F_1, M_{1,0})$ *and* $\mathscr{N}_2 = (P_2, T_2, F_2, M_{2,0})$ *such that* $P_1 \cap P_2 = \emptyset$, *their composition* $\mathscr{N}_1 \,|\, \mathscr{N}_2$ *is the PT-net system* (P, T, F, M_0) *where* P *is the union of* P_1 *and* P_2, T *is the union of* T_1 *and* T_2, *and* F *and* M_0 *are the unions of the maps* F_i *and* $M_{i,0}$ *respectively, for* $i = 1, 2$. *Also let* $\mathscr{U}(\mathscr{N}_1) \,|\, \mathscr{U}(\mathscr{N}_2) = \mathscr{U}(\mathscr{N}_1 \,|\, \mathscr{N}_2)$.

Note that synchronisation occurs over those transitions that are shared by the two nets, that is, for a transition t that occurs both in T_1 and T_2, we have that, e.g., $F(p,t) = F_1(p,t)$ if $p \in P_1$, $F(p,t) = F_2(p,t)$ otherwise.

Definition 2.8 (Restriction of a net system)**.** *Given a PT-net system $\mathscr{N} = (P,T,F,M_0)$ and a subset of transitions $T' \subseteq T$, let $\mathscr{N} \setminus T' = (P, T \setminus T', F', M_0)$ where F' is the induced restriction of F on $T \setminus T'$. Also let $\mathscr{U}(\mathscr{N}) \setminus T' = (P, T \setminus T', F')$.*

Definition 2.9 (Labeled net system)**.** *A* labeled net system $(\mathscr{N}, \lambda)$ *is a PT-net system $\mathscr{N} = (P,T,F,M_0)$ with a* transition labelling map $\lambda : T \to \Sigma_o \cup \{\varepsilon\}$ *(the subscript o in Σ_o means an alphabet of observations). The* labeled reachability graph *of $(\mathscr{N}, \lambda)$ is the partially observed LTS over $\Sigma = \Sigma_o \cup \{\varepsilon\}$ which derives from $RG(\mathscr{N})$ by replacing each transition $M[t\rangle M'$ with a corresponding transition $(M, \lambda(t), M')$.*

Definition 2.10 (Weak simulation)**.** *Given two labeled net systems $(\mathscr{N}, \lambda)$ and $(\mathscr{N}', \lambda')$ over the same set of labels Σ_o, $(\mathscr{N}, \lambda)$ is weakly simulated by $(\mathscr{N}', \lambda')$ if the labeled reachability graph of $\mathscr{N}$ is weakly simulated by the labeled reachability graph of $\mathscr{N}'$.*

Definition 2.11 (Equivalences of labeled net systems)**.** *Two labeled net systems $(\mathscr{N}, \lambda)$ and $(\mathscr{N}', \lambda')$ over the same set of labels Σ_o are:*

- language equivalent *(in notation, $(\mathscr{N}, \lambda) \sim (\mathscr{N}', \lambda')$ or for short $\mathscr{N} \sim \mathscr{N}'$ when the labelling maps are clear from the context) if their labeled reachability graphs are language equivalent;*
- weakly bisimilar *(in notation, $(\mathscr{N}, \lambda) \approx (\mathscr{N}', \lambda')$ or for short $\mathscr{N} \approx \mathscr{N}'$ when the labelling maps are clear from the context) if their labeled reachability graphs are weakly bisimilar.*

A weak bisimulation between the labeled reachability graphs of two labeled net systems is called a weak bisimulation between them.

A particular case is with *partially observed net systems*, i.e. when $\Sigma_o = T_o \subseteq T$, $\lambda(t) = t$ for $t \in T_o$, and $\lambda(t) = \varepsilon$ for $t \in T \setminus T_o$. For partially observed net systems, $(\mathscr{N}, \lambda) \sim (\mathscr{N}', \lambda')$ if and only if the reachability graphs of $\mathscr{N}$ and $\mathscr{N}'$, considered as partially observed LTS's with $\Sigma_{uo} = T \setminus T_o$, are language equivalent in the sense of Definition 2.3. In the same conditions, $(\mathscr{N}, \lambda) \approx (\mathscr{N}', \lambda')$ if and only if $RG(\mathscr{N}) \approx RG(\mathscr{N}')$ in the sense of Definition 2.5.

Proposition 2.12. *If λ is the identity, $(\mathscr{N}, \lambda) \approx (\mathscr{N}', \lambda)$ iff $(\mathscr{N}, \lambda) \sim (\mathscr{N}', \lambda)$*

3 Classical non-interference in PT-nets

In this section, we focus on systems that can perform two kinds of actions: high-level actions, representing the interaction of the system with high-level users, and low-level actions, representing the interaction of the system with low-level users. The system has the property of non-interference if the interplay between its low-level part and high-level part cannot affect the low level user's view of the system, even assuming that the low-level user knows the structure of the system. As already said in the introduction, the goal of this section is to provide the technical basis that we need for showing subsequently the decidability of intransitive non-interference for PT-nets, which we feel has more direct interest for applications in the context of discrete event systems. We must therefore postpone the presentation of motivating examples.

Definition 3.1 (Two-level net system)**.** *A two-level PT-net system is a PT-net system $\mathscr{N} = (P,T,F,M_0)$ whose set of transitions T is partitioned into* low level *transitions $l \in L$ and* high level *transitions $h \in H$, such that $T = L \cup H$ and $L \cap H = \emptyset$. A net system $\mathscr{N}$ is a* high-level net system *if all transitions in T are high-level transitions. It is a* low-level net system *if all transitions in T are low-level transitions.*

Henceforth, *two-level net systems are considered as partially observed net systems* where the transitions in L are observable while the transitions in H are unobservable ($\Sigma_o = L$ and $\Sigma_{uo} = H$). This interpretation applies to all instances of the relations $\mathscr{N} \sim \mathscr{N}'$ or $\mathscr{N} \approx \mathscr{N}'$ between two-level net systems. We denote by $\mathscr{L}(\mathscr{N})$ the language of a two-level net system $\mathscr{N}$, that is to say, the set of images $\lambda(t_1 t_2 \dots t_n)$ of sequences of transitions $M_0[t_1 t_2 \dots t_n\rangle M$ under the labelling map $\lambda(t) = t$ for $t \in L$ and $\lambda(t) = \varepsilon$ for $t \in H$.

Definition 3.2 (NDC-BNDC)**.** *A* two-level *net system $\mathscr{N}$ has the property NDC (Non-Deducibility on Compositions), resp. BNDC (Bisimulation-Based Non-Deducibility on Compositions), if for any high-level net system $\mathscr{N}'$ with a set of transitions H' not intersecting L, the two-level net systems $\mathscr{N} \setminus H$ and $(\mathscr{N} | \mathscr{N}') \setminus (H \setminus H')$ are language equivalent, resp. weakly bisimilar.*

The definitions of NDC and BNDC are very strong, and their verification is indeed quite demanding: infinitely many equivalence checks are required, one for each choice of a high-level net system $\mathscr{N}'$. Moreover, each equivalence check may be a problem, as both language equivalence and bisimulation equivalence are undecidable over unbounded labeled PT-nets and likewise over unbounded partially observed PT-nets [9, 11]. We shall discuss about the strength of these notions in section 4. For the moment, what we need is an alternative characterization of these properties, more amenable for an algorithmic treatment in view of showing decidability.

3.1 Deciding on NDC

In this section, we show that $\mathscr{N}$ enjoys NDC if and only if $\mathscr{N}$ and $\mathscr{N} \setminus H$ are language equivalent.

Proposition 3.3. *For any high-level net system $\mathscr{N}'$ with set of transitions H' not intersecting L, $\mathscr{N} \setminus H$ is weakly simulated by $(\mathscr{N} | \mathscr{N}') \setminus (H \setminus H')$ which in turn is weakly simulated by $\mathscr{N}$ (where all net systems under consideration have the same set of observable transitions $\Sigma_o = L$).*

Proof. Any transition from L has similar place neighbourhoods in $\mathscr{N} \setminus H$, $(\mathscr{N} | \mathscr{N}') \setminus (H \setminus H')$ and $\mathscr{N}$, and the transitions from L and H' have disjoint place neighbourhoods in $(\mathscr{N} | \mathscr{N}') \setminus (H \setminus H')$. □

Proposition 3.4. *$\mathscr{N}$ has the property NDC iff $\mathscr{N} \sim \mathscr{N} \setminus H$. Moreover, this property can be decided.*

Proof. By definition, $\mathscr{N}$ has the property NDC iff, for any high-level net system $\mathscr{N}'$ with a set of transitions H' not intersecting L, the two-level net systems $\mathscr{N} \setminus H$ and $(\mathscr{N} | \mathscr{N}') \setminus (H \setminus H')$ are language equivalent. Now, the chain of inclusion relations $\mathscr{L}(\mathscr{N} \setminus H) \subseteq \mathscr{L}((\mathscr{N} | \mathscr{N}') \setminus (H \setminus H')) \subseteq \mathscr{L}(\mathscr{N})$ holds for Proposition 3.3. Both bounds are reached for some net system $\mathscr{N}'$; indeed, the lower bound is reached when $\mathscr{N}'$ has no place and $H' = \emptyset$, and the upper bound is reached when $\mathscr{N}'$ has no place and $H' = H$. Suppose $\mathscr{N}$ has the property NDC, then $\mathscr{L}(\mathscr{N} \setminus H) = \mathscr{L}((\mathscr{N} | \mathscr{N}') \setminus (H \setminus H')) = \mathscr{L}(\mathscr{N})$ for $\mathscr{N}'$ realizing the upper bound. Conversely, suppose that $\mathscr{L}(\mathscr{N} \setminus H) = \mathscr{L}(\mathscr{N})$, then necessarily $\mathscr{L}(\mathscr{N} \setminus H) = \mathscr{L}((\mathscr{N} | \mathscr{N}') \setminus (H \setminus H'))$. Hence, the first claim in the proposition has been established. As all transitions are observable in the net system $\mathscr{N} \setminus H$, the language $\mathscr{L}(\mathscr{N} \setminus H)$ is a free Petri net language. By E. Pelz's theorem and corollary (Theorem 6.4 in the appendix), one can decide whether $\mathscr{L}(\mathscr{N}) \subseteq \mathscr{L}(\mathscr{N} \setminus H)$, and hence whether the two languages are equal. □

Example 3.5. *The net system $\mathscr{N}_1$ of Figure 1(a) is insecure, as $\mathscr{N}_1$ can perform the low transition l at some stage, while $\mathscr{N}_1 \setminus H$ cannot. On the contrary, the net system $\mathscr{N}_2$ in Figure 1(b) enjoys NDC.*

Figure 1: Two simple two-level net systems

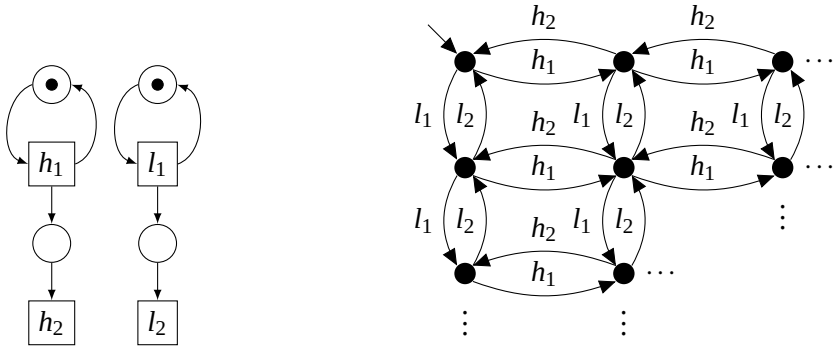


Figure 2: An infinite-state net system (l.h.s.) and its labeled reachability graph (r.h.s.)

Example 3.6. *Consider the disconnected net system $\mathcal{N}$ in Figure 2 (l.h.s.). Intuitively, we expect that this system is secure because the high part of the net (the left part) and the low part of the net (the right part) are disconnected and so it appears that no interference is possible. In view of Definition 3.2, it seems however difficult to verify this property by direct inspection of the infinite labeled reachability graph of $\mathcal{N}$ shown in Figure 2 (r.h.s.). With the help of Proposition 3.4, this verification becomes straightforward: the transition system that generates the language $\mathcal{L}(\mathcal{N} \setminus H)$, which corresponds with the left column of the picture, and the deterministic transition system that generates the language $\mathcal{L}(\mathcal{N})$ (obtained by replacing all labels h_i by ε and then applying the usual subset construction) are indeed identical.*

3.2 Reducing BNDC to SBNDC

For BNDC, things are a bit more complex, although we have the following property.

Lemma 3.7. *If $\mathcal{N}$ has the property BNDC, then $\mathcal{N} \approx \mathcal{N} \setminus H$.*

Proof. Let $\mathcal{N}'$ be the high-level net system with no place and with the set of transitions $H' = H$, then the reachability graphs of $\mathcal{N}$ and $(\mathcal{N} \,|\, \mathcal{N}') \setminus (H \setminus H')$ are isomorphic, hence they are weakly bisimilar, that is $\mathcal{N} \approx (\mathcal{N} \,|\, \mathcal{N}') \setminus (H \setminus H')$. If $\mathcal{N}$ has the property BNDC, then $\mathcal{N} \setminus H \approx (\mathcal{N} \,|\, \mathcal{N}') \setminus (H \setminus H')$, and the lemma follows since $\approx$ is an equivalence. □

Example 3.8. *Consider the net system $\mathcal{N}$ in Figure 3. $\mathcal{N}$ is NDC because $\mathcal{N} \sim \mathcal{N} \setminus H$. However, $\mathcal{N}$ is not BNDC because $\mathcal{N} \not\approx \mathcal{N} \setminus H$. Indeed, this net is insecure: a low-level user who is unable to perform transition l can deduce from this failure that the high-level transition h has been performed.*

In the rest of the section, we show that $\mathcal{N}$ enjoys BNDC if and only if it enjoys the property SBNDC defined below.

Definition 3.9 (SBNDC)**.** *A two-level net system $\mathcal{N}$ has the property SBNDC (Bisimulation-Based Strong Non-Deducibility on Compositions) if, for any reachable marking M_1 of $\mathcal{N} = (N, M_0)$ and for any high-level transition $h \in H$, $M_1[h\rangle M_2$ entails that $(N \setminus H, M_1)$ and $(N \setminus H, M_2)$ are weakly bisimilar.*

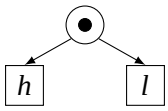


Figure 3: A simple two-level net system

Note that, in view of Proposition 2.12, the relation between M_1 and M_2 required in Definition 3.9 may be equivalently expressed as $\mathscr{L}(\mathscr{N} \setminus H, M_1) = \mathscr{L}(\mathscr{N} \setminus H, M_2)$.

Definition 3.10. *Let $R \subseteq RS(\mathscr{N} \setminus H) \times RS(\mathscr{N})$ be the binary relation on markings which is generated from the axiom $M_0 R M_0$ by the following two inference rules where $h \in H$ and $l \in L$:*

- *$M_1 R M_2$ and $M_1 = M_1'$ and $M_2[h\rangle M_2'$ entail $M_1' R M_2'$*
- *$M_1 R M_2$ and $M_1[l\rangle M_1'$ and $M_2[l\rangle M_2'$ entail $M_1' R M_2'$*

Paraphrasing the definition, MRM' if and only if there exist $w \in L^*$ and $w' \in (L \cup H)^*$ such that $M_0[w\rangle M$, $M_0[w'\rangle M'$, and w is the projection of w' on L^*. In the specific case where $\mathscr{N}$ is BNDC, R is a weak bisimulation between $\mathscr{N} \setminus H$ and $\mathscr{N}$, and it is indeed the *least* weak bisimulation between them.

Lemma 3.11. *Let $\mathscr{N} = (N, M_0)$ be a net system with the BNDC property and let M_1 and M_2 be reachable markings of $\mathscr{N} \setminus H$ and $\mathscr{N}$, respectively. If $M_1 R M_2$, then $\mathscr{L}(N \setminus H, M_1) = \mathscr{L}(N \setminus H, M_2)$.*

Proof. As $M_1 R M_2$, there exist $w \in L^*$ and $w' \in (L \cup H)^*$ such that $M_0[w\rangle M_1$, $M_0[w'\rangle M_2$, and w is the projection of w' on L^*. Let $k = |w'| - |w|$ be the difference of length between w' and w. Consider the high-level net system $\mathscr{K} = (K, M_k)$ where K is a net with a unique place p_k, the set of transitions H, and flow relations $F(p_k, h) = 1$ and $F(h, p_k) = 0$ for every transition h, and where $M_k(p_k) = k$. Let M_0' and M_2' be the markings of $N' = \mathscr{U}(\mathscr{N} | \mathscr{K})$, extending M_0 and M_2, respectively, such that $M_0'(p_k) = k$ and $M_2'(p_k) = 0$. By construction, $M_0'[w'\rangle M_2'$ in $\mathscr{N} | \mathscr{K}$. As $\mathscr{N}$ has the property BNDC and $\mathscr{K}$ is a high-level net system, $\mathscr{N} \setminus H \approx (\mathscr{N} | \mathscr{K}) \setminus (H \setminus H) = \mathscr{N} | \mathscr{K}$. As all transitions of $\mathscr{N} \setminus H$ are observable and w is the observable projection of w', M_1 and M_2' are two weakly bisimilar markings of $\mathscr{N} \setminus H$ and $\mathscr{N} | \mathscr{K}$, hence $\mathscr{L}(N \setminus H, M_1) = \mathscr{L}(N | K, M_2')$. As $M_2'(p_k) = 0$, no transition in H can occur in any sequence fired from M_2' in $N | K$, and therefore $\mathscr{L}(N | K, M_2') = \mathscr{L}(N \setminus H, M_2)$. Altogether, $\mathscr{L}(N \setminus H, M_1) = \mathscr{L}(N \setminus H, M_2)$. □

Proposition 3.12. *$\mathscr{N} = (N, M_0)$ has the property BNDC* iff *for all reachable markings M_1 and M_2 of $N \setminus H$ and N, respectively, $M_1 R M_2$ entails $\mathscr{L}(N \setminus H, M_1) = \mathscr{L}(N \setminus H, M_2)$.*

Proof. The direct implication has already been established. To show the converse implication, consider any high-level net system $\mathscr{N}'$ with set of transitions H' not intersecting L. Let B be the relation between the reachable markings of $\mathscr{N} \setminus H$ and $(\mathscr{N} | \mathscr{N}') \setminus (H \setminus H')$ defined as follows. Let $(M_2 | M_2')$ denote the marking of $(\mathscr{N} | \mathscr{N}')$ that projects on the markings M_2 and M_2' of $\mathscr{N}$ and $\mathscr{N}'$, respectively. Then, let $M_1 B (M_2 | M_2')$ iff $M_1 R M_2$. Assume that $M_1 R M_2$ entails $\mathscr{L}(N \setminus H, M_1) = \mathscr{L}(N \setminus H, M_2)$. We will show that B is a weak bisimulation between $\mathscr{N} \setminus H$ and $(\mathscr{N} | \mathscr{N}') \setminus (H \setminus H')$, entailing that $\mathscr{N}$ has the property BNDC. As $M_1 R M_2$ for $M_1 = M_0$ and $M_2 = M_0$, the relation B holds between the initial states of the two net systems. Now consider any occurrence $M_1 B (M_2 | M_2')$ of the relation B, hence $M_1 R M_2$ (by construction of B).

- Let $M_1[l\rangle \widetilde{M_1}$ for $l \in L$. As $M_1 R M_2$ entails $\mathscr{L}(N \setminus H, M_1) = \mathscr{L}(N \setminus H, M_2)$, necessarily, $M_2[l\rangle \widetilde{M_2}$ for some marking $\widetilde{M_2}$, and then by definition of R, $\widetilde{M_1} R \widetilde{M_2}$. Thus, $(M_2 | M_2')[l\rangle (\widetilde{M_2} | M_2')$ with $\widetilde{M_1} B (\widetilde{M_2} | M_2')$.

- Let $(M_2|M_2')[l\rangle(\widetilde{M_2}|M_2')$ for $l \in L$. As M_1RM_2 entails $\mathscr{L}(N \setminus H, M_1) = \mathscr{L}(N \setminus H, M_2)$, necessarily $M_1[l\rangle\widetilde{M_1}$ for some marking $\widetilde{M_1}$ such that $\widetilde{M_1}R\widetilde{M_2}$, hence $M_1B(\widetilde{M_2}|M_2')$ by definition of B.
- Let $(M_2|M_2')[h\rangle(\widetilde{M_2}|M_2'')$ for $h \in H$, then certainly $M_2[h\rangle\widetilde{M_2}$ in $\mathscr{N}$. Suppose $M_1[h\rangle M_2$, then we have also $M_1R\widetilde{M_2}$ by definition of R, hence $M_1B(\widetilde{M_2}|M_2'')$ by definition of B.

Summing up, B is a weak bisimulation and $\mathscr{N}$ has the property BNDC. □

Proposition 3.13. *$\mathscr{N}$ has the property SBNDC iff for any reachable marking M_1 of $\mathscr{N} = (N, M_0)$ and for any high-level transition $h \in H$, $M_1[h\rangle M_2$ entails that $\mathscr{L}(\mathscr{N} \setminus H, M_1) = \mathscr{L}(\mathscr{N} \setminus H, M_2)$.*

Proof. As for $\mathscr{N} \setminus H$ the labelling is the identity $\lambda(l) = l$, the thesis follows by Proposition 2.12. □

Theorem 3.14. *$\mathscr{N}$ has the property BNDC* iff *it has the property SBNDC.*

Proof. Suppose that $\mathscr{N}$ has the property BNDC. Then , by Lemma 3.7, $\mathscr{N} \approx \mathscr{N} \setminus H$, hence $\mathscr{L}(\mathscr{N}) = \mathscr{L}(\mathscr{N} \setminus H)$. Let $M_0[s\rangle M_1$ in $\mathscr{N}$, then necessarily, $M_0[s'\rangle M_1'$ in $\mathscr{N} \setminus H$ for s' defined as the observable projection of s. Thus $M_1'RM_1$ by definition of R. As $M_1[h\rangle M_2$, we have also $M_1'RM_2$. By Proposition 3.12, $\mathscr{L}(\mathscr{N} \setminus H, M_1) = \mathscr{L}(\mathscr{N} \setminus H, M_1') = \mathscr{L}(\mathscr{N} \setminus H, M_2)$, hence $\mathscr{N}$ has the property SBNDC.

Now assume that $\mathscr{N}$ has the property SBNDC. By Proposition 3.12, in order to prove that $\mathscr{N}$ has the property BNDC, it suffices to show that M_1RM_2 entails $\mathscr{L}(\mathscr{N} \setminus H, M_1) = \mathscr{L}(\mathscr{N} \setminus H, M_2)$ for all reachable markings M_1 and M_2 of $\mathscr{N} \setminus H$ and $\mathscr{N}$, respectively. Let M_1 and M_2 be two such markings and assume that M_1RM_2. In view of Definition 3.10, this relation has been derived from the axiom MRM using the two inference rules (where we have exchanged the M_i and the M_i' from Definition 3.10):

- $M_1'RM_2'$ and $M_1' = M_1$ and $M_2'[h\rangle M_2$ entail M_1RM_2
- $M_1'RM_2'$ and $M_1'[l\rangle M_1$ and $M_2'[l\rangle M_2$ entail M_1RM_2

If $M_1 = M_2$, then there is nothing to prove. In the converse case, one can assume by induction on the derivation of M_1RM_2 that $\mathscr{L}(\mathscr{N} \setminus H, M_1') = \mathscr{L}(\mathscr{N} \setminus H, M_2')$. The desired conclusion follows then from Definition 3.9 for the first rule, and from the definition of R and the injective labelling of nets for the second rule. □

Despite the fact that SBNDC requires infinitely many equivalence checks, one for each reachable marking enabling a high-level transition, it (and hence also BNDC) can be decided, as will be seen in the next section.

3.3 Deciding SBNDC

In this section, we reduce SBNDC to the conjunction, for all high-level transitions h and for all low-level transitions l, of a predicate $P(h,l)$ meaning that the enabling or disabling of l in the net after a sequence of low transitions $s \in L^*$ gives no indication on whether h has been fired immediately before s.

Definition 3.15. *Given a two-level net system $\mathscr{N}$ and two transitions $h \in H$ and $l \in L$, we say that $P(h,l)$ holds* iff *for any words $s \in L^*$ and $w \in (L \cup H)^*$, if $M_0[w\rangle M_1$, $M_1[h\rangle M_2$, $M_1[s\rangle M_3$, and $M_2[s\rangle M_4$, then $M_3[l\rangle$ iff $M_4[l\rangle$.*

Figure 4 shows a situation where $P(h,l)$ is *not* satisfied, because l is enabled at M_4 but not at M_3. This corresponds roughly to *causal information flow* [2] from h to l. The other situation in which $P(h,l)$ is not satisfied is the symmetric one, when l is enabled at M_3 but disabled at M_4; this roughly corresponds to *conflict information flow* [2] from h to l.

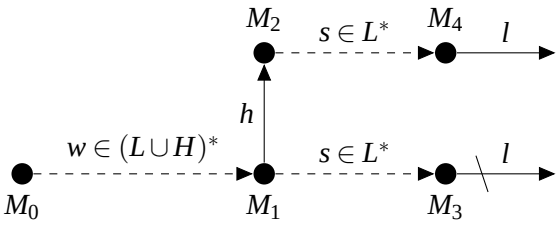


Figure 4: Illustration of Property $P(h,l)$

Proposition 3.16. *$\mathscr{N}$ has the property SBNDC* iff *$P(h,l)$ holds for any high-level action $h \in H$ and for any low-level action $l \in L$.*

Proof. This is a direct consequence of Proposition 3.13. Indeed, $M_1[h\rangle M_2$ and $P(h,l)$ for all l entail $\mathscr{L}(\mathscr{N} \setminus H, M_1) = \mathscr{L}(\mathscr{N} \setminus H, M_2)$, and conversely, $\mathscr{L}(\mathscr{N} \setminus H, M_1) = \mathscr{L}(\mathscr{N} \setminus H, M_2)$ for all transitions $M_1[h\rangle M_2$ entail $P(h,l)$ for all l. □

We will now show that $P(h,l)$ is a decidable property, entailing that one can decide whether a given net system $\mathscr{N}$ has the property SBNDC (because in a finite net, there are finitely many pairs (h,l)).

Proposition 3.17. *$P(h,l)$ is a decidable property.*

Proof. Let a net $\mathscr{N}$ with initial marking M_0 and two fixed transitions $h \in H$ and $l \in L$ be given. Let $\mathscr{N}_1$ be an exact copy of $\mathscr{N}$, with place set P_1, except that it also contains another 'local' copy l_1' of transition l. Let $\mathscr{N}_2$ be another exact copy of $\mathscr{N}$, with place set P_2 (disjoint from P_1), except that it also contains a local copy l_2' of transition l and a local copy h' of transition h. Let $\mathscr{N}'$ be defined as $\mathscr{N}_1|\mathscr{N}_2$ plus two further places x and y and the following extension of F':

(a) x is connected to all transitions in H by a side-condition loop.

(b) $F'(x,h') = 1$, $F'(h',y) = 1$, $F'(y,l_1') = 1$ and $F'(y,l_2') = 1$.

Finally, let x be initially marked with 1 token and y with 0 tokens. The idea is that $\mathscr{N}'$ contains two components, one simulating the path from M_0 to M_3 in Figure 4, and another one simulating the path from M_0 to M_4, if such paths exist.

It is claimed that $P(h,l)$ holds true in $\mathscr{N}$ if and only if in the net $\mathscr{N}'$ so constructed, it is *not* possible to reach a marking M' such that

$$(M'[l_1'\rangle \wedge \neg M'[l_2'\rangle) \vee (\neg M'[l_1'\rangle \wedge M'[l_2'\rangle). \tag{1}$$

To see ($\Rightarrow$), suppose that $M_0'[v\rangle M'$ where M_0' is the initial marking of $\mathscr{N}'$ defined above, and where M' satisfies (1). By (b) and because M' enables either l_1' or l_2', h' occurs exactly once in v, and neither l_1' nor l_2' occur in v. Hence v can be split as $M_0'[v_1h'v_2\rangle M'$ such that v_1 and v_2 contain only transitions of $H \cup L$. By (a), v_2 contains only transitions from L. Because h' does not change the tokens on place set P_1, v_1v_2 is an execution sequence of $\mathscr{N}_1$, whence $M_0[v_1v_2\rangle$ in $\mathscr{N}$. Because h' acts on P_2 exactly as h does, v_1hv_2 is an execution sequence of $\mathscr{N}_2$, whence $M_0[v_1hv_2\rangle$ in $\mathscr{N}$. Because l_1' and l_2' act on P_1 and P_2, respectively, as does l, $M_0'[v_1h'v_2l_1'\rangle$ in $\mathscr{N}'$ iff $M_0[v_1v_2l\rangle$ in $\mathscr{N}$ and $M_0'[v_1h'v_2l_2'\rangle$ in $\mathscr{N}'$ iff $M_0[v_1hv_2l\rangle$ in $\mathscr{N}$. Because M' satisfies (1), this means that $P(h,l)$ is false in $\mathscr{N}$. More precisely, referring to Definition 3.15, putting $w = v_1$ and $s = v_2$ yields $M_0[ws\rangle M_3$ and $M_0[whs\rangle M_4$ with $\neg(M_3[l\rangle \Leftrightarrow M_4[l\rangle)$ in $\mathscr{N}$.

This argument can easily be reversed in order to prove ($\Leftarrow$).

The proof is finished because by Corollary 6.8, it is decidable whether or not a marking satisfying (1) is reachable in $\mathscr{N}'$. □

Corollary 3.18. *SBNDC is decidable for finite PT-nets.*

Corollary 3.19. *BNDC is decidable for finite PT-nets.*

Figures 5 and 6 show an example for the construction in the preceding proof. In Figure 5, which depicts the net $\mathscr{N}$ with $H = \{h\}$ and $L = \{k, l\}$ on its left-hand side, we have

$$M_0[k\rangle M_3 \text{ with } \neg M_3[l\rangle \text{ and } M_0[hk\rangle M_4 \text{ with } M_4[l\rangle,$$

that is, $P(h,l)$ is violated in $\mathscr{N}$. In Figure 6, which depicts the net $\mathscr{N}'$ resulting from the construction in the proof, we have

$$M_0'[h'k\rangle M' \text{ with } \neg M'[l_1'\rangle \text{ and } M'[l_2'\rangle,$$

that is, we find a reachable marking M' satisfying (1).

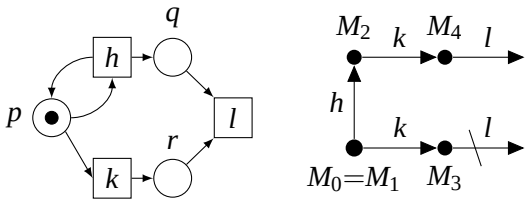


Figure 5: A system $\mathscr{N}$ violating $P(h,l)$

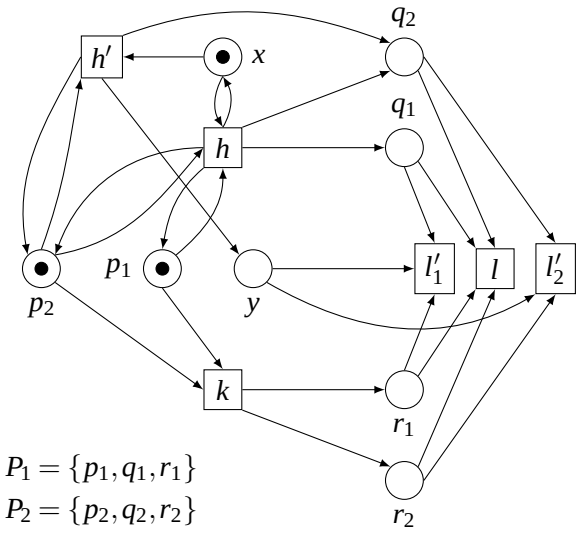


Figure 6: A system $\mathscr{N}'$ satisfying (1) for some M'

4 Intransitive non-interference

We enter now a less technical part of the paper, where we try to show how the decision results established in Section 3 may be applied to check quality of control in the framework of discrete event systems. As it would be difficult to present applications to real systems, we shall consider toy examples which we hope will at least make the intuitions clear.

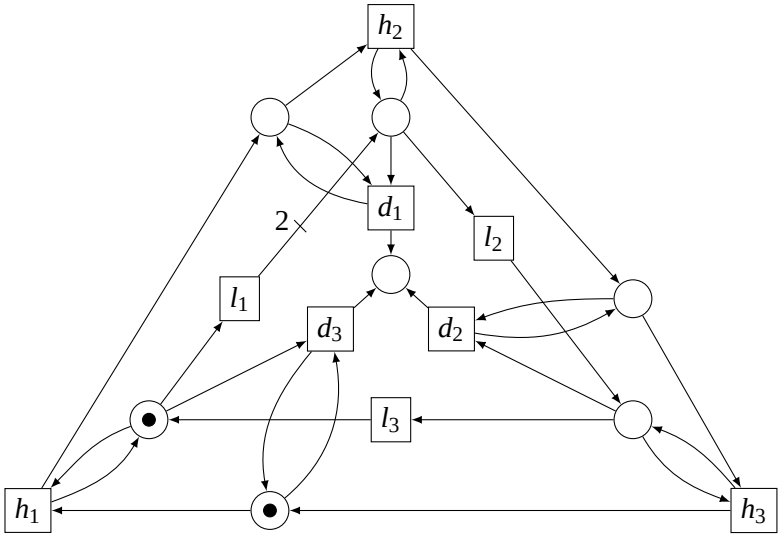


Figure 7: A three-level net system

Our first example is the net system shown in Figure 7. This net is composed of two directed rings interconnected by bidirectional arcs plus a sink place (in the center) fed by three transitions connected to both rings. Each arc from a place p to a transition t means a flow $F(p,t) = 1$. Each arc from a transition t to a place p means a flow $F(t,p) = 1$, except for the arc from l_1 labeled with 2, meaning that $F(l_1,p) = 2$ for the target place p. The internal ring formed with the low-level transitions l_1, l_2, l_3 represents a flock of prey that travel clockwise from place to place, and split each time they go through l_1. The external ring formed with the high-level transitions h_1, h_2, h_3 represents an observer that also travels clockwise and watches the prey but moves only if some prey has been detected in the location currently observed. The three (downgrading) transitions d_1, d_2, d_3 represent the actions of a predator that receives delayed notification of the presence of prey from the observer, and therefore anticipates their possible moves by one position. The objective of the observer and predator is of course to catch prey. The transitions l_1, l_2, l_3 are scheduled by a guardian that pursues the opposite objective. Whenever a prey is caught, this has direct effect on the set of the possible schedules in $\{l_1, l_2, l_3\}^*$, hence there exist interferences between d_1, d_2, d_3 and l_1, l_2, l_3. If the set of possible schedules in $\{l_1, l_2, l_3\}^*$ was directly affected by the transitions in h_1, h_2, h_3, the guardian could glean information on the position of the observer and therefore drive the prey to safe locations. This is actually not the case, because the high-level transitions do not affect the contents of the places connected to the low-level transitions. The fact that each d_i transition reveals that the last transition of the observer was the corresponding h_i makes no problem since the prey has already been caught. This is the essence of downgrading transitions and intransitive non-interference in PT-nets, whose definitions follow.

Definition 4.1 (Three-level net system)**.** *A* three-level *PT-net system is a PT-net system* $\mathcal{N} = (P, T, F, M_0)$

whose set T of transitions is partitioned into low level *transitions $l \in L$,* downgrading *transitions $d \in D$, and* high level *transitions $h \in H$, such that $T = L \cup D \cup H$ and the sets L, D and H do not intersect.*

The low-level transitions are supposed to be observed by the low user, while the high-level transitions cannot be observed and should hopefully be kept *secret*, i.e. they should not be revealed to the low user by the observation of the firing sequences in which they occur. The downgrading transitions may be observed by the low user, but when such a transition occurs, the requirement that all high-level transitions that possibly occurred before should be kept secret is cancelled. This is a strong form of declassification, but we do not know at present about the decidability of INI or BINI for more flexible forms of declassification, where each transition $d \in D$ would declassify a corresponding subset H_d of H (Lemma 4.3, which is crucial to our proofs, does not apply in such a case).

Definition 4.2 (INI-BINI)**.** *A three-level net system (N, M_0) has the property INI (Intransitive Non-Interference), resp. BINI (Bisimulation-Based Intransitive Non-Interference) iff the two-level net system $(N \setminus D, M)$ has the property NDC, resp. BNDC, for $M = M_0$ and for any marking M such that $M_0[\upsilon d\rangle M$ in N for some sequence $\upsilon \in T^*$ and for some downgrading transition $d \in D$.*

The intuition under Definition 4.2 is as follows. The *secret* to be covered is that some high-level transition h has occurred *after the last downgrading transition d*, if any such transition was ever fired in $\mathscr{N}$. Whenever some downgrading transition d is fired, the current secret is deemed obsolete (the high-level transitions that may have occurred before may be revealed by the downgrading transition itself or by subsequent low-level transitions), and a new secret (namely, that some high-level transition may have occurred after the new downgrading transition) is decreed. Thus, INI (resp. BINI) is just a clocked version of NDC (resp. BNDC), where the ticks of the clock are the downgrading transitions. INI/BINI are weakenings of NDC/BNDC but they are still very strong security properties. We feel that such strong properties are really needed in the general context of games, including discrete event systems control as a particular case, where *any* piece of information leaked about the strategy of a player to reach its objective can be used by the adversary to the opposite goal.

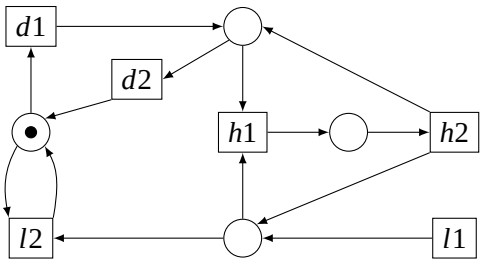


Figure 8: Another three-level net system

In order to illustrate better non-interference in unbounded PT-nets, we would like to present a second example in which the high-level transitions do modify the (contents of the) input places of the low-level transitions. Consider the net system shown in Figure 8. The low-level transition $l1$ is always enabled and it represents the arrival of goods in a shop. The low-level transition $l2$ represents a sale operation and it can only be performed when the shop is open, which is indicated by the presence of one token in the leftmost place. The downgrading transitions $d1$ (closing the shop) and $d2$ (opening the shop) are operated by a guard whose friend takes one article from the shop after closing time (high-level transition $h1$) and brings it back before opening (high-level transition $h2$). It is easily seen that the two high-level transitions form a T-invariant and that $l2$ cannot be fired between $h1$ and $h2$ because the shop is closed during this

period. However, in principle, the guard's friend might grab the key of the shop ($h1$) immediately after each release (by $h2$), and this would impact the low view of the system since the transition $l2$ could then stay blocked forever (blocking may be perceived in weak-bisimulation based semantics). Our definition of BINI does not take this pathologic behaviour into account. Intuitively, Definition 4.2 means that high-level transitions are transparent to the low-level user (that is to say, to the controlled system) unless they cause a starvation of the downgrading transitions (that is to say, of the controller). Therefore, the net system of Figure 8 is secure w.r.t. BINI.

In the rest of the section, we show that both properties INI and BINI can be decided for unbounded PT-nets. $\mathscr{N} = (N, M_0)$ denotes always a three-level net system where $N = (P, T, F)$ and T is partitioned into low-level transitions $l \in L$, high-level transitions $h \in H$, and downgrading transitions $d \in D$.

Lemma 4.3. *(N, M_0) has the property INI iff $(N \setminus D, M) \sim (N \setminus (H \cup D), M)$ for $M = M_0$ and for any marking M such that $M_0[\upsilon d\rangle M$ (in N) with $\upsilon \in T^*$ and $d \in D$.*

Proof. This is a direct application of Proposition 3.4. □

Proposition 4.4. *One can decide whether (N, M_0) has the property INI.*

Proof. First, it can be checked whether $(N \setminus D, M_0) \sim (N \setminus (H \cup D), M_0)$, because all transitions of the net system $(N \setminus (H \cup D), M_0)$ are observable. As a matter of fact, $\mathscr{L}((N \setminus (H \cup D), M_0))$ is always included in $\mathscr{L}((N \setminus D, M_0))$, and by E. Pelz's theorem and corollary (Theorem 6.4 in the appendix), the reverse inclusion can be decided since $\mathscr{L}((N \setminus (H \cup D), M_0))$ is a free PT-net language.

Now fix some downgrading transition $d \in D$. Let $\mathscr{N}_d$ be the net system (with underlying net N_d) constructed as follows.

- N_d has all places of N plus two places p_d and p'_d (the complement of p_d). The initial marking M_{0d} of $\mathscr{N}_d$ extends M_0 by setting one token in p_d and leaving p'_d empty.
- N_d has all transitions t of N with flow relations extended by $F(p_d, t) = 1$ and $F(t, p_d) = 1$.
- N_d has a new transition d' with the same flow relations as d except that $F(d', p_d) = 0$ and $F(d', p'_d) = 1$ (whereas $F(d, p_d) = 1$ and $F(d, p'_d) = 0$).
- N_d has a fresh copy t' of each transition $t \in L \cup H$, with the same flow relations as t except that $F(p'_d, t') = 1$ and $F(t', p'_d) = 1$ (whereas $F(p_d, t) = 1$ and $F(t, p_d) = 1$).
- all transitions of N_d, including H and D, are low-level transitions except for $H' = \{t' \mid t \in H\}$.

We claim that $(N \setminus D, M) \sim (N \setminus H \cup D, M)$ for any M such that $M_0[\upsilon d\rangle M$ in N for the fixed $d \in D$ and for some $\upsilon \in T^*$ iff $\mathscr{N}_d \sim \mathscr{N}_d \setminus H'$ (the proof of this claim, easy but a bit lengthy, is given in the annex, see Claim 6.9). As all transitions of $\mathscr{N}_d \setminus H'$ are observable, the language of this net system is a free PT-net language. It follows by E. Pelz's theorem and corollary (Theorem 6.4 in the appendix) that one can decide on the inclusion relation $\mathscr{L}(\mathscr{N}_d) \subseteq \mathscr{L}(\mathscr{N}_d \setminus H')$. As there are finitely many downgrading transitions $d \in D$, by the above claim, one can decide whether a PT-net system has the property INI. □

Lemma 4.5. *(N, M_0) has the property BINI iff for any reachable marking M_1 of $\mathscr{N}$ and for any high-level transition $h \in H$, $M_1[h\rangle M_2$ entails $\mathscr{L}(N \setminus (H \cup D), M_1) = \mathscr{L}(N \setminus (H \cup D), M_2)$.*

Proof. By Proposition 3.13 and Theorem 3.14, (N, M_0) has the property BINI iff the following entailment relation is satisfied for $M = M_0$ and for any marking M such that $M_0[\upsilon d\rangle M$ (in N) for some $\upsilon \in T^*$ and $d \in D$:
if $M[w\rangle M_1$ in $N \setminus D$ for some $w \in (H \cup L)^*$

and $M_1[h\rangle M_2$ in $N \setminus D$ for some $h \in H$,
then $\mathscr{L}(N \setminus (H \cup D), M_1) = \mathscr{L}(N \setminus (H \cup D), M_2)$.
Grouping the case $M = M_0$ with the other cases, one obtains the lemma. □

Definition 4.6. *Given a three-level net system $\mathscr{N}$ and two transitions $h \in H$ and $l \in L$, we say that $Q(h,l)$ holds* iff *for any words $\chi \in T^*$ and $s \in L^*$, if $M_0[\chi\rangle M_1$, $M_1[h\rangle M_2$, $M_1[s\rangle M_3$, and $M_2[s\rangle M_4$, then $M_3[l\rangle$* iff *$M_4[l\rangle$.*

Proposition 4.7. *One can decide whether (N, M_0) has the property BINI.*

Proof. By Lemma 4.5, $\mathscr{N}$ has the property BINI iff $Q(h,l)$ holds for every high-level action h and for every low-level action l. As $Q(h,l)$ is the same as $P(h,l)$, up to replacing H with $H \cup D$, $Q(h,l)$ is decidable. Therefore, the BINI property can be decided for PT-net systems. □

As nets are labeled injectively on transitions, $\mathscr{L}(N \setminus (H \cup D), M_1) = \mathscr{L}(N \setminus (H \cup D), M_2)$ iff $M_1 \approx M_2$ w.r.t. $\Sigma_o = L$. Therefore, BINI coincides exactly with the property BNID specified by Definition 5.7 in [8].

5 Conclusion and future work

The examples we have discussed seem to suggest that there is a clear, structural reason why an interference is present in a net system: either a high-level transition is causing a low-level transition (e.g., Example 3.5) or a high-level transition and a low-level one are competing for the same token in a place (e.g., Example 3.8). As a matter of fact, in [2] one of the authors showed that precisely this is the case when restricting net systems to elementary net systems (which are essentially PT-nets where each place can contain at most one token). More precisely, a (contact-free) elementary net system $\mathscr{N}$ is BNDC if and only if it is never the case that a low transition consumes a token that *must* have been produced by a high transition nor that a high transition and a low-transition compete for the very same token in a place.

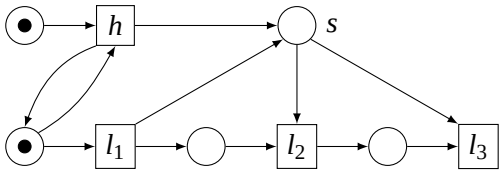


Figure 9: A non BNDC net

Unfortunately, generalizing this characterization in the setting of general PT-nets seems problematic. Consider the net system $\mathscr{N}$ shown in Figure 9. Let M_0 be the initial marking indicated in the figure. Set $M_0[h\rangle M_1$ and set also $M_0[l_1 l_2\rangle M_2$ and $M_1[l_1 l_2\rangle M_3$. Clearly, transition l_3 is enabled at M_2 but disabled at M_3, hence $\mathscr{N}$ is not BNDC. However, in the firing sequence $M_0[h l_1 l_2 l_3\rangle$, the token consumed from place s by the low-level transition l_3 *may* have been produced by the high-level transition h but it *may* also have been produced alternatively by the low-level transition l_1.

As regards continuations of this work, it would be useful to look at flexible versions of downgrading, where each downgrading action bears upon a specific subset of high-level actions. A wider perspective would be to investigate non-interference in the framework of games of partial information, see e.g. [15] for a survey on Games for Security.

Acknowledgment

The authors would like to thank the reviewers for their comments.

References

[1] N. Busi and R. Gorrieri. A Survey on Non-Interference with Petri Nets. *Advanced Course on Petri Nets 2003*, Springer LNCS 3098:328-344, 2004.

[2] N. Busi and R. Gorrieri. Structural Non-Interference in Elementary and Trace Nets. *Mathematical Structures in Computer Science*, 19(6):1065-1090, 2009.

[3] M. Dam. Decidability and Proof Systems for Language-based Noninterference Relations, in Proc. *POPL'2006* 67-78, 2006.

[4] R. Focardi, R. Gorrieri. A Classification of Security Properties. *Journal of Computer Security* 3(1) pp.5-33, 1995.

[5] R. Focardi, R. Gorrieri. Classification of Security Properties (Part I: Information Flow), *Foundations of Security Analysis and Design* - Tutorial Lectures (R. Focardi and R. Gorrieri, Eds.), Springer LNCS 2171:331-396, 2001.

[6] J.A. Goguen, J. Meseguer. Security Policy and Security Models. Proc. of Symposium on Security and Privacy (SSP'82), IEEE CS Press, pp. 11-20, 1982.

[7] S. Ginsburg, E.H. Spanier. Bounded Algol-like languages. *Trans. Amer. Math. Soc.* 113:333-368, 1964

[8] R.Gorrieri, M. Vernali. On Intransitive Non-interference in Some Models of Concurrency. submitted, 2009.

[9] M.H.T. Hack. "Petri Net Languages", Technical Report 159, MIT, 1976.

[10] M.H.T. Hack. Decidability questions for Petri nets. PhD thesis, MIT, 1976. *available at* http://dspace.mit.edu/handle/1721.1/27441

[11] P. Jančar. Undecidability of bisimilarity for Petri nets and some related problems. *Theoretical Computer Science* 148(2):281-301, 1995.

[12] E.W. Mayr. An Algorithm for the General Petri Net Reachability Problem. *SIAM J. Comput.* 13(3): 441-460, 1984.

[13] D. McCullough. Noninterference and the Composability of Security Properties. In Proceedings 1988 IEEE Symposium on Security and Privacy, pages 178-186, IEEE Computer Society Press, April 1988.

[14] E. Pelz. Closure Properties of Deterministic Petri Nets. Proc. of STACS'87, Springer LNCS 247:671-681, 1987.

[15] S. Roy, C. Ellis, S. Shiva, D. Dasgupta, D. Shandilya and Q. Wu. A Survey of Game Theory as Applied to Network Security. Proc. HICSS'10, IEEE Computer Society, 1-10, 2010.

[16] P.J. Ramadge and W.M. Wonham. Supervisory Control of a Class of Discrete Event Systems. *SIAM J. Control and Optimization* 25: 206-230, 1987.

[17] P.J. Ramadge and W.M. Wonham. The Control of Discrete Event Systems. *Proc. IEEE, Special Issue on Dynamics of Discrete Event Systems* 77: 81-98, 1989.

[18] J. Rushby. Noninterference, Transitivity, and Channel-control Security Policies. Technical Report CSL-92-02, SRI International, 1992.

[19] P.Y.A. Ryan. Mathematical Models of Computer Security. *Foundations of Security Analysis and Design* - Tutorial Lectures (R. Focardi and R. Gorrieri, Eds.), Springer LNCS 2171:1-62, 2001.

[20] P.Y.A. Ryan, S. Schneider. Process Algebra and Noninterference, Proc. of 12th Computer Security Foundations Workshop, IEEE CS Press, pp. 214-227, 1999.

[21] J.T. Wittbold, D.M. Johnson. Information Flow in Nondeterministic Systems, In Proceedings of the 1990 IEEE Symposium on Research in Security and Privacy, pages 144-161, IEEE Computer Society Press 1990.

6 Annex

Definition 6.1 (PT-nets)**.** *A* PT-net *is a bi-partite graph $N = (P,T,F)$, where P and T are* finite *disjoint sets of vertices, called* places *and* transitions*, respectively, and $F : (P \times T) \cup (T \times P) \to \mathbb{N}$ is a set of directed edges with non-negative integer weights. A* marking *of N is a map $M : P \to \mathbb{N}$. A transition $t \in T$ is* enabled at *a marking M (notation:$M[t\rangle$) if $M(p) \geq F(p,t)$ for all places $p \in P$. It t is enabled at M, then it can* be fired*, leading to the new marking M' (notation: $M[t\rangle M'$) defined by $M'(p) = M(p) + F(t,p) - F(p,t)$ for all $p \in P$. These definitions are extended inductively to transition sequences $s \in T^*$: for the empty sequence ε, $M[\varepsilon\rangle$ and $M[\varepsilon\rangle M$ are always true; for a non-empty sequence st with $t \in T$, $M[st\rangle$ (or $M[st\rangle M'$) iff $M[s\rangle M''$ and $M''[t\rangle$ (or $M''[t\rangle M'$, respectively) for some M''. A marking M' is* reachable *from a marking M if $M[s\rangle M'$ for some $s \in T^*$. The set of markings reachable from M is denoted by $[M\rangle$.*

Theorem 6.2 (Mayr [12])**.** *Given a PT-net N and two markings M and M', one can decide whether M' is reachable from M.*

Definition 6.3 (Free language of a net system)**.** *The* free language *of a Petri net system $\mathscr{N}$ is the language of the LTS $RG(\mathscr{N})$, where all transitions are considered observable, i.e., $\Sigma_o = T$. In this case, we write $\mathscr{L}(\mathscr{N})$ to denote the free language.*

Theorem 6.4 (Pelz [14])**.** *The complement in Σ_o^* of the free language of a net system may be generated by a labeled net $(\mathscr{N},\lambda)$ with a finite set of final partial markings, characterized by a formula $\mathscr{F}$ built from the logical connectives $\wedge$ and $\vee$ and atomic formulas $M(p) = i$ (with $p \in P$ and $i \in \mathbb{N}$). In other words, a sequence $s \in \Sigma_o^*$ belongs to this complement if and only if $s = \lambda(t_1 t_2 \ldots t_n)$ for some sequence of transitions $M_0[t_1 t_2 \ldots t_n\rangle M$ of $\mathscr{N}$ such that M satisfies $\mathscr{F}$.*

Corollary 6.5 (Pelz)**.** *The problem whether the language of a labeled net system $\mathscr{N}_1$ is included in the free language of a net system $\mathscr{N}_2$ is decidable.*

Proof. The language of $\mathscr{N}_1$ is included in the free language of $\mathscr{N}_2$ if and only if no marking satisfying $\mathscr{F}$ can be reached in $\mathscr{N}_1 \,|\, \mathscr{N}_2'$ where $\mathscr{N}_2'$ is the complementary net of $\mathscr{N}_2$ and $\mathscr{F}$ is the logical formula defining the final partial markings of $\mathscr{N}_2'$. The latter reachability property can be decided in view of the Proposition 6.7 recalled below in this appendix. □

In order to make the statement of Proposition 6.7 understandable, let us recall first the basics of semi-linear sets and their decidable properties. Given a number $n \in \mathbb{N}$, we consider the commutative monoid $(\mathbb{N}^n, +)$ where $+$ denotes the componentwise addition of n-vectors and the null n-vector is the neutral element. Typically, n is the number of places of a Petri net and then $\mathbb{N}^n$ is the realm of all possible markings of this net (markings are seen as vectors in which each entry defines the number of tokens in the corresponding place for some fixed enumeration of the places of the net).

A subset $E \subseteq \mathbb{N}^n$ is called *linear* if it is of the form

$$E = \{a + k_1 \cdot b_1 + \ldots + k_m \cdot b_m \mid k_1, \ldots, k_m \in \mathbb{N}\}$$

for some specific vectors $a \in \mathbb{N}^n$ and $b_1, \ldots, b_m \in \mathbb{N}^n$. For example, let an unmarked net with n places and a transition t be given. Then the set of markings enabling t is linear, since any such marking M can be expressed as the following sum:

$$M \;=\; M_t + k_1 \cdot b_1 + \ldots + k_n \cdot b_n$$

where M_t is the (unique!) minimal marking enabling t and the $b_1,\ldots,b_n$ are the unit vectors corresponding to the places of the net. The natural numbers $k_1,\ldots,k_n$ simply describe excess tokens which may be present in M but are not needed for enabling t.

A subset $E \subseteq \mathbb{N}^n$ is called *semi-linear* if it is a finite union of linear sets. For example, if t_1 and t_2 are two transitions, then the set of markings enabling t_1 or t_2 (or both) is semi-linear, since it is the union of the set of markings enabling t_1 and the set of markings enabling t_2.

Theorem 6.6 (Ginsburg and Spanier [7])**.** *The semi-linear subsets of* $\mathbb{N}^n$ *form an effective boolean algebra.*

Thus, if E, E_1 and E_2 are semi-linear subsets of $\mathbb{N}^n$, then so are $\mathbb{N}^n \setminus E$, $E_1 \cap E_2$ and $E_1 \cup E_2$. The effectiveness part of Ginsburg and Spanier's theorem concerns the possible description of semi-linear sets as linear expressions, and it states that the expressions of a composed set (such as $E_1 \cap E_2$) can be computed effectively from the linear expressions of the constituent set(s) (such as E_1 and E_2).

Proposition 6.7. *Given a PT-net system* $\mathcal{N} = (P,T,F,M_0)$ *and a semi-linear subset of markings* $E \subseteq \mathbb{N}^n$*, where* $n = |P|$*, one can decide whether (some marking in)* E *can be reached from* M_0*.*

The above proposition follows from Lemma 4.3 in [10] where the semi-linear reachability problem is reduced to the reachability problem, and from Theorem 6.2.

In this paper, we use Proposition 6.7 and Theorem 6.6 in the special form as follows.

Corollary 6.8. *Let* $\mathcal{N}$ *be a PT-net system with initial marking* M_0 *and let* t_1 *and* t_2 *be two transitions. The question whether there is some marking* $M \in [M_0\rangle$ *with*

$$(M[t_1\rangle \wedge \neg\, M[t_2\rangle) \vee (\neg\, M[t_1\rangle \wedge M[t_2\rangle) \tag{2}$$

is decidable.

Proof. The set of all markings M satisfying (2) is semi-linear. This follows from Theorem 6.6, together with the fact that the set of markings enabling a single transition is linear. The claim now follows directly from Proposition 6.7. □

We finally give a detailed proof of the claim made in the proof of Proposition 4.4.

Claim 6.9. *With the notations used in the proof of Proposition 4.4* $(N \setminus D, M) \sim (N \setminus H \cup D, M)$ *for any* M *such that* $M_0[\upsilon d\rangle M$ *in* N *for some* $\upsilon \in T^*$ **iff** $\mathcal{N}_d \sim \mathcal{N}_d \setminus H'$*.*

Proof. We need examining closely the relationship between the firing sequences of N and N_d. Let $M_0[\upsilon d\rangle M$ be a firing sequence of N and let $M[t_1\ldots t_n\rangle$ be a firing sequence of $N \setminus D$. Then $M_{0d}[\upsilon d\rangle M_d$ in $\mathcal{N}_d$ where $M_d(p_d) = 1$, $M_d(p'_d) = 0$, and $M_d(p) = M(p)$ for every place p of N. Clearly, $M_d[t_1\ldots t_n\rangle$ is a firing sequence of $N_d \setminus D$. In a similar way, $M_{0d}[\upsilon d'\rangle M'_d$ in $\mathcal{N}_d$ where $M'_d(p_d) = 0$, $M'_d(p'_d) = 1$, and $M'_d(p) = M(p)$ for every place p of N. Also clearly, $M'_d[t'_1\ldots t'_n\rangle$ is a firing sequence of $N_d \setminus D$. Conversely, consider now a firing sequence $M_{0d}[u\rangle$ in $\mathcal{N}_d$. If d' does not occur in u, then $M_0[u\rangle$ in N. If $u = \upsilon d' w$, then necessarily, $M_0[\upsilon d\rangle M$ for some M in N, and $w = t'_1\ldots t'_n$ for some sequence $t_1\ldots t_n \in (L \cup H)^*$ such that $M[t_1\ldots t_n\rangle$ in N and hence also in $N \setminus d$.

Suppose that $(N \setminus D, M) \sim (N \setminus H \cup D, M)$ for any M such that $M_0[\upsilon d\rangle M$ in N for the fixed $d \in D$ and for some $\upsilon \in T^*$. By construction, any sequence of transitions of $\mathcal{N}_d$ not including d' is also a sequence of transitions of $\mathcal{N}_d \setminus H'$. Now any sequence of transitions of $\mathcal{N}_d$ including d' is of the form $M_{0d}[\upsilon d' t'_1\ldots t'_n\rangle$, where no transition from H' occurs in υ and $t'_1\ldots t'_n$ is the primed version of some sequence $t_1\ldots t_n \in (L \cup H)^*$. Then, $M_0[\upsilon d\rangle M$ and $M[t_1\ldots t_n\rangle$ for some M in N. For all t_j let $\lambda(t_j) = \varepsilon$

if $t_j \in H$ and $\lambda(t_j) = t_j$ otherwise. As $(N \setminus D, M) \sim (N \setminus H \cup D, M)$, one has also $M[\lambda(t_1)\ldots\lambda(t_n)\rangle$. Therefore, if we let $\lambda'(t'_j) = \varepsilon$ if $t'_j \in H'$ and $\lambda'(t'_j) = t'_j$ otherwise, then $M'_d[\lambda'(t'_1)\ldots\lambda'(t'_n)\rangle$ in $N_d \setminus D$ where M'_d is the marking of $\mathscr{N}_d$ defined with $M'_d(p_d) = 0$, $M'_d(p'_d) = 1$, and $M'_d(p) = M(p)$ for every place p of N. As no transition from H' occurs in $\upsilon d'\lambda'(t'_1)\ldots\lambda'(t'_n)$, this sequence is a firing sequence of $\mathscr{N}_d \setminus H'$. Thus, $\mathscr{N}_d \sim \mathscr{N}_d \setminus H'$.

In order to establish the converse implication, suppose now that $\mathscr{N}_d \sim \mathscr{N}_d \setminus H'$. Consider any two firing sequences $M_0[\upsilon d\rangle M$ and $M[t_1\ldots t_n\rangle$ of N with $t_1\ldots t_n \in (L \cup H)^*$. By construction of $\mathscr{N}_d$, $M_{0d}[\upsilon d' t'_1\ldots t'_n\rangle$. As no transition from H' occurs in υ, by the above assumption, $M_{0d}[\upsilon d'\lambda'(t'_1)\ldots\lambda'(t'_n)\rangle$ in $\mathscr{N}_d \setminus H'$ where $\lambda'(t'_j) = \varepsilon$ if $t'_j \in H'$ and $\lambda'(t'_j) = t'_j$ otherwise. Thus, if we set $\lambda(t_j) = \varepsilon$ if $t_j \in H$ and $\lambda(t_j) = t_j$ otherwise, then $M_{0d}[\upsilon d\lambda(t_1)\ldots\lambda(t_n)\rangle$ by construction of $\mathscr{N}_d$. . As a consequence, $M[\lambda(t_1)\ldots\lambda(t_n)\rangle$ in N and hence also in $N \setminus H \cup d$, concluding the proof of the claim. □

Covert channel detection using Information Theory

Loïc Hélouët Aline Roumy
INRIA Rennes, Campus de Beaulieu, 35042 Rennes Cedex, France
loic.helouet@irisa.fr aline.roumy@inria.fr

This paper presents an information theory based detection framework for covert channels. We first show that the usual notion of interference does not characterize the notion of deliberate information flow of covert channels. We then show that even an enhanced notion of "iterated multivalued interference" can not capture flows with capacity lower than one bit of information per channel use. We then characterize and compute the capacity of covert channels that use control flows for a class of systems.

1 Introduction

The term covert channel was first introduced by Lampson [13], and designates an information flow that violates a system's security policy. In a system, this policy can define who is allowed to communicate with whom, through which channels, and forbid all exchanges other than these legitimate ones. Security policies can also define filtering or billing policies when legal channels are used, and which exchanges should be observed and recorded. They can be implemented by system monitors, that ensure that unauthorized communications do not occur, and record some events of the system. Within this context, a *covert channel* is a perverted use of a system by two legal users. These users have access to system's functionalities, but use them in a way that bypasses the security policy (for instance to create a communication channel between two users that are not allowed to communicate, or to pass information between authorized users without paying for it, etc.). One usual assumption is that both corrupted users know perfectly the system, and have agreed on a particular use of the functionalities to encode and decode information.

Unsurprisingly, this preoccupation for covert information flows appeared in the 70's, with a particular attention paid to information systems. The fear during this period was that an agent with high level accreditation would read classified information, and send them discretely to another agent with low accreditation. This covert channel problem also has an economic interpretation: covert flows can be used to establish free communications over paying services. Nowadays, with the increase of online transactions and personal computers, the problem seems more individual: the threat is that a Trojan horse can communicate personal information (agenda, credit card numbers,...) to a third party via covert channels that would bypass all protections of the computer (firewalls, anti viruses,...).

Many security recommendations [5, 17] consider covert channels in their lists of threats, and ask for the application of reproducible methods to characterize channels, evaluate their capacity, and depending on the severity of the threat, to close or lower the information leak. Of course, "reproducible methods" advocates for the use of formal models and formal techniques. Many model-based methods have been proposed, such as shared matrices, non-interference checking, etc. Note however that it is commonly agreed that one particular technique can not capture all kinds of information leaks. The first formal model allowing the automation of security leaks discovery is the well-known Bell & La Padula model introduced in [3, 2]. It can be modeled as a matrix [12] defining accesses of agents to objects in the system, and a security leak occurs if the transitive closure of the matrix contains a forbidden access. Since

K. Chatzikokolakis and V. Cortier (Ed.): 8th International Workshop
on Security Issues in Concurrency (SecCo 2010)
EPTCS 51, 2011, pp. 34–51, doi:10.4204/EPTCS.51.3

the 80's, information leak is mainly considered through the notion of *interference* [8], that characterizes information leak from a high level (confidential) part of the system to low level (public) part. At first sight, this looks very similar to the definition of covert flows: one can search for interference from u to v by declaring as confidential all actions of u, and public all actions of v. However, we will show that interference and covert flows are orthogonal notions of information leakage. A first difference is that covert flows are situations in which two agents cooperate to allow transfer of information, while an interference means that some classified or confined information can be recovered by an agent from its observations of the running system, without collaboration. A second difference is that corrupted users of a covert flow must be able to transfer any message of arbitrary size in a bounded time (this is called the "small message criterion" [15]), while leaking a single bit of information in a run, or the same information an arbitrary number of times is sufficient for a system to be interferent. An immediate idea to extend interference is to consider a notion of iterated, deliberate and multivalued interference. We will detail this possibility in the paper, and show that such notion still misses some obvious covert flows. We hence propose a new characterization for covert flows in systems modeled as transition systems. This characterization considers that a covert channel exists between two users u,v if u and v can use the system to simulate a memoryless discrete channel with state of capacity greater than 0.

This paper is organized as follows: section 2 introduces the notion of interference and the formal material that is used in the paper. Section 3 shows some differences between the notions of interference and covert channels. The notion of communication channel used in the paper needs some elements of information theory, that are introduced in Section 4. Section 5 shows that former trials that extend the notion of interference either through a notion of "iterated" interference , or via the quantification of common knowledge of processes fail to characterize covert channels. Section 6 is an easy covert channel example that illustrates the characterization of covert channels proposed in section 7. Section 8 discusses some technical choices, and concludes. Some details omitted in the paper can be found in an extended version at `www.irisa.fr/distribcom/Personal_Pages/helouet/Papers/Secco2010_extended.ps`.

2 Non-interference

The term *non-interference* was first introduced in [8]. In the original definition, an agent u interferes with an agent v in a system S iff "what u does can affect what v can observe or do". The proposed model on which interference is checked is a kind of transition system, in which moves from one state to another are performed by one agent, and where each agent has in addition some "capacities" that allow him to test inputs/outputs to the system or observe the values of some variables in each state of the system. Within this context, we can see S as a system composed of several agents and subsystems, i.e. $S = u \mid p_1 \mid \cdots \mid p_k \mid v$, and u does not interfere with v in system S iff the system behaves similarly from v's point of view, independently from the fact that u performs some actions or not. This can be written formally as: $\Pi_v(u \mid p_1 \mid \cdots \mid p_k \mid v) \sim \Pi_v(p_1 \mid \cdots \mid p_k \mid v)$, where $\Pi_v(S)$, the projection of S on process v represents what v can observe from S, and $\sim$ is an equivalence relation between systems.

This latter definition of non-interference is very generic, as we have not precised the nature of $S, \Pi,$ or $\sim$. One may immediately notice that non-interference is not uniquely defined for a kind of system, and depends on the considered equivalence, which in some sense captures the discriminating power of an observer of the system. The choice of a given kind of model and of an equivalence between models (trace equivalence, bisimulation, testing equivalence,...) allows the definition of a large variety of non-interferences. To be able to decide if there exists an interference between u and v, non-interference must rely on a decidable equivalence relation for the models used to represent the behaviors of the system. For instance, if S is modeled by communicating automata, and $\sim$ is trace equivalence, then non-interference

is undecidable. In the rest of the paper, we will use *transition systems* to represent distributed systems behaviors.

Definition 1 *A* transition system *is a tuple $S = (Q, \longrightarrow, \Sigma, q_0)$ where Q is a finite set of states, q_0 is the initial state of the system, $\longrightarrow \subseteq Q \times \Sigma \times Q$ is a transition relation, Σ is an alphabet of actions. We will furthermore consider the unobservable action $\tau \notin \Sigma$ such that for all $a \in \Sigma$, $\tau.a = a.\tau = a$. Transition systems define the behaviors of a set of processes $\mathscr{P}$. Each action in Σ is executed by a single process, and observed by several processes. This is modeled by two functions $Ex : \Sigma \longrightarrow \mathscr{P}$ and $Obs : \Sigma \longrightarrow 2^{\mathscr{P}}$.*
A *path* in a transition system S is a sequence of transitions $\rho = (q_1, \sigma_1, q_2)(q_2, \sigma_1, q_3)\dots(q_{k-1}, \sigma_{k-1}, q_k)$. We will also write $\rho = q_1 \xrightarrow{\sigma_1} q_2 \xrightarrow{\sigma_2} q_3 \dots q_{k-1} \xrightarrow{\sigma_{k-1}} q_k$, and denote by $Path(x,y)$ the set of paths starting in x and ending in y.

Definition 2 *The* language *of a transition system S is the set of words $\mathscr{L}(S) \subseteq \Sigma^*$ such that for all $w = \sigma_1.\sigma_2\dots\sigma_k \in \mathscr{L}(S)$ there exists a path $q_0 \xrightarrow{\sigma_1} q_1 \xrightarrow{\sigma_2} \cdots \xrightarrow{\sigma_k} q_k$ starting in the initial state of S. We will say that two transition systems S and S' are* equivalent *(denoted by $S \sim S'$) iff $\mathscr{L}(S) = \mathscr{L}(S')$.*

As we want to consider systems in which some processes (for instance the environment) behave non-deterministically, we will attach to the firing of transitions from a given state a discrete probabilistic distribution. We then associate to a transition system S a probability function $P_S : Q \times \Sigma \times Q \longrightarrow \mathbb{R}$, with the constraint that $\forall q \in Q, \sum_{q' \in Q, a \in \Sigma} P_S(q,a,q') = 1$. To simplify notations, P_S is only partially defined, and we assume a uniform distribution on outgoing transitions from each state q for which P_S is not defined. Function P_S also allows for the probabilization of paths and words. For $\rho = q_0 \xrightarrow{\sigma_1} q_1 \xrightarrow{\sigma_2} \cdots \xrightarrow{\sigma_k} q_k$, we have $P_S(\rho) = P_S(q_0, \sigma_1, q_1).P_S(q_1, \sigma_2, q_2)\dots P_S(q_{k-1}, \sigma_k, q_k)$.

Definition 3 *The* projection *of a transition system $S = (Q, \longrightarrow, \Sigma, q_0)$ over an alphabet $X \subseteq \Sigma$ is the system $\Pi_X(S) = (Q, \longrightarrow', \Sigma, q_0)$, such that $\longrightarrow' = \{(q,a,q') \mid a \in X\} \cup \{(q,\tau,q') \mid \exists (q,a,q') \wedge a \notin X\}$. The* restriction *of S to X is the system $S_{\setminus X} = (Q'', \longrightarrow'', \Sigma, q_0)$, where $\longrightarrow'' = \{(q,a,q') \mid a \in X\}$ and Q'' is the restriction of Q to states that remain accessible via $\longrightarrow''$.*

Projections and restrictions can be used to focus on a specific aspect of a system: $\Pi_{Obs^{-1}(u)}(S)$ defines what process u observes from S. $S_{\setminus Ex^{-1}(u)}$ defines allowed behaviors of S when process u does not perform any action. To simplify notations, we will write $\Pi_u(S) = \Pi_{Obs^{-1}(u)}(S)$ and $S_{\setminus u} = S_{\setminus Ex^{-1}(u)}$. Of course, projections extend to words and languages by defining for every $X \subseteq \Sigma$, every $a \in \Sigma$, and every word $w \in \Sigma^*$ the projection as $\Pi_X(a.w) = \Pi(w)$ if $a \notin X$, and $\Pi_\Sigma(a.w) = a.\Pi_X(w)$ otherwise.

Definition 4 *User u* interferes *with user v in system S if and only if $\Pi_v(S) \not\sim \Pi_v(S_{\setminus u})$*

More intuitively, definition 4 says that what user v sees from system S (i.e. $\Pi_v(.)$) changes when process u is allowed to do some actions or not (i.e. if v can distinguish if it observes S or $S_{\setminus u}$). This definition is only one among many definitions of interference. It is usually called SNNI (Strong Non-deterministic Non Interference) in the literature. A similar notion called BSNNI (Bisimulation based SNNI) exists where $\sim$ is replaced by a bisimulation relation. We refer interested reader to [6], which defines, compares and classifies several interferences for systems described with process algebra. Another interesting state of the art can also be found in [18]. In the rest of the paper, we will focus on SNNI, but keeping in mind that the differences highlighted in sections 3 and 5 hold for interference in general. Note also from definition 4 that during his observation of a system S, an agent v may observe sequences of actions that provide him with some information on u's behavior, but that no cooperation from u is a priori needed to get this information.

3 First differences between interference and covert flows

An interference in a distributed system means that a process of the system (or a user) can obtain some confidential information on values of variables, or about other users behaviors through its observations.

Several papers consider that covert channels are a sub-case of interference. In this section, we will show that interference captures a notion of *information leak*, but does not necessarily characterize *deliberate information flows* of arbitrary size. Let us consider the examples of Figure 1, that depict the behavior of systems involving two users u and v, and where action a is executed and observed by u, and actions $\{b,c,d\}$ are observed and executed by v. For these three transition systems, the initial state is state 0. In $S1$, user u can perform action a, and then user v can execute any prefix of $(bcc+bdc)*$. The projection $\Pi_v(\mathscr{L}(S1))$ is the set of prefixes of $(bcc+bdc)^*$, and the projection $\Pi_v(\mathscr{L}(S1_{\setminus u}))$ is the empty word ε. Hence, from definition 4, processes u interferes with v in $S1$. Note however that this interference is due to a single transition, which can be fired only once in each execution of the system. System $S2$ depicts the converse situation: user v can execute any sequence of actions in $(bcc+bdc)^*(bc+bd)$ before user u executes action a, hence v interferes with u in $S2$. However, when u executes a, it is impossible for him to detect which sequence of actions of v occurred before a. Furthermore, after executing a, the system remains deadlocked in state 1. Transition systems $S1$ and $S2$ are example of interferent specification where interference occurs *only once*, and which can not be used to transmit a message of *arbitrary size*, as usually expected in covert channels.

Considering covert channels as a sub-case of interference implicitly supposes that the detected interferences can be repeated an arbitrary number of time, to transmit a message of arbitrary size. This means that when a system always reaches a sink state after interfering, it is then set back to its initial state. We think that this interpretation does not hold for most systems (for instance when sink states represent faulty deadlocked states), and that system resets should be explicitly modeled in the specification if they can occur. Note also that covert channels are supposed to be as discreet as possible, and that causing a fault that needs resetting a system does not really comply with this assumption.

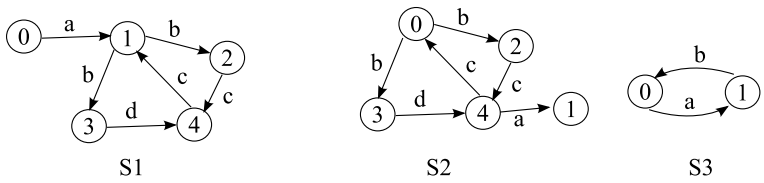


Figure 1: Examples of interferent systems

The last system $S3$ in Figure 1 contains interferences from u to v and from v to u. These interferences can be repeated an arbitrary number of times, which would a priori allow for the encoding of a message of arbitrary size. However, user v observes a sequence of b's of arbitrary size, without knowing whether the covert message has been completely transmitted or not. This situation hence does not allow an encoding of a covert message. Yet, there is a possibility to pass some information from u to v (or conversely) if time can be used as a vector of information, that is if u and v can measure the time elapsed between two a's or b's. Processes u and v can for instance agree that if action a is fired within a short time interval after it is enabled, it means bit 0, and if the same action is delayed, it means bit 1. Covert channels that use time measurement to pass information are frequently called *timing channels*. We will not address timing channels in this paper, and focus on covert channels that use control flows of systems to transfer information. Note however that time elapsing and measurement can respectively be seen as the inputs and outputs of an untimed channel, and that the characterization of section 7 may still work in the timed cases.

4 Information Theory

The characterization proposed in section 7 for covert flows shows that a pair of users can simulate a *memoryless communication channel*, an usual notion of coding theory. We hence recall some elements of information theory that are needed in the rest of the paper. Interested readers are referred to [4] for a reference book on information theory.

A *discrete random variable* X takes values in a discrete alphabet $\mathscr{X}$ and is characterized by its probability mass function $P(X = x)$ that gives the probability that X is exactly equal to some value $x \in \mathscr{X}$. In the sequel, we will denote by $p(x)$ this probability. Consider for example the reduced card set of Figure 2-a:

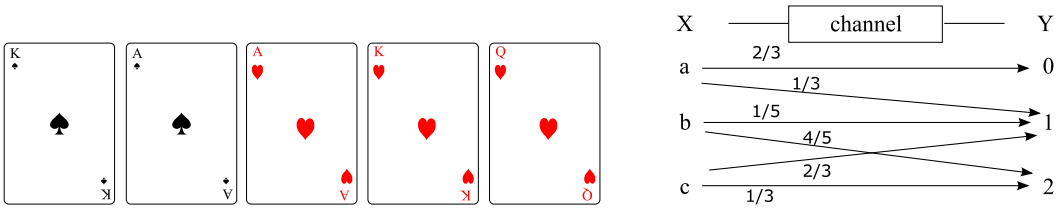


Figure 2: a) A reduced card set b) A discrete channel model

One can consider as random variable *Card* the complete name of a card, that is a pair $(value, color)$ in a set $\{(King, \spadesuit), (Ace, \spadesuit), (Ace, \heartsuit), (King, \heartsuit), (Queen, \heartsuit)\}$, and associate probability $1/5$ to each value. Now, if we only consider the color of the card, we define a random variable (say *Color*) with value set $\{\spadesuit, \heartsuit\}$, and associated probabilities $2/5$ and $3/5$. We can also define another random variable *Value*, over a domain $\{King, Ace, Queen\}$, with associated probabilities $2/5, 2/5, 1/5$. The random experience can be repeated, and we will also consider sequences of random variables $X_1.X_2 \ldots X_n$, denoting n consecutive choices of a value for X. In the sequel, x_n denotes the n^{th} value taken by the variable X_n and we will write X^n (resp. x^n) instead of $X_1 \ldots X_n$ (resp. $x_1 \ldots x_n$).

The **entropy** (expressed in bits) is a measure of the uncertainty associated with a random variable, and is defined as $H(X) = -\sum_{x \in \mathscr{X}} p(x).\log_2 p(x)$. In some sense, entropy measures the average number of binary questions to ask to know the value of a random variable after a random choice. Let us get back to the card set of Figure 2-a. The entropy of variable *Card* is $H(Cards) = -5.(1/5).\log_2(1/5) = 2.32$. Now, our card set can be seen as a pair of random variables *Value*,*Color*. Let us randomly choose a card. Knowing the card set, we can apply the following strategy to guess the correct pair of random variables: first, discover the color of the card (this is done with a single question), and then discover its value (this can be done asking at most two questions). The average total number of question to ask with this strategy is 2.4 and this is more efficient than enumerating all values, which leads to asking 2.8 questions on average. Note also that knowing the color of a card provides some information on its value: if a card is a spade, then it is useless to ask whether it is a queen to discover the chosen card. This is explained by the fact that variables *Color* and *Value* are not independent. The quantity of information shared between two random variables is called the **mutual information**, and is defined as $I(X;Y) = H(X) + H(Y) - H(X,Y)$.

From this definition, one can show that $I(X;Y) = H(X) - H(X|Y)$, which provides a quite intuitive explanation of mutual information. The mutual information between X and Y is the uncertainty on X minus the uncertainty on X that remains when Y is known. This notion of mutual information will be used later to evaluate the capacity of communication channels. This value is symmetric, so we have $I(X;Y) = H(X) - H(X|Y) = H(Y) - H(Y|X) = I(Y;X)$

In communication systems, a frequent challenge is to measure the amount of information that can be passed from a source to a destination. The communication medium can be a wire, a network, an

hertzian channel, etc. Transmissions from the source to the destination are encoded and transmitted along the medium to the receiver with a fixed rate. One usually considers that the symbols that are input in the medium and those that are received at the other end, have not necessarily the same alphabets. Furthermore, the channel can be noisy, and there is no one-to-one correspondence between the inputs and outputs of the channel. This situation is modeled as a triple $(X,Y,p(y \mid x))$, where X (resp. Y) is a random variable taking values in $\mathscr{X}$ (resp. $\mathscr{Y}$), the set of input (resp. output) values of the channel, and $p(y \mid x)$ is a conditional probability law that associates to each input of $x \in \mathscr{X}$ the probability to obtain $y \in \mathscr{Y}$ as an output. This probability is usually referred to as the transition probability of the channel. Communication channels are usually represented as in Figure 2-b: the input symbols of the channel are placed on the left of the picture, the output symbols on the right, and the probability to obtain some $y \in \mathscr{Y}$ after sending a $x \in \mathscr{X}$ in the channel is depicted by an arrow from x to y, labeled by the conditional probability $p(y \mid x)$. The channel represented on Figure 2-b has input alphabet $\{a,b,c\}$ and output alphabet $\{0,1,2\}$. The probability to obtain 0 as an output of the channel after sending symbol a is $2/3$.

The *capacity* of a channel is the average mutual information per use of the channel. It is defined as $C = \lim_{n\to\infty} \max_{p(x^n)} \frac{1}{n} I(X^n;Y^n)$ where X^n are the n possible inputs sent over the channel and Y^n are the n consecutive symbols received. The maximization is performed over all probability mass functions of the input sequence. Note that one can transfer *less than one bit* of information at each use of the channel. Note also that this capacity formula involves a maximization which might not be feasible in practice. When a discrete channel is *memoryless*, the capacity reduces to $C = \max_{p(x)} I(X;Y)$. Such a closed form is called a "single letter characterization" of the channel capacity. It is simple to compute, as it only involves a maximization over a set of probability mass functions of a single discrete random variable. However, single letter characterizations do not necessarily exist for all kinds of channels. Let us recall at this point that a channel is seen as a fixed rate use of a communication medium, and that every use of the channel is performed within a time period T. Hence, time elapsed between two consecutive uses of the channel is not considered within this setting as carrying information.

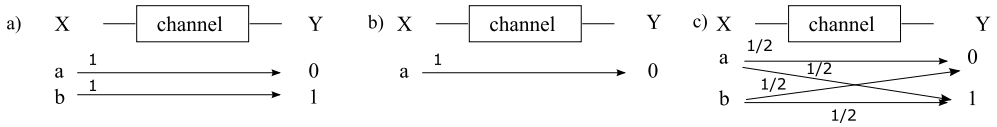


Figure 3: Some well known discrete channels

Let us now review some useful discrete memoryless channels. First consider the *perfect channel*, depicted in Figure 3.a). This channel is noiseless and establishes a one-to-one correspondence between inputs and outputs. Therefore its capacity is $\log_2(|\{a,b\}|) = 1$, where $|\mathscr{X}|$ is the size of the set $\mathscr{X}$ i.e. one can transfer one bit of information at each use of the channel. In a more general way, the capacity of a channel of this kind that associates a unique output to every input with probability 1 is $\log_2(|\mathscr{X}|)$. Similarly, there are zero-capacity channels. Consider for instance the channel of Figure 3-b. The capacity of this channel is $C = -1.\log_2(1) = 0$. This is not surprising, as there is no possibility to encode two distinct values. The capacity of the channel of Figure 3-c is also zero, as $C = \max_{p(x)} I(X;Y) = \max_{p(x)} H(Y) - H(Y \mid X)$ and for every distribution over $\mathscr{X}$, $H(Y \mid X) = H(Y)$. A similar property holds for an arbitrary number of inputs and outputs for channels where $p(y \mid X = x)$ is uniform for every input $x \in \mathscr{X}$.

In this paper, we want to test if a transition system contains a communication channel. By definition,

the possible actions taken by a process (i.e. the inputs of the channel) depend on the state of the system. Therefore, the communication channels we are looking for are *state channels* as defined below:

Definition 5 (State Channel) *A memoryless discrete channel with independent and identically-distributed (i.i.d.)* state *is a tuple* $K = (S, p(s), \{X_s, Y_s, p(y \mid x,s)\}_{s\in\mathscr{S}})$*, where S is random variable defined over a set* $\mathscr{S} = \{1,\ldots,h\}$*,* $p(s)$ *is the probability for the system to be in state* $s \in \mathscr{S}$*.* X_s, Y_s *are random variables respectively ranging over a set of input/output values* $\mathscr{X}_s$ *(resp.* $\mathscr{Y}_s$*) and represent the input and output of the channel. The choice of the state is statistically independent of previous states and previous input or output letters in the channel.* $p(y \mid x,s)$ *is the conditional distribution that defines the probability to get* $y \in \mathscr{Y}_s$ *as output when the channel is in state* $s \in \mathscr{S}$ *and the input* $x \in \mathscr{X}_s$ *is transmitted.*

This definition differs from the one given in [19] since here the input alphabets depend on the state. Moreover, Shannon has noticed that some additional information is frequently available at the transmitter, such as the state of the communication channel. Knowing this information can help increasing the capacity of the channel [19].

Theorem 1 *Let* $K = (S, p(s), \{X_s, Y_s, p(y \mid x,s)\}_{s\in\mathscr{S}})$ *be a memoryless discrete channel with i.i.d. states defined in Definition 5. The capacity of the channel K with side state information known causally at the transmitter (at time n, the current state* S_n *is known) is equal to the capacity* $C = \max_{p(t)} I(T;Y)$ *of the memoryless channel* $K' = (T, Y, r(y \mid t))$ *(without side information) with the output alphabet* $\mathscr{Y} = \cup_{s\in\mathscr{S}}\mathscr{Y}_s$ *and an input alphabet* $\mathscr{T} = \mathscr{X}_1 \times \cdots \times \mathscr{X}_{|\mathscr{S}|}$*. The transition probabilities of the channel are given by:* $r(y \mid t) = \sum_{s\in\mathscr{S}} p(s).p(y \mid t(s),s)$*, where* $t(s)$ *stands for the* s^{th} *component in the vector* t*.*

The proof follows the same lines as in [19]. The difference is that in [19] each t is a particular function from the state alphabet $\mathscr{S}$ to the input alphabet of the original channel $\mathscr{X}$ whereas here t is a vector that spans the set $\mathscr{X}_1 \times \cdots \times \mathscr{X}_{|\mathscr{S}|}$.

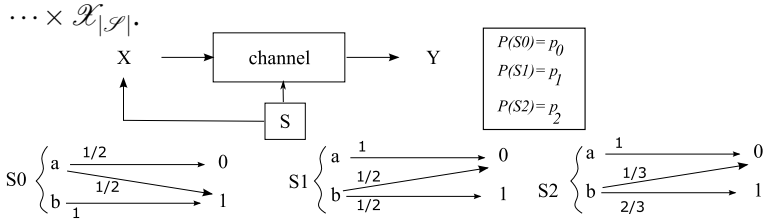


Figure 4: A memoryless discrete channel with side state information

5 More differences between interference and covert flows

As shown in previous sections, interference occurs when a process can learn some hidden information from his observation of the system, but does not always characterize covert flows of information. In section 7, we show a characterization that highlights presence of a covert flow while measuring its capacity. It is important to note two important facts on covert channels: first, efficient coding techniques do not impose to send at least one bit of information per use of the channel. Second, the notion of mutual information between processes behaviors (i.e observable sequences of actions) does not necessarily characterize a *deliberate* information flow. Hence, one must be able to differentiate between coincidence and covert flows. In the sequel, we first show that the intuitive extension of interference to a notion of *iterated interference* with several distinguishable values (as proposed in [9]) fails in general to capture covert flows with capacity < 1. We then show that mutual information measurement between processes (as proposed in [14]) does not capture either the idea of a covert protocol between users of a covert channel, and may then consider as a covert flow a situation where two processes observe the same phenomena.

5.1 Discrete covert channel

As shown in section 3, interferences capture information leaks in systems, but does not always mean the existence of a covert channels. What is missing in interference is the possibility to iterate a covert transmission (as in systems $S1$ and $S2$ of Figure 1), but also the ability to vary the symbols (i.e. send a bit 0 or 1) transmitted at each interference (this is the case for system $S3$ in Figure 1). An immediate idea is then to define a new notion of iterated interference, with the possibility to "transmit" at least two distinct values at each interference, as in [10, 9]. In the rest of this section, we adapt the definition of covert channels given in [9] to transition systems, and show that it may still miss some obvious channels.

Definition 6 *Let S be a transition system, q be a state of S, and $t = (q,a,q')$ be an outgoing transition of q. The* language of S from q after t *is the language $L_{q,t} = \{a.w \mid w \in \mathscr{L}(S_{q'})\}$ where $S_{q'}$ is a copy of S with initial state q'. Let u and v be two users of the system. A state q of S is called an* encoding state *for process u iff there exists two distinct transitions $t_1 = (q,a_1,s_1)$ and $t_2 = (q,a_2,t_2)$ outgoing from q and such that $Ex(a_1) = Ex(a_2) = u$, $(\Pi_v(L_{q,t_1}) \cup \varepsilon) \cap \Pi_v(L_{q,t_2}) = \emptyset$ and $(\Pi_v(L_{q,t_2}) \cup \varepsilon) \cap \Pi_v(L_{q,t_1}) = \emptyset$.*

More intuitively, an encoding state allows u to execute two distinct actions, which consequences are disjoint and necessarily observable by v. Note however that discovering an encoding state is not sufficient to characterize a covert flow of information. One also needs to be able to use this kind of state (not necessarily the same at each use of the channel) an arbitrary number of times. Hence, establishing a covert channel also supposes that the sending process has means to control the system in such a way that it necessarily gets back to some encoding state. This is captured by the notion of *strategy*

Definition 7 *A* strategy *for a user u of a system S is a partial function $f_u : (Q \times \Sigma \times Q)^* \longrightarrow 2^{\longrightarrow}$ that associates to every sequence of transitions $t_1.t_2 \ldots t_k$ of S a subset of fireable transitions from the state reached after t_k.*

In the sequel, we will only consider positional strategies, i.e. functions that only depend on the final state reached after a sequence of transitions, and we will denote by $f(q)$ the set of transitions allowed by f from state q. A transition $t = (q,a,q')$ *conforms* to f iff $f(q)$ is not defined or if $t \in f(q)$. We will denote by $S_{|f}$ the restriction of S to transitions that conform to f.

Definition 8 *A system S contains a* discrete channel *from u to v if and only if there exists two strategies f_u and f_v and a set of states Q_{enc} such that all states $q \in Q_{enc}$ are encoding states for process u in $S_{|f_u \cup f_v}$, and f_u and f_v allow passing infinitely often through a nonempty subset of states of Q_{enc}.*

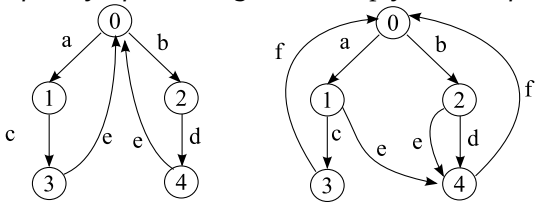


Figure 5: Two systems with and without discrete channel

Obviously, systems of Figure 1 do not contain discrete channels. Now, let us consider the leftmost example of Figure 5, which is composed of two users u and v and in which actions a and b are executed and observed by u and all other actions are executed and observed by v. This system contains a discrete channel: the strategies f_u and f_v that allow all transitions in all states ensure that the system can pass infinitely often through the encoding state q_0. State q_0 is an encoding state as there are two transitions $t_1 = (q_0,a,q_1)$ and $t_2 = (q_0,a,q_2)$ fireable from q_0, and such that $(\Pi_v(L_{q_0,t_1}) \cup \varepsilon) \cap \Pi_v(L_{q_0,t_2}) = \emptyset$ and $(\Pi_v(L_{q_0,t_2}) \cup \varepsilon) \cap \Pi_v(L_{q_0,t_1}) = \emptyset$. Hence, the consequences of firing t_1 or t_2 that can be observed by v do

not contain the empty word, and are disjoint. It then suffices for instance to perform action a from state 0 to pass a bit 0 and to perform action b to pass bit 1 from u to v. Note that our definition of discrete channel makes no supposition on the code established by the two corrupted users, but only ensures that at least two distinct choices of v have distinct observable consequences for v.

Let us consider the rightmost system of Figure 5, comporting three agents u, v and r, and such that actions a and b are executed and observed by u, actions c, d and e executed by r and observed by r and v, and action f is executed and observed by v. This system does not contain a discrete channel from u to v, as $\Pi_v(L_{q_0,t_1}) \cap \Pi_v(L_{q_0,t_1}) = e.f((c+e+d).f)^*$. If we consider definition 8, we can notice that the kind of covert channel considered by the definition uses the control flow of a system to transfer information. However, this approach only detects control flow channels in which more than one bit of information is transferred at each use of the channel. Efficient communication techniques allow communications over channels event when a fragment of bit is sent at each use of the channel, and covert channel can clearly benefit from efficient coding and decoding schemes. Consider for instance the example of Figure 6-a. Actions x, y and z are executed and observed by an agent u, actions a, b and c are executed by an agent r and observed by r and v, and last actions d and e are executed and observed by agent v. According to definition 8, this system contains no discrete channel from u to v, because for any choice of a transition t_i by u from state q_0, there exists another choice t_j such that the observable consequences of t_i on agent v are not disjoint from the consequences of t_j. Now, let us consider more precisely the sequences of actions that can be seen by agent v. When v observes a word $a.d.e$, he can not know whether the former choice of u was x or y, but he knows for sure that this choice was not z. This situation is similar for every word observed by v between two choices of u. Hence, at each use of this system, agent u can send "not x", "not y", or "not z" that is three distinct values to agent v, even if the system does not contain discrete channels in the sense of definition 8.

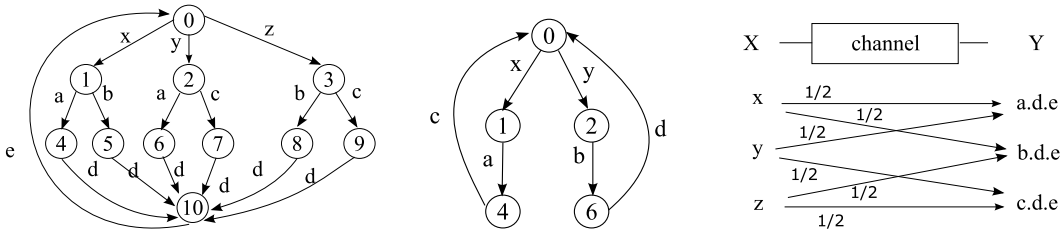


Figure 6: a) A covert channel free system? b) No covert channel c) The equivalent channel of Fig. 6-a

The system of Figure 6-a can be seen as a simple communication channel, in which user u sends symbols from the input alphabet $\mathscr{X} = \{x, y, z\}$ and user v receives symbols from the output alphabet $\mathscr{Y} = \{a.d.e, b.d.e, c.d.e\}$. This communication channel is represented in Figure 6-c, and its capacity is 0.5849 bit per use of the covert flow (details are omitted here, but can be found in the extended version). The translation from the transition system Figure 6-a to the channel Figure 6-c is straightforward, and the non null capacity shows the presence of a covert flow in the system. Remark that this channel passes less than one bit of information from u to v at each use, and hence is not captured by definition 8.

5.2 First use of information theory to discover information leaks

This paper is not the first attempt to use information theory to discover information leaks. Millen [14] considers a machine model that accepts inputs from agents and produces outputs. Inputs are chosen by several users from an alphabet I, and after a sequence of inputs $w \in I^*$, every agent u in the system can observe some outputs, denoted $Y_u(w)$. If we denote by $X_u(w)$ the sequence of inputs performed by agent u during input sequence w, and by $\overline{\pi}_{X_u}(w)$ the projection of w on inputs performed by agents other than u, then a non-interference property between u and v can be written as $\forall w \in I^*, Y_v(w) = Y_v(\overline{\pi}_{X_u}(w))$. Millen

then shows that if u and v do not interfere, then if all inputs of users other than u are independent from X_u, the mutual information $I(X_u;Y_v)$ between X_u and Y_v should be null. Note that this is only an implication, and that there might be cases where $I(X_u;Y_v)=0$, but nevertheless u interferes with v. The converse property is more interesting: a non null mutual information between X_u and Y_v denotes an interference. The mutual information $I(X_u;Y_v)$ is computed over input sequences performed by u between two outputs to v.

The machine model is not explicitly given in this approach: it is a black box that receives a *finite* sequence of inputs from all users and produces a *finite* sequence of outputs. Within the context of transition systems, the inputs X_u of agent u are the actions that u can execute ($Ex^{-1}(u)$), and the outputs to v the actions that v can observe. Note however that the independence hypothesis between inputs of the system does not necessarily hold, and furthermore that nothing guarantees that an output to v happens after a *finite number* of inputs from u, nor that outputs are finite. Hence, computing $I(X_u;Y_v)$ means computing the mutual information between sets of inputs/outputs of arbitrary size. Without giving any hint on how to compute this value, Millen's characterization of information leaks can be rewritten as follows for transition systems. The average information leak (per transition) between two agents u and v in system S is $Leak(u,v)=\lim\limits_{n\to\infty}\ \frac{1}{n}.\max\limits_{P(\mathscr{L}^n(S))}\ I\left(\Pi_{Ex^{-1}(u)}(\mathscr{L}^n(S));\Pi_v(\mathscr{L}^n(S))\right)$, where $\mathscr{L}^n(S)$ is the set of words in $\mathscr{L}(S)$ of length n. If $Leak(u,v)>0$, then there is an interference from u to v. However, this average mutual information defines a correlation between actions of u and observations of v, which can be called average interference, but *is not* a characterization of a covert channel. Indeed, we can show on a very simple example that non-zero value for $Leak(u,v)$ implies an average interference greater than zero, but not a deliberate information flow between two users. Consider for instance the system of Figure 6-b. Let a,b be the actions executed and observed by u, x,y the actions executed and observed by r and c,d the actions executed and observed by v. In this example, successive actions and observations of processes r,u and v are independent. If we suppose that choices of process r are equiprobable, then the average information leak is $1/3$ bits at each transition (the details of the calculus are provided in the extended version). This means that user v infers an average number of $1/3$ bits on u's behavior at each transition of the system. There is clearly an interference in the system, as user v can almost infer the exact behaviors of users u (up to the last a or b) from his own observations. Furthermore, such kind of interference can occur an arbitrary number of times. Let us recall at this point a major difference between covert channels between two agents u,v and interference between u and v. When an interference occurs, process v has learned some information about u's actions or states. In a covert channel, u **decides** to send some hidden information to v, and both processes have agreed on a protocol to convey this information. Hence, we can not consider that the system of Figure 6-b contains a covert channel from u to v (nor the converse direction), as the decision to execute a or b is not a choice from process u, but a consequence of a choice of process r (that chooses between action x and y). However, a covert channel exists from r to u and from r to v, as it suffices for process r to play action x or y to allow distinct observations on u and on v. The covert channel characterization introduced in section 7 shows a dependency between the *decisions of a sending process*, that is actions performed from states in which this process has more than one fireable action, and *observable consequences on a receiving process*. However, even reformulated this way, nothing guarantees that the capacity of such covert channels can be expressed as a closed formula.

Several other works have used information theory to compute the bandwidth of already identified covert flows. I. Moskowitz shows in [15] how a very simple information aggregation system can be perverted to create a covert channel between two users, and computes its capacity. We will adapt this example in section 6 to illustrate our characterization of covert channels. Several other works have studied and quantified covert flows in TCP/IP [1, 16]. Some channels simply consist in filling useless

bits in TCP frames to hide information (this security leak called TCP piggybacking has been corrected ever since). More elaborate coding strategies modulate the use of packets in time windows to pass information (the shape of the function denoting the cumulated number of messages encodes the input symbols in a covert channel) [7]. I. Moskowitz remarks that adding non-deterministic delays for the transmission of packets reduces the capacity of timing covert channels, and has proposed a mechanism called the "network pump". The pump stores sent packets for a random time, and then delivers them to their destination [11]. This mechanisms does not close covert flows, but reduces their capacity, as demonstrated by [7]. However, it also impose a time penalty to honest users of the system.

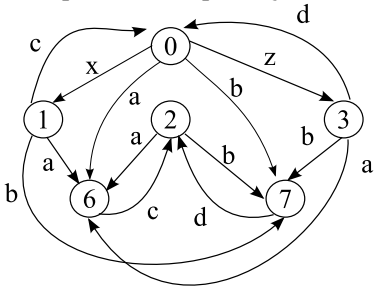


Figure 7: A system that is not interferent, but contains a covert channel

At this point, let us summarize the mentionned differences between interference and presence of covert flows in systems. Systems that contain a leak but do not allow iteration of leaks, or that leak the same information from an agent to another an arbitrary number of times do not contain covert flows, but are interferent. A generalization of interference to iterated multivalued interference (definition 8 in this paper) fails to characterize covert flows of information with capacity lowr than 1. When interference is characterized as a non-null mutual information between actions of an agent and observations of another (as in [14]), some systems might be found interferent, bu yet do not contain covert flows. Last, consider the example of Figure 7, where actions a, b are executed and observed by agent u, actions c,d executed and observed by agent v, and actions x, y are executed and observed by agent r. In this system, u does not interfere with v (both if we consider SNNI or BSNNI), but this system contains a cover channel, as all a's are followed by a c, and all b's are followed by a d. We do not know if such situation ecapes a characterization by all notions of interference, but so far, it seems that although interference and covert flow presence look very similar, they are indeed *orthogonal properties* of systems.

6 An example: the acknowledgment channel

We have shown in previous sections that a characterization of covert channels should 1) highlight dependencies between iterated and deliberate choices of one sending process on one side, and the observed consequences of these choices on another process, but also that 2) each occurrence of these choices can allow the transfer of less than one bit of information. Language theory is well suited to deal with 1) while information theory is well adapted to deal with 2). Section 7 reconciles both parts of the problem, and brings covert channel discovery back to the computation of a capacity for channels with side state information.

Let us illustrate our approach with a toy example inspired from [15]. Figure 8 describes an information aggregation system. Several agents called *collectors* collect information that are then sent to another agent called the *central observer*. The role of the central observer is to analyze the data provided by observers, and to give an overall expertise (statistics, computation of mean values, ...). Data are sent at regular rate from each collector to the central, using data messages of type m. The ascending communication path is not reliable, and can lose messages, with probability p. The descending path is reliable,

and the central sends acknowledgment messages to its collectors to indicate whether it received or not the last message. Message losses are detected using timeouts. When a message is successfully received, the central sends an *Ack* message to the collector. In case a timer expires, the central sends a *Nack* message with the expected packet number to the collector. Collector wait for the answer from the central before sending a new packet, that is at a given time, there is at most one data packet transiting from each collector to the central.

In addition to this simple mechanism, the system has strict security rules: collectors have no means to communicate, and the central can not communicate with collectors. The main reason is that the individual data collected locally has almost no value, but that the global data obtained by aggregation or the diagnosis computed should be protected. For this reason, communications from the central to the collectors are observed, and a monitor filters all messages that are not $Ack/Nacks$. In addition to this, the central observer is frequently audited, and should have received all acknowledged packets. An immediate idea to establish a covert channel is to use *Ack* and *Nack* messages to transfer 0 or 1 from the central to a chosen collector. However, the central can not declare as received a lost packet, as this will be discovered during the audit. Nevertheless, when a packet is received, it can be falsely declared as lost, and a useless retransmission is done by the collector.

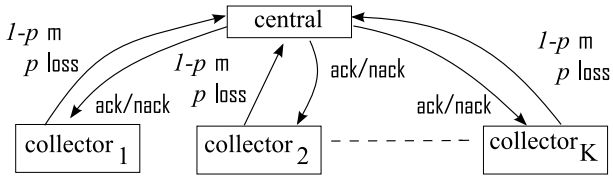


Figure 8: An information aggregation system.

Figure 9-a depicts the normal behavior of a pair collector/central, i.e. when none of them have been corrupted to establish a covert information flow. The lossy communication medium has been represented as an additional process, that simply transfers *Ack* and *Nack* messages, and transfers or looses *m* messages along the ascending way. To depict this system, we will consider three processes: *ct*, the central observer, *med*, the communication medium, and *co* the collector. Actions of the system are of the form $p!x$, $p?x$, $p\ loss$, $p\ set$ and $p\ tout$, denoting respectively the sending of a message of type x by a process p, the reception of a message of type x by a process p, the loss of a message, a timer setting or a timeout. In addition to the transitions, we add to the model a function P_S that associates probability p to the loss of message m. We hence have $P_S(2, med!m, 3) = (1-p)$ and $P_S(2, medloss, 4) = p$. This system can not be used to transfer information from CT to CO, as CT just copies the choices of the communication medium.

Figure 9-b shows a slightly modified system, where the central observer can cheat, and declare lost a packet it has received. With this new protocol, when the communication medium does not loose the transmitted packet (this occurs with probability $1-p$), the central observer can pass without loss or noise a bit of information (sending *ack*/*nack* eventually leads to a reception of the acknowledgment message) to the collector. This is a perfect channel. Conversely, when a packet is lost (this occurs with probability p, the central observer can only send a *nack* packet, that will eventually be received by the collector. This is a zero capacity channel. If we look at this system from the covert channel point of view, it is a state channel, where the state only depends on the medium (and not on the inputs/outputs (*ack*/*nack*)) and is iid. Moreover, the central knows causally in which state the system is. We can then apply the technique of [19] described in Thm 1 to compute the capacity $C_{Mosk}(CT, CO)$ of this channel.

Figure 10-a shows our two-state channel with side information. According to the state in which the system is, CT and CO communicate either through a perfect or zero-capacity channel. When in state $S5$, the input symbols used by CT are the sending of *ack*/*nack* messages, i.e. $X_{S=S5} \in \{!ack, !nack)\}$ and

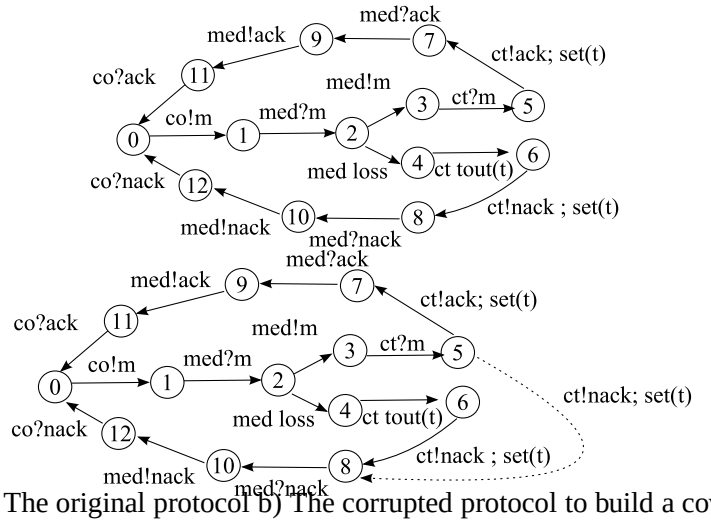
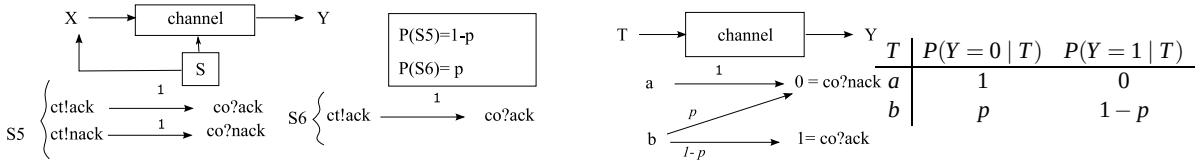


Figure 9: a) The original protocol b) The corrupted protocol to build a covert channel



T	$P(Y=0 \mid T)$	$P(Y=1 \mid T)$
a	1	0
b	p	$1-p$

Figure 10: a) A state channel for the flow from CT to CO b) A stateless channel with identical capacity

when in state $S6$, $X_{S=S6} \in \{!ack\}$. Whatever the state is, the output symbols read by CO are the receptions of *ack*/*nack*, $Y \in \{1 = ?ack, 0 = ?nack\}$. From Thm 1, the state channel (see Figure 10-a) is equivalent to a stateless channel depicted in Figure 10-b. Its input is a 2-length vector $T = (X_{S=S5}, X_{S=S6})$ that can take 2 values: $T \in \{a = (!nack, !nack), b = (!ack, !nack)\}$ and the output alphabet is unchanged. The transition probabilities (see table in Figure 10-b) are those of the so-called Z-channel [4]. Therefore the capacity of the equivalent stateless channel is $C_{Mosk}(CT,CO) = \log_2(1+2^{-s(p)})$, where $s(p) = \frac{-p\log p-(1-p)\log(1-p)}{1-p}$. It is interesting to compare this capacity $C_{Mosk}(CT,CO)$ with the case with perfect state knowledge at the collector as well. With perfect state knowledge at both central and collector, the capacity is $1-p$ since 1 bit of information is sent each time the channel is in state $S5$, which occurs with probability $1-p$. In our example, only the transmitter knows the state and the capacity $C_{Mosk}(CT,CO) \leq 1-p$.

7 IT based characterization of covert flows

We have shown in previous sections that the terms interference and covert channels refer to different kinds of leaks, and orthogonal properties of systems. Similarly, discrete channels characterization misses some obvious channels with capacity lower than 1. This section proposes an information theoretic characterization of covert flows. The main idea is to consider that a hidden channel exists from user u to user v in a system S if u and v can use S to simulate a memoryless communication channel with state with capacity greater than 0. We define this characterization in two steps: we first define some transition systems that simulate a communication channel with state. These systems will be called *Half-Duplex* systems. We then define control flows covert channels as the capacity for a pair of users u,v to change their interactions with the rest of the system, i.e. transform a system S into a new system S' in such a way that S' is a Half-duplex system, and is observationnaly equivalent to S for all other agents. Then, computing the capacity of such covert flow resumes to computing the capacity of the communication channel simulated by S'.

Definition 9 *Let $S = (Q, \longrightarrow, \Sigma, q_0)$ be a transition system. A state $q \in Q$ is* controlled *by process $p \in \mathscr{P}$*

iff for any transition (q,a,q'), $Ex(a) = p$. *The system S is in* Half-Duplex *between two processes u and v iff there exists a state x (called the* control state*), a set of states* $EN = q_1,\ldots,q_n$ *controlled by u (called the* encoding states*) and two bounds* K_1, K_2 *such that:*

- *any path originating from x of length greater or equal to* K_1 *passes through one state of* EN,
- *any path originating from a state* $q_i \in EN$ *of length greater or equal to* K_2 *passes through x and contains at least one transition labeled by an action in* $Obs^{-1}(v)$,
- *no path from x to one of the states in* EN *uses transitions labeled by* $Ex^{-1}(u)$.

For simplicity, we will also assume that $Y = \Pi_{Obs^{-1}(v)}(\mathscr{L}(Path(x,x)))$ *forms a code, i.e. any word w in* Y^* *has a unique factorization* $w = Y_1.Y_2\ldots Y_k$.

More intuitively, the states of EN are states from which the sending process u in a covert channel will encode information. Passing from state x to a state $q_i \in EN$ simulates a choice of a new channel state, and is an essential condition to allow for a translation of transition systems into state channels. This does not mean however that the capacity of hidden flows can not be computed in transition systems that do not meet this condition. The additional condition that all states in EN are controlled by u is not a real constraint. In fact, we can easily ensure that all states are controlled by a single process by adding an additional actor to the system that schedules the next process allowed to move. Clearly, Half-duplex channels simulate the behavior of communication channels with side state information, and the states of this channel will be the encoding states of the half-duplex system. Chosing an action (or a sequence of actions) of u from an encoding state q_i simulates the sending of an input symbol in the communication channel with state, and the observation performed by v before returning to state x simulates the reception of an output symbol. With the constraints imposed by the definition, we can ensure that the choice of an encoding state is an i.i.d variable, that is independent from actions of u and v. The code assumption means that the receiving process in the covert flow also knows how many times the channel was used. This hypothesis can be easily relaxed, but we will use it in the rest of the paper as it simplifies translation to a communication channel model. We refer interested readers to the extended version of this work for a more general framework without this code assumption.

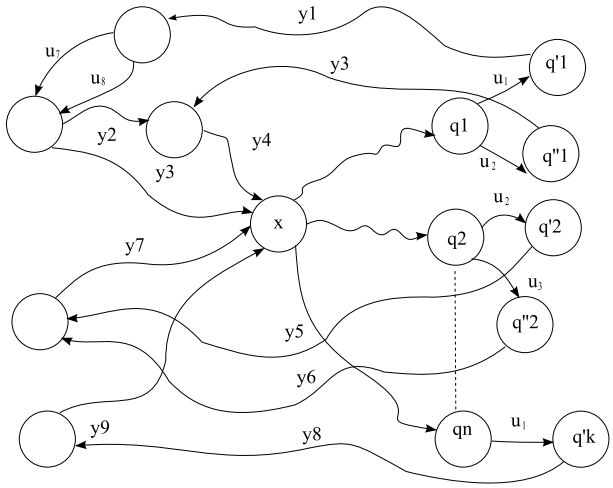


Figure 11: General shape of a Half-duplex transition system

Figure 11 shows the general shape of half-duplex systems. There is a single concentrator state x. Unlabelled arrows originating from x denote a sequence of transitions that is unobservable for v. Arrows labeled by u_i denote an action of process u. Arrows labeled by y_i denote sequences of transitions for which word y_i is observed by process v. Half-duplex transition systems simulate the behavior of some

communication channel with state. We can easily build the simulated channel, and then compute the capacity of the obtained model to quantify the importance of a leak.

Definition 10 *Let $S = (Q, \longrightarrow, \Sigma, q_0)$ be a Half-duplex transition system from u to v, with encoding states EN, control state x, and probability function P_S. The* state channel from *u* to *v contained in S is the state channel with side information $Ch_{S,u,v} = (S, X, Y, g_s, \{p_s(y \mid x)\}_{s \in \mathscr{S}})$, where:*

- *S is a random variable over a set of states $\mathscr{S} = EN$*
- *$g_{q_i} = \Sigma_{\rho \in Path(x,q_i)} P_S(\rho)$ is the probability to reach state q_i from state x.*
- *X is a random variable defined over $\mathscr{X} = \bigcup_{q_i \in EN} \Pi_{Ex^{-1}(u)}(\mathscr{L}(Path(q_i, x)))$, the set of sequences of actions executed by process u to move from a state $q_i \in EN$ to state x.*
- *Y is a random variable defined over $\mathscr{Y} = \bigcup_{q_i \in EN} \Pi_{Obs^{-1}(v)}(\mathscr{L}(Path(q_i, x)))$, the set of sequences of actions observed by process v to go from state q_i to state x.*
- *$p_{q_i}(y \mid w) = \sum_{\rho \in Path(q_i,x), \Pi_u(\mathscr{L}(\rho)) = w, \Pi_v(\mathscr{L}(\rho)) = y} P_S(\rho)$*

The translation from a half duplex transition system to a state channel immediately gives a capacity of information flows from *u* to *v*. Note however that this implicitly means that the sending process in the covert channel must have perfect knowledge of the system's state to achieve this capacity. When process *u* does not know perfectly the state of the system, the capacity of the state channel computed from *S* should only be seen as an upper bound of an achievable capacity.

The capacity $C_{S,u,v}$ of the state channel computed from a half duplex transition system *S* gives the average mutual information between sequences of choices executed by *u* and observations of process *v* between two occurrences of state *x*. Non-zero capacity of such information flow from *u* to *v* in a Half duplex system then highlights a capacity to transfer information from *u* to *v* using system *S*. Note however, that a capacity is an average number of bits *per use* of the channel, and does not give the bandwidth of the channel, which is an average number of bits transferred *per time unit*. Usually, capacity and bandwidth are tightly connected for communication channels, as it is frequently assumed that channels are used at a constant rate T. However, in our covert flow setting, each channel use may have a distinct duration. We do not yet know whether there exists a closed form or single letter characterization for the bandwidth of a covert channel. However, non-zero capacity means non-zero bandwidth, and capacity is still relevant to characterize covert flows.

Now that Half-duplex systems are defined, we can propose a characterization for some covert channels. The main intuition behind the following definition is that a covert flow exists when two users u, v can modify their behavior in such a way that the resulting system is observationnally equivalent for all other agents, but simulates a communication channel with state from *u* to *v*, with capacity greater than 0.

Definition 11 *A transition system S contains a* control-flow covert channel *from u to v if communications from u to v are not allowed by the security policy, and there exists a Half-duplex transition system S' (from u to v) such that:*

i) $\pi_{\mathscr{P} \setminus u,v}(S') \equiv \pi_{\mathscr{P} \setminus u,v}(S)$,
ii) $\pi_u(S')$ *and* $\pi_v(S')$ *are defined over the same sets of messages than* $\pi_u(S)$ *and* $\pi_v(S)$.
iii) $C_{S',u,v} > 0$

The system S' is obtained by replacing processes *u* and *v* in *S* by corrupted processes *u*' and *v*' that implement a covert channel. The half-duplex requirement ensures that a capacity for a supposed covert flow from *u*' to *v*' can be effectively computed. Item *i*) means that the corrupted system must be equivalent from the non-corrupted users point of view. This means in particular that only processes *u* and

v can be changed. Item ii) means that processes u' and v' must not implement direct communications from u' to v', which would be detected by the mechanisms enforcing the security policy. In addition to this, when direct and uncensored communications from u to v exist, establishing a covert channel makes no sense. The last item iii) means that u' and v' implement a channel with non null capacity, i.e a covert channel.

One may notice that there exists an infinite number of candidate processes to replace u and v while satisfying conditions i) and ii). We can however restrict arbitrarily our search to systems that are equivalent up to a bounded size. One can immediately remark that when a system S is seen as the composition of independent processes that communicate via channels, i.e. $S = S_u||S_1||\dots||S_k||S_v$, S' is obtained by replacing S_u and S_v by some variants S'_u and S'_v that compose similarly with the rest of the system (the sequences of observed messages transiting between u,v and the rest of the system are similar). It might be useless to test **all** models up to a certain size, and we think we can limit the search to a finite set of canonical models in which processes u and v have the same number of states but more transitions, and behave as expected by the rest of the system. This remains however to be demonstrated.

Definition 11 characterizes covert flows in a situation where communications from u to v are forbidden by the security policy. However, covert flows can also appear over legal communications (this is for instance the case of TCP piggybacking). In such situations, a security policy may consist in monitoring or record all messages from u to v, and forbid illegal message or contents. Covert flow in this context are called *legitimate channels* and their purpose is to bypass the monitoring mechanism.The example of section 6 should be considered as a legitimate channel, as collectors and central observer are allowed to communicate.

Definition 12 *A transition system S contains a* legitimate covert channel *from u to v if u and v are allowed to communicate by the security policy, and there exists a Half duplex transition system S' such that $\pi_u(S')$ and $\pi_v(S')$ are defined over the same alphabets as $\pi_u(S)$ et $\pi_v(S)$, $\pi_{\mathscr{P}\setminus u,v}(S') \equiv \pi_{\mathscr{P}\setminus u,v}(S)$, and $C_{S',u,v} - C_{S,u,v} > 0$*

Definition 12 characterizes situations where modification of the behavior of two processes can add information to the legal contents exchanged when using the original system. Coming back to the example of section 6, we can notice that there exists no information flow from CT to CO in the original model, even if both processes are allowed to exchange acknowledgment messages. The communication state channel computed from this description alternates channels where only a single input/output is allowed (one state allows to send *ack*, the other allows to send *nack*), hence with zero capacity. This is not surprising, as in the original specification CT only forwards to CO a choice from the environment that is independent from its own actions. Note however that the initial capacity of information flows between two processes that can establish a covert channel is not necessary null. Indeed, some communications can be allowed by the security policy. A covert channel should then be seen as a mechanism that increases the capacity of information flows between two designated parties (the designated sender and receiver processes).

So far, definitions 11 and 12 remain imprecise on the security policy. Both definitions could require that S' complies with the security policy, but this means expressing this policy as a formal model. For instance, if the security policy is enforced by a monitoring process P_M that detects deviations from a normal behavior, this additional requirement is ensured by ii) $\pi_{\mathscr{P}\setminus u,v}(S') \equiv \pi_{\mathscr{P}\setminus u,v}(S)$. In the global security policy is given as a transition system S_{sec}, we simply must ensure that $\mathscr{L}(S') \subseteq \mathscr{L}(S_{sec})$.

8 Conclusion

We have shown how to characterize some covert channels using transition systems and information theory. This characterization shows that a chosen pair of users can simulate a communication channel.

We then translate this transition system to a finite state channel model, for which we have effective means to compute a capacity. A capacity greater than 0 means that the chosen users can establish a covert flow. This characterization works even for covert information flows with capacity lower than 1 bit, but is restricted to a class of transition systems called half-duplex systems. This restriction is mainly motivated by the obligation to compute a capacity for information flows. It imposes in particular that all inputs to the covert channels are independent from the past. However, in many systems, an input and the corresponding outputs can influence the next state. Our techniques should be extended to handle this situation. Note however that effective capacity approximation for channels with memory it is still an open question in information theory. We do not expect to obtain closed forms for capacities of covert flows, but good upper bounds can certainly be achieved, and are still useful to characterize leaks. We also have assumed that the output alphabet in a covert channel was a code. This assumption was mainly written to simplify the description of our translation from transition systems to communication channels. It can be easily removed at the cost of an approximation, as uncertainty in factorization of received outputs can be seen as adding some randomness in a channel. Similarly, our channel model supposes that the sender has perfect information on the state of the system. This is not always the case in real systems, but capacities achieved with imperfect information are necessarily lower than with perfect information, so this assumption causes no harm to the proposed characterization.

Our covert channel characterization defines a covert flow as the possibility to corrupt a system in an unobservable way. This definition does not bring an effective algorithm to check for covert channels, as there might be an infinite number of such variants of a system. Current solution is to bound the memory of attackers, and work with variant models of size up to this limit. We think however that the search can be limited to a set of canonical variants, as creating new states in the corrupted processes simply means unfolding the original processes in a way that is equivalent to the original observation. In a memoryless system, this does not seem to bring any more encoding power to attackers than allowing more behaviors with the same number of states. This however has to be demonstrated.

Last, notice that the proposed characterization only deals with a specific kind of covert flow, and is defined for transition systems. It does not consider time, and exhibits a memoryless communication channel with state between two users. We do not know yet if this approach generalizes to wider classes of models, covert flows, and communication channels. Of course, we do not expect to be able to characterize any kind of covert flow using this kind of technique, and the approach proposed in this paper should be seen as complementary of other security tools such as non-interference. However, we think that such characterization captures the essence of covert channels, that is the capacity for a pair of users to establish a communication.

References

[1] K. Ahsan & D. Kundur (2002): *Practical Data Hiding in TCP/IP*. In: *Workshop on Multimedia Security at ACM Multimedia '02.*

[2] D.E Bell & J.J. La Padula (1973): *Secure Computer Systems: a mathematical model*. MITRE technical report 2547, MITRE. Vol II.

[3] D.E. Bell & J.J. La Padula (1973): *Secure Computer Systems: mathematical foundations*. MITRE Technical report 2547, MITRE. Vol I.

[4] T.M. Cover & J. A. Thomas (1991): *Elements of Information Theory*. Wiley.

[5] Common Criteria (1999): *Common Criteria for Information Technology Security Evaluation Part 3: Security assurance requirements*. Technical Report CCIMB-99-033, CCIMB.

[6] R. Focardi & R. Gorrieri (2000): *Classification of Security Properties (Part I: Information Flow)*. In: *FOSAD 2000*. pp. 331–396.

[7] J Giles & B. Hajek (2002): *An information-theoretic and game-theoretic study of timing channels*. *IEEE Transactions on Information Theory* 48(9), pp. 2455–2477.

[8] J.A. Goguen & J. Meseguer (1982): *Security policies and security Models*. In IEEE Computer Society Press, editor: *Proc of IEEE Symposium on Security and Privacy*. pp. 11–20.

[9] L. Hélouët, M. Zeitoun & A. Degorre (2004): *Scenarios and Covert channels, another game...* In: *Games in Design and Verification, GDV '04*. Electronic Notes in Theoretical Computer Science, Elsevier.

[10] L. Hélouët, M. Zeitoun & C. Jard (2003): *Covert channels detection in protocols using scenarios*. In: *SPV'03 Security Protocols Verification*.

[11] M.H Kang, I. Moskowitz & D.C. Lee (1996): *A Network Pump*. *IEEE Trans. Software Eng.* 22(5), pp. 329–338.

[12] R.A. Kemmerer (1983): *Shared ressources matrix methodology: an approach to indentifying storage and timing channels*. *ACM transactions on Computer systems* 1(3), pp. 256–277.

[13] B. Lampson (1973): *A note on the confinement problem*. *Communication of the ACM* 16(10), pp. 613–615.

[14] J. Millen (1987): *Covert Channel Capacity*. In: *IEEE Symposium on Security and Privacy*. pp. 60–66.

[15] I. Moskowitz & M. Kang (1994): *Covert Channels - Here to stay ?* In: *COMPASS'94*. IEEE Press, pp. 235–243.

[16] S.J Murdoch & S. Lewis (2005): *Embedding Covert Channels into TCP/IP*. In: *Information Hiding*. pp. 247–261.

[17] NSA/NCSC (1993): *A guide to Understanding Covert Channel Analysis of Trusted Systems*. Technical Report, NSA/NCSC.

[18] A. Sabelfeld & A.C. Myers (2003): *Language-based Information-flow security*. *IEEE Journal on selected areas in communications* 21(1).

[19] C.E. Shannon (1958): *Channels with Side Information at the Transmitter*. *IBM Journal of Research and Development* 2(4), pp. 289–293.