Journal of Universal Computer Science, vol. 6, no. 11 (2000), 1054-1070
submitted: 8/6/00, accepted: 19/9/00, appeared: 28/11/00

Nondeterministic Admissible Interference

John Mullins
(École Polytechnique de Montréal, Canada
John.Mullins@polymtl.ca)

Abstract: In this article we address the issue of confidentiality of information in the context of downgrading systems i.e. systems admitting information flow between secrecy levels only through a downgrader. Inspired by the intuition underlying the usual definition of admissible information flow, we propose an analogue based on trace equivalence as developed in the context of concurrency theory and on a modification of the usual definition of purge function. We also provide unwinding conditions to guarantee a consistent and complete proof method in terms of communicating transition systems. We take advantage of this framework to investigate its compositionality issues w.r.t. the main operators over communicating transition systems. We conclude the article with a short presentation of this work's most promising aspects in the perspective of future developments.

Key Words: Specification techniques, program verification, security models, information flow properties, confidentiality, noninterference, intransitive noninterference.
Category: D.2.1, D.2.4, F.3.1.

1 Introduction

Covert channel analysis refers to a validation process of a system based on a multi-level security (MLS) architecture. In such a system, a relation between the users setting out the admissible information flow (generally from low to high level) and the inadmissible one (generally from high to low) is defined. The information flow approach consists in "integrating" the covert channel analysis i.e. the detection of prohibited information flow, to the specification level by requiring some special purpose *information flow properties* to be satisfied by the system. *Noninterference* and its generalisations are such properties [McLean 94, Goguen, Meseguer 82, Sutherland 86, McCullough 87, McCullough 88]. A system satisfies noninterference if there is no information flow from a high level user to a low level one.

The problem is that absence of information flow is rarely very interesting as a description of confidentiality, as many practical applications are intended to preserve confidentiality, but nonetheless leak information as illustrated by the following simple scenario.

An agent originating from a merchant is executed by some customer at his local site. The agent requests some information (name, address,...) which is less sensitive, and some information (account no.) which should definitely be treated in a confidential manner. The details of the protocol used are not relevant to the illustration. Some secure electronic payment protocol implements credit card-based transactions between the customer and the merchant using financial network for clearing and authorisation e.g. 1KP [Bellare et al. 95]. Even though it assumes communication between the customer and the clearing centre to be secure in some way, sensitive information is clearly leaked from the customer to the merchant, in the sense that the reception of a payment authorisation from the clearing centre leaks information about the account number submitted by the customer. This leakage is real and can be used as a covert channel, but in the intended scenario it could be entirely harmless if for example the downgrading channel bandwidth deems low enough. The point is that the flow of information concerning the account number in the above example is an intended, or *admissible* one. So we are looking for a notion of information flow which can accommodate such admissible information flow while detecting any other information flow as an inadmissible one.

This problem has first been addressed in [Goguen, Meseguer 84]. To describe it, they proposed *conditional noninterference* (CNI) . The first satisfactory treatment of conditional noninterference appeared in [Haigh, Young 87]. The intuitive meaning of conditional noninterference is to disallow high level to interfere with low level unless the interference occurs through a controlled channel, i.e. through a dedicated downgrading level [Goguen, Meseguer 84] (in the above example, the channel dedicated to authorisation). A clever and complete development of channel control can be found in [Rushby 92] for the deterministic case. Channel control is formulated therein in terms of noninterference in which the interference relation $\rightsquigarrow$ over the set of security levels is intransitive. e.g. $secret \rightsquigarrow downgrader$ and $downgrader \rightsquigarrow unclassified$ but $secret \not\rightsquigarrow unclassified$.

All these definitions are generally unproblematic in the context of deterministic machines but can raise problems when applied to nondeterministic systems. Nondeterminism is inherent to concurrency, as it arises naturally from the manners in which parallel combinations of processes are scheduled. The best known attempts to accurately assess the security of nondeterministic processes are those of [Focardi, Gorrieri 95]. They use semantic models in a modified CCS which extends usual definitions of transitive noninterference on deterministic processes. We extend this approach to handle conditional information flow. More precisely, we propose a generalisation of Rushby's conditional noninterference called *admissible interference* (AI) property where noninterference here, has to be understood as Focardi and Gorrieri's *strong nondeterministic noninterference* (SNNI).

Definitions of noninterference are usually expressed in terms of *purging* and *unwinding*. Purging means removal from the history of the system all those elements (typically communications with high-level) required not to interfere with the low level. On the other hand, unwinding provides sound local conditions on individual transitions w.r.t. the purging-based definition i.e. sufficient to guarantee noninterference. We believe that process algebras like CCS provide a general framework for describing communicating systems as well as comparing and generalising the numerous definitions of confidentiality. Moreover, the use of a process algebraic framework allows us to apply a number of well-established results such as completeness of unwinding rules w.r.t the purging-based definition and compositionality.

The paper is organised as follows. Admissible interference is presented in [Section 3] in terms of purge function. The model used to define unwinding rules for admissible interference is an extended transition system model equipped with operators à la CCS [Milner 89]. This model is presented in [Section 2]. [Section 4] contains the main results of this paper: completeness of unwinding rules i.e. a characterisation of AI in terms of the transition system model and compositionality of AI w.r.t. the main operators over extended transition systems. This section ends with an example: the specification of an electronic transaction system involving a purchasing agent. Finally, in [Section 5] we discuss some interesting issues for further research.

1.1 Related Works

A number of very interesting attempts to unify various formulations of noninterference have been published recently in the context of CSP [Hoare 85], notably [Ryan, Schneider 99] and [Roscoe, Goldsmith, 99]. Their approach is quite different from ours. It is based on determinism. Roughly, the determinism-based notion of noninterference is as follows: high-level actions are turned into nondeterministic choices, and if the low-level view is deterministic, then there is no information flow from the high level to the low level. In [Ryan, Schneider 99], the authors demonstrate a correspondence between some definitions of noninterference and some notions of process equivalence and describe some ideas for generalising noninterference to handle partial and conditional information flow as well. In [Roscoe, Goldsmith, 99], the authors show that the standard definition of intransitive noninterference in terms of a purge function is undesirably permissive: it allows the effects of "too many" high-level actions to become visible in the low-level view after a downgrading action. They propose their definition based on determinism as remedy.

2 Downgrading Transition Systems

We are looking at extended *labelled transition systems* (LTS) to model processes and computations of computing entities interacting at different trust levels in the case where high-level entities are allowed to downgrade information to low-level ones but only through downgrading-level entities.

Definition 1 Labelled Transition System. A *labelled transition system* is a triple

$$T = (Act, S, R)$$

such that:

- $Act = Vis \cup \{\tau\}$ where $Vis = In \cup Out$ is an at most denumerable set of *visible actions*, In is a set of *input actions*, $Out = \overline{In}$ is a set of *output actions*, the function $\overline{[.]} : Vis \to Vis$ is such that $\overline{\overline{a}} = a$ and $\tau \notin Vis$ stands for an *internal action*.
- S is a non-empty set of *states*
- $R : Act \to \mathcal{P}(S \times S)$ such that for every $\alpha \in Act$, $R(\alpha)$ is the α-labelled *transition relation* denoted by $\stackrel{\alpha}{\to}$.

The *pointed transition system* T_s is the labelled transition system T with s as initial state.

A *computation* of $u = a_0 a_1 \dots a_n \in Act^*$ in a pointed transition system T_s is a finite string of transitions satisfying $s_0 \xrightarrow{a_0} s_1 \xrightarrow{a_1} \cdots \xrightarrow{a_n} s_{n+1}$ with $s = s_0$ ($s_0 \xrightarrow{u} s_{n+1}$ for short). A state $s'' \in S$ is *reachable* from s' if there is a computation $s' \xrightarrow{u} s''$ for some $u \in Act^*$. For the sake of simplicity we will assume that any state of T_s is reachable from s.

We note $s \xrightarrow{u} s'$ when u is the *visible trace* of a computation $s \xrightarrow{v} s'$ of v of T, i.e. when $v \in \{\tau\}^* a_0 \{\tau\}^* a_1 \ldots \{\tau\}^* a_n \{\tau\}^*$, $u = a_0 a_1 \ldots a_n$ and $a_i \in Vis$. We note $s \longrightarrow s'$ when u is the empty visible string. Let $\mathcal{L}(T_s)$ be the set of visible finite traces of T_s. We will say that T_s and $T'_{s'}$ are *trace equivalent* or simply *equivalent* (noted $T_s \simeq T'_{s'}$) if $\mathcal{L}(T_s) = \mathcal{L}(T'_{s'})$. We will sometimes use the notation $T_s \sqsubseteq T'_{s'}$ when $\mathcal{L}(T_s) \subseteq \mathcal{L}(T'_{s'})$.

For $T_s = (Act, S, R)$, $T'_{s'} = (Act, S', R')$ and $L, L' \subseteq Vis$, T_s/L is T_s where actions in L have been internalised, that is the transition system where actions in L have been turned into τ. $T_s \setminus L$ is the restriction of T_s to actions not in L that is the transition system obtained from T_s by restricting R in Def. 1 to actions not in L. $T_s | T'_{s'}$ is the concurrent composition of T_s and $T'_{s'}$ that is the transition system having $S \times S'$ as set of states, (s, s') as initial state and the α-labelled transition relations defined inductively from the following rules:

$$\text{LEFT} \quad \frac{(s_1, s_2) \xrightarrow{\alpha} (s'_1, s_2)}{s_1 \xrightarrow{\alpha} s'_1}$$

$$\text{RIGHT} \quad \frac{(s_1, s_2) \xrightarrow{\alpha} (s_1, s'_2)}{s_2 \xrightarrow{\alpha} s'_2}$$

$$\text{SYNC} \quad \frac{(s_1, s_2) \xrightarrow{\tau} (s'_1, s'_2)}{s_1 \xrightarrow{\alpha} s'_1 \quad s_2 \xrightarrow{\alpha} s'_2}$$

The intended meaning of the concurrent composition is to model concurrent interaction. The approach is to appeal to handshake communication as primitive. Concurrent computations are then interleaved computations of the two transition systems, possibly synchronised on complementary input/output actions, producing a τ action.

Definition 2 Downgrading Transition System. A *Downgrading Transition System* (DTS) is an *Act*-labelled pointed transition system whose set of visible actions Vis is a partition of 3 sets Lo, Hi and Dwn each of cardinality at least 2 such that $\overline{Lo} = Lo$, $\overline{Hi} = Hi$ and $\overline{Dwn} = Dwn$.

The next proposition proved in [Milner 89] shows that trace equivalence is a congruence w.r.t. the concurrent and restriction operators and that there is a weak form of distributivity of the restriction operator over the concurrent one.

Proposition 3. *If* $\mathrm{T}_{1s_1} \simeq \mathrm{T}'_{1s'_1}$ $\mathrm{T}_{2s_2} \simeq \mathrm{T}'_{2s'_2}$ *then*

1. $\mathrm{T}_{1s_1} | \mathrm{T}_{2s_2} \simeq \mathrm{T}'_{1s'_1} | \mathrm{T}'_{2s'_2}$
2. $\mathrm{T}_{1s_1} \setminus L \simeq \mathrm{T}'_{1s'_1} \setminus L$
3. *If* T_{1s_1} *and* T_{2s_2} *do not synchronise on actions in* L *then*

$$(\mathrm{T}_{1s_1} | \mathrm{T}_{2s_2}) \setminus L \simeq (\mathrm{T}_{1s_1} \setminus L) | (\mathrm{T}_{2s_2} \setminus L)$$

3 Admissible Interference

Informally, a high-level user group does not interfere with a low-level user group if for any command string α the low-level users cannot observe any difference between the system executing α and the system executing the command string α' obtained from α in deleting all high-level commands. The original definition was only applicable to deterministic systems [Goguen, Meseguer 82]. In [McCullough 88], this definition has been extended to cope with nondeterministic systems. A more restrictive form called *strong non-deterministic noninterference* (SNNI) has been proposed in [Focardi, Gorrieri 95]. It requires that, for every trace γ, the sequence obtained from γ by deleting or purging all high-level actions is still a trace. It can formally be defined as follows:

$$\forall_{\gamma \in \mathcal{L}(T_s)} low(\gamma) \in \mathcal{L}(T_s) \tag{1}$$

where the function $low : Vis^* \rightarrow (Vis \setminus Hi)^*$ is defined as follows: $low(\alpha)$ is the projection of α over its action subsequence visible from the low-level. Let us consider how this notion of information flow prohibition arises in interacting transition systems, taking a simple example. The transition diagram $\circ \xrightarrow{a} \circ \xrightarrow{\bar{b}} \circ$ represents a behaviour which synchronises at a as input, then synchronises at b as output, and does nothing. Suppose we attach secrecy levels to each action, for example $a \in Hi$ and $b \in Lo$. Intuitively this means that we wish interaction at a to be secret, while interaction at b may be known by a wider public: any high-level action may interact at a and b, while a low-level action may interact only at b. Then this transition system represents a prohibited information flow from the high-security level to the low-security one: any low-level observer i.e. any observer allowed to observe only the low-level projection of traces, has the possibility to know whether an interaction with a happened or not. The reason is that there is some causal dependency from the high-level action a to b, a low-level one. Such a dependency does not exist anymore in the transition diagram sketched in [Fig. 1]:

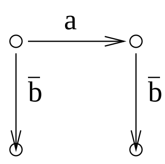

Figure 1: Transition system satisfying strong non-deterministic noninterference

Suppose now that one admits communication between the high-level and the low-level user groups but only through a certain command set *Dwn* of a downgrading-level. The intuitive meaning of *conditional noninterference* (CNI) proposed in [Goguen, Meseguer 84] to describe it is that at any time in any action string, high-level actions have no effect on low-level ones after the last downgrading-level action. The set of action strings being a prefix-closed set, it states that if the high-level user group interferes somewhere and somehow with the low-level one, it must be through downgrading-level actions only. All the formalisations of this idea (e.g. [Pinsky 95, Rushby 92, Haigh, Young 87]) are based on a purge function in which the purgeability of an action within a string requires the examination of the suffix of the string from that point on. A high-level action is purgeable only if there is no substring beginning at that point and admitting interference. In a first attempt to formalise this idea starting from SNNI as defined in [Eq. 1], we get the following approximation which closely mimics the above description of purge:

$$\forall_{\gamma \in \mathcal{L}(T_s)} \gamma_1 low(\gamma_2) \in \mathcal{L}(T_s) \tag{2}$$

where $\gamma = \gamma_1\gamma_2$ and γ_2 is the longest suffix of γ in $(Vis \setminus Dwn)^*$. That is for any action string α the low-level users cannot observe any difference between the system executing α and the system executing the string obtained from α in deleting only high-level actions which are not followed by downgrading-level ones. In the transition diagrams depicted in [Fig. 2.(a) and 2.(b)], suppose we attach the high-secrecy level to action a, the low-secrecy level to action b and suppose we require any interaction at a to be secret except through an action c declassifying it that is $c \in Dwn$. In both examples, a low-level observer is not allowed to observe any occurrence of a and transmit it through b unless a is leaked through c and nothing more is allowed to flow from the high-level to the low-level.

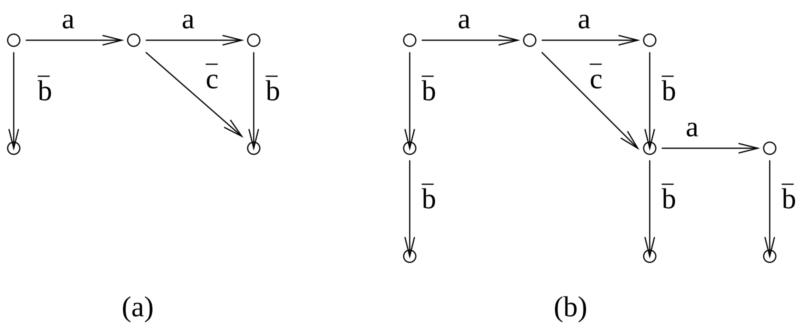

Figure 2: Transition systems satisfying conditional noninterference

Now, let see the way inadmissible information flow is detected through simple examples. In the transition diagram depicted in [Fig. 3, suppose $a \in Hi$, $b \in Lo$ and $c \in Dwn$. In [Fig. 3.(a)], $aa\overline{b}$ is a trace of the transition system while b is not. Giving an intuitive interpretation to [Eq. 2], it means that any low-level user knows the second occurrence of the high-level action a as soon as he observes $\overline{b}$ i.e. the projection of the trace $aa\overline{b}$ onto $(Vis \setminus Hi)^*$ while this observation is not allowed. Of course, any low-level user knows about the first occurrence of the high-level action as soon as $\overline{c}$ i.e. the projection of the trace $a\overline{c}$ onto $(Vis \setminus Hi)^*$, is observed but this information flow through c is allowed. With a similar reasoning, one detects a prohibited or inadmissible information flow in [Fig. 3.(b)]: the causal dependency between the trace $a\overline{c}a\overline{b}$ and its low-level observation $\overline{cb}$. Since $\overline{cb}$ is not a trace, the third occurrence of the high-level action a is revealed to any low-level observer.

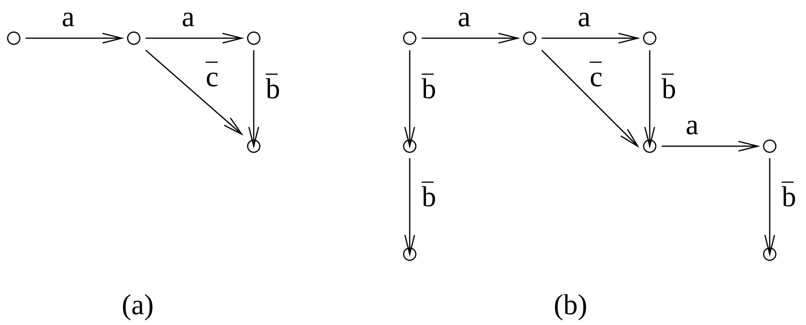

Figure 3: Transition systems failing to satisfy conditional noninterference

Unfortunately, [Eq. 2] does not cope with nondeterminism when it appears on downgrading-level actions to ensure that after any string γ_1 , there is no low-level user able to discriminate neither the string γ_2 from the string γ_2 purged from its high-level actions nor the state of the system just before γ_2 as it is seen in [Fig. 4]. As in the previous example, let $a \in Hi$, $b \in Lo$ and $c \in Dwn$. There is an obvious disclosure of the occurrence of the high-level action a next to the upward $\overline{c}$ transition and this disclosure is not detected with [Eq. 2] since $\overline{cb}$ is a trace.

The property we are going to propose, called *admissible interference* (AI) copes with such a situation and appears naturally as a nondeterministic generalisation of CNI in the following sense: in the deterministic case, it is easy to see that [Def. 4] implies [Eq. 2].

Definition 4 Admissible Interference. A DTS $T_s = (Act, S, R)$ satisfies admissible interference if

$$\forall_{\gamma\in\mathcal{L}(T_s)}\forall_{s'\in S}(\gamma_2 \in \mathcal{L}(T_{s'}) \Rightarrow low(\gamma_2) \in \mathcal{L}(T_{s'}))$$

for every $\gamma_2 \in (Hi \cup Lo)^*$ s.t. $\gamma = \gamma_1\gamma_2$.

That is for any trace γ and any computation of γ the low-level users cannot observe any difference between the system executing γ and the system executing the string obtained from γ in purging only the suffix γ_2 of γ containing no downgrading-level actions. Otherwise stated, for every $\gamma \in \mathcal{L}(T_s)$ and every computation of γ, there is no information flow after γ_1 i.e. after the last downgrading action of γ. Applying the definition to the transition diagram depicted in [Fig. 4] and taking the trace $a\overline{c}a\overline{b}$, there is a state s_u (the sink state of the upward c transition) such that $a\overline{b} \in \mathcal{L}(T_{s_u})$ but $\overline{b}$, its observation from s_u by a

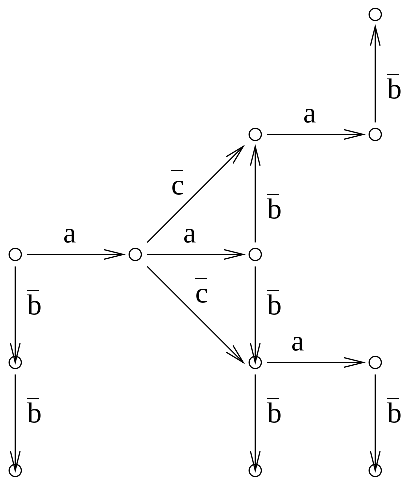

Figure 4: Prohibited information flow undetected by conditional noninterference

low-level user , is not in $\mathcal{L}(T_{s_u})$. So, [Def. 4] detects the unauthorised information disclosure and gone undetected by [Eq 2], as discussed above.

4 A DTS-based Unwinding Theorem for AI

We now provide a DTS-based unwinding characterisation of AI. Such a characterisation is more amenable to automated proof than the purging-based definition. For comparison, here is the DTS-based unwinding theorem for SNNI established in [Focardi, Gorrieri 95]:

Proposition 5 Unwinding Theorem for SNNI. *A DTS* T_s *satisfies SNNI if and only if*

$$\mathrm{T}_s/Hi \simeq \mathrm{T}_s \setminus Hi$$

The extra quantification appearing in the AI definition compared to the SNNI definition is reflected in its DTS-based unwinding theorem:

Proposition 6 Unwinding Theorem for AI. *A DTS* T_s *satisfies AI if and only if*

$$(\mathrm{T}_{s'} \setminus Dwn)/Hi \simeq (\mathrm{T}_{s'} \setminus Dwn) \setminus Hi$$

for any state s' *of* T_s.

Proof. $(\Rightarrow)$ It is sufficient to show that for any state s',

$$\mathcal{L}((T_{s'} \setminus Dwn)/Hi) \subseteq \mathcal{L}((T_{s'} \setminus Dwn) \setminus Hi)$$

because the opposite inclusion is trivial.

Let $s' \in S$ and $\gamma \in \mathcal{L}((T_{s'} \setminus Dwn)/Hi)$; then $\exists_{\gamma' \in \mathcal{L}(T_{s'} \setminus Dwn)} low(\gamma') = \gamma$. Since γ' is also in $\mathcal{L}(T_{s'})$ and s' is reachable by hypothesis, there is a $\gamma_1 \in Act^*$ such that $\gamma_1\gamma' \in \mathcal{L}(T_s)$. Since $\gamma' \in \mathcal{L}(T_{s'}) \cap (Hi \cup Lo)^*$ and T_s satisfies AI then $\gamma \in \mathcal{L}(T_{s'})$. Hence $\gamma \in \mathcal{L}((T_{s'} \setminus Dwn) \setminus Hi)$ because $\gamma \in Lo^*$.

($\Leftarrow$) If $\gamma = \gamma_1\gamma_2 \in \mathcal{L}(T_s)$ where $\gamma_2 \in (Hi \cup Lo)^*$ and $s' \in S$ s.t. $\gamma_2 \in \mathcal{L}(T_{s'})$ then $\gamma_2 \in \mathcal{L}(T_{s'} \setminus Dwn)$ and

$$\exists_{\delta \in \mathcal{L}((T_{s'} \setminus Dwn)/Hi)} \delta = low(\gamma_2).$$

Since by hypothesis $\mathcal{L}((T_{s'} \setminus Dwn) \setminus Hi) = \mathcal{L}((T_{s'} \setminus Dwn)/Hi)$ then $\delta \in \mathcal{L}((T_{s'} \setminus Dwn) \setminus Hi)$. And finally, since $\mathcal{L}((T_{s'} \setminus Dwn) \setminus Hi) \subseteq \mathcal{L}(T_{s'})$ then $\delta \in \mathcal{L}(T_{s'})$.

The next proposition establishes the compositionality of AI w.r.t. the restriction operator and a weak form of compositionality of AI w.r.t. the concurrent operator. The proof requires the following lemma stating that the functional composition of restriction to Dwn with hiding Hi is distributive over the concurrent composition:

Lemma 7. *If* T_s *and* $\mathrm{T}'_{s'}$ *do not synchronise on down-level actions then*

$$((\mathrm{T}_s \setminus Dwn)/Hi)|((\mathrm{T}'_{s'} \setminus Dwn)/Hi)) \simeq ((\mathrm{T}_s|\mathrm{T}'_{s'}) \setminus Dwn)/Hi$$

Proof. It is sufficient to show that

$$\mathcal{L}(((T_s \setminus Dwn)/Hi)|((T'_{s'} \setminus Dwn)/Hi)) \supseteq \mathcal{L}(((T_s|T'_{s'}) \setminus Dwn)/Hi)$$

because the opposite inclusion results trivially from [Prop. 3.3].

For the $\supseteq$ inclusion, one proceeds by structural induction on the concurrent composition rules. The only difficult case is the one raised from a τ transition by high-level action synchronisation resulting from application of the SYNC rule. Let $(s_1, s'_1) \xrightarrow{\tau} (s_2, s'_2)$, a $(((T_s|T'_{s'}) \setminus Dwn)/Hi)$-transition issued from the T_s-transition $s_1 \xrightarrow{a} s_2$ and the $T'_{s'}$-transition $s'_1 \xrightarrow{\overline{a}} s'_2$ with $a \in Hi$. It results in the $((T_s \setminus Dwn)/Hi)$-transition $s_1 \xrightarrow{\tau} s_2$ and the $((T'_{s'} \setminus Dwn)/Hi)$-transition $s'_1 \xrightarrow{\tau} s'_2$ to obtain the $(((T_s \setminus Dwn)/Hi)|((T'_{s'} \setminus Dwn)/Hi))$-transition $(s_1, s'_1) \xrightarrow{\tau} (s_2, s'_2)$.

Proposition 8 Compositionality Theorem for AI. *1. Let* $L \subseteq Act$. *If* T_s *satisfies AI then* $\mathrm{T}_s \setminus L$ *satisfies AI.*

2. If T_{1s_1} *and* T_{2s_2} *do not synchronise on down-level actions and satisfy AI then* $\mathrm{T}_{1s_1}|\mathrm{T}_{2s_2}$ *satisfies AI.*

Proof. 1. Let s' some state of $T_s \setminus L$. In view of [Prop. 6], it is sufficient to show that

$$((T_{s'} \setminus L) \setminus Dwn)/Hi \sqsubseteq ((T_{s'} \setminus L) \setminus Dwn) \setminus Hi$$

because the opposite inclusion is trivial. We have:

$$\begin{aligned}
((T_{s'} \setminus L) \setminus Dwn)/Hi &\simeq ((T_{s'} \setminus Dwn) \setminus L)/Hi \\
&\sqsubseteq ((T_{s'} \setminus Dwn)/Hi) \setminus L \\
&\simeq ((T_{s'} \setminus Dwn) \setminus Hi) \setminus L \\
&\quad \text{(by [Prop 6 and Prop. 3.2])} \\
&\simeq ((T_{s'} \setminus L) \setminus Dwn) \setminus Hi
\end{aligned}$$

2. Let (s'_1, s'_2) be a state of $T_{1s_1} | T_{2s_2}$. It is sufficient to show that:

$$((T_{1s'_1} | T_{2s'_2}) \setminus Dwn/Hi) \sqsubseteq ((T_{1s'_1} | T_{2s'_2}) \setminus Dwn \setminus Hi).$$

because the opposite inclusion is trivial. We have:

$$\begin{aligned}((T_{1s'_1} | T_{2s'_2}) \setminus Dwn)/Hi &\simeq ((T_{1s'_1} \setminus Dwn)/Hi)|((T_{2s'_2} \setminus Dwn)/Hi)) \\ &\quad \text{(by lemma 7)} \\ &\simeq ((T_{1s'_1} \setminus Dwn) \setminus Hi)|((T_{2s'_2} \setminus Dwn) \setminus Hi) \\ &\quad \text{(by [Prop. 6 and Prop. 3.1])} \\ &\simeq (T_{1s'_1} \setminus (Dwn \cup Hi))|(T_{2s'_2} \setminus (Dwn \cup Hi)) \\ &\simeq ((T_{1s'_1} \setminus (Dwn \cup Hi))|(T_{2s'_2} \setminus (Dwn \cup Hi))) \\ &\quad \setminus(Dwn \cup Hi) \\ &\sqsubseteq (T_{1s'_1} | T_{2s'_2}) \setminus (Dwn \cup Hi) \\ &\simeq ((T_{1s'_1} | T_{2s'_2}) \setminus Dwn) \setminus Hi.\end{aligned}$$

Example 1. As an illustration of our results, we model some critical configurations of the transaction system briefly described in the introduction. The components are sketched on [Fig. 5].

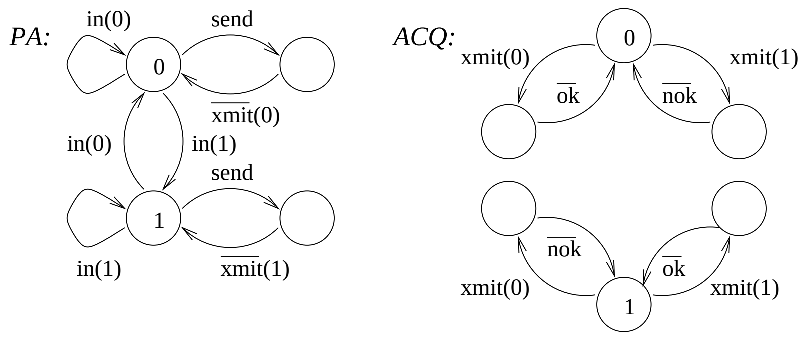

Figure 5: Components of a simple transaction system.

To be consistent with the notation uniformly used throughout the paper, C_i will stands for the DTS C with i as initial state. For the sake of simplicity, the domain of values for account numbers is restricted to $V = \{0, 1\}$. An action is sometimes a pair (c, i) where c could be viewed as a channel and i, an account number. In this case, an information i transmitted over two channels c_1 and c_2 is represented by the actions (c_1, i) and (c_2, i). Also, it is natural to define $\overline{(c, i)}$ as $(\overline{c}, i)$ and to attach security levels to channels. In the following, *ok* and *nok* stand respectively for the payment authorisation and refusal actions sent from the acquirer to the merchant, *cc* models some covert channel, *in* is the channel used by users to transmit their account number to the purchasing agent, $xmit$ is the channel used by the purchasing agent to transmit the just received user's account number to the acquirer following a request from the user through the action *send*. Actions $in, xmit, send$ are used along the confidential transaction while ok, nok are used to downgrade to the merchant the required part of information about the transaction and the channel cc will be used by a non authorised low-level user to sneak confidential information i.e. as a covert channel. It is reflected by the following security level setting:

$$
\begin{aligned}
Dwn &= \{ok, nok\} \\
Lo &= \{cc\} \\
Hi &= \{in, xmit, send\}
\end{aligned}
$$

The DTS PA_i depicted in [Fig. 5] models the purchasing agent ready to transmit the account number i on reception of the *send* request from the user, to the acquirer along the channel $xmit$. The account number is received from a user along the input channel in. Also, the purchasing agent allows the users to update the submitted account number.

The DTS ACQ_i also depicted in [Fig. 5] models the acquirer owning his own copy of the account number i. Due to a limitation of the model, this copy will not be allowed to be modified along the transaction since it is used by the acquirer as the standard reference to compare with the account number received from the purchasing agent along the channel $xmit$. The acquirer then transmits to the merchant (not modelled herein) a payment authorisation *ok* or refusal *nok* depending whether both numbers agree or not.

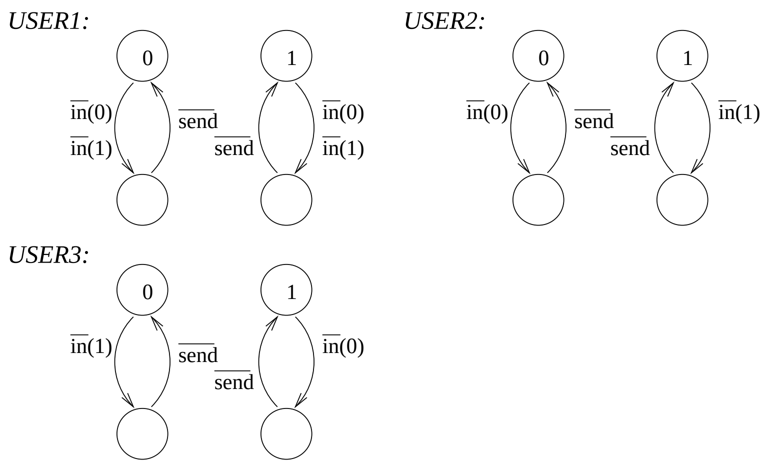

Figure 6: User behaviours properly managed by standard access control.

Users may behave in different ways. Let first have a look to quite standard ones depicted in [Fig. 6] and briefly described below. The DTS $USER1_i$ models the "chaotic" behaviour of the user having i as account number: he sends at random 0 or 1 as his account number to the purchasing agent along the channel in, followed by a *send* request; the DTS $USER2_i$ models the "angelic" behaviour of the user having i as account number: he sends his own account number to the purchasing agent along the channel in, followed by a *send* request; the DTS $USER3_i$ models the "demonic" behaviour of the user having i as account number: he always provides a wrong account number $((i+1) \bmod 2)$ to the purchasing agent along the channel in, followed by a *send* request. Intuitively, the specification of the transaction system is robust enough to easily deal with these different behaviours and manage properly, in any case, to refuse payment authorisation to any user providing a wrong account number to the purchasing agent. Actually, under standard conditions of use, access policies as the one implemented in the transaction system are sufficient to provide confidentiality.

On the other hand, here is an attack which could not be caught by any information flow property. It is depicted in [Fig. 7]. The DTS $USER4_0$ models a user having 0 as account number and succeeding to get access rights in a no specified manner through some internal action τ, to the account of the user having 1 as account number and modelled by the DTS $USER4_1$. So $USER4_0$ is now on free to act on behalf of $USER4_1$ even without his knowledge. It is a typical example of what could happen when access restrictions are subject to user discretion e.g. access rights based on file ownership. The most important property of such access rights is that they can be passed to other users. As a result, it is subject to a Trojan Horse attack. The thing is that information flow is just concerned with issues such as protecting files from unauthorised reproduction by authorised users. So as long as $USER4_0$ does not try to go beyond $USER4_1$'s rights, he is free to act with complete impunity.

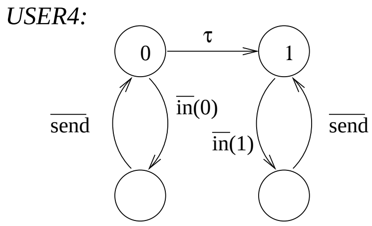

Figure 7: Trojan Horse attacking a discretionary access controlled system.

The two user behaviours depicted in [Fig. 8] model much more powerful attackers and provide interesting insights on the way admissible interference works or fails to work. The DTS $USER5_0$ models a user having 0 as account number and succeeding to sneak the $USER5_1$'s account number along the channel in in a no specified manner. $USER5_0$ could then act as Trojan Horse, taking advantage for example, of the admissible downgrading actions ok, nok to create a binary covert channel and leak $USER5_1$'s account number to some low-level user having access to the information downgraded to the merchant. This covert channel creates a causal dependency between the high-level action $in(1)$ and a downgraded one (ok or nok). So, SNNI catches this flow but unfortunately, admissible interference fails to catch it because such a dependency has been specified as admissible. In *USER6*, the Trojan Horse $USER6_0$ sneaks the $USER6_1$'s account number and creates a covert channel through the low-level channel cc to leak sensitive information. This flow will be caught by admissible interference since the way $USER6_0$ works to leak information is to behave differently when the account number is equal to 1, creating in this way a non-admissible causal dependency between the high-level action $in(1)$ and the low-level one $\overline{cc}(1)$.

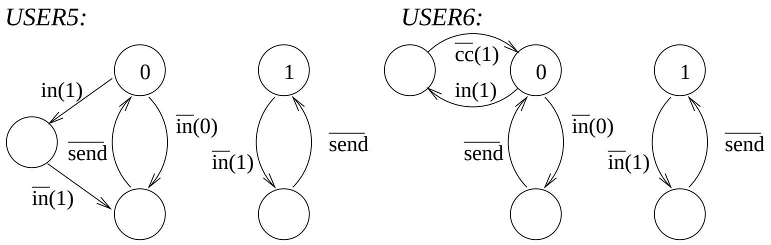

Figure 8: Trojan Horse attacks properly managed through information flow.

The above discussion can be formally stated from our results. It is indeed easy to check the following using [Prop. 6]:

1. *PA* satisfies AI,
2. *ACQ* satisfies AI,
3. *USERk* satisfies AI for $k = 1, 2, 3, 4, 5$,
4. *USER6* does not satisfy AI.

Also, the global interaction between transaction system components and users can be modelled from the models of the components interacting with the users models through the concurrent composition: for every $i \in V$ and every $1 \leq k \leq 6$,

$$SYS(k,i) = (PA|ACQ_i|USERk_0|USERk_1) \setminus Hi$$

is the transaction system resulting from the concurrent composition of the users DTS depicted by *USERk* with the purchasing agent's DTS and the acquirer's DTS set to i as referential account number and packaged with restriction to the non-confidential actions to protect the system against intruders. So, since $PA, ACQ, USERk_0$ and $USERk_1$ never synchronise on low-level actions ok nor nok, we may apply [Prop. 8] (actually we need the full strength of the compositionality result) and conclude that $SYS(k,i)$ satisfies AI for $k \in \{1,2,3,4,5\}$ and $i \in \{1,2\}$ while $SYS(6,i)$ does not, for $i \in \{1,2\}$.

[Example 1] exemplifies a fundamental aspect of security properties: to prove that a system satisfies a security property, one has to prove its specification against every potential attacker. Nevertheless, admissible interference is still meaningful to express that a single component of the system abstracted from its execution environment is not a malicious agent i.e. does not leak information on behalf of a low-level user through a non admissible action as it is illustrated in the following version of the purchasing agent [Fig. 9] trying to leak the user's account number to some low-level agent no specified here, through the covert channel cc. Indeed, such a flow contains causal dependencies between $in(0)$ and $\overline{cc}(0)$ and between $in(1)$ and $\overline{cc}(1)$ which are detected as inadmissible by applying [Prop. 6].

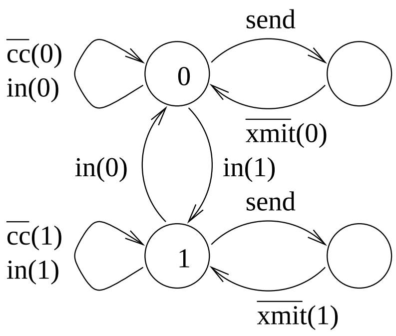

Figure 9: A malicious purchasing agent

5 Conclusion

In this paper we have introduced admissible interference, a new security property based on trace equivalence to cater to confidentiality in presence of downgrading. We first rephrased the usual purging-based definition of conditional noninterference. As it turns out, AI is a nondeterministic generalisation of this definition. We then provided unwinding conditions for AI in terms of DTS, an extended model of communicating transition system. This formal framework allowed us to state the completeness of the unwinding conditions and to investigate some compositionality properties of AI w.r.t. operators widely used to describe operational semantics of concurrent processes. These results are finally illustrated on a model of transaction system.

In the following we would like to highlight three specific aspects we intend to focus our attention on, in view of further developments: first, the formulation of our results into more accurate notions of equivalence and into extensions of our communicating automaton algebra suited to model locality and mobility. Second, proof method permitting to prove the specification of the property only against the "most powerful" system intruder. Third, ongoing research about automating tools for checking security properties and particularly the property we have investigated here.

5.1 System Semantics and AI

It is certainly the case that the notion of security is deeply affected by the choice of semantic model for systems. Actually, semantics based on traces are too weak: in particular, it is not sensitive to high level deadlocks which could be used to construct a covert channel as shown in the next example.

Suppose we have an external port a and a high level action explicitly switching off a Trojan Horse process T acting on behalf of some intruder I. Suppose also that T tries to communicate a binary secret to I. If this secret is *one* then T switches off. Of course, the I trying to make an access to T through the port a makes $\mathbf{T}|\mathbf{I}$ enter into a deadlock and is unable to conclude that the process is off. However I can deduce that the secret is *zero* every time T accepts an access request through a. This cannot be detected by trace equivalence in the context of the SNNI property [Prop. 5] nor in the context of the trace-based AI property [Prop. 6] since $\mathbf{T/Hi}$ and $\mathbf{T} \setminus \mathbf{Hi}$ have the same traces [Fig. 10].

Figure 10: (a) $\mathbf{T}/Hi$ (b) $\mathbf{T} \setminus Hi$

The coarser deadlock sensitive equivalence refining trace equivalence is the Hennessy's *test equivalence* [Hennessy 88]. However test equivalence is not a congruence w.r.t. the usual operators of process algebra. Bisimulation [Milner 89] possesses this property and refines test equivalence. We are therefore studying the effect of rephrasing our AI security property over these finer behavioural equivalences inspired by [Pinsky 95]. It appears indeed that the algorithm presented by Pinsky to construct a minimal equivalence and its associated unwinding condition for a downgrading policy, can be thought of as an algorithm to construct the appropriate bisimulation.

In the last couple of years a number of process calculi have been designed to model communicating mobile computations with localities [Hennessy, Riely 99, Cardelli, Gordon 98, Vitek, Castagna 99]. Given the importance of writing secure distributed applications over the Internet, we plan to reformulate our security results for such calculi and extend them with cryptographic primitives à la spi calculus [Abadi, Gordon 97] to cope with encryption and decryption of sensitive information.

5.2 Verification Methods for AI

We have already mentioned the following basic principle concerning the analysis of security properties: a security property should be satisfied even in presence of hostile environment. Also, it should be resistant to every potential attacker. Checking this condition is quite intractable. Some general conditions that permit to check a property only against its "most powerful attacker" have been defined in [Focardi, Martinelli 99]. We are studying this approach for proving AI.

It is interesting to mention that the spi calculus [Abadi, Gordon 97] overcomes this problem by representing security properties as weakened form of testing equivalence. Let $P(X)$ be a process P processing a piece of data X. From the spi calculus point of view, P preserves secrecy of X if there is no test with the capability to discriminate $P(X)$ from $P(X')$, for every X'. A test nicely formalises the idea of a generic experiment or observation that another spi process (a potential attacker) might perform on P. So P and Q are testing equivalent if there exists no attacker powerful enough to discriminate them. Also, it is a good motivation to experiment a new formulation of AI in terms of testing equivalence.

5.3 Computer-aided Verification

We are presently designing and implementing a tool based on the Concurrency Workbench [Parrow et al. 93]. This tool, called *Computer Security Workbench* takes finite DTS specifications in input and checks whether they satisfy admissible interference or not by application of [Prop. 6] and by taking advantage of the compositionality properties by exploitation of [Prop. 8] to keep the combinatorial explosion under control. A detailed documentation of the tool with several examples is in preparation.

Also in project, is a significative and realistic case-study specifying a real-world Canadian electronic banking application in view of its eventual deployment over the Internet.

Acknowledgements

Mads Dam and Pablo Giambiagi at the Dept. of Teleinformatics, KTH (Sweden) were both closely involved in the initial development of this work and Gaétan Hains at the Laboratoire d'Informatique Fondamentale d'Orléans (France) suggested improvements to a draft of this paper. Thanks are due also to Herbert Sander at Ericsson Utvecklings AB in Stockholm for helpful discussions.

References

[Abadi, Gordon 97] Abadi, M., and Gordon A. D.: "A calculus for cryptographic protocols: The spi calculus"; Proc. 4th ACM Conference on Computer and Communications Security (1997), 36–47. Full version available as tech. rep. 414, Univ. Cambridge Computer Lab.

[Bellare et al. 95] Bellare, M., Garay, J., Hauser, R., Herzberg, A., Krawczyk, H., Steiner, M., Tsudik, G., and Waidner, M.: "*i*KP – a family of secure electronic payment protocols"; First USENIX Workshop on Electronic Commerce (1995).

[Cardelli, Gordon 98] Cardelli, L, Gordon, A.G.: "Mobile ambients"; In Maurice Nivat, editor, Proceedings of FoSSaCS '98, volume 1378 of LNCS, Springer (1998), 140–155.

[Focardi, Gorrieri 95] Focardi, R., Gorrieri, R.: "A classification of security properties for process algebras"; J. of Computer Security, 3,1 (1994/1995), 5–33.

[Focardi, Martinelli 99] Focardi, R., Martinelli, R.: "A uniform approach for the definition of security properties"; In J. Wing, J. Woodcock, and J. Davies, editors, FM'99–Formal Methods, volume 1708 of LNCS, Springer (1999), 794–813.

[Goguen, Meseguer 82] Goguen, J.A., Meseguer J.: "Security policies and security models"; Proceedings of the IEEE Symp. on Research in Security and Privacy, IEEE Computer Society, Oakland, CA (1982), 11–20.

[Goguen, Meseguer 84] Goguen, J.A., Meseguer J.: "Unwinding and inference control"; Proceedings of the IEEE Symp. on Research in Security and Privacy, IEEE Computer Society, Oakland, CA (1984), 75–285.

[Haigh, Young 87] Haigh, J.T., Young, W.D.: "Extending the noninterference version of MLS for SAT"; IEEE Trans. on Software Engineering, 13, 2 (1987), 141–150.

[Hennessy 88] Hennessy, M.: "Algebraic theory of processes"; The MIT Press (1988).

[Hoare 85] Hoare, C.A.R.: "Communicating sequential processes"; Prentice-Hall (1985).

[Hennessy, Riely 99] Hennessy, M., Riely, J.: "Type-safe execution of mobile agents in anonymous networks"; In Jan Vitek and Christian Damsgaard Jensen, editors, Proceedings of MOS '98, volume 1603 of LNCS, Springer (1999). Full version as CogSci Report 3/98, University of Sussex, Brighton.

[McCullough 87] McCullough, D.: "Specifications for multi-level security and hook-up property"; Proceedings of the IEEE Symp. on Research in Security and Privacy, IEEE Computer Society, Oakland, CA (1987), 161–166.

[McCullough 88] McCullough, D.: "Noninterference and the composability of security properties"; Proceedings of the IEEE Symp. on Research in Security and Privacy, IEEE Computer Society, Oakland, CA (1988), IEEE Computer Society.177–186.

[McLean 94] McLean, J.: "Security models"; In J. Marciniak, editor, Encyclopedia of Software Engineering, John Wyley & Sons (1994), 1136–1145.

[Milner 89] Milner, R.: "Communication and concurrency"; Prentice-Hall (1989).

[Parrow et al. 93] Parrow, J., Cleaveland, R., Steffen, B.: "The concurrency workbench: a semantics based tool for the verification of concurrent processes"; ACM Transactions on Programming Languages and Systems, 15, 1 (1993), 36–72.

[Pinsky 95] Pinsky, S.: "Absorbing covers and intransitive non-interference"; Proceedings of the IEEE Symp. on Research in Security and Privacy, IEEE Computer Society, Oakland, CA (1995), 102–113

[Roscoe, Goldsmith, 99] Roscoe, A.W., Goldsmith, M.H.: "What is intransitive noninterference?" In Proceedings of the Twelfth IEEE Computer Security Foundations Workshop (CSFW-12), Mordano, Italy (1999).

[Ryan, Schneider 99] Ryan, P.Y.A., Schneider, S.A.: "Process algebra and noninterference"; In Proceedings of the Twelfth IEEE Computer Security Foundations Workshop (CSFW-12), Mordano, Italy (1999).

[Rushby 92] Rushby, J.: "Noninterference, transitivity and channel-control security policies"; Technical Report CSL-92-02, SRI International, Menlo Park CA, USA (1992).

[Sutherland 86] Sutherland, D.: "A model of information"; Proceedings of the IEEE Symp. on Research in Security and Privacy, IEEE Computer Society, Oakland, CA (1986), 11–20.

[Vitek, Castagna 99] Vitek, J., Castagna, G.: "Seal: A framework for secure mobile computation"; In H.E. Bal, B. Belkhouche, and L. Cardelli, editors, Internet Programming Languages, volume 1686 of LNCS, Springer (1999).