An Information Flow Security Property for CCS
(Extended Abstract)

Riccardo Focardi  Roberto Gorrieri

Technical Report UBLCS-93-20
October 1993

Laboratory for Computer Science
University of Bologna
Piazza di Porta S. Donato, 5
40127 Bologna (Italy)
The University of Bologna Laboratory for Computer Science Research Technical Reports are available via anonymous FTP from the area ftp.cs.unibo.it:/pub/TR/UBLCS in compressed PostScript format. Abstracts are available from the same host in the directory /pub/TR/UBLCS/ABSTRACTS in plain text format. All local authors can be reached via e-mail at the address last-name@cs.unibo.it.

UBLCS Technical Report Series

93-7 *Distributed Programming with Logic Tuple Spaces*, by P. Ciancarini, April 1993.
93-8 *Coordinating Rule-Based Software Processes with ESP*, by P. Ciancarini, April 1993.
93-14 *Guard Compilation in Logic Shared Dataspaces Languages*, by M. Garpari, June 1993.
An Information Flow Security Property for CCS

(Extended Abstract)

Riccardo Focardi2 Roberto Gorrieri2

Technical Report UBLCS-93-20

October 1993

Abstract

Multilevel security has been introduced to limit the activity of Trojan Horses – malicious programs which try to broadcast secret information. Every information is classified with a security level and the multilevel secure system must guarantee that information cannot flow from a level to a lower one. So a Trojan Horse, which operates at a certain level, has no way to downgrade information and the effect of its execution is restricted into such a level.

In this paper, a formal property of security, called Non Deducibility on Compositions (NDC), is proposed and defined on CCS agents. For the sake of simplicity, we assume to have two security levels only: high and low. In this context, a Trojan Horse is a high level process which tries to pass information to the low level. NDC is based on the following intuition: a system is NDC secure if, when connected to all possible high level processes, the computations of the low level users are not affected; hence, no information flow from the high level to the low one. An alternative formulation of NDC, exploiting local information only, is proposed, which has the merits of being algorithmically testable for finite-state systems. Moreover, it is useful for modular verification of secure systems as it is a composable w.r.t. the CCS operators of parallel composition and restriction.

Finally, we show that NDC, based on trace semantics, may be insufficient in some cases. Hence, NDC is formulated also assuming the finer (weak) bisimulation as basic underlying semantics.

1 Introduction

Security is a crucial property of system behaviour. It requires that there is a strict control over the information flow among the components of a system. The main problem is to limit, and possibly avoid, the damages produced by malicious programs, usually called Trojan Horses, which try to broadcast secret information. There are several approaches to security, most of which based on some access control policy.

In the Discretionary Access Control security (DAC for short), every subject (i.e., an active agent such as a user), decides the access properties of its objects (i.e., passive agents such as files). An example of DAC is the file management in Unix where a user can decide the access possibilities of her/his files. So, if a user executes a Trojan Horse program, this can modify the security properties of user’s objects (usually, a program inherits executor’s capabilities).

A solution to this problem is the security discipline called Mandatory Access Control (MAC for short), where access rules are imposed by the system. An example of MAC is Multilevel Security [1]: every object is bound to a security level, and so every subject is; information can flow from a certain object to a certain subject only if the level of the subject is greater than the level of the object. So a Trojan Horse, which operates at a certain level, has no way to downgrade information and its action is restricted inside such a level. There are two access rules (see figure 1):

- **No Read Up**: A subject cannot read data from an upper level object.
- **No Write Down**: A subject cannot write data to a lower level object.

However these access rules do not guarantee an absence of information flow from upper levels to lower ones. It could be possible to indirectly transmit information using some system side effect. For example, if two levels – “high” and “low” – share some finite storage resource (e.g., a hard disk), it is possible to transmit data from level “high” to level “low” by using the “resource full” error message. It is sufficient, for a high level transmitter, to alternatively fill or empty the resource in order to transmit a “1” or a “0” data. Simultaneously, the low level receiver tries to write on the resource, decoding every error message with a “1” and every successful write with a “0”.

It is clear that such indirect ways of transmission, called covert channels, do not violate the two multilevel access rules. Therefore, to guarantee a correct information flow control, it is necessary to integrate a MAC discipline with a covert channel analysis [9, 16, 10, 14].

An alternative, more general approach to security requires to control directly the whole flow of information, rather than the accesses of subjects to objects. To make this, it is necessary to choose a formal model of system behaviour and to define what the flow of information is in such a model. By imposing some information flow rule, we can control any kind of transmission, be it direct or indirect.

Our proposal is based on labelled transition systems [8] (lts for short), a very general and simple model, used to give semantics to many concurrent languages. In particular, (a variant of) Milner’s Calculus of Communicating Systems (CCS, for short) [13] is used in this paper to specify the behaviour of concurrent systems.

For the sake of simplicity, we assume to have two security levels only: high and low. Each action will be bound to a security level; hence, it will be sufficient to partition the CCS action set in the sets $\text{Act}_L$ of low level actions and $\text{Act}_H$ of high level ones. Agents built with high (low) level actions only are often called high (low) level agents. In this context, an information flow from the high level to the low one takes place when the effects of high level

---

3. The generalization of the approach to $n$ levels can be easily done.
actions are visible by low level users. So, a Trojan Horse is a high level agent which tries to modify the low level executions.

In the literature, there are already many different definitions of this kind, based on several system models [4, 15, 11, 7, 17, 12]. In [3] we have reformulated them under the lts model, then compared, classified and “evaluated” according to their usefulness. This study led to the discovery of the general security notion presented in this abstract: **Non Deducibility on Compositions** (NDC, for short).

NDC may be seen as a transposition on lts’s of the so-called **Non Deducibility on Strategies** (NDS) security property, which was originally defined in [17] over a very restrictive model. NDC assumes two security levels and is based on the following intuition: a system is NDC secure if, when connected to all possible high level agents (or Trojan Horses), the computations of the low level users are not affected. So, such a property guarantees that no information flow from the high level to the low one can take place. We also show that NDC is closed with respect to system composition. This property is very useful in verification and permits to write a new system as a composition of subsystems satisfying NDC. This is the subject of Section 3.

An example of application of the theory is reported in Section 4. It presents an access monitor, which is proved to be NDC secure.

It is then shown – Section 5 – that a system which satisfies the NDC property may show up a non secure behaviour. This is due to the fact that NDC has been defined exploiting a rather weak notion of system behaviour, namely the so-called Hoare’s *trace* semantics, which equates systems when they have the same set of execution traces. We extend NDC to the finer equivalence of *weak bisimulation* [13], hence obtaining a much finer security property which, in our opinion, seems to be a reasonable candidate to express formally security constraints in the area of process algebraic languages.
2 Preliminary Definitions

We assume the reader familiar with the CCS language and the lts model, shortly presented in the following (see [13] for a comprehensive treatment).

Definition 2.1 A labelled transition system (lts) is a triple \((S, T, \rightarrow)\) such that:

- \(S\) is a set of states
- \(T\) is a set of labels (actions)
- \(\rightarrow \subseteq S \times T \times S\) is a set of labelled transitions

\((S_1, \alpha, S_2) \in \rightarrow\) (or equivalently \(S_1 \xrightarrow{\alpha} S_2\)) means that the system can move from the state \(S_1\) to the state \(S_2\) through the action \(\alpha\). CCS syntax is based on the following elements:

- A set \(I = \{a, b, \ldots\}\) of input actions, a set \(O = \{\tilde{a}, \tilde{b}, \ldots\}\) of output actions, and a set \(\mathcal{L} = I \cup O\) of visible actions.
- A function \(\gamma : \mathcal{L} \rightarrow \mathcal{L}\) such that \(\forall a \in A, \tilde{a} \in \tilde{A}\) and \(\forall \tilde{a} \in \tilde{A}, \tilde{a} = a, a \in A\).
- A set \(\mathcal{A} = \mathcal{L} \cup \{\tau\}\) (\(\tau\) is the internal action) of actions or signals.
- A set \(K\) of constants

The syntax of CCS agents is defined as follows:

\[E ::= 0 \mid \mu.E \mid E + E \mid E[E] \mid E \setminus L \mid E[f] \mid Z\]

where \(\mu \in \mathcal{A}\), \(L \subseteq I\), \(Z \in K\) and \(f : \mathcal{A} \rightarrow \mathcal{A}\) is such that \(f(\tilde{a}) = \overline{f(a)}\), \(f(\tau) = \tau\). Moreover, for every constant \(Z \in K\) there must be the corresponding definition: \(Z \triangleq E\), where \(Z\) may occur in \(E\) only if \(\overline{Z}\) occurs in a prefix context. Intuitively, \(0\) is the empty process, which cannot do any action; \(\mu.E\) can perform action \(\mu\) and then behaves like \(E\); \(E_1 + E_2\) is a process which can nondeterministically choose to behave like \(E_1\) or \(E_2\); \(E_1[E_2\) is the parallel composition of \(E_1\) and \(E_2\), where the executions of the two systems are interleaved, possibly synchronised on complementary input/output actions; \(E \setminus L\) can execute all the actions \(E\) is able to do, provided that they do not belong to \(L\) or \(\overline{L}\); finally, if \(E\) can execute action \(\mu\), then \(E[f]\) performs \(f(\mu)\). In the following, when we say “a certain system \(E\)” we mean the CCS agent \(E\).

Let \(\mathcal{E}\) be the set of all CCS agents, ranged over by \(E, F\) (possibly indexed). Let \(\mathcal{L}(E)\) denote the sort of process \(E\), i.e., the set of actions executable by \(E\). Then, the operational semantics of CCS is the lts \((\mathcal{E}, \mathcal{A}, \rightarrow)\) where the transition relation \(\rightarrow\) can be found in [13].

A widely used system composition operator requires that two systems are composed in parallel by forcing the synchronisation on their common (complementary) actions. This can be implemented in CCS by means of parallel composition (which permits synchronizations) and restriction (which forbids asynchronous moves, hence forcing synchronizations):

Definition 2.2 The system composition of two agents \(E\) and \(F\) is \((E|F) \setminus (\mathcal{L}(E) \cap \mathcal{L}(F))\).

Now we firstly introduce execution traces and trace equivalence on CCS process terms; then, we report the definition of the standard notion of equivalence on CCS agents: observational equivalence, based on weak bisimulation.

Definition 2.3 Let \(t = a_1 \ldots a_n \in \mathcal{A}^*\) be a sequence of actions; then \(E \xrightarrow{t} E'\) if and only if \(E(t) \xrightarrow{t} \overline{(t)} E'\). For all \(E \in \mathcal{E}\) the set \(T_E \subseteq (I \cup O)^*\) of traces associated
with $E$ is defined as follows: given a sequence $\gamma = \alpha_1 \ldots \alpha_n$, $\gamma \in (I \cup O)^*$,

$$\gamma \in T_E \iff \exists E_1, \ldots, E_n \in \mathcal{E} \text{ such that } E \xrightarrow{\alpha_1} E_1 \xrightarrow{\alpha_2} \cdots \xrightarrow{\alpha_n} E_n$$

Two systems are trace equivalent if they have the same set of traces: $E \approx_T F \iff T_E = T_F$.

**Definition 2.4** If $t \in \text{Act}^*$, then $\tau \in (I \cup O)^*$ is the sequence obtained by deleting all occurrences of $\tau$ from $t$. A binary relation $B \subseteq \mathcal{E} \times \mathcal{E}$ over agents is a weak bisimulation if $(P, Q) \in B$ implies, for all $\alpha \in \text{Act}$,

(i) If $P \xrightarrow{\alpha} P'$ then, $\exists Q' : Q \xrightarrow{\alpha} Q' \land (P', Q') \in B$

(ii) If $Q \xrightarrow{\alpha} Q'$ then, $\exists P' : P \xrightarrow{\alpha} P' \land (P', Q') \in B$

We say that $P$ is observation equivalent to $Q$, denoted $P \approx_B Q$, if there exists a weak bisimulation $B$ such that $(P, Q) \in B$.

A few auxiliary definitions will be useful in the following.

**Definition 2.5** Let $A$ be a set. Let $\gamma, \gamma' \in A^*$ be two sequences of elements in $A$. $\gamma'$ is a subsequence of $\gamma$ ($\gamma' \prec \gamma$) if and only if $\gamma$ and $\gamma'$ are in the form:

$$\gamma = \alpha_1 \ldots \alpha_n, \quad \gamma' = \alpha_{k(1)} \ldots \alpha_{k(m)} \quad \text{with } m \leq n$$

where $k : [1, m] \rightarrow [1, n]$, is an increasing monotone function.

**Notation:** With abuse of notation $\alpha_i \in \gamma$ stands for $\alpha_i$ is an element of the sequence $\gamma$. Let $\gamma, \gamma'$ be two sequences such that $\gamma' \prec \gamma$, then $k_{\gamma', \gamma}$ denotes the subsequence function in definition 2.5.

A set $C$ of observable actions is complete iff for every input $i \in I$ in $C$, the corresponding output $\overline{i} \in O$ is still in $C$. The set of all the complete sets of actions is denoted with $\mathcal{U} \overset{\text{def}}{=} \{ U \subset (I \cup O) \mid U \text{ is complete} \}$.

The following functions will be used to define NDC. They are connected to the idea of security levels and assume that the set of actions $\text{Act}$ has been partitioned in two complete sets $\text{Act}_H, \text{Act}_L \in \mathcal{U}$ which represent the sets of high and low level actions, respectively.

The first function, $\text{low}$, extracts from a sequence of actions the subsequence composed of low level actions only. The second function, $\text{lowviews}$, returns the set of low traces of a certain system.

**Definition 2.6** Let $\text{Act}_H, \text{Act}_L \in \mathcal{U}$. The functions $\text{low}$ and $\text{lowviews}$ are defined as follows:

- $\text{low} : \text{Act}^* \rightarrow \text{Act}_L^*$; for every $\gamma \in \text{Act}^*$, $\gamma' \in \text{Act}_L^*$, $\gamma' = \text{low}(\gamma)$ is such that:

  $$\gamma' \prec \gamma \land \forall \alpha_i \in \gamma, \alpha_i \in \text{Act}_L \Rightarrow \exists j : i = k_{\gamma', \gamma}(j)$$

- $\text{lowviews} : \mathcal{E} \rightarrow \mathcal{P}(\text{Act}_L^*)$. For every agent $E \in \mathcal{E}$:

  $$\text{lowviews}(E) \overset{\text{def}}{=} \{ \gamma \in \text{Act}_L^* \mid \exists \gamma' \in T_E, \gamma = \text{low}(\gamma') \}$$

4. In particular, $\tau$ is the empty sequence $\epsilon$. By the definition above, $E \xrightarrow{\tau} E$ for any agent $E$. 

UBLCS-93-20
3 Non Deducibility on Compositions

Non Deducibility on Compositions (NDC) may be seen as a transposition on the lts model of Non Deducibility on Strategies (NDS) [17]. The basic idea of NDS is the following: A Trojan Horse is viewed as a high level input "strategy" [17] (or high level process) $\Pi$ which chooses a new input for the system depending on the sequence of previous input/output actions. Therefore, a system is NDS if, for a certain strategy $\Pi$, all the low level views are still possible, i.e., if $\Pi$ does not restrict the executions of low level agents. This is often summarised by saying that $\Pi$ does not "interfere" with the low level.

The original definition [17] is based on a very restrictive system model in which, at every clock signal, a system takes an input from every input channel and emits an output to every output one. In such a model, a strategy is simply a function which at time $n$ computes a new input using a $n-1$ long sequence of previous input/output actions. In a CCS-like setting, a strategy cannot be represented as a function. In fact, when a system emits an output, it must wait for a corresponding external input which could never arrive: a deadlock may arise on a system output, because of the synchronous nature of CCS communication. A strategy can be represented as a function of previous input/output signals only if all such outputs have been accepted by the external user (in particular by a Trojan Horse).

In our framework, a strategy may be seen as a process $\Pi$ which uses only high level actions. To fix a certain strategy on a system $E$, it is sufficient to consider the system $(E \bowtie \Pi) \setminus \text{Act}_H$. So a system $E$ is NDC if for every high level process $\Pi$ the low level view sets of $E$ and $(E \bowtie \Pi) \setminus \text{Act}_H$ are the same. In other words, a system $E$ is NDC if the composition of $E$ with all possible Trojan Horses does not change its set of low level views.

**Definition 3.1** $\text{NDC}_H$ (Non Deducible on Compositions for actions in $H$).

Let $H \subseteq \text{Act}_H$. Then,

$$E \in \text{NDC}_H \iff \text{lowviews}(E) = \text{lowviews}((E \bowtie \Pi) \setminus H),$$

$$\forall \Pi \in \mathcal{E}, \mathcal{L}(\Pi) \subseteq \text{Act}_H \cup \{I\}$$

NDC is defined as $\text{NDC}_H$ where $H = \text{Act}_H$: $\text{NDC} \stackrel{\text{def}}{=} \text{NDC}_{\text{Act}_H}$.

This definition states that there is no way of altering the low level executions of a system $E$ by composing with it a high level agent $\Pi$ which communicates with $E$ only through actions in $H$. Of course, this definition is difficult to use in practice: it is not possible to verify the security of a system connecting to it all Trojan Horses and observing their effect on low level computations (it is an exhaustive procedure which does not terminate, in general). We need an alternative formulation of NDC, which exploits local information, only. It is based on the following theorem, which asserts that the property of ‘transparence with respect to all Trojan Horse attacks’ corresponds to a static property of an agent $E$, namely, to a property that can be verified by examining the traces of system $E$ only.

**Theorem 3.1** (Static characterization of $\text{NDC}_H$)

$$E \in \text{NDC}_H \iff \text{lowviews}(E) = \text{lowviews}(E \setminus H)$$

This characterisation of $\text{NDC}_H$ has the merits of being algorithmically testable (at least, for finite state systems) and useful for modular verification of secure systems. Indeed, NDC is
4 An Example: Access Monitor

Figure 2. Access monitor

a \textit{composable} property w.r.t. the CCS operators of parallel composition and restriction; hence, a new NDC secure system may be described as a composition of NDC secure subsystems.

\textbf{Theorem 3.2} Let \( E \) and \( E' \) be two \( \text{NDC}_H \) agents, \( S \subseteq H \) and \( S' \subseteq \text{Act}_L \). The following hold:

\begin{enumerate}
  \item \( E \mid E' \in \text{NDC}_H \),
  \item \( E \setminus S \in \text{NDC}_H \),
  \item \( E \setminus S' \in \text{NDC}_H \),
  \item \( (E \mid E') \setminus (L(E) \cap L(E')) \in \text{NDC}_H \), if \( (L(E) \cap L(E') \cap \text{Act}_H) \subseteq H \).
\end{enumerate}

Finally, the following theorem states that the \( \text{NDC}_H \) sets are partially ordered w.r.t. set-theoretic (reversed) inclusion over the sets \( H \).

\textbf{Proposition 3.1} \( H, H' \subseteq \text{Act}_H : H' \subseteq H \implies \text{NDC}_H \subseteq \text{NDC}_{H'} \)

4 An Example: Access Monitor

In this section, the NDC property is illustrated through an example: an Access Monitor (AM for short, see figure 2), which is a process managing accesses to a certain resource. As a resource can be viewed as a set of objects containing information, then an AM is a very important element in the implementation of a multilevel security policy (it is sufficient to know all security levels of all subjects and objects to correctly manage every access).

Let us consider the usual multilevel policy (Read-down, Write-up). Let \( SL \) be the set of security levels and \( \leq \subseteq SL \times SL \) be a total order relation. Let \( ID \), ranged over by \( id \), be the set of user identifiers; let also \( O \), ranged over by \( x \), be the set of object identifiers. Finally, let \( lvl : ID \cup O \rightarrow SL \) be a function which returns the security level of a certain user \( id \) or object \( x \). If objects are variables (every object contains a value), then AM can be implemented as follows:

\[
\text{Monitor} \overset{\text{def}}{=} \sum_{id \in ID} \text{access}_{\text{r}}(id). (\text{if} \text{ lvl}(x) \leq \text{ lvl}(id) \text{ then } \text{read}_{x}(y).(\text{value}_{id}(y) \mid \text{Monitor}) \text{ else } \overline{\text{value}_{id}(ERROR)} \mid \text{Monitor}) + \sum_{id \in ID} \text{access}_{\text{w}}(id, z). (\text{if} \text{ lvl}(x) \geq \text{ lvl}(id) \text{ then } \text{write}_{x}(z) \mid \text{Monitor})
\]
Signals read and write are used by the Monitor for accesses to objects. They are parametrized w.r.t. the names of the objects. Signal \( \text{access}_{x_{id}}(x) \) represents a request for a read operation to object \( x \) made by the user with identifier \( id \). The request is always accepted and a value is always returned to the user. However, the value stored in the object \( x \) is returned only if the security level of \( x \) is less than, or equal to, the security level of the user \( id \); otherwise, an error message is transmitted.\(^5\) Note that the Monitor is ready to accept new signals before delivering the value to the user, because of the (inner) parallel composition. Signal \( \text{access}_{y_{id}}(x, z) \) represents a request, made by a user \( id \), to write the value \( z \) into object \( x \). This request is always accepted, but the write operation is really performed only if the level of \( x \) is greater than the level of \( id \).

The objects can be implemented as follows:

\[
\text{Obj}_x(y) \overset{\text{def}}{=} \text{read}_x(y).\text{Obj}_x(y) + \text{write}_x(z).\text{Obj}_x(z)
\]

If there are \( n \) variables then the system is:

\[
\text{Access}_{\text{monitor}} \overset{\text{def}}{=} (\text{Monitor} \mid \text{Obj}_1(0) \mid \ldots \mid \text{Obj}_n(0)) \setminus L
\]

where

\[
L = \bigcup_{x \in [1,n]} Acs(\text{Obj}_x)
\]

\[
Acs(\text{Obj}_x) \overset{\text{def}}{=} \{\text{read}_x, \text{write}_x\}
\]

By restricting the executions on set \( L \), we make objects not directly accessible by a user, but filtered through the interactions with the Monitor. So the user \( id \) which interacts with the Monitor can execute this set of actions:

\[
Acs_{id}(\text{Access}_{\text{monitor}}) = \{\text{access}_r_{id}, \text{access}_w_{id}, \text{value}_{id}\}
\]

The set of all the actions accepted by the monitor is:

\[
Acs(\text{Access}_{\text{monitor}}) = \bigcup_{id \in ID} Acs_{id}(\text{Access}_{\text{monitor}})
\]

Since NDC is closed with respect to composition of systems then there are two different approaches to verify if Access\(_{\text{monitor}}\) is NDC secure:

(i) Verify if the whole system is NDC, or
(ii) decompose system in subsystems and verify if everyone is NDC (a possible decomposition is suggested by system definition itself: Monitor and Obj\(_x\)'s).

In the full paper (and in [2]), we prove that AM is NDC (following the first approach), while Monitor is not secure for every kind of object. In particular, we report an example of finite storage device management in which a user can transmit information using the resource full error message. However, the compositionality property of NDC is important when the monitor is used: as a user is an agent which executes only actions of the same level, then it is NDC. So, by connecting to AM any number of users, we obtain an enlarged system which is still NDC secure.

\(^5\) Obviously, a user must get the replay to a certain read request before asking monitor for another read: a restriction we consider for the sake of simplicity.
Extending the Approach to Bisimulation Semantics

NDC is based on trace semantics: it requires that a system, to be secure, gives to a low user the same low trace sets for every high execution. More precisely, this statement is formalized through the following theorem.

Theorem 5.1

\[ \mathcal{E} \in N D C \iff \mathcal{E} \setminus \mathcal{A}_{n} \approx_{T} (\mathcal{E} \ | \ \Pi) \setminus \mathcal{A}_{n}, \forall \Pi \in \mathcal{E}, \mathcal{L}(\Pi) \subseteq \mathcal{A}_{n} \cup \{\tau\} \]

Unfortunately, trace semantics is rather weak; it is unable to distinguish systems which give different observations to a user. For instance, systems \( A = ab + a \) and \( B = ab \) are trace equivalent but the first can deadlock after a signal \( a \). Hence, a low user could distinguish two low trace equivalent systems and so have information about high level users. As a simple example, let us implement the monitor as follows:

\[
\begin{align*}
\text{Monitor} & \text{ def } \sum_{id \in ID} \text{access}_{r}(x) \cdot (\text{if } lvl(x) \leq lvl(id) \text{ then } \text{read}_{x}(y) \cdot \text{value}_{id}(y) \text{ else } \text{value}_{id}(\text{ERROR})) \\
& + \sum_{id \in ID} \text{access}_{w}(x, z) \cdot (\text{if } lvl(x) \geq lvl(id) \text{ then } \text{write}_{x}(z))
\end{align*}
\]

Surprisingly enough, it still satisfies the NDC property, but, of course, it is not multilevel secure. Indeed, a high level user can block the monitor – and so interfere with low level users – by making a read request and then refusing to accept the corresponding answer. Trace equivalence does not detect this kind of “attacks” and so it is necessary to substitute it with a stronger one. One possibility is weak bisimulation [13] which is the “standard” CCS equivalence and implies other important equivalences such as testing [5] and failure [6]. We denote it with \( \approx_{B} \) (see section 2). The new security notion is called BNDC.

Definition 5.1

\[ \mathcal{E} \in B N D C \iff \mathcal{E} \setminus \mathcal{A}_{n} \approx_{B} (\mathcal{E} \ | \ \Pi) \setminus \mathcal{A}_{n}, \forall \Pi \in \mathcal{E}, \mathcal{L}(\Pi) \subseteq \mathcal{A}_{n} \cup \{\tau\} \]

One can easily verify that the second version of Monitor does not enjoy the BNDC property.

A static characterization of BNDC – which does not involve composition with every processes \( \Pi \) – is not immediate. A first idea could be to elaborate on the same condition we proposed for NDC. The resulting proposal is the following:

\[ \mathcal{E} \setminus \mathcal{A}_{n} \approx_{B} \mathcal{E} \]

where \( / \) denotes a hiding operator which transforms all high actions in internal \( \tau \)’s. Unfortunately, it is easy to see that it is only a necessary condition for \( \mathcal{E} \) to be BNDC. At present, we have not yet found a static characterization of BNDC. We think this is a very interesting problem, left for future research.

To conclude this paper we propose a sufficient condition to BNDC, which is static and compositional (with \( \mathcal{E} \Rightarrow \mathcal{E}' \) we denote that \( \mathcal{E}' \) is reachable from \( \mathcal{E} \), that is \( \exists \gamma \) such that \( \mathcal{E} \xrightarrow{\gamma} \mathcal{E}' \)):
6 Conclusions

Definition 5.2 Static Bisimulation NDC (SBNDC):

\[ E \in \text{SBNDC} \iff \forall E' : E \Rightarrow E', \forall E'' : \exists h \in \text{Act}_H, E' \xrightarrow{h} E'' \text{ then } E' \setminus \text{Act}_H \approx_B E'' \setminus \text{Act}_H \]

Proposition 5.1 The following hold:

(i) \( \text{SBNDC} \subseteq \text{BNDC} \),

(ii) \( E, F \in \text{SBNDC} \implies (E \parallel F) \in \text{SBNDC} \),

(iii) \( E \in \text{SBNDC} \Rightarrow E \setminus S \in \text{SBNDC} \), if \( S \subseteq I \).

6 Conclusions

This paper presents a formal approach to information flow security, in the framework of process algebraic languages like CCS. NDC is a security property which guarantees that no information flow from high level to low level can occur (hence, ensuring the absence of covert channels). NDC is closed with respect to system composition and can be used to verify if a CCS agent represents a multilevel secure system.

It has been observed that such a property may be too weak in some cases, essentially because it is related to trace equivalence; hence, a stronger property, based on bisimulation [13], has been proposed. Future research is concerned with a static characterization of BNDC and further extensions to other well-known equivalences (e.g., testing [5], failure [6], ...).

The final goal of the present research is the implementation of a verification tool, in order to test such extensions on a number of significative examples.

References

REFERENCES


