Non-interference properties for nondeterministic processes

Richard Forster

1. Introduction

Given a system which has multiple users interacting with it, for example a file system or a key-exchange protocol, we may be interested in preventing information from flowing between the users. As well as preventing the users from directly communicating, such as sending messages, or deliberate indirect communication, such as creating a file containing such information, we need to prevent information flowing through covert channels. These channels of communication may be accidental as well as non-obvious. They may arise from an apparently unrelated sequence of events or a contention for resources, and might manifest themselves long after the events that caused them to occur.

For example we might have a simple filing system which allows users to create, read and destroy files. By restricting read and destroy actions on files to the user who has created the file we can remove the obvious communication channels. All the users may still be able to see the existence of all the files, and since this could communicate information we need to prevent this. If a user tries to create a file that already exists, although created by another user an unexpected error may be received. Similarly a user might fill up some common storage space, causing another unexpected error when another user tries to create a file. In both these cases a contention for resources leads to a covert channel.

Information flow might occur because two users are manipulating the system in an attempt to communicate. If might also occur because a user is trying to gain information about/from another user without their assistance. We need to prevent both types of communication, as well as the cases where the system itself is providing the information without prodding. These can be treated as the same problem — that no undesirable information flow can ever arise from any operation of the system.

Systems do not need to be complicated for (potentially complicated) covert channels to arise. It therefore makes sense to formally model systems, and specify security properties, in our attempt to show systems secure. We might look at them as transition systems, or write them in a process algebra such as CSP[6] or CCS[7]. In such a form we may be able to use an automatic verification tool, such as FDR[5] or the Security Checker[4] , to prove or disprove our security properties.

Given a process P that we want to show secure (ie. without undesirable information flow), we can partition its alphabet into disjoint subsets H and L. These subsets represent the available actions of a high and low-level user respectively. If we think of the high-level user as having access to higher classified information than the low-level user, it makes sense to prevent any information from flowing from the high-level user to the low-level user. In other words the low-level user must be unable to infer any information about how the high-level user is interacting, or has interacted, with P.

We restrict ourselves to looking at the asymmetric flow from high to low. Information is allowed to flow from the low-level user to the high-level user. This is sufficient. If we require no information to flow in either direction we can additionally impose our security properties on the system with the roles of the high-level and low-level users reversed.

In a similar way we can restrict our security properties to systems which interact with two users. Systems with larger numbers of users can be dealt with by considering alternate partitioning of the alphabet. Given a hierarchy of users we can consider, for each user, the user together with the other users we are permitted communication with, against the users that must not be communicated with.

A variety of different security conditions have been proposed by different authors. One of the difficulties that has arisen is that of specifying exactly what we mean by 'secure', and many definitions that seemed adequate can be shown to permit subtle and previously unthought of information flows. In the next few sections several current properties will be discussed, and two new ones introduced.

Section 3 presents conditions based on abstracting processes, which can be used to look at deterministic processes. Section 4 introduces some recent conditions for nondeterministic processes, and section 5 looks at how these deal with a variety of nondeterministic examples that exhibit subtle information flows. Section 6 introduces the new non-interference conditions, looks at some properties they exhibit, and sees how they handle the previous examples. Section 7 demonstrates that the new conditions correspond with the existing conditions for the deterministic processes. Section 8 discussed the refinement paradox, a problem which all nondeterministic conditions, including the new ones, appear to suffer from.

2. Notation and Definitions

Systems will be modelled as labelled transtition systems[1]. Transition systems consist of a set of possible states, and the potential transitions between states. The transitions within the system are labelled with the action/event which triggers the transition.

Definition 2.1 Labelled transition system

A *labelled transition system* is a 5-tuple $A =< S, T, \alpha, \beta, \lambda >$ where

- S is a finite or infinite set of *states*,
- T is a finite or infinite set of *transitions*,
- α and β are two mappings from T to S which take each transition $t \in T$ to the two states $\alpha(t)$ and $\beta(t)$, respectively the *source* and the *target* of the transition t.
- λ is a mapping from T to A taking each transition t to its label $\lambda(t)$.

It often makes sense to think of the initial state of a transition system. If P is a state in a transition system we can refer to the process P, meaning the transition system where P is the initial state.

Labelled transition systems will often be illustrated by diagrams. States are shown as circles, and the transitions between them by lines. Arrows on the transition lines point from the source (α mapping) to the target (β mapping) of the transition. Identifiers within circles indicate specifically named states. Two circles with the same (non-blank) indentifier indicate the same state, wheras all unidentified states should be treated as distinct. Transition lines are marked by their label, with τ actions being represented by the absence of a mark.

$P \xrightarrow{\mu} P'$ means we can get from state P to state P' by a transition labelled with μ. $P \stackrel{\mu}{\Longrightarrow} P'$ is a shorthand notation for $P(\xrightarrow{\tau})^* P_1 \xrightarrow{\mu} P_2(\xrightarrow{\tau})^* P'$, where $(\xrightarrow{\tau})^*$ stands for a sequence of τ labelled transitions, which may be empty. The notation $P \stackrel{\mu}{\not\Longrightarrow}$ means that $\not\exists P'$ such that $P \stackrel{\mu}{\Longrightarrow} P'$. The '$\Longrightarrow$' notation is also extended to sequences of actions. If $\delta = \delta_1\delta_2 \ldots \delta_n$ then $P \stackrel{\delta}{\Longrightarrow} P'$ means $\exists\, P_1, \ldots, P_{n-1}$ such that $P \stackrel{\delta_1}{\Longrightarrow} P_1 \stackrel{\delta_2}{\Longrightarrow} \cdots \stackrel{\delta_{n-1}}{\Longrightarrow} P_{n-1} \stackrel{\delta_n}{\Longrightarrow} P'$.

The transitions systems looked at in the following are specified using both CSP and *Security Process Algebra* (SPA). The latter is an extension of CCS with two additional operators; the hiding operator of CSP and an additional input restriction operator. Since CSP hiding and CCS restriction are normally written in the same way the following notation has been used : $P \setminus H$ (CSP hiding) is the process P with the H actions hidden, and $P \setminus_H$ (CCS restriction) is the process P that can execute all the actions of P provided they do not belong to H. P/h is the CSP process P after h. Since the CSP and SPA notations are being used to specify labelled transition systems we will switch freely between them.

In the following definition $P \stackrel{\hat{\alpha}}{\Longrightarrow} P'$ stands for $P \stackrel{\alpha}{\Longrightarrow} P'$ if $\alpha \neq \tau$, and $P(\xrightarrow{\tau})^* P'$ if $\alpha = \tau$.

Definition 2.2 Weak Bisimulation

A relation $R \subseteq \mathcal{E} \times \mathcal{E}$ is a *weak bisimulation* if $(P, Q) \in R \Rightarrow \forall\, \alpha \in \Sigma$,

(i) Whenever $P \xrightarrow{\alpha} P'$ then, for some Q', $Q \stackrel{\hat{\alpha}}{\Longrightarrow} Q'$ and $(P', Q') \in R$.

(ii) Whenever $Q \xrightarrow{\alpha} Q'$ then, for some P', $P \stackrel{\hat{\alpha}}{\Longrightarrow} P'$ and $(P', Q') \in R$.

Throughout this document *bisimilar* will always refer to weak bisimulation.

Definition 2.3 Determinism

A process P is *deterministic* if $divergences[\![P]\!] = \{\}$ and $s\frown t \in Traces[P]$ implies $(s, \{t\}) \notin Failures[P]$.

A process is deterministic if it is non-divergent, and can never refuse an event that it can accept. A process that does not satisfy this property is said to be *nondeterministic*.

Finally we will define two standard CSP processes. CHAOS_H is the most nondeterministic divergence free process that communicates in H. RUN_H is a process that is always willing to communicate any event in H.

Definition 2.4 CHAOS_H

$\mathrm{CHAOS}_H = \mathrm{STOP} \sqcap \mathrm{x}{:}H \rightarrow \mathrm{CHAOS}_H$

Definition 2.5 RUN_H

$\mathrm{RUN}_H = \mathrm{x}{:}H \rightarrow \mathrm{RUN}_H$

3. Lazy abstraction

We are concerned with what information the low-level user can determine, and it therefore makes sense to consider how the system is observed by the low-level user. An abstracted version of a process P can be considered, where the only observable events are those in L. The question is what type of abstraction do we use. Security properties based on *eager* and *lazy* abstractions were introduced in [9, 10], and are discussed in [11, 12].

One possible abstraction is to hide all the events in H. Since these actions become τ events any state that was previously reachable, even if it required a high-level action to occur, is still available. This abstraction is eager because it assumes the high-level user is always prepared to communicate. The abstracted process may allow low-level actions that occur after high-level actions, since we are unable to delay τ actions.

Definition 3.1 Eager abstraction

$\mathcal{E}_L(P) = P \setminus H$

Example 3.2

Let $L = \{a, b\}$ and $H = \{x\}$ in

$$\begin{aligned} P \quad = \quad & a \rightarrow P \\ & \Box\, b \rightarrow P \\ & \Box\, x \rightarrow (a \rightarrow P \,\Box\, b \rightarrow P) \end{aligned}$$

then

$$\mathcal{E}_L(P) = (a \rightarrow \mathcal{E}_L(P) \,\Box\, b \rightarrow \mathcal{E}_L(P)) \sqcap (a \rightarrow \mathcal{E}_L(P) \,\Box\, b \rightarrow \mathcal{E}_L(P))$$

Hiding H allows the x events to be ignored, and these events are absent in the eager abstraction . This is a process we would like to think of as secure, since the low-level user is always able to communicate a and b events.

Example 3.3

Let $L = \{a, b\}$ and $H = \{x\}$ in

$$Q = x \rightarrow (a \rightarrow Q \,\Box\, b \rightarrow Q)$$

then

$$\mathcal{E}_L(Q) = (a \rightarrow \mathcal{E}_L(Q) \,\Box\, b \rightarrow \mathcal{E}_L(Q))$$

This is a process which we would not think of as secure. The availability of a and b events to the low-level user implies that a high-level event has been communicated. The eager abstraction of this process is equivalent to the eager abstraction of the secure process P in example 3.2, so this abstraction cannot distinguish between some secure and insecure processes.

As well as being invisible to the low-level user the actions in H are under the control of the high-level user. We cannot rely on the high-level user always being willing to communicate; the high-level user being free to choose whether to communicate, as well as what to communicate. The eager abstraction is therefore not sufficient for our needs. If we allow the high-level user to choose which events to refuse, and hence remove this eagerness to communicate, we can formulate a lazy abstraction.

Definition 3.4 Lazy abstraction (failures-divergences model)

If P is a divergence-free process, the *projection* of P into the alphabet L is the divergence-free process $P \upharpoonright L$ with failure-set $\{(s \setminus H, X) \mid (s, X \cap L) \in failures(P)\}$. The *lazy abstraction* (for the failures-divergences model) $\mathcal{L}_L(P)$ is defined to be $P \upharpoonright L$.

Lazy abstraction ensures every subset of the high-level actions can be refused before the hiding occurs, without introducing divergence if there is unbounded high-level behaviour.

The high-level user's actions should not interfere with the low-level user, so it is necessary that the low-level user's view of the system cannot be altered by the occurrence of events in H. For a deterministic process this can be specified by saying the low-level abstracted view of the system is deterministic. Since internal actions in this abstraction can be considered as under the control of the high-level user, a deterministic abstraction implies no behaviours of the high-level user can effect the low-level user.

Definition 3.5 Lazily independent (lazily secure)

Let P be a non-divergent, finitely nondeterministic and $\surd$-free process. P is said to be *lazily independent* in L, written $\mathcal{L}IND_L(P)$ if and only if $\mathcal{L}_L(P)$ is deterministic.

Suppose that $\mathcal{L}IND_L(P)$ holds. For any divergence-free process Q whose alphabet is H we get, since $\mathcal{L}_L(P)$ is deterministic it is equivalent to $\mathcal{L}_L(P \,|[\, H \,]|\, Q)$.

Example 3.6

The lazy abstractions P' and Q' of the processes defined in examples 3.2 and 3.3 are

$$\begin{aligned} P' &= (a \to P' \,\Box\, b \to P') \\ &\quad \sqcap (a \to P' \,\Box\, b \to P') \\ Q' &= \text{STOP} \sqcap (a \to Q' \,\Box\, b \to Q') \end{aligned}$$

These processes no longer have the same abstraction. Process P' is equivalent to the process $\mu\, R.(a \to R \,\Box\, b \to R)$ that continually offers the events a and b to the low-level user. Process Q' can continually offer a and b events, but after any number of them it can nondeterministically choose to stop.

Process P satisfies the lazily secure condition, since P' is deterministic, and this is a process we would like to think of as secure. The nondeterministic behaviour of Q' means process Q is not lazily secure, and this is a process we have argued as being insecure.

In the original treatment a lazy abstraction was produced by obfuscating the process; interleaving P with RUN_H. This made H actions ambiguous, which effectively imitated actions being delayed. The occurrence of an action $h \in H$ might be indicating a high-level action taking the system to a new state, or a 'false' event / delay, returning the system to an equivalent state. A non-divergent, deterministic processes was said to be lazily secure if this abstraction, $P \mathbin{|||} \mathrm{RUN}_H$, was deterministic.

There are two main disadvantages to the older abstraction when looking at security. Firstly it cannot be said to be how the low-level user observes the system since all of the process's H events, as well as those introduced by RUN_H are visible. Secondly the additional transitions produce a much larger system for an automatic verifier such as FDR to check, and one that "does not lead itself nearly as well to facilitating compression"[11]. This effects both the size of the problems that can be dealt with, and the time taken to check them.

The interleaving abstraction has the advantage that it is defined in terms of the standard interleaving operator. FDR implements the $|||$ operator directly, and can verify whether or not a process is deterministic. Despite producing larger systems than the new abstraction, the interleaving abstraction is therefore the one usually used for automatic verification. We can use this abstraction because of the following result [12] :

Theorem 3.7

The lazy abstraction $\mathcal{L}_L(P)$ is deterministic if and only if $P \mathbin{|||} \mathrm{RUN}_H$ is deterministic.

The given lazy abstraction is for the failures-divergences model, and under different models the definition changes. This is not too surprising since, for example, the infinite traces-failures-divergences model can distinguish more processes than the failures-divergences one.

For the infinite traces-failures-divergences model the lazy abstraction is defined as $(P \parallel SemiFair(L, H)) \setminus H$, where $SemiFair(L, H)$ is a process which allows any finite combinations of L's and H's, can never refuse events in L, and will only allow an infinite number of H events to occur if infinitely many L events occur. Under the stable failures model the lazy abstraction is simply $(P \parallel \mathrm{CHAOS}_H) \setminus H$.

For a divergence-free, finitely nondeterministic process P, the determinism of the lazy abstractions for the failures-divergences, stable failures, and infinite traces-failures-divergences models can be shown to be equivalent [11]. This is also true of the older abstraction. In other words if we have shown there can be no information flow under one of these models we also know there will be no information flow even if the low-level user is observing the system using one of the alternative models.

The conditions based on determinism can be considered too strong, since they only allow processes that appear deterministic to the low-level user to be classed as secure. Nondeterminism in a low-level abstraction need not have been caused by potential high-level behaviours, and hence introduced by the abstraction. There is no reason why a nondeterministic process should not be secure, and there are certainly nondeterministic processes that it would seem reasonable to class as secure. To take an extreme example, any process that just contains low-level actions would seem to be secure.

4. Bisimulation Non Deducibility on Compositions

The *Bisimulation Non Deducibility on Compositions* (BNDC) property was proposed in [2] by Focardi and Gorrieri. It is a security property which tries to deal with nondeterministic processes. The property is based on comparing the low-level user's views of the process for all possible behaviours of the high-level user. A system E is BNDC if for every high-level process Π a low-level user is unable to distinguish between $(E \mid \Pi) \setminus_H$ and E, two processes being treated as indistinguishable if they are weakly bisimilar. Once again we require the low-level user's view of the system should be unaffected by whatever actions the high-level user may take.

Definition 4.1 (BNDC)

$E \in \text{BNDC} \Leftrightarrow \forall \Pi \in \mathcal{E}_H, E \setminus H \approx_B (E \mid \Pi) \setminus_H$.

An additional property was proposed in [4] — *Strong Bisimulation Strong Nondeterministic Non Interference* (SBSNNI). BNDC is difficult to deal with since it requires composition with every high-level process Π, and no static characterisation is known. SBSNNI was proposed as a sufficient condition for BNDC which avoids universal quantification. SBSNNI is defined using a similar property — *Bisimulation Strong Nondeterministic Non Interference* (BSNNI). A process satisfies BSNNI if hiding the H actions is the same as restriction the process from them, and a process satisfies SBSNNI if each state satisfies the previous property.

Definition 4.2 (BSNNI)

$E \in \text{BSNNI} \Leftrightarrow E \setminus H \approx_B E \setminus_H$

Definition 4.3 (SBSNNI)

$E \in \text{SBSNNI} \Leftrightarrow \forall E'$ such that $\exists \gamma, E \stackrel{\gamma}{\Longrightarrow} E'$ we have $E' \in \text{BSNNI}$.

SBSNNI is strictly stronger than BNDC. In [2] it is proved :

Theorem 4.4

$\text{SBSNNI} \subset \text{BNDC} \subset \text{BSNNI}$.

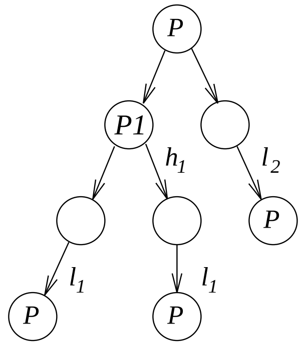
BSNNI is not a sufficiently strong security condition, since it cannot detect some deadlocks that can be caused by the high-level user. For example, the deterministic process $P = l_1 \rightarrow P \,\square\, h_1 \rightarrow h_1 \rightarrow l_1 \rightarrow P$ satisfies it, despite preventing L actions immediately after its first h_1 event. BNDC was introduced to try and solve this problem.

5. Nondeterministic examples

The following nondeterministic examples will be used to demonstrate some subtle information flows. We will consider how the bisimulation non-deducibility conditions treat these systems. Since all the following examples are nondeterministic the lazy abstraction properties are not considered. In each of the following examples $H = \{h_1, h_2, \ldots\}$ and $L = \{l_1, l_2, \ldots\}$.

Example 5.1

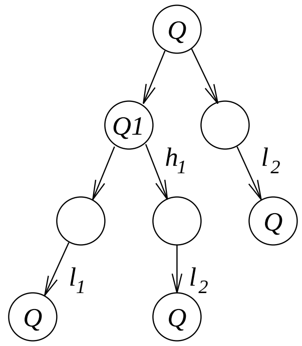
$$P = \tau.(\tau.l_1.P + h_1.l_1.P) + \tau.l_2.P$$



P is a process we would probably view as secure. If the process is in a state where the high-level user is able to communicate an h_1 action, neither the occurrence or non-occurrence of this action affects what the low-level user can observe. [The process is equivalent to $P' = \tau.(\tau + h_1).l_1.P' + \tau.l_2.P'$.] This process satisfies SBSNNI, and hence BNDC by theorem 4.4.

Example 5.2

$$Q = \tau.(\tau.l_1.Q + h_1.l_2.Q) + \tau.l_2.Q$$

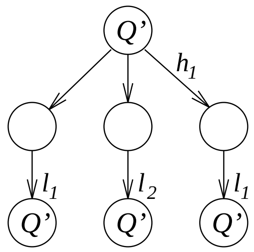


This process, whilst very similar to the previous one, is not secure. If the low-level user is able to communicate an l_2 action is cannot be determined whether this is the result of an h_1 action, or an internal choice. Whenever the system is able to communicate an l_1 action however, the low-level user knows for certain that the high-level user has not communicated an h_1. This may be because the system communicated the τ event before the high-level user could communicate an h_1, or that the high-level user chose not to execute this event. There is still information flow because the low-level user knows an event that might have happened did not — the file was not created, the missile was not launched etc...

This processes fails to satisfy SBSNNI. Q fails SBSNNI because state $Q1$ fails BSNNI — $\langle l_2 \rangle \in traces(Q1 \setminus H)$ but $\notin traces(Q1 \setminus_H)$. The process does satisfy BNDC since letting $E = \tau.l_1.E + \tau.l_2.E$ we have $\forall \Pi \in \mathcal{E}_H, Q \setminus H \approx_B E \approx_B (Q \mid \Pi) \setminus_H$.

Example 5.3

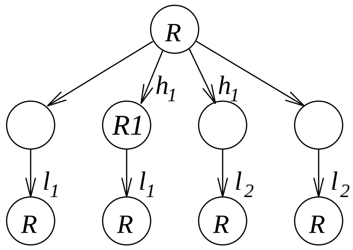
$$Q' = \tau.l_1.Q' + \tau.l_2.Q' + h_1.l_1.Q'$$



Q' demonstrates the same problem as Q. The ability for the low-level user to communicate an l_2 action implying that the high-level user has not just communicated an h_1. This process satisfies SBSNNI however, and hence BNDC.

Example 5.4

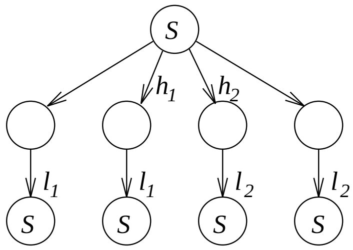
$$R = \tau.l_1.R + \tau.l_2.R + h_1.l_1.R + h_1.l_2.R$$



R is another process we would probably view as secure. If the high-level user is able to, and chooses to, communicate an h_1 action a nondeterministic choice is still made, leading to either a state where an l_1 action is available, or one where an l_2 action is available. These states are equivalent to the ones reached by the nondeterministic choice made if the high-level user is unable, or unwilling, to communicate. This process satisfies SBSNNI and hence BNDC.

Example 5.5

$$S = \tau.l_1.S + \tau.l_2.S + h_1.l_1.S + h_2.l_2.S$$



This process is not secure. Whilst the availability of an l_1 or an l_2 action does not indicate whether the corresponding action in H has just occurred, any actions of the high-level user will be mirrored in the actions seen by the low-level user. S satisfies SBSNNI for exactly the same reason R does, because the property does not differentiate between different events in H. $S \setminus_H \approx_B R \setminus_H$ and $S \setminus H \approx_B R \setminus H$. Since S satisfies SBSNNI it also satisfies BNDC.

Example 5.6

$$\begin{aligned} T &= \tau.T_1 + h_1.T_2 \\ T_1 &= l_1 \to T_1 \sqcap \cdots \sqcap l_n \to T_1 \\ T_2 &= h_1.l_1.T_2 + \cdots + h_n.l_n.T_2 \end{aligned}$$

T is a process that is insecure. If initially an h_1 actions occurs then the high-level user will control exactly what the low-level user is able to communicate. This process does not satisfy BNDC but it does satisfy BSNNI. This occurs because the restriction operator if BSNNI's definition causes it to treat states after an initial actions in H differently to those that occur before one.

All of these processes, except T, satisfy BNDC because in each case the actions of the high-level user take the system to a state equivalent to one reachable by some internal nondeterministic choice. A process E is BNDC if composing any high-level processes with E does not modify what the low-level user sees of the system, so a system satisfies BNDC if the behaviours causable by the high-level user are potential system behaviours without high-level interaction. The fact that the observed behaviours might tell the low-level user what the high-lever user has not done is not recognised.

For processes Q and Q' an action in H is able to resolve some nondeterminism in the system. The actions of the high-level user can take the system to states equivalent to a subset of those states reachable by the nondeterministic choices. The low-level user can therefore infer the absence of H events, when states outside this subset are reached.

The high-level user is again able to resolve some nondeterminism in process S. In this case, equivalent states to all the states reachable by the nondeterministic choices are reachable from the high-level user's actions. Since these states can be differentiated the low-level user observes information about the high-level user, even though it cannot be decided whether a particular piece of information was caused by the high-level user or the system's nondeterminism.

Example 5.7

In general, given any process P we can construct a process P' which satisfies BNDC. Let $S = \{(P \parallel Q) \setminus H \mid \alpha Q = H\}$, ie. S is the set of system views the low-level user may see. Let $P' = (\bigsqcap S) \sqcap P$. This process is BNDC, but is insecure whenever P is, since communication with H tells the low-level user whatever P does.

Proof (P' is BNDC)

For P' to be BNDC, $\forall \Pi \in \mathcal{E}_H$ we require $P' \setminus H \approx_B (P \mid \Pi) \setminus_H$. Now $P' \setminus H \approx_B \bigsqcap S$, since $P \setminus H \approx_B (P \parallel \mathrm{RUN}_H) \setminus H$ and $(P \parallel \mathrm{RUN}_H) \setminus H \in S$. No process in S contains any H actions, so $(P' \mid \Pi) \setminus_H \approx_B (\bigsqcap S) \sqcap (P \mid \Pi) \setminus_H$. Now $(P \parallel \bar{\Pi}) \setminus H \approx_B (P \mid \Pi) \setminus_H$. Since $(P \parallel \Pi) \setminus H \in S$, we have $(P' \parallel \Pi) \setminus_H \approx_B \bigsqcap S$. Therefore, $P' \setminus H \approx_B \bigsqcap S \approx_B (P \mid \Pi) \setminus_H$, ie. P' is BNDC.

$\square$

Neither BNDC or SBSNNI appear to be strict enough, since they incorrectly identify some insecure processes as secure. The problem with BNDC seems particularly severe, since we can construct a supposedly secure process which is capable of leaking whatever information the high-level user requires to the low-level user.

6. New conditions : (strong) local non-interference

If there is no information flow the low-level user's view of the system must be unaffected by the high-level user's actions. One way of specifying this is to say that whenever a high-level action takes place we are taken to a state that is indistinguishable (to the low-level user) to the state we were in before the action.

Given a state σ in a process that has a transition (labelled by an element of H) to a state σ', we can consider what the process σ can do before communicating an event in H. Since we require high-level actions to take us to states that are indistinguishable to the low-level user, it is necessary that this is equivalent to what the process σ' can do before communicating an event in H. This is the idea behind *strong local non-interference*; for every such pair of states we need to show such an equivalence holds.

Definition 6.1 (C1 — strong local non-interference)

E satisfies strong local non-interference, written $\mathcal{C}1_L(E)$, if $\sigma \xrightarrow{h} \sigma'$, where $h \in H$ and $\sigma, \sigma' \in \mathrm{states}(E)$, implies $\sigma \setminus_H \approx_B \sigma' \setminus_H$.

This condition might be considered too strict, since it fails to take into account nondeterminism arising from identically labelled H transitions leading from the same state. When the high-level user communicates such an action there is no choice made over which transition takes place, and it seems reasonable to consider this nondeterministic choice after the event is communicated. *Local non-interference* tries to solve this problem.

Definition 6.2 (C2 — local non-interference)

E satisfies local non-interference, written $\mathcal{C}2_L(E)$, if $\sigma \xrightarrow{h} \sigma_1, \ldots, \sigma \xrightarrow{h} \sigma_n$, where $h \in H$ and $\sigma, \sigma_1, \ldots, \sigma_n \in \mathrm{states}(E)$, implies $\sigma \setminus_H \approx_B \tau.\sigma_1 \setminus_H + \ldots + \tau.\sigma_n \setminus_H$.

This condition effectively rewrites the process so no state has identically labelled H actions leading from it. The rewritten process is then considered by the strong local non-interference condition. It seems reasonable to do this rewrite, even though $h.A + h.B$ and $h.(\tau.A + \tau.B)$ are not weakly bisimilar. This is because $(h.A + h.B)/h$ and $h.(\tau.A + \tau.B)/h$ are weakly bisimilar.

Example 6.3 (Examples 3.2 & 3.3 revisited)

- $P = a \to P \,\square\, b \to P \,\square\, x \to (a \to P \,\square\, b \to P)$

  Process P satisfies both $\mathcal{C}1_L$ and $\mathcal{C}2_L$. Only the initial state allows an x labelled transition, and the state after such a transition, when restricted to low-level events, is bisimilar to the initial state restricted to low-level events.

- $Q = x \to (a \to Q \,\square\, b \to Q)$

  Process Q fails to satisfy both conditions since $Q \setminus_H$ = STOP is not bisimilar to $a \to Q \setminus_H \,\square\, b \to Q \setminus_H$.

We can look at how the local non-interference properties deal with the nondeterministic examples given in section 5.

Example 6.4

- $P = \tau.(\tau.l_1.P + h_1.l_1.P) + \tau.l_2.P$

  P is a process we would view as secure; it satisfies both $\mathcal{C}1_L$ and $\mathcal{C}2_L$, as well as SBSNNI and BNDC. The only state we need to check is $P1$, since this is the only state with a transition labelled by an element of H. The conditions are satisfied since $\tau.l_1.P \setminus_H$ is bisimilar to $l_1.P \setminus_H$.

- $Q = \tau.(\tau.l_1.Q + h_1.l_2.Q) + \tau.l_2.Q$

  Q is an insecure process; it fails both $\mathcal{C}1_L$ and $\mathcal{C}2_L$, as well as SBSNNI, but it satisfies BNDC. Once again there is a single state we need to check, $Q1$, but this time the system after the h_1 transition, $l_2.Q \setminus_H$, is not bisimilar to the system without this event, $\tau.l_1.Q \setminus_H$.

- $Q' = \tau.l_1.Q' + \tau.l_2.Q' + h_1.l_1.Q'$

  Q' is an insecure process; it fails both $\mathcal{C}1_L$ and $\mathcal{C}2_L$, though SBSNNI and BNDC are both satisfied. The local non-interference conditions are not satisfied since $(Q'/h_1) \setminus_H = l_1.Q' \setminus_H$ is not bisimilar to $Q' \setminus_H = \tau.l_1.Q' \setminus_H + \tau.l_2.Q' \setminus_H$.

- $R = \tau.l_1.R + \tau.l_2.R + h_1.l_1.R + h_1.l_2.R$

  R is a secure process; it satisfies $\mathcal{C}2_L$ but fails $\mathcal{C}1_L$, as well as satisfying SBSNNI and BNDC. This process does not satisfy $\mathcal{C}1_L$ because the state $R1$ can only offer l_1 events, but both l_1 and l_2 events are available if no high-level action takes place. $\mathcal{C}1_L$ does not take into account the nondeterministic choice that is made as to which h_1 action occurs from state R. It was for precisely this reason that $\mathcal{C}2_L$ was introduced, and R satisfies it.

- $S = \tau.l_1.S + \tau.l_2.S + h_1.l_1.S + h_2.l_2.S$

  S is an insecure process; it fails both $\mathcal{C}1_L$ and $\mathcal{C}2_L$, though SBSNNI and BNDC are both satisfied. $\mathcal{C}1_L$ and $\mathcal{C}2_L$ treat S in the same way, since no state permits identical high-level actions. Process S fails both conditions for the same reason R failed to satisfy $\mathcal{C}1_L$.

-

$$
\begin{aligned}
T &= \tau.T_1 + h_1.T_2 \\
T_1 &= l_1 \to T_1 \sqcap \cdots \sqcap l_n \to T_1 \\
T_2 &= h_1.l_1.T_2 + \cdots + h_n.l_n.T_2
\end{aligned}
$$

  T is an insecure process; it fails both $\mathcal{C}1_L$ and $\mathcal{C}2_L$, as well as SBSNNI and BNDC, although BSNNI is satisfied. $\mathcal{C}1_L$ and $\mathcal{C}2_L$ are not satisfied since state $T2$ allows low-level actions after high-level actions, but permits no low-level actions without high-level ones.

  The example processes Q, Q', S and T fail both $\mathcal{C}1_L$ and $\mathcal{C}2_L$. These failures occur because whenever an H action takes place we arrive at a state that is distinguishable, by the low-level user, from at least one state that might have been reached had the action not occurred.

- A process P' constructed, using the method in example 5.7, from a P that fails to satisfy $\mathcal{C}1_L$ ($\mathcal{C}2_L$) will not satisfy $\mathcal{C}1_L$ ($\mathcal{C}2_L$). Failure to satisfy either condition implies there exists a state in process P that fails the relevant requirement. In the process P' we can still reach this state, and the same failure will occur.

There are a number of basic properties which the local non-interference properties satisfy. The following properties follow directly from the definitions 6.1 and 6.2.

Theorem 6.5

1. STOP satisfies $\mathcal{C}1_L$ and $\mathcal{C}2_L$.
2. If P does not communicate in H then $\mathcal{C}1_L(P)$ and $\mathcal{C}2_L(P)$ hold.
3. If P is a non-divergent process that does not communicate in L then $\mathcal{C}1_L(P)$ and $\mathcal{C}2_L(P)$ hold.
4. $\mathcal{C}1_L(P) \Longrightarrow \mathcal{C}2_L(P)$.

If a system can start behaving as an insecure system it would seem reasonable that the whole system be regarded as insecure. The definitions of the local non-interference conditions encourage this, since they have to be satisfied for each state where high-level transitions are possible.

Theorem 6.6

If $\mathcal{C}1_L(P)$ holds and S is a state of P, then $\mathcal{C}1_L(S)$ holds. Similarly for $\mathcal{C}2_L$.

Proof (for strong local non-interference)

Suppose there exists a state S such that $\mathcal{C}1_L(S)$ does not hold. By definition 6.1 there exists states $\sigma, \sigma' \in$ states(S) and a transition $h \in H$ such that $\sigma \xrightarrow{h} \sigma'$ but $\sigma \setminus_H \not\approx_B \sigma' \setminus_H$. Since $\sigma, \sigma' \in$ states(S) we have $\sigma, \sigma' \in$ states(P). Hence P does not satisfy $\mathcal{C}1_L(P)$, and we have a contradiction.

$\square$

The preceding theorem can be used to prove results about processes, such as the effect of an initial choice.

Theorem 6.7

$\mathcal{C}1_L(P)$ and $\mathcal{C}2_L(Q)$ hold if and only if $\mathcal{C}1_L(\tau.P + \tau.Q)$ holds. Similarly for $\mathcal{C}2_L(P)$.

Proof

($\Leftarrow$) Suppose $\mathcal{C}1_L(\tau.P + \tau.Q)$ holds. P and Q are states of $\tau.P + \tau.Q$, hence by theorem 6.5 $\mathcal{C}1_L(P)$ and $\mathcal{C}2_L(Q)$ are satisfied.

($\Rightarrow$) Suppose $\mathcal{C}1_L(P)$ and $\mathcal{C}2_L(Q)$ hold. We can create the state transition sytem for $\mathcal{C}1_L(\tau.P + \tau.Q)$ by taking the systems for P and Q, renaming (if necessary) states to avoid conflict, and adding a new initial state that has a τ labelled transition to each of the two original initial states. The transitions between states in P and Q remain unchanged. The only new state is the initial state, and since this only has τ labelled transitions it satisfies $\mathcal{C}1_L$. Hence $\mathcal{C}1_L(\tau.P + \tau.Q)$ holds as required.

$\square$

Theorem 6.8

Let a and b be low-level events, and x and y be high-level events.

- $\mathcal{C}1_L(P)$ and $\mathcal{C}1_L(Q)$ hold if and only if $\mathcal{C}1_L(a.P + b.Q), \mathcal{C}1_L(a.P + a.Q)$ and $\mathcal{C}1_L(a.P + \tau.Q)$ hold.
- $\mathcal{C}1_L(x.P + \tau.Q)$ holds if and only if $\mathcal{C}1_L(P)$ and $\mathcal{C}1_L(Q)$ hold, and $P \setminus_H \approx_B Q \setminus_H$.
- $\mathcal{C}1_L(x.P + a.Q)$ holds if and only if $\mathcal{C}1_L(P)$ and $\mathcal{C}1_L(Q)$ hold, and $P \setminus_H \approx_B a.Q \setminus_H$.

The equivalence used by the local non-interference conditions is weak bisimulation. Since $P \approx_B Q$ implies $P =_{FD} Q$, for non-divergent processes, this makes the conditions stricter than their failure-divergence counterparts. The stricter the equivalence the more processes that can be differentiated, so a user who observes the system recognising a strict equivalence can gain more information than one observing it with a weaker equivalence. We want a process to be secure against the strongest possible user, so the weak bisimulation equivalence is used.

We can define local non-interference properties which are identical to the previous conditions, but have the weak bisimulation equivalence replaced by the failures-divergences one. These conditions are useful when comparing the local non-interference properties to the lazy abstraction ones.

Definition 6.9

E satisfies strong local non-interference, written $\mathcal{C}1_L^{FD}(E)$, if $\sigma \xrightarrow{h} \sigma'$, where $h \in H$ and $\sigma, \sigma' \in \text{states}(E)$, implies $\sigma \setminus_H =_{FD} \sigma' \setminus_H$.

Definition 6.10

E satisfies local non-interference, written $\mathcal{C}2_L^{FD}(E)$, if $\sigma \xrightarrow{h} \sigma_1, \ldots, \sigma \xrightarrow{h} \sigma_n$, where $h \in H$ and $\sigma, \sigma_1, \ldots, \sigma_n \in \text{states}(E)$, implies $\sigma \setminus_H =_{FD} \tau.\sigma_1 \setminus_H + \ldots + \tau.\sigma_n \setminus_H$.

We can show corresponding theorems to 6.5–6.8 for these new conditions. The proofs for these theorems do not rely on the fact that we were using bisimulation for the local non-interference conditions. Where bisimulation was mentioned (in the proof of theorem 6.6) it can be simply changed to the failures-diverges equivalence.

In addition we can show that, for non-divergent, deterministic processes, the reverse implication of theorem 6.5.4 holds.

Theorem 6.11

If E is a non-divergent, deterministic process then $\mathcal{C}2_L^{FD} \Rightarrow \mathcal{C}1_L^{FD}$.

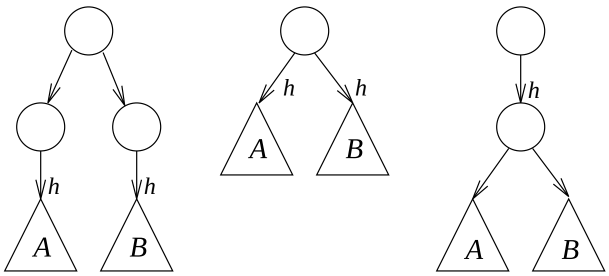
Proof

Let σ be a state of E, and t a transition where $\lambda(t) \in H$. Suppose $\sigma \xrightarrow{t} \sigma_1, \ldots, \sigma \xrightarrow{t} \sigma_n$. $\mathcal{C}2_L^{FD}$ gives us $\sigma \setminus_H =_{\text{FD}} \tau.\sigma_1 \setminus_H + \ldots + \tau.\sigma_n \setminus_H$. Since E is deterministic $\sigma_i \setminus_H =_{\text{FD}} \sigma_j \setminus_H \;\; \forall i, j$. Hence

$$\begin{aligned}
\sigma \setminus_H &=_{\text{FD}} \tau.\sigma_1 \setminus_H + \ldots + \tau.\sigma_n \setminus_H \\
&=_{\text{FD}} \tau.\sigma_i \setminus_H + \ldots + \tau.\sigma_i \setminus_H \\
&=_{\text{FD}} \tau.\sigma_i \setminus_H \\
&=_{\text{FD}} \sigma_i \setminus_H \qquad \forall i
\end{aligned}$$

□

If the occurrence of a high-level event is going to cause some nondeterministic choice, for example $h.A + h.B$, there are three ways we might look at it.

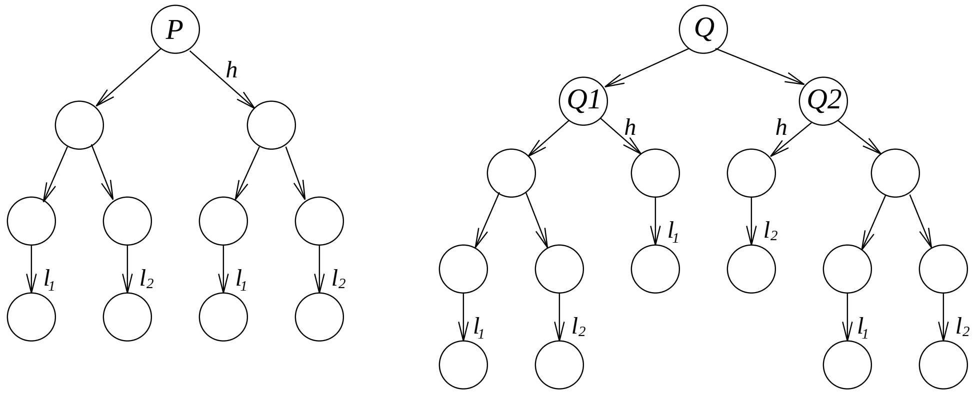


Firstly we can view the nondeterministic choice as occurring before the high-level event. Secondly we might view the choice as happening while the event is being communicated. Finally the nondeterministic choice can be considered as occurring after the event.

These three ways of viewing such a nondeterministic choice are not bisimilar, and are treated differently by the local non-interference conditions. Local non-interference, in contrast to the strong local non-interference condition, treats the 'after' and 'during' approaches as equivalent. The 'before' approach will frequently however be treated as insecure when the other two approaches might be considered secure.

Example 6.12

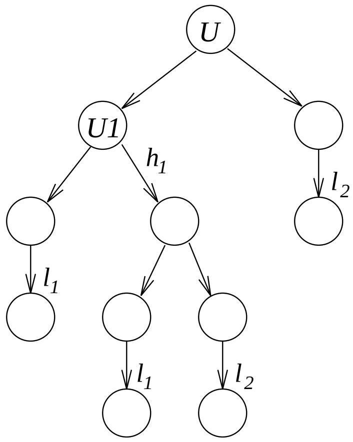
$$P = \tau.(\tau.l_1 + \tau.l_2) + h.(\tau.l_1 + \tau.l_2)$$
$$Q = \tau.(\tau.(\tau.l_1 + \tau.l_2) + h.l.1) + \tau.(h.l_2 + \tau.(\tau.l_1 + \tau.l_2))$$



Process P is considered secure by $\mathcal{C}1_L$ and $\mathcal{C}2_L$, whereas process Q is considered insecure by both properties. Whether we believe process Q is secure or not depends on what we believe the high and low-level user are capable of seeing/doing. If, for example, the high-level user is aware of whether the system is in state $Q1$ or $Q2$ it would be undesirable to call this system secure, since the high-level user is able to resolve an observable nondeterminism.

Example 6.13

Consider the process $U = \tau.(\tau.l_1 + h.(\tau.l_1 + \tau.l_2)) + \tau.l_2$



U is another process that we might like to think of as secure. If either an l_1 or l_2 event becomes available, the low-level user cannot tell whether an h_1 event has been communicated. This relies on the global behaviour of the system, since the process starting at $U1$ would be considered insecure.

These examples suggest that the local non-interference conditions may be too strict. Since the conditions only look forward, or 'down' the transition system, they reject process U precisely because $U1$ is insecure, and similarly reject process Q. Global conditions that consider the previous states of the transition system might solve these problems.

7. Comparisons

We can show that for non-divergent, deterministic processes the local non-interference properties, bisimulation non-deducibility conditions and the lazy security property are all equivalent. This will be shown initially for the failure-divergence version of these properties, and the result for the weak bisimulation versions follows.

Lemma 7.1

Suppose $\mathcal{C}1_L^{FD}$ holds for process E. If $(\delta, K) \in failures[\![E]\!]$ then $(\delta \upharpoonright L, K) \in failures[\![E \setminus_H]\!]$.

Proof

Let $\delta = \delta_{l_1} \frown \langle h_1 \rangle \frown \ldots \frown \langle h_n \rangle \frown \delta_{l_{n+1}}$, where $\forall\, i \bullet \delta_{l_i} \in traces[\![\mathrm{RUN}_L]\!]$, and hence may be empty. Since $\delta \in traces[\![E]\!]$ we can choose E_i so that

$$E \overset{\delta_{l_1}}{\Longrightarrow} E_1 \overset{h_1}{\longrightarrow} E_1' \ldots E_n \overset{h_n}{\longrightarrow} E_n' \overset{\delta_{l_{n+1}}}{\Longrightarrow} E'$$

and $(\langle\rangle, K) \in failures[\![E']\!]$. We have $\mathcal{C}1_L^{FD}$ so $E_n \setminus_H =_{\mathrm{FD}} E_n' \setminus_H$. Hence, since $E_n' \overset{\delta_{l_{n+1}}}{\Longrightarrow} E'$ implies $(\delta_{l_{n+1}}, K) \in failures[\![E_n' \setminus_H]\!]$, we have $(\delta_{l_{n+1}}, K) \in failures[\![E_n \setminus_H]\!]$.

Suppose $(\delta_{l_{i+1}} \frown \ldots \frown \delta_{l_{n+1}}, K) \in failures[\![E_i \setminus_H]\!]$. Since $E_{i-1}' \overset{\delta_{l_i}}{\Longrightarrow} E_i$, we have $(\delta_{l_i} \frown \ldots \frown \delta_{l_{n+1}}, K) \in failures[\![E_{i-1}' \setminus_H]\!]$. Since $\mathcal{C}1_L^{FD}$ holds we get $(\delta_{l_i} \frown \ldots \frown \delta_{l_{n+1}}, K) \in failures[\![E_{i-1} \setminus_H]\!]$.

Hence by induction $(\delta_{l_2} \frown \ldots \frown \delta_{l_{n+1}}, K) \in failures[\![E_1 \setminus_H]\!]$. Therefore, since $E \overset{\delta_{l_1}}{\Longrightarrow} E_1$ and $S \upharpoonright L = \delta_{l_1} \frown \ldots \frown \delta_{l_{n+1}}$ we have $(S \upharpoonright L, K) \in failures[\![E \setminus_H]\!]$.

□

Theorem 3.7 gives us that a process P is lazily secure if and only if the interleaving condition, $P \,|||\, \mathrm{RUN}_H$, is deterministic. The following results for lazily secure processes will be shown using this interleaving condition.

Theorem 7.2

If E is a non-divergent, deterministic process then $\mathcal{C}1_L^{FD}(E) \Rightarrow E$ is lazily secure.

Proof

We need to show that $E \,|||\, \mathrm{RUN}_H$ is deterministic. Suppose $\exists\, \delta, a$ such that $(\delta, a) \in failures[\![E \,|||\, \mathrm{RUN}_H]\!]$ and $\delta \frown \langle a \rangle \in traces[\![E \,|||\, \mathrm{RUN}_H]\!]$. [Trivially a cannot be in H since $E \,|||\, \mathrm{RUN}_H$ can always communicate events in H.]

Consider a transition system for $(E \,|||\, \mathrm{RUN}_H)$ generated by taking a transition system for E and for each state s adding a transition for each element $h \in H$ returning to s. $\exists\, E'$ such that $E \,|||\, \mathrm{RUN}_H \overset{\delta}{\Longrightarrow} E'$ and $E' \overset{a}{\not\longrightarrow}$. Let γ be a path from E to E' along the same transitions as δ but ignoring those which are labelled by an element of H and return to the state they start at. We have $E \stackrel{\gamma}{\Longrightarrow} E'$ where $E' \stackrel{a}{\not\longrightarrow}$, and $\delta \upharpoonright L = \gamma \upharpoonright L$. Similarly we can get $E \stackrel{\gamma'}{\Longrightarrow} E''$ where $E'' \stackrel{a}{\longrightarrow} E'''$ and $\delta \upharpoonright L = \gamma' \upharpoonright L$.

By lemma 7.1, since $(\gamma, \{a\}) \in failures[\![E']\!]$, we get $(\gamma \upharpoonright L, \{a\}) \in failures[\![E]\!]$. Similarly $\exists K \bullet (\gamma' \frown \langle a \rangle, K) \in failures[\![E''']\!]$ which implies $(\gamma' \upharpoonright L) \frown \langle a \rangle \in traces[\![E]\!]$. Since $\gamma \upharpoonright L = \delta \upharpoonright L = \gamma' \upharpoonright L$ we have $(\gamma \upharpoonright L) \frown \langle a \rangle \in traces[\![E]\!]$, but this contradicts our hypothesis that E is deterministic. Hence $(E \mid\mid\mid \mathrm{RUN}_H)$ is deterministic.

□

Lemma 7.3

If E is deterministic and $E \stackrel{\gamma}{\Longrightarrow} E'$, then E' is deterministic.

Proof

Suppose $(\gamma, \{a\}) \in failures[\![E']\!]$ and $\gamma \frown \langle a \rangle \in traces[\![E']\!]$. Hence $(\delta \frown \gamma, \{a\}) \in failures[\![E]\!]$ and $\delta \frown \gamma \frown \langle a \rangle \in traces[\![E]\!]$, but this contradicts our hypothesis that E is determinisitic.

□

Theorem 7.4

If E is a non-divergent, deterministic process then E is lazily secure $\Rightarrow \mathcal{C}1_L^{FD}(E)$.

Proof

For all states σ we need to show if $\sigma \stackrel{t}{\longrightarrow} \sigma', \lambda(t) \in H$ then $\sigma \setminus_H =_{\mathrm{FD}} \sigma' \setminus_H$.

For all $(\delta, K) \in failures[\![\sigma \setminus_H]\!]$ where $\delta = \langle \delta_1 \rangle \frown \ldots \frown \langle \delta_n \rangle$, suppose $\exists\, m < n$ such that $(\langle \delta_1 \rangle \frown \ldots \frown \langle \delta_m \rangle, \{\}) \in failures[\![\sigma' \setminus_H]\!]$. Therefore $(\langle t \rangle \frown \langle \delta_1 \rangle \frown \ldots \frown \langle \delta_m \rangle, \{\}) \in failures[\![\sigma]\!]$. Since δ is a trace of σ, $\langle t \rangle \frown \delta$ is a trace of $\sigma \mid\mid\mid \mathrm{RUN}_H$. Hence $\sigma \mid\mid\mid \mathrm{RUN}_H$ is not deterministic, but this implies (by lemma 7.3) neither is $E \mid\mid\mid \mathrm{RUN}_H$, which contradicts our hypothesis that E is lazily secure.

Suppose $\exists\, k \in K$ such that $\delta \frown \langle k \rangle \in traces[\![\sigma' \setminus_H]\!]$. Therefore $\langle t \rangle \frown \delta \frown \langle k \rangle \in traces[\![\sigma \mid\mid\mid \mathrm{RUN}_H]\!]$ and $(\langle t \rangle \frown \delta, k) \in failures[\![\sigma \mid\mid\mid \mathrm{RUN}_H]\!]$, which again contradicts our hypothesis. Hence $(\delta, K) \in failures[\![\sigma' \setminus_H]\!]$.

Similarly we can show that $(\delta, K) \in failures[\![\sigma' \setminus_H]\!]$ implies $(\delta, K) \in failures[\![\sigma \setminus_H]\!]$.

□

Theorem 7.5

If E is a non-divergent, deterministic process then E is lazily secure is equivalent to $\mathcal{C}1_L^{FD}(E)$ and $\mathcal{C}2_L^{FD}(E)$.

Proof

$\mathcal{C}1_L^{FD}$ is equivalent to E being lazily secure, follows immediately from theorems 7.2 and 7.4. $\mathcal{C}1_L^{FD} \Rightarrow \mathcal{C}2_L^{FD}$ follows directly from the definitions, and the reverse direction is proved in theorem 6.11.

□

Theorem 7.6

For non-divergent, deterministic processes, weak bisimulation ($\approx_B$) and failures-divergences equivalence ($=_{\mathrm{FD}}$) are interchangable.

Proof

$E \approx_{\mathrm{B}} F$ implies $E =_{\mathrm{FD}} F$. For non-divergent, deterministic processes the reverse is true – we can define a relation R where $(E', F') \in R$ if and only if $\exists\, \gamma \bullet E \stackrel{\gamma}{\Longrightarrow} E', F \stackrel{\gamma}{\Longrightarrow} F'$.

□

Corollary 7.7

Theorems 6.11, 7.2, 7.4 and 7.5, and lemmas 7.1 and 7.3 hold If we replace $\mathcal{C}1_L^{FD}$ and $\mathcal{C}2_L^{FD}$ with $\mathcal{C}1_L(E)$ and $\mathcal{C}2_L(E)$ respectively.

In [3] it shown that BNDC and SBSNNI are equivalent to the lazy security property for non-divergent deterministic processes. These results are to be expected since, as remarked in [9] — "... all the different [untimed] models of concurrency that treat nondeterminism ... collapse at the deterministic processes." A user who is able to observe the process under a different model to the one we are considering is unable to gain any additional information.

8. Refinement Paradox

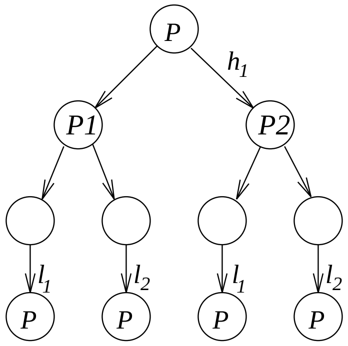
The security properties for nondeterministic systems suffer from the *refinement paradox*. There are some processes that are considered secure, which have refinements that are insecure. This seems undesirable, and counter-intuitive, since a refinement of a process can be thought of as one of the ways the process can behave. [It is perfectly reasonable to expect insecure processes to be refined by secure ones. The process $P \sqcap Q$, where P is a secure process and Q is an insecure one, can be refine by P.]

The determinism based conditions avoid the refinement paradox since they are closed under refinement. This follows since deterministic processes are maximal under refinement. For non-divergent, deterministic processes we have shown that the local non-interference and bisimulation non deducibility conditions agree with the lazily secure property, so we know the refinement paradox does not affect such processes.

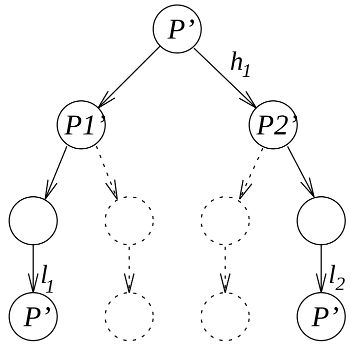
Example 8.1

We do not need a complicated process to exhibit the refinement paradox.

$$P = \tau.(\tau.l_1.P + \tau.l_2.P) + h_1.(\tau.l_1.P + \tau.l_2.P)$$



This is a process it may seem reasonable to class as secure. Whenever the high-level user communicates an h_1 event the system is taken to a state that is equivalent to the state that we would have reached had the event not been communicated. The occurrence of an l_1 leading from state $P1$ is indistinguishable to an l_1 leading from state $P2$, and similarly for l_2. This process satisfies both BNDC, $\mathcal{C}1_L$ and $\mathcal{C}2_L$.



The process can be refined by $P' = \tau.(\tau.l_1.P') + h_1.(\tau.l_2.P')$ by deciding how the internal choices at $P1$ and $P2$ are resolved. This process is clearly insecure. The low-level user knows an h_1 event has occurred if an l_2 event can be communicated, and similarly knows an h_1 event has not occurred if l_1 is available. This process satisfies neither BNDC, $\mathcal{C}1_L$ or $\mathcal{C}2_L$.

Once cause of information flow in nondeterministic processes is when the high-level user resolves some nondeterminism in the system. Some forms of refinement also resolve nondeterminism, and this can remove states equivalent to states that other events, possibly under the control of the high-level user, can still place the system in. This can cause some (potentially) secure processes to have refinements that are insecure. When we refined P to P' in example 8.1 two nondeterministic choices were resolved; the one at $P1$ and the one at $P2$. The security of P partly relies on the fact that $P1$ and $P2$ are equivalent, but this is no longer true of the relative states $P1'$ and $P2'$ of P'. This refinement problem does not arise in deterministic systems, since there are no 'observable' nondeterminisms to resolve.

It is difficult to see how to prevent this problem without seriously restricting the nondeterministic processes we count as secure. It would be necessary, for example, to have a condition that ensures, for all states S where a high-level event is available, all refinements of the low-level system view after the high-level event are equivalent to all refinements of the low-level system view without the event. This would limit any low-level nondeterminism to before the high-level user is able to act. This is because any nondeterminism once high-level actions can occur might be refined differently depending on whether or not these actions do occur.

A second form of refinement comes from rewriting a process by an equivalent one. Different process algebras, and models, make differing assumptions on how processes can be observed, and hence what it means for two processes to be equivalent. Allowing such refinement under some model assumes the users are only using the type of observations that the given model contains. If the low-level user can observe the system in a way that differentiates between processes we consider equivalent such refinements might allow the user to infer additional, undesirable information.

Example 8.2

We can define a process LEAK that echoes to the low-level user the events the high-level user has comminucated.

$$\text{LEAK} = \text{high?x} \rightarrow \text{low!x} \rightarrow \text{LEAK}$$

This is a process that is clearly insecure, and none of the conditions we have discussed would be satisfied. On the other hand it would seem reasonable that the process CHAOS was secure, since any observations by the low-level user may have happened with or without high-level communications.

Under the various denotational models of CSP the process LEAK $\sqcap$ CHAOS is equivalent to CHAOS, since CHAOS is the zero of the external choice operator. It is difficult to think of LEAK $\sqcap$ CHAOS as secure, since it can make an initial choice to behave as an insecure process. Since this equivalence holds however, the secure process CHAOS can be refined to the insecure process LEAK under such models.

This idea of a 'most nondeterministic non-divergent process' can be represented by infinitely many different transition systems. These transition systems are not all bisimilar, and indeed some will satisfy our conditions whilst others will not. CHAOS $\sqcap$ LEAK will never satisfy the local non-interference conditions because, as shown in theorem 6.7, LEAK is insecure. This is also true of the failures-divergences based local non-interference conditions, despite the equivalence to CHAOS, since the conditions consider the individual states of the transition system.

It is important to ask how we are modeling nondeterminism. *Probabilistic* nondeterminism arises when a process behaves according to the laws of probability. Under this nondeterminism model we have completely established distributions, that are maximal under the refinement order. Process P would not be secure if, for example, the distributions at $P1$ and $P2$ were different.

An alternative view of nondeterminism, and one more often considered, is *don't-care* or *under-specification* nondeterminism. Under this nondeterminism a process does not have to exhibit all the behaviours shown in its model, but every behaviour it does exhibit must be shown. The actual behaviour might be deterministic, probabilistically nondeterministic, resolved at an unmodelled level or by unmodelled agents.

The security properties considered for nondeterministic processes assume that there is no probabilistic information to be determined about either local or global behaviours. In addition they also assume that, as far as the users can determine, all of the nondeterministic behaviours that appear in the transition system can arise. If they did not the fact that a refinement P' of P existed that was insecure would mean that P was classed as insecure.

The treatment of nondeterminism for secure processes does not fit into either the 'probabilistic' or 'don't care' molds. All transitions in our nondeterministic systems are viewed as actual possibilities that may occur, and so it makes little sense to refine such a system and remove one or more of these possibilities.

Nondeterminism is viewed here as being *possibilistic*.

9. Conclusions

For deterministic processes the idea of security, or lack of information flow, seems to be well understood. There are various reasons for believing this. Firstly the conditions we have for different models are equivalent for deterministic processes, with the different untimed models of concurrency agreeing for such processes. Secondly the different security properties that we have for nondeterministic processes are equivalent for deterministic processes. Finally experience with the determinism conditions and a wide variety of processes has produced the expected results.

Nondeterministic processes appear to present more difficulties for specifying security properties. Part of this problem is the difficulty in modeling nondeterminism. There are several different ways of viewing nondeterminism, and the processes that we class as secure will change depending on which view we take. It is also much harder to encapsulate what we really want to mean by 'secure'. The idea of a high-level transition taking us to a state we may have reached without it, while still necessary, is not sufficient.

Studying processes as transition systems permits us to deal with the individual possibilities/transitions. Process algebras such as CSP can abstract too many of the details of nondeterminism. This makes it difficult, if not impossible, to directly determine facts that seem important in determining whether a process is secure; as in the CHAOS $\sqcap$ LEAK example. This suggests that the standard denotational models of CSP are not sufficient when looking at information flow in nondeterministic systems, and it is necessary to consider processes as transition systems.

Existing properties, such as BNDC and SBSNNI appear to classify some insecure processes as secure. We have introduced two new properties, local non-interference and strong local non-interference, and shown they correspond with our expectations for deterministic processes. For nondeterministic processes they appear to reject those that BNDC and SBSNNI incorrectly classify as secure, though there appear to be secure processes that are classified as insecure.

It seems more acceptable to be rejecting secure processes as insecure, than accepting insecure processes as secure. The first case makes the system designer more cautious, by potentially making it harder to design a secure process. The second case means we might fail in our aim of identifying undesirable information flow.

There are various areas into which this work can be continued :

1. Some elementary properties have already been shown for these new conditions. More properties have been discussed for the deterministic conditions [8, 12], as well as alternate nondeterministic conditions, and these properties can be considered for the new conditions.

2. The new conditions fail to identify some secure processes as secure. Further conditions can hopefully be developed that can identify more secure nondeterministic processes as secure, without incorrectly identifying insecure process as secure.

3. The complexity of the non-interference conditions can be considered.

4. For large processes it is advantageous, if not necessary, to be use automatic verification tools. Work needs to be done on seeing how (and if) the current conditions, and those that follow, can be automatically verified.

5. There are a variety of problems which are relevant when looking at nondeterministic systems. These include : how we model the system, what form of equivalence we use, how the users are able to observe and manipulate the system, how nondeterminism should be viewed, and what forms of refinement are valid.

Bibliography

[1] André Arnold. *"Finite Transition Systems"*. Prentice Hall. 1994.

[2] R. Focardi and R.Gorrieri. *"A Classification of Security Properties for Process Algebras"*. Journal of Computer Security, 3(1):5–33, 1994/1995.

[3] R. Focardi. *"Comparing Two Information Flow Security Properties"*.

[4] R.Focardi, R.Gorrieri and V. Panini. *"The Security Checker : a Semantics-based Tool for the Verification of Security Properties"*. Proceedings of the 8th IEEE Computer Security Foundation Workshop (CSFW'95), p60–69, June 1995.

[5] Formal Systems Europe. *"Failures-Divergences Refinement Manual"*. 1994.

[6] C.A.R. Hoare. *"Communicating Sequential Processes"*. Prentice Hall. 1985.

[7] Robin Milner. *"Communication and Concurrency"*. Prentice Hall. 1989.

[8] A.W. Roscoe, L. Wulf. *"Composing and Decomposing Systems under Security Properties"*. 1995.

[9] A.W. Roscoe. *"CSP and determinism in security modelling"*. Proceedings of 1995 IEEE Symposium on Security and Privacy.

[10] A.W. Roscoe, J.C.P Woodcock, L.Wulf. *"Non-interference through Determinism"*. Proceedings of ESORICS 94, LNCS #875.

[11] A.W. Roscoe. *"The Theory and Practice of Concurrency"*.

[12] Lars Wulf. *"Interaction and Security in Distributed Computing"*. Submitted DPhil thesis, University of Oxford.

Contents