What Semantic Equivalences Are Suitable for Non-interference Properties in Computer Security*

Xiaowei Huang[1], Li Jiao[2], and Weiming Lu[1]

[1] Academy of Mathematics and System Science,
Chinese Academy of Sciences, P.R. China
xwhuang@amss.ac.cn
[2] State Key Laboratory of Computer Science,
Institute of Software, Chinese Academy of Sciences, P.R. China

Abstract. Non-interference properties are an important class of security properties. Many different non-interference properties have been presented based on different underlying models including the process algebraic languages. Usually, in specifying the non-interference properties using process algebraic languages, a specific semantic equivalence is introduced. Though weak bisimulation based non-interference properties have been studied extensively, it is not always satisfactory. This paper considers the topic on pursuing a probably more suitable semantic equivalence for specifying the non-interference properties. We find several alternatives, e.g., should testing equivalence, impossible future equivalence and possible future equivalence, etc. As another topic in the paper, based on the structural operational semantics, we suggest a compositional rule format, the SISNNI format, for an impossible future equivalence based non-interference property, i.e., the SISNNI property. We show that the SISNNI property is compositional in any SISNNI languages, i.e., languages in the SISNNI format.

Keywords: semantic equivalences, rule format, non-interference, computer security.

1 Introduction

To verify whether a system satisfies certain information flow property [1], a formal description of this property is necessary. Non-interference is proposed as a formal description of the information flow property. Though practical systems are usually designed to be with multi-level security [9], they may be simply represented as two-level systems [1,11]. Based on two level systems, the non-interference properties guarantee that a high level user should not interfere with the low level user.

Practical systems are modeled with various mathematical models, and thus non-interference properties should have totally different representations [8]. Besides, even in a given mathematical model, several information flow properties are always available. In this paper, we will focus on those properties based on process algebraic languages. We suppose three specific operators, i.e., the hiding operator of CSP and the parallel

* This research was financially supported by the National Natural Science Foundation of China (No. 60421001).

S. Qing, H. Imai, and G. Wang (Eds.): ICICS 2007, LNCS 4861, pp. 334–349, 2007.

composition operator and restriction operator of CCS, since they have been used in the formal characterizations of some information flow properties.

In [1] and [7], many information flow properties have been proposed based on the variants of CCS and CSP, respectively. We only concern those properties in [1]. On one hand, NNI (Non-determinism Non Interference) is proposed as a direct translation of the Non Interference, which requires that for each trace σ of the given system p, there exists another trace δ with the same subsequence of low level actions and without high inputs [1]. However, it is not satisfactory when synchronous communications are considered, and therefore lead to another property, named SNNI (Strong NNI). On the other hand, NDC (Non-Deducibility on Compositions) says that a low level user sees of the system is not modified by composing any high level process to the system.

As an extension to these properties, two strengthened properties are proposed to make the above properties suitable for the dynamic context. P-NDC (Persistent NDC) requires that each reachable state should satisfy the NDC property, and likewise, SSNNI (Strong SNNI) requires that each reachable state should satisfy the SNNI property.

Furthermore, observing that the above properties are all based on weak trace equivalence, they may be further promoted by other finer weak equivalences, such as the testing equivalence and the weak bisimulation. Therefore, several weak bisimulation based properties, e.g., BNNI (Bisimulation NNI), BSNNI (Bisimulation SNNI), SBSNNI (Strong BSNNI), BNDC (Bisimulation NDC) and P-BNDC (Persistent BNDC), have been proposed in [1].

We will show in the paper that, weak bisimulation is not always satisfactory, because it requires that two related processes have the same branching structures, which is not needed for the non-interference. In fact, a counterexample will be put forward to illustrate that a trivially secure process will be deemed to be insecure if weak bisimulation based non-interference properties are applied.

Based on this observation, we suggest a criterion to make clear what semantic equivalences are suitable for the non-interference properties. Then, several semantic equivalences are suggested to be more suitable than weak bisimulation, e.g., should testing equivalence [2], impossible future equivalence [26,25] and possible future equivalence [3].

As an example, we will take impossible future equivalence into consideration, and propose the non-interference properties based on it. These properties include INNI (Impossible-future NNI), ISNNI (Impossible-future SNNI), SISNNI (Strong ISNNI), INDC (Impossible-future NDC) and P-INDC (Persistent INDC).

After that, we will show the compositional results of the SISNNI property by proposing a rule format for it. Any languages observing the rule format will be compositional for the SISNNI property, i.e., if all subprocesses hold the SISNNI property, then the composite process will also hold the SISNNI property.

Rule format is a concept coming from the structural operational semantics (SOS). SOS [12] have been widely used in defining the meanings of the operators in various process algebraic language, such as CCS [13] and ACP [19]. Transition System Specifications (TSSs) [16], which borrowed from logic programming, form a theoretical basis for SOS. By imposing some syntactic restrictions on TSS, one can retrieve so-called

rule formats. From a specified rule format, one may deduce some interesting properties, within which two properties are concerned in the paper.

On one hand, rule format guarantees the congruence of some equivalence. As stated in [20], equivalence relation $\sim$ is congruent on some TSS, if $\sim$ satisfies the compatibility property, which states that, for any n-ary function symbol f in the TSS and processes p_i, q_i, if $p_i \sim q_i$ for $1 \leq i \leq n$ then $f(p_1, ..., p_n) \sim f(q_1, ..., q_n)$. On the other hand, rule format guarantees the compositionality of some property. A property φ is compositional on some TSS if $f(p_1, ..., p_n)$ satisfies property φ when f is any n-ary function symbol in the TSS and processes p_i satisfies property φ for $1 \leq i \leq n$. We will use language as an alias of the TSS.

Up to now, some rule formats have been presented to be congruence format for semantic equivalences, for examples, GSOS format [21] and ntyft/nxyft format [17] have been proved to be congruent on the strong bisimulation, de Simone [18] format was proved to be congruent on the failure equivalence, and so on. On the other hand, Tini [11] have proposed rule formats, e.g., rooted SBSNNI format and P-BNDC format, to be compositional formats for the SBSNNI property and P-BNDC property, respectively. The readers are referred to Mousavi, Reniers and Groote [14] for a latest review on the rule formats.

The authors have proposed a rule format, i.e., the weak ω-failure format, to be a congruence format for the impossible future equivalence [6]. In this paper, we will prove that, if we make a clear cut on the observable actions into high level actions and low level actions, and restrict that the high level actions (resp. the low level actions) cannot interplay with the low level actions (resp. the high level actions) in any transition rule, then the weak ω-failure format is also a compositional format for the SISNNI property.

The structure of this paper is as follows. The next section will provide some preliminaries on the process algebraic languages and the SOS, and make a division on the observable actions into high level actions and low level actions. Section 3 will make a review on several non-interference properties. In Section 4, we will make a discussion on what make an weak equivalence suitable for the non-interference properties and then present several alternatives for the weak bisimulation. Section 5 will discuss along the canonical way the properties based on impossible future equivalence, which is proved to be one of the suitable alternative weak equivalences. Then, in Section 6, we will propose the SISNNI format and prove that it is a compositional format for the SISNNI property. Finally, we will conclude the paper in Section 7.

2 Preliminaries on Process Algebraic Languages

Let Act denote a set of names which will be used to label on events and Act^* be the set of all action sequences. We usually use $a, b, ...$ to range over the actions in Act, and use $A, B, ...$ to range over the sets of actions in Act. τ is generally used to denote the internal actions which can not be observed by the outer world, and we use $\alpha, \beta, ...$ to range over the actions in $Act \cup \{\tau\}$. $\delta, \mu, \sigma, ...$ is to range over the sequences of actions. Φ is to range over the sets of sequences. $p, q, ...$ will be used to represent processes. Besides, $\varepsilon = \tau^*$. $\mathbb{N}$ is the set of natural numbers and ω is the cardinality of $\mathbb{N}$.

Basically, presenting a set of syntactic constructions is the first step to define a process algebraic languages, e.g., CCS, CSP and ACP.

Definition 2.1 [15]. Let $V = \{x_1, x_2, ...\}$ be a set of variables. A signature Σ is a collection of function symbols $f \notin V$ equipped with a function $ar : \Sigma \to N$. The set $\mathbb{T}(\Sigma)$ of terms over a signature Σ is defined recursively by: 1) $V \subseteq \mathbb{T}(\Sigma)$; 2) if $f \in \Sigma$ and $t_1, ..., t_{ar(f)} \in \mathbb{T}(\Sigma)$, then $f(t_1, ..., t_{ar(f)}) \in \mathbb{T}(\Sigma)$.

A term $c()$ is abbreviated as c. For $t \in \mathbb{T}(\Sigma)$, $var(t)$ denotes the set of variables that occur in t. $\mathrm{T}(\Sigma)$ is the set of closed terms over Σ, i.e., the terms $p \in \mathbb{T}(\Sigma)$ with $var(p) = \emptyset$. A term is an open term if it is not a closed term. A Σ substitution ζ is a mapping from V to $\mathbb{T}(\Sigma)$.

In the paper, we will use $p, q, ...$ to range over the closed terms, and call them processes.

As stated in the introduction, we require that three previously-defined operators, i.e., the hiding operator of CSP, the restriction operators and parallel composition operator of CCS, are in the language, because they have been used in [1] to characterize some non-interference properties which will be discussed in the paper. Therefore, the syntax of a language may be:

$p ::= p\backslash A \mid p/A \mid p|p \mid ...$, where $A \subseteq Act$ is a set of observable actions.

SOS has been widely accepted as a tool to define operational semantics of processes. TSSs are a formalization of SOS [12]. The readers are referred to Aceto, Fokkink and Verhoef [15] for a comprehensive review on SOS.

Definition 2.2. A positive Σ-literal is an expression $t \xrightarrow{\alpha} t'$ and a negative Σ-literal is an expression $t \xrightarrow{\alpha} \!\!\!\!\!\!/\;\;$ with $t, t' \in \mathbb{T}(\Sigma)$ and $\alpha \in Act \cup \{\tau\}$. A transition rule over Σ is an expression of the form $\frac{H}{C}$ with H a set of Σ literals (the premises of the rule) and C a positive Σ-literal (the conclusion). The left- and right-hand side of C are called the source and the target of the rule, respectively. Moreover, if $r = \frac{H}{C}$ then let $ante(r) = H$ and $cons(r) = C$.

A TSS, written as (Σ, Ψ), consists of a signature Σ and a set Ψ of transition rules over Σ. A TSS is positive if the premises of its rules are positive. In the paper, we often use language as an alias of the TSS.

Here, as an example, the rules for the restriction operator, the hiding operator and the parallel composition operator are as follows.

$$p/A: \frac{p \xrightarrow{\alpha} p'}{p/A \xrightarrow{\alpha} p'/A}\ \alpha \notin A \qquad \frac{p \xrightarrow{a} p'}{p/A \xrightarrow{\tau} p'/A}\ a \in A$$

$$p\backslash A: \frac{p \xrightarrow{\alpha} p'}{p\backslash A \xrightarrow{\alpha} p'\backslash A}\ \alpha \notin A$$

$$p|q: \frac{p \xrightarrow{\alpha} p'}{p|q \xrightarrow{\alpha} p'|q} \qquad \frac{q \xrightarrow{\alpha} q'}{p|q \xrightarrow{\alpha} p|q'} \qquad \frac{p \xrightarrow{a} p', q \xrightarrow{b} q'}{p|q \xrightarrow{\tau} p'|q'}\ (a, b) \in f$$

where $(a, b) \in f$ means that actions a and b may communicate synchronously.

Following it, the labeled transition systems (LTSs) are to be defined. LTSs are standard semantic models for the various process algebraic languages, and in fact, each process has an equivalent LTS by the help of the transition rules.

Definition 2.3. Let Σ be a signature. A transition relation over Σ is a relation $\mathrm{Tr} \subseteq \mathrm{T}(\Sigma) \times Act \cup \{\tau\} \times \mathrm{T}(\Sigma)$. Element (p, α, p') of a transition relation is written as $p \xrightarrow{\alpha} p'$.

Thus a transition relation over Σ can be regarded as a set of closed positive Σ-literals (transitions).

Definition 2.4. A labeled transition system (LTS) is a triple (T,Tr, $Act \cup \{\tau\}$), where T is the set of processes, i.e., the set of closed terms, and Tr is the transition relation defined as above.

After assigning the processes their corresponding LTSs by SOS, a natural topic is to decide whether two processes with different syntactic expressions are equivalent. Certainly, two processes with isomorphic LTSs should be deemed to be equivalent, which forms the so-called tree equivalence. However, tree equivalence may be too strong for practical uses. Therefore, various semantic equivalences weaker than tree equivalence are presented for different aims. The readers are referred to Glabbeek [3,4] for comprehensive reviews on semantic equivalences.

A common characterization of the equivalences is that, any semantics of some process p can be characterized denotationally by a function $O(p)$, which constitute the observable behaviors of p [3]. Therefore, the equivalence $\sim_O$ can be defined by $p \sim_O q \Longleftrightarrow O(p) = O(q)$.

Here, as an example, we present the definition of weak trace equivalence. Other equivalences used in the paper will be introduced whenever they are mentioned.

For an action sequence $\delta = \alpha_1 ... \alpha_n$, if there exists $p_1, ..., p_n \in \mathrm{T}(\Sigma)$ such that $p \xrightarrow{\alpha_1} p_1 \xrightarrow{\alpha_2} ... \xrightarrow{\alpha_n} p_n$, then we call δ a trace of p, denoted as $p \xrightarrow{\delta}$ or $p \xrightarrow{\alpha_1} ... \xrightarrow{\alpha_n}$.

In weak semantics, the weak transition relations and the weak traces also need to be defined. Let p be a process, we write $p \stackrel{a}{\Longrightarrow}$ iff $p \xrightarrow{\tau^*} \xrightarrow{a} \xrightarrow{\tau^*}$, where τ^* denotes any number of internal transitions. Hence, for an observable action sequence $\delta = a_1 ... a_n$, $p \stackrel{\delta}{\Longrightarrow}$ iff $p \stackrel{a_1}{\Longrightarrow} ... \stackrel{a_n}{\Longrightarrow}$.

Definition 2.5. Let p, q be two processes and set $\mathcal{T}(p) = \{\delta \in Act_H^* | p \stackrel{\delta}{\Longrightarrow}\}$ be the set of all trace of p. Then, p and q are weak trace equivalent, denoted as $p \sim_t q$, iff $\mathcal{T}(p) = \mathcal{T}(q)$.

For a given equivalence, one of the most frequently-asked problems is whether or not it can be preserved under some frequently-used operators, such as prefixing, choice, parallel composition, etc., in classical process algebraic languages like CCS [13], CSP [27] and ACP [19]. Generally, there exist two ways to deal with this problem: the first one is to prove the congruence properties of these operators one by one. It is a straightforward and intuitive way, but may be somewhat clumsy. The second one is to pursue a rule format for this specified equivalence. And the given equivalence can be preserved under any operator in this format.

Definition 2.6. Let Σ be a signature. A context $C(x_1, ..., x_n)$ of n holes over Σ is simply a term in $\mathbb{T}(\Sigma)$ in which n variables occur, each variable only once. If $t_1, ..., t_n$ are terms over Σ, then $C(t_1, ..., t_n)$ denotes the term obtained by substituting t_1 for the first variable occurring in C, t_2 for the second variable occurring, etc. If $x_1, ..., x_n$ are all different variables, then $C(x_1, ..., x_n)$ denotes a context of n holes in which x_i is the ith occurring variable.

In the following, we give the definition on the congruence of an equivalence in a language.

Definition 2.7. Let $\mathcal{L} = (\Sigma, \Psi)$ be a language. An equivalence relation $\sim$ is congruent in language $\mathcal{L}$ iff $\forall i \in \{1, ..., n\} : p_i \sim q_i \Longrightarrow C(p_1, ..., p_n) \sim C(q_1, ..., q_n)$ for any context $C(x_1, ..., x_n)$ of n holes in language $\mathcal{L}$, where p_i and q_i are closed terms, i.e., processes, over Σ.

In fact, a congruence format is to make restrictions on the syntax and rules to guarantee that, in a given language satisfying this format, the given equivalence will be congruent.

Likewise, we can define the compositionality of some property in a language.

Definition 2.8. Let $\mathcal{L} = (\Sigma, \Psi)$ be a language. A property φ is compositional in language $\mathcal{L}$ iff $\forall i \in \{1, ..., n\} : p_i \in \varphi \Longrightarrow C(p_1, ..., p_n) \in \varphi$, for any context $C(x_1, ..., x_n)$ of n holes in language $\mathcal{L}$, where $p_i \in \varphi$ means that p_i satisfies the property φ.

A compositional format is to make restrictions on the syntax and rules to guarantee that, in a given language satisfying this format, the given property will be compositional.

For the application of the above process algebraic theory in specifying the non-interference properties, two kinds of partitions on the observable actions are necessary:

1) Set Act is to be separated into two disjoint sets Act_H and Act_L in order to divide clearly the observable actions into two levels. And, we have $Act = Act_H \cup Act_L$. This division comes from an idea that the users will be classified into two classes, High and Low.

2) Set Act is to be separated into two disjoint sets O and I in order to divide clearly the functions of each action into a output action or an input action. Similarly, we need $Act = O \cup I$.

Finally, function $purge_L(\delta)$ takes a trace δ and returns an action sequence with all Low actions removed.

3 Several Related Non-interference Properties

In this section, we will simply review several non-interference properties presented in the seminal paper of Focardi and Gorrieri [1].

Definition 3.1 [1]. Let p be a process.

1) $p \in \text{NNI} \Longleftrightarrow (p\backslash_I Act_H)/Act_H \sim_t p/Act_H$, where $p\backslash_I Act_H$ is defined as $p\backslash(Act_H \cap I)$.
2) $p \in \text{SNNI}$ (Strong NNI) $\Longleftrightarrow p/Act_H \sim_t p\backslash Act_H$.
3) $p \in \text{NDC} \Longleftrightarrow \forall q \in \text{T}_H : p/Act_H \sim_t (p|q)\backslash Act_H$.

Proposition 3.2 [1]. SNNI $\subset$ NNI, NDC = SNNI.

The above three properties are based on the weak trace equivalence. And if substituting the weak trace equivalence with other weak equivalences, we may have three corresponding non-interference properties. But, the relations between them maybe will not be akin to Proposition 3.2 yet. For example, the properties based on the weak bisimulation BNNI, BSNNI and BNDC will be in the following relation.

Proposition 3.3 [1]. BNDC $\subset$ BNNI, BNDC $\subset$ BSNNI, BNNI $\not\subseteq$ BSNNI and BSNNI $\not\subseteq$ BNNI.

However, these properties may not be used in the dynamic contexts [10]. Therefore, one more promotion can be made on them.

Definition 3.4 [10,28]. Let p be a process.

1) $p \in$ SBSNNI iff, for all $\sigma, p' : p \stackrel{\sigma}{\Longrightarrow} p'$, we have $p' \in$ BSNNI.
2) $p \in$ P-BNDC iff, for all $\sigma, p' : p \stackrel{\sigma}{\Longrightarrow} p'$, we have $p' \in$ BNDC.

4 What Semantic Equivalences Are Suitable

In this section, we will make a discussion on one of the main topics in the paper, i.e., what semantic equivalences are suitable for the non-interference properties. The first two subsections come from the works of Focardi and Gorrieri [1] which interpret the reason why weak bisimulation is chosen as the specific weak equivalence for non-interference properties in their works. Then, in the third subsection, we will give a counterexample to show that the weak bisimulation is not always satisfactory. This situation makes us consider the actual requirements for a given weak equivalence to be suitable and therefore propose a criterion in the fourth subsection. Then, in the following several subsections, we will seek for several suitable weak equivalences.

4.1 Weak Trace Equivalence vs. Testing Equivalence

Focardi and Gorrieri [1] have discussed the different performances of these two equivalences when defining the non-interference properties, and claimed that, compared with the weak trace equivalence, testing equivalence is more suitable for non-interference due to its ability in detecting the high level deadlocks. They presented an example to show that, if there exist high level deadlocks, low level user may deduce information from the high level user and make the information flow insecure.

Weak trace equivalence has been defined in Section 2 and, to ease the following discussion, we will also give the definition of the testing equivalence.

Testing equivalence is a combination of the may testing equivalence and the must testing equivalence, which are two important testing-theoretical equivalences in the testing theory firstly proposed by Nicola and Hennessy [5]. Tests are processes with a special action $\surd$ to denote the successful termination. The way testing a process p is to synchronize p with a test t on all observable actions except for $\surd$, denoted as $p|t$ in the CCS terms or $p||_{Act}t$ in the CSP terms. Then, various testing theoretical equivalences are defined based on the necessity of the presence of the successful termination in $p|t$.

In defining the must testing equivalence, a notion named maximal run needs to be introduced beforehand. A maximal run of a transition system is a sequence of states $(p_i|t_i)_{0\le i\le n}$ for $n \in \mathbb{N} \cup \{\omega\}$ such that $p_i|t_i \xrightarrow{\alpha_{i+1}} p_{i+1}|t_{i+1}$ for all $0 \le i \le n-1$ and if $n \neq \omega$ then $\nexists \alpha \in Act \cup \{\tau\} : (p_n|t_n) \xrightarrow{\alpha}$.

Definition 4.1.1 [2]. Let p, q be two processes and t be a test.

1) p may t iff $\exists \delta \in Act^* : p|t \overset{\delta\surd}{\Longrightarrow}$.

2) p must t iff, for all maximal runs $(p_i|t_i)_{i<n} : (p|t = p_0|t_0)$ implies $\exists i < n : p_i|t_i \xrightarrow{\surd}$.

Moreover, p and q are may (resp. must) testing equivalence, denoted as $p \sim_{may} q$ (resp. $p \sim_{must} q$) , iff, for all tests t, p may (resp. must) t implies q may (resp. must) t, and vice versa. And p and q are testing equivalence, denoted as $p \sim_{test} q$, iff they are both may testing equivalence and must testing equivalence, i.e., $p \sim_{test} q \equiv p \sim_{may} q \wedge p \sim_{must} q$.

4.2 Testing Equivalence vs. Weak Bisimulation

The defect of testing equivalence has also been pointed out that it may falsely deem all systems with high level loops insecure. This defect forms one of the main attacks made by Focardi and Gorrieri, and then they turn to the weak bisimulation. The other reason is that the weak bisimulation may be checked in polynomial time with respect to the number of the states in the LTSs, but for the testing equivalence, it may be PSPACE complete.

Definition 4.2.1. A relation $R \subseteq \mathrm{T}\times\mathrm{T}$ is a weak bisimulation if $(p, q) \in R$ implies, for all $\alpha \in Act \cup \{\tau\}$,

1) if $p \xrightarrow{\alpha} p'$, then there exists q' such that $q \overset{\hat{\alpha}}{\Longrightarrow} q'$ and $(p', q') \in R$, and

2) if $q \xrightarrow{\alpha} q'$, then there exists p' such that $p \overset{\hat{\alpha}}{\Longrightarrow} p'$ and $(p', q') \in R$,

where $q \overset{\hat{\alpha}}{\Longrightarrow} q'$ stands for $q \overset{\alpha}{\Longrightarrow} q'$ if $\alpha \in Act$ and $q \xrightarrow{\tau^*} q'$ if $\alpha = \tau$.

4.3 Weak Bisimulation Is Not Always Satisfactory

However, though the weak-bisimulation-based non-interference properties, such as BNNI, BSNNI and BNDC, etc., have better performance than those based on trace equivalence and testing equivalence, they are not always satisfactory. The reason is that if two processes are weak bisimulated then their corresponding LTSs have the same branching structures. But on the other hand, non-interference cares little on the branching structures of processes, because a low level user cannot deduce anything from the branching structure of a give LTS. Therefore, the requirement on the branching structures may make the weak bisimulation too strong for specifying the non-interference properties.

In fact, we may construct a system to be secure by intuitiveness but may be falsely deemed to be insecure by the weak simulation based non-interference properties. It can

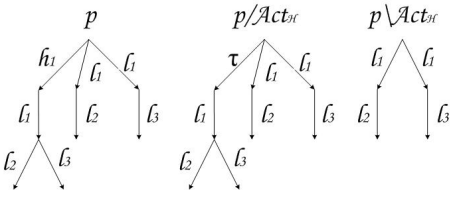

Fig. 1. Weak bisimulation may be too strong in characterizing the non-interference properties

be verified that the process p in Figure 1 does not satisfy the BSNNI, BNDC, SBSNNI properties, because $p/Act_H \approx_{ws} p\backslash Act_H$. However, it is secure if the only high level action h_1 is just a high level output action.

On the other hand, p will satisfy the corresponding non-interference properties based on impossible future equivalence, should testing equivalence or possible future equivalence, which care less branching structures than the weak bisimulation. We will introduce them in the following subsections. In fact, in Figure 1, $\forall p', \delta : p \stackrel{\delta}{\Longrightarrow} p'$, we have $p'/Act_H \sim_{shd} p'\backslash Act_H$, $p'/Act_H \sim_{if} p'\backslash Act_H$ and $p'/Act_H \sim_{pf} p'\backslash Act_H$.

4.4 Criterions

Based on the above observations, we may have a general criterion in seeking a suitable semantic equivalence as the underlying equivalence for characterizing non-interference properties. In fact, this equivalence is expected to

1) deal with high level deadlocks as the testing equivalence does,
2) deal with high level loops in a fair way as the weak bisimulation,
3) be decidable in polynomial time in case that the LTSs have only finite states and labels, and
4) be compositional on the hiding operator, the restriction operator and the parallel composition operator.

It is trivial that the weak bisimulation meets these criterions, but as we have argued in the preceding subsection, it may be too strong in some cases. Therefore, we want to find several alternatives.

4.5 From Testing Equivalence to Acceptance Testing

Leveling with the above criterions, we may find that the testing equivalence can only satisfy the first clause. The fact that it does not satisfy the second clause has been stated in Section 4.2, the fact that it does not satisfy the third clause has been pointed out in [1], and the fact that it does not satisfy the fourth clause has been shown by several papers including the authors [2,24,6].

In fact, the reason that the testing equivalence does not satisfy the second and the third clause exists in the maximal run of must testing equivalence. It is the maximal run that makes the internal loops catastrophic [2].

Definition 4.5.1. Let p, q be two processes and t be a test. p acc t iff, for all $\sigma \in Act^*$ and for all $p'|t'$ with $p|t \stackrel{\sigma}{\Longrightarrow} p'|t'$, there must exist $a \in Act \cup \{\surd\}$ such that $p'|t' \stackrel{a}{\Longrightarrow}$.

Moreover, p and q are acceptance testing equivalence, denoted as $p \sim_{acc} q$, iff, for all tests t, p acc t implies q acc t, and vice versa.

In fact, from the above definition, we find that, after excluding the concept of maximal runs from its definition, acceptance testing equivalence can deal with the internal loops in a fair way that the internal transitions may be executed an arbitrary but only finite number of times.

4.6 From Acceptance Testing to Should Testing

We have observed that the acceptance testing equivalence, or failure equivalence, may not be invariant under the hiding operator of CSP, which makes them not be the equivalence what we are seeking for. In [2], Rensink and Vogler have proposed the should testing equivalence to amend the defect of the acceptance testing equivalence.

Definition 4.6.1. Let p, q be two processes and t be a test. p shd t iff, for all $\sigma \in Act^*$ and for all $p'|t'$ with $p|t \stackrel{\sigma}{\Longrightarrow} p'|t'$, there must exist $\delta \in Act^*$ such that $p'|t' \stackrel{\delta\surd}{\Longrightarrow}$.

Moreover, p and q are should testing equivalent, denoted as $p \sim_{shd} q$, iff, for all tests t, p shd t implies q shd t, and vice versa.

4.7 Several Other Alternatives

Two other possible alternatives will be presented in this subsection. We will show their relationships with the should testing equivalence and the weak bisimulation.

Definition 4.7.1. $(\sigma, \Phi) \in Act^* \times \mathcal{P}(Act^+)$ is a weak impossible future pair of process p iff there exists some p' such that $p \stackrel{\sigma}{\Longrightarrow} p'$ and $\Phi \cap \mathcal{T}(p') = \emptyset$. The set of all weak impossible future pairs of process p is called the weak impossible future of p, denoted by $\mathcal{IF}(p)$.

For any two processes p and q, they are weak impossible future equivalent, denoted as $p \sim_{if} q$, iff $\mathcal{IF}(p) = \mathcal{IF}(q)$.

Definition 4.7.2 $(\sigma, \Phi) \in Act^* \times \mathcal{P}(Act^+)$ is a weak possible future pair of process p iff there exists some p' such that $p \stackrel{\sigma}{\Longrightarrow} p' \wedge \Phi = \mathcal{T}(p')$. The set of all weak possible future pairs of process p is called the weak possible future of p, denoted as $\mathcal{PF}(p)$.

Moreover, for any two processes p and q, they are weak possible future equivalent, denoted as $p \sim_{pf} q$, iff $\mathcal{PF}(p) = \mathcal{PF}(q)$.

Proposition 4.7.3. Let p and q be two processes. $p \sim_{wb} q \Longrightarrow p \sim_{pf} q \Longrightarrow p \sim_{if} q \Longrightarrow p \sim_{shd} q$.

4.8 Semantic Lattice

The left graph in Figure 2 shows a semantic lattice of the equivalences mentioned above. The arrows denote the 'coarser than' relations between two related semantic equivalences.

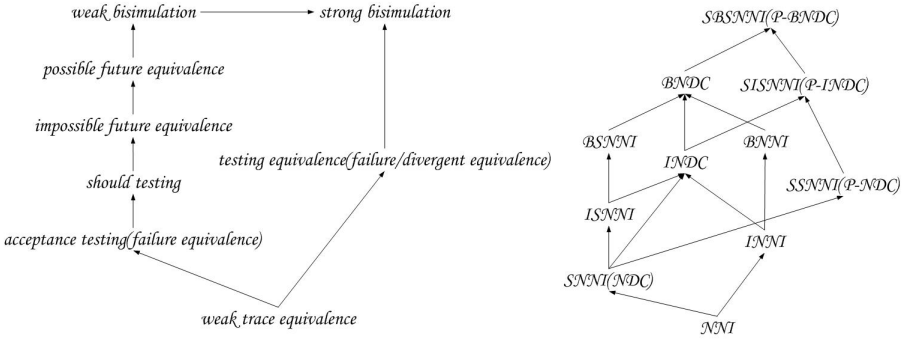

Fig. 2. Lattices for semantic equivalences and non-interference properties

5 Properties Based on Impossible Future Equivalence

In this section, we will present the non-interference properties based on impossible future equivalence. The reason why we choose impossible future equivalence is that it has a simpler denotational characterization than should testing equivalence, and it is coarser than possible future equivalence. The properties based on other equivalences may be defined and analyzed similarly.

Definition 5.1 (Impossible Future NNI, SNNI, NDC)

(1) $p \in \text{INNI} \Longleftrightarrow p/Act_H \sim_{if} (p\backslash_I Act_H)/Act_H$;
(2) $E \in \text{ISNNI} \Longleftrightarrow E/Act_H \sim_{if} E\backslash Act_H$;
(3) $E \in \text{INDC} \Longleftrightarrow \forall F \in \mathcal{P}_H : E/Act_H \sim_{if} (E|F)\backslash Act_H$.

Proposition 5.2. (1) BNNI $\subset$ INNI $\subset$ NNI; (2) BSNNI $\subset$ ISNNI $\subset$ SNNI. (3) BNDC $\subset$ INDC $\subset$ NDC.

Proof. Immediately from the fact $\sim_{wb}$ is finer than $\sim_{if}$, and $\sim_{if}$ is finer than $\sim_t$. □

Definition 5.3. (1) $E \in \text{SISNNI} \Longleftrightarrow E' \in \text{ISNNI}$ for all E' with $E \Longrightarrow E'$;
(2) $E \in$ P-INDC $\Longleftrightarrow$ for all E' with $E \Longrightarrow E'$, we have $E' \in$ INDC.

Proposition 5.4. SISNNI $\subset$ INDC $\subset$ ISNNI.

Proof. To show that SISNNI $\subset$ INDC, it is enough to prove that if $p \in$ SISNNI, then $p\backslash Act_H \sim_{if} (p|q)\backslash Act_H$ for all $q \in Process_H$, because $p \in$ SISNNI implies $p\backslash Act_H \sim_{if} p/Act_H$. Then, by the distributive law of $\backslash Act_H$ over parallel composition operator, which will be prove in Proposition 6.3.3, $(p|q)\backslash Act_H \sim_{if} p\backslash Act_H|q\backslash Act_H$. Then $p\backslash Act_H|q\backslash Act_H \sim_{if} p\backslash Act_H$ since $p\backslash Act_H$ is an empty process. Finally, we obtain $(p|q)\backslash Act_H \sim_{if} p\backslash Act_H|q\backslash Act_H \sim_{if} p\backslash Act_H$.

INDC $\subset$ ISNNI is trivial from their definitions. □

Proposition 5.5. SISNNI = P-INDC.

The right graph in Figure 2 shows a lattice of the non-interference properties mentioned in the paper. The arrows denote the 'weaker than' relations between two related properties.

6 Compositional Format for the SISNNI Property

6.1 Definition of the SISNNI Format

In this section, a compositional format for the SISNNI property will be proposed. Firstly, we need to define a class of special rules, i.e., patience rules.

Definition 6.1.1 [15,14]. A rule of the form $\dfrac{x_i \xrightarrow{\tau} x_i'}{f(x_1,...,x_i,...,x_n) \xrightarrow{\tau} f(x_1,...,x_i',...,x_n)}$ with $1 \le i \le n$ is called a patience rule of the ith argument of f.

Definition 6.1.2 [29]. An argument $i \in N$ of an operator f is active if f has a rule in which x_i appears as left-hand side of a premise. A variable x occurring in a term t is receiving in t if t is the target of a rule in which x is the right-hand side of a premise. An argument $i \in N$ of an operator f is receiving if a variable x is receiving in a term t that has a subterm $f(t_1,...,t_n)$ with x occurring in t_i.

Definition 6.1.3 [6]. A de Simone language $\mathcal{L}$ is in weak ω-failure format if

1) patience rules are the only rules with τ-premises, and
2) patience rules for active arguments and receiving arguments are necessary.

Definition 6.1.4. A weak ω-failure language $\mathcal{L}$ is in SISNNI format if

1) labels in a rule are either all in set $Act_H \cup \{\tau\}$ or all in set $Act_L \cup \{\tau\}$. It means that a rule is in the form of $\dfrac{\{x_i \xrightarrow{h_i} x_i'\}_{i\in I}}{f(x_1,...,x_n) \xrightarrow{h} g(x_1',...,x_n')}$ or $\dfrac{\{x_i \xrightarrow{l_i} x_i'\}_{i\in I}}{f(x_1,...,x_n) \xrightarrow{l} g(x_1',...,x_n')}$ such that $I \subseteq \{1,...,n\}$, h_i and h are all in set $Act_H \cup \{\tau\}$, l_i and l are all in set $Act_L \cup \{\tau\}$, and
2) No rules in the form $\dfrac{}{f(x_1,...,x_n) \xrightarrow{h} g(x_1',...,x_n')}$ with $h \in Act_H \cup \{\tau\}$ are allowed.

The first clause of Definition 6.1.4 restricts that high level actions and low level actions should not occur at the same time in a rule, which is a standard restriction of the CCS language. In CCS, only the parallel composition operator allows the presence of the two positive premises, and it also needs that the actions of these two premises are corresponding, i.e., if one is action a then the other is action $\bar{a}$.

The second clause of Definition 6.1.4 excludes the high level prefixing operator from the SISNNI format, which is consistent with the fact that high level prefixing operator is not compositional on the SISNNI property.

Moreover, the nondeterministic choice operator of CCS and SPA is also not in SISNNI format. In fact, the nondeterministic choice operator is not invariant under the impossible future equivalence.

Fortunately, as argued in the ACP [19] language, the prefixing operator may be substituted with a sequential composition operator. And the nondeterministic choice operator may be substituted with several operators in CSP [27], including internal choice and external choice. These operators satisfy the SISNNI format.

6.2 Ruloids and Ruloid Theorem

Ruloids and the ruloid theorem originated from the works of Bloom [22,23] for the GSOS format.

For a language $\mathcal{L} = (\Sigma, \Psi)$ in the SISNNI format, the ruloids $\mathscr{R}(C, \alpha)$, for a context $C(x_1, ..., x_n)$ of n holes and an action α, are a set of expressions like the transition rules:

$$\frac{\{x_i \xrightarrow{\alpha_i} x_i'\}_{i \in I}}{C(x_1, ..., x_n) \xrightarrow{\alpha} D(y_1, ..., y_n)} \tag{1}$$

such that D is another context, $y_i = x_i'$ for $i \in I$ and $y_i = x_i$ for $i \notin I$, where $I \subseteq \{1, 2, ..., n\}$. These expressions characterize all possible behaviors of the context $C(x_1, ..., x_n)$ in the language. Moreover, we define $\mathscr{R}(C) = \bigcup_{\alpha \in Act \cup \{\tau\}} \mathscr{R}(C, \alpha)$.

It should be noted that the context D does not need to have exactly n holes. In fact, after leaving out the copying and the lookahead in the de Simone format (the SISNNI format is a subformat of the de Simone format), the number of the holes of D should be less than or equivalent to n. But for convenience, in form (1), we still write it as $D(y_1, ..., y_n)$.

Furthermore, two properties need to be imposed on $\mathscr{R}(C, \alpha)$, we call them soundness property and completeness property, by a little abusing the terminologies.

Definition 6.2.1. Let $\mathcal{L} = (\Sigma, \Psi)$ be a language in the SISNNI format, and $C(x_1, ..., x_n)$ be any context of n holes in $\mathcal{L}$. A set $\mathscr{R}(C, \alpha)$ of ruloids of form (1) are ruloids of context C and action α, with $\alpha \in Act \cup \{\tau\}$, iff

1) Soundness. Let $r \in \mathscr{R}(C, \alpha)$ be a ruloid of form (1). If ζ is a closed Σ substitution such that $\zeta(x_i) \xrightarrow{\alpha_i} \zeta(x_i')$ for all $i \in I$, then there must exist a context D such that $\zeta(C(x_1, ..., x_n)) \xrightarrow{\alpha} \zeta(D(y_1, ..., y_n))$.

2) Completeness. Let ζ be any closed Σ substitution. If $\zeta(C(x_1, ..., x_n)) \xrightarrow{\alpha}$, then there must exist a ruloid r of form (1) in ruloids $\mathscr{R}(C, \alpha)$, and $\zeta(x_i) \xrightarrow{\alpha_i}$ for all $i \in I$.

Here, a strategy to obtain the ruloids for some context C of n holes is possible, and it can be proved that the ruloids obtained by the strategy will satisfy the soundness and completeness properties of Definition 6.2.1, which forms the ruloid theorem.

As a corollary to the ruloids and the ruloid theorem, we may have the following proposition which will be used in the next subsection. This proposition states that each trace of the composite process can be decomposed into traces of its subprocesses.

Proposition 6.2.2. Let $\mathcal{L} = (\Sigma, \Psi)$ be a SISNNI language, and $C(x_1, ..., x_n)$ be any context of n holes. Suppose that ζ is any closed Σ substitution mapping x_i into p_i. If σ is a trace in $\mathcal{T}(C(p_1, ..., p_n))$, then, for all $1 \leq i \leq n$, there should be a trace σ_i in $\mathcal{T}(p_i, \omega)$ such that, when $C(p_1, ..., p_n) \stackrel{\sigma}{\Longrightarrow} C'(p_1', ..., p_n')$, we have $p_i \stackrel{\sigma_i}{\Longrightarrow} p_i'$.

6.3 The Proof of the Compositional Theorem

To ease to proving of following propositions, an alternative characterization on the impossible future equivalence is needed.

Proposition 6.3.1. Let p, q be two processes. For $1 \leq i \leq \omega$, $p \sim_{if} q$ iff

1) for any $\sigma \in \mathcal{T}(p)$ and p' with $p \stackrel{\sigma}{\Longrightarrow} p'$, there exists q' such that $q \stackrel{\sigma}{\Longrightarrow} q'$ and $\mathcal{T}(q') \subseteq \mathcal{T}(p')$, and

2) for any $\sigma \in \mathcal{T}(q)$ and q' with $q \stackrel{\sigma}{\Longrightarrow} q'$, there exists p' such that $p \stackrel{\sigma}{\Longrightarrow} p'$ and $\mathcal{T}(p') \subseteq \mathcal{T}(q')$.

The following two propositions say that, in any SISNNI language, operators $/Act_H$ and $\backslash Act_H$ are both distributive on any context of n holes.

Proposition 6.3.2. Let $\mathcal{L}$ be an SISNNI language, and C be any context of n holes. If $p_1, ..., p_n$ are any processes, then $C(p_1, ..., p_n)/Act_H \sim_{if} C(p_1/Act_H, ..., p_n/Act_H)$.

Proof. The only problem may exist when a ruloid like $\dfrac{\{x_i \stackrel{h_i}{\rightarrow} x_i'\}_{i \in I}}{C'(x_1, ..., x_n) \stackrel{\tau}{\longrightarrow} D(y_1, ..., y_m)}$ is applied, where $I \subseteq \{1, ..., n\}$, $|I| > 1$ and $\forall i \in I : h_i \in Act_H$. Therefore, when $C(p_1, ..., p_n)/Act_H \Longrightarrow C'(p_1', ..., p_n')/Act_H \stackrel{\tau}{\longrightarrow} C''(p_1'', ..., p_n'')/Act_H$, it is possible that $C(p_1/Act_H, ..., p_n/Act_H) \Longrightarrow C'(p_1'/Act_H, ..., p_n'/Act_H) \stackrel{\tau}{\longrightarrow} C_1'(p_{11}'/Act_H, ..., p_{n1}'/Act_H) \stackrel{\tau}{\longrightarrow} C_2'(p_{12}'/Act_H, ..., p_{n2}'/Act_H) \stackrel{\tau}{\longrightarrow} ... \stackrel{\tau}{\longrightarrow} C_k'(p_{1k}'/Act_H, ..., p_{nk}'/Act_H) \equiv D(p_1''/Act_H, ..., p_n''/Act_H)$ occurs in $C(p_1/Act_H, ..., p_n/Act_H)$ with $k = |I|$.

However, observe that all $\mathcal{T}(C_m'(p_{1m}'/Act_H, ..., p_{nm}'/Act_H))$ with $1 \leq m \leq k$ are equivalent. Therefore, each impossible future pair of $C(p_1/Act_H, ..., p_n/Act_H)$ has corresponding impossible future pair of $C(p_1, ..., p_n)/Act_H$, and vice versa. □

Proposition 6.3.3. If $p_1, ..., p_n$ are any processes satisfying the SISNNI property, then $C(p_1, ..., p_n)\backslash Act_H \sim_{if} C(p_1\backslash Act_H, ..., p_n\backslash Act_H)$.

Theorem 6.3.4 [6]. Weak ω-failure format is a congruence format for the impossible future equivalence.

Theorem 6.3.5. SISNNI format is a compositional format for the SISNNI property.

Proof. It is enough to prove that $C(p_1, ..., p_n)/Act_H \sim_{if} C(p_1, ..., p_n)\backslash Act_H$ if $p_i/Act_H \sim_{if} p_i\backslash Act_H$ for all $1 \leq i \leq n$. By Proposition 6.3.2, $C(p_1, ..., p_n)/Act_H \sim_{if} C(p_1/Act_H, ..., p_n/Act_H)$. By Proposition 6.3.3, $C(p_1, ..., p_n)\backslash Act_H \sim_{if} C(p_1\backslash Act_H, ..., p_n\backslash Act_H)$. Now, $C(p_1/Act_H, ..., p_n/Act_H) \sim_{if} C(p_1\backslash Act_H, ..., p_n\backslash Act_H)$ can be guaranteed by Theorem 6.3.4 and $p_i/Act_H \sim_{if} p_i\backslash Act_H$. Finally, we will obtain the conclusion by the transitivity of the impossible future equivalence, because it is an equivalence relation. □

7 Conclusions

We have observed in the paper that, for characterizing the non-interference-like information flow properties, the weak bisimulation may not be the only suitable semantic equivalence, and even may falsely deem some systems insecure though they are actually secure. Then, we present several alternative semantic equivalences and argue that they can play the same role with the weak bisimulation in verifying whether a system

holds some information flow security. Moreover, they will not lead to the faults caused by the weak bisimulation, or at least make less faults than weak bisimulation does.

As the second topic, we propose a rule format to guarantee the compositionality of the SISNNI property, which is an impossible future equivalence based non-interference property. This rule format make little modifications on the congruence format for the impossible future equivalence proposed by the authors. If a language is in this format, then impossible future equivalence is a congruence and SISNNI property is compositional.

References

1. Focardi, R., Gorrieri, R.: Classification of Security Properties(Part I: Information Flow). In: Focardi, R., Gorrieri, R. (eds.) Foundations of Security Analysis and Design. LNCS, vol. 2171, pp. 331–396. Springer, Heidelberg (2001)
2. Rensink, A., Vogler, W.: Fair Testing. Information and Computation 205(2), 125–198 (2007)
3. van Glabbeek, R.J.: The Linear Time - Branching Time Spectrum I: The Semantics of Concrete, Sequential Processes. In: Bergstra, J.A., Ponse, A., Smolka, S.A. (eds.) Handbook of Process Algebra, ch. 1, pp. 3–100. Elsevier, Amsterdam (2001)
4. van Glabbeek, R.J.: The Linear Time - Branching Time Spectrum II: The semantics of sequential systems with silent moves. In: Best, E. (ed.) CONCUR 1993. LNCS, vol. 715, pp. 66–81. Springer, Heidelberg (1993)
5. De Nicola, R., Hennessy, M.: Testing Equivalences for Processes. Theoretical Computer Science 34, 83–133 (1984)
6. Huang, X., Jiao, L., Lu, W.: Rule Formats for Weak Parametric Failure Equivalences (submitted)
7. Ryan, P.Y.A.: Mathematical Models of Computer Security. In: Focardi, R., Gorrieri, R. (eds.) Foundations of Security Analysis and Design. LNCS, vol. 2171, pp. 1–62. Springer, Heidelberg (2001)
8. van der Meyden, R., Zhang, C.: A Comparison of Semantic Models for Noninterference. In: Dimitrakos, T., Martinelli, F., Ryan, P.Y.A, Schneider, S. (eds.) FAST 2006. LNCS, vol. 4691, Springer, Heidelberg (2006)
9. Bell, D.E., Padula, L.J.L.: Secure Computer Systems: Unified Exposition and Multics Interpretation. ESD-TR-75-306, MITRE MTR-2997 (March 1976)
10. Focardi, R., Rossi, S.: Information Flow Security in Dynamic Contexts. In: Proceeding of the IEEE Computer Security Foundations Workshop, pp. 307–319. IEEE Computer Society Press, Los Alamitos (2002)
11. Tini, S.: Rule Formats for Compositional Non-Interference Properties. Journal of Logic and Algebraic Progamming 60, 353–400 (2004)
12. Plotkin, G.D.: A Structural Approach to Operational Semantics. The Journal of Logic and Algebraic Programming, 60–61,17–139 (2004)
13. Milner, R.: Communication and Concurrency. Prentice-Hall, Englewood Cliffs (1989)
14. Mousavi, M.R., Reniers, M.A., Groote, J.F.: SOS formats and meta-theory: 20 years after. Theoretical Computer Science 373, 238–272 (2007)
15. Aceto, L., Fokkink, W.J., Verhoef, C.: Structural Operational Semantics. In: Bergstra, J.A., Ponse, A., Smolka, S.A. (eds.) Handbook of Process Algebra, Ch. 3, pp. 197–292. Elsevier, Amsterdam (2001)
16. Groote, J.F., Waandrager, F.W.: Structural Operational Semantics and Bisimulation as a Congruence. Information and Computation 100(2), 202–260 (1992)
17. Groote, J.F.: Transition System Specifications with Negative Premises. Theoretical Computer Science 118, 263–299 (1993)

18. Simone, R.D.: Higher-level synchronising devices in Meiji-SCCS. Theoretical Computer Science 37, 245–267 (1985)
19. Baeten, J.C.M., Weijiland, W.P.: Process Algebra. Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, Cambridge (1990)
20. Burris, S., Sankappanavar, H.P.: A Course in Universal Algebra. Graduate Texts in Mathematics, vol. 78. Springer, Berlin (1981)
21. Bloom, B., Istrail, S., Meyer, A.R.: Bisimualtion can't be Traced. Journal of the ACM 42(1), 232–268 (1995)
22. Bloom, B.: Structural Operational Semantics for Weak Bisimulations. Theoretical Computer Science 146, 27–68 (1995)
23. Bloom, B.: Ready Simulation, Bisimulation, and the Semantics of CCS-Like Languages. PhD thesis, MIT (1990)
24. Ulidowski, I.: Finite Axiom Systems for Testing Preorder and De Simone Process Languages. Theoretical computer Science 239(1), 97–139 (2000)
25. van Glabbeek, R.J., Voorhoeve, M.: Liveness, Fairness and Impossible Futures. In: Baier, C., Hermanns, H. (eds.) CONCUR 2006. LNCS, vol. 4137, pp. 126–141. Springer, Heidelberg (2006)
26. Voorhoeve, I., Mauw, S.: Impossible Futures and Determinism. Information Processing Letters 80(1), 51–58 (2001)
27. Hoare, C.A.R.: Communicating Sequential Processes. Prentice-Hall, Englewood Cliffs (1985)
28. Bossi, A., Focardi, R., Piazza, C., Rossi, S.: Unwinding in Information Flow Security. Electronic Notes of Theoretical Computer Science 99, 127–154 (2004)
29. van Glabbeek, R.J.: On Cool Congruence Formats for Weak Bisimulations. In: Van Hung, D., Wirsing, M. (eds.) ICTAC 2005. LNCS, vol. 3722, pp. 331–346. Springer, Heidelberg (2005)