Branching Bisimulation Semantics Enables Noninterference Analysis of Reversible Systems

Andrea Esposito, Alessandro Aldini, and Marco Bernardo

Dipartimento di Scienze Pure e Applicate, Università di Urbino, Urbino, Italy

Abstract. The theory of noninterference supports the analysis and the execution of secure computations in multi-level security systems. Classical equivalence-based approaches to noninterference mainly rely on weak bisimulation semantics. We show that this approach is not sufficient to identify potential covert channels in the presence of reversible computations. As illustrated via a database management system example, the activation of backward computations may trigger information flows that are not observable when proceeding in the standard forward direction. To capture the effects of back and forth computations, it is necessary to move to a sufficiently expressive semantics that, in an interleaving framework, has been proven to be branching bisimilarity in a previous work by De Nicola, Montanari, and Vaandrager. In this paper we investigate a taxonomy of noninterference properties based on branching bisimilarity along with their preservation and compositionality features, then we compare it with the classical hierarchy based on weak bisimilarity.

1 Introduction

Noninterference was introduced by Goguen and Meseguer [22] to reason about the way in which illegitimate information flows can occur in multi-level security systems from high-level agents to low-level ones due to covert channels. Since the first definition conceived for deterministic state machines, in the last four decades a lot of work has been done that led to a variety of extensions (dealing with nondeterminism or quantitative domains) in multiple frameworks (from language-based security to concurrency theory); see, e.g., [15,2,32,24,25] and the references therein. Analogously, the techniques proposed to verify information-flow security properties based on noninterference have followed several different approaches, ranging from the application of type theory [44] and abstract interpretation [19] to control flow analysis and equivalence or model checking [16,33,3].

Noninterference guarantees that low-level agents can never infer from their observations what high-level agents are doing. Regardless of its specific definition, noninterference is closely tied to the notion of behavioral equivalence, because the idea is to compare the system behavior with high-level actions being prevented and the system behavior with those actions being hidden. Historically, one of the most established formal definitions of noninterference properties relies on weak bisimilarity in a process algebraic framework [34], as it naturally lends itself to reason formally about covert channels and illegitimate information flows.

While the literature concentrated on weak bisimilarity so far [15], in this paper we claim that it is worth studying nondeterministic noninterference in a different setting, relying on branching bisimulation semantics. Branching bisimilarity was introduced in [21] as a refinement of weak bisimilarity to preserve the branching structure of processes also when abstracting from invisible actions. It features a complete axiomatization whose only τ-axiom is $a.(\tau.(y + z) + y) = a.(y + z)$, where a is an action, τ is an invisible action, and y and z are process terms. Moreover, while weak bisimilarity can be verified in $O(n^2 \cdot m \cdot \log\, n)$, where m is the number of transitions and n is the number of states of the labeled transition system underlying the process at hand, branching bisimilarity can be verified more efficiently. An $O(m \cdot n)$ algorithm has been provided in [23] and, more recently, an even faster $O(m \cdot \log\, n)$ algorithm has been developed in [26].

A clear motivation for passing to branching bisimilarity is provided by the setting of reversible computing – for which no information flow security approach exists to the best of our knowledge – where weak bisimilarity does not represent a proper tool for the comprehensive analysis of covert channels. In this setting, the model of computation features both forward and backward computations, i.e., computational processes are reversible [28,6]. This paradigm has turned out to have interesting applications in computational biology [38,39], parallel discrete-event simulation [36,41], robotics [31], control theory [42], fault tolerant systems [10,12,29,43], and concurrent program debugging [18,30].

Behavioral equivalences for reversible processes must take into account the fact that computations are allowed to proceed not only forward but also backward. To this aim, back-and-forth bisimilarity, introduced in [11], requires that two systems are able to mimic each other's behavior stepwise not only in performing actions that follow the arrows of the labeled transition systems, but also in undoing those actions when going backwards. Formally, back-and-forth bisimulations are defined on computation paths instead of states thus preserving not only causality but also history, as backward moves are constrained to take place along the same path followed in the forward direction even in the presence of concurrency. In [11] it was shown that strong back-and-forth bisimilarity coincides with the usual notion of strong bisimilarity, while weak back-and-forth bisimilarity is surprisingly finer than standard weak bisimilarity, and it coincides with branching bisimilarity. In particular, this latter result will allow us to investigate the nature of covert channels in reversible systems by using a standard process calculus, e.g., without having to decorate executed actions like in [37] or store them into stack-based memories like in [9].

Once established that branching bisimilarity enables noninterference analysis of reversible systems, the novel contribution of this paper is the study of noninterference security properties based on branching bisimilarity. In addition to investigating preservation and compositionality features, we compare the resulting properties with those based on weak bisimilarity [15] and we establish a taxonomy of the former that can be naturally applied to those based on weak back-and-forth bisimilarity for reversible systems. Moreover, we show that, in the setting of reversible systems, weak bisimilarity does not provide a proper frame-

work for the identification of subtle covert channels, while branching bisimilarity does. This is carried out through a database management system example.

This paper is organized as follows. In Section 2, we recall background definitions and results for several bisimulation equivalences and information-flow security properties based on weak bisimilarity, along with a process language to formalize those properties. In Section 3, we introduce the database management system example. In Section 4, we recast the same information-flow security properties in terms of branching bisimilarity, then we present some results about preservation of those properties under branching bisimilarity and compositionality with respect to the operators of the considered language. Moreover, we show results about inclusion among all the previously discussed properties, which are summarized in a new taxonomy. In Section 5, we recall the notion of back-and-forth bisimulation and its relationship with the aforementioned bisimulations, emphasizing that weak back-and-forth bisimilarity coincides with branching bisimilarity, which allows us to apply our results to reversible systems. In Section 6, we add reversibility to the database management system example to illustrate the need of branching-bisimilarity-based noninterference. Finally, in Section 7 we provide some concluding remarks and discuss future work.

2 Background Definitions and Results

In this section, we recall bisimulation equivalences (Section 2.1) and introduce a basic process language (Section 2.2) through which we express bisimulation-based information-flow security properties (Section 2.3).

2.1 Bisimulation Equivalences

To represent the behavior of a process we use a labeled transition system [27], which is a state-transition graph whose transitions are labeled with actions.

Definition 1. *A* labeled transition system (LTS) *is a triple* $(S, A, \longrightarrow)$ *where* $S \neq \emptyset$ *is an at most countable set of states,* $A \neq \emptyset$ *is a countable set of actions, and* $\longrightarrow \subseteq S \times A \times S$ *is a transition relation.* ■

A transition (s, a, s') is written $s \xrightarrow{a} s'$, where s is the source state and s' is the target state. We say that s' is reachable from s, written $s' \in \mathit{reach}(s)$, iff $s' = s$ or there is a sequence of finitely many transitions such that the target state of each of them coincides with the source state of the subsequent one, with the source of the first one being s and the target of the last one being s'.

Strong bisimilarity [34,35] identifies processes that are able to mimic each other's behavior stepwise. This preserves the branching structure of processes.

Definition 2. *Let* $(S, A, \longrightarrow)$ *be an LTS and* $s_1, s_2 \in S$*. We say that* s_1 *and* s_2 *are* strongly bisimilar*, written* $s_1 \sim s_2$*, iff* $(s_1, s_2) \in \mathcal{B}$ *for some strong bisimulation* $\mathcal{B}$*. A symmetric binary relation* $\mathcal{B}$ *over* S *is a* strong bisimulation *iff, whenever* $(s_1, s_2) \in \mathcal{B}$*, then for all actions* $a \in A$*:*

- *whenever* $s_1 \xrightarrow{a} s_1'$*, then* $s_2 \xrightarrow{a} s_2'$ *with* $(s_1', s_2') \in \mathcal{B}$*.* ■

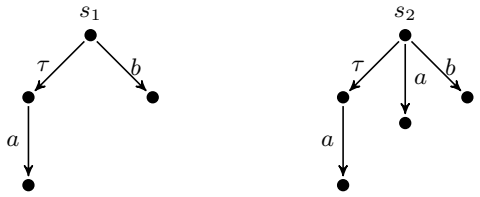

Fig. 1. States s_1 and s_2 are weakly bisimilar but not branching bisimilar

Weak bisimilarity [34] abstracts from unobservable actions, which are denoted by τ. Let $s \stackrel{\tau^*}{\Longrightarrow} s'$ means that $s' \in \mathit{reach}(s)$ and, whenever $s' \neq s$, there is a finite sequence of transitions from s to s' each of which is labeled with τ.

Definition 3. *Let $(S, A, \longrightarrow)$ be an LTS and $s_1, s_2 \in S$. We say that s_1 and s_2 are* weakly bisimilar*, written $s_1 \approx s_2$, iff $(s_1, s_2) \in \mathcal{B}$ for some weak bisimulation $\mathcal{B}$. A symmetric binary relation $\mathcal{B}$ over S is a* weak bisimulation *iff, whenever $(s_1, s_2) \in \mathcal{B}$, then:*

- *whenever $s_1 \stackrel{\tau}{\longrightarrow} s_1'$, then $s_2 \stackrel{\tau^*}{\Longrightarrow} s_2'$ with $(s_1', s_2') \in \mathcal{B}$;*
- *whenever $s_1 \stackrel{a}{\longrightarrow} s_1'$ for $a \in A \setminus \{\tau\}$, then $s_2 \stackrel{\tau^*}{\Longrightarrow} \stackrel{a}{\longrightarrow} \stackrel{\tau^*}{\Longrightarrow} s_2'$ with $(s_1', s_2') \in \mathcal{B}$.* ■

Branching bisimilarity [21] is finer than weak bisimilarity as it preserves the branching structure of the abstracted τ-actions.

Definition 4. *Let $(S, A, \longrightarrow)$ be an LTS and $s_1, s_2 \in S$. We say that s_1 and s_2 are* branching bisimilar*, written $s_1 \approx_{\mathrm{b}} s_2$, iff $(s_1, s_2) \in \mathcal{B}$ for some branching bisimulation $\mathcal{B}$. A symmetric binary relation $\mathcal{B}$ over S is a* branching bisimulation *iff, whenever $(s_1, s_2) \in \mathcal{B}$, then for all actions $a \in A$:*

- *whenever $s_1 \stackrel{a}{\longrightarrow} s_1'$, then:*
  - *either $a = \tau$ and $(s_1', s_2) \in \mathcal{B}$;*
  - *or $s_2 \stackrel{\tau^*}{\Longrightarrow} \bar{s}_2 \stackrel{a}{\longrightarrow} s_2'$ with $(s_1, \bar{s}_2) \in \mathcal{B}$ and $(s_1', s_2') \in \mathcal{B}$.* ■

An example that highlights the higher distinguishing power of branching bisimilarity is given in Figure 1, where every LTS is depicted as a directed graph in which vertices represent states and action-labeled edges represent transitions. The initial states s_1 and s_2 of the LTSs are weakly bisimilar but not branching bisimilar. The only transition that distinguishes s_1 and s_2 is the a-transition of s_2, which can be mimicked by s_1 according to weak bisimilarity by performing the τ-transition followed by the a-transition. However, s_1 cannot respond in the same way according to branching bisimilarity. If s_1 performs the τ-transition followed by the a-transition, then the state reached after the τ-transition should be branching bisimilar to s_2, which is not the case because of the b-transition departing from s_2.

2.2 A Process Calculus with High and Low Actions

We now introduce a basic process calculus to formalize the security properties of interest. To address two security levels, actions are divided into high and low. We denote by $\mathcal{A} = \mathcal{A}_\mathcal{H} \cup \mathcal{A}_\mathcal{L}$ the set of visible actions, where $\mathcal{A}_\mathcal{H} \cap \mathcal{A}_\mathcal{L} = \emptyset$, with $\mathcal{A}_\mathcal{H}$ being the set of high-level actions, ranged over by h, and $\mathcal{A}_\mathcal{L}$ being the set of low-level actions, ranged over by l. Furthermore $\mathcal{A}_\tau = \mathcal{A} \cup \{\tau\}$, where $\tau \notin \mathcal{A}$ is the invisible or silent action.

The set $\mathbb{P}$ of process terms is obtained by considering typical operators from [34,8]. In particular, in addition to the usual operators for sequential, alternative, and parallel compositions, we include restriction and hiding as they are necessary to formalize noninterference properties. The syntax is:

$$P ::= \underline{0} \mid a\,.\,P \mid P + P \mid P \parallel_L P \mid P \setminus L \mid P \,/\, L$$

where:

- $\underline{0}$ is the terminated process.
- $a\,.\,\_$, for $a \in \mathcal{A}_\tau$, is the action prefix operator describing a process that initially performs action a.
- $\_ + \_$ is the alternative composition operator expressing a nondeterministic choice between two processes based on their executable actions.
- $\_ \parallel_L \_$, for $L \subseteq \mathcal{A}$, is the parallel composition operator that forces two processes to synchronize on any action in L.
- $\_ \setminus L$, for $L \subseteq \mathcal{A}$, is the restriction operator, which prevents the execution of actions in L.
- $\_ \,/\, L$, for $L \subseteq \mathcal{A}$, is the hiding operator, which turns all the executed actions in L into the invisible action τ.

The operational semantic rules for the process language are shown in Table 1 and produce the LTS $(\mathbb{P}, \mathcal{A}_\tau, \longrightarrow)$ where $\longrightarrow \subseteq \mathbb{P} \times \mathcal{A}_\tau \times \mathbb{P}$, to which the bisimulation equivalences defined in the previous section are applicable.

2.3 Weak-Bisimilarity-Based Information-Flow Security Properties

The intuition behind noninterference in a two-level security system is that, whenever a group of agents at the high security level performs some actions, the effect of those actions should not be seen by any agent at the low security level. Below is a representative selection of weak-bisimilarity-based noninterference properties – *Nondeterministic Non-Interference* (NNI) and *Non-Deducibility on Composition* (NDC) – followed by their relationships [15], which we then comment.

Definition 5. *Let $P \in \mathbb{P}$:*

- $P \in \text{BSNNI} \iff P \setminus \mathcal{A}_\mathcal{H} \approx P \,/\, \mathcal{A}_\mathcal{H}$.
- $P \in \text{BNDC} \iff$ *for all $Q \in \mathbb{P}$ such that every $Q' \in reach(Q)$ can execute only actions in $\mathcal{A}_\mathcal{H}$ and for all $L \subseteq \mathcal{A}_\mathcal{H}$, $P \setminus \mathcal{A}_\mathcal{H} \approx ((P \parallel_L Q) \,/\, L) \setminus \mathcal{A}_\mathcal{H}$.*
- $P \in \text{SBSNNI} \iff P \in \text{BSNNI}$ *and for all $P' \in reach(P)$, $P' \in \text{BSNNI}$.*
- $P \in \text{SBNDC} \iff$ *for all $P' \in reach(P)$ and for all P'' such that $P' \xrightarrow{a} P''$ for some $a \in \mathcal{A}_\mathcal{H}$, $P' \setminus \mathcal{A}_\mathcal{H} \approx P'' \setminus \mathcal{A}_\mathcal{H}$.* ■

Theorem 1. $\text{SBNDC} \subset \text{SBSNNI} \subset \text{BNDC} \subset \text{BSNNI}$. ■

Prefix	$a \,.\, P \xrightarrow{a} P$
Choice	$\dfrac{P_1 \xrightarrow{a} P_1'}{P_1 + P_2 \xrightarrow{a} P_1'} \qquad \dfrac{P_2 \xrightarrow{a} P_2'}{P_1 + P_2 \xrightarrow{a} P_2'}$
Synchronization	$\dfrac{P_1 \xrightarrow{a} P_1' \quad P_2 \xrightarrow{a} P_2' \quad a \in L}{P_1 \parallel_L P_2 \xrightarrow{a} P_1' \parallel_L P_2'}$
Interleaving	$\dfrac{P_1 \xrightarrow{a} P_1' \quad a \notin L}{P_1 \parallel_L P_2 \xrightarrow{a} P_1' \parallel_L P_2} \qquad \dfrac{P_2 \xrightarrow{a} P_2' \quad a \notin L}{P_1 \parallel_L P_2 \xrightarrow{a} P_1 \parallel_L P_2'}$
Restriction	$\dfrac{P \xrightarrow{a} P' \quad a \notin L}{P \setminus L \xrightarrow{a} P' \setminus L}$
Hiding	$\dfrac{P \xrightarrow{a} P' \quad a \in L}{P/L \xrightarrow{\tau} P'/L} \qquad \dfrac{P \xrightarrow{a} P' \quad a \notin L}{P/L \xrightarrow{a} P'/L}$

Table 1. Operational semantic rules

Historically, one of the first and most intuitive proposals is the *Bisimulation-based Strong Nondeterministic Non-Interference* (BSNNI). Basically, it is satisfied by any process P that behaves the same when its high-level actions are prevented (as modeled by $P \setminus \mathcal{A}_{\mathcal{H}}$) or when they are considered as hidden, unobservable actions (as modeled by $P \,/\, \mathcal{A}_{\mathcal{H}}$). The equivalence between these two low-level views of P states that a low-level observer cannot distinguish the high-level behavior of the system. For instance, in $l \,.\, \underline{0} + h \,.\, l \,.\, \underline{0}$ a low-level agent that observes the execution of l cannot infer anything about the execution of h. Indeed, $(l \,.\, \underline{0} + h \,.\, l \,.\, \underline{0}) \setminus \{h\} \approx (l \,.\, \underline{0} + h \,.\, l \,.\, \underline{0}) \,/\, \{h\}$ because $l \,.\, \underline{0} \approx l \,.\, \underline{0} + \tau \,.\, l \,.\, \underline{0}$.

BSNNI is not powerful enough to capture covert channels that derive from the behavior of the high-level agent interacting with the system. For instance, $l \,.\, \underline{0} + h_1 \,.\, h_2 \,.\, l \,.\, \underline{0}$ is BSNNI for the same reason discussed above. However, a high-level agent could decide to enable h_1 and then disable h_2, thus turning the low-level view of the system into $l \,.\, \underline{0} + \tau \,.\, \underline{0}$, which is clearly distinguishable from $l \,.\, \underline{0}$, as only in the former the low-level observer may not observe l. To overcome such a limitation, the most obvious solution consists of checking explicitly the interaction between the system and every possible high-level agent Q. The resulting property is the *Bisimulation-based Non-Deducibility on Composition* (BNDC), which is characterized by a universal quantification over Q.

To circumvent the verification problems related to such a quantifier, several properties have been proposed that are stronger than BNDC. They all express some persistency conditions, stating that the security checks shall be somehow extended also to the derivatives of a secure process. Three of the most representative ones are the variant of BSNNI that requires every reachable state to satisfy BSNNI itself, called *Strong* BSNNI (SBSNNI), the variant of BNDC that requires every reachable state to satisfy BNDC, called *Persistent* BNDC

(P_BNDC), and the *Strong* BNDC (SBNDC), which requires the low-level view of every reachable state to be the same before and after the execution of any high-level action. Notice that the SBNDC condition states that the execution of high-level actions must be completely transparent to the low-level agents. The properties P_BNDC and SBSNNI have been proven to be equivalent in [17], hence we will focus only on SBSNNI.

3 Use Case: DBMS Transactions – Part I

Consider a multi-threaded system supporting the execution of concurrent transactions operating on a healthcare database. Authorized users can write data on such a database, which is then accessed by a dedicated module to feed the training set for a machine learning model built for data analysis purposes.

On the one hand, different authentication mechanisms can be employed to identify users and ensure data authenticity for each transaction. We address a simple password-based mechanism (*pwd*), a more sophisticated two-factor authentication system (*2fa*), and finally a scheme based on single sign on (*sso*) [7]. On the other hand, to protect the privacy of health data in the trained model, only data transmitted through a highly secure mechanism, i.e., *2fa* or *sso*, can be used to feed the training set. In any case, users must not be aware of which data are actually chosen to train the machine learning model [5]. To this aim, the database management system (DBMS) is enabled to internally and transparently decide not to consider for the training set some transactions.

A simplified model describing how a write transaction is handled by the considered DBMS is represented by the following process term, whose LTS is depicted in Figure 2:

$$WT \;:=\; l_{pwd} \,.\, \underline{0} + \tau \,.\, (\tau \,.\, l_{sso} \,.\, \underline{0} + \tau \,.\, l_{2fa} \,.\, \underline{0}) + (h \,.\, l_{sso} \,.\, \underline{0} + h \,.\, l_{2fa} \,.\, \underline{0})$$

The low-level actions of the form $l_\star$ express that the transaction is conducted under the authentication method represented by $\star$, while the high-level action h expresses a private interaction with the machine learning module intended to avoid the transfer of the transaction data to the training set.

The DBMS is ready to manage the transaction through the password-based mechanism, as described by subterm $l_{pwd} \,.\, \underline{0}$. Alternatively, it internally decides that the transaction data will be passed to the training set and, therefore, one of the two highly secure mechanisms must be chosen nondeterministically, as described by subterm $\tau \,.\, (\tau \,.\, l_{sso} \,.\, \underline{0} + \tau \,.\, l_{2fa} \,.\, \underline{0})$. Otherwise, it can interact with the machine learning module, while nondeterministically choosing one of the two highly secure mechanisms, as described by subterm $h \,.\, l_{sso} \,.\, \underline{0} + h \,.\, l_{2fa} \,.\, \underline{0}$. This interaction is intended to confuse the user, who should not infer whether the transaction data will be used for the training set or not by simply observing which kind of authentication is required by the DBMS. This privacy condition is ensured if the interaction with the machine learning module does not interfere with the low-level view of the system perceived by the user, which can be verified as a noninterference property.

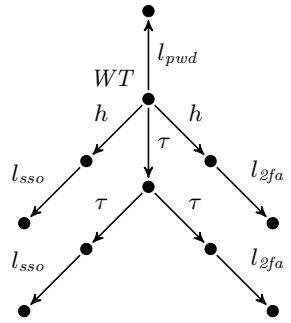

Fig. 2. LTS underlying WT

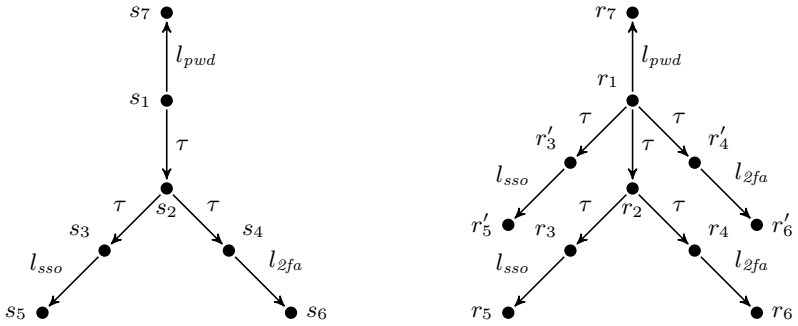

Fig. 3. LTSs of the low-level views of WT: $WT \setminus \mathcal{A}_{\mathcal{H}}$ (left) and $WT/\mathcal{A}_{\mathcal{H}}$ (right)

As far as $\approx$-based noninterference is concerned, WT does not leak any information from high level to low level. Indeed, the system is SBSNNI and hence also BNDC and BSNNI by virtue of Theorem 1. First, by observing Figure 3, it is easy to see that $WT \setminus \mathcal{A}_{\mathcal{H}} \approx WT / \mathcal{A}_{\mathcal{H}}$. The weak bisimulation relating the two low-level views of WT is given by the following partition of the state space:

$$\{\{s_1, r_1\}, \{s_2, r_2\}, \{s_3, r_3, r'_3\}, \{s_4, r_4, r'_4\}, \{s_5, r_5, r'_5, s_6, r_6, r'_6, s_7, r_7\}\}$$

Then, by observing that the only high-level action is enabled at the initial state of WT, it follows that WT is SBSNNI.

4 Security Properties Based on Branching Bisimilarity

While the literature on noninterference mainly concentrates on weak bisimulation semantics, in this section we recast information-flow security definitions in terms of branching bisimilarity and investigate their characteristics as well as their relationships with the definitions based on weak bisimilarity.

The noninterference properties that reformulate the ones in Definition 5 by replacing the weak bisimilarity check with the branching bisimilarity check are termed, respectively, BrSNNI, BrNDC, SBrSNNI, and SBrNDC.

4.1 Preservation and Compositionality

All the $\approx_{\rm b}$-based noninterference properties turn out to be preserved by $\approx_{\rm b}$. This means that, whenever a process P_1 is secure under any of such properties, then every other branching bisimilar process P_2 is secure too. This is very useful for automated property verification, as it allows one to work with the process with the smallest state space among the equivalent ones.

Theorem 2. *Let* $P_1, P_2 \in \mathbb{P}$ *and* $\mathcal{P} \in \{\text{BrSNNI}, \text{BrNDC}, \text{SBrSNNI}, \text{SBrNDC}\}$. *If* $P_1 \approx_{\rm b} P_2$, *then* $P_1 \in \mathcal{P} \iff P_2 \in \mathcal{P}$. ■

As far as modular verification is concerned, like in the weak bisimilarity case [15] only the local properties SBrSNNI and SBrNDC are compositional, i.e., are preserved by the operators of the calculus.

Theorem 3. *Let* $P, P_1, P_2 \in \mathbb{P}$ *and* $\mathcal{P} \in \{\text{SBrSNNI}, \text{SBrNDC}\}$. *Then:*

1. $P \in \mathcal{P} \Longrightarrow a\,.\,P \in \mathcal{P}$ *for all* $a \in \mathcal{A}_\tau \setminus \mathcal{A}_\mathcal{H}$.
2. $P_1, P_2 \in \mathcal{P} \Longrightarrow P_1 \parallel_L P_2 \in \mathcal{P}$ *for all* $L \subseteq \mathcal{A}$.
3. $P \in \mathcal{P} \Longrightarrow P \setminus L \in \mathcal{P}$ *for all* $L \subseteq \mathcal{A}_\mathcal{L}$ *if* $\mathcal{P} = \text{SBrSNNI}$, $L \subseteq \mathcal{A}$ *if* $\mathcal{P} = \text{SBrNDC}$.
4. $P \in \mathcal{P} \Longrightarrow P / L \in \mathcal{P}$ *for all* $L \subseteq \mathcal{A}_\mathcal{L}$. ■

Note that, like for weak bisimilarity, no property based on branching bisimilarity is compositional with respect to alternative composition. As an example, let us consider processes $P_1 := l\,.\,\underline{0}$ and $P_2 := h\,.\,\underline{0}$. Both are BrSNNI, as $l\,.\,\underline{0} \setminus \{h\} \approx_{\rm b} l\,.\,\underline{0} / \{h\}$ and $h\,.\,\underline{0} \setminus \{h\} \approx_{\rm b} h\,.\,\underline{0} / \{h\}$, but $P_1 + P_2 \notin$ BrSNNI because $(l\,.\,\underline{0} + h\,.\,\underline{0}) \setminus \{h\} \approx_{\rm b} l\,.\,\underline{0} \not\approx_{\rm b} l\,.\,\underline{0} + \tau\,.\,\underline{0} \approx_{\rm b} (l\,.\,\underline{0} + h\,.\,\underline{0}) / \{h\}$. It can be easily checked that $P_1 + P_2 \notin \mathcal{P}$ for $\mathcal{P} = \{\text{BrNDC}, \text{SBrSNNI}, \text{SBrNDC}\}$.

We point out that compositionality with respect to action prefix and hiding, although limited to non-high actions and low actions respectively, is established by Theorem 3 but was not investigated in [15] under weak bisimilarity.

4.2 Taxonomy of Security Properties

First of all, the relationships among the $\approx_{\rm b}$-based noninterference properties follow the same pattern as Theorem 1.

Theorem 4. SBrNDC $\subset$ SBrSNNI $\subset$ BrNDC $\subset$ BrSNNI. ■

All the inclusions above are strict as we now show:

- The process $\tau\,.\,l\,.\,\underline{0} + l\,.\,l\,.\,\underline{0} + h\,.\,l\,.\,\underline{0}$ is SBrSNNI because $(\tau\,.\,l\,.\,\underline{0} + l\,.\,l\,.\,\underline{0} + h\,.\,l\,.\,\underline{0}) \setminus \{h\} \approx_{\rm b} (\tau\,.\,l\,.\,\underline{0} + l\,.\,l\,.\,\underline{0} + h\,.\,l\,.\,\underline{0}) / \{h\}$ and action h is enabled only by the initial process so every derivative is BrSNNI. It is not SBrNDC because the low-level view of the process reached after action h, i.e., $(l\,.\,\underline{0}) \setminus \{h\}$, is not branching bisimilar to $(\tau\,.\,l\,.\,\underline{0} + l\,.\,l\,.\,\underline{0} + h\,.\,l\,.\,\underline{0}) \setminus \{h\}$.

- The process $l\,.\,\underline{0}+l\,.\,l\,.\,\underline{0}+l\,.\,h\,.\,l\,.\,\underline{0}$ is BrNDC because, whether there are synchronizations with high-level actions or not, the overall process can always perform either an l-action or a sequence of two l-actions without incurring any problematic branching. The process is not SBrSNNI because the reachable process $h\,.\,l\,.\,\underline{0}$ is not BrSNNI.
- The process $l\,.\,\underline{0}+h\,.\,h\,.\,l\,.\,\underline{0}$ is BrSNNI due to $(l\,.\,\underline{0}+h\,.\,h\,.\,l\,.\,\underline{0})\backslash\{h\} \approx_{\rm b} (l\,.\,\underline{0}+h\,.\,h\,.\,l\,.\,\underline{0})/\{h\}$, but is not BrNDC due to $(((l\,.\,\underline{0}+h\,.\,h\,.\,l\,.\,\underline{0})\,\|_{\{h\}}(h\,.\,\underline{0}))/\{h\})\setminus\{h\} \not\approx_{\rm b} (l\,.\,\underline{0}+h\,.\,h\,.\,l\,.\,\underline{0})\setminus\{h\}$ as $(l\,.\,\underline{0}+h\,.\,h\,.\,l\,.\,\underline{0})\setminus\{h\}$ behaves as $l\,.\,\underline{0}$.

Secondly, we observe that all the $\approx_{\rm b}$-based noninterference properties listed in Theorem 4 imply the corresponding properties listed in Definition 5 due to the fact that $\approx_{\rm b}$ is finer than $\approx$ [21].

Theorem 5. *The following properties hold:*

1. BrSNNI $\subset$ BSNNI.
2. BrNDC $\subset$ BNDC.
3. SBrSNNI $\subset$ SBSNNI.
4. SBrNDC $\subset$ SBNDC. ■

All the inclusions above are strict due to the following result.

Theorem 6. *Let $P_1, P_2 \in \mathbb{P}$ be such that $P_1 \approx P_2$ but $P_1 \not\approx_{\rm b} P_2$. If P_1 and P_2 do not include high-level actions, then $Q \in \{P_1 + h\,.\,P_2, P_2 + h\,.\,P_1\}$ is such that:*

1. $Q \in$ BSNNI *but* $Q \notin$ BrSNNI.
2. $Q \in$ BNDC *but* $Q \notin$ BrNDC.
3. $Q \in$ SBSNNI *but* $Q \notin$ SBrSNNI.
4. $Q \in$ SBNDC *but* $Q \notin$ SBrNDC. ■

An alternative strategy to explore the differences between $\approx$ and $\approx_{\rm b}$ with respect to B/BrSNNI and SB/BrSNNI is to consider the two τ-axioms $\tau\,.\,x+x = \tau\,.\,x$ and $a\,.\,(\tau\,.\,x+y)+a\,.\,x = a\,.\,(\tau\,.\,x+y)$ for $\approx$ [34]. The strategy is inspired by the initial remarks in [21], where it is noted that the two mentioned axioms are not valid for $\approx_{\rm b}$ and are responsible for the lack of distinguishing power of $\approx$ over τ-branching processes. For each axiom, the strategy consists of constructing a pair of new processes from the ones equated in the axiom, such that they are weakly bisimilar by construction but not branching bisimilar. Then from this pair of processes we define a new process P such that $P\setminus\mathcal{A}_{\mathcal{H}}$ and $P\,/\,\mathcal{A}_{\mathcal{H}}$ are isomorphic to the constructed processes.

Theorem 7. *From $\tau\,.\,x+x = \tau\,.\,x$ it is possible to construct $P \in \mathbb{P}$ such that $P \in$ BSNNI but $P \notin$ BrSNNI and $P \in$ SBSNNI but $P \notin$ SBrSNNI.* ■

Theorem 8. *From $a\,.\,(\tau\,.\,x+y)+a\,.\,x = a\,.\,(\tau\,.\,x+y)$ it is possible to construct $P \in \mathbb{P}$ such that $P \in$ BSNNI but $P \notin$ BrSNNI and $P \in$ SBSNNI but $P \notin$ SBrSNNI.* ■

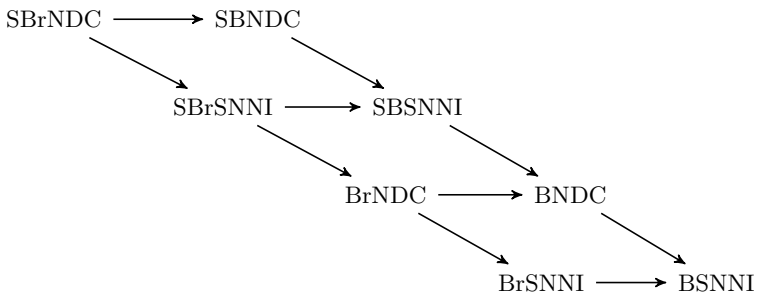

Fig. 4. Taxonomy of security properties based on weak and branching bisimilarities

Based on the results in Theorems 1, 4, and 5, the diagram in Figure 4 summarizes the inclusions among the various noninterference properties, where $\mathcal{P} \to \mathcal{Q}$ means that $\mathcal{P}$ is strictly included in $\mathcal{Q}$. The missing arrows in the diagram, witnessing incomparability, are justified by the following counterexamples:

- SBNDC vs. SBrSNNI. The process $\tau \,.\, l \,.\, \underline{0} + l \,.\, l \,.\, \underline{0} + h \,.\, l \,.\, \underline{0}$ is BrSNNI as $\tau \,.\, l \,.\, \underline{0} + l \,.\, l \,.\, \underline{0} \approx_{\mathrm{b}} \tau \,.\, l \,.\, \underline{0} + l \,.\, l \,.\, \underline{0} + \tau \,.\, l \,.\, \underline{0}$. It is also SBrSNNI because every reachable state does not enable any more high-level actions. However it is not SBNDC, because after the process has performed the high-level action h it can perform a single action l, while the original process with the restriction on high-level actions can go down a path where it performs two l-actions. On the other hand, the process Q mentioned in Theorem 6 is SBNDC but neither BrSNNI nor SBrSNNI.
- SBSNNI vs. BrNDC. The process $l \,.\, h \,.\, l \,.\, \underline{0} + l \,.\, \underline{0} + l \,.\, l \,.\, \underline{0}$ is BrSNNI as $l \,.\, \underline{0} + l \,.\, \underline{0} + l \,.\, l \,.\, \underline{0} \approx_{\mathrm{b}} l \,.\, \tau \,.\, l \,.\, \underline{0} + l \,.\, \underline{0} + l \,.\, l \,.\, \underline{0}$. In particular, the subprocesses $l \,.\, \tau \,.\, l \,.\, \underline{0}$ and $l \,.\, l \,.\, \underline{0}$ are equated by virtue of the other axiom of weak bisimilarity, $a \,.\, \tau \,.\, x = a \,.\, x$, which holds also for branching bisimilarity. The same process is also BrNDC as it includes only one high-level action, hence the only possible high-level strategy coincides with the check conducted by BrSNNI. However, the process is not SBSNNI because of the reachable process $h \,.\, l \,.\, \underline{0}$, which is not BSNNI. On the other hand, the process Q mentioned in Theorem 6 is SBSNNI but not BrSNNI and, therefore, cannot be BrNDC.
- BNDC vs. BrSNNI. The process $l \,.\, \underline{0} + h_1 \,.\, h_2 \,.\, l \,.\, \underline{0}$ is not BNDC (see Section 2.3), but it is BrSNNI as $l \,.\, \underline{0} \approx_{\mathrm{b}} l \,.\, \underline{0} + \tau \,.\, \tau \,.\, l \,.\, \underline{0}$. In contrast, the process Q mentioned in Theorem 6 is both BSNNI and BNDC, but not BrSNNI.

It is worth noting that the strongest property based on weak bisimilarity (SBNDC) and the weakest property based on branching bisimilarity (BrSNNI) are incomparable. The former is a very restrictive property because it requires a local check every time a high-level action is performed, while the latter requires a check only on the initial state. On the other hand, as shown in Theorem 6 it is very easy to construct processes that are secure under properties based on $\approx$ but not on $\approx_{\mathrm{b}}$, due to the minimal number of high-level actions in Q.

5 Noninterference in Reversible Processes

As anticipated, we use reversible computing to motivate the study of branching-bisimilarity-based noninterference properties. To this aim, we now recall from [11] back-and-forth bisimilarity and its relationship with standard bisimilarity.

An LTS represents a reversible process if each of its transitions is seen as bidirectional. This means that any transition can be undone and that any undone transition can be redone. When going backward, it is of paramount importance to respect causality. While this is straightforward for sequential processes, it is not obvious for concurrent ones, because the last performed action is the first one to be undone but this action may not necessarily be identifiable uniquely in the presence of concurrency.

Consider for example a process that can perform action a in parallel with action b. This process can be represented as a diamond-like LTS where from the initial state an a-transition and a b-transition depart, which are respectively followed by a b-transition and an a-transition, both of which reach the final state. Suppose that action a completes before action b, so that the a-transition is executed before the b-transition. Once in the final state, either the b-transition is undone before the a-transition, or the a-transition is undone before the b-transition. Both options are causally consistent, as a and b are independent of each other, but only the former is history preserving too.

The history-preserving option is the one that was addressed in [11] in order to study reversible processes in an interleaving setting. To accomplish this, strong and weak bisimulations were redefined as binary relations between histories, formalized below as runs, instead of states. The resulting behavioral equivalences are respectively called strong and weak back-and-forth bisimilarities in [11].

Definition 6. *A sequence* $\xi = s_0 \xrightarrow{a_1} s_1 \xrightarrow{a_2} s_2 \ldots s_{n-1} \xrightarrow{a_n} s_n$ *is called a* path *from state* s_0 *of length* $n \in \mathbb{N}$*. We let* $first(\xi) = s_0$ *and* $last(\xi) = s_n$*; the empty path is indicated with* ε*. We denote by* $Path(s)$ *the set of paths from state* s*.* ■

Definition 7. *A pair* $\rho = (s, \xi)$ *is called a* run *from state* s *iff* $\xi \in Path(s)$*, in which case we let* $path(\rho) = \xi$*,* $first(\rho) = first(\xi)$*,* $last(\rho) = last(\xi)$*, with* $first(\rho) = last(\rho) = s$ *when* $\xi = \varepsilon$*. We denote by* $Run(s)$ *the set of runs from state* s*.* ■

Definition 8. *Let* $\rho = (s, \xi) \in Run(s)$ *and* $\rho' = (s', \xi') \in Run(s')$*:*

- *Their composition* $\rho\rho' = (s, \xi\xi') \in Run(s)$ *is defined iff* $last(\rho) = first(\rho')$*.*
- *We write* $\rho \xrightarrow{a} \rho'$ *iff there exists* $\rho'' = (s, s \xrightarrow{a} s')$ *with* $s = last(\rho)$ *such that* $\rho' = \rho\rho''$*.* ■

In the behavioral equivalences of [11], for the LTS $(S, A, \longrightarrow)$ the set R of its runs is considered in lieu of the set S of its states.

Definition 9. *Let* $(S, A, \longrightarrow)$ *be an LTS and* $s_1, s_2 \in S$*. We say that* s_1 *and* s_2 *are* strongly back-and-forth bisimilar*, written* $s_1 \sim_{\rm bf} s_2$*, iff* $((s_1, \varepsilon), (s_2, \varepsilon)) \in \mathcal{B}$

for some strong back-and-forth bisimulation $\mathcal{B}$. A symmetric binary relation $\mathcal{B}$ over R is a strong back-and-forth bisimulation *iff, whenever $(\rho_1, \rho_2) \in \mathcal{B}$, then for all actions $a \in A$:*

- *whenever $\rho_1 \xrightarrow{a} \rho_1'$, then $\rho_2 \xrightarrow{a} \rho_2'$ with $(\rho_1', \rho_2') \in \mathcal{B}$;*
- *whenever $\rho_1' \xrightarrow{a} \rho_1$, then $\rho_2' \xrightarrow{a} \rho_2$ with $(\rho_1', \rho_2') \in \mathcal{B}$.* ■

Definition 10. *Let $(S, A, \longrightarrow)$ be an LTS and $s_1, s_2 \in S$. We say that s_1 and s_2 are* weakly back-and-forth bisimilar*, written $s_1 \approx_{\rm bf} s_2$, iff $((s_1, \varepsilon), (s_2, \varepsilon)) \in \mathcal{B}$ for some weak back-and-forth bisimulation $\mathcal{B}$. A symmetric binary relation $\mathcal{B}$ over R is a* weak back-and-forth bisimulation *iff, whenever $(\rho_1, \rho_2) \in \mathcal{B}$, then:*

- *whenever $\rho_1 \xrightarrow{\tau} \rho_1'$, then $\rho_2 \stackrel{\tau^*}{\Longrightarrow} \rho_2'$ with $(\rho_1', \rho_2') \in \mathcal{B}$;*
- *whenever $\rho_1' \xrightarrow{\tau} \rho_1$, then $\rho_2' \stackrel{\tau^*}{\Longrightarrow} \rho_2$ with $(\rho_1', \rho_2') \in \mathcal{B}$;*
- *whenever $\rho_1 \xrightarrow{a} \rho_1'$ for $a \in A \setminus \{\tau\}$, then $\rho_2 \stackrel{\tau^*}{\Longrightarrow} \xrightarrow{a} \stackrel{\tau^*}{\Longrightarrow} \rho_2'$ with $(\rho_1', \rho_2') \in \mathcal{B}$;*
- *whenever $\rho_1' \xrightarrow{a} \rho_1$ for $a \in A \setminus \{\tau\}$, then $\rho_2' \stackrel{\tau^*}{\Longrightarrow} \xrightarrow{a} \stackrel{\tau^*}{\Longrightarrow} \rho_2$ with $(\rho_1', \rho_2') \in \mathcal{B}$.*

■

In [11] it was shown that strong back-and-forth bisimilarity coincides with strong bisimilarity. Surprisingly, weak back-and-forth bisimilarity does not coincide with weak bisimilarity. Instead, it coincides with branching bisimilarity.

Theorem 9. *Let $(S, A, \longrightarrow)$ be an LTS and $s_1, s_2 \in S$. Then:*

- *$s_1 \sim_{\rm bf} s_2$ iff $s_1 \sim s_2$.*
- *$s_1 \approx_{\rm bf} s_2$ iff $s_1 \approx_{\rm b} s_2$.* ■

As a consequence, the properties BrSNNI, BrNDC, SBrSNNI, and SBrNDC do not change if $\approx_{\rm b}$ is replaced by $\approx_{\rm bf}$. This allows us to study noninterference properties for reversible systems by using $\approx_{\rm b}$ in a standard process calculus like the one of Section 2.2, without having to decorate executed actions like in [37] or store them into stack-based memories like in [9].

6 Use Case: DBMS Transactions – Part II

The example provided in Section 3 is useful to illustrate the limitations of weak bisimilarity when investigating potential covert channels in reversible systems.

It turns out that $WT \setminus \mathcal{A}_{\mathcal{H}} \not\approx_{\rm b} WT / \mathcal{A}_{\mathcal{H}}$, i.e., WT is not BrSNNI, and hence not even BrNDC, SBrSNNI, and SBrNDC by virtue of Theorem 4. As can be seen in Figure 3, the reason is that, if $WT/\mathcal{A}_{\mathcal{H}}$ performs the leftmost τ-action and hence moves to state r_3', from which the only executable action is l_{sso}, then according to the definition of branching bisimilarity $WT \setminus \mathcal{A}_{\mathcal{H}}$ can either:

1. stay idle, but from that state $WT \setminus \mathcal{A}_{\mathcal{H}}$ can then perform actions other than l_{sso} that cannot be matched on the side of $WT/\mathcal{A}_{\mathcal{H}}$;

2. perform two τ-actions thereby reaching state s_3, but the traversed state s_2 is not branching bisimilar to the initial state of $WT/\mathcal{A}_{\mathcal{H}}$.

In a standard model of execution, where the computation can proceed only forward, the distinguishing power of branching bisimilarity may be considered too severe, as no practical covert channel actually occurs and the system can be considered noninterfering as shown in Section 3. Indeed, a low-level user has no possibility of distinguishing the internal move performed by $WT/\mathcal{A}_{\mathcal{H}}$ and leading to $l_{sso} \,.\, \underline{0}$ from the sequence of internal moves performed by $WT \setminus \mathcal{A}_{\mathcal{H}}$ and leading to $l_{sso} \,.\, \underline{0}$ as well. This motivates the fact that, historically, weak bisimilarity has been preferred in the setting of noninterference.

Now we know that, if we replace the branching bisimulation semantics with the weak back-and-forth bisimulation semantics, nothing changes about the outcome of noninterference verification. Assuming that the DBMS allows transactions to be reversed, it is instructive to discuss why BrSNNI is not satisfied by following the formalization of the weak back-and-forth bisimulation semantics provided in Section 5.

After $WT/\mathcal{A}_{\mathcal{H}}$ performs the run $(r_1, (r_1 \xrightarrow{\tau} r'_3 \xrightarrow{l_{sso}} r'_5))$, process $WT\setminus\mathcal{A}_{\mathcal{H}}$ can respond by performing the run $(s_1, (s_1 \xrightarrow{\tau} s_2 \xrightarrow{\tau} s_3 \xrightarrow{l_{sso}} s_5))$. If either process goes back by undoing l_{sso}, then the other one can undo l_{sso} as well and the states r'_3 and s_3 are reached. However, if $WT \setminus \mathcal{A}_{\mathcal{H}}$ goes further back by undoing $s_2 \xrightarrow{\tau} s_3$, then $WT/\mathcal{A}_{\mathcal{H}}$ can either:

- undo $r_1 \xrightarrow{\tau} r'_3$, but in this case r_1 enables action l_{pwd} while s_2 does not;
- stay idle, but in this case r'_3 enables only l_{sso}, while s_2 can go down the path $s_2 \xrightarrow{\tau} s_4 \xrightarrow{l_{2fa}} s_6$ as well.

This line of reasoning immediately allows us to reveal a potential covert channel under reversible computing. In fact, let us assume that the transaction modeled by WT is not only executed forward, but also enables backward computations triggered, e.g., whenever debugging mode is activated. This may happen in response to some user-level malfunctioning, which may be due, for instance, to the authentication operation or to the transaction execution. As formally shown above, if the action l_{sso} performed just after the high-level interaction is undone, then the system enables again the execution of the action l_{pwd}. This is motivated in our example by the fact that, in any case, the transaction data will not be transferred to the training set, so that any kind of authentication is admissible. On the other hand, this is not possible by undoing the action l_{sso} departing from state r_3 in $WT/\mathcal{A}_{\mathcal{H}}$, because in such a case the transaction data must be protected through a highly secure mechanism. In other words, by reversing the computation the low-level user can become aware of the fact that the transaction data are feeding the training set or not.

In the literature, there are several reverse debuggers working in this way like, e.g., UndoDB [13], a Linux-based interactive time-travel debugger that can handle multiple threads and their backward execution. For instance, it is integrated

within the DBMS SAP HANA [1] in order to reduce time-to-resolution of software failures. In our example, by virtue of the observations conducted above, if the system is executed backward just after performing l_{sso}, a low-level user can decide whether a high-level action had occurred before or not, thus revealing a covert channel. Such a covert channel is completely concealed during the forward execution of the system and is detected only when the system is executed backward. More in general, this may happen when the reverse debugger is activated by virtue of some unexpected event (e.g., segmentation faults, stack overflow errors, memory corruption) caused intentionally or not, and by virtue of which some undesired information flow emerges towards the low-level users.

7 Conclusions

Our study of branching-bisimilarity-based noninterference properties has established a connection with reversible computing in the sense that those properties, which we have investigated in a standard process calculus, are directly applicable to reversible systems. To the best of our knowledge, this is the first attempt of defining noninterference properties relying on branching bisimilarity and of reasoning about covert channels in reversible systems.

Firstly, we have rephrased in the setting of branching bisimilarity the classical taxonomy of nondeterministic noninterference properties based on weak bisimilarity. This generates an extended taxonomy that is conservative with respect to the classical one and emphasizes the strictness of certain inclusions as well as the incomparability of certain properties. In addition, we have studied preservation and compositionality features of the new noninterference properties.

Secondly, we have shown that potential covert channels arising in reversible systems cannot be revealed by employing weak bisimulation semantics. Indeed, the higher discriminating power of branching bisimilarity is necessary to capture information flows emerging whenever backward computations are admitted. The correspondence discovered in [11] between branching bisimilarity and weak back-and-forth bisimilarity confirms the adequacy of our approach.

As for future work, we are planning to further extend the noninterference taxonomy to include more expressive properties taking into account quantitative aspects of processes [4,25]. Moreover, unlike the corresponding results for weak-bisimilarity-based noninterference properties, whose proofs rely on the use of an up-to technique for weak bisimilarity [40], in Theorems 3 and 4 we have proceeded by induction on the depth of the tree-like LTS underlying the considered process term, because the up-to techniques for branching bisimilarity [20,14] seem to be too restrictive with respect to the conditions required by the noninterference checks. Thus, we would like to study whether there exist suitable relaxations of the latter techniques so as to be able to reformulate the proofs of Theorems 3 and 4 accordingly, which would open the way to including recursion in the language.

Acknowledgment. This research has been supported by the PRIN project *NiRvAna – Noninterference and Reversibility Analysis in Private Blockchains.*

References

1. UndoDB case studies, https://undo.io/resources/type/case-studies/, last visited in March 2023
2. Aldini, A.: Classification of security properties in a Linda-like process algebra. Science of Computer Programming **63**(1), 16–38 (2006)
3. Aldini, A., Bernardo, M.: Component-oriented verification of noninterference. Journal of Systems Architecture **57**(3), 282–293 (2011)
4. Aldini, A., Bernardo, M., Corradini, F.: A Process Algebraic Approach to Software Architecture Design. Springer (2010)
5. Bai, Y., Fan, M., Li, Y., Xie, C.: Privacy risk assessment of training data in machine learning. In: Proc. of the 34th IEEE Int. Conf. on Communications (ICC 2022). pp. 1015–1015. IEEE-CS Press (2022)
6. Bennett, C.H.: Logical reversibility of computation. IBM Journal of Research and Development **17**(6), 525–532 (1973)
7. Boonkrong, S.: Authentication and Access Control. Apress Berkeley (2020)
8. Brookes, S.D., Hoare, C.A.R., Roscoe, A.W.: A theory of communicating sequential processes. Journal of the ACM **31**, 560–599 (1984)
9. Danos, V., Krivine, J.: Reversible communicating systems. In: Proc. of the 15th Int. Conf. on Concurrency Theory (CONCUR 2004). LNCS, vol. 3170, pp. 292–307. Springer (2004)
10. Danos, V., Krivine, J.: Transactions in RCCS. In: Proc. of the 16th Int. Conf. on Concurrency Theory (CONCUR 2005). LNCS, vol. 3653, pp. 398–412. Springer (2005)
11. De Nicola, R., Montanari, U., Vaandrager, F.: Back and forth bisimulations. In: Proc. of the 1st Int. Conf. on Concurrency Theory (CONCUR 1990). LNCS, vol. 458, pp. 152–165. Springer (1990)
12. de Vries, E., Koutavas, V., Hennessy, M.: Communicating transactions. In: Proc. of the 21th Int. Conf. on Concurrency Theory (CONCUR 2010). LNCS, vol. 6269, pp. 569–583. Springer (2010)
13. Engblom, J.: A review of reverse debugging. In: Proc. of the 4th System, Software, SoC and Silicon Debug Conf. (S4D 2012). pp. 1–6. IEEE-CS Press (2012)
14. Erkens, R., Rot, J., Luttik, B.: Up-to techniques for branching bisimilarity. In: Proc. of the 46th Int. Conf. on Current Trends in Theory and Practice of Informatics (SOFSEM 2020). LNCS, vol. 12011, pp. 285–297. Springer (2020)
15. Focardi, R., Gorrieri, R.: Classification of security properties. In: Proc. of the 1st Int. School on Foundations of Security Analysis and Design (FOSAD 2000). LNCS, vol. 2171, pp. 331–396. Springer (2001)
16. Focardi, R., Piazza, C., Rossi, S.: Proofs methods for bisimulation based information flow security. In: Proc. of the 3rd Int. Workshop on Verification, Model Checking, and Abstract Interpretation (VMCAI 2002). LNCS, vol. 2294, pp. 16–31. Springer (2002)
17. Focardi, R., Rossi, S.: Information flow security in dynamic contexts. Journal of Computer Security **14**(1), 65–110 (2006)
18. Giachino, E., Lanese, I., Mezzina, C.: Causal-consistent reversible debugging. In: Proc. of the 17th Int. Conf. on Fundamental Approaches to Software Engineering (FASE 2014). LNCS, vol. 8411, pp. 370–384. Springer (2014)
19. Giacobazzi, R., Mastroeni, I.: Abstract non-interference: A unifying framework for weakening information-flow. ACM Trans. on Privacy and Security **21**(2) (2018)

20. van Glabbeek, R.J.: A complete axiomatization for branching bisimulation congruence of finite-state behaviours. In: Proc. of the 18th Int. Symp. on Mathematical Foundations of Computer Science (MFCS 1993). LNCS, vol. 711, pp. 473–484. Springer (1996)
21. van Glabbeek, R.J., Weijland, W.P.: Branching time and abstraction in bisimulation semantics. Journal of the ACM **43**, 555–600 (1996)
22. Goguen, J.A., Meseguer, J.: Security policies and security models. In: Proc. of the 2nd IEEE Symp. on Security and Privacy (SSP 1982). pp. 11–20. IEEE-CS Press (1982)
23. Groote, J.F., Vaandrager, F.: An efficient algorithm for branching bisimulation and stuttering equivalence. In: Proc. of the 17th Int. Coll. on Automata, Languages and Programming (ICALP 1990). LNCS, vol. 443, pp. 626–638. Springer (1990)
24. Hedin, D., Sabelfeld, A.: A perspective on information-flow control. Software Safety and Security - Tools for Analysis and Verification **33**, 319–347 (2012)
25. Hillston, J., Marin, A., Piazza, C., Rossi, S., Casagrande, A., Omodeo, E.G., Proietti, M.: Persistent stochastic non-interference. Fundamenta Informaticae **181**(1), 1–35 (2021)
26. Jansen, D.N., Groote, J.F., Keiren, J.J.A., Wijs, A.: An $O(m \log n)$ algorithm for branching bisimilarity on labelled transition systems. In: Proc. of the 26th Int. Conf. on Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2020). LNCS, vol. 12079, pp. 3–20. Springer (2020)
27. Keller, R.M.: Formal verification of parallel programs. Communications of the ACM **19**, 371–384 (1976)
28. Landauer, R.: Irreversibility and heat generated in the computing process. IBM Journal of Research and Development **5**, 183–191 (1961)
29. Lanese, I., Lienhardt, M., Mezzina, C.A., Schmitt, A., Stefani, J.B.: Concurrent flexible reversibility. In: Proc. of the 22nd European Symp. on Programming (ESOP 2013). LNCS, vol. 7792, pp. 370–390. Springer (2013)
30. Lanese, I., Nishida, N., Palacios, A., Vidal, G.: CauDEr: A causal-consistent reversible debugger for Erlang. In: Proc. of the 14th Int. Symp. on Functional and Logic Programming (FLOPS 2018). LNCS, vol. 10818, pp. 247–263. Springer (2018)
31. Laursen, J.S., Ellekilde, L.P., Schultz, U.P.: Modelling reversible execution of robotic assembly. Robotica **36**, 625–654 (2018)
32. Mantel, H.: Information flow and noninterference. In: Encyclopedia of Cryptography and Security. pp. 605–607. Springer (2011)
33. Martinelli, F.: Analysis of security protocols as open systems. Theoretical Computer Science **290**(1), 1057–1106 (2003)
34. Milner, R.: Communication and Concurrency. Prentice Hall (1989)
35. Park, D.: Concurrency and automata on infinite sequences. In: Proc. of the 5th GI Conf. on Theoretical Computer Science. LNCS, vol. 104, pp. 167–183. Springer (1981)
36. Perumalla, K.S., Park, A.J.: Reverse computation for rollback-based fault tolerance in large parallel systems - Evaluating the potential gains and systems effects. Cluster Computing **17**, 303–313 (2014)
37. Phillips, I., Ulidowski, I.: Reversing algebraic process calculi. Journal of Logic and Algebraic Programming **73**, 70–96 (2007)
38. Phillips, I., Ulidowski, I., Yuen, S.: A reversible process calculus and the modelling of the ERK signalling pathway. In: Proc. of the 4th Int. Workshop on Reversible Computation (RC 2012). LNCS, vol. 7581, pp. 218–232. Springer (2013)

39. Pinna, G.M.: Reversing steps in membrane systems computations. In: Proc. of the 17th Int. Conf. on Membrane Computing (CMC 2017). LNCS, vol. 10725, pp. 245–261. Springer (2017)
40. Sangiorgi, D., Milner, R.: The problem of "weak bisimulation up to". In: Proc. of the 3rd Int. Conf. on Concurrency Theory (CONCUR 1992). LNCS, vol. 630, pp. 32–46. Springer (1992)
41. Schordan, M., Oppelstrup, T., Jefferson, D.R., Barnes Jr., P.D.: Generation of reversible C++ code for optimistic parallel discrete event simulation. New Generation Computing **36**, 257–280 (2018)
42. Siljak, H., Psara, K., Philippou, A.: Distributed antenna selection for massive MIMO using reversing Petri nets. IEEE Wireless Communication Letters **8**, 1427–1430 (2019)
43. Vassor, M., Stefani, J.B.: Checkpoint/rollback vs causally-consistent reversibility. In: Proc. of the 10th Int. Conf. on Reversible Computation (RC 2018). LNCS, vol. 11106, pp. 286–303. Springer (2018)
44. Zheng, L., Myers, A.: Dynamic security labels and noninterference (extended abstract). In: Proc. of the 2nd Int. Workshop on Formal Aspects in Security and Trust (FAST 2004), IFIP AICT, vol. 173, pp. 27–40. Springer (2005)

A Proofs of Results

Proof of Theorem 2.
The result derives from the proof of compositionality of $\approx_{\mathrm{b}}$ with respect to the operators needed to define the various information-flow security properties. Hence, in the following we have to show that $\approx_{\mathrm{b}}$ is a congruence with respect to $\_ \setminus L$, $\_/L$, and $\_ \|_L \_$.
Case 1: $\_ \setminus L$. Consider, by hypothesis, a branching bisimulation $\mathcal{B}$ such that $(P_1, P_2) \in \mathcal{B}$. Then, the relation $\mathcal{B}'$ defined as $(P' \setminus L, Q' \setminus L) \in \mathcal{B}'$ if and only if $(P', Q') \in \mathcal{B}$ is a branching bisimulation too. It is sufficient to consider two cases:

- $P' \setminus L \xrightarrow{a} P'' \setminus L$. In this case it is clear that $a \notin L$. By hypothesis if $P' \xrightarrow{a} P''$ then there exist $\bar{Q}'$ and Q'' such that $Q' \stackrel{\tau^*}{\Longrightarrow} \bar{Q}' \xrightarrow{a} Q''$, with $(P', \bar{Q}') \in \mathcal{B}$ and $(P'', Q'') \in \mathcal{B}$. Since the restriction operator does not apply to $\{\tau\}$ and $a \notin L$ it follows that $Q' \setminus L \stackrel{\tau^*}{\Longrightarrow} \bar{Q}' \setminus L \xrightarrow{a} Q'' \setminus L$, with $(P' \setminus L, \bar{Q}' \setminus L) \in \mathcal{B}'$ and $(P'' \setminus L, Q'' \setminus L) \in \mathcal{B}'$.
- $P' \setminus L \xrightarrow{\tau} P'' \setminus L$. By hypothesis, we have that $P' \xrightarrow{\tau} P''$ and either $(P'', Q') \in \mathcal{B}$, or there exist $\bar{Q}'$ and Q'' such that $Q' \stackrel{\tau^*}{\Longrightarrow} \bar{Q}' \xrightarrow{\tau} Q''$ with $(P', \bar{Q}') \in \mathcal{B}$ and $(P'', Q'') \in \mathcal{B}$. In the former case it is sufficient to note that, since the restriction operator does not apply to $\{\tau\}$, $Q' \setminus L$ is allowed to stay idle and $(P'' \setminus L, Q' \setminus L) \in \mathcal{B}'$. In the latter case we can reason similarly as in the previous point.

In both cases $\mathcal{B}'$ is a branching bisimulation. The same reasoning applies to $Q' \setminus L \xrightarrow{a} Q'' \setminus L$ and $Q' \setminus L \xrightarrow{\tau} Q'' \setminus L$.
Case 2: $\_/L$. As in the previous case, consider, by hypothesis, a branching bisimulation $\mathcal{B}$ such that $(P_1, P_2) \in \mathcal{B}$. Then, the relation $\mathcal{B}'$ defined as $(P' \,/\, L, Q' \,/\, L) \in \mathcal{B}'$ if and only if $(P', Q') \in \mathcal{B}$ is a branching bisimulation too. There is only one interesting case: $P' \,/\, L \xrightarrow{\tau} P'' \,/\, L$ with $P' \xrightarrow{a} P''$ and $a \in L$. By hypothesis there exist $\bar{Q}'$ and Q'' such that $Q' \stackrel{\tau^*}{\Longrightarrow} \bar{Q}' \xrightarrow{a} Q''$ with $(P', \bar{Q}') \in \mathcal{B}$ and $(P'', Q'') \in \mathcal{B}$. Since the hiding operator does not apply to $\{\tau\}$ it follows that $Q' \,/\, L \stackrel{\tau^*}{\Longrightarrow} \bar{Q}' \,/\, L$, with $(P' \,/\, L, \bar{Q}' \,/\, L) \in \mathcal{B}'$, and there exists a τ-transition such that $\bar{Q}' \,/\, L \xrightarrow{\tau} Q'' \,/\, L$ with $(P'' \,/\, L, Q'' \,/\, L) \in \mathcal{B}'$. Hence, the relation $\mathcal{B}'$ is a branching bisimulation. The same reasoning applies if $Q' \,/\, L \xrightarrow{\tau} Q'' \,/\, L$ with $Q' \xrightarrow{a} Q''$.
Case 3: $\_ \|_L \_$. Consider, by hypothesis, a branching bisimulation $\mathcal{B}$ such that $(P_1, P_2) \in \mathcal{B}$. Then, the relation $\mathcal{B}'$ defined as $(P_1' \|_L Q, P_2' \|_L Q) \in \mathcal{B}'$ if and only if $(P_1', P_2') \in \mathcal{B}$, for any process Q and synchronization set L, is a branching bisimulation too. There is only one interesting case: $P_1' \|_L Q \xrightarrow{a} P_1'' \|_L Q'$ with $a \in L$. By hypothesis there exist $\bar{P}_2'$ and P_2'' such that $P_2' \stackrel{\tau*}{\Longrightarrow} \bar{P}_2' \xrightarrow{a} P_2''$ with $(P_1', \bar{P}_2') \in \mathcal{B}$ and $(P_1'', P_2'') \in \mathcal{B}$. From this fact we derive $P_2' \|_L Q \stackrel{\tau*}{\Longrightarrow} \bar{P}_2' \|_L Q \xrightarrow{a} P_2'' \|_L Q'$, with $(P_1' \|_L Q, \bar{P}_2' \|_L Q) \in \mathcal{B}'$ and $(P_1'' \|_L Q', P_2'' \|_L Q') \in \mathcal{B}'$. Hence, $\mathcal{B}'$ is a branching bisimulation. The same reasoning applies if $P_2' \|_L Q \xrightarrow{a} P_2'' \|_L Q'$. ■

Proof of Theorem 3.
We divide the proof into two parts. We first prove the result for the SBrSNNI:

1. This point follows from the fact that $\approx_{\rm b}$ is a congruence with respect to the prefix operator [21] and that $a \notin \mathcal{A}_{\mathcal{H}}$. Because of these two facts, for a given process $P \in \mathbb{P}$ we have that if $P \setminus \mathcal{A}_{\mathcal{H}} \approx_{\rm b} P / \mathcal{A}_{\mathcal{H}}$, then also $a \,.\, (P \setminus \mathcal{A}_{\mathcal{H}}) \approx_b a \,.\, (P / \mathcal{A}_{\mathcal{H}})$ and $(a \,.\, P) \setminus \mathcal{A}_{\mathcal{H}} \approx_b (a \,.\, P)/\mathcal{A}_{\mathcal{H}}$ as the two operators $\setminus$ and $/$ do not make any change on the action a. Moreover, all the derivatives of P satisfy the BrSNNI property by hypothesis.
2. Consider the relation $((P_1' \parallel_L P_2') \setminus \mathcal{A}_{\mathcal{H}}, (P_1' \parallel_L P_2') / \mathcal{A}_{\mathcal{H}}) \in \mathcal{B}$ for every P_1' and P_2' reachable from P_1 and P_2 respectively. We need to prove that $\mathcal{B}$ is a branching bisimulation. The only interesting cases are:
   - $(P_1' \parallel_L P_2')/\mathcal{A}_{\mathcal{H}} \xrightarrow{\tau} (P_1'' \parallel_L P_2')/\mathcal{A}_{\mathcal{H}}$ with $P_1' \xrightarrow{h} P_1''$, $P_1' / \mathcal{A}_{\mathcal{H}} \xrightarrow{\tau} P_1''/\mathcal{A}_{\mathcal{H}}$ and $h \notin L$. By hypothesis $P_1' / \mathcal{A}_{\mathcal{H}} \approx_{\rm b} P_1' \setminus \mathcal{A}_{\mathcal{H}}$, this implies that:
     - Either $P_1' \setminus \mathcal{A}_{\mathcal{H}} \approx_{\rm b} P_1'' / \mathcal{A}_{\mathcal{H}}$, in which case it is clear that $((P_1' \parallel_L P_2') \setminus \mathcal{A}_{\mathcal{H}}, (P_1'' \parallel_L P_2') / \mathcal{A}_{\mathcal{H}}) \in \mathcal{B}$ because $\tau \notin L \cup \mathcal{A}_{\mathcal{H}}$, so it cannot be a synchronization action.
     - Or $P_1' \setminus \mathcal{A}_{\mathcal{H}} \stackrel{\tau}{\Longrightarrow} \bar{P}_1' \setminus \mathcal{A}_{\mathcal{H}} \xrightarrow{\tau} P_1''' \setminus \mathcal{A}_{\mathcal{H}}$ with $P_1'/\mathcal{A}_{\mathcal{H}} \approx_{\rm b} \bar{P}_1' \setminus \mathcal{A}_{\mathcal{H}}$ and $P_1'' / \mathcal{A}_{\mathcal{H}} \approx_b P_1''' \setminus \mathcal{A}_{\mathcal{H}}$ in which case we have that $(P_1' \parallel_L P_2') \setminus \mathcal{A}_{\mathcal{H}} \stackrel{\tau}{\Longrightarrow} (\bar{P}_1' \parallel_L P_2') \setminus \mathcal{A}_{\mathcal{H}} \xrightarrow{\tau} (P_1''' \parallel_L P_2') \setminus \mathcal{A}_{\mathcal{H}}$, which implies that $((\bar{P}_1' \parallel_L P_2') \setminus \mathcal{A}_{\mathcal{H}}, (P_1' \parallel_L P_2') / \mathcal{A}_{\mathcal{H}}) \in \mathcal{B}$ and also that $((P_1''' \parallel_L P_2') \setminus \mathcal{A}_{\mathcal{H}}, (P_1'' \parallel_L P_2') / \mathcal{A}_{\mathcal{H}}) \in \mathcal{B}$.

     In both cases it follows that $\mathcal{B}$ is a branching bisimulation. The same reasoning applies if $(P_1' \parallel_L P_2') / \mathcal{A}_{\mathcal{H}} \xrightarrow{\tau} (P_1' \parallel_L P_2'') / \mathcal{A}_{\mathcal{H}}$ with $P_2' \xrightarrow{h} P_2''$.
   - $(P_1' \parallel_L P_2') / \mathcal{A}_{\mathcal{H}} \xrightarrow{\tau} (P_1'' \parallel_L P_2'') / \mathcal{A}_{\mathcal{H}}$ with $P_1' \xrightarrow{h} P_1''$, $P_2' \xrightarrow{h} P_2''$, $P_1' / \mathcal{A}_{\mathcal{H}} \xrightarrow{\tau} P_1'' / \mathcal{A}_{\mathcal{H}}$, $P_2' / \mathcal{A}_{\mathcal{H}} \xrightarrow{\tau} P_2'' / \mathcal{A}_{\mathcal{H}}$. By hypothesis we have that $P_1' \setminus \mathcal{A}_{\mathcal{H}} \approx_{\rm b} P_1' / \mathcal{A}_{\mathcal{H}}$ and $P_2' \setminus \mathcal{A}_{\mathcal{H}} \approx_{\rm b} P_2' / \mathcal{A}_{\mathcal{H}}$, this implies that:
     - Either $P_i' \setminus \mathcal{A}_{\mathcal{H}} \approx_{\rm b} P_i'' / \mathcal{A}_{\mathcal{H}}$ with $i \in \{1, 2\}$, in which case we have that $(P_1' \parallel_L P_2') \setminus \mathcal{A}_{\mathcal{H}}$ can stay idle and $((P_1'' \parallel_L P_2'') / \mathcal{A}_{\mathcal{H}}, (P_1' \parallel_L P_2') \setminus \mathcal{A}_{\mathcal{H}}) \in \mathcal{B}$.
     - Or $P_i' \setminus \mathcal{A}_{\mathcal{H}} \stackrel{\tau}{\Longrightarrow} \bar{P}_i' \setminus \mathcal{A}_{\mathcal{H}} \xrightarrow{\tau} P_i''' \setminus \mathcal{A}_{\mathcal{H}}$ with $P_i'/\mathcal{A}_{\mathcal{H}} \approx_{\rm b} \bar{P}_i' \setminus \mathcal{A}_{\mathcal{H}}$ and $P_i'' / \mathcal{A}_{\mathcal{H}} \approx_{\rm b} P_i''' \setminus \mathcal{A}_{\mathcal{H}}$, where $i \in \{1, 2\}$, in which case there exist the processes $(\bar{P}_1' \parallel_L \bar{P}_2') \setminus \mathcal{A}_{\mathcal{H}}$ and $(P_1''' \parallel_L P_2''') \setminus \mathcal{A}_{\mathcal{H}}$ such that $(P_1' \parallel_L P_2') \setminus \mathcal{A}_{\mathcal{H}} \stackrel{\tau^*}{\Longrightarrow} (\bar{P}_1' \parallel_L \bar{P}_2') \setminus \mathcal{A}_{\mathcal{H}} \xrightarrow{\tau} (P_1''' \parallel_L P_2''') \setminus \mathcal{A}_{\mathcal{H}}$, with $((P_1' \parallel_L P_2') / \mathcal{A}_{\mathcal{H}}, (\bar{P}_1' \parallel_L \bar{P}_2') \setminus \mathcal{A}_{\mathcal{H}}) \in \mathcal{B}$ and $((P_1'' \parallel_L P_2'')/\mathcal{A}_{\mathcal{H}}, (P_1''' \parallel_L P_2''') \setminus \mathcal{A}_{\mathcal{H}}) \in \mathcal{B}$.

     In both cases $\mathcal{B}$ satisfies the conditions for branching bisimulation.
3. Since the processes we are considering are not recursive we can treat them as trees, and hence we can proceed by induction on their depth. In this case we will proceed by induction on the depth of P. We will prove that if $P \in$ SBrSNNI, then $P \setminus L \in$ SBrSNNI:
   - If the depth of P is 0 then it has no outgoing transitions and behaves as $\underline{0}$. This obviously entails the result.
   - If the depth of P is $n + 1$, where $n \in \mathbb{N}$, then take any P' of depth n such that $P \xrightarrow{a} P'$. By hypothesis $P, P' \in$ SBrSNNI and by induction hypothesis $P' \setminus L$ is SBrSNNI. Hence, we just need to prove that

$(P \setminus L) \setminus \mathcal{A}_\mathcal{H} \approx_\mathrm{b} (P \setminus L) / \mathcal{A}_\mathcal{H}$.
There is only one interesting case: $(P \setminus L) / \mathcal{A}_\mathcal{H} \xrightarrow{\tau} (P' \setminus L) / \mathcal{A}_\mathcal{H}$ with τ obtained by hiding $a \in \mathcal{A}_\mathcal{H}$. Clearly if $(P \setminus L) / \mathcal{A}_\mathcal{H}$ can perform such τ then we also have that $P / \mathcal{A}_\mathcal{H} \xrightarrow{\tau} P' / \mathcal{A}_\mathcal{H}$. Since $P \in$ SBrSNNI it follows that $P \setminus \mathcal{A}_\mathcal{H} \approx_\mathrm{b} P / \mathcal{A}_\mathcal{H}$, and by following Definition 4:

- Either $P \setminus \mathcal{A}_\mathcal{H} \approx_\mathrm{b} P' / \mathcal{A}_\mathcal{H}$, and then from the hypothesis $P' / \mathcal{A}_\mathcal{H} \approx_\mathrm{b} P' \setminus \mathcal{A}_\mathcal{H}$ we derive $P \setminus \mathcal{A}_\mathcal{H} \approx_\mathrm{b} P' \setminus \mathcal{A}_\mathcal{H}$ and $(P \setminus L) \setminus \mathcal{A}_\mathcal{H} \approx_\mathrm{b} (P' \setminus L) \setminus \mathcal{A}_\mathcal{H}$ and, by induction hypothesis, $(P' \setminus L) \setminus \mathcal{A}_\mathcal{H} \approx_\mathrm{b} (P' \setminus L) / \mathcal{A}_\mathcal{H}$. This means that in response to $(P \setminus L) / \mathcal{A}_\mathcal{H} \xrightarrow{\tau} (P' \setminus L) / \mathcal{A}_\mathcal{H}$ we have that $(P \setminus L) \setminus \mathcal{A}_\mathcal{H}$ can stay idle, thus satisfying Definition 4.
- Or there exist $\bar{P}$ and P'' such that $P \setminus \mathcal{A}_\mathcal{H} \xRightarrow{\tau*} \bar{P} \setminus \mathcal{A}_\mathcal{H} \xrightarrow{\tau} P'' \setminus \mathcal{A}_\mathcal{H}$ with $P / \mathcal{A}_\mathcal{H} \approx_\mathrm{b} \bar{P} \setminus \mathcal{A}_\mathcal{H}$ and $P' / \mathcal{A}_\mathcal{H} \approx_\mathrm{b} P'' \setminus \mathcal{A}_\mathcal{H}$. Since τ cannot be restricted by $\_ \setminus L$ we have $(P \setminus L) \setminus \mathcal{A}_\mathcal{H} \xRightarrow{\tau*} (\bar{P} \setminus L) \setminus \mathcal{A}_\mathcal{H} \xrightarrow{\tau} (P'' \setminus L) \setminus \mathcal{A}_\mathcal{H}$. Since $P / \mathcal{A}_\mathcal{H} \approx_\mathrm{b} \bar{P} \setminus \mathcal{A}_\mathcal{H}$ we have $P / \mathcal{A}_\mathcal{H} \setminus L \approx_\mathrm{b} \bar{P} \setminus \mathcal{A}_\mathcal{H} \setminus L$ and, provided that L and $\mathcal{A}_\mathcal{H}$ are disjoint, $P \setminus L / \mathcal{A}_\mathcal{H} \approx_\mathrm{b} \bar{P} \setminus L \setminus \mathcal{A}_\mathcal{H}$. Moreover, since $P' / \mathcal{A}_\mathcal{H} \approx_\mathrm{b} P'' \setminus \mathcal{A}_\mathcal{H}$ we have, by hypothesis, $P' \setminus \mathcal{A}_\mathcal{H} \approx_\mathrm{b} P'' \setminus \mathcal{A}_\mathcal{H}$, and then $P' \setminus L \setminus \mathcal{A}_\mathcal{H} \approx_\mathrm{b} P'' \setminus L \setminus \mathcal{A}_\mathcal{H}$. Since, by induction hypothesis, we have $P' \setminus L \setminus \mathcal{A}_\mathcal{H} \approx_\mathrm{b} P' \setminus L / \mathcal{A}_\mathcal{H}$, we derive $P' \setminus L / \mathcal{A}_\mathcal{H} \approx_\mathrm{b} P'' \setminus L \setminus \mathcal{A}_\mathcal{H}$, thus satisfying Definition 4.

4. The proof follows the same steps as in the previous case related to the restriction operator.

We now prove the theorem for the property SBrNDC:

1. For this point it is sufficient to observe that $a \notin \mathcal{A}_\mathcal{H}$ and that P is SBrNDC.
2. Given the relation $\mathcal{B}$ defined as $((P_1' \parallel_L P_2') \setminus \mathcal{A}_\mathcal{H}, (P_1'' \parallel_L P_2'') \setminus \mathcal{A}_\mathcal{H}) \in \mathcal{B}$ for every P_1' and P_2' reachable from P_1 and P_2, and for every P_1'' and P_2'' such that $P_1' \xrightarrow{h} P_1''$ and $P_2' \xrightarrow{h} P_2''$, we have to prove that $\mathcal{B}$ is a branching bisimulation. The only interesting case is $(P_1' \parallel_L P_2') \setminus \mathcal{A}_\mathcal{H} \xrightarrow{l} (P_1''' \parallel_L P_2''') \setminus \mathcal{A}_\mathcal{H}$ with $l \in L$, $P_1' \xrightarrow{l} P_1'''$ and $P_2' \xrightarrow{l} P_2'''$. By hypothesis $P \in$ SBrNDC, so it follows that there exist the processes $\bar{P}_1''$, P_1'''', $\bar{P}_2''$, and P_2'''' such that $P_1'' \xRightarrow{\tau*} \bar{P}_1'' \xrightarrow{l} P_1''''$ and $P_2'' \xRightarrow{\tau*} \bar{P}_2'' \xrightarrow{l} P_2''''$ with $P_1' \approx_\mathrm{b} \bar{P}_1''$, $P_2' \approx_\mathrm{b} \bar{P}_2''$, $P_1''' \approx_\mathrm{b} P_1''''$ and $P_1''' \approx_\mathrm{b} P_1''''$. Since τ cannot be in L, and hence it cannot be a synchronization action, it follows that that $(P_1'' \parallel_L P_2'') \setminus \mathcal{A}_\mathcal{H} \xRightarrow{\tau*} (\bar{P_1}'' \parallel_L \bar{P}_2'') \setminus \mathcal{A}_\mathcal{H} \xrightarrow{l} (P_1'''' \parallel_L P_2'''') \setminus \mathcal{A}_\mathcal{H}$ with $((P_1' \parallel_L P_2') \setminus \mathcal{A}_\mathcal{H}, (\bar{P_1}'' \parallel_L \bar{P}_2'') \setminus \mathcal{A}_\mathcal{H}) \in \mathcal{B}$ and $((P_1''' \parallel_L P_2''') \setminus \mathcal{A}_\mathcal{H}, (P_1'''' \parallel_L P_2'''') \setminus \mathcal{A}_\mathcal{H}) \in \mathcal{B}$, so that $\mathcal{B}$ is a branching bisimulation.
The reasoning is the same if $(P_1'' \parallel_L P_2'') \setminus \mathcal{A}_\mathcal{H}$ moves first.
3. By hypothesis $P \in$ SBrNDC, so for every P' reachable from P and for every P'' such that $P' \xrightarrow{h} P''$ we have that $P' \setminus \mathcal{A}_\mathcal{H} \approx_\mathrm{b} P'' \setminus \mathcal{A}_\mathcal{H}$. Furthermore since $\approx_\mathrm{b}$ is a congruence with respect to the restriction operator (Theorem 2) it follows that $(P' \setminus \mathcal{A}_\mathcal{H}) \setminus L \approx_\mathrm{b} (P'' \setminus \mathcal{A}_\mathcal{H}) \setminus L$. This fact holds if and only if $(P' \setminus L) \setminus \mathcal{A}_\mathcal{H} \approx_\mathrm{b} (P'' \setminus L) \setminus \mathcal{A}_\mathcal{H}$, which is the desired thesis.

4. By hypothesis $P \in \mathrm{SBrNDC}$, so for every P' reachable from P and for every P'' such that $P' \xrightarrow{h} P''$ we have that $P' \setminus \mathcal{A}_{\mathcal{H}} \approx_{\mathrm{b}} P'' \setminus \mathcal{A}_{\mathcal{H}}$. Moreover because of Theorem 4 P is also SBrSNNI, hence $P'/\mathcal{A}_{\mathcal{H}} \approx_{\mathrm{b}} P' \setminus \mathcal{A}_{\mathcal{H}}$ and $P''/\mathcal{A}_{\mathcal{H}} \approx_{\mathrm{b}} P'' \setminus \mathcal{A}_{\mathcal{H}}$. This fact in turn implies, by transitivity, $P' / \mathcal{A}_{\mathcal{H}} \approx_{\mathrm{b}} P'' / \mathcal{A}_{\mathcal{H}}$. Since $\approx_{\mathrm{b}}$ is a congruence with respect to the hiding operator (Theorem 2), we have that $(P' / \mathcal{A}_{\mathcal{H}}) / L \approx_{\mathrm{b}} (P'' / \mathcal{A}_{\mathcal{H}}) / L$, which in turn implies $(P' / L) / \mathcal{A}_{\mathcal{H}} \approx_{\mathrm{b}} (P'' / L) / \mathcal{A}_{\mathcal{H}}$. Finally, because of the compositionality of SBrSNNI with respect to the hiding operator (Theorem 3) we obtain $(P' / L) \setminus \mathcal{A}_{\mathcal{H}} \approx_{\mathrm{b}} (P'' / L) \setminus \mathcal{A}_{\mathcal{H}}$, which is the desired thesis. ■

Proof of Theorem 4.
We divide the proof into three parts:

- SBrNDC $\subset$ SBrSNNI. We need to prove that for a given $P \in \mathbb{P}$, if $P \in$ SBrNDC, it follows that for every P' reachable from P, $P' \in$ BrSNNI. Since the processes we are considering are not recursive we can treat them as trees, and hence we can proceed by induction on their depth. In this case we will proceed by induction on the depth of P:
  - If the depth of P is 0 then P has no outgoing transitions and it behaves as $\underline{0}$. This obviously entails that $P \setminus \mathcal{A}_{\mathcal{H}} \approx_b P / \mathcal{A}_{\mathcal{H}}$.
  - If the depth of P is $n+1$ with $n \in \mathbb{N}$, then take any P' of depth n such that $P \xrightarrow{a} P'$. By hypothesis, $P, P' \in$ SBrNDC and by induction hypothesis $P' \in$ SBrSNNI. Hence, we just need to prove that $P \setminus \mathcal{A}_{\mathcal{H}} \approx_{\mathrm{b}} P / \mathcal{A}_{\mathcal{H}}$. There are two cases:
    * If $a \notin \mathcal{A}_{\mathcal{H}}$ then both $P \setminus \mathcal{A}_{\mathcal{H}}$ and $P / \mathcal{A}_{\mathcal{H}}$ can execute a and reach, respectively, $P' \setminus \mathcal{A}_{\mathcal{H}}$ and $P' / \mathcal{A}_{\mathcal{H}}$, which are branching bisimilar by induction hypothesis. Thus Definition 4 is respected.
    * If $a \in \mathcal{A}_{\mathcal{H}}$ we have that $P / \mathcal{A}_{\mathcal{H}} \xrightarrow{\tau} P' / \mathcal{A}_{\mathcal{H}}$, with $P \xrightarrow{a} P'$. By induction hypothesis we have that $P' \setminus \mathcal{A}_{\mathcal{H}} \approx_{\mathrm{b}} P' / \mathcal{A}_{\mathcal{H}}$, and since $a \in \mathcal{A}_{\mathcal{H}}$ and $P \in$ SBrNDC we have $P \setminus \mathcal{A}_{\mathcal{H}} \approx_{\mathrm{b}} P' \setminus \mathcal{A}_{\mathcal{H}}$. By transitivity it follows that $P \setminus \mathcal{A}_{\mathcal{H}} \approx_{\mathrm{b}} P' / \mathcal{A}_{\mathcal{H}}$ which, combined with $P / \mathcal{A}_{\mathcal{H}} \xrightarrow{\tau} P' / \mathcal{A}_{\mathcal{H}}$, determines the condition required by Definition 4.

  The inclusion is strict because of the counterxample discussed in Section 4.
- SBrSNNI $\subset$ BrNDC. As in the previous case, we proceed by induction on the depth of P:
  - If the depth of P is 0 then it has no outgoing transitions and it behaves as $\underline{0}$. This obviously entails that $P \setminus \mathcal{A}_{\mathcal{H}} \approx_{\mathrm{b}} ((P \parallel_L Q) / L) \setminus \mathcal{A}_{\mathcal{H}}$.
  - If the depth of P is $n+1$, with $n \in \mathbb{N}$, then take any P' of depth n such that $P \xrightarrow{a} P'$. By hypothesis, $P, P' \in$ SBrSNNI and by induction hypothesis $P' \in$ BrNDC. Hence, we need to prove that $P \in$ BrNDC, i.e., the relation $\mathcal{B}$ defined as $(P \setminus \mathcal{A}_{\mathcal{H}}, ((P \parallel_L Q) / L) \setminus \mathcal{A}_{\mathcal{H}}) \in \mathcal{B}$, for every high-level process Q and synchronization set $L \in \mathcal{A}_{\mathcal{H}}$, is a branching bisimulation. We need to prove that regardless of what kind of action a is, whenever it is performed by $P \setminus \mathcal{A}_{\mathcal{H}}$ or $((P \parallel_L Q) / L) \setminus \mathcal{A}_{\mathcal{H}}$ the other process can respond appropriately respecting the conditions of Definition 4. We distinguish two cases:

* If $a \notin \mathcal{A}_\mathcal{H}$ then both processes can perform it and reach the processes $P' \setminus \mathcal{A}_\mathcal{H}$ and $((P' \parallel_L Q) / L) \setminus \mathcal{A}_\mathcal{H}$, which are branching bisimilar by induction hypothesis.
* If $a \in S$ it means that a is a high level action on which P must synchronize with Q. Since the actions in S are hidden it follows that $((P \parallel_L Q) / L) \setminus \mathcal{A}_\mathcal{H} \xrightarrow{\tau} ((P' \parallel_L Q') / L) \setminus \mathcal{A}_\mathcal{H}$. By hypothesis $P \in$ SBrSNNI so it follows that $P \setminus \mathcal{A}_\mathcal{H} \approx_\mathrm{b} P / \mathcal{A}_\mathcal{H}$ and $P' \setminus \mathcal{A}_\mathcal{H} \approx_\mathrm{b} P' / \mathcal{A}_\mathcal{H}$. The process $P / \mathcal{A}_\mathcal{H}$ can perform a τ-action and reach $P' / \mathcal{A}_\mathcal{H}$ by hiding a. By induction hypothesis $P' \setminus \mathcal{A}_\mathcal{H} \approx_\mathrm{b} ((P' \parallel_L Q) / L) \setminus \mathcal{A}_\mathcal{H}$ and by transitivity it follows that $P' / \mathcal{A}_\mathcal{H} \approx_\mathrm{b} ((P' \parallel_L Q) / L) \setminus \mathcal{A}_\mathcal{H}$. Therefore, it follows that $P \setminus \mathcal{A}_\mathcal{H}$ can either stay idle, and be branching bisimilar to $P' / \mathcal{A}_\mathcal{H}$, or there exists a process $\bar{P} \setminus \mathcal{A}_\mathcal{H}$ such that $P \setminus \mathcal{A}_\mathcal{H} \stackrel{\tau*}{\Longrightarrow} \bar{P} \setminus \mathcal{A}_\mathcal{H} \xrightarrow{\tau} P' \setminus \mathcal{A}_\mathcal{H}$ with $\bar{P} \setminus \mathcal{A}_\mathcal{H} \approx_\mathrm{b} P / \mathcal{A}_\mathcal{H} \approx_\mathrm{b} ((P \parallel_L Q) / L) \setminus \mathcal{A}_\mathcal{H}$ and $P' \setminus \mathcal{A}_\mathcal{H} \approx_\mathrm{b} P' / \mathcal{A}_\mathcal{H} \approx_\mathrm{b} ((P' \parallel_L Q) / L) \setminus \mathcal{A}_\mathcal{H}$. In both cases the condition of branching bisimulation is respected.

Since in each case, whenever $P \setminus \mathcal{A}_\mathcal{H}$ or $((P \parallel_L Q) / L) \setminus \mathcal{A}_\mathcal{H}$ perform an action the other process can respond appropriately it follows that $\mathcal{B}$ is a branching bisimulation.

The inclusion is strict because of the counterxample discussed in Section 4.

– BrNDC $\subset$ BrSNNI. For the inclusion it is sufficient to notice that if a process $P \in \mathbb{P}$ is BrNDC it means that $P \setminus \mathcal{A}_\mathcal{H} \approx_\mathrm{b} (P \parallel_L Q) / L) \setminus \mathcal{A}_\mathcal{H}$ for every high level process Q and synchronization set $L \in \ / \mathcal{A}_\mathcal{H}$. In particular, it holds also for the process Q mimicking, step by the step, the high level behavior of P, thus being able to synchronize with all the high level actions of P. Hence, the semantics of $((P \parallel_L Q) / L) \setminus \mathcal{A}_\mathcal{H}$ is isomorphic to that of $P / \mathcal{A}_\mathcal{H}$ and hence the process P satisfies the conditions for BrSNNI. The inclusion is strict because of the counterxample discussed in Section 4. ■

Proof of Theorem 6.
The general structure of the discussed processes can be found in Figure 5:

1. $Q \in$ BSNNI, $Q \notin$ BrSNNI. Let $\mathcal{B}$ be the weak bisimulation between P_1 and P_2 and $Q = P_1 + h \,.\, P_2$. To prove that $Q \in$ BSNNI it is sufficient to notice that by hypothesis, $Q \setminus \mathcal{A}_\mathcal{H} = P_1$ and $Q / \mathcal{A}_\mathcal{H} = P_1 + \tau \,.\, P_2$ then, clearly the bisimulation $\mathcal{B}' = \mathcal{B} \cup \{(Q \setminus \mathcal{A}_\mathcal{H}, Q / \mathcal{A}_\mathcal{H}), (Q / \mathcal{A}_\mathcal{H}, Q \setminus \mathcal{A}_\mathcal{H})\}$ is again a weak bisimulation. The only interesting case is the one in which $Q / \mathcal{A}_\mathcal{H}$ performs a τ-action towards P_2. According to the definition of weak bisimulation (Definition 3), $Q \setminus \mathcal{A}_\mathcal{H}$ can stay idle. Then, it remains to verify that $P_1 / \mathcal{A}_\mathcal{H} \approx P_2 \setminus \mathcal{A}_\mathcal{H}$, which is true by hypothesis. On the other hand, to check that $Q \notin$ BrSNNI it is sufficient to notice that if $Q / \mathcal{A}_\mathcal{H}$ performs a τ-action towards P_2 then $Q \setminus \mathcal{A}_\mathcal{H}$ cannot respond adequately because by hypothesis even if it stays idle according to the definition of branching bisimulation (Definition 4) then $P_1 \approx_\mathrm{b} P_2$ must be checked, which is false by hypothesis.
2. $Q \in$ BNDC, $Q \notin$ BrNDC. Since $Q \in$ SBSNNI and SBSNNI $\subset$ BNDC (Theorem 1) it follows that $Q \in$ BNDC. Similarly since $Q \notin$ BrSNNI and BrNDC $\subset$ BrSNNI (Theorem 4) it follows that $Q \notin$ BrNDC.

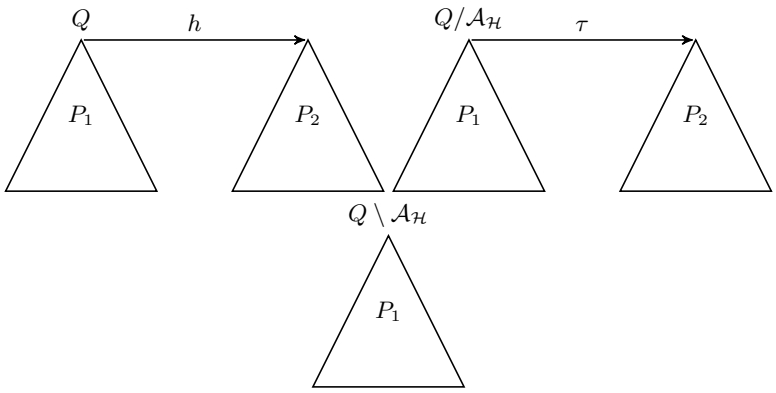

Fig. 5. General strategy to produce counterexamples for various security properties

3. $Q \in$ SBSNNI, $Q \notin$ SBrSNNI. Since $Q \notin$ BrSNNI then it follows that $Q \notin$ SBrSNNI. To prove that $Q \in$ SBSNNI it is sufficient to notice that the only high level action in Q is performed from the initial state, which is BSNNI, and hence for every reachable state Q', $Q' \setminus \mathcal{A}_{\mathcal{H}}$ is isomorphic to $Q' / \mathcal{A}_{\mathcal{H}}$.
4. $Q \in$ SBNDC, $Q \notin$ SBrNDC. To prove both points it is sufficient to notice that the only high level action in Q is performed from the initial state and it reaches the process P_2, and that $Q \setminus \mathcal{A}_{\mathcal{H}}$ corresponds to P_1. Since by hypothesis P_1 and P_2 do not contain any high level actions, to verify the two properties it is sufficient to check that $P_1 \approx P_2$ and $P_1 \approx_{\mathrm{b}} P_2$. By hypothesis, the former is true and the latter is false, hence the desired result.

An example of processes that satisfy the above theorem is given by the pair in Figure 6, $P_1 = \tau \,.\, l_1 \,.\, \underline{0} + l_2 \,.\, \underline{0}$ and $P_2 = \tau \,.\, l_1 \,.\, \underline{0} + l_1 \,.\, \underline{0} + l_2 \,.\, \underline{0}$, which are weakly bisimilar but not branching bisimilar (see Figure 1). From these processes it is possible to construct the process $Q = \tau \,.\, l_1 \,.\, \underline{0} + l_2 \,.\, \underline{0} + h \,.\, (\tau \,.\, l_1 \,.\, \underline{0} + l_1 \,.\, \underline{0} + l_2 \,.\, \underline{0})$ which is BSNNI, BNDC, SBSNNI, and SBNDC, but not BrSNNI, BrNDC, SBrSNNI, or SBrNDC. ■

Proof of Theorem 7.
Consider $\tau \,.\, x + x = \tau \,.\, x$, with $x = \tau \,.\, l_1 \,.\, \underline{0} + \tau \,.\, l_2 \,.\, \underline{0}$ and then add the subprocess $+ l_3 \,.\, \underline{0}$ to both processes in the equation, thus obtaining the equation $l_3 \,.\, \underline{0} + \tau \,.\, (\tau \,.\, l_1 \,.\, \underline{0} + \tau \,.\, l_2 \,.\, \underline{0}) + (\tau \,.\, l_1 \,.\, \underline{0} + \tau \,.\, l_2 \,.\, \underline{0}) = l_3 \,.\, \underline{0} + \tau \,.\, (\tau \,.\, l_1 \,.\, \underline{0} + \tau \,.\, l_2 \,.\, \underline{0})$, which holds for weak bisimilarity but not for branching bisimilarity. Now let us define the process P as $l_3 \,.\, \underline{0} + \tau \,.\, (\tau \,.\, l_1 \,.\, \underline{0} + \tau \,.\, l_2 \,.\, \underline{0}) + (h \,.\, l_1 \,.\, \underline{0} + h \,.\, l_2 \,.\, \underline{0})$ (see also Figure 2, up to renaming of the chosen low level names) for which it holds that $P / \mathcal{A}_H$ and $P \setminus \mathcal{A}_H$ are isomorphic to the two terms of the equation, respectively. Hence, by construction it follows that P is BSNNI but not BrSNNI. Furthermore to prove that $P \in$ SBSNNI it is sufficient to notice that since the only high level action is performed from the initial state, which is BSNNI, it follows that for

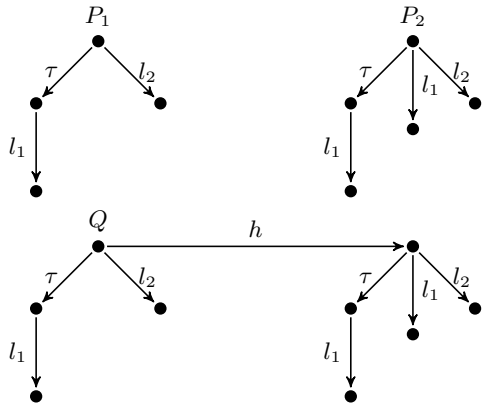

Fig. 6. Example of a pair of processes from which it is possible to construct a new process not secure for branching bisimulation-based information-flow security properties

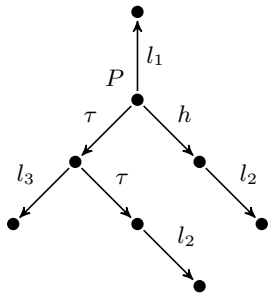

Fig. 7. Counterexample constructed from the τ-axiom $a \,.\, (\tau \,.\, x+y) = a \,.\, (\tau \,.\, x+y)+a \,.\, x$

every other reachable state P', we have that $P' \setminus \mathcal{A}_{\mathcal{H}}$ is isomorphic to $P' \,/\, \mathcal{A}_{\mathcal{H}}$. ■

Proof of Theorem 8.
Consider $a \,.\, (\tau \,.\, x + y) + a \,.\, x = a \,.\, (\tau \,.\, x + y)$, instantiate a as τ, let $x = l_2 \,.\, \underline{0}$ and $y = l_3 \,.\, \underline{0}$, and then add the subprocess $+l_1 \,.\, \underline{0}$ to both terms in the equation, thus obtaining the equation $l_1 \,.\, \underline{0} + \tau \,.\, (\tau \,.\, l_2 \,.\, \underline{0} + l_3 \,.\, \underline{0}) + \tau \,.\, l_2 \,.\, \underline{0} = l_1 \,.\, \underline{0} + \tau \,.\, (\tau \,.\, l_2 \,.\, \underline{0} + l_3 \,.\, \underline{0})$, which holds for weak bisimilarity but does not hold for branching bisimilarity. Now let us define the process P as $l_1 \,.\, \underline{0} + \tau \,.\, (\tau \,.\, l_2 \,.\, \underline{0} + l_3 \,.\, \underline{0}) + h \,.\, l_2 \,.\, \underline{0}$ (see Figure 7) for which it holds that $P \,/\, \mathcal{A}_H$ and $P \setminus \mathcal{A}_H$ are isomorphic to the two terms of the equation, respectively. Hence, by construction it follows that P is BSNNI but not BrSNNI. For SBSNNI the reasoning is the same as Theorem 7. ■