Classification of security properties in a Linda-like process algebra

Alessandro Aldini *

University of Urbino “Carlo Bo”, Istituto di Scienze e Tecnologie dell’Informazione, Urbino, Italy

Received 15 May 2004; received in revised form 15 April 2005; accepted 15 July 2005
Available online 7 July 2006

Abstract

We provide a classification of noninterference-based security properties for the formal analysis of secure information flow in concurrent and distributed systems. This is done in the setting of a process algebra modeling some Linda coordination primitives (asynchronous communication and read operation). For this purpose, we define relaxed notions of behavioural equivalence that take into account the observational power of the external observer. The resulting taxonomy is compared with analogous security definitions based on synchronous communication models, thus emphasizing the influence of the Linda coordination model upon the expressivity of the security properties, by giving a new intuition to the relative merits.

Keywords: Information flow analysis; Noninterference properties; Coordination model; Process algebra; Behavioural equivalence

1. Introduction

New networking technologies, such as mobile and portable devices, support an increasing development of concurrent and distributed applications that adapt themselves to the dynamic nature of the environment. Such new classes of applications require a careful design process. In particular, formal models and languages, which can be successfully used during the preliminary modeling phase, must deal with the several issues that affect the design and management of dynamic systems. Coordination models and languages represent a promising approach for modeling and analysing applications based on a dynamically reconfigurable communication system (see, e.g., Linda [16] and [10,9] for a survey on coordination languages). Such models and languages distinguish between the internal behaviour of the components and their interactions. Since the environment may replace or alter the entities and the interactions involved in the communication, critical security aspects come into play. In order to analyse the robustness of open systems against potentially hostile mobile components or unintentional security flaws, confidentiality issues have been deeply treated in several formal models for the design of multi-level security systems (e.g., see [14,30] for a survey). In particular, semantics-based models relying on the noninterference approach to information flow theory represent a successful approach (see, e.g., [23,15] and the references therein). In this setting the system behaviour is analysed in order to capture possible interferences among components at different levels of security. More precisely,
the noninterference idea [17] is expressed by the following intuition. The lack of information flow from high- to low-security levels is guaranteed if the interactions observed at the low level are invariant under changes in the high level.

As a step towards bridging the gap between the security and the coordination communities, in this paper we relate the noninterference approach to secure information flow analysis and the coordination model for communication. In [1] we preliminarily started such an integration between noninterference analysis and coordination model by rephrasing two noninterference properties [13] in the setting of a process algebra including some Linda coordination primitives, asynchronous communication and read operation. In this paper, we extend the formal framework of [1] to study more deeply the influence of the communication structure upon the security issues. For this purpose, we formalize several security properties and we compare the resulting taxonomy with a corresponding classification of security definitions obtained in a classical framework based on synchronous communication, thus emphasizing the merits of every security definition within each particular communication structure.

With respect to a synchronous communication model, in which the system must agree with the external environment on the input/output events to be performed if we want that they really happen, in an asynchronous model the output operations are autonomously performed by the system. In essence, the system emits an output message and afterwards proceeds with its execution while the message travels in the ether until it reaches its destination. Instead, like in the synchronous communication model, an input event takes place the instant the environment activates that input. In the security community, a symmetric management of input and output events is motivated by the need of capturing each possible interference between high-level components and low-level ones (see, e.g., [18]). On the other hand, in real cases there may be several output communications that cannot be refused, such as the appearance of information on a screen, and that allow the system to proceed just after performing them (as also emphasized, e.g., in [29]). Since the occurrence of such events is entirely uninfluenced by the (potentially hostile) environment, it is clear that the expressive power of security properties naturally depends on the communication model we adopt.

The formal framework we employ to study how the communication model affects the security properties is borrowed from [8] and [13]. More precisely, we use an adapted version of LINPA [8] (LINda in Process Algebra) for security analysis, which can be considered a variant of SPA [13] (Security Process Algebra) with asynchronous communication and read operation. Both calculi are inspired by the nondeterministic process algebra CCS [25]. In this integrated framework, we rephrase some security properties taken from [13,15,34].

In the model we adopt, asynchronous communication is realized by means of a so-called tuplespace, which represents a shared box to which messages (also called tuples) can be explicitly sent through output operations and from which the same messages can be read (removed) via reading (input) operations. From the viewpoint of an external observer, the focus moves from the interaction between the system and the environment to the content of the tuplespace. In other words, since communications take place through access to the tuplespace, what an observer can see is just the tuplespace. In a security context, external observers that access the tuplespace are classified depending on their access clearance. In particular, as usual in security models, we take into consideration a two-level security architecture. Therefore, we divide the tuplespace into a high-level part containing confidential tuples (which are handled by high-level users only) and a low-level part containing public messages (which are generated and can be consumed by low-level users only). An illegal information flow occurs whenever alterations of the public portion of the tuplespace are caused by the operations of the high-level user conducted on the confidential portion of the tuplespace. To compare the effect (visible at the low level) of the interactions between the system and the tuplespace under different high-level strategies, we need a notion of behavioural equivalence. The security properties we investigate are essentially based on two possible assumptions concerning the semantics of a system: a trace-based semantics and a bisimilarity-based semantics. In both cases, we define adequate notions of equivalences that are appropriate to express the observational power of the external observer in the presence of an explicit communication medium modeled by the tuplespace. In particular, we show that the taxonomy of the security properties we obtain, as well as compositionality results and expressive power, strictly depend on the equivalence notion we adopt.

The rest of the paper is organized as follows. In Section 2 we describe the process algebra, called security asynchronous language (SAL, for short), its syntax and semantics, and the behavioural equivalences that represent the formal framework for the definition of the security properties. In particular, we define a trace-based equivalence and three bisimilarity-based equivalences. Then, in Section 3 we introduce several noninterference properties along the same line of [13]. Through some examples, we study their expressive power, by emphasizing similarities and

differences with respect to the synchronous setting of [13]. Finally, in Section 4 we report on related work and some conclusions.

2. Security asynchronous language

The constituent elements of any synchronous process algebra are the actions, which represent interactions between the system and the environment. In a Linda-like asynchronous setting, the basic elements are the messages, which can be put in (removed from) the tuplespace. Formally, we denote by \( M \) the system and the environment. In a Linda-like asynchronous setting, the basic elements are the messages, which can be put in (removed from) the tuplespace. Formally, we denote by \( M \), ranged over by \( a, b, \ldots \), the set of message names. As usual in security models, we distinguish between high-level message names, denoted by set \( M_H \), and low-level message names, denoted by set \( M_L \). \( M_H \) and \( M_L \) are two disjoint sets that form a covering of \( M \).

The security asynchronous language (SAL) is a variant of LINPA [8], which in turn is an asynchronous version of CCS [25]. In particular, SAL is equipped with the input, output, and read operations, and, with respect to LINPA, is enriched with the hiding operator, which is needed for the definition of security properties. Indeed, abstraction is used to specify the observational power of the external observer, depending on the security level of such an observer. The set of process terms is generated by the grammar:

\[
C ::= \emptyset | \mu. C | C|C | C + C | C\langle a \rangle | C/a | Z
\]

where \( \emptyset \) is the empty process (we usually omit it when it is clear from the context), \( \langle a \rangle \) and \( a \) are the standard CCS parallel composition and alternative choice operators, \( \langle a \rangle \) is the restriction operator, which prevents the execution of actions of type \( a \in M \). \( \langle a \rangle \) is the hiding operator, which turns all the actions of type \( a \in M \) into internal, invisible actions. Moreover, for each constant \( Z \) we have a corresponding definition \( Z \equiv C \), where \( C \) is guarded on constants [25]. The possible prefixes \( \mu \) are:

\[
\mu ::= \tau | in(a) | out(a) | rd(a)
\]

where \( \tau \) is the internal, unobservable action, and \( in(a) \) and \( out(a) \) express the usual input and output primitives of Linda, respectively. More precisely, \( out(a) \) produces, in one internal step, a new tuple \( \langle a \rangle \) (containing message \( a \)) that is put in the tuplespace. An output operation is non-blocking as it allows the system to proceed just after performing the rendering of the tuple. On the other hand, \( in(a) \) removes, if present, the tuple \( \langle a \rangle \) from the tuplespace. If the tuple to be removed is not present in the tuplespace, then the input operation is blocked. Similarly as the input operation, \( rd(a) \) denotes the blocking reading of any message \( a \) from the tuplespace. With respect to the input operation, \( rd(a) \) does not remove a tuple \( \langle a \rangle \) from the tuplespace. Tuples are not considered in the syntax of processes. Instead, they are described as states, which are defined as terms generated by the syntax:

\[
P ::= \langle a \rangle | C | P | P/a | P/a.
\]

In practice, states model the parallel composition of tuples that are present in the tuplespace and processes that handle the tuplespace. We call \( A \), ranged over by \( P, Q, \ldots \), the set of agents generated by the grammar above.

The operational semantics of SAL is defined through the structural congruence \( \equiv \), which is defined as the smallest congruence that satisfies the axioms of Table 1. In particular, in axioms (x) and (xi) function \( fn(P) \) denotes the set of free names of \( P \) and is defined as follows:

\[
fn(\emptyset) = \emptyset
\]

\[
fn(\langle a \rangle) = \{a\}
\]

\[
fn(P | Q) = fn(P + Q) = fn(P) \cup fn(Q)
\]

\[
fn(P \langle a \rangle) = fn(P/a) = fn(P) \setminus \{a\}
\]

\[
fn(in(a), P) = fn(out(a), P) = fn(rd(a), P) = \{a\} \cup fn(P)
\]

\[
fn(Z) = fn(P) \text{ if } Z \equiv P.
\]

With abuse of notation, in axiom (xii), which is \( \alpha \)-conversion, we use \( P[b/a] \) to denote the term obtained by renaming all the free occurrences of the name \( a \) in \( P \) with the fresh name \( b \).

Then, we define the operational semantics of SAL as the labelled transition system \( (A, Act, \rightarrow) \), where states are agents of SAL, \( Act \) is the set of transition labels, and \( \rightarrow \subseteq A \times Act \times A \) is defined as the least transition relation satisfying the axioms and the rules of Table 2. Set \( Act \) is ranged over by \( \pi, \pi', \ldots \) and contains label \( \tau \), denoting an
Table 1
Structural congruence for SAL

| (i) | $P | \emptyset \equiv P$ |
| (ii) | $P | Q \equiv Q | P$ |
| (iii) | $(P | Q) | R \equiv P | (Q | R)$ |
| (iv) | $P + Q \equiv P$ |
| (v) | $P + P \equiv P$ |
| (vi) | $P + Q \equiv Q + P$ |
| (vii) | $(P + Q) + R \equiv P + (Q + R)$ |
| (viii) | $\text{Op}_a \equiv \emptyset$ |
| (ix) | $(\text{Op}_a) \text{Op}_b \equiv (\text{Op}_b) \text{Op}_a$ |
| (x) | $(P + Q) \text{Op}_a \equiv P + (Q \text{Op}_a)$ |
| (xi) | $(P | Q) \text{Op}_a \equiv P | (Q \text{Op}_a)$ |
| (xii) | $\text{Op}_a \equiv P | b/a | \text{Op}_b$ |
| (xiii) | $\emptyset \equiv P$ |

Table 2
Operational semantics of SAL

| (prefix) | $\tau. P \xrightarrow{\tau} P$ |
| (in) | $\text{in}(a). P \xrightarrow{\text{in}(a)} P$ |
| (out) | $\text{out}(a). P \xrightarrow{\text{out}(a)} \emptyset$ |
| (rd) | $\text{rd}(a). P \xrightarrow{\text{rd}(a)} P$ |
| (tuple) | $\langle a \rangle \xrightarrow{\text{op}_a} \emptyset$ |
| (sum) | $P + Q \xrightarrow{P + Q} P'$ |
| (par) | $P | Q \xrightarrow{P | Q} P' | Q'$ |
| (res) | $P \xrightarrow{\pi} P' \quad \pi \notin \{a, \bar{a}, g\}$ |
| (hid) | $P \xrightarrow{\pi} P' \quad \pi \notin \{a, \bar{a}, g\}$ |
| (congr) | $P \equiv Q \quad Q \xrightarrow{Q' P'} P' \equiv Q'$ |

internal action, and the set of labels denoting visible actions. In particular, for each $a \in \mathcal{M}$, $\text{Act}$ includes label $\bar{a}$, expressing the output of a tuple $\langle a \rangle$ to the environment, label $a$, expressing the consumption of a tuple $\langle a \rangle$ from the tuplespace, and label $\bar{a}$, expressing the reading of a tuple $\langle a \rangle$ from the tuplespace. We denote with $O$ the set of labels expressing output operations and with $I$ the set of labels denoting input/read operations, so that $\text{Act} = \{\tau\} \cup O \cup I$. Moreover, we assume $\mathcal{L} = O \cup I$, ranged over by $\alpha, \alpha', \ldots$.

Since the set of message names is partitioned into high-level names and low-level ones, we also partition $\mathcal{L}$ into high-level labels and low-level ones, denoted by sets $\text{Act}_H$ and $\text{Act}_L$, respectively, such that $\alpha \in \{a, \bar{a}, g\}$ belongs to $\text{Act}_H$ ($\text{Act}_L$) iff $a \in \mathcal{M}_H$ ($a \in \mathcal{M}_L$). Finally, we call $\mathcal{L}(P)$ the set of the transition labels that syntactically occur in the operational semantics of $P$, $\mathcal{A}_H = \{P | A | \mathcal{L}(P) \subseteq \text{Act}_H \cup \{\tau\}\}$ the set of high-level agents of $A$, and $\mathcal{A}_H^\emptyset = \{P | A | \mathcal{L}(P) \subseteq \{\text{Act}_H \cap O\} \cup \{\tau\}\}$ the set of high-level agents that do not execute input/read operations.
In the security framework, we sometimes employ static operators that limit their scope to certain transition labels only. For instance, in the case of the restriction operator, we define the operators $P \backslash I a$ and $P \backslash O a$, as specified by the following semantics:

\[
\begin{align*}
&\text{if } \pi \not\in \{a, A\} \text{ then } P \overset{\pi}{\to} P' \Rightarrow P' \backslash I a \\
&\text{if } \pi \not\in \{a, A\} \text{ then } P \overset{\pi}{\to} P' \Rightarrow P' \backslash O a
\end{align*}
\]

which restrict input and read operations only and output operations only, respectively. Similarly, we also define the operators $P / O a$ and $P / I a$:

\[
\begin{align*}
&\text{if } \pi = A \text{ then } P \overset{\pi}{\to} P' \Rightarrow P / O a \\
&\text{if } \pi \neq A \text{ then } P \overset{\pi}{\to} P' \Rightarrow P / I a
\end{align*}
\]

which hide $A$ labelled transitions only, and $a, A$ labelled transitions only, respectively.

2.1. Behavioural equivalence

The noninterference properties are necessarily stated using behavioural equivalences. For this purpose, in this section we define a theory of behavioural equivalence for SAL agents. As we will see, a notion of indistinguishability between agents must focus on the observable interactions between the environment and the system at hand. On the one hand, in the setting of a Linda-like asynchronous communication model, an external observer can see and manipulate the tuplespace, while he has no means for inferring whether or not the system is executing input (read) operations.

Based on the same intuition, in the literature several notions of equivalence for asynchronous calculi are defined which are mainly based on the bisimilarity theory (see, e.g., [21, 2, 8, 24] and the references therein). On the other hand, the expressive power of the external observer is also limited by the fact that the internal activities of the system cannot be observed, as they do not represent interactions with the environment to which the observer belongs. Hence, we need a notion of behavioural equivalence that abstracts away from $\tau$ actions [13, 15]. For instance, based on such a notion of observation, in [4] a trace-based characterization for an operational definition of the may testing preorder is defined in the setting of the asynchronous versions of CCS and $\pi$-calculus.

Based on these considerations, we now introduce adequate equivalences for SAL which join the requirements expressed above and can be used to express noninterference-based security properties.

**Definition 1.** Expression $P \overset{\alpha}{\Rightarrow} P'$ stands for $P(\overset{\tau}{\to})^* P_1 \overset{\alpha}{\Rightarrow} P_2 (\overset{\tau}{\to})^* P'$, where $(\overset{\tau}{\to})^*$ denotes a possibly empty sequence of $\tau$ labelled transitions. Denoted with $\gamma = \alpha_1 \ldots \alpha_n \in L^*$ a sequence of visible actions, the expression $P \overset{\gamma}{\Rightarrow} P'$ stands for $P \overset{[0]}{\Rightarrow} P_1 \overset{\alpha_1}{\Rightarrow} \ldots \overset{\alpha_{n-1}}{\Rightarrow} P_n \overset{\alpha_n}{\Rightarrow} P'$. If $\gamma$ is the empty sequence $\langle \rangle$, then $P \overset{0}{\Rightarrow} P'$ stands for $P (\overset{\tau}{\to})^* P'$.

We say that $P'$ is reachable from $P$, denoted $P \Rightarrow P'$, if $\exists \gamma \in L^* : P \overset{\gamma}{\Rightarrow} P'$.

With abuse of notation, we say that $\alpha \in \gamma$ if $\alpha$ occurs in the sequence $\gamma \in L^*$.

**Definition 2.** Given a sequence $\gamma \in L^*$, we denote with $S_\gamma$ the set of sequences obtained from $\gamma$ by replacing each $\bar{a} \in \gamma$ with $(\bar{a})^*$.

Note that $\gamma \in S_\gamma$ and $S_{\gamma} = \{\gamma\}$ if $\gamma$ does not include read operations. We now are ready to define a trace-based equivalence for SAL agents.

**Definition 3.** Given $P \in A$, the set $T(P)$ of traces associated with $P$ is defined as $T(P) = \{\gamma \in L^* \mid \exists P' : P \overset{\gamma}{\Rightarrow} P'\}$. Given $P, Q \in A$, $P$ and $Q$ are rd-trace equivalent, denoted $P \approx_{rd} Q$, if and only if for each $\gamma \in T(P)$ (resp. $T(Q)$) there exists $\delta \in T(Q)$ (resp. $T(P)$) such that $\delta \in S(\gamma)$.
A labelled transition can be simulated by a mixed (possibly empty) sequence of \( \tau \) and \( a \) labelled transitions. This is because a read operation does not alter the content of the tuplespace and, therefore, cannot be revealed by an external observer [8].

The intuition that the arrival ordering of messages is not under the control of the processes is expressed by the fact that agents \( \text{out}(a).\text{out}(b).P \) and \( \text{out}(b).\text{out}(a).P \) are equivalent. On the other hand, with respect to the trace-based characterization of [4], we do not have that \( \text{in}(a).\text{in}(b).P \) and \( \text{in}(b).\text{in}(a).P \) are equivalent. Such an equivalence would lead to ignore some information flow in a security setting. Indeed, although processes are insensitive to the arrival ordering of messages, the inputs they can execute depend on the environment behaviour. We also point out that the \( rd \)-trace equivalence extends the standard trace equivalence. Since the external observer is not allowed to observe internal activities and read operations, \( \emptyset \) is equivalent to \( rd(a).\emptyset \) as well as \( \emptyset \) is equivalent to \( \tau.\emptyset \).

**Example 4.** Agents \( rd(a).\emptyset \) and \( \emptyset \) are \( rd \)-trace equivalent. Obviously, for both agents \( \emptyset \) is in the set of possible traces. Moreover, the unique non-empty trace that can be performed by \( rd(a).\emptyset \) is \( \gamma = a \), from which we have \( S(\gamma) = (a)^* \) and \( \emptyset \in S(\gamma) \).

From the example above, we immediately derive \( rd(a).\emptyset \approx_{rd} \tau.\emptyset \) and \( rd(a).\emptyset \approx_{rd} rd(b).\emptyset \) that means both agents do not produce any visible event which allows the external observer to distinguish between them.

**Example 5.** Agents \( rd(a).\text{in}(e) \) and \( rd(b).\text{in}(e) \) are not \( rd \)-trace equivalent. For example, given the trace of the left-hand agent \( \gamma = ae \), there does not exist a trace of the right-hand agent of the form \( (a)^*e \). The motivation for distinguishing between such agents is that in the first case removing a tuple \( \langle e \rangle \) reveals to the external observer that a tuple \( \langle a \rangle \) is in the tuplespace, while in the second case the same input operation reveals that a tuple \( \langle b \rangle \) has been read. For a similar reason, we have that agents \( rd(a).\text{in}(e) \) and \( \tau.\text{in}(e) \) are not \( rd \)-trace equivalent.

As standard in a trace-based scenario, we have the following compositionality results.

**Proposition 6.** If \( P_1 \approx_{T_{rd}} P_2, Q_1 \approx_{T_{rd}} Q_2, \) and \( L \subseteq M \) then:

(i) \( P_1 \mid Q_1 \approx_{T_{rd}} P_2 \mid Q_2 \)

(ii) \( P_1 \parallel L \approx_{T_{rd}} P_2 \parallel L \)

**Proof.** It is a trivial application of that of [25]. \( \square \)

To deal with more kinds of security flaws – trace semantics is not adequate to reveal, e.g., deadlock – we now introduce a finer notion of equivalence based on the weak bisimulation semantics. In the following, we assume \( P \overset{\tau}{\Rightarrow} P' \) to stand for \( P(\tau)^*P' \). The weak bisimulation equivalence for SAL agents follows the same intuition behind the \( rd \)-trace equivalence.

**Definition 7.** An equivalence relation \( R \) on \( A \) is a weak \( rd \)-bisimulation if \( (P, Q) \in R \) implies:

- whenever \( P \overset{\tau}{\Rightarrow} P' \), with \( \tau \neq a \), then \( \exists Q' \) such that \( Q \overset{\tau}{\Rightarrow} Q' \) and \( (P', Q') \in R \);
- whenever \( P \overset{a}{\Rightarrow} P' \) then \( \exists Q' \) such that \( Q((\overset{a}{\Rightarrow})^*Q')^*Q' \) and \( (P', Q') \in R \).

Two agents \( P \) and \( Q \) are weakly \( rd \)-bisimilar, written \( P \approx_{B_{rd}} Q \), if there exists a weak \( rd \)-bisimulation \( R \) such that \( (P, Q) \in R \).

We point out that all the pairs of (non-)\( rd \)-trace equivalent agents shown in the previous examples are (non-)weak \( rd \)-bisimulation equivalent.

**Lemma 8.** \( P \approx_{B_{rd}} Q \) implies \( P \approx_{T_{rd}} Q \).

**Proof.** Assume \( P \) and \( Q \) weakly \( rd \)-bisimilar. Then, \( Q \) can simulate the behaviour of \( P \), i.e. each visible move of the left-hand agent is equated by a corresponding weak transition of the right-hand agent, each \( \tau \) move of the left-hand agent is equated by a possibly empty sequence of \( \tau \) moves of the right-hand agent, and each action \( rd(a) \) of the left-hand agent is equated by a possibly empty sequence of \( \tau \) and \( rd(a) \) actions of the right-hand agent (the agents we reach in such a way from \( P \) and \( Q \) preserve the same property). That means, for every trace \( \gamma \) of \( P \) there exists a trace \( \delta \) of \( Q \) with \( \delta \in S(\gamma) \). We can argue similarly by exchanging the roles of \( P \) and \( Q \). \( \square \)
**Proposition 9.** \(\approx_{B_{rd}}\) is preserved by parallel composition and restriction.

**Proof.** By following the same arguments applied in [25], we shall only prove the result for parallel composition. We have to show that

\[
\mathcal{R} = \{(P|R, Q|R). \ P, Q, R \in A \land P \approx_{B_{rd}} Q\}
\]

is a weak \(rd\)-bisimulation:

- If \(P|R \xrightarrow{\alpha} P'|R\), with \(\alpha \neq \bar{\alpha}\), then \(Q|\bar{R} \xrightarrow{\bar{\alpha}} Q'|R\), and \((P'|R, Q'|R) \in \mathcal{R}\).
- If \(P|R \xrightarrow{\bar{\alpha}} P'|R\), with \(P \xrightarrow{\alpha} P'\), then \(Q|\bar{R} \xrightarrow{\bar{\alpha}} Q'|R\), and \((P'|R, Q'|R) \in \mathcal{R}\).
- If \(P|R \xrightarrow{\bar{\tau}} P'|R\), with \(P \xrightarrow{\alpha} P'\) and \(R \xrightarrow{\bar{a}} R'\), then \(Q((\xrightarrow{\bar{\tau}})^*(\xrightarrow{\bar{a}})^*)^* Q'\). Hence \(Q|R \xrightarrow{\bar{\tau}} Q'|R\) and \((P'|R, Q'|R) \in \mathcal{R}\).
- If \(P|R \xrightarrow{\bar{\tau}} P'|R\), then \(Q|\bar{R} ((\xrightarrow{\bar{\tau}})^*(\xrightarrow{\bar{a}})^*)^* Q'|R\), and \((P'|R, Q'|R) \in \mathcal{R}\).
- If \(P|R \xrightarrow{\tau} P'|R'\), with \(P \xrightarrow{\bar{\tau}} P' \land R \xrightarrow{\bar{a}} R'\) or \(P \xrightarrow{\bar{\tau}} P' \land R \xrightarrow{\bar{a}} R'' \land R = R'\) or \(P \xrightarrow{\bar{\tau}} P' \land R \xrightarrow{\bar{a}} R'' \land R = R'\) or \(P \xrightarrow{\bar{\tau}} P' \land R \xrightarrow{\bar{a}} R'' \land R = R'\), then \(Q|R \xrightarrow{\tau} Q'|R'\), and \((P'|R, Q'|R') \in \mathcal{R}\).

By a symmetric argument, \(\mathcal{R}\) is a weak \(rd\)-bisimulation. \(\Box\)

The intuition behind \(\approx_{B_{rd}}\) and \(\approx_{T_{rd}}\) is not exactly the same as that reported in [8], where a strong \(rd\)-bisimulation is defined in such a way that a \(\tau\)-labelled transition cannot be simulated by a \(\bar{\alpha}\) labelled transition. In essence, \(rd(\alpha).\bar{0}\) and \(\tau.\bar{0}\) are not strong \(rd\)-bisimulation equivalent. This is because the behaviour of \(rd(\alpha).\bar{0}\) is more restrictive than that of \(\tau.\bar{0}\), since the execution of \(rd(\alpha)\) implies that a tuple \(\langle a \rangle\) is present. Both \(\approx_{B_{rd}}\) and \(\approx_{T_{rd}}\) partially lose such an intuition. In our weak setting, we have that \(rd(\alpha).\bar{0}\) and \(\tau.\bar{0}\) are equivalent as we abstract away from both internal and read actions. However, we also have that, e.g., \(rd(\alpha).\bar{in}(e)\) and \(\tau.in(\bar{e})\) are not equivalent as the execution of \(in(e)\) reveals the previous history. Now we show how to define a weak equivalence that preserves the same intuition as that of [8]. This is done by resorting to a simple variant of the observation congruence [25].

**Definition 10.** Given \(P, Q \in A\), we say that \(P\) and \(Q\) are \(rd\)-equal (observation \(rd\)-congruent), written \(P =_{rd} Q\), if and only if

- whenever \(P \xrightarrow{\alpha} P'\), with \(\alpha \neq \bar{\alpha}\), then \(\exists Q'\) such that \(Q \xrightarrow{\bar{\alpha}} Q'\) and \(P' \approx_{B_{rd}} Q'\) (and symmetrically);
- whenever \(P \xrightarrow{\tau} P'\) then \(\exists Q'\) such that \(Q \xrightarrow{\bar{\tau}} Q'\) and \(P' \approx_{B_{rd}} Q'\) (and symmetrically);
- whenever \(P \xrightarrow{\bar{\alpha}} P'\) then \(\exists Q'\) such that \(Q((\xrightarrow{\bar{\tau}})^*(\xrightarrow{\bar{a}})^*)^* \pi ((\xrightarrow{\bar{\tau}})^*(\xrightarrow{\bar{a}})^*)^* Q'\), with \(\pi = \tau\) or \(\pi = \bar{\alpha}\), and \(P' \approx_{B_{rd}} Q'\) (and symmetrically).

In essence, the observation \(rd\)-congruence expresses almost the same condition as that of the weak \(rd\)-bisimulation. The only difference is that in the case of two \(rd\)-equal agents \(P\) and \(Q\), the first move of \(P\) (resp. \(Q\)) must be equated by a transition of \(Q\) (resp. \(P\)).

**Example 11.** The agents \(rd(\alpha).\bar{0}\) and \(\bar{0}\), as well as the agents \(rd(\alpha).\bar{0}\) and \(rd(b).\bar{0}\), are not \(rd\)-equal. In particular, by following the same intuition explained in [8], \(rd(\alpha).\bar{0}\) and \(\bar{0}\) are not \(rd\)-equal, while \(rd(\alpha).\bar{P} + \tau.P =_{rd} \tau.P\).

Obviously, observation \(rd\)-congruence is an equivalence relation [25] and, as stated by the following results, implies weak \(rd\)-bisimilarity and is preserved by \(\setminus, |, +\).

**Lemma 12.** \(P =_{rd} Q\) implies \(P \approx_{B_{rd}} Q\).

**Proof.** Trivial. \(\Box\)

**Proposition 13.** \(=_{rd}\) is preserved by parallel composition, restriction and summation.

**Proof.** The proof is standard and can be derived by following the same arguments applied in [25] and in the proof of Proposition 9. \(\Box\)

Since observation \(rd\)-congruence is not a bisimulation, we can employ another behavioural equivalence that is both a bisimulation and a congruence, like, e.g., the progressing bisimulation equivalence of [26]. A notion of progressing \(rd\)-bisimulation is obtained by extending the definition of observation \(rd\)-congruence.
Definition 14. An equivalence relation $\mathcal{R}$ on $\mathcal{A}$ is a progressing $rd$-bisimulation if $(P, Q) \in \mathcal{R}$ implies:

- whenever $P \overset{a}{\rightarrow} P'$, with $\pi \neq a$, then $\exists Q'$ such that $Q \overset{a}{\Rightarrow} Q'$ and $(P', Q') \in \mathcal{R}$;
- whenever $P \overset{\tau}{\rightarrow} P'$ then $\exists Q'$ such that $Q \overset{\tau}{\Rightarrow} Q'$ and $(P', Q') \in \mathcal{R}$;
- whenever $P \overset{\alpha}{\Rightarrow} P'$ then $\exists Q'$ such that $Q((\overset{\tau}{\rightarrow})^*(\overset{a}{\Rightarrow})^*\pi) \overset{\tau}{\Rightarrow} (\overset{\tau}{\rightarrow})^*(\overset{a}{\Rightarrow})^*Q'$, with $\pi = \tau$ or $\pi = a$, and $(P', Q') \in \mathcal{R}$.

Two agents $P$ and $Q$ are progressing $rd$-bisimilar, written $P \approx_{rd}^p Q$, if there exists a progressing $rd$-bisimulation $\mathcal{R}$ such that $(P, Q) \in \mathcal{R}$.

Example 15. On the one hand, we have $rd(a).P + \tau.P \approx_{rd}^p \tau.P$. On the other hand, $\alpha.\tau(a)_0$ and $\alpha_0$ are not progressing $rd$-bisimilar, while they turn out to be $rd$-equal.

Lemma 16. $P \approx_{rd}^p Q$ implies $P =_{nd} Q$.

Proof. Trivial. $\square$

Proposition 17. $\approx_{rd}^p$ is preserved by parallel composition, restriction, and summation.

Proof. The proof is standard and can be derived by following the same arguments applied in [26]. $\square$

We conclude the presentation of behavioural equivalences by observing what follows. Relations like failure/testing equivalences are not considered in our setting since they turn out to be sensitive to divergent behaviours (cycles of $\tau$ actions), so that they would not be interesting to define the noninterference-based security properties (as shown, e.g., in [15]).

3. Security properties

In this section we provide a taxonomy of security properties based on the equivalence relations introduced in the previous section. In essence, the several definitions we investigate are taken from the classification of security properties described in [13,15] for a synchronous communication model, which are in turn inspired by the noninterference theory introduced in [17]. In the setting of a Linda-like asynchronous communication model, the goal of such security definitions is to verify whether a low-level user can infer the behaviour of the high-level user by observing the content of the tuplespace at run time.

For the sake of readability, given $L = \{a_1, \ldots, a_n\} \subseteq \mathcal{A}$, we use the abbreviation $P \op L$ to stand for $P \op a_1 \ldots \op a_n$, with $\op \in \{\\setminus, /\}$.

3.1. Trace-based properties

We start the presentation by defining the first natural formalization of the standard noninterference idea: the nondeterministic noninterference property. The intuition behind this property is that there is no observable distinction, from the low-level user standpoint, between the behaviour of the system when high-level inputs are accepted and the behaviour of the same system when none of the high-level inputs are executed. In the setting of SAL, we change such an intuition by observing that the low-level view of the tuplespace is not to be affected by high-level input/read operations performed on the tuplespace. In a trace-based scenario this intuition is expressed by requiring that for each trace including high-level input/read operations there exists another trace, containing no high-level input/read operations, such that the two traces are the same from the viewpoint of a low-level observer. We formalize the related property, termed Nondeterministic Noninterference (NNI, for short) [13], by defining the $rd$-trace NNI ($rd$-NNI). For this purpose, we introduce the following notation: Function $\low : \mathcal{L}^* \rightarrow \Act_H^*$ takes a trace $\gamma$ and removes all the high-level actions from it (i.e., $\low(\gamma)$ returns the low-level sub-sequence of $\gamma$), and function $\highir : \mathcal{L}^* \rightarrow (\Act_H \cap I)^*$ extracts from a trace the sub-sequence formed by all of its high-level input and read operations.

Definition 18 ($rd$-NNI). $P \in rd$-NNI $\iff \forall \gamma \in T(P), \exists \delta \in T(P)$ such that $\low(\delta) \in S(\low(\gamma)) \land \highir(\delta) = \emptyset$. 

In synchronous communication models, NNI is not adequate to reveal interferences due to high-level outputs. In particular, the emission of a high-level output occurs if and only if a corresponding input offered by the external environment is available. Such an input could never arrive, thus causing a system deadlock. Therefore, it is necessary to strengthen the noninterference property by symmetrically handling input and output actions. Such an intuition leads to the definition of Strong Nondeterministic Noninterference (SNNI, for short), which states that both high-level inputs and high-level outputs must not affect the low-level behaviour of the system. In the setting of SAL we introduce the rd-trace SNNI (rd-SNNI), which intuitively states that a low-level user should not be able to guess the content of the high-level portion of the tuplespace by interacting with the low-level part of the tuplespace.

**Definition 19** (rd-SNNI). \( P \in \text{rd-SNNI} \iff \forall \gamma \in T(P), \exists \delta \in T(P) \text{ such that } \delta \in S(\text{low}(\gamma)). \)

In an asynchronous setting a strong version of the noninterference property is not needed, as we now formally show in the context of SAL. In fact, since output is asynchronous, the high-level user cannot prevent the system from executing an out(a) operation or, in other words, there is no relation between the emission of a message and the behaviour of the environment. In a sense, information can only flow from transmitters to receivers, while the vice versa does not hold. As a consequence, high-level output operations cannot help the low-level user to deduce the behaviour of the high-level user.

**Proposition 20.** \( P \in \text{rd-NNI} \iff P \in \text{rd-SNNI}. \)

**Proof.** (\( \Leftarrow \)) It immediately follows from the definitions of rd-NNI and rd-SNNI.

(\( \Rightarrow \)) By hypothesis, \( \forall \gamma \in T(P), \exists \delta \in T(P) \text{ such that } \text{high}(\delta) = \emptyset \text{ and } \text{low}(\delta) \in S(\text{low}(\gamma)). \) Since \( \text{high}(\delta) = \emptyset \) and output is non-blocking, there exists \( \delta' \in T(P) \) such that \( \delta' = \text{low}(\delta), \) from which the result follows. \( \square \)

Both rd-NNI and rd-SNNI can be given an algebraic characterization based on SAL, as stated by the following proposition.

**Proposition 21** (SAL versions of rd-NNI and rd-SNNI).

(i) \( P \in \text{rd-NNI} \iff P/\mathcal{M}_H \approx_{\text{rd}} (P/\mathcal{M}_H)/\mathcal{M}_H. \)

(ii) \( P \in \text{rd-SNNI} \iff P/\mathcal{M}_H \approx_{\text{rd}} P/\mathcal{M}_H. \)

**Proof.** (i) \( (\Leftarrow ) \forall \gamma \in T(P), \text{low}(\gamma) \in T(P/\mathcal{M}_H). \) By hypothesis, \( \exists \gamma' \in T((P/\mathcal{M}_H)/\mathcal{M}_H) \text{ such that } \gamma' \in S(\text{low}(\gamma)). \) Therefore, \( \text{high}(\gamma') = \emptyset \) and \( \exists \delta \in T(P) \text{ such that } \gamma' = \text{low}(\delta) \text{ and } \text{high}(\delta) = \emptyset, \) from which the result follows.

(\( \Rightarrow \)) On the one hand, \( \forall \gamma \in T((P/\mathcal{M}_H)/\mathcal{M}_H), \exists \delta \in T(P/\mathcal{M}_H) \text{ such that } \delta \in S(\gamma) \text{ since } T((P/\mathcal{M}_H)/\mathcal{M}_H) \subseteq T(P/\mathcal{M}_H). \) On the other hand, \( \forall \gamma \in T(P/\mathcal{M}_H), \exists \gamma' \in T(P) \text{ such that } \text{low}(\gamma') = \gamma. \) By hypothesis, \( \exists \delta \in T(P) \text{ such that } \text{low}(\delta) \in S(\text{low}(\gamma')) \text{ and } \text{high}(\delta) = \emptyset. \) Therefore, \( \exists \delta \in T(P/\mathcal{M}_H) \text{ such that } \delta \in S(\text{low}(\gamma')) \) and \( \text{low}(\delta) \in S(\text{low}(\gamma')). \) The result derives from \( \forall \gamma \in T(P), \text{low}(\gamma) \in T(P/\mathcal{M}_H). \) By hypothesis, \( \exists \delta \in T(P/\mathcal{M}_H) \text{ such that } \delta \in S(\text{low}(\gamma)), \) from which we derive the result, since \( \delta \in T(P). \)

(ii) \( (\Leftarrow ) \forall \gamma \in T(P), \text{low}(\gamma) \in T(P/\mathcal{M}_H). \) By hypothesis, \( \exists \delta \in T(P/\mathcal{M}_H) \text{ such that } \delta \in S(\text{low}(\gamma)), \) from which we derive the result, since \( \delta \in T(P). \)

(\( \Rightarrow \)) On the one hand, \( \forall \gamma \in T(P/\mathcal{M}_H), \exists \delta \in T(P/\mathcal{M}_H) \text{ such that } \delta \in S(\gamma) \text{ since } T(P/\mathcal{M}_H) \subseteq T(P/\mathcal{M}_H). \) On the other hand, \( \forall \gamma \in T(P/\mathcal{M}_H), \exists \gamma' \in T(P) \text{ such that } \text{low}(\gamma') = \gamma. \) By hypothesis, \( \exists \delta \in T(P) \text{ such that } \delta \in S(\text{low}(\gamma')) \) and \( \text{low}(\delta) \in S(\text{low}(\gamma')). \) From which we derive the result, since \( \delta \in T(P/\mathcal{M}_H). \) \( \square \)

The following examples make it clear the relation between the properties presented above and the counterparts defined in the synchronous setting of [13].

**Example 22.** Consider agent \( \text{in}(h).\text{out}(l).0, \) with \( h \in \mathcal{M}_H \) and \( l \in \mathcal{M}_L. \) A low-level observer can easily distinguish between the case in which tuple \( (h) \) is not in the high-level part of the tuplespace and that in which the agent consumes a tuple \( (l) \) previously emitted by the high-level user. Indeed, by verifying the presence of tuple \( (l) \) in the low-level part of the tuplespace (and without directly accessing its high-level part), a low-level observer can infer the behaviour of the high-level user (see Fig. 1). Formally, the agent is not rd-NNI. Similarly, in the synchronous setting of [13], the corresponding agent is not NNI as the low-level output is blocked until the execution of the high-level input, which is an event governed by the high-level user.

The intuition behind the example above is that NNI and rd-NNI capture the same kind of interferences. This is because both synchronous and asynchronous models of communication consider inputs as blocking operations.
Example 23. Consider the same example surveyed above and replace the input operation by a corresponding output operation, thus obtaining the agent \textit{out}(h).out(l).0. By observing the low-level part of the tuplespace, the low-level user cannot infer anything about the behaviour of the high-level user. Indeed, the emission of tuple \textlangle l\textrangle and the consumption (by the high-level user) of the emitted tuple \textlangle h\textrangle are two independent events (see Fig. 2). Formally, the agent is \textit{rd}-SNNI. On the contrary, in the synchronous setting of [13], the corresponding agent is NNI, but not SNNI, as the output operation is blocking, i.e. the execution of the low-level output reveals that a high-level user accepted the high-level output.

As an expected result, in a synchronous communication model the high-level user has at its disposal more kinds of interferences with respect to the asynchronous setting to set up a covert channel from high level to low level. Indeed, when communication is asynchronous, outputs cannot be exploited to deduce the behaviour of the high-level user.

We now investigate the relation between the two different models of communication by considering other security properties based on the noninterference approach to information flow analysis. Among them, we choose the most intuitive security property defined in the process algebraic setting of [13]: Nondeducibility of Composition (NDC, for short). Such a property explicitly states what a high-level user can do and how it can interact with the system in order to affect the low-level behaviour of the system. More precisely, the high-level interferences are determined by checking the effect of putting in parallel a high-level agent with the system. In the context of SAL, NDC reveals which kind of interactions between the tuplespace and the high-level user contribute to establish an information flow. The definition of \textit{rd}-trace NDC (\textit{rd-NDC}) is based on the function lowtr : \mathcal{A} \rightarrow \mathcal{P}(\mathcal{Act}_{L})^\ast, which returns the set of traces of an agent \textit{P} as observed by a low-level user: lowtr(\textit{P}) = \{\gamma \in \mathcal{Act}_{L}^\ast | \exists \delta \in T(\textit{P}). \gamma = \text{low}(\delta)\}. Algebraically, we have that lowtr(\textit{P}) = T(\textit{P}|\mathcal{M}_H).

\textbf{Definition 24 (\textit{rd-NDC}).} \textit{P} \in \textit{rd-NDC} \iff lowtr(\textit{P}) = lowtr((\textit{P}|\Pi)|\mathcal{M}_H) \text{ for all } \Pi \in \mathcal{A}_H.

A SAL characterization of \textit{rd-NDC} is as follows.

\textbf{Proposition 25 (SAL version of \textit{rd-NDC}).} \textit{P} \in \textit{rd-NDC} \iff \textit{P}/\mathcal{M}_H \approx_{\text{rd}} (\textit{P}|\Pi)|\mathcal{M}_H \text{ for all } \Pi \in \mathcal{A}_H.

\textbf{Proof.} The result immediately derives from lowtr(\textit{P}) = lowtr(\textit{Q}) if and only if \textit{P}/\mathcal{M}_H \approx_{\text{rd}} \textit{Q}/\mathcal{M}_H. \qed

Differently from \textit{rd-NNI} and \textit{rd-SNNI}, \textit{rd-NDC} is not decidable. This is because of the universal quantification over all high-level agents. However, like in the synchronous setting, a static characterization of \textit{rd-NDC} is given by \textit{rd-SNNI}, which is equal to \textit{rd-NDC} despite of their very different intuitions and formalizations.

\textbf{Proposition 26.} \textit{P} \in \textit{rd-NDC} \iff \textit{P} \in \textit{rd-SNNI}.
Proof. Let $P$ be a SAL agent.

$(\Rightarrow)$ If we take $\Pi \triangleq \emptyset$, it follows $P/\mathcal{M}_H \approx_{\text{rd}} ((P/\mathcal{M}_H)[\emptyset]) \setminus \mathcal{M}_H$ from which we derive $P/\mathcal{M}_H \approx_{\text{rd}} P \setminus \mathcal{M}_H$.

$(\Leftarrow)$ Let $\Pi \in \mathcal{A}_H$. On the one hand, $T((P/\Pi)[\mathcal{M}_H]) \subseteq T(P/\mathcal{M}_H)$. Hence, $\forall \gamma \in T((P/\Pi)[\mathcal{M}_H]), \exists \delta \in T(P/\mathcal{M}_H)$ such that $\delta \in S(\gamma)$. On the other hand, by hypothesis we have that if $\gamma \in T(P/\mathcal{M}_H)$ then $\exists \delta \in T(P/\mathcal{M}_H)$ such that $\delta \in S(\gamma)$. Hence, $\delta \in T((P/\Pi)[\mathcal{M}_H])$ as $T(P/\mathcal{M}_H) \subseteq T((P/\Pi)[\mathcal{M}_H])$. \hfill $\Box$

Note that from such a result and from Proposition 20 we immediately derive that $\text{rd-NDC} = \text{rd-NNI}$. That means the real interference caused by $\Pi$ has some effect only on the high-level input/read interface of $P$. In practice, $P$ can perform all of its high-level output operations independently of the behaviour of $\Pi$, as they model signals whose occurrence is indistinguishable from internal operations. Instead, the high-level inputs of $P$ can perform all of its high-level output operations independently of the behaviour of $\Pi$. In practice, in an asynchronous communication model where the observational power of the external low-level observer is determined by the $\text{rd}$-trace semantics, the expressive power of each interfering attacker is defined by the high-level outputs it performs.

Definition 27 (rd-NDC w.r.t. High-level Output Agents). $P \in \text{rd-NDC}_{out} \iff \text{lowtr}(P) = \text{lowtr}((P/\mathcal{M}_H)[\Pi]) \setminus \mathcal{M}_H$ for all $\Pi \in \mathcal{A}_H^0$.

Proposition 28 (SAL version of rd-NDC$_{out}$). $P \in \text{rd-NDC}_{out} \iff P/\mathcal{M}_H \approx_{\text{rd}} ((P/\mathcal{M}_H)[\Pi]) \setminus \mathcal{M}_H$ for all $\Pi \in \mathcal{A}_H^0$.

Proof. Trivial. \hfill $\square$

The $\text{rd-NDC}_{out}$ property is quantified over all high-level output agents, i.e. high-level agents that perform internal and output operations only. In spite of such a reduced set of testers, $\text{rd-NDC}_{out}$ turns out to be equal to $\text{rd-NDC}$, which is a result that immediately derives from the following proposition.

Proposition 29. $P \in \text{rd-NDC}_{out} \iff P \in \text{rd-NNI}$.

Proof. Let $P$ be a SAL agent.

$(\Rightarrow)$ If we take $\Pi \triangleq \emptyset$, it follows $P/\mathcal{M}_H \approx_{\text{rd}} ((P/\mathcal{M}_H)[\emptyset]) \setminus \mathcal{M}_H \approx_{\text{rd}} (P/\mathcal{M}_H) \setminus \mathcal{M}_H \approx_{\text{rd}} (P/\mathcal{M}_H) \setminus \mathcal{M}_H$.

$(\Leftarrow)$ Let $\Pi \in \mathcal{A}_H^0$. On the one hand, $T(((P/\mathcal{M}_H)[\Pi]) \setminus \mathcal{M}_H) \subseteq T(P/\mathcal{M}_H)$. Hence, $\forall \gamma \in T(((P/\mathcal{M}_H)[\Pi]) \setminus \mathcal{M}_H)$, $\exists \delta \in T(P/\mathcal{M}_H)$ such that $\delta \in S(\gamma)$. On the other hand, if $\gamma \in T(P/\mathcal{M}_H)$ then $\exists \delta \in T(P/\mathcal{M}_H)$ such that $\delta \in S(\gamma)$. Since $T(((P/\mathcal{M}_H)[\Pi]) \setminus \mathcal{M}_H) \subseteq T(((P/\mathcal{M}_H)[\Pi]) \setminus \mathcal{M}_H)$. \hfill $\Box$

In practice, in an asynchronous communication model where the observational power of the external low-level observer is determined by the $\text{rd}$-trace semantics, the expressive power of each interfering attacker is defined by the high-level outputs it performs.

The intuition behind the rd-NDC$_{out}$ property is the same as that underlying the definition of Nondeducibility on Strategies of [34] (NDS, for short). The model used in [34] consists of a nondeterministic state machine that is controlled by a low-level user and by a high-level user. Inputs to the machine come from both users and, on the basis of the received values, the machine produces separate outputs for both users. NDS states that a system $P$ is secure when all of its low-level views are still possible when composing $P$ with any high-level strategy $\Pi$. By employing a definition borrowed from [34], a high-level strategy $\Pi = \Pi^1, \ldots, \Pi^n$ is a sequence of functions $\Pi^i : \text{High Input} \times \text{High Output} \rightarrow \text{High Input}$, where each function reads the previous pair of high-level input and output and returns the next high-level input. The NDS definition model requires the input totality property, i.e. systems are always able to accept any input or, in other words, outputs are never blocked. In practice, a nondeterministic state machine can emit its outputs regardless of the environment behaviour. In our model, the communication medium, which is explicitly modeled by the tuplespace, has, in a sense, the input totality property, since it ensures that output is a non-blocking operation. Similarly to the NDS definition, rd-NDC$_{out}$ determines, on the basis of the past interactions between $P$ and the high-level agent $\Pi$, which input/read actions of $P$ are enabled for execution. Instead, the output actions of $P$ are completely disregarded since they are not under the control of $\Pi$. Note
that in the synchronous setting of [13], NDC and NDS cannot be directly compared, unless the input totality property is explicitly stated. However, we point out that by avoiding the need for explicitly introducing processes that enable all possible inputs, in our model we also avoid infinitary descriptions (differently from, e.g., [21]).

In the following, we formalize the relation between rd-NDC_{out} and NDS by giving an alternative characterization of rd-NDC_{out} inspired by NDS. To this aim, we first replace the notion of high process \( H \) by a notion of high-level strategy in the same line of \([34]\). Then, we check if the low-level view of the tuplespace is invariant under changes in the high-level strategy applied to \( P \). Our definition of strategy must take into account two issues. On the one hand, a strategy should control both input and read actions of \( P \), which are blocking operations whose execution is guided by the environment. On the other hand, it is not worth considering the output actions of \( P \), because they occur independently of the environment behaviour. Therefore, a high-level strategy just determines which (if any) high-level input/read actions the system is allowed to execute step by step.

Formally, we define a high-level strategy as a partial function \( \mathcal{S} : (\text{Act}_H)^* \rightarrow \text{Act}_H \cap I \). In the following, we assume that each state of the labelled transition system associated to agent \( P \) and strategy \( \mathcal{S} \) is described by a pair \((P', \gamma)\), where \( P' \) is a SAL agent reachable from \( P \) and \( \gamma \) expresses the sequence of high-level operations executed along the path from \( P \) to \( P' \) by following strategy \( \mathcal{S} \). For the initial state, the associated configuration is \((P, \langle \rangle)\).

Note that for each agent \( P' \) reachable from \( P \) we can have several different states of the form \((P', \gamma)\) depending on the number of traces (differing in their high-level actions) that may be executed to reach \( P' \). We now are ready to introduce a strategy operator \((P, \gamma)|\mathcal{S}\), which is defined by the semantic rules of Table 3. \((P, \gamma)|\mathcal{S}\) blocks all the high-level input/read actions of \( P \) except for the one enabled by \( \mathcal{S} \), which is executed as an unobservable transition. On the other hand, all the high-level output actions of \( P \) are simply hidden as they are not observable at low level. Moreover, note that \( \gamma \) stores all the high-level actions that are executed before reaching \( P \). Then, our definition of Nondeducibility on Strategies for SAL agents, which we call rd-NDS, is as follows.

**Definition 30** (rd-NDS). \( P \in \text{rd-NDS} \iff \text{lowtr}(P) = \text{lowtr}((P, \langle \rangle)|\mathcal{S}) \) for all high-level strategies \( \mathcal{S} \).

**Proposition 31** (SAL version of rd-NDS). \( P \in \text{rd-NDS} \iff P/\mathcal{M}_H \approx_{\text{rd}} (P, \langle \rangle)|\mathcal{S} \) for all high-level strategies \( \mathcal{S} \).

**Proof.** Trivial. \( \square \)

As in the case of rd-NDC, we need a static characterization of rd-NDS, which is not decidable because of the universal quantification over all high-level strategies.

**Proposition 32.** \( P \in \text{rd-NDS} \iff P \in \text{rd-NNI} \).

**Proof.** Let \( P \) be a SAL agent.

\((\Rightarrow)\) By taking the strategy that does not associate any high-level action to any sequence, by hypothesis and by the semantic rules of Table 3 we easily derive \( P/\mathcal{M}_H \approx_{\text{rd}} (P/\mathcal{M}_H)/\mathcal{M}_H \).

\((\Leftarrow)\) Let \( \mathcal{S} \) be a high-level strategy. On the one hand, \( T((P, \langle \rangle)|\mathcal{S}) \subseteq T(P/\mathcal{M}_H) \). Hence, \( \forall \gamma \in T((P, \langle \rangle)|\mathcal{S}) \), \( \exists \delta \in T(P/\mathcal{M}_H) \) such that \( \delta \in \mathcal{S}(\gamma) \). On the other hand, if \( \gamma \in T(P/\mathcal{M}_H) \) then \( \exists \delta \in T((P/\mathcal{M}_H)/\mathcal{M}_H) \) such that \( \delta \in \mathcal{S}(\gamma) \). Since \( T((P/\mathcal{M}_H)/\mathcal{M}_H) \subseteq T((P, \langle \rangle)|\mathcal{S}) \) then \( \delta \in T((P, \langle \rangle)|\mathcal{S}) \). \( \square \)

From such a result we immediately derive that rd-NDS = rd-NDC_{out}. We can conclude by observing that all the security properties defined so far have the same expressive power. Hence, compositionality results for all properties can be summarized in the following proposition.
Proposition 33.

(1) \( P \in \text{rd-NNI} \Rightarrow P \setminus L \in \text{rd-NNI}, L \subseteq \mathcal{M} \).

(2) \( P, Q \in \text{rd-NNI} \Rightarrow P \mid Q \in \text{rd-NNI} \).

Proof. The proof is standard. It follows from Proposition 6 and from the corresponding proof of [13]. □

The classification of security properties together with a comparison with the synchronous setting of [13] is summarized in Fig. 3. There, by NDCIT we denote the NDC property restricted to input total systems, which is the version of NDC introduced in [13] as a counterpart of the NDS property of [34]. We omit the definition of further security properties – in particular those properties requiring the input totality condition – as they would not yield novel insights in our setting, where the asynchronous communication model is explicitly assumed.

3.2. Bisimulation-based properties

In this section we rephrase some security property in the context of the bisimulation semantics. We start by discussing in detail the case of the weak \( \text{rd-} \)-bisimulation. On the one hand, we show that the taxonomy we obtain introduces inclusion relations that do not coincide with those observed in the previous section. On the other hand, since weak \( \text{rd-} \)-bisimulation implies \( \text{rd-} \)-trace equivalence, every property we present in this section is finer than its trace-based counterpart. Finally, we discuss the extension to the case of the congruence-based semantics.

In the following definition we introduce the \( \approx_{\text{Brd}} \)-based variants of NNI, SNNI, and NDC, respectively.

Definition 34 (\( \text{rd-BNNI}, \text{rd-BSNNI}, \text{rd-BNDC} \)).

- \( P \in \text{rd-BNNI} \Leftrightarrow P / \mathcal{M}_H \approx_{\text{Brd}} (P \setminus_1 \mathcal{M}_H) / \mathcal{M}_H \).
- \( P \in \text{rd-BSNNI} \Leftrightarrow P / \mathcal{M}_H \approx_{\text{Brd}} P \setminus \mathcal{M}_H \).
- \( P \in \text{rd-BNDC} \Leftrightarrow P / \mathcal{M}_H \approx_{\text{Brd}} (P \mid \Pi) \setminus_0 \mathcal{M}_H \text{ for all } \Pi \in \mathcal{A}_H \).

With respect to the trace-based scenario, the equivalence relation between \( \text{rd-BNNI} \) and \( \text{rd-BSNNI} \) does not hold. This is because the weak \( \text{rd-} \)-bisimulation equivalence is able to detect deadlocks and to discriminate agents according to the nondeterministic structure of their labelled transition systems.

Example 35. Consider agent \( P \overset{\text{def}}{=} \text{out}(h).\text{in}(h).\text{out}(l) \), where \( h \in \mathcal{M}_H \) and \( l \in \mathcal{M}_L \), and verify the \( \text{rd-BSNNI} \) property. On the one hand, the semantics of \( P \setminus \mathcal{M}_H \) is described as follows:

\[
\begin{align*}
\bullet \quad & (\text{out}(h).\text{in}(h).\text{out}(l)) \setminus \mathcal{M}_H \\
\tau \quad & ((h) \mid \text{in}(h).\text{out}(l)) \setminus \mathcal{M}_H \\
\tau \quad & (\text{out}(l)) \setminus \mathcal{M}_H \\
\tau \quad & ((l)) \setminus \mathcal{M}_H \\
\tilde{i} \quad & 0
\end{align*}
\]
Intuitively, the agent emits a high-level tuple $\langle h \rangle$ and then can consume it since the environment is prevented from observing $h$. Then, the agent outputs the low-level message, which can be read/consumed by the low-level user. On the other hand, the semantics of $P/\mathcal{M}_H$ is given by:

![Diagram]

The intuition is that since each high-level operation is hidden, then agent $P$ emits, after a number of internal steps, tuple $\langle l \rangle$. Therefore, $P \in rd$-BSNNI. Such a property does not reveal a potential interference of the high-level user that can consume the high-level message $h$ before the execution of the input operation $in(h)$, thus preventing the agent from emitting tuple $\langle l \rangle$. Such an interference is captured by the $rd$-BNNI property. Indeed, the semantics of $(P\backslash I, \mathcal{M}_H)/\mathcal{M}_H$ is given by:

![Diagram]

As a consequence, $P \notin rd$-BNNI. The problem revealed by the $rd$-BNNI property can be solved by requiring $h$ to be a local name of $P$ that cannot be consumed by any high-level user. For instance, we have that $P\backslash h \in rd$-BNNI, $rd$-BSNNI. Finally, we point out that in the synchronous setting of [13], $P$ is neither BNNI nor BSNNI.

**Example 36.** Consider the following slight variant of agent $P$ of Example 35: $Q \overset{\text{def}}{=} out(h).in(h).out(l) + in(h)$. We first observe that $Q \in rd$-BNNI. Indeed, agent $(Q\backslash I, \mathcal{M}_H)/\mathcal{M}_H$ behaves exactly as agent $(P\backslash I, \mathcal{M}_H)/\mathcal{M}_H$, i.e. the emission of tuple $\langle l \rangle$ depends on the high-level environment behaviour. Similarly, we have that $Q/\mathcal{M}_H \approx_{B_{rd}} (P/\mathcal{M}_H + \tau_0).Q$, i.e. the agent can either evolve into a deadlock state or execute a process that leads to the emission of tuple $\langle l \rangle$. However, $rd$-BNNI does not reveal that the absence of any high-level interference causes the agent to emit tuple $\langle l \rangle$. This is shown by verifying that $Q \notin rd$-BSNNI. In fact, the semantics of $Q/\mathcal{M}_H$ is the same as that of $P\backslash \mathcal{M}_H$, which is forced to emit tuple $\langle l \rangle$. Hence, $rd$-BSNNI reveals a covert channel that is not captured by $rd$-BNNI. Similarly as seen in Example 35, note that $Q\backslash h \in rd$-BNNI, $rd$-BSNNI. Finally, we point out that in the synchronous setting of [13], $Q$ is neither BNNI nor BSNNI.
As we have seen in the examples above, rd-BNNI and rd-BSNNI do not match whenever the agent can consume a tuple that it has previously emitted. Indeed, in such a case the interleaving between the high-level user operations and the agent activities interferes with the low-level view of the tuplespace. We can avoid such a kind of interference if we assume that the agent distinguishes between tuples offered to the environment, which are disregarded by the agent itself, and tuples that the agent uses for its internal calculations, which must be modeled through local names and cannot be observed by the environment.

**Definition 37.** An agent \( P \in A \) is said to be a non-consuming producer (ncp, for short) if \( \forall P' \). \( P \Rightarrow P' \), whenever \( P' = \langle a \rangle | Q \) then \( \exists Q' \) such that \( Q \overset{a}{\Rightarrow} Q' \) or \( Q \overset{\bar{a}}{\Rightarrow} Q' \).

An ncp agent does not read/consume the tuples offered to the environment by the agent itself. For instance, agent \( P \) of Example 35 is not an ncp agent. Indeed, we have seen, \( P \) can first emit a tuple \( \langle h \rangle \), and then can either offer it to the environment or consume it via a corresponding input operation. On the other hand, process \( P \setminus h \) is an ncp agent, since it cannot offer a tuple \( \langle h \rangle \) to the environment. We can argue similarly for agent \( Q \) described in Example 36.

**Lemma 38.** Let \( \langle a \rangle | P \) be an ncp agent. Then \( (\langle a \rangle | P) a \approx_{B_{rd}} P \setminus a \).

**Proof.** It is enough to observe that \((\langle a \rangle | P) a \) and its derivatives do not enable \( \bar{a} \) labelled transitions and, by definition of ncp agent, do not enable synchronizations involving tuple \( \langle a \rangle \). \( \square \)

Intuitively, in an asynchronous communication model it is natural to view an output operation offered to the environment as a signal emitted by the agent, which is eventually received by an external process. Under such an assumption, many realistic systems can be interpreted as non-consuming producers. By restricting ourselves to ncp agents, we now show that rd-BNNI and rd-BSNNI coincide. Instead, we recall that in the synchronous communication setting BNNI \( \not\subset \) BSNNI and BSNNI \( \not\subset \) BNNI [13].

**Proposition 39.** If \( P \in A \) is an ncp agent, then

\( P \in rd\text{-}BNNI \iff P \in rd\text{-}BSNNI. \)

**Proof.** Let \( P \) be a SAL agent and \( R \) be a relation defined as follows:

\((P' \setminus I)M_H, P'' \setminus I)M_H) \in R, \forall P', P \Rightarrow P'. \)

We now prove that \( R \) is a weak rd-bisimulation up to \( \approx_{B_{rd}}. \) By inspection of possible cases, it follows:

- if \( P'' \setminus I)M_H \overset{\pi}{\Rightarrow} P'' \setminus I)M_H \) then \((P'' \setminus I)M_H) / M_H \overset{\pi}{\Rightarrow} (P'' \setminus I)M_H) / M_H \), since the transitions enabled in \( P'' \setminus I)M_H \) are a subset of the transitions enabled in \((P'' \setminus I)M_H) / M_H \):
- if \( (P'' \setminus I)M_H) / M_H \overset{\pi}{\Rightarrow} (P'' \setminus I)M_H) / M_H \), where \( \pi \) is either a visible action or a \( \tau \) action that is not obtained by hiding a \( h \) labelled transition enabled in \( P'' \setminus I)M_H \), then \( P'' \setminus I)M_H \overset{\pi}{\Rightarrow} P'' \setminus I)M_H \), since \( \pi \) models an event of term \( P' \) that is not restricted in \( P'' \setminus I)M_H \).
- if \( P' = \langle h \rangle | P'' \) and \((\langle h \rangle | P'' \setminus I)M_H) / M_H \overset{\tau}{\Rightarrow} (P'' \setminus I)M_H) / M_H \), then, by virtue of Lemma 38 and Proposition 9, we have that \((\langle h \rangle | P'') \setminus I)M_H \approx_{B_{rd}} P'' \setminus I)M_H \).

Hence, \( P'' \setminus I)M_H \approx_{B_{rd}} (P'' \setminus I)M_H) / M_H \), from which we derive the result. \( \square \)

From the proof of Proposition 39 we immediately derive the following corollary, which confirms that the high-level output operations do not interfere with the low-level content of the tuplespace in the case of ncp agents.

**Corollary 40.** If \( P \in A \) is an ncp agent, then

\( P/O)M_H \approx_{B_{rd}} P/O)M_H. \)

\(^1\) The definition is a straightforward extension of that of [31].
Proposition 41 (Inclusion Relations).

1. \(rd\text{-BNDC} \subseteq rd\text{-BNNI}\).
2. \(rd\text{-BNDC} \subseteq rd\text{-BSNNI}\).

**Proof.** Let \(P\) be a SAL agent. To show condition (1), consider the process that enables all the high-level input operations: \(\Pi \triangleq \sum_{a \in \mathcal{M}_H} \text{in}(a)\). Then, by hypothesis \(P/\mathcal{M}_H \approx_{\text{B}_{rd}} (P/\Pi)/\mathcal{M}_H \approx_{\text{B}_{rd}} (P/\emptyset)/\mathcal{M}_H \approx_{\text{B}_{rd}} P/\mathcal{M}_H\). To show condition (2), consider \(\Pi \triangleq 0\). Then, by hypothesis \(P/\mathcal{M}_H \approx_{\text{B}_{rd}} (P/\emptyset)/\mathcal{M}_H \approx_{\text{B}_{rd}} P/\mathcal{M}_H\).

The inclusions are strict. For instance, agent \(\text{out}(l) + rd(h).rd(h').\text{out}(l)\), with \(h, h' \in \mathcal{M}_H\) and \(l \in \mathcal{M}_L\), is both \(rd\text{-BNNI}\) and \(rd\text{-BSNNI}\), but not \(rd\text{-BNDC}\). Indeed, it is enough to take \(\Pi \triangleq \text{out}(h)\). \(\square\)

Note that the same inclusion relations hold in the synchronous setting of [13].

**Lemma 42 (rd-BNDC: Alternative Definition).** \(P \in rd\text{-BNDC} \iff P/\mathcal{M}_H \approx_{\text{B}_{rd}} (P/\Pi)/\mathcal{M}_H\) for all \(\Pi \in \mathcal{A}_H\).

**Proof.** (\(\Rightarrow\)) A consequence of condition (2) of Proposition 41 and of Definition 34.

(\(\Leftarrow\)) It is enough to follow an argument similar to that in the proof of Proposition 41 to show that \(P/\mathcal{M}_H \approx_{\text{B}_{rd}} P/\mathcal{M}_H\), from which the result stems. \(\square\)

While both \(rd\text{-BSNNI}\) and \(rd\text{-BNNI}\) are decidable, \(rd\text{-BNDC}\) is not, because of the universal quantification over all high-level agents. Here, differently from the previous section, we do not provide a static characterization of \(rd\text{-BNDC}\). Instead, we define a decidable property that does not involve composition with every high-level process \(\Pi\) and is stricter than \(rd\text{-BNDC}\). Such a property, termed Strong \(rd\text{-BNDC}\) (\(rd\text{-SBNNDC}\), for short), is defined in [13].

**Definition 43 (rd-SBNNDC).** \(P \in rd\text{-SBNNDC} \iff \forall P'. P \Rightarrow P'\) and \(\forall P''. P' \stackrel{\alpha}{\rightarrow} P''\), where \(\alpha \in \text{Act}_H\), then \(P/\text{Act}_H \approx_{\text{B}_{rd}} P''/\text{Act}_H\).

By employing the alternative definition of \(rd\text{-BNDC}\) given in Lemma 42, we now show that \(rd\text{-SBNNDC}\) is finer than \(rd\text{-BNDC}\).

**Proposition 44.** \(rd\text{-SBNNDC} \subseteq rd\text{-BNDC}\).

**Proof.** Let \(P\) be a SAL agent, \(\Pi\) be a high-level agent and \(R\) be a relation defined as follows:

\[(P'/\mathcal{M}_H, (P'/\Pi)/\mathcal{M}_H) \in R, \forall P'. P \Rightarrow P'\) and \(\forall \Pi''. \Pi \Rightarrow \Pi''.\]

By following the same argument shown in the proof of Proposition 39, we now prove that \(R\) is a weak \(rd\text{-bisimulation}\) up to \(\approx_{\text{B}_{rd}}\). By inspection of possible transitions, the unique interesting case is given by \((P'/\Pi)/\mathcal{M}_H \Rightarrow (P''/\Pi'')/\mathcal{M}_H\), such that there exist \(\alpha, \alpha' \in \text{Act}_H\), with \(P' \stackrel{\alpha}{\rightarrow} P''\), \(\Pi' \stackrel{\alpha'}{\rightarrow} \Pi''\), and \(\alpha, \alpha'\) are of the same type and can synchronize. By hypothesis, \(P'/\mathcal{M}_H \approx_{\text{B}_{rd}} P''/\mathcal{M}_H\). Therefore, since \((P'/\mathcal{M}_H, (P''/\Pi'')/\mathcal{M}_H) \in R\) it follows that \(R\) is a weak \(rd\text{-bisimulation}\) up to \(\approx_{\text{B}_{rd}}\) and \(P\) is \(rd\text{-BNDC}\).

The inclusion is strict. For instance, it can be verified that the following agent \(\text{out}(l) + \text{out}(l').\text{out}(l') + \text{out}(l).rd(h).\text{out}(l')\), with \(h \in \mathcal{M}_H\) and \(l, l' \in \mathcal{M}_L\), is \(rd\text{-BNDC}\) but not \(rd\text{-SBNNDC}\). \(\square\)

Similarly as in the trace-based scenario, it is not worth imposing any check on the high-level outputs, provided that we restrict ourselves to \(ncp\) agents. To show this, we define a variant of \(rd\text{-SBNNDC}\), termed \(rd\text{-SBNNDC}_{\text{out}}\), which does not consider high-level outputs at all. In spite of such a relaxation, \(rd\text{-SBNNDC}\) and \(rd\text{-SBNNDC}_{\text{out}}\) have the same expressive power.

**Definition 45 (rd-SBNNDC\text{\text{out}}).** \(P \in rd\text{-SBNNDC}_{\text{out}} \iff \forall P'. P \Rightarrow P'\) and \(\forall P''. P' \stackrel{\alpha}{\rightarrow} P''\), where \(\alpha \in \text{Act}_H \cap I\), then \(P/\text{Act}_H \approx_{\text{B}_{rd}} P''/\text{Act}_H\).

**Proposition 46.** If \(P \in \mathcal{A}\) is an \(ncp\) agent, then \(P \in rd\text{-SBNNDC} \iff P \in rd\text{-SBNNDC}_{\text{out}}\).
We shall prove that Proposition 9 and from the corresponding proof of Proposition 49, we have \((\langle h \rangle|P''\rangle|\mathcal{M}_H \approx_{B_{rd}} P''\rangle|\mathcal{M}_H, from which we derive the result. □

Corollary 47. If \(P \in A\) is an ncp agent, then \(rd\)-SBND

The corollary states that a property that does not perform any check on the high-level outputs of an ncp agent is stricter than a property that involves composition of the agent with every high-level process. Since the high-level outputs do not affect the result of the security check, the definition of \(rd\)-BND can be relaxed. In particular, we can hide the output operations of \(P\) before composing in parallel \(P\) and II and then assume that II can perform output operations only. Formally, similarly as in the trace-based scenario, the most intuitive and natural notion of NDC is given by the following property, termed \(rd\)-BND.

**Definition 48** (\(rd\)-BND w.r.t. High-level Output Agents). \(P \in rd\)-BND \(\iff P/\mathcal{M}_H \approx_{B_{rd}} ((P|_{\mathcal{M}_H})|II)\rangle|\mathcal{M}_H\) for all \(II \in \mathcal{A}_n^\Pi\).

**Proposition 49.**

1. \(P \in rd\)-SBND \(\Rightarrow P|L \in rd\)-SBND, \(L \subseteq \mathcal{M}\).
2. \(P, Q \in rd\)-SBND \(\Rightarrow P|Q \in rd\)-SBND.

**Proof.** The proof is standard. It follows from Proposition 9 and from the corresponding proof of [13]. □

The classification of security properties together with a comparison with the synchronous setting of [13] is summarized in Fig. 4.

We now show what happens if we replace \(\approx_{B_{rd}}\) by \(=_{rd}\). Obviously, since \(=_{rd}\) implies \(\approx_{B_{rd}}\) we have that the properties rephrased in the setting of observation \(rd\)-congruence are stricter than the corresponding properties defined so far.

As far as \(rd\)-BNNI and \(rd\)-BSNNI are concerned, we call their counterparts \(rd\)-ENNI and \(rd\)-ESNNI, respectively. Unfortunately, such properties do not preserve the result of Proposition 39. Instead, the following result holds.

**Proposition 50.** If \(P \in A\) is an ncp agent, then

\(P \in rd\)-ESNNI \(\Rightarrow P \in rd\)-ENNI.

**Proof.** We shall prove that \((P|_{\mathcal{M}_H})/\mathcal{M}_H =_{rd} P|\mathcal{M}_H\). By hypothesis, we have that \(P\) is both \(rd\)-BSNNI and \(rd\)-BNNI, so that \((P|_{\mathcal{M}_H})/\mathcal{M}_H \approx_{B_{rd}} P|\mathcal{M}_H\). Thus it is enough to verify the equality between the initial states of such processes. To this aim, we shall prove the most interesting case, which is \(P \equiv \langle h \rangle|Q\) with \((P|_{\mathcal{M}_H})/\mathcal{M}_H \overset{\tau}{\Rightarrow} (Q|_{\mathcal{M}_H})/\mathcal{M}_H\). By hypothesis, \(P|\mathcal{M}_H =_{rd} P/\mathcal{M}_H\). Hence, \((\langle h \rangle|Q)/\mathcal{M}_H \overset{\tau}{\Rightarrow} Q/\mathcal{M}_H\). By hypothesis, \(P\) is \(rd\)-BNNI, from which we derive \(Q/\mathcal{M}_H \approx_{B_{rd}} (Q|_{\mathcal{M}_H})/\mathcal{M}_H\) and then the result.

The inverse does not hold. Indeed, consider agent \(P \overset{\text{def}}{=} \langle h \rangle|in(l), with h \in \mathcal{M}_H and l \in \mathcal{M}_L\). \(P\) is clearly \(rd\)-ENNI since it does not enable high-level read/input operations. On the other hand, \(P\) is not \(rd\)-ESNNI, since \(P/\mathcal{M}_H\) enables a \(\tau\) action which cannot be matched by \(P|\mathcal{M}_H\). □
From Proposition 50 we observe that the inverse inclusion relation holds for ncp agents that do not enable high-level actions at their initial state.

The counterpart of rd-BNDC, called rd-ENNI, preserves the result of Proposition 41, i.e. it implies both rd-ENNI and rd-ESNNI. This can be easily seen by replacing $\approx_{rd}$ by $=_{rd}$ in the corresponding proof. In particular, by applying the same counterexample, it can be viewed that the inclusion is strict.

As far as rd-SBNDC is concerned, we point out that the corresponding property, called rd-SENDC, does not preserve the inclusion relation of Proposition 44, as shown by the same counterexample in the proof of Proposition 50. Moreover, note that rd-SENDC does not imply neither rd-ENNI nor rd-ESNNI. For instance, agent $\langle h \rangle |in(a)$ is rd-SENDC, but not rd-ENNI, rd-ESNNI. This is because observation rd-congruence requires the first unobservable action enabled at the initial state of an agent to be matched by (at least) a corresponding move of the other agent. On the other hand, the result of Proposition 46 still holds in the $=_{rd}$-based scenario, as can be easily verified by replacing $\approx_{Bn}$ by $=_{rd}$ in the corresponding proof. That means, in the case of ncp agents it is not worth imposing any condition on the outputs performed by a system in order to reveal its covert channels. More interestingly, although $=_{rd}$ is preserved by $|$ and $+$, we have that rd-SENDC is not compositional with respect to such operators.

Example 51. Assume $h, h' \in \mathcal{M}_H$ and $a, b \in \mathcal{M}_L$. Agents $P \overset{\text{def}}{=} \langle h \rangle |in(a)$ and $Q \overset{\text{def}}{=} \langle h \rangle |in(a) |in(b)$ are rd-SENDC. Now, consider the composition $P |Q$ and verify the rd-SENDC property. For instance, take $P |Q \overset{h}{\rightarrow} P |in(b)$: we observe that $(P |in(b)) |\mathcal{M}_H =_{rd} \langle a \rangle |in(b)$, which does not enable $\tau$ transitions, while $(P |Q) |\mathcal{M}_H \overset{\text{def}}{=} \langle a \rangle |in(b)$. Hence, $P |Q$ is not rd-SENDC.

Agents $P \overset{\text{def}}{=} |in(a) |+ |in(h) |.in(a)$ and $Q \overset{\text{def}}{=} |in(b) |+ |in(h') |.in(b)$ are rd-SENDC. Instead, $P + Q$ is not. For instance, the consumption of a tuple $\langle a \rangle$ informs the low-level user that the high-level tuple $h'$ has not been consumed. Formally, agent $\langle a \rangle$, which is reachable from $P + Q$ by executing a high-level input, is not $=_{rd}$-equal to $(P + Q) |\mathcal{M}_H$.

From the observations above, we can derive the classification of $=_{rd}$-based security properties depicted in Fig. 5.

As an expected result, replacing $=_{rd}$ by $=_{rd}$, leads to the same considerations surveyed above. The corresponding properties, which we call $rdp$-BNNI, $rdp$-BSNNI, $rdp$-BNDC, and $rdp$-SBNNC, respectively, are stricter than the counterparts based on $=_{rd}$. Moreover, it is worth noting what follows. With respect to the $=_{rd}$-based scenario, we do not have a counterpart of Proposition 50.

Example 52. Similarly as seen in the counterexample of Proposition 50, we have that agent $(\langle h \rangle |in(l))$ is $rdp$-BNNI but not $rdp$-BSNNI. Now, taken agents $Q \overset{\text{def}}{=} out(h) |.in(k) |.in(l)$ and $R \overset{\text{def}}{=} \tau |. in(l)$, with $h, k \in \mathcal{M}_H$ and $l \in \mathcal{M}_L$, consider agent $P \overset{\text{def}}{=} Q + \tau + R$. Since $Q |\mathcal{M}_H \approx_{rd} \tau$, we have that $P |\mathcal{M}_H$ is progressing $rd$-bisimilar to agent $\tau + \tau$. On the other hand, since $Q |\mathcal{M}_H \approx_{rd} \tau$, we have that $P |\mathcal{M}_H$ is progressing $rd$-bisimilar to agent $\tau + \tau$. Hence, $P$ is $rdp$-BSNNI. Now, since $(Q |\mathcal{M}_H) |\mathcal{M}_H$ is progressing $rd$-bisimilar to agent $\tau$, we derive that $(P |\mathcal{M}_H) |\mathcal{M}_H \approx_{rd} \tau + \tau + \tau + R$, which is not progressing $rd$-bisimilar to agent $\tau + \tau + R$ because of the branch $\tau$. Indeed, $\tau + \tau + R \not\approx_{rd} \tau + \tau + R$. Hence, $P$ is not $rdp$-BNDC.

Similarly as seen in the $=_{rd}$-based scenario, we have that $rdp$-BNDC implies both $rdp$-BNNI and $rdp$-BSNNI, as can be easily verified by applying the same proof of Proposition 41. As far as $rdp$-SBNNC is concerned, we can argue as in the case of the $rd$-SENDC property. In particular, the same counterexamples shown above reveal that
Table 4
Access monitor example

\[
\begin{align*}
\text{Low}_{\text{Bit}} & \triangleq \text{in}(r).rd(b_0).out(r_0).\text{Low}_{\text{Bit}} + rd(b_1).out(r_1).\text{Low}_{\text{Bit}} + \text{in}(w_0).rd(b_0).out(ack).\text{Low}_{\text{Bit}} + \text{in}(w_1).rd(b_1).out(ack).\text{Low}_{\text{Bit}} + \\
\text{Monitor} & \triangleq \text{in}(\text{low}_r).out(r).\text{in}(r_0).out(l_0).\text{Monitor} + \text{in}(r_1).out(l_1).\text{Monitor} + \text{in}(\text{high}_r).out(r).\text{in}(r_0).out(b_0).\text{Monitor} + \text{in}(r_1).out(b_1).\text{Monitor} + \\
& \text{in}(\text{low}_w).out(w_0).in(ack).\text{Monitor} + \text{in}(\text{low}_w).out(w_1).in(ack).\text{Monitor} + \text{in}(\text{high}_w).\text{Monitor} + \text{in}(\text{high}_w).\text{Monitor} \\
\text{AM} & \triangleq (\text{Monitor}(|\text{Low}_{\text{Bit}}(b_0)|\backslash\{b_0, b_1\})\{r, r_0, r_1, w_0, w_1, \text{ack}\})
\end{align*}
\]

rdp-SBNDC is not compositional with respect to | and +. In conclusion, in the case of the \(\approx^p_{rd}\)-based semantics, the inclusion relation among the security properties is as shown in Fig. 5.

3.3. The access monitor and the master/worker examples

As a case study we now consider two examples, one taken from the synchronous communication framework of [15] – a monitor accessing a shared memory – and one taken from the classical coordination literature — a master/worker system coordinating job execution [11]. The former is used to show in practice the difference, in terms of information flow analysis, between the two communication models. The latter emphasizes the adequacy of SAL in modeling and analysing typical examples of the coordination world.

System AM of Table 4 represents an access monitor that serves read and write requests on a low-level binary variable enforcing the multi-level security policy [3]. The policy says that a process at \(\sigma\)-security level may only (i) write variables at the same level as \(\sigma\) or above (write up), and (ii) read variables at the same level as \(\sigma\) or below (read down). The system is made of agent Monitor, which accepts the user requests that follow the security policy, and agent Low_{Bit}, which directly controls the binary variable, whose initial value is 0 as specified by tuple \(\langle b_0 \rangle\). All the interactions between Monitor and Low_{Bit} and the operations internally performed by Low_{Bit} are local, i.e. every user cannot offer/consume the related tuples. Formally, such a condition is guaranteed by the restrictions defined in system AM. Agent Monitor is the unique process of the system that can interact with the environment through messages exchanged via the tuplespace. In particular, agent Monitor can consume the messages \text{low}_r, denoting a low-level read request, \text{high}_r, denoting a high-level read request, \text{low}_w, denoting a low-level write request. Moreover, after a high- (low-)level read request, agent Monitor can emit the high- (low-)level messages \text{h}_i (l_i), for \(i \in \{0, 1\}\), denoting that the value \(i\) is communicated to the environment. According to the multi-level security policy, high-level write requests (\text{high}_w) are not satisfied, since a high-level user cannot write a low-level variable. Finally, we point out that AM is an ncp agent.

In the following, we assume that a single low-level user interacts with the system and we show what such a user can learn about the high-level operations performed on the low-level variable. First, system AM is \(rd\)-BSNNI. The intuition is that all the high-level operations do not affect the content of the low-level variable and do not prevent the low-level user from accessing such a variable. Since AM is an ncp agent, \(rd\)-BNNI holds too. Moreover, it can be shown that AM is \(rd\)-SBNDC. Formally, the low-level view of AM does not change when executing the high-level input \text{in}(\text{high}_r) denoting the reception of a high-level read request. In particular, even if the environment refuses to consume the high-level output \text{out}(h_1), which is provided by the monitor in response to the high-level read request, the system does not deadlock. This is because of the asynchronous communication assumption. Instead, the corresponding system based on synchronous communication does not satisfy SBNDC [15]. Finally, by assuming the \(\approx_{nd}\) semantics, the system is not secure. For instance, AM is not \(rd\)-ENNI since, e.g., the internal move that can be performed by hiding action \text{in}(\text{high}_r) cannot be matched by an internal move whenever \text{in}(\text{high}_r) is prevented.

Now, let us consider a variant of agent Monitor that accepts high-level write requests, thus violating the write up condition. Formally, this behaviour is obtained by replacing \text{in}(\text{high}_w).\text{Monitor} + \text{in}(\text{high}_w).\text{Monitor} by:
in\((high\_w_0).\text{out}(w_0).\text{in}(ack).\text{Monitor}\) +
\quad in\((high\_w_1).\text{out}(w_1).\text{in}(ack).\text{Monitor}\).

After such a modification, rd-BNNI and rd-NNI do not hold any more. For both properties, the security check captures the fact that, e.g., a low-level user that does not perform write operations can perceive the high-level interference by reading two different values \((l_0\text{ and } l_1)\) in two different read operations. The same covert channel is revealed by BNNI (NNI) in [15].

As another variant of agent Monitor, assume that a high-level output is added to inform the high-level user that a low-level write operation occurred. This can be modeled by adding the high-level output out\((\text{written}_i)\), for \(i \in \{0, 1\}\), as follows:
\[
\begin{align*}
in\((low\_w_0).\text{out}(w_0).\text{in}(ack).\text{out}(\text{written}_0).\text{Monitor} & + \\
in\((low\_w_1).\text{out}(w_1).\text{in}(ack).\text{out}(\text{written}_1).\text{Monitor}.)
\end{align*}
\]

In the synchronous setting such a version of the monitor does not satisfy SNNI [15], because the high-level user can refuse the feedback concerning the low-level write operation. Hence, two consecutive low-level write operations can be exploited to inform the low-level user that the high-level user is still active. Such a behaviour is more than enough to set up a 1-bit covert channel from high level to low level. In our asynchronous setting, it can be verified that such a version of the access monitor is still secure. In particular, all the weak rd-bisimulation-based security properties defined in Section 3 hold. This is because the high-level user cannot block or delay the output offered by the monitor, which is interpreted as an independent activity (a signal) instead of a synchronous, direct interaction with the high-level user.

The second system we model is a typical example taken from the coordination world: a master/worker system for the interaction between a set of clients, called masters, which produce job requests, and a set of servers, called workers, which process each request and return a result. Communication among the agents is asynchronous. In particular, a master produces a job request by sending a tuple to a shared tuplespace, while a worker performs the job by consuming the related tuple and by emitting a new tuple containing the result. In a multi-level security system we may imagine that jobs at different security levels are handled. For this purpose, we employ a monitor ensuring that the result of a high-level job is not communicated to a low-level master.

For the sake of simplicity, in Table 5 we model a scenario with a single worker interacting with a monitor that accepts job requests from masters at different levels of security. The system, called MW, is made of agent Monitor, which accepts a high- (resp. low-)level job request through a high- (resp. low-)level tuple of type high_req (resp. low_req), and agent Worker, which models the unique worker that executes the job passed by the monitor. After execution, agent Worker generates a result that can be consumed by the monitor only. In turn, agent Monitor emits such a result in the form of a high- (resp. low-)level tuple of type high_res (resp. low_res). Similarly as seen in the case of the access monitor example, it can be observed that the system is secure. In particular, the system is rd-SBNDC, because the potential interaction with a high-level master does not prevent the system from serving low-level job requests. Again, note that the emission of a high-level tuple (of type high_res) is an operation that is independent of the environment behaviour and, therefore, cannot be blocked by a misbehaving high-level master.

The multi-level security constraint can be guaranteed without resorting to the monitor. Indeed, we can define a set of high-level workers and a set of low-level workers that separately handle high- and low-level job requests, respectively. The separation of concerns, as formally illustrated in Table 6 which also shows two possible master processes, ensures that operations at different levels of security cannot interfere each other. In essence, each master directly communicates only with the corresponding worker through the tuplespace. Formally, we can verify that, e.g., system Low_Worker|High_Worker is rd-BSNNI. More interestingly, the system is secure independently of the behaviour of the high-level masters, that means the system is both rd-BNDC and rd-SBNDC.
In a synchronous setting SNNI is stricter than NNI, while BSNNI and BNNI do not satisfy any inclusion relation. By defining a NDS-like property

\[ \text{High} \mapsto \text{Low} \mapsto \text{Free} \]

In an asynchronous setting, the security properties can be relaxed by weakening the constraint imposed on outputs. This relaxation is possible by hiding high-level outputs and considering high-level processes that perform output operations only.

A major direction for future work is the implementation in a software tool of the security checks based on the noninterference properties. For this purpose, it would be interesting to investigate the possibility of extending the security model to accommodate the needs of the high-level user.
security checker illustrated in [15] to deal with SAL processes. Similarly, we think that the theory recently developed in [6,28] could be adapted with minor changes in the setting of SAL. To this aim, it is enough to extend the generalized notions of weak bisimulation of [6] in our asynchronous setting. This can be done by enriching the definitions of [6] to deal with read operations exactly as done in the case of the weak \( rd \)-bisimulation, which is an extension of the weak bisimulation with an additional relaxed condition on the read operations. Then, properties like persistent BNDC (P_BNDC) and progressing persistent BNDC (PP_BNDC) can be rephrased in SAL by following the same approach adopted when passing from SBND to \( rd \)-SBND. The main benefit would be that the related verification techniques could be applied in our asynchronous setting too. For instance, a characterization of P_BNDC based on the unwinding conditions [23] and on the weak \( bisim \)-definition, call it \( \text{rd-P}_\text{BNDC} \), states that \( P \equiv \text{rd-P}_\text{BNDC} \) if and only if \( \forall P', P \Rightarrow P' \) and \( \forall P''. P' \overset{\alpha}{\rightarrow} P'' \), where \( \alpha \in \text{Act}_H \), then \( P' \overset{\alpha}{\Rightarrow} P'' \) and \( P'' \setminus \text{Act}_H \equiv B_d \ P'' \setminus \text{Act}_H \). As can be easily verified, \( \text{rd-P}_\text{BNDC} \) preserves the same results known for P_BNDC, i.e. it is compositional (except with respect to summation) and \( \text{rd-SBND} \subset \text{rd-P}_\text{BNDC} \subset \text{rd-BNDC} \). In particular, by rephrasing the proof of Proposition 46, it follows that high-level outputs can be disregarded in the definition of \( \text{rd-P}_\text{BNDC} \), which can be formally redefined as follows: an ncp agent \( P \) is \( \text{rd-P}_\text{BNDC} \) if and only if \( \forall P', P \Rightarrow P' \) and \( \forall P''. P' \overset{\alpha}{\rightarrow} P'' \), where \( \alpha \in \text{Act}_H \cap I \), then \( P' \overset{\alpha}{\Rightarrow} P'' \) and \( P'' \setminus \text{Act}_H \equiv B_d \ P'' \setminus \text{Act}_H \). More interestingly, PP_BNDC is essentially a version of P_BNDC based on the progressing bisimulation [26]. However, differently from \( \text{rdp-SBND} \), it is preserved by parallel composition and summation. Hence, in order to obtain full compositional property in our asynchronous setting, the adequate property to be considered – together with the progressing bisimulation semantics – is \( \text{P}_\text{BNDC} \) instead of SBND.

Another issue that is planned for future work is the extension of SAL with other Linda-like primitives, like conditional input and conditional read operations [8]. Then, once that automated tools supporting analysis will be provided, it would be interesting to investigate the effectiveness of our approach by analysing some complex case study taken from the literature of coordination models, e.g., conference management on the Web [12], asynchronous circuits [5], and distributed architectures [27].

Along the line of investigation inspired by the noninterference approach, other works in the literature cope with secure information flow analysis and asynchronous communication model. A seminal work in this direction is by Smith and Volpano in [33]. There, the authors develop a type system to ensure secure information flow in a multi-threaded imperative language with nondeterministic scheduling policy. The asynchronous communication model is formalized by means of a single global memory, shared by all threads, through which the concurrent threads can interact. A more restrictive type system is also defined which guarantees noninterference regardless of how (non-probabilistic) scheduling is done. Starting from this point, other works investigate the property of noninterference for well-typed systems with asynchronous parallelism [22,7]. In [22] the authors show that the calculus of [33] is an embedding of a typed calculus that is a variant of \( \pi \)-calculus. The authors of [7] present a type system that is more natural and less restrictive than that of [33]. Both in [22] and in [7] the behavioural theory of typed processes is based on the bisimilarity semantics. Similarly, in [20] resource access control and secure information flow are verified using a typed asynchronous \( \pi \)-calculus. There, the noninterference result is based on a notion of may testing equivalence. The result has been extended in [19] with respect to the more discriminating must testing equivalence. In [32] a security condition called causal flow property is defined to formally prove that software wrappers, which in real systems represent the glue between interacting components, actually preserve secure information flow constraints. This is done in the context of a typed box-\( \pi \) process calculus, which can be viewed as an extension of an asynchronous version of the \( \pi \)-calculus.

References
