Component-Oriented Verification of Noninterference

Alessandro Aldini Marco Bernardo

Dipartimento di Matematica, Informatica, Fisica e Chimica – Università di Urbino, Italy

Abstract

Component-based software engineering often relies on libraries of trusted components that are combined to build dependable and secure software systems. Resource dependences, constraint conflicts, and information flow interferences arising from component combination that may violate security requirements can be revealed by means of the noninterference approach to information flow analysis. However, the security of large component-based systems may be hard to assess in an efficient and systematic way. In this paper, we propose a component-oriented formulation of noninterference that enables compositional security verification driven by system topology. This is realized by implementing scalable noninterference checks in the formal framework of a process algebraic architectural description language equipped with equivalence checking techniques.

Key words: component-based software systems, noninterference analysis, architectural description languages, process algebra, equivalence checking.

1. Introduction

Modern software architectures are built on components that can be developed in-house or provided by third, possibly trusted parties. With the increasing demand for component-based and evolution-resilient applications, the technology for designing these applications must be accompanied by rigorous analysis methods. On the one hand, the specification of these applications can be supported by a significant number of architectural description languages addressing the typical aspects of a software architecture design like, e.g., the elicitation of components, connectors, and topology. On the other hand, their formal verification requires the translation of the architectural representation into a model – e.g., a labeled transition system – that can

be automatically verified through techniques like model checking (see, e.g., [33, 12]) and equivalence checking (see, e.g., [6, 21, 15, 2]). However, models deriving from software architectures that include many interacting components typically suffer from the state space explosion problem. Among the various approaches proposed to face this problem, we concentrate on compositional verification, which basically follows a "divide and conquer" principle where system properties are decomposed into properties of the components. Compositional verification, which employs abstraction capabilities together with analysis techniques like the two mentioned before, plays a fundamental role in the design of component-oriented software architectures (see, e.g., [11] and the references therein).

In this paper, we exploit compositional verification to design secure component-oriented systems, which is often a hard task because of resource dependences, constraint conflicts, and information flow interferences. Indeed, following the rely-guarantee reasoning of [22], a violation of the rely condition concerning the expected environment behavior may interfere with the satisfaction of the guarantee about the behavior of each component. While in the rely-guarantee approach the specification of the interference is global, our objective is to apply compositional verification in a component-oriented fashion in order to securely integrate software components.

As an example, consider a multilevel security routing system like, e.g., the NRL Pump [25], in which several agents at different security clearances compete for the same resources, communicate by sharing common channels, and ask for services involving the interaction of many components that form complex topologies. In this scenario, interferences occur all the time and revealing those that may violate security policies is a challenging task. In order to avoid the analysis of the system as a whole, which could be impractical, it is important to investigate locally the impact upon properties of interest – like the absence of illegal information flows – determined by the behavior of every critical component that is replaced, altered, or added to the system.

A technique that can be usefully employed in order to address many of the various issues sketched above is the noninterference approach to information flow analysis [18]. Its application to security is based on the idea that an undesired interference from a high-security level part of the system to another one at low-security level can be revealed by comparing different system views that are obtained by changing the behavior of the interfering high-security level components (see, e.g., [29, 16, 27]). While the original definition of noninterference is not component oriented, several approaches

in the literature have subsequently proposed a compositional treatment of it (see, e.g., [23, 13, 28, 19, 32, 14, 26, 8, 7]). In this paper, we extend previous work by introducing a component-oriented formulation of noninterference that enables compositional security verification driven by system topology.

More precisely, our approach to noninterference analysis requires a formal framework that can describe software applications at the architectural level and, at the same time, is equipped with a formal semantics that maps architectural descriptions into models to which abstraction and equivalence checking can be applied. The reason for resorting to equivalence checking among the various verification techniques is twofold. On the one hand, there are already several works addressing component-oriented verification of functional properties based on equivalence checking (see, e.g., [6, 21, 15, 2]). On the other hand, equivalence checking has been successfully applied to the compositional verification of noninterference (see, e.g., [16, 13, 32, 26]).

All the ingredients mentioned above are provided, e.g., by the process algebraic architectural description language PADL [2], which makes it particularly intuitive to specify the behavior of components, their interactions, and the topology of the entire system. This choice facilitates the identification of the components and of the component behaviors subject to noninterference analysis and provides an ideal setting for the definition of architectural noninterference checks. Moreover, PADL is enriched with a fully automated translation semantics into process algebra that allows us to apply equivalence checking based on behavioral equivalences. Finally, PADL has been demonstrated to be usable in practice for the design of real-world software architectures [4] and has been enriched with an automatic translation procedure from architectural descriptions to multithreaded Java programs [10].

Our approach to noninterference analysis is instantiated in the setting of PADL by first introducing a component-oriented formulation of noninterference at the software architecture level. In this way, we simplify the elicitation of the kind of illegal interferences that are under examination during the software architecture design phase. Then, this formulation is used on the semantic model underlying a PADL description in order to define two component-oriented noninterference checks that are inspired by [2]. The first check verifies noninterference on a basic topological format that we call star, because it is formed by a central component and a number of other components that can communicate only with the central one. The second check assesses noninterference on a different basic topological format that constitutes a cycle of components. For each of the two basic topological formats,

we provide sufficient conditions that allow us to infer noninterference for the entire format from properties verified locally at some components of the format itself. Moreover, we establish further sufficient conditions under which the noninterference checks scale from individual basic topological formats to arbitrary topologies.

The paper is organized as follows. Sect. 2 is devoted to the PADL background. In Sect. 3, we reformulate noninterference at the software architecture level. In Sect. 4, we present the two component-oriented noninterference checks and the various sufficient conditions mentioned above. In all the three sections, a simplified version of the NRL Pump multilevel security routing system is employed as a running example. Finally, in Sect. 5 we comment on related and future work.

2. The Architectural Description Language PADL

PADL [2, 10, 4] is a process algebraic architectural description language accompanied by the software tool TwoTowers [9]. In this section, we start with some notions of process algebra and then we recall the basics of the syntax, the semantics, and the architectural checks for PADL.

2.1. Process Algebra

Process algebra (see, e.g., [30]) provides a set of operators by means of which the behavior of a system can be described in an action-based, compositional way. Given a set *Name* of action names including τ for invisible actions, we consider a process algebra PA with the process term syntax shown in Table 1.

P	::=	$\underline{0}$	inactive process	
	|	B	process constant	$(B \triangleq P)$
	|	$a\,.\,P$	action prefix	$(a \in Name)$
	|	$P + P$	alternative composition	
	|	$P \parallel_S P$	CSP parallel composition	$(S \subseteq Name - \{\tau\})$
	|	$P \,/\, H$	hiding	$(H \subseteq Name - \{\tau\})$
	|	$P \setminus L$	restriction	$(L \subseteq Name - \{\tau\})$
	|	$P\,[\varphi]$	relabeling	$(\varphi : Name \to Name,$ $\varphi^{-1}(\tau) = \{\tau\})$

Table 1: Syntax of PA

$$(\textsc{Pre})\quad \frac{}{a\,.\,P \xrightarrow{a} P}$$

$$(\textsc{Alt}_1)\quad \frac{P_1 \xrightarrow{a} P_1'}{P_1 + P_2 \xrightarrow{a} P_1'} \qquad (\textsc{Alt}_2)\quad \frac{P_2 \xrightarrow{a} P_2'}{P_1 + P_2 \xrightarrow{a} P_2'}$$

$$(\textsc{Par}_1)\quad \frac{P_1 \xrightarrow{a} P_1' \qquad a \notin S}{P_1 \parallel_S P_2 \xrightarrow{a} P_1' \parallel_S P_2} \qquad (\textsc{Par}_2)\quad \frac{P_2 \xrightarrow{a} P_2' \qquad a \notin S}{P_1 \parallel_S P_2 \xrightarrow{a} P_1 \parallel_S P_2'}$$

$$(\textsc{Syn})\quad \frac{P_1 \xrightarrow{a} P_1' \qquad P_2 \xrightarrow{a} P_2' \qquad a \in S}{P_1 \parallel_S P_2 \xrightarrow{a} P_1' \parallel_S P_2'}$$

$$(\textsc{Hid}_1)\quad \frac{P \xrightarrow{a} P' \qquad a \in H}{P \,/\, H \xrightarrow{\tau} P' \,/\, H} \qquad (\textsc{Hid}_2)\quad \frac{P \xrightarrow{a} P' \qquad a \notin H}{P \,/\, H \xrightarrow{a} P' \,/\, H}$$

$$(\textsc{Res})\quad \frac{P \xrightarrow{a} P' \qquad a \notin L}{P \setminus L \xrightarrow{a} P' \setminus L} \qquad (\textsc{Rel})\quad \frac{P \xrightarrow{a} P'}{P\,[\varphi] \xrightarrow{\varphi(a)} P'\,[\varphi]}$$

$$(\textsc{Rec})\quad \frac{B \triangleq P \qquad P \xrightarrow{a} P'}{B \xrightarrow{a} P'}$$

Table 2: Operational semantics rules for PA

The semantics for PA is formalized through a labeled transition system. This is a graph $(\mathbb{P}, \mathit{Name}, \longrightarrow)$ including all computations and branching points, where: $\mathbb{P}$ is the set of vertices, each denoting a state corresponding to a closed and guarded process term; Name is the set of edge labels, each corresponding to an action; $\longrightarrow \subseteq \mathbb{P} \times \mathit{Name} \times \mathbb{P}$ is the set of edges, forming a state transition relation. Each labeled transition $(P, a, P') \in \longrightarrow$ is represented as $P \xrightarrow{a} P'$ to emphasize its source and target states and the action that determines the corresponding state change.

The labeled transition system above is built by inferring one single transition at a time through the application of operational semantics rules to the source state of the transition itself, with the rules being defined by induction on the syntactical structure of process terms. More precisely, the transition relation $\longrightarrow$ is the smallest subset of $\mathbb{P} \times \mathit{Name} \times \mathbb{P}$ satisfying the opera-

tional semantics rules of Table 2. The labeled transition system for a specific process term $P \in \mathbb{P}$ is denoted by $[\![P]\!]$ and has P as initial state.

Each of the operational semantics rules of Table 2 is formed by a premise (above the horizontal line) and a conclusion (below the horizontal line) and establishes which actions can be performed and when they can be performed for a specific behavioral operator. The action prefix operator and the alternative composition operator are called dynamic operators, as they disappear in the conclusions of their rules when moving from the left-hand side to the right-hand side. By contrast, the parallel composition operator, the hiding operator, the restriction operator, and the relabeling operator are called static operators, as they occur on both sides of the conclusions of their rules.

Process terms are compared and manipulated by means of behavioral equivalences. Among the various approaches, for PA we consider weak bisimilarity $\approx_{\mathrm{B}}$, according to which two process terms are equivalent if they are able to mimic each other's visible behavior stepwise [30].

2.2. PADL Textual and Graphical Notations

A PADL description represents an architectural type, which is a family of software systems sharing certain constraints on the observable behavior of their components as well as on their topology. As shown in Table 3, the textual description of an architectural type in PADL starts with its name and its formal parameters (initialized with default values), then comprises an architectural behavior section and an architectural topology section.

The first section defines the overall behavior of the system family by means of types of software components and connectors, which are collectively called architectural element types. The definition of an AET, which starts with its name and its formal parameters, consists of the specification of its behavior and of its interactions.

The behavior of an AET has to be provided in the form of a sequence of behavioral equations written in a verbose variant of PA allowing only for the inactive process (rendered as `stop`), the action prefix operator supporting possible boolean guards and value passing, the alternative composition operator (rendered as `choice`), and recursion.

The interactions of an AET are actions occurring in the process algebraic specification of the behavior of the AET that act as interfaces for the AET itself, while all the other actions are assumed to represent internal activities. Each interaction has to be equipped with three qualifiers, with the first qualifier establishing whether the interaction is an input or output interaction.

ARCHI_TYPE	◁*name and initialized formal parameters*▷
ARCHI_BEHAVIOR	
⋮	⋮
ARCHI_ELEM_TYPE	◁*AET name and formal parameters*▷
BEHAVIOR	◁*sequence of process algebraic equations built from stop, action prefix, choice, and recursion*▷
INPUT_INTERACTIONS	◁*input synchronous/semi-synchronous/asynchronous uni/and/or-interactions*▷
OUTPUT_INTERACTIONS	◁*output synchronous/semi-synchronous/asynchronous uni/and/or-interactions*▷
⋮	⋮
ARCHI_TOPOLOGY	
ARCHI_ELEM_INSTANCES	◁*AEI names and actual parameters*▷
ARCHI_INTERACTIONS	◁*architecture-level AEI interactions*▷
ARCHI_ATTACHMENTS	◁*attachments between AEI local interactions*▷
END	

Table 3: Structure of a PADL textual description

The second qualifier represents the synchronicity of the communications in which the interaction can be involved. We distinguish among: synchronous interactions, which are blocking (qualifier `SYNC`); semi-synchronous interactions, which cause no blocking as they raise an exception if prevented (qualifier `SSYNC`); asynchronous interactions, which are completely decoupled from the other parties involved in the communication (qualifier `ASYNC`). Every semi-synchronous interaction is implicitly equipped with a boolean variable usable in the architectural description, which is automatically set to true if the interaction can be executed, false if an exception is raised.

The third qualifier describes the multiplicity of the communications in which the interaction can be involved. We distinguish among: uni-interactions, which are mainly involved in one-to-one communications (qualifier `UNI`); and-interactions, which guide inclusive one-to-many communications (qualifier `AND`); or-interactions, which guide selective one-to-many communications (qualifier `OR`).

The second section of a PADL description defines the topology of the system family. This is accomplished in three steps. Firstly, we have the declaration of the instances of the AETs – called AEIs – which represent

the actual system components and connectors, together with their actual parameters. Secondly, we have the declaration of the architectural (as opposed to local) interactions, which are some of the interactions of the AEIs that act as interfaces for the whole systems of the family. Thirdly, we have the declaration of the architectural attachments among the local interactions of the AEIs, which make the AEIs communicate with each other. An attachment is admissible only if it goes from an output interaction of an AEI to an input interaction of another AEI. Moreover, a uni-interaction can be attached only to one interaction, whereas an and/or-interaction can be attached only to uni-interactions. These restrictions forbid the definition of ambiguous modeling patterns and are not in any relation with the results of this paper.

Besides the textual notation, PADL comes equipped with a graphical notation that is an extension of the flow graph notation [30]. As will be shown in Figs. 1 to 4, in an enriched flow graph AEIs are depicted as boxes, local interactions are depicted as small black circles on the box border, and attachments are depicted as directed edges between pairs each composed of a local output interaction and a local input interaction. The small circle of an interaction is extended inside the AEI box with an arc (resp. a buffer) if the interaction is semi-synchronous (resp. asynchronous). Likewise, the small circle of an interaction is extended outside the AEI box with a triangle (resp. a bisected triangle) if the interaction is an and-interaction (resp. an or-interaction).

Example 2.1. We illustrate PADL through a multilevel security routing system that will be used as a running example throughout the paper. This system can be viewed as an abstraction of the NRL Pump secure routing system [25]. Here, we consider a model with two access clearance levels, high and low, and users playing two different roles, sender and receiver. The communication between these users is controlled by a router that regulates the exchange of messages among senders and receivers on the basis of their level. We assume that there is one high (resp. low) sender and one high (resp. low) receiver. Here is the architectural description header:

```
ARCHI_TYPE ML_Sec_Routing(void)
```

The system comprises four AETs: the sender, the buffer, the router, and the receiver.

The sender AET, which repeatedly sends messages, is as follows:

```
ARCHI_ELEM_TYPE Sender_Type(void)
  BEHAVIOR
    Sender(void; void) =
      send . Sender()
  INPUT_INTERACTIONS  void
  OUTPUT_INTERACTIONS SYNC UNI send
```

while the receiver AET, which is waiting for incoming messages, is as follows:

```
ARCHI_ELEM_TYPE Receiver_Type(void)
  BEHAVIOR
    Receiver(void; void) =
      receive . Receiver()
  INPUT_INTERACTIONS  SYNC UNI receive
  OUTPUT_INTERACTIONS void
```

The routing system is made of two one-position buffers – one for each level – and a shared router. The buffer AET is as follows:

```
ARCHI_ELEM_TYPE Buffer_Type(void)
  BEHAVIOR
    Buffer(void; void) =
      deposit . withdraw . Buffer()
  INPUT_INTERACTIONS  SYNC UNI deposit
  OUTPUT_INTERACTIONS SYNC UNI withdraw
```

The router accepts messages arriving from senders and, after some internal computation, asynchronously transmits them to receivers of the corresponding level. The router AET is as follows:

```
ARCHI_ELEM_TYPE Router_Type(void)
  BEHAVIOR
    Router(void; void) =
      choice
      {
        get_high . process_high . trans_high . Router(),
        get_low . process_low . trans_low . Router()
      }
  INPUT_INTERACTIONS  SYNC  UNI get_high; get_low
  OUTPUT_INTERACTIONS ASYNC UNI trans_high; trans_low
```

Finally, the architectural topology section, which is illustrated by the enriched flow graph of Fig. 1, is as follows:

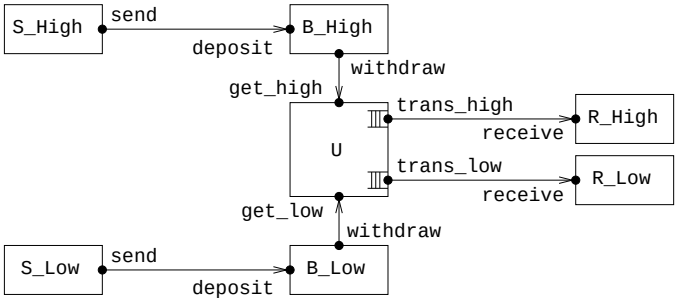

Figure 1: Enriched flow graph of the multilevel security routing system

```
ARCHI_ELEM_INSTANCES
  S_High : Sender_Type();
  S_Low  : Sender_Type();
  B_High : Buffer_Type();
  B_Low  : Buffer_Type();
  U      : Router_Type();
  R_High : Receiver_Type();
  R_Low  : Receiver_Type()
ARCHI_INTERACTIONS
  void
ARCHI_ATTACHMENTS
  FROM S_High.send     TO B_High.deposit;
  FROM S_Low.send      TO B_Low.deposit;
  FROM B_High.withdraw TO U.get_high;
  FROM B_Low.withdraw  TO U.get_low;
  FROM U.trans_high    TO R_High.receive;
  FROM U.trans_low     TO R_Low.receive
```

■

2.3. The Semantics for PADL

The semantics of a PADL description is a two-step translation into a process term of PA. In the first step, the semantics of each AEI is defined to be the behavior of the corresponding AET with: all the action occurrences being preceded by the name of the AEI; the AET formal data parameters being substituted for by the corresponding AEI actual data parameters.

Let $\mathcal{C}$ be an AET with $m \in \mathbb{N}_{\geq 0}$ formal data parameters $fp_1, \ldots, fp_m$ and behavior given by a sequence $\mathcal{E}$ of PA equations. Let C be an AEI of type $\mathcal{C}$ with $m \in \mathbb{N}_{\geq 0}$ actual data parameters $ap_1, \ldots, ap_m$ consistent with $fp_1, \ldots, fp_m$ by order and type. The isolated semantics of C is $[\![C]\!] = C.\mathcal{E}\{ap_1 \hookrightarrow fp_1, \ldots, ap_m \hookrightarrow fp_m\}$ where "$C.$" introduces the dot notation – not to be confused with action prefix – and $\hookrightarrow$ is syntactical substitution.

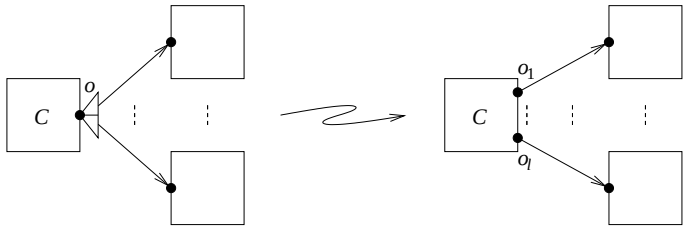

Figure 2: Topological management of local output or-interactions

Example 2.2. In the running example, $[\![$`S_High`$]\!]$ (resp. $[\![$`S_Low`$]\!]$) coincides with the sequence of PA equations of `Sender_Type` where action names are preceded by `S_High` (resp. `S_Low`). We can argue similarly in the case of $[\![$`B_High`$]\!]$ (resp. $[\![$`B_Low`$]\!]$) and $[\![$`R_High`$]\!]$ (resp. $[\![$`R_Low`$]\!]$). As will be shown, $[\![$`U`$]\!]$ requires more attention as it includes local asynchronous uni-interactions. ■

If C contains local or-interactions, then each of them is replaced by as many fresh local uni-interactions as there are attachments involving the considered interaction. This reflects the fact that an or-interaction can result in several alternative communications, as shown in Fig. 2 for an output or-interaction. Formally, this is achieved through the application of a function named *or-rewrite*. In this case, the isolated semantics of C is given by $\mathit{or\text{-}rewrite}(C.\mathcal{E}\{ap_1 \hookrightarrow fp_1, \ldots, ap_m \hookrightarrow fp_m\})$.

If C has local asynchronous interactions, then the decoupling of the beginning and the end of the communications in which these interactions are involved is managed by means of additional implicit AEIs behaving as unbounded buffers, as shown in Fig. 3 for uni-interactions. Each additional implicit input asynchronous queue (IAQ) and output asynchronous queue (OAQ) is of the following type:

```
ARCHI_ELEM_TYPE Async_Queue_Type(void)
  BEHAVIOR
    Queue(int n := 0; void) =
      choice
      {
        cond(true)  -> arrive . Queue(n + 1),
        cond(n > 0) -> depart . Queue(n - 1)
      }
  INPUT_INTERACTIONS  SYNC UNI arrive
  OUTPUT_INTERACTIONS SYNC UNI depart
```

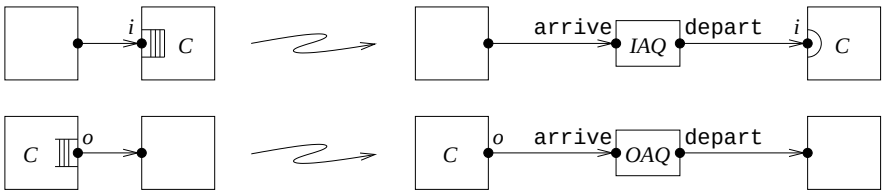

Figure 3: Topological management of local asynchronous uni-interactions

where `arrive` is an always-enabled input synchronous uni-interaction while `depart` is an output synchronous uni-interaction enabled only if the buffer is not empty.

Asynchronous communications are never blocking, as we now illustrate in the case of asynchronous uni-interactions (see Fig. 3). On the one hand, the local input asynchronous interaction i of C is converted into a semi-synchronous interaction and implicitly reattached to the `depart` interaction of the corresponding additional implicit IAQ. Note that i becomes semi-synchronous because the communication between the `depart` interaction and i must not block C whenever the buffer is empty. On the other hand, the local output asynchronous interaction o of C is never blocked because it is implicitly converted into a synchronous interaction and reattached to the always-enabled `arrive` interaction of the corresponding additional implicit OAQ. By contrast, the `depart` interaction of this additional implicit OAQ is attached to the input interaction originally attached to o.

The isolated semantics of C is obtained from the parallel composition of $\mathit{or\text{-}rewrite}(C.\mathcal{E}\{ap_1 \hookrightarrow fp_1, \ldots, ap_m \hookrightarrow fp_m\})\,[\varphi_{C,\mathrm{async}}]$ with the behavior of each IAQ and OAQ that is needed. The relabeling function $\varphi_{C,\mathrm{async}}$ transforms the originally asynchronous local interactions of C and the local interactions of the additional implicit AEIs attached to them – which may have names different from each other – into the respective fresh names, so that they can communicate with each other through the PA parallel composition operator. The choice of fresh names is easily achieved by concatenating the original names of all the involved interactions via symbol #, which will be used throughout the paper to denote an attachment between interactions or AEIs.

Example 2.3. In the running example, we have that $[\![\mathtt{U}]\!]$ requires two additional implicit OAQs for the two local output asynchronous uni-interactions `U.trans_high` and `U.trans_low`. As a consequence, assuming that $\mapsto$ de-

notes action relabeling and that the PA parallel composition operator is left associative, we obtain:

$$
\begin{aligned}
[\![\mathtt{U}]\!] \;=\; & \mathtt{Router}[\mathtt{U.trans\_high} \mapsto \mathtt{U.trans\_high\#OAQ\_High.arrive},\\
& \qquad \mathtt{U.trans\_low} \mapsto \mathtt{U.trans\_low\#OAQ\_Low.arrive}]\\
& \quad \|_{\{\mathtt{U.trans\_high\#OAQ\_High.arrive}\}}\\
& \mathtt{OAQ\_High.Queue(0)}[\mathtt{OAQ\_High.arrive} \mapsto \mathtt{U.trans\_high\#OAQ\_High.arrive}]\\
& \quad \|_{\{\mathtt{U.trans\_low\#OAQ\_Low.arrive}\}}\\
& \mathtt{OAQ\_Low.Queue(0)}[\mathtt{OAQ\_Low.arrive} \mapsto \mathtt{U.trans\_low\#OAQ\_Low.arrive}]
\end{aligned}
$$

■

In the second step of the translation, the semantics of the entire architectural description is derived by composing in parallel the semantics of its AEIs according to the declared attachments. This is achieved by transparently exploiting the parallel composition and relabeling operators. Let $\{C_1, \ldots, C_n\}$ be a set of AEIs. Fixed an AEI C_j in the set, let $\mathcal{LI}_{C_j}$ be the set of local interactions of C_j and $\mathcal{LI}_{C_j;C_1,\ldots,C_n} \subseteq \mathcal{LI}_{C_j}$ be the set of local interactions of C_j attached to $\{C_1, \ldots, C_n\}$. Since local or-interactions and local asynchronous interactions have been suitably transformed, here by local interactions of C_j we mean: its original local nonasynchronous uni-/and-interactions; its fresh local nonasynchronous uni-interactions that replace its original local nonasynchronous or-interactions; the local interactions of its additional implicit AEIs that are not attached to its originally asynchronous local interactions.

Similar to Ex. 2.3, in order to make the process terms representing the semantics of these AEIs communicate in the presence of attached interactions having different names, we need sets of fresh action names. For instance, $C_j.o\#C_g.i$ is the fresh action name for the case in which the local output uni-interaction o of C_j is attached to the local input uni-interaction i of C_g. Formally, we need suitable injective relabeling functions $\varphi_{C_j;C_1,\ldots,C_n}$ in such a way that $\varphi_{C_j;C_1,\ldots,C_n}(C_j.a_1) = \varphi_{C_g;C_1,\ldots,C_n}(C_g.a_2)$ if and only if $C_j.a_1$ and $C_g.a_2$ are attached to each other or to the same and-interaction. The interacting semantics of $C_j \in \{C_1, \ldots, C_n\}$ with respect to $\{C_1, \ldots, C_n\}$ is defined as $[\![C_j]\!]_{C_1,\ldots,C_n} = [\![C_j]\!]\,[\varphi_{C_j;C_1,\ldots,C_n}]$. In general, the interacting semantics of $\{C'_1, \ldots, C'_{n'}\} \subseteq \{C_1, \ldots, C_n\}$ with respect to $\{C_1, \ldots, C_n\}$ is the parallel composition of the interacting semantics of the individual AEIs:

$$
\begin{aligned}
[\![C'_1, \ldots, C'_{n'}]\!]_{C_1,\ldots,C_n} \;=\; & [\![C'_1]\!]_{C_1,\ldots,C_n} \|_{\mathcal{S}(C'_1,C'_2;C_1,\ldots,C_n)}\\
& \qquad [\![C'_2]\!]_{C_1,\ldots,C_n} \|_{\mathcal{S}(C'_1,C'_3;C_1,\ldots,C_n) \cup \mathcal{S}(C'_2,C'_3;C_1,\ldots,C_n)} \cdots\\
& \qquad\quad \cdots \|_{\bigcup_{i=1}^{n'-1} \mathcal{S}(C'_i,C'_{n'};C_1,\ldots,C_n)} [\![C'_{n'}]\!]_{C_1,\ldots,C_n}
\end{aligned}
$$

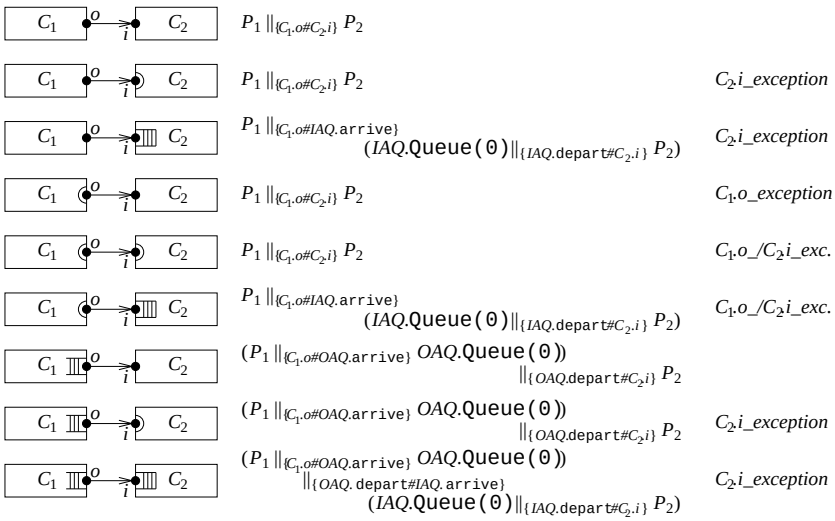

Figure 4: Semantic treatment of the forms of communication synchronicity in PADL

where we have that: $\mathcal{S}(C'_j; C_1, \ldots, C_n) = \varphi_{C'_j;C_1,\ldots,C_n}(\mathcal{LI}_{C'_j;C_1,\ldots,C_n})$ is the synchronization set of C'_j with respect to $\{C_1, \ldots, C_n\}$; $\mathcal{S}(C'_j, C'_g; C_1, \ldots, C_n) = \mathcal{S}(C'_j; C_1, \ldots, C_n) \cap \mathcal{S}(C'_g; C_1, \ldots, C_n)$ is the pairwise synchronization set of C'_j and C'_g with respect to $\{C_1, \ldots, C_n\}$; the unions of pairwise synchronization sets are consistent with the left associativity of the parallel composition operator.

In Fig. 4, we summarize the semantic treatment of the nine forms of communications resulting from the attachment of a local output synchronous, semi-synchronous, or asynchronous interaction o of an AEI C_1 – whose interacting semantics is process term P_1 – to a local input synchronous, semi-synchronous, or asynchronous interaction i of an AEI C_2 – whose interacting semantics is process term P_2.

Finally, the semantics of an architectural description $\mathcal{A}$ formed by the set of AEIs $\{C_1, \ldots, C_n\}$ is defined as $[\![\mathcal{A}]\!] = [\![C_1, \ldots, C_n]\!]_{C_1,\ldots,C_n}$.

Example 2.4. On the basis of the isolated semantics of the individual AEIs discussed in Exs. 2.2 and 2.3, the semantics of the entire PADL description of the running example, i.e., $[\![\texttt{ML\_Sec\_Routing}]\!]$, is given by the process term of Table 4. ■

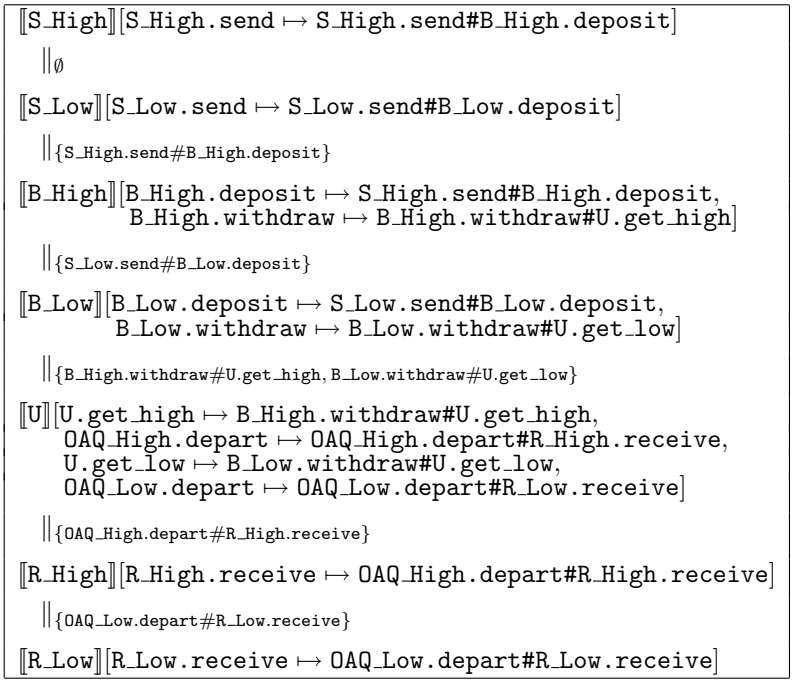

Table 4: Semantics of the PADL description of the multilevel security routing system

2.4. Architectural Checks for PADL

PADL is equipped with techniques for inferring architectural properties like correct component coordination from the properties of the individual AEIs. The idea is to verify the absence of coordination mismatches resulting in property violations through a topological reduction process based on equivalence checking. Given an architectural description, the starting point is constituted by an abstract variant of its enriched flow graph, where vertices correspond to AEIs and two vertices are linked by an edge if and only if attachments have been declared among their interactions. From a topological viewpoint, the resulting graph is a combination of possibly intersecting stars and cycles, which are thus viewed as basic topological formats.

The strategy then consists of applying specific checks locally to all stars and cycles occurring in the abstract graph. Each check verifies whether the star/cycle contains an AEI behaviorally equivalent to the whole star/cycle, in which case the star/cycle can be replaced by that AEI. The process suc-

cessfully terminates when the whole graph has been reduced to a single behaviorally equivalent AEI, as at that point it is sufficient to verify whether that AEI satisfies the given property or not. In case of failure, the mentioned checks provide diagnostic information useful to pinpoint components responsible for possible property violations within a single star/cycle.

Given a property $\mathcal{P}$ belonging to a class Ψ of properties of interest, in order to be applicable the strategy requires the existence of a behavioral equivalence $\approx_{\mathcal{P}}$ that possesses the following characteristics. Firstly, the equivalence must preserve $\mathcal{P}$, which is fundamental for enabling the topological reduction process. Secondly, it must be a congruence with respect to static operators, thus allowing the topological reduction process to be applied to single portions of the topology of an architectural description – which is more efficient than considering the entire topology at once – without affecting the possible validity of $\mathcal{P}$. Thirdly, it must be able to abstract from internal actions, as an architectural property is typically expressed in terms of the possibility/necessity of executing local interactions in a certain order.

Note that, if $\approx_{\mathcal{P}}$ has a modal logic characterization, then it is immediate to produce diagnostic information in case of failure of the topological reduction process. For instance, if the considered property is deadlock freedom, then an action-based modal or temporal logic like the Hennessy-Milner logic [20] and a behavioral equivalence like $\approx_{\mathrm{B}}$ are good candidates.

Before applying the check to a star/cycle made of the AEIs $C_1, \ldots, C_n$, for each AEI C_j in the set we have to hide all of its actions that do not model interactions within the star/cycle (see, e.g., the internal actions of the AEI `U` in the running example), because they cannot result in mismatches within the star/cycle, but may hamper the topological reduction process if left visible. Formally, the only actions that remain observable are those in $\mathcal{LI}_{C_j;C_1,\ldots,C_n}$ and those in $\mathcal{OALI}_{C_j}$, which contains the originally asynchronous local interactions of C_j together with the local interactions of the related additional implicit AEIs to which they have been reattached.

In order to set the visibility of each action as needed, we define a partially closed variant and a totally closed variant of the interacting semantics. For this purpose, we first introduce the interacting semantics of C_j with respect to $\{C_1, \ldots, C_n\}$ without buffers for its originally asynchronous local interactions, denoted by $[\![C_j]\!]^{\mathrm{wob}}_{C_1,\ldots,C_n}$. Then, we indicate with $[\![C_j]\!]^{\#C''_1,\ldots,C''_{n''}}_{C_1,\ldots,C_n}$ the variant of $[\![C_j]\!]^{\mathrm{wob}}_{C_1,\ldots,C_n}$ including the buffers for the originally asynchronous local interactions of C_j attached to $\{C''_1, \ldots, C''_{n''}\}$.

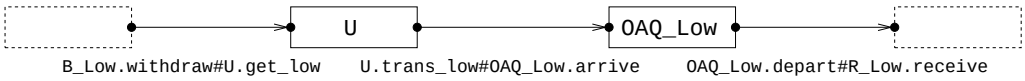

Figure 5: Graphical representation of the semantics of the AEI U from Ex. 2.7

Definition 2.5. The partially closed interacting semantics of an AEI $C_j \in \{C_1, \ldots, C_n\}$ with respect to $\{C_1, \ldots, C_n\}$ including its buffers attached to $\{C''_1, \ldots, C''_{n''}\}$ is:

$$[\![C_j]\!]^{\mathrm{pc};\#C''_1,\ldots,C''_{n''}}_{C_1,\ldots,C_n} = [\![C_j]\!]^{\#C''_1,\ldots,C''_{n''}}_{C_1,\ldots,C_n} / (Name - \mathcal{V}_{C_j;C_1,\ldots,C_n})$$

where $\mathcal{V}_{C_j;C_1,\ldots,C_n} = \varphi_{C_j;C_1,\ldots,C_n}(\mathcal{LI}_{C_j;C_1,\ldots,C_n}) \cup \varphi_{C_j,\mathrm{async}}(\mathcal{OALI}_{C_j})$ and we write $[\![C_j]\!]^{\mathrm{pc;wob}}_{C_1,\ldots,C_n}$ if $n'' = 0$. ■

Definition 2.6. The totally closed interacting semantics of an AEI $C_j \in \{C_1, \ldots, C_n\}$ with respect to $\{C_1, \ldots, C_n\}$ including its buffers attached to $\{C''_1, \ldots, C''_{n''}\}$ is:

$$[\![C_j]\!]^{\mathrm{tc};\#C''_1,\ldots,C''_{n''}}_{C_1,\ldots,C_n} = [\![C_j]\!]^{\mathrm{pc};\#C''_1,\ldots,C''_{n''}}_{C_1,\ldots,C_n} / \varphi_{C_j,\mathrm{async}}(\mathcal{OALI}_{C_j})$$

and we write $[\![C_j]\!]^{\mathrm{tc;wob}}_{C_1,\ldots,C_n}$ if $n'' = 0$. ■

Example 2.7. As illustrated in Fig. 5, the partially closed interacting semantics of the AEI U with respect to {U, B_Low, R_Low} including its buffer attached to R_Low is:

$$[\![\mathtt{U}]\!]^{\mathrm{pc};\#\mathtt{R\_Low}}_{\mathtt{U},\mathtt{B\_Low},\mathtt{R\_Low}} = [\![\mathtt{U}]\!]^{\#\mathtt{R\_Low}}_{\mathtt{U},\mathtt{B\_Low},\mathtt{R\_Low}} / (Name - \{\mathtt{B\_Low.withdraw\#U.get\_low}, \mathtt{U.trans\_low\#OAQ\_Low.arrive}, \mathtt{OAQ\_Low.depart\#R\_Low.receive}\})$$

Note that this semantics does not include the buffer of U attached to R_High and hides all the interactions among U and the AEIs managing high messages. The totally closed version is obtained by hiding U_trans_low#OAQ_Low.arrive, which expresses the interaction between U and its additional implicit output asynchronous queue OAQ_Low. ■

The partially (resp. totally) closed interacting semantics of $\{C'_1, \ldots, C'_{n'}\} \subseteq \{C_1, \ldots, C_n\}$ with respect to $\{C_1, \ldots, C_n\}$ including their buffers attached to $\{C''_1, \ldots, C''_{n''}\}$ is defined as the parallel composition of the interacting semantics of the individual AEIs. The variant totally closed up to $\{C'''_1, ..., C'''_{n'''}\} \subset \{C'_1, \ldots, C'_{n'}\}$, where $[\![C'''_j]\!]^{\mathrm{pc};\#C''_1,\ldots,C''_{n''}}_{C_1,\ldots,C_n}$ is used in place of $[\![C'''_j]\!]^{\mathrm{tc};\#C''_1,\ldots,C''_{n''}}_{C_1,\ldots,C_n}$, is denoted by $[\![C'_1, \ldots, C'_{n'}]\!]^{\mathrm{tc};\#C''_1,\ldots,C''_{n''};C'''_1,\ldots,C'''_{n'''}}_{C_1,\ldots,C_n}$.

2.4.1. Architectural Compatibility for Stars

A star is a portion of the abstract enriched flow graph of an architectural description, which is not part of a cyclic subgraph. It is formed by a central

AEI K and a border $\mathcal{B}_K = \{C_1, \ldots, C_n\}$ including all the AEIs attached to K. The validity of an architectural property over a star can be investigated by analyzing the interplay between the central AEI K and each of the AEIs in the border, as there cannot be attachments among AEIs in the border. In order to achieve a correct coordination between K and $C_j \in \mathcal{B}_K$, the actual observable behavior of C_j should coincide with the observable behavior expected by K. In other words, the observable behavior of K should not be altered by the insertion of C_j into the border of the star.

Definition 2.8. Given an architectural description $\mathcal{A}$ and a property $\mathcal{P} \in \Psi$, let K be the central AEI of a star of $\mathcal{A}$, $\mathcal{B}_K = \{C_1, \ldots, C_n\}$ be the border of the star, C_j be an AEI in $\mathcal{B}_K$, H_{K,C_j} be the set of interactions of additional implicit AEIs of K that are attached to interactions of C_j, and E_{K,C_j} be the set of exceptions that may be raised by semi-synchronous interactions involved in attachments between K and C_j. We say that K is $\mathcal{P}$-compatible with C_j iff:

$$(\llbracket K \rrbracket_{\mathcal{A}}^{\mathrm{pc};\#C_j} \parallel_{\mathcal{S}(K,C_j;\mathcal{A})} \llbracket C_j \rrbracket_{K,\mathcal{B}_K}^{\mathrm{tc};\#K}) \,/\, (H_{K,C_j} \cup E_{K,C_j}) \;\approx_{\mathcal{P}}\; \llbracket K \rrbracket_{\mathcal{A}}^{\mathrm{pc;wob}}$$ ∎

Intuitively, the $\mathcal{P}$-compatibility of K with respect to C_j ensures that the behavior of C_j does not limit the behavior of K as observed in isolation. If this condition holds for each AEI $C_j \in \mathcal{B}_K$, then we derive the following result concerning the preservation of any property $\mathcal{P} \in \Psi$ satisfied by the central AEI K.

Proposition 2.9. Given an architectural description $\mathcal{A}$ and a property $\mathcal{P} \in \Psi$, let K be the central AEI of a star of $\mathcal{A}$, $\mathcal{B}_K = \{C_1, \ldots, C_n\}$ be the border of the star, and H_{K,C_j} and E_{K,C_j} be the same sets as Def. 2.8. Whenever K is $\mathcal{P}$-compatible with every $C_j \in \mathcal{B}_K$, then:

$$\llbracket K, \mathcal{B}_K \rrbracket_{K,\mathcal{B}_K}^{\mathrm{tc};\#K,\mathcal{B}_K;K} \,/\, \bigcup_{j=1}^{n} (H_{K,C_j} \cup E_{K,C_j}) \;\approx_{\mathcal{P}}\; \llbracket K \rrbracket_{\mathcal{A}}^{\mathrm{pc;wob}}$$

hence $\llbracket K, \mathcal{B}_K \rrbracket_{K,\mathcal{B}_K}^{\mathrm{tc};\#K,\mathcal{B}_K;K} \,/\, \bigcup_{j=1}^{n} (H_{K,C_j} \cup E_{K,C_j})$ satisfies $\mathcal{P}$ iff so does $\llbracket K \rrbracket_{\mathcal{A}}^{\mathrm{pc;wob}}$. ∎

It is worth noting that the use of partially/totally closed semantics and abstractions simplifies the structure of process terms without compromising the capability of revealing coordination mismatches.

Example 2.10. Let $\mathcal{A}$ be the architectural description of the running example and $\mathcal{P}$ be deadlock freedom. Consider the star with central AEI `U`

and border formed by the AEIs B_Low, B_High, R_Low, R_High. Then U is $\mathcal{P}$-compatible with R_Low, because:

$$(\llbracket \mathtt{U} \rrbracket_{\mathcal{A}}^{\mathrm{pc};\#\mathtt{R\_Low}} \parallel_{\mathcal{S}(\mathtt{U},\mathtt{R\_Low};\mathcal{A})} \llbracket \mathtt{R\_Low} \rrbracket_{\mathtt{U},\mathcal{B}_{\mathtt{U}}}^{\mathrm{tc};\#\mathtt{U}}) \,/\, (H_{\mathtt{U},\mathtt{R\_Low}} \cup E_{\mathtt{U},\mathtt{R\_Low}}) \approx_{\mathcal{P}} \llbracket \mathtt{U} \rrbracket_{\mathcal{A}}^{\mathrm{pc;wob}}$$

Since U is also $\mathcal{P}$-compatible with B_Low, B_High, and R_High, we derive that the considered star is deadlock free because so is U. ∎

2.4.2. Architectural Interoperability for Cycles

The architectural compatibility check is not enough in the presence of cycles. A cycle is a closed simple path in the abstract enriched flow graph of an architectural description, which traverses a set $\mathcal{Y} = \{C_1, \ldots, C_n\}$ of $n \geq 3$ AEIs. The validity of an architectural property over a cycle cannot be investigated by analyzing the interplay between pairs of AEIs, because of the possible presence of arbitrary interferences among the various AEIs in the cycle. In order to achieve a correct coordination inside the cycle, the actual observable behavior of any AEI C_j in the cycle should coincide with the observable behavior expected by the rest of the cycle. In other words, the observable behavior of C_j should not be altered by the insertion of C_j itself into the cycle.

The architectural interoperability is defined as follows when considering sets of adjacent AEIs in a cycle. For symmetry reasons, the size of each such set can be limited to half of the number of AEIs traversed by the cycle.

Definition 2.11. Given an architectural description $\mathcal{A}$ and a property $\mathcal{P} \in \Psi$, let $\mathcal{Y} = \{C_1, \ldots, C_n\}$ be the set of AEIs traversed by a cycle of $\mathcal{A}$, $\mathcal{J} = \{C'_1, \ldots, C'_l\}$ be a set of $1 \leq l \leq n/2$ adjacent AEIs in the cycle, $\mathcal{T} = \mathcal{Y} - \mathcal{J}$ be the set of the other AEIs in the cycle, $H_{C'_j,\mathcal{T}}$ be the set of interactions of additional implicit AEIs of $C'_j \in \mathcal{J}$ that are attached to $\mathcal{T}$, and $E_{C'_j,\mathcal{T}}$ be the set of exceptions that may be raised by semi-synchronous interactions involved in attachments between $C'_j \in \mathcal{J}$ and $\mathcal{T}$. We say that $\mathcal{J}$ $\mathcal{P}$-interoperates with the other AEIs in the cycle iff:

$$\llbracket \mathcal{Y} \rrbracket_{\mathcal{A}}^{\mathrm{tc};\#\mathcal{Y};\mathcal{J}} \,/\, (Name - \bigcup_{j=1}^{l} \mathcal{V}_{C'_j;\mathcal{A}}) \,/\, \bigcup_{j=1}^{l} (H_{C'_j,\mathcal{T}} \cup E_{C'_j,\mathcal{T}}) \approx_{\mathcal{P}} \llbracket \mathcal{J} \rrbracket_{\mathcal{A}}^{\mathrm{pc};\#\mathcal{J}}$$ ∎

Proposition 2.12. Given an architectural description $\mathcal{A}$ and a property $\mathcal{P} \in \Psi$, let $\mathcal{Y} = \{C_1, \ldots, C_n\}$ be the set of AEIs traversed by a cycle of $\mathcal{A}$. Whenever there exists $\mathcal{J} = \{C'_1, \ldots, C'_l\} \subseteq \mathcal{Y}$, $1 \leq l \leq n/2$, that $\mathcal{P}$-interoperates with the other AEIs in the cycle, then $\llbracket \mathcal{Y} \rrbracket_{\mathcal{A}}^{\mathrm{tc};\#\mathcal{Y};\mathcal{J}} \,/\, (Name -$

$\bigcup_{j=1}^{l} \mathcal{V}_{C'_j;\mathcal{A}}) \,/\, \bigcup_{j=1}^{l} (H_{C'_j,\mathcal{T}} \cup E_{C'_j,\mathcal{T}})$ satisfies $\mathcal{P}$ iff so does $[\![\mathcal{J}]\!]_{\mathcal{A}}^{\mathrm{pc};\#\mathcal{J}}$, where $\mathcal{T}$, $H_{C'_j,\mathcal{T}}$, and $E_{C'_j,\mathcal{T}}$ are the same sets as Def. 2.11. ■

2.4.3. Architectural Checks for Arbitrary Topologies

In the case of an arbitrary topology, compatibility and interoperability checks are applied several times in a way that hopefully converges towards the reduction of the entire topology to a single architectural element, which is finally checked against the architectural property of interest. A prominent role is played by AEIs belonging to the intersection of cycles with acyclic portions of the topology or other cycles, where acyclic portions are intended not to be in cyclic subgraphs.

Formally, let $\mathcal{A}$ be an architectural description and $\{C_1, \ldots, C_n\}$ be a set of AEIs of $\mathcal{A}$. The frontier of $\{C_1, \ldots, C_n\}$ is $\mathcal{F}_{C_1,\ldots,C_n} = \{C_j \in \{C_1, \ldots, C_n\} \mid \mathcal{LI}_{C_j;C_1,\ldots,C_n} \neq \mathcal{LI}_{C_j}\}$. Moreover, we denote with $\mathcal{CU}_{C_j}$ the cyclic union of C_j, which is the union of the sets of AEIs traversed by a cycle that traverses also C_j. The cycles in the abstract enriched flow graph of $\mathcal{A}$ are managed by means of a cycle covering algorithm, which takes all the AEIs belonging to at least one cycle and groups them to form a set $\mathcal{CU}$ of cyclic unions. The cycle covering process is such that any two cyclic unions in $\mathcal{CU}$ are connected at most through a single shared AEI or through the attachments between a single AEI of one cyclic union and a single AEI of the other cyclic union. The set $\mathcal{CU}$ is said to be total iff the topology becomes acyclic after replacing every cyclic union $\mathcal{Y} = \{C_1, \ldots, C_n\} \in \mathcal{CU}$ with an AEI whose behavior is given by:

$$[\![\mathcal{Y}]\!]_{\mathcal{A}}^{\mathrm{tc};\#\mathcal{Y};\mathcal{F}_{C_1,\ldots,C_n}} \,/\, (Name - \bigcup_{C_j \in \mathcal{F}_{C_1,\ldots,C_n}} \mathcal{V}_{C_j;\mathcal{A}}) \,/\, \bigcup_{C_j \in \mathcal{F}_{C_1,\ldots,C_n}} (H_{C_j,\mathcal{Y}} \cup E_{C_j,\mathcal{Y}}).$$

Hence, the objective is turning an arbitrary topology into an acyclic topology through these substitutions that take into account the frontier of each original cyclic union.

Given a property $\mathcal{P} \in \Psi$, arbitrary topologies are addressed by combining the sufficient conditions for stars and cycles, as formalized by the theorem below. The importance of AEIs belonging to the frontier of cyclic unions is emphasized by the fact that each of these AEIs must be $\mathcal{P}$-compatible with every AEI attached to it that belongs to an acyclic portion of the topology and, at the same time, must $\mathcal{P}$-interoperate with the other AEIs in the cyclic union to which it belongs. As a consequence, it is convenient to reduce all the cyclic unions before reducing acyclic portions. The above conditions concerning compatibility and interoperability together with the existence of

a total set of cyclic unions enable the topological reduction process. The result is that the satisfaction of $\mathcal{P}$ for the whole topology can be inferred from the satisfaction of $\mathcal{P}$ for a single AEI.

Theorem 2.13. Let $\mathcal{A}$ be an architectural description and $\mathcal{P} \in \Psi$ be a property for which the following two conditions hold:

1. For each $K \in \mathcal{A}$ belonging to an acyclic portion or to the intersection of some cycle with acyclic portions of the abstract enriched flow graph of $\mathcal{A}$, K is $\mathcal{P}$-compatible with every $C \in \mathcal{B}_K - \mathcal{CU}_K$.
2. If $\mathcal{A}$ is cyclic, then there exists a total set $\mathcal{CU}$ of cyclic unions for $\mathcal{A}$ such that for each cyclic union $\{C_1, \ldots, C_n\} \in \mathcal{CU}$:
   (a) If $\mathcal{F}_{C_1,\ldots,C_n} = \emptyset$, then there exists $C_j \in \{C_1, \ldots, C_n\}$ that $\mathcal{P}$-interoperates with the other AEIs in the cyclic union.
   (b) If $\mathcal{F}_{C_1,\ldots,C_n} \neq \emptyset$, then every $C_j \in \mathcal{F}_{C_1,\ldots,C_n}$ $\mathcal{P}$-interoperates with the other AEIs in the cyclic union.
   (c) If no $C_j \in \mathcal{F}_{C_1,\ldots,C_n}$ is such that $[\![C_j]\!]^{\mathrm{pc;wob}}_{\mathcal{A}}$ satisfies $\mathcal{P}$ and there exists $C_g \in \{C_1, \ldots, C_n\} - \mathcal{F}_{C_1,\ldots,C_n}$ such that $[\![C_g]\!]^{\mathrm{pc;wob}}_{\mathcal{A}}$ satisfies $\mathcal{P}$, then at least one such C_g $\mathcal{P}$-interoperates with the other AEIs in the cyclic union.

Then $[\![\mathcal{A}]\!]^{\mathrm{pc};\#\mathcal{A}}$ satisfies $\mathcal{P}$ iff so does $[\![C]\!]^{\mathrm{pc;wob}}_{\mathcal{A}}$ for some $C \in \mathcal{A}$. ■

Example 2.14. The system topology of the running example is acyclic (see Fig. 1). Since each AEI is $\mathcal{P}$-compatible with every AEI attached to it and, e.g., `U` is deadlock free, we can infer that the system is deadlock free without constructing the interacting semantics of the entire PADL description. ■

3. Rephrasing Noninterference at the Architectural Level

In this section, we reformulate noninterference at the architectural level. As we will see, this raises several subtleties in the translation from architectural descriptions to process algebra, especially when managing the nine forms of communications summarized in Fig. 4.

The objective of noninterference analysis [18] is to reveal direct and indirect dependences among components, e.g., in order to study the influence of events caused by nontrusted components (that are added to the system) upon the behavior of components performing security-critical applications. The basic idea behind noninterference relies on the classification of the system activities into two disjoint levels:

- *High* represents the set of (high-level) activities performed by components of which we intend to verify the potential interference.

- *Low* represents the set of (low-level) activities related to the system behavior we intend to monitor.

Then, independent of the specific formalization of the noninterference notion, checking noninterference is actually verifying the indistinguishability of the monitored views of the system that are obtained by changing the behavior of the interfering components.

Several notions of noninterference have been designed to analyze security properties of systems in the formal setting of nondeterministic process algebra (see, e.g., [16, 31, 27]). Without loss of generality, we concentrate on the following noninterference property, which is inspired by strong nondeterministic noninterference [16]. This establishes whether – from the viewpoint of the monitored behavior – the view of the system in which the interfering components are active is the same – according to $\approx_{\mathrm{B}}$ – as that observed in the absence of these components. Formally, a process term P representing the behavior of a system has no illegal information flow if and only if:

$$P \,/\, (Name - Low) \;\approx_{\mathrm{B}}\; P \setminus High \,/\, (Name - Low)$$

In the left-hand term, we use the hiding operator to conceal the part of the system that is not monitored, including the high-level activities performed by the interfering components. In the right-hand term, before hiding the same activities as before, all the high-level activities are prevented from execution by applying to them the restriction operator. A weak behavioral equivalence is needed because the comparison requires the ability of abstracting from certain sets of activities whose observation would invalidate the analysis. In particular, $\approx_{\mathrm{B}}$ is sufficiently expressive to be sensitive to interferences causing, e.g., deadlock and violations of properties that depend on the branching structure of the models.

In the setting of an architectural description language like PADL, the sets *High* and *Low* simply contain local interactions chosen by the designer on the basis of the kind of interference under analysis. These local interactions must then be translated into adequate actions on the basis of the semantics of interacting elements. All the remaining, unclassified activities are simply disregarded and, therefore, hidden. Among them, we include both internal actions and architectural interactions, as they do not contribute to describe communications among components.

For each pair of attached interactions $C_1.o$ and $C_2.i$, we assume that if one of them is declared to be high (resp. low) then the other is set to high (resp. low) too. The reason is that attaching a high interaction to a low interaction would violate the policy that prohibits any direct information flow from high level to low level. For instance, if the aim is to evaluate the impact of a component C_1 on the behavior of a component C_2, then the local interactions of C_1 (and those of C_2 attached to C_1) are declared to be high, while all the remaining local interactions of C_2 are declared to be low.

Moreover, due to the semantics of interacting elements, we recall that the local interactions of every component are subject to relabeling for synchronization purposes, rewriting in the case of or-interactions, and reattachments in the case of asynchronous communications. Hence, at the semantic level we cannot use the sets *High* and *Low* as they are. For instance, an asynchronous low interaction is split at the semantic level into several communications through an additional implicit buffer. Some of these communications keep the low level while some others must not be considered. The intuitive reason is that an asynchronous output is not subject to any interference from the environment, which cannot block its execution. Thus, it is very important to map carefully the high/low classification into the semantics of the composite architectural description. This is achieved transparently as follows. With each AEI K of an architectural description $\mathcal{A}$, we associate the sets $High_K$ and Low_K of its high and low interactions, respectively. The set $High_K$ is defined as the smallest set satisfying the following conditions (Low_K is defined similarly):

- If $K.a \in \mathcal{LI}_K$ is a local nonasynchronous uni-/and-interaction and $K.a \in High$, then $\varphi_{K;\mathcal{A}}(K.a) \in High_K$.
- If $K.a_i \in \mathcal{LI}_K$ is a fresh local nonasynchronous uni-interaction among those replacing the original local nonasynchronous or-interaction $K.a$ and $K.a \in High$, then $\varphi_{K;\mathcal{A}}(K.a_i) \in High_K$.
- If $K.a$ is an originally asynchronous local input interaction and $K.a \in High$, then $\varphi_{K,\mathrm{async}}(K.a) \in High_K$.

We assume $High_{C_1,\ldots,C_n} = \bigcup_{i=1}^{n} High_{C_i}$ and $Low_{C_1,\ldots,C_n} = \bigcup_{i=1}^{n} Low_{C_i}$. Moreover, we denote by $High_{K\#C}$ (resp. $Low_{K\#C}$) the subset of $High_K$ (resp. Low_K) containing the high (resp. low) actions that are obtained from attachments involving K and C.

In order to clarify the classification above, consider the nine attachments reported in Fig. 4 and assume that $C_1.o, C_2.i \in \mathit{High}$. Then, each action of the form $\_\#C_2.i$ is in High_{C_2}, while each action of the form $C_1.o\#\_$ is in High_{C_1} iff $C_1.o$ is not asynchronous. If $C_1.o$ is asynchronous, then $C_1.o\#\mathit{OAQ}.\texttt{arrive}$ and $\mathit{OAQ}.\texttt{depart}\#\_$ are not included in High_{C_1}. Therefore, they cannot represent any form of interference from OAQ back to C_1 as they are simply hidden. The reason is that asynchronous outputs are nonblocking and do not reveal any information flow until the completion of the communication [1]. In other words, the interference goes from the sender to the receiver, while the vice versa does not hold.

After the appropriate classification of local interactions, we can compare the two system views that can be seen by a low-level observer when the interfering activities are enabled/disabled. These views, which are defined as behavioral modifications by employing the static operators for hiding and restriction, are compared according to $\approx_{\mathrm{B}}$.

Definition 3.1. Let $\mathcal{A}$ be an architectural description and $C_1, \ldots, C_n$ be some of its AEIs. Let $\{C_1^h, \ldots, C_g^h\}$ and $\{C_1^l, \ldots, C_j^l\}$ be two subsets of $\{C_1, \ldots, C_n\}$. We say that $\{C_1, \ldots, C_n\}$ is noninterfering with respect to $\mathit{High}_{C_1^h,\ldots,C_g^h}$ and $\mathit{Low}_{C_1^l,\ldots,C_j^l}$ iff:

$$[\![C_1, \ldots, C_n]\!]^{\mathrm{pc};\#C_1,\ldots,C_n}_{C_1,\ldots,C_n} / (\mathit{Name} - \mathit{Low}_{C_1^l,\ldots,C_j^l})$$
$$\approx_{\mathrm{B}}$$
$$[\![C_1, \ldots, C_n]\!]^{\mathrm{pc};\#C_1,\ldots,C_n}_{C_1,\ldots,C_n} \setminus \mathit{High}_{C_1^h,\ldots,C_g^h} / (\mathit{Name} - \mathit{Low}_{C_1^l,\ldots,C_j^l})$$ ■

If the equivalence check based on $\approx_{\mathrm{B}}$ is satisfied, then the proved absence of any information flow ensures transparency of the interfering components $C_1^h, \ldots, C_g^h$ from the viewpoint of the monitored components $C_1^l, \ldots, C_j^l$.

Example 3.2. Let us analyze the running example when the aspect of interest is security against the interference of the high sender on the low receiver. In order to study possible dependences from component `S_High` to component `R_Low`, from the architectural standpoint it is sufficient to assume `S_High.send` to be high and `R_Low.receive` to be low. Then, at the semantics level we check whether `ML_Sec_Routing` is noninterfering with respect to $\mathit{High}_{\texttt{S\_High}}$ and $\mathit{Low}_{\texttt{R\_Low}}$. The result is positive, i.e., the two system views to compare behave the same. Intuitively, the availability to transmit low messages is never compromised, so that the low receiver cannot deduce anything about the behavior of the high sender in spite of the fact that they interact with the same router. ■

4. Component-Oriented Noninterference Check

A noninterference check based on Def. 3.1 does not proceed in a component-oriented manner. Similar to Sect. 2.4, for efficiency reasons the absence of architectural interferences within the description of a software system should be inferred from the properties of its individual architectural elements. Most importantly, under certain conditions, the absence of architectural interferences verified in basic portions of the topology should scale to the whole topology. Since Def. 3.1 is based on a behavioral equivalence, in this section we show how it can be turned into an architectural check like the compatibility and interoperability checks.

Let us start with acyclic topologies. Observed that Def. 3.1 is based on a global notion of noninterference, where the set of components under investigation is considered as a whole, we need a local notion of noninterference that analyzes the interplay between pairs of components. Consider the central AEI K of a star including AEIs that perform high activities. While the noninterference notion of Def. 3.1 establishes the impact of the border of the star, taken as a whole, on the low behavior of K, the local noninterference notion is intended to verify the interference of each AEI in the border of the star on the behavior of K.

Definition 4.1. Given an architectural description $\mathcal{A}$, let K be the central AEI of a star of $\mathcal{A}$ and C_i be an AEI in $\mathcal{B}_K$ performing high activities. We say that C_i does not locally interfere with K iff:

$$[\![K, C_i]\!]^{\mathrm{pc};\#K,C_i}_{K,\mathcal{B}_K} \,/\, High_{K\#C_i} \approx_{\mathrm{B}} [\![K, C_i]\!]^{\mathrm{pc};\#K,C_i}_{K,\mathcal{B}_K} \setminus High_{K\#C_i}$$ ■

Example 4.2. Consider the AEIs `S_High` and `B_High` of the running example. The local interference of `S_High` on `B_High` can be checked by setting to the high level the interaction between them. Then, it is easy to see that `S_High` locally interferes with `B_High`, because intuitively the buffer reaction strictly depends on the sender behavior. ■

Based on the notion of local noninterference, the following proposition provides an efficient and scalable verification of the condition of Def. 3.1 in the case of star-shaped topologies where some AEIs in the border of the central AEI K are high components. In particular, the proposition states sufficient conditions for ensuring that the interactions among K and these high components do not interfere with the low behavior of the star.

Proposition 4.3. Given an architectural description $\mathcal{A}$, let K be the central AEI of a star of $\mathcal{A}$ and $\mathcal{B}_K = \{C_1^h, \ldots, C_g^h, C_1, \ldots, C_n\}$ be the border of the star, such that $High_K = \bigcup_{i=1}^{g} High_{K\#C_i^h}$ and $High_{K\#C_i^h} \cap High_{K\#C_j^h} = \emptyset$ for $i \neq j$. If every C_i^h does not locally interfere with K, then $\{K\} \cup \mathcal{B}_K$ is noninterfering with respect to $High_K$ and $Low_{K,C_1,\ldots,C_n}$. ∎

Note that $High_{K\#C_i^h} \cap High_{K\#C_j^h} \neq \emptyset$ if and only if K has an and-interaction involving C_i^h and C_j^h. If local noninterference is satisfied by each pair of AEIs composed of the central AEI K and one of the high AEIs in the border, then we can infer the absence of interferences in the entire star. This result can be viewed as the counterpart of the compatibility proposition for star-shaped topologies.

For these topologies, local noninterference and compatibility are similar – both are intended to check whether the central AEI of a star safely interacts with its border – but not related in any formal way. However, we now show that the compatibility check can help to conduct component-oriented noninterference analysis. In order to verify whether the border of a star interferes with the central AEI K of the star, it is sufficient to analyze the interacting semantics of K alone, provided that K is $\mathcal{P}_{\mathrm{B}}$-compatible with every AEI in the border. Here, $\mathcal{P}_{\mathrm{B}}$ is any property belonging to the class Ψ that is characterized by $\approx_{\mathrm{B}}$. We thus derive the following sufficient condition for noninterference based on compatibility.

Proposition 4.4. Given an architectural description $\mathcal{A}$, let K be the central AEI of a star of $\mathcal{A}$ and $\mathcal{B}_K = \{C_1, \ldots, C_n\}$ be the border of the star. If K is $\mathcal{P}_{\mathrm{B}}$-compatible with every AEI in $\mathcal{B}_K$, then $\{K\} \cup \mathcal{B}_K$ is noninterfering with respect to $High_K$ and Low_K iff:

$$[\![K]\!]^{\mathrm{pc;wob}}_{K,\mathcal{B}_K} / (Name - Low_K) \approx_{\mathrm{B}} [\![K]\!]^{\mathrm{pc;wob}}_{K,\mathcal{B}_K} \setminus High_K / (Name - Low_K)$$ ∎

Prop. 4.4 provides an alternative characterization of Def. 3.1 for stars that employs compatibility to make the noninterference check local and scalable. In order to make it more flexible, we now extend Prop. 4.4 to work with generalized arbitrary acyclic topologies. In particular, we employ the compatibility based topological reduction process for revealing undesired interferences from an AEI K^h to an AEI K^l. The intuitive idea is that the compatibility check is applied several times to reduce the entire acyclic topology to the path from K^h to K^l, which is unique because the topology is acyclic. Then, an extension of the noninterference check of Prop. 4.4 is applied to this path in order

to establish the absence of any interfering information flow from K^h to K^l. Intuitively, if there exists a prefix of this path that is noninterfering with respect to the high activities of K^h and the interactions with the remaining portion of the path, then no illegal information flow goes from K^h to K^l. This approach is formalized in the following theorem.

Proposition 4.5. Given an acyclic architectural description $\mathcal{A}$, let K^h and K^l be two AEIs of $\mathcal{A}$ connected by a path of $n \geq 0$ AEIs $C_1, \ldots, C_n$ in the abstract enriched flow graph of $\mathcal{A}$. If every AEI of $\mathcal{A}$ is $\mathcal{P}_{\mathrm{B}}$-compatible with each AEI attached to it and there exists $C_i \in \{C_1, \ldots, C_n, C_{n+1}\}$, with $C_{n+1} = K^l$, such that:

$$[\![K^h, C_1, \ldots, C_i]\!]_{\mathcal{A}}^{\mathrm{pc};\#K^h,C_1,\ldots,C_i} \,/\, (Name - Low'_{C_i})$$
$$\approx_{\mathrm{B}}$$
$$[\![K^h, C_1, \ldots, C_i]\!]_{\mathcal{A}}^{\mathrm{pc};\#K^h,C_1,\ldots,C_i} \setminus High_{K^h} \,/\, (Name - Low'_{C_i})$$

where $Low'_{C_i} = \mathcal{V}_{C_i;C_{i+1}}$ for $1 \leq i \leq n$ and $Low'_{C_i} = Low_{K^l}$ for $i = n+1$, then $\mathcal{A}$ is noninterfering with respect to $High_{K^h}$ and Low_{K^l}. ∎

The presence of an AEI C_i satisfying the hypothesis of the theorem ensures that every information flow starting from K^h stops without reaching K^l. From a methodological standpoint, the noninterference check is applied in an incremental way by starting from C_1 and stopping as soon as C_i is found that satisfies the noninterference condition. If this check propagates to K^l without success, then K^h can interfere with K^l.

Example 4.6. Let us reconsider the analysis of the running example. Since the architectural topology of this system is acyclic, we can apply Prop. 4.5 in order to analyze the potential interference of component `S_High` on component `R_Low`. According to the theorem, the path to analyze is represented by the AEIs `S_High`, `B_High`, `U`, and `R_Low`. As can be easily seen, `S_High` interferes with `B_High`, but this pair of components does not interfere with the view of `U` interacting with `R_Low`. Hence, the information flow starting from `S_High` stops in `U` without reaching `R_Low`.

Now, consider an extension of the multilevel security routing system in which `U` is split into a chain of AEIs modeling different intermediate routing tasks needed to exchange correctly messages from each sender to the corresponding receiver. By virtue of Prop. 4.5, it is sufficient to check the compatibility between the additional AEIs in order to confirm the noninterference result shown above. ∎

In the case of cyclic topologies, noninterference can still be analyzed in a component-oriented fashion if we exploit the interoperability proposition. By following the same approach surveyed above, we now show an alternative characterization of Def. 3.1 for cycles that employs interoperability to make the noninterference check local and scalable. In particular, we describe sufficient conditions ensuring that the high interactions within a cycle do not interfere with the low behavior of an AEI C_j in the cycle.

Proposition 4.7. Given an architectural description $\mathcal{A}$, let $\mathcal{Y} = \{C_1, ..., C_n\}$ be the set of AEIs traversed by a cycle of $\mathcal{A}$, such that all the high and low local interactions of $\mathcal{Y}$ are involved in attachments between AEIs in $\mathcal{Y}$. For each $C_j \in \mathcal{Y}$ that $\mathcal{P}_{\mathrm{B}}$-interoperates with the other AEIs in the cycle, we have that $\mathcal{Y}$ is noninterfering with respect to $High_{C_1,...,C_n}$ and Low_{C_j} iff:

$$\llbracket C_j \rrbracket_{\mathcal{A}}^{\mathrm{pc;wob}} \,/\, (Name - Low_{C_j})$$
$$\approx_{\mathrm{B}}$$
$$\llbracket \mathcal{Y} \rrbracket_{\mathcal{A}}^{\mathrm{pc};\#\mathcal{Y}} \setminus High_{C_1,...,C_n} \,/\, (Name - Low_{C_j})$$

■

From a methodological standpoint, we observe that it may not be necessary to consider the interacting semantics of the whole cycle in order to infer the impact of the high local interactions of the cycle upon the low behavior of one of its AEIs. Indeed, let us assume that there exists an AEI C_i in the cycle such that all of its local interactions belong to *High*. Then, when preventing the high activities from being executed, C_i turns out to be isolated from the other AEIs in the cycle, i.e., the cycle becomes a chain because of the removal of C_i. Under this assumption, verifying the condition of Prop. 4.7 for an AEI C_j reduces to checking the compatibility of C_j with respect to such a chain. Hence, in this case we improve the verification efficiency as it is sufficient to apply repeatedly the compatibility check for acyclic topologies in order to shrink the chain and reduce it to C_j.

The last step towards the most general definition of a component-oriented noninterference check consists of extending the previous results to arbitrary topologies. Now, we consider the interference from an AEI K^h to an AEI K^l and we rephrase Prop. 4.5 by combining the sufficient conditions for stars and cycles introduced in this section with those of Thm. 2.13. In accordance with the topological reduction process, compatibility and interoperability checks are applied several times until the entire topology is reduced either to a single cyclic union – including both K^h and K^l – that satisfies Prop. 4.7, or to a path from K^h to K^l that satisfies Prop. 4.5. In the latter case, observed that

some consecutive AEIs in the path from K^h to K^l may be adjacent AEIs in a cyclic union, we can reduce the cyclic union to these adjacent AEIs iff such AEIs $\mathcal{P}_{\mathrm{B}}$-interoperate with the other AEIs in the cyclic union.

Theorem 4.8. Let $\mathcal{A}$ be an architectural description, K^h, K^l be two of its AEIs, and $\mathcal{CU}$ be a total set of cyclic unions for $\mathcal{A}$ if $\mathcal{A}$ is cyclic. Assume that the following conditions hold:

1. For each $C \in \mathcal{A}$ belonging to an acyclic portion or to the intersection of some cycle with acyclic portions of the abstract enriched flow graph of $\mathcal{A}$, C is $\mathcal{P}_{\mathrm{B}}$-compatible with every $C' \in \mathcal{B}_C - \mathcal{CU}_C$.
2. For each cyclic union $\{C_1, \ldots, C_n\} \in \mathcal{CU}$, every $C_j \in \mathcal{F}_{C_1,\ldots,C_n}$ $\mathcal{P}_{\mathrm{B}}$-interoperates with the other AEIs in the cyclic union.
3. If both K^h and K^l belong to a cyclic union $\mathcal{Y} \in \mathcal{CU}$, then $\mathcal{Y}$ satisfies the equality of Prop. 4.7 with respect to $High_{K^h}$ and Low_{K^l}, otherwise there exists a path connecting K^h to K^l through $n \geq 0$ AEIs $C_1, \ldots, C_n$ in the abstract enriched flow graph of $\mathcal{A}$ such that:
   (a) For each $\{C'_1, \ldots, C'_g\} \subseteq \{K^h, C_1, \ldots, C_n, K^l\}$ where $\{C'_1, \ldots, C'_g\}$ are adjacent AEIs in a cyclic union of $\mathcal{CU}$, $\{C'_1, \ldots, C'_g\}$ $\mathcal{P}_{\mathrm{B}}$-interoperate with the other AEIs in the cyclic union.
   (b) An AEI in $\{C_1, \ldots, C_n, K^l\}$ satisfies the equality of Prop. 4.5.

Then $\mathcal{A}$ is noninterfering with respect to $High_{K^h}$ and Low_{K^l}. ■

Example 4.9. Consider an extension of the running example in which `U` becomes the frontier of a cycle of AEIs modeling different intermediate routing tasks. By virtue of Thm. 4.8 and of Ex. 4.6, it is sufficient to check the $\mathcal{P}_{\mathrm{B}}$-interoperability of `U` with respect to such a cycle in order to infer that noninterference from `S_High` to `R_Low` is preserved. ■

5. Conclusions

In this paper, we have shown that a noninterference-like security property can be verified by means of architectural checks that proceed in a component-oriented manner on the basis of the architectural topology of the system. Formally, the proposed approach relies on labeled transition systems and techniques like abstraction and equivalence checking. Hence, it can be used not only in PADL, but in general within the architectural level of any development methodology that is supported by such a semantic framework.

Working at the architectural level is fundamental to make the formal verification of software architectures a feasible option also for practitioners. In particular, in the case of our approach, the designer must provide only the architectural description of the system and the list of components to which the noninterference analysis must be applied. Then, the feedback provided by the automatic checks can be exploited by the designer to locate the security-critical parts of the system and adopt adequate countermeasures – ranging from system topology reengineering to the substitution of components – against any undesired interference.

In the literature, there are several examples of compositional noninterference properties. For instance, [32] presents an overview of properties that are preserved by constructs of all process algebras with structural operational semantic rules. A similar classification is proposed in [28] in the setting of event systems. In the model of probabilistic dataflow, [23] shows a compositionality result inspired by the rely-guarantee principle: the system ensures the guarantee property whenever the environment offers the rely condition. If two systems are secure according to this principle, then their composition is secure as well. Based on similar ideas, several papers ([13, 19, 14, 26, 8, 7] to cite a few) apply compositional reasoning to the verification of security protocols.

The main difference with respect to all these approaches is that the results of our work show to which extent the specific architectural topology can be exploited to conduct compositional noninterference analysis for component-based software architectures. In general, our approach could be applied also to the formal models mentioned above, provided that on top of them architectural description languages are defined. On the other hand, it could be interesting to apply it in the setting of UML-like modeling paradigms, provided that a labeled transition system semantics is defined for them. For instance, we expect that this can be done in the setting of [24], where noninterference is formalized using abstract state machines in order to extend UML for secure system development. We also emphasize that our approach is not limited to the verification of noninterference-based security properties, but applies as well to dependability properties like, e.g., safety and availability, which can be easily rephrased in terms of noninterference. This result has been demonstrated in [4], where the noninterference approach has been used for the analysis of real-world case studies, like the NRL Pump secure routing system and power-manageable systems.

As future work, we plan to implement our approach in TwoTowers [9] as

well as to further enhance its scalability by taking advantage of architectural regularities and symmetries. In fact, architectural checks like compatibility and interoperability have been demonstrated to scale from a single architectural description to suitable extensions dealing with internal behavioral variations and (exogenous, endogenous, and multiplicity) topological variations [4]. Under analogous conditions, we expect that component-based noninterference scales to all these extensions as well. For instance, with regard to our running example, since we have proved the compatibility between the router and the low receiver, with a minor effort we may immediately derive the compatibility between the router and several concurrent low receivers, which guarantees the preservation of the noninterference property.

We conclude by observing that fine-grain information, such as probability distributions or temporal durations of events, can be added so as to augment the distinguishing power of the noninterference check (see, e.g., [17, 5, 3] and the references therein). To this aim, the noninterference notion must be defined in terms of behavioral equivalences like, e.g., weak probabilistic bisimilarity and weak Markovian bisimilarity. In this case, the architectural noninterference checks can still be used provided that the underlying behavioral equivalence is a congruence with respect to static operators, so that the topological reduction process can take place.

Acknowledgment: We would like to thank the anonymous reviewers for their useful and constructive comments that helped us to improve the quality of our original submission.

References

[1] A. Aldini, *"Classification of Security Properties in a Linda-like Process Algebra"*, in Science of Computer Programming 63:16–38, 2006.

[2] A. Aldini and M. Bernardo, *"On the Usability of Process Algebra: An Architectural View"*, in Theoretical Computer Science 335:281–329, 2005.

[3] A. Aldini and M. Bernardo, *"A General Framework for Nondeterministic, Probabilistic, and Stochastic Noninterference"*, in Proc. of the *Joint Workshop on Automated Reasoning for Security Protocol Analysis and Issues in the Theory of Security (ARSPA-WITS'09)*, Springer, LNCS 5511:18–33, 2009.

[4] A. Aldini, M. Bernardo, and F. Corradini, *"A Process Algebraic Approach to Software Architecture Design"*, Springer, 2010.

[5] A. Aldini, M. Bravetti, and R. Gorrieri, *"A Process-Algebraic Approach for the Analysis of Probabilistic Noninterference"*, in Journal of Computer Security 12:191–245, 2004.

[6] R. Allen and D. Garlan, *"A Formal Basis for Architectural Connection"*, in ACM Trans. on Software Engineering and Methodology 6:213–249, 1997.

[7] S. Andova, C. Cremers, K. Gjøsteen, S. Mauw, S. Mjølsnes, and S. Radomirovic, *"A Framework for Compositional Verification of Security Protocols"*, in Information and Computation 206:425–459, 2008.

[8] M. Backes, A. Datta, A. Derek, J.C. Mitchell, and M. Turuani, *"Compositional Analysis of Contract-Signing Protocols"*, in Theoretical Computer Science 367:33–56, 2006.

[9] M. Bernardo, TwoTowers 5.1 User Manual, 2006 `http://www.sti.uniurb.it/bernardo/twotowers/`.

[10] M. Bernardo, E. Bontà, and A. Aldini, *"Handling Communications in Process Algebraic Architectural Description Languages: Modeling, Verification, and Implementation"*, in Journal of Systems and Software 83:1404–1429, 2010.

[11] M. Bernardo and P. Inverardi (eds.), *"Formal Methods for Software Architectures"*, Springer, LNCS 2804, 2003.

[12] D. Beyer, T. Henzinger, R. Jhala, and R. Majumdar, *"The Software Model Checker Blast: Applications to Software Engineering"*, in Journal on Software Tools for Technology Transfer 9:505–525, 2007.

[13] M. Boreale and D. Gorla, *"On Compositional Reasoning in the Spi-Calculus"*, in Proc. of the *Int. Conf. on Foundations of Software Science and Computation Structures (FOSSACS'02)*, Springer, LNCS 2303:67–81, 2002.

[14] M. Bugliesi, R. Focardi, and M. Maffei, *"Compositional Analysis of Authentication Protocols"*, in Proc. of the *European Symp. on Programming (ESOP'04)*, Springer, LNCS 2986:140–154, 2004.

[15] C. Canal, E. Pimentel, and J.M. Troya, *"Compatibility and Inheritance in Software Architectures"*, in Science of Computer Programming 41:105–138, 2001.

[16] R. Focardi and R. Gorrieri, *"A Classification of Security Properties"*, in Journal of Computer Security 3:5–33, 1995.

[17] R. Focardi, F. Martinelli, and R. Gorrieri, *"Information Flow Analysis in a Discrete-Time Process Algebra"*, in Proc. of the *Computer Security Foundations Workshop (CSFW'00)*, IEEE-CS Press, pp. 170–184, 2000.

[18] J.A. Goguen and J. Meseguer, *"Security Policy and Security Models"*, in Proc. of the *Symp. on Security and Privacy (SSP'82)*, IEEE-CS Press, pp. 11–20, 1982.

[19] R. Gorrieri, F. Martinelli, M. Petrocchi, and A. Vaccarelli, *"Compositional Verification of Integrity for Digital Stream Signature Protocols"*, in Proc. of the *Int. Conf. on Application of Concurrency to System Design (ACSD'03)*, IEEE-CS Press, pp. 142–149, 2003.

[20] M. Hennessy and R. Milner, *"Algebraic Laws for Nondeterminism and Concurrency"*, in Journal of the ACM 32:137–162, 1985.

[21] P. Inverardi, A.L. Wolf, and D. Yankelevich, *"Static Checking of System Behaviors Using Derived Component Assumptions"*, in ACM Trans. on Software Engineering and Methodology 9:239–272, 2000.

[22] C.B. Jones, *"Specification and Design of (Parallel) Programs"*, in IFIP Congress, North-Holland, pp. 321–332, 1983.

[23] J. Jürjens, *"Secure Information Flow for Concurrent Processes"*, in Proc. of the *Int. Conf. on Concurrency Theory (CONCUR'00)*, Springer, LNCS 1877:395–409, 2000.

[24] J. Jürjens, *"Secure Systems Development with UML"*, Springer, 2005.

[25] M. Kang, A. Moore, and I. Moskowitz, *"Design and Assurance Strategy for the NRL Pump"*, in IEEE Computer Magazine 31:56–64, 1998.

[26] S. Lafrance, *"Using Equivalence-Checking to Verify Robustness to Denial of Service"*, in Computer Networks 50:1327–1348, 2006.

[27] G. Lowe, *"Casper: A Compiler for the Analysis of Security Protocols"*, in Proc. of the *Computer Security Foundations Workshop (CSFW'97)*, IEEE-CS Press, pp. 18–30, 1997.

[28] H. Mantel, *"On the Composition of Secure Systems"*, in Proc. of the *Symp. on Security and Privacy (SSP'02)*, IEEE-CS Press, pp. 88–104, 2002.

[29] J. McLean, *"Security Models and Information Flow"*, in Proc. of the *Symp. on Security and Privacy (SSP'90)*, IEEE-CS Press, pp. 180–187, 1990.

[30] R. Milner, *"Communication and Concurrency"*, Prentice Hall, 1989.

[31] P.Y.A. Ryan and S.A. Schneider, *"Process Algebra and Non-interference"*, in Journal of Computer Security 9:75–103, 2001.

[32] S. Tini, *"Rule Formats for Compositional Non-interference Properties"*, in Journal of Logic and Algebraic Programming 60:353–400, 2004.

[33] W. Visser, K. Havelund, G. Brat, S. Park, and F. Lerda, *"Model Checking Programs"*, in Automated Software Engineering Journal 10:203–232, 2003.

A. Appendix: Proofs of Results

We start by showing the proof of Thm. 2.13 addressing compatibility and interoperability checks for arbitrary topologies. This will be useful to understand the topological reduction process that is at the basis of some of the following noninterference results.

We first notice that the topological reduction process for an arbitrary topology can take place in several different ways depending on the interleaving of applications of the architectural compatibility and interoperability checks. In turn, this depends on the order in which the various stars and cycles of the topology are considered. In any case, all cycles have to be taken into account sooner or later, as the reduction process tends to transform an arbitrary topology into an acyclic topology.

The cycles of an architectural description $\mathcal{A}$ are managed by means of a cycle covering algorithm like the following, which relies on the notion of cyclic union:

1. All the AEIs of $\mathcal{A}$ are initially unmarked.
2. While there are unmarked AEIs in the cycles of the abstract enriched flow graph of $\mathcal{A}$:
   (a) Pick out one such AEI, say C.
   (b) Mark all the AEIs in $\mathcal{CU}_C$.

The cycle covering process is nondeterministic and results into a set $\mathcal{CU}$ of cyclic unions that include every AEI belonging to a cycle in the abstract enriched flow graph of $\mathcal{A}$. Any two cyclic unions in $\mathcal{CU}$ are connected at most through a single shared AEI or through the attachments between a single AEI of one cyclic union and a single AEI of the other cyclic union. The set $\mathcal{CU}$ is said to be total iff the topology becomes acyclic after replacing every cyclic union $\mathcal{Y} = \{C_1, \ldots, C_n\} \in \mathcal{CU}$ with an AEI whose behavior is given by:

$$[\![\mathcal{Y}]\!]_{\mathcal{A}}^{\mathrm{tc};\#\mathcal{Y};\mathcal{F}_{C_1,\ldots,C_n}} \,/\, (Name - \bigcup_{C_j \in \mathcal{F}_{C_1,\ldots,C_n}} \mathcal{V}_{C_j;\mathcal{A}}) \,/ \bigcup_{C_j \in \mathcal{F}_{C_1,\ldots,C_n}} (H_{C_j,\mathcal{Y}} \cup E_{C_j,\mathcal{Y}}).$$

Now, we are ready to present the proof of Thm 2.13.

Proof We proceed by induction on the number m of cycles in the abstract enriched flow graph of $\mathcal{A}$.

[Base] If $m = 0$ then $\mathcal{A}$ is acyclic. In this case, we prove the result by induction on the number s of stars in the abstract enriched flow graph of $\mathcal{A}$.

If $s = 1$, then let us assume that the only star in the abstract enriched flow graph of $\mathcal{A}$, composed of the AEIs $K, C_1, \ldots, C_n$, is centered on K. In order to avoid trivial cases, let us assume $n > 0$. We distinguish among three cases.

Case a. $[\![K]\!]^{\mathrm{pc;wob}}_{\mathcal{A}}$ satisfies $\mathcal{P}$. By virtue of 1., since K is $\mathcal{P}$-compatible with the AEIs in $\mathcal{B}_K$, we derive that $[\![K, \mathcal{B}_K]\!]^{\mathrm{tc};\#K,\mathcal{B}_K;K}_{K,\mathcal{B}_K} / \bigcup_{l=1}^{n}(H_{K,C_l} \cup E_{K,C_l})$ satisfies $\mathcal{P}$, and so do $[\![\mathcal{A}]\!]^{\mathrm{tc};\#K,\mathcal{B}_K;K}_{\mathrm{bbm}}$ and $[\![\mathcal{A}]\!]^{\mathrm{pc};\#\mathcal{A}}_{\mathrm{bbm}}$, because $\mathcal{P}$ does not contain any free use of negation.

Case b. $[\![C_j]\!]^{\mathrm{pc;wob}}_{\mathcal{A}}$, with $C_j \in \mathcal{B}_K$, satisfies $\mathcal{P}$. First, we observe that:

$$[\![K]\!]^{\mathrm{tc};\#C_j}_{C_j,\mathcal{B}_{C_j}} \approx_{\mathcal{P}} [\![K]\!]^{\mathrm{pc};\#\mathrm{C_j}}_{C_j,\mathcal{B}_{C_j}} / \varphi_{K,\mathrm{async}}(\mathcal{OAL}\mathcal{I}_K)$$

Moreover, we have that:

$$[\![K]\!]^{\mathrm{pc};\#\mathrm{C_j}}_{C_j,\mathcal{B}_{C_j}} / \varphi_{K,\mathrm{async}}(\mathcal{OAL}\mathcal{I}_K)$$
$$\approx_{\mathcal{P}}$$
$$[\![K]\!]^{\mathrm{pc};\#\mathrm{C_j}}_{\mathcal{A}} / (Name - \mathcal{V}_{K;C_j}) / \varphi_{K,\mathrm{async}}(\mathcal{OAL}\mathcal{I}_K)$$

Now, by virtue of 1., we have that K is $\mathcal{P}$-compatible with the AEIs in $\mathcal{B}_K - \{C_j\}$, from which we derive:

$$[\![K, \mathcal{B}_K - \{C_j\}]\!]^{\mathrm{tc};\#K,\mathcal{B}_K-\{C_j\};K}_{K,\mathcal{B}_K} / \bigcup_{l=1,l\neq j}^{n}(H_{K,C_l} \cup E_{K,C_l}) \approx_{\mathcal{P}} [\![K]\!]^{\mathrm{pc;wob}}_{\mathcal{A}}$$

Since $[\![K]\!]^{\mathrm{pc};\#\mathrm{C_j}}_{\mathcal{A}} / (Name - \mathcal{V}_{K;C_j}) / \varphi_{K,\mathrm{async}}(\mathcal{OAL}\mathcal{I}_K)$ contains $[\![K]\!]^{\mathrm{pc;wob}}_{\mathcal{A}}$ in parallel with the buffers associated with the originally asynchronous local interactions of K attached to C_j, from the last equality we derive:

$$[\![K]\!]^{\mathrm{pc};\#\mathrm{C_j}}_{\mathcal{A}} / (Name - \mathcal{V}_{K;C_j}) / \varphi_{K,\mathrm{async}}(\mathcal{OAL}\mathcal{I}_K)$$
$$\approx_{\mathcal{P}}$$
$$[\![K, \mathcal{B}_K - \{C_j\}]\!]^{\mathrm{tc};\#K,\mathcal{B}_K;K}_{K,\mathcal{B}_K} / \bigcup_{l=1,l\neq j}^{n}(H_{K,C_l} \cup E_{K,C_l})$$
$$/ (Name - \mathcal{V}_{K;C_j}) / \varphi_{K,\mathrm{async}}(\mathcal{OAL}\mathcal{I}_K)$$

Thanks to the last two hidings, all the actions but those in $\varphi_{K;C_j}(\mathcal{LI}_{K;C_j})$ are hidden, hence the first hiding is redundant:

$$[\![K, \mathcal{B}_K - \{C_j\}]\!]^{\mathrm{tc};\#K,\mathcal{B}_K;K}_{K,\mathcal{B}_K} / \bigcup_{l=1,l\neq j}^{n}(H_{K,C_l} \cup E_{K,C_l})$$
$$/ (Name - \mathcal{V}_{K;C_j}) / \varphi_{K,\mathrm{async}}(\mathcal{OAL}\mathcal{I}_K)$$
$$\approx_{\mathcal{P}}$$
$$[\![K, \mathcal{B}_K - \{C_j\}]\!]^{\mathrm{tc};\#K,\mathcal{B}_K;K}_{K,\mathcal{B}_K} / (Name - \mathcal{V}_{K;C_j}) / \varphi_{K,\mathrm{async}}(\mathcal{OAL}\mathcal{I}_K)$$

By definition of totally closed semantics, we then derive:

$$[\![K, \mathcal{B}_K - \{C_j\}]\!]^{\mathrm{tc};\#K,\mathcal{B}_K;K}_{K,\mathcal{B}_K} / (Name - \mathcal{V}_{K;C_j}) / \varphi_{K,\mathrm{async}}(\mathcal{OAL}\mathcal{I}_K)$$
$$\approx_{\mathcal{P}}$$
$$[\![K, \mathcal{B}_K - \{C_j\}]\!]^{\mathrm{tc};\#K,\mathcal{B}_K}_{K,\mathcal{B}_K} / (Name - \mathcal{V}_{K;C_j})$$

Hence, summarizing, we have proved that:

$$[\![K]\!]^{\mathrm{tc};\#C_j}_{C_j,\mathcal{B}_{C_j}} \approx_{\mathcal{P}} [\![K,\mathcal{B}_K-\{C_j\}]\!]^{\mathrm{tc};\#K,\mathcal{B}_K}_{K,\mathcal{B}_K} / (Name - \mathcal{V}_{K;C_j})$$

Now, by virtue of 1., C_j is $\mathcal{P}$-compatible with K:

$$([\![C_j]\!]^{\mathrm{pc};\#K}_{\mathcal{A}} \parallel_{\mathcal{S}(C_j,K;\mathcal{A})} [\![K]\!]^{\mathrm{tc};\#C_j}_{C_j,\mathcal{B}_{C_j}}) / (H_{C_j,K} \cup E_{C_j,K}) \approx_{\mathcal{P}} [\![C_j]\!]^{\mathrm{pc;wob}}_{\mathcal{A}}$$

Hence, from the last two equalities we obtain:

$$\begin{gathered}([\![C_j]\!]^{\mathrm{pc};\#K}_{\mathcal{A}} \parallel_{\mathcal{S}(C_j,K;\mathcal{A})} ([\![K,\mathcal{B}_K-\{C_j\}]\!]^{\mathrm{tc};\#K,\mathcal{B}_K}_{K,\mathcal{B}_K} / (Name - \mathcal{V}_{K;C_j}))) \\ / (H_{C_j,K} \cup E_{C_j,K}) \\ \approx_{\mathcal{P}} \\ [\![C_j]\!]^{\mathrm{pc;wob}}_{\mathcal{A}}\end{gathered}$$

Since $\approx_{\mathcal{P}}$ preserves $\mathcal{P}$, then the left-hand term of the previous equality satisfies $\mathcal{P}$. Then, since $\mathcal{P}$ does not contain any free use of negation, we derive that:

$$[\![K,\mathcal{B}_K]\!]^{\mathrm{tc};\#K,\mathcal{B}_K;C_j}_{K,\mathcal{B}_K}$$

satisfies $\mathcal{P}$ and, similarly, $[\![\mathcal{A}]\!]^{\mathrm{pc};\#\mathcal{A}}_{\mathrm{bbm}}$ so does.

Case c. No AEI in the star satisfies $\mathcal{P}$. By following the same arguments as **Case a**, we reduce the star to the AEI K, which does not satisfy $\mathcal{P}$, from which the result immediately follows.

Now, let the result hold for a certain $s \geq 1$ and suppose that the abstract enriched flow graph of $\mathcal{A}$ is composed of $s+1$ stars. Due to the acyclicity of $\mathcal{A}$, there must be a star, say composed of the AEIs $K, C_1, \ldots, C_n$ and centered on K, that is attached – with one of the AEIs in its border, say C_i – to only one other star in the abstract enriched flow graph of $\mathcal{A}$. Then, we distinguish among four cases.

Case I. $[\![C_i]\!]^{\mathrm{pc;wob}}_{\mathcal{A}}$ does not satisfy $\mathcal{P}$ and there is $C_j \in \{C_1, \ldots, C_n\} - \{C_i\}$ such that $[\![C_j]\!]^{\mathrm{pc;wob}}_{\mathcal{A}}$ satisfies $\mathcal{P}$.

By following the same arguments as **Case b**, where $\mathcal{B}_K - \{C_i\}$ is considered instead of $\mathcal{B}_K$, it is straightforward to obtain that:

$$[\![K,\mathcal{B}_K-\{C_i\}]\!]^{\mathrm{tc};\#K,\mathcal{B}_K-\{C_i\};C_j}_{K,\mathcal{B}_K}$$

satisfies $\mathcal{P}$. Now, by virtue of 1., K is $\mathcal{P}$-compatible with C_i:

$$([\![K]\!]^{\mathrm{pc};\#C_i}_{\mathcal{A}} \parallel_{\mathcal{S}(K,C_i;\mathcal{A})} [\![C_i]\!]^{\mathrm{tc;\#K}}_{K,\mathcal{B}_K}) / (H_{K,C_i} \cup E_{K,C_i}) \approx_{\mathcal{P}} [\![K]\!]^{\mathrm{pc;wob}}_{\mathcal{A}}$$

Since $\mathcal{P}$ is a congruence with respect to static operators, we derive:

$$\begin{gathered}([\![K,\mathcal{B}_K-\{C_i\}]\!]^{\mathrm{tc};\#K,\mathcal{B}_K;C_j}_{K,\mathcal{B}_K} \parallel_{\mathcal{S}(K,C_i;\mathcal{A})} [\![C_i]\!]^{\mathrm{tc};\#K}_{K,\mathcal{B}_K}) / (H_{K,C_i} \cup E_{K,C_i}) \\ \approx_{\mathcal{P}} \\ [\![K,\mathcal{B}_K-\{C_i\}]\!]^{\mathrm{tc};\#K,\mathcal{B}_K-\{C_i\};C_j}_{K,\mathcal{B}_K}\end{gathered}$$

and, as a consequence, also the left-hand term of this equality satisfies $\mathcal{P}$. Note that such a term is $\approx_{\mathcal{P}}$-equivalent to:

$$[\![K, \mathcal{B}_K]\!]^{\mathrm{tc};\#K,\mathcal{B}_K;C_j}_{K,\mathcal{B}_K} \,/\, (H_{K,C_i} \cup E_{K,C_i})$$

Since $\mathcal{P}$ does not contain any free use of negation, we derive that:

$$[\![K, \mathcal{B}_K]\!]^{\mathrm{tc};\#K,\mathcal{B}_K;C_j}_{K,\mathcal{B}_K}$$

satisfies $\mathcal{P}$ and, similarly:

$$[\![K, \mathcal{B}_K]\!]^{\mathrm{tc};\#K,\mathcal{B}_K;C_j}_{\mathcal{A}}$$

so does. Hence, the same holds also for:

$$[\![K, \mathcal{B}_K]\!]^{\mathrm{tc};\#K,\mathcal{B}_K;C_j}_{\mathcal{A}} \,/\, E_{K,\mathcal{B}_K}$$

where $E_{K,\mathcal{B}_K}$ is the set of exceptions that may be raised by semi-synchronous interactions involved in attachments between K and the AEIs in $\mathcal{B}_K$.

Now, consider the architectural description $\mathcal{A}'$ obtained by replacing the AEIs $K, C_1, \ldots, C_n$ with the AEI K' isomorphic to:

$$[\![K, \mathcal{B}_K]\!]^{\mathrm{tc};\#K,\mathcal{B}_K;C_j}_{\mathcal{A}} \,/\, E_{K,\mathcal{B}_K}$$

We now show that K' satisfies 1. Let C be an AEI attached to K' because it was previously attached to C_i. By virtue of 1., we have that:

$$([\![C_i]\!]^{\mathrm{pc};\#C}_{\mathcal{A}} \,\|_{\mathcal{S}(C_i,C;\mathcal{A})}\, [\![C]\!]^{\mathrm{tc};\#C_i}_{C_i,\mathcal{B}_{C_i}}) \,/\, (H_{C_i,C} \cup E_{C_i,C}) \;\approx_{\mathcal{P}}\; [\![C_i]\!]^{\mathrm{pc;wob}}_{\mathcal{A}}$$

from which it follows that:

$$([\![K']\!]^{\mathrm{pc};\#C}_{\mathcal{A}'} \,\|_{\mathcal{S}(K',C;\mathcal{A}')}\, [\![C]\!]^{\mathrm{tc};\#K'}_{K',\mathcal{B}_{K'}}) \,/\, (H_{K',C} \cup E_{K',C}) \;\approx_{\mathcal{P}}\; [\![K']\!]^{\mathrm{pc;wob}}_{\mathcal{A}'}$$

because $\approx_{\mathcal{P}}$ is a congruence with respect to the parallel composition operator.

On C side, we have to prove that since by virtue of 1.:

$$([\![C]\!]^{\mathrm{pc};\#C_i}_{\mathcal{A}} \,\|_{\mathcal{S}(C,C_i;\mathcal{A})}\, [\![C_i]\!]^{\mathrm{tc};\#C}_{C,\mathcal{B}_C}) \,/\, (H_{C,C_i} \cup E_{C,C_i}) \;\approx_{\mathcal{P}}\; [\![C]\!]^{\mathrm{pc;wob}}_{\mathcal{A}}$$

then it holds that:

$$([\![C]\!]^{\mathrm{pc};\#K'}_{\mathcal{A}'} \,\|_{\mathcal{S}(C,K';\mathcal{A}')}\, [\![K']\!]^{\mathrm{tc};\#C}_{C,\mathcal{B}_C}) \,/\, (H_{C,K'} \cup E_{C,K'}) \;\approx_{\mathcal{P}}\; [\![C]\!]^{\mathrm{pc;wob}}_{\mathcal{A}'}$$

Hence, we have to prove that:

$$[\![K']\!]^{\mathrm{tc};\#C}_{C,\mathcal{B}_C} \;\approx_{\mathcal{P}}\; [\![C_i]\!]^{\mathrm{tc};\#C}_{C,\mathcal{B}_C}$$

On the one hand, since K' is attached to C in $\mathcal{A}'$ because C_i is attached to C in $\mathcal{A}$, it holds that:

$$[\![K']\!]^{\mathrm{tc};\#C}_{C,\mathcal{B}_C}$$
$$\approx_{\mathcal{P}}$$
$$[\![K, \mathcal{B}_K]\!]^{\mathrm{tc};\#C,K,\mathcal{B}_K;C_j}_{\mathcal{A}} \,/\, E_{K,\mathcal{B}_K} \,/\, \varphi_{C_j,\mathrm{async}}(\mathcal{OALI}_{C_j}) \,/\, (\mathit{Name} - \mathcal{V}_{C_i;C})$$
$$\approx_{\mathcal{P}}$$
$$[\![K, \mathcal{B}_K]\!]^{\mathrm{tc};\#C,K,\mathcal{B}_K}_{\mathcal{A}} \,/\, E_{K,\mathcal{B}_K} \,/\, (\mathit{Name} - \mathcal{V}_{C_i;C})$$

On the other hand, it holds that:

$$[\![C_i]\!]^{\mathrm{tc};\#C}_{C,\mathcal{B}_C} \;\approx_{\mathcal{P}}\; [\![C_i]\!]^{\mathrm{pc};\#C}_{C,\mathcal{B}_C} \,/\, \varphi_{C_i,\mathrm{async}}(\mathcal{OALI}_{C_i})$$
$$\approx_{\mathcal{P}}$$
$$[\![C_i]\!]^{\mathrm{pc};\#C}_{\mathcal{A}} \,/\, \varphi_{C_i,\mathrm{async}}(\mathcal{OALI}_{C_i}) \,/\, (\mathit{Name} - \mathcal{V}_{C_i;C})$$

which includes $[\![C_i]\!]^{\mathrm{pc;wob}}_{\mathcal{A}}$ in parallel with the buffers associated with the originally asynchronous local interactions of C_i attached to C. By virtue of 1.,

C_i is $\mathcal{P}$-compatible with K:

$$[\![C_i]\!]^{\mathrm{pc;wob}}_{\mathcal{A}} \approx_{\mathcal{P}} ([\![C_i]\!]^{\mathrm{pc;\#K}}_{\mathcal{A}} \|_{\mathcal{S}(C_i,K;\mathcal{A})} [\![K]\!]^{\mathrm{tc;}\#C_i}_{C_i,\mathcal{B}_{C_i}}) \,/\, (H_{C_i,K} \cup E_{C_i,K})$$
$$\approx_{\mathcal{P}}$$
$$[\![C_i, K]\!]^{\mathrm{tc;}\#C_i,K;C_i}_{C_i,\mathcal{B}_{C_i}} \,/\, (H_{C_i,K} \cup E_{C_i,K})$$

from which we derive:

$$[\![C_i]\!]^{\mathrm{pc;}\#C}_{\mathcal{A}} \,/\, \varphi_{C_i,\mathrm{async}}(\mathcal{OALI}_{C_i}) \,/\, (Name - \mathcal{V}_{C_i;C})$$
$$\approx_{\mathcal{P}}$$
$$[\![C_i, K]\!]^{\mathrm{tc;}\#C,C_i,K;C_i}_{C_i,\mathcal{B}_{C_i}} \,/\, (H_{C_i,K} \cup E_{C_i,K}) \,/\, \varphi_{C_i,\mathrm{async}}(\mathcal{OALI}_{C_i}) \,/\, (Name - \mathcal{V}_{C_i;C})$$

because $\approx_{\mathcal{P}}$ is a congruence with respect to static operators.

Since $H_{C_i,K} \subseteq (Name - \mathcal{V}_{C_i;C})$, the right-hand term of this equality is $\approx_{\mathcal{P}}$-equivalent to:

$$[\![C_i, K]\!]^{\mathrm{tc;}\#C,C_i,K;C_i}_{C_i,\mathcal{B}_{C_i}} \,/\, E_{C_i,K} \,/\, \varphi_{C_i,\mathrm{async}}(\mathcal{OALI}_{C_i}) \,/\, (Name - \mathcal{V}_{C_i;C})$$

Note that this term includes $[\![K]\!]^{\mathrm{pc;wob}}_{\mathcal{A}}$. By virtue of 1., K is $\mathcal{P}$-compatible with $\mathcal{B}_K - \{C_i\}$, i.e.:

$$[\![K]\!]^{\mathrm{pc;wob}}_{\mathcal{A}} \approx_{\mathcal{P}} [\![K, \mathcal{B}_K - \{C_i\}]\!]^{\mathrm{tc;}\#K,\mathcal{B}_K - C_i;K}_{K,\mathcal{B}_K} \,/\, \overset{n}{\underset{l=1,l\neq i}{\cup}} (H_{K,C_l} \cup E_{K,C_l})$$

Since $\approx_{\mathcal{P}}$ is a congruence with respect to static operators, from this equality we derive that:

$$[\![C_i, K]\!]^{\mathrm{tc;}\#C,C_i,K;C_i}_{C_i,\mathcal{B}_{C_i}} \,/\, E_{C_i,K} \,/\, \varphi_{C_i,\mathrm{async}}(\mathcal{OALI}_{C_i}) \,/\, (Name - \mathcal{V}_{C_i;C})$$
$$\approx_{\mathcal{P}}$$
$$[\![C_i, K, \mathcal{B}_K - \{C_i\}]\!]^{\mathrm{tc;}\#C,K,\mathcal{B}_K;C_i}_{C_i,\mathcal{B}_{C_i}} \,/\, \overset{n}{\underset{l=1,l\neq i}{\cup}} (H_{K,C_l} \cup E_{K,C_l}) \,/\, E_{C_i,K}$$
$$/\, \varphi_{C_i,\mathrm{async}}(\mathcal{OALI}_{C_i}) \,/\, (Name - \mathcal{V}_{C_i;C})$$

Since $\overset{n}{\underset{l=1,l\neq i}{\cup}} H_{K,C_l} \subseteq (Name - \mathcal{V}_{C_i;C})$ and by definition of totally closed semantics, the right-hand term of this equality is $\approx_{\mathcal{P}}$-equivalent to:

$$[\![K, \mathcal{B}_K]\!]^{\mathrm{tc;}\#C,K,\mathcal{B}_K}_{C_i,\mathcal{B}_{C_i}} \,/\, \overset{n}{\underset{l=1}{\cup}} E_{K,C_l} \,/\, (Name - \mathcal{V}_{C_i;C})$$
$$\approx_{\mathcal{P}}$$
$$[\![K, \mathcal{B}_K]\!]^{\mathrm{tc;}\#C,K,\mathcal{B}_K}_{C_i,\mathcal{B}_{C_i}} \,/\, E_{K,\mathcal{B}_K} \,/\, (Name - \mathcal{V}_{C_i;C})$$

Since the hiding operation hides all the actions but the interactions from C_i attached to C, this term is $\approx_{\mathcal{P}}$-equivalent to:

$$[\![K, \mathcal{B}_K]\!]^{\mathrm{tc;}\#C,K,\mathcal{B}_K}_{\mathcal{A}} \,/\, E_{K,\mathcal{B}_K} \,/\, (Name - \mathcal{V}_{C_i;C})$$

Therefore, we have shown that:

$$[\![K']\!]^{\mathrm{tc;}\#C}_{C,\mathcal{B}_C} \approx_{\mathcal{P}} [\![C_i]\!]^{\mathrm{tc;}\#C}_{C,\mathcal{B}_C}$$

from which the $\mathcal{P}$-compatibility of C with respect to K' immediately follows.

Since $\mathcal{A}'$ has an acyclic topology with one fewer star topology with respect to $\mathcal{A}$, we can apply the induction hypothesis, from which it follows that

$[\![\mathcal{A}']\!]^{\mathrm{pc};\#\mathcal{A}'}_{\mathrm{bbm}}$ satisfies $\mathcal{P}$ because so does $[\![K']\!]^{\mathrm{pc;wob}}_{\mathcal{A}'}$. Therefore, since $\mathcal{P}$ does not contain any free use of negation, we immediately derive that $[\![\mathcal{A}]\!]^{\mathrm{pc};\#\mathcal{A}}_{\mathrm{bbm}}$ satisfies $\mathcal{P}$ because so does $[\![C_j]\!]^{\mathrm{pc;wob}}_{\mathcal{A}}$.

Case II. $[\![C_i]\!]^{\mathrm{pc;wob}}_{\mathcal{A}}$ satisfies $\mathcal{P}$, i.e. with respect to the first case C_i and C_j coincide. Hence, the proof straightforwardly derives from **Case I** and, in particular, K' turns out to be isomorphic to $[\![K, \mathcal{B}_K]\!]^{\mathrm{tc};\#K,\mathcal{B}_K;C_i}_{\mathcal{A}}$.

Case III. $[\![K]\!]^{\mathrm{pc;wob}}_{\mathcal{A}}$ satisfies $\mathcal{P}$. Hence, the proof straightforwardly derives from **Case a** and **Case I**, and, in particular, K' turns out to be isomorphic to $[\![K, \mathcal{B}_K]\!]^{\mathrm{tc};\#K,\mathcal{B}_K;K}_{\mathcal{A}}$.

Case IV. No AEI in the star satisfies $\mathcal{P}$. It is sufficient to apply the same arguments of the previous case and observe that K' does not satisfy $\mathcal{P}$.

[Induction] Now, let the result hold for a certain $m \geq 0$ and suppose that the abstract enriched flow graph of $\mathcal{A}$ has $m+1$ cycles. Since $\mathcal{CU}$ is total, let $\mathcal{Y} = \{C_1, \ldots, C_n\}$ be a cyclic union in $\mathcal{CU}$ that directly interacts with at most one cyclic union in $\mathcal{CU}$. In the following we assume $I = \{C_g\} \cup \mathcal{F}_{C_1,\ldots,C_n}$ if there exists C_g satisfying 2.c, and $I = \mathcal{F}_{C_1,\ldots,C_n}$ otherwise.

Now we replace the AEIs $C_1, \ldots, C_n$ with a new AEI C whose behavior is isomorphic to:

$$[\![\mathcal{Y}]\!]^{\mathrm{tc};\#\mathcal{Y};I}_{\mathcal{A}} \,/\, (\mathit{Name} - \cup_{C' \in I} \mathcal{V}_{C';\mathcal{A}}) \,/\, \cup_{C' \in I} (H_{C',\mathcal{Y}} \cup E_{C',\mathcal{Y}})$$

thus obtaining an architectural description $\mathcal{A}'$ such that:

1. $[\![C]\!]^{\mathrm{pc;wob}}_{\mathcal{A}'}$ satisfies $\mathcal{P}$ if and only if at least one AEI in $\mathcal{Y}$ so does. Indeed, one such AEI exists in $\mathcal{Y}$ if and only if, by virtue of 2.b and 2.c, I includes an AEI C' such that $[\![C']\!]^{\mathrm{pc;wob}}_{\mathcal{A}}$ satisfies $\mathcal{P}$ and C' $\mathcal{P}$-interoperates with $\mathcal{Y}$, if and only if:
$$[\![\mathcal{Y}]\!]^{\mathrm{tc};\#\mathcal{Y};C'}_{\mathcal{A}} \,/\, (\mathit{Name} - \mathcal{V}_{C';\mathcal{A}}) \,/\, (H_{C',\mathcal{Y}} \cup E_{C',\mathcal{Y}})$$
satisfies $\mathcal{P}$ and $[\![C]\!]^{\mathrm{pc;wob}}_{\mathcal{A}'}$ so does because $\mathcal{P}$ does not contain any free use of negation.
2. C preserves 1. In fact, let K be an AEI attached to C because it was previously attached to an AEI C_j of $\mathcal{F}_{C_1,\ldots,C_n}$. Hence, C_j is $\mathcal{P}$-compatible with K and vice versa. On C_j side, we have that:
$$([\![C_j]\!]^{\mathrm{pc};\#K}_{\mathcal{A}} \parallel_{\mathcal{S}(C_j,K;\mathcal{A})} [\![K]\!]^{\mathrm{tc};\#C_j}_{C_j,\mathcal{B}_{C_j}}) \,/\, (H_{C_j,K} \cup E_{C_j,K}) \;\approx_{\mathcal{P}}\; [\![C_j]\!]^{\mathrm{pc;wob}}_{\mathcal{A}}$$
from which it follows that:
$$([\![C]\!]^{\mathrm{pc};\#K}_{\mathcal{A}'} \parallel_{\mathcal{S}(C,K;\mathcal{A}')} [\![K]\!]^{\mathrm{tc};\#C}_{C,\mathcal{B}_C}) \,/\, (H_{C,K} \cup E_{C,K}) \;\approx_{\mathcal{P}}\; [\![C]\!]^{\mathrm{pc;wob}}_{\mathcal{A}'}$$
because $\approx_{\mathcal{P}}$ is a congruence with respect to the parallel composition operator.
On K side, it can be similarly shown that from:
$$([\![K]\!]^{\mathrm{pc};\#C_j}_{\mathcal{A}} \parallel_{\mathcal{S}(K,C_j;\mathcal{A})} [\![C_j]\!]^{\mathrm{tc};\#K}_{K,\mathcal{B}_K}) \,/\, (H_{K,C_j} \cup E_{K,C_j}) \;\approx_{\mathcal{P}}\; [\![K]\!]^{\mathrm{pc;wob}}_{\mathcal{A}}$$

we derive:

$$(\llbracket K \rrbracket^{\mathrm{pc};\#C}_{\mathcal{A}'} \parallel_{\mathcal{S}(K,C;\mathcal{A}')} \llbracket C \rrbracket^{\mathrm{tc};\#K}_{K,\mathcal{B}_K}) \,/\, (H_{K,C} \cup E_{K,C}) \;\approx_{\mathcal{P}}\; \llbracket K \rrbracket^{\mathrm{pc,wob}}_{\mathcal{A}'}$$

because C_j $\mathcal{P}$-interoperates with its cyclic union.

3. If $\mathcal{A}'$ is cyclic, then 2. is preserved. In fact, let $\mathcal{CU}'$ be the set of cyclic unions for $\mathcal{A}'$ obtained from $\mathcal{CU}$ by replacing in each original cyclic union every occurrence of $C_1, \ldots, C_n$ with C. Every cyclic union in $\mathcal{CU}'$ that does not include C has a corresponding topologically equivalent cyclic union in $\mathcal{CU}$.
   Now, consider $\mathcal{X}' \in \mathcal{CU}'$ formed by the AEIs $H_1, \ldots, H_m, C$. Hence, $\mathcal{CU}$ includes a cyclic union $\mathcal{X}$ formed by the AEIs $H_1, \ldots, H_m, C_j$, where $C_j \in \mathcal{F}_{C_1,\ldots,C_n}$. By virtue of 2.b:

$$\llbracket \mathcal{X} \rrbracket^{\mathrm{tc};\#\mathcal{X};C_j}_{\mathcal{A}} \,/\, (\mathit{Name} - \mathcal{V}_{C_j;\mathcal{A}}) \,/\, (H_{C_j,\mathcal{X}} \cup E_{C_j,\mathcal{X}}) \;\approx_{\mathcal{P}}\; \llbracket C_j \rrbracket^{\mathrm{pc;wob}}_{\mathcal{A}}$$

   Since $\approx_{\mathcal{P}}$ is a congruence with respect to the parallel composition operator:

$$\llbracket \mathcal{X}' \rrbracket^{\mathrm{tc};\#\mathcal{X}';C}_{\mathcal{A}'} \,/\, (\mathit{Name} - \mathcal{V}_{C;\mathcal{A}'}) \,/\, (H_{C,\mathcal{X}'} \cup E_{C,\mathcal{X}'}) \;\approx_{\mathcal{P}}\; \llbracket C \rrbracket^{\mathrm{pc;wob}}_{\mathcal{A}'}$$

   As a consequence, if $\mathcal{F}_{H_1,\ldots,H_m,C} = \emptyset$ then 2.a is preserved, otherwise if $C \in \mathcal{F}_{H_1,\ldots,H_m,C}$, then C preserves 2.b.
   Similarly as shown before, for each $H_l \in \mathcal{F}_{H_1,\ldots,H_m,C} - \{C\}$ by hypothesis we have:

$$\llbracket \mathcal{X} \rrbracket^{\mathrm{tc};\#\mathcal{X};H_l}_{\mathcal{A}} \,/\, (\mathit{Name} - \mathcal{V}_{H_l;\mathcal{A}}) \,/\, (H_{H_l,\mathcal{X}} \cup E_{H_l,\mathcal{X}}) \;\approx_{\mathcal{P}}\; \llbracket H_l \rrbracket^{\mathrm{pc;wob}}_{\mathcal{A}}$$

   from which we derive:

$$\llbracket \mathcal{X}' \rrbracket^{\mathrm{tc};\#\mathcal{X}';H_l}_{\mathcal{A}'} \,/\, (\mathit{Name} - \mathcal{V}_{H_l;\mathcal{A}'}) \,/\, (H_{H_l,\mathcal{X}'} \cup E_{H_l,\mathcal{X}'}) \;\approx_{\mathcal{P}}\; \llbracket H_l \rrbracket^{\mathrm{pc;wob}}_{\mathcal{A}'}$$

   because C_j $\mathcal{P}$-interoperates with its cyclic union. Hence, H_l preserves 2.b.
   Now, let us consider 2.c and assume that no AEI in the frontier of $\mathcal{X}$ satisfies $\mathcal{P}$. If by virtue of 2.c there is $H_g \in \mathcal{X}$ such that $\llbracket H_g \rrbracket^{\mathrm{pc;wob}}_{\mathcal{A}}$ satisfies $\mathcal{P}$ and H_g $\mathcal{P}$-interoperates with $\mathcal{X}$, then, by virtue of the same arguments surveyed above for H_l, we immediately derive that H_g $\mathcal{P}$-interoperates with $\mathcal{X}'$, thus preserving 2.c.
   On the other hand, if C_j is such that $\llbracket C_j \rrbracket^{\mathrm{pc;wob}}_{\mathcal{A}}$ satisfies $\mathcal{P}$, then we have shown that $\llbracket C \rrbracket^{\mathrm{pc;wob}}_{\mathcal{A}'}$ satisfies $\mathcal{P}$ and $\mathcal{P}$-interoperates with $\mathcal{X}'$. Hence, C preserves 2.c in the case it does not belong to the frontier of $\mathcal{X}'$.
4. The abstract enriched flow graph of $\mathcal{A}'$ has at most m cycles.

Then, by the induction hypothesis, the theorem holds for $\llbracket \mathcal{A}' \rrbracket^{\mathrm{pc};\#\mathcal{A}'}_{\mathrm{bbm}}$. Since $\mathcal{P}$ does not contain any free use of negation, we immediately derive that the theorem holds also for $\llbracket \mathcal{A} \rrbracket^{\mathrm{pc};\#\mathcal{A}}_{\mathrm{bbm}}$. ∎

Now, we are ready to introduce the proofs of the results presented in Sect. 4.

Proposition 4.3 We have to show that:

$$[\![K, \mathcal{B}_K]\!]^{\mathrm{pc};\#K,\mathcal{B}_K}_{K,\mathcal{B}_K} /(Name - Low_{K,C_1,\ldots,C_n}) \approx_{\mathrm{B}} [\![K, \mathcal{B}_K]\!]^{\mathrm{pc};\#K,\mathcal{B}_K}_{K,\mathcal{B}_K} \backslash High_K / (Name - Low_{K,C_1,\ldots,C_n})$$

We prove the result by induction on the number g of AEIs in $\mathcal{B}_K$ parameterizing the restriction operator. The case $g = 0$ is trivial. Hence, assume that the result holds for $g - 1$. By hypothesis:

$$[\![C^h_g, K]\!]^{\mathrm{pc};\#C^h_g,K}_{K,\mathcal{B}_K} / High_{K\#C^h_g} \approx_{\mathrm{B}} [\![C^h_g, K]\!]^{\mathrm{pc};\#C^h_g,K}_{K,\mathcal{B}_K} \backslash High_{K\#C^h_g}$$

Call K_1 (resp. K_2) the left-hand (resp. right-hand) term of this equality.

Since $High_{K\#C^h_g} \subseteq (Name - Low_{K,C_1,\ldots,C_n})$ and $High_{K\#C^h_i} \cap High_{K\#C^h_j} = \emptyset$, $i \neq j$, it holds that:

$$[\![K, \mathcal{B}_K]\!]^{\mathrm{pc};\#K,\mathcal{B}_K}_{K,\mathcal{B}_K} /(Name - Low_{K,C_1,\ldots,C_n}) \approx_{\mathrm{B}}$$
$$([\![K_1]\!]^{\mathrm{pc};\#\mathcal{B}_K - C^h_g}_{K,\mathcal{B}_K} \|_{\mathcal{S}(K;\mathcal{B}_K - C^h_g)} [\![\mathcal{B}_K - C^h_g]\!]^{\mathrm{pc};\#K}_{K,\mathcal{B}_K}) /(Name - Low_{K,C_1,\ldots,C_n}) \approx_{\mathrm{B}}$$
$$([\![K_2]\!]^{\mathrm{pc};\#\mathcal{B}_K - C^h_g}_{K,\mathcal{B}_K} \|_{\mathcal{S}(K;\mathcal{B}_K - C^h_g)} [\![\mathcal{B}_K - C^h_g]\!]^{\mathrm{pc};\#K}_{K,\mathcal{B}_K}) /(Name - Low_{K,C_1,\ldots,C_n})$$

We now observe that we can apply the induction hypothesis, thus obtaining:

$$([\![K_2]\!]^{\mathrm{pc};\#\mathcal{B}_K - C^h_g}_{K,\mathcal{B}_K} \|_{\mathcal{S}(K;\mathcal{B}_K - C^h_g)} [\![\mathcal{B}_K - C^h_g]\!]^{\mathrm{pc};\#K}_{K,\mathcal{B}_K}) /(Name - Low_{K,C_1,\ldots,C_n}) \approx_{\mathrm{B}}$$
$$([\![K_2]\!]^{\mathrm{pc};\#\mathcal{B}_K - C^h_g}_{K,\mathcal{B}_K} \|_{\mathcal{S}(K;\mathcal{B}_K - C^h_g)} [\![\mathcal{B}_K - C^h_g]\!]^{\mathrm{pc};\#K}_{K,\mathcal{B}_K}) \backslash High_{K\#C^h_1} \ldots \backslash High_{K\#C^h_{g-1}}$$
$$/(Name - Low_{K,C_1,\ldots,C_n}) \approx_{\mathrm{B}}$$
$$[\![K, \mathcal{B}_K]\!]^{\mathrm{pc};\#K,\mathcal{B}_K}_{K,\mathcal{B}_K} \backslash High_{K\#C^h_1} \ldots \backslash High_{K\#C^h_g}$$
$$/(Name - Low_{K,C_1,\ldots,C_n}) \approx_{\mathrm{B}}$$
$$[\![K, \mathcal{B}_K]\!]^{\mathrm{pc};\#K,\mathcal{B}_K}_{K,\mathcal{B}_K} \backslash High_K / (Name - Low_{K,C_1,\ldots,C_n})$$

because $High_K = \bigcup_{i=1}^{g} High_{K\#C^h_i}$. ∎

Proposition 4.4 Since $\mathcal{OALI}_{C_i} \cap Low_K = \emptyset$, with $1 \leq i \leq n$, then:

$$[\![K, \mathcal{B}_K]\!]^{\mathrm{pc};\#K,\mathcal{B}_K}_{K,\mathcal{B}_K} /(Name - Low_K) \approx_{\mathrm{B}} [\![K, \mathcal{B}_K]\!]^{\mathrm{tc};\#K,\mathcal{B}_K;K}_{K,\mathcal{B}_K} /(Name - Low_K)$$

Similarly:

$$[\![K, \mathcal{B}_K]\!]^{\mathrm{tc};\#K,\mathcal{B}_K;K}_{K,\mathcal{B}_K} /(Name - Low_K) \approx_{\mathrm{B}}$$
$$[\![K, \mathcal{B}_K]\!]^{\mathrm{tc};\#K,\mathcal{B}_K;K}_{K,\mathcal{B}_K} / \bigcup_{i=1}^{n} (H_{K,C_i} \cup E_{K,C_i}) /(Name - Low_K)$$

and, by compatibility hypothesis:

$$[\![K, \mathcal{B}_K]\!]^{\mathrm{tc};\#K,\mathcal{B}_K;K}_{K,\mathcal{B}_K} / \bigcup_{i=1}^{n} (H_{K,C_i} \cup E_{K,C_i}) /(Name - Low_K) \approx_{\mathrm{B}}$$
$$[\![K]\!]^{\mathrm{pc;wob}}_{\mathcal{A}} /(Name - Low_K) \approx_{\mathrm{B}}$$
$$[\![K]\!]^{\mathrm{pc;wob}}_{K,\mathcal{B}_K} /(Name - Low_K)$$

Then, by noninterference hypothesis:

$$[\![K]\!]^{\mathrm{pc;wob}}_{K,\mathcal{B}_K}/(Name - Low_K) \approx_{\mathrm{B}} [\![K]\!]^{\mathrm{pc;wob}}_{K,\mathcal{B}_K}\backslash High_K/(Name - Low_K)$$

Now, by repeating backward the same argument:

$$[\![K]\!]^{\mathrm{pc;wob}}_{K,\mathcal{B}_K}\backslash High_K/(Name - Low_K) \approx_{\mathrm{B}} [\![K]\!]^{\mathrm{pc;wob}}_{\mathcal{A}}\backslash High_K/(Name - Low_K) \approx_{\mathrm{B}} [\![K,\mathcal{B}_K]\!]^{\mathrm{tc};\#K,\mathcal{B}_K;K}_{K,\mathcal{B}_K}/\bigcup_{i=1}^{n}(H_{K,C_i}\cup E_{K,C_i})\backslash High_K/(Name - Low_K) \approx_{\mathrm{B}} [\![K,\mathcal{B}_K]\!]^{\mathrm{tc};\#K,\mathcal{B}_K;K}_{K,\mathcal{B}_K}\backslash High_K/(Name - Low_K)$$

because $High_K \cap \bigcup_{i=1}^{n}(H_{K,C_i}\cup E_{K,C_i}) = \emptyset$. Finally:

$$[\![K,\mathcal{B}_K]\!]^{\mathrm{tc};\#K,\mathcal{B}_K;K}_{K,\mathcal{B}_K}\backslash High_K/(Name - Low_K) \approx_{\mathrm{B}} [\![K,\mathcal{B}_K]\!]^{\mathrm{pc};\#K,\mathcal{B}_K}_{K,\mathcal{B}_K}\backslash High_K/(Name - Low_K)$$

because $\mathcal{OALI}_{C_i} \cap High_K = \emptyset$, with $1 \leq i \leq n$. ∎

Proposition 4.5 By the compatibility hypothesis we can reduce $[\![\mathcal{A}]\!]^{\mathrm{pc};\#\mathcal{A}}_{\mathrm{bbm}}$ to $[\![\mathcal{A}']\!]^{\mathrm{pc};\#\mathcal{A}}_{\mathrm{bbm}}$, whose $n+2$ constituting AEIs are $\approx_{\mathcal{P}}$-equivalent to $[\![K^h]\!]^{\mathrm{pc;wob}}_{\mathcal{A}}$, $[\![C_1]\!]^{\mathrm{pc;wob}}_{\mathcal{A}}, \ldots, [\![C_n]\!]^{\mathrm{pc;wob}}_{\mathcal{A}}$, $[\![K^l]\!]^{\mathrm{pc;wob}}_{\mathcal{A}}$, respectively. With abuse of notation we continue to use $K^h, C_1, \ldots, C_n, K^l$ to refer to such AEIs and $\mathcal{A}$ to denote their set.

Note that the abstract enriched flow graph of $\mathcal{A}'$ is a chain from K^h to K^l through $C_1, \ldots, C_n$. Hence, it holds that:

$$[\![K^h, C_1, \ldots, C_n, K^l]\!]^{\mathrm{pc};\#\mathcal{A}}_{\mathcal{A}}/(Name - Low_{K^l}) \approx_{\mathrm{B}} ([\![K^h]\!]^{\mathrm{pc};\#\mathcal{A}}_{\mathcal{A}} \|_{\mathcal{S}(K^h,C_1;\mathcal{A})} [\![C_1]\!]^{\mathrm{pc};\#\mathcal{A}}_{\mathcal{A}} \ldots \|_{\mathcal{S}(K^h,C_i;\mathcal{A})\cup\bigcup_{j=1}^{i-1}\mathcal{S}(C_j,C_i;\mathcal{A})} [\![C_i]\!]^{\mathrm{pc};\#\mathcal{A}}_{\mathcal{A}} \ldots \|_{\mathcal{S}(K^h,K^l;\mathcal{A})\cup\bigcup_{j=1}^{n}\mathcal{S}(C_j,K^l;\mathcal{A})} [\![K^l]\!]^{\mathrm{pc};\#\mathcal{A}}_{\mathcal{A}})/(Name - Low_{K^l}) \approx_{\mathrm{B}} ((([\![K^h]\!]^{\mathrm{pc};\#\mathcal{A}}_{\mathcal{A}} \|_{\mathcal{S}(K^h,C_1;\mathcal{A})} [\![C_1]\!]^{\mathrm{pc};\#\mathcal{A}}_{\mathcal{A}} \ldots \|_{\mathcal{S}(K^h,C_i;\mathcal{A})\cup\bigcup_{j=1}^{i-1}\mathcal{S}(C_j,C_i;\mathcal{A})} [\![C_i]\!]^{\mathrm{pc};\#\mathcal{A}}_{\mathcal{A}})/(Name - Low'_{C_i}))\ldots \|_{\mathcal{S}(K^h,K^l;\mathcal{A})\cup\bigcup_{j=1}^{n}\mathcal{S}(C_j,K^l;\mathcal{A})} [\![K^l]\!]^{\mathrm{pc};\#\mathcal{A}}_{\mathcal{A}})/(Name - Low_{K^l}) \approx_{\mathrm{B}} ([\![K^h, C_1, \ldots, C_i]\!]^{\mathrm{pc};\#\mathcal{A}}_{\mathcal{A}}/(Name - Low'_{C_i}) \ldots \|_{\mathcal{S}(K^h,K^l;\mathcal{A})\cup\bigcup_{j=1}^{n}\mathcal{S}(C_j,K^l;\mathcal{A})} [\![K^l]\!]^{\mathrm{pc};\#\mathcal{A}}_{\mathcal{A}})/(Name - Low_{K^l}) \approx_{\mathrm{B}}$$

$$(\llbracket K^h, C_1, \ldots, C_i \rrbracket_{\mathcal{A}}^{\mathrm{pc};\#\mathcal{A}} \backslash \mathit{High}_{K^h} / (\mathit{Name} - \mathit{Low}'_{C_i}) \ldots \|_{\mathcal{S}(K^h,K^l;\mathcal{A}) \cup \bigcup_{j=1}^{n} \mathcal{S}(C_j,K^l;\mathcal{A})} \llbracket K^l \rrbracket_{\mathcal{A}}^{\mathrm{pc};\#\mathcal{A}}) / (\mathit{Name} - \mathit{Low}_{K^l})$$

$$\approx_{\mathrm{B}}$$

$$(\llbracket K^h, C_1, \ldots, C_i \rrbracket_{\mathcal{A}}^{\mathrm{pc};\#\mathcal{A}} \ldots \|_{\mathcal{S}(K^h,K^l;\mathcal{A}) \cup \bigcup_{j=1}^{n} \mathcal{S}(C_j,K^l;\mathcal{A})} \llbracket K^l \rrbracket_{\mathcal{A}}^{\mathrm{pc};\#\mathcal{A}}) \backslash \mathit{High}_{K^h} / (\mathit{Name} - \mathit{Low}_{K^l})$$

$$\approx_{\mathrm{B}}$$

$$\llbracket K^h, C_1, \ldots, C_n, K^l \rrbracket_{\mathcal{A}}^{\mathrm{pc};\#\mathcal{A}} \backslash \mathit{High}_{K^h} / (\mathit{Name} - \mathit{Low}_{K^l})$$

■

Proposition 4.7 Consider $C_j \in \mathcal{Y}$ that $\mathcal{P}_{\mathrm{B}}$-interoperates with the other AEIs in the cycle. Hence:

$$\llbracket \mathcal{Y} \rrbracket_{\mathcal{A}}^{\mathrm{tc};\#\mathcal{Y};C_j} / (\mathit{Name} - \mathcal{V}_{C_j;\mathcal{A}}) / (H_{C_j,\mathcal{Y}} \cup E_{C_j,\mathcal{Y}}) \approx_{\mathrm{B}} \llbracket C_j \rrbracket_{\mathcal{A}}^{\mathrm{pc;wob}}$$

and:

$$\llbracket \mathcal{Y} \rrbracket_{\mathcal{A}}^{\mathrm{tc};\#\mathcal{Y};C_j} / (\mathit{Name} - \mathcal{V}_{C_j;\mathcal{A}}) / (H_{C_j,\mathcal{Y}} \cup E_{C_j,\mathcal{Y}}) / (\mathit{Name} - \mathit{Low}_{C_j}) \approx_{\mathrm{B}} \llbracket C_j \rrbracket_{\mathcal{A}}^{\mathrm{pc;wob}} / (\mathit{Name} - \mathit{Low}_{C_j})$$

from which we derive that:

$$\llbracket \mathcal{Y} \rrbracket_{\mathcal{A}}^{\mathrm{pc};\#\mathcal{Y}} / (\mathit{Name} - \mathit{Low}_{C_j}) \approx_{\mathrm{B}} \llbracket C_j \rrbracket_{\mathcal{A}}^{\mathrm{pc;wob}} / (\mathit{Name} - \mathit{Low}_{C_j})$$

because all the visible actions in $\llbracket \mathcal{Y} \rrbracket_{\mathcal{A}}^{\mathrm{pc};\#\mathcal{Y}}$ that are hidden in:

$$\llbracket \mathcal{Y} \rrbracket_{\mathcal{A}}^{\mathrm{tc};\#\mathcal{Y};C_j} / (\mathit{Name} - \mathcal{V}_{C_j;\mathcal{A}}) / (H_{C_j,\mathcal{Y}} \cup E_{C_j,\mathcal{Y}})$$

belong to $(\mathit{Name} - \mathit{Low}_{C_j})$.

Now, by the last equality we have proved it follows that the equality:

$$\llbracket C_j \rrbracket_{\mathcal{A}}^{\mathrm{pc;wob}} / (\mathit{Name} - \mathit{Low}_{C_j}) \approx_{\mathrm{B}} \llbracket \mathcal{Y} \rrbracket_{\mathcal{A}}^{\mathrm{pc};\#\mathcal{Y}} \backslash \mathit{High}_{C_1,\ldots,C_n} / (\mathit{Name} - \mathit{Low}_{C_j})$$

holds iff:

$$\llbracket \mathcal{Y} \rrbracket_{\mathcal{A}}^{\mathrm{pc};\#\mathcal{Y}} / (\mathit{Name} - \mathit{Low}_{C_j}) \approx_{\mathrm{B}} \llbracket \mathcal{Y} \rrbracket_{\mathcal{A}}^{\mathrm{pc};\#\mathcal{Y}} \backslash \mathit{High}_{C_1,\ldots,C_n} / (\mathit{Name} - \mathit{Low}_{C_j})$$

iff:

$$\llbracket \mathcal{Y} \rrbracket_{\mathcal{Y}}^{\mathrm{pc};\#\mathcal{Y}} / (\mathit{Name} - \mathit{Low}_{C_j}) \approx_{\mathrm{B}} \llbracket \mathcal{Y} \rrbracket_{\mathcal{Y}}^{\mathrm{pc};\#\mathcal{Y}} \backslash \mathit{High}_{C_1,\ldots,C_n} / (\mathit{Name} - \mathit{Low}_{C_j})$$

because, by initial hypothesis, all the local interactions attached to AEIs that are not in $\mathcal{Y}$ can be safely hidden. ■

Theorem 4.8 Firstly, assume that both K^h and K^l belong to a cyclic union $\mathcal{Y} \in \mathcal{CU}$. By virtue of 1. and 2. we can apply Thm. 2.13 in order to reduce each acyclic portion of $\mathcal{A}$ and every cyclic union $\mathcal{Y}' \in \mathcal{CU}$ different from $\mathcal{Y}$. In particular, $\mathcal{F}_{\mathcal{Y}'} \neq \emptyset$, otherwise $\mathcal{Y}'$ would be isolated from the portion of $\mathcal{A}$ including K^l and K^h. By virtue of 2., we replace $\mathcal{Y}'$ with a new AEI whose behavior is isomorphic to:

$$\llbracket \mathcal{Y}' \rrbracket_{\mathcal{A}}^{\mathrm{tc};\#\mathcal{Y}';\mathcal{F}_{\mathcal{Y}'}} / (\mathit{Name} - \cup_{C \in \mathcal{F}_{\mathcal{Y}'}} \mathcal{V}_{C;\mathcal{A}}) / \cup_{C \in \mathcal{F}_{\mathcal{Y}'}} (H_{C,\mathcal{Y}'} \cup E_{C,\mathcal{Y}'})$$

By executing this topological reduction process we remain with a single cyclic union with empty frontier that is $\approx_{\mathcal{P}}$-equivalent to $\mathcal{Y}$. Then, by virtue of 3., we immediately derive the result from the application of Prop. 4.7.

Secondly, assume that $C_1, \ldots, C_n$, with $n \geq 0$, is a path of AEIs connecting K^h to K^l such that 3.a and 3.b hold. By applying the same approach surveyed above, by virtue of 1. and 2. we can apply Thm. 2.13 in order to reduce each acyclic portion of $\mathcal{A}$ and every cyclic union $\mathcal{Y}$ of $\mathcal{CU}$ that do not include K^h, K^l, and no AEIs in the path $C_1, \ldots, C_n$. By executing this topological reduction process we remain with a path of $2 + n$ AEIs that are $\approx_{\mathcal{P}}$-equivalent to $K^h, C_1, \ldots, C_n, K^l$, respectively, with this path possibly intersecting a number of cyclic unions. With abuse of notation we continue to use $K^h, C_1, \ldots, C_n, K^l$ to refer to the corresponding AEIs and $\mathcal{A}$ to refer to the obtained architectural description.

For every remaining cyclic union $\mathcal{Y}$ there exist adjacent AEIs $C'_1, \ldots, C'_g$ in $\mathcal{Y}$ such that $\mathcal{Y} \cap \{K^h, C_1, \ldots, C_n, K^l\} = \{C'_1, \ldots, C'_g\}$ and, by virtue of the reduction process described above, $\mathcal{F}_{\mathcal{Y}} \subseteq \{C'_1, \ldots, C'_g\}$.

Then, by virtue of 3.a:

$$[\![\mathcal{Y}]\!]_{\mathcal{A}}^{\mathrm{tc};\#\mathcal{Y};\{C'_1,\ldots,C'_g\}} \,/\, \bigcup_{i=1}^{g}(Name - \mathcal{V}_{C'_i;\mathcal{A}}) \,/\, \bigcup_{i=1}^{g}(H_{C'_i,\mathcal{Y}-\{C'_1,\ldots,C'_g\}} \cup E_{C'_i,\mathcal{Y}-\{C'_1,\ldots,C'_g\}})$$
$$\approx_{\mathrm{B}}$$
$$[\![C'_1, \ldots, C'_g]\!]_{\mathcal{A}}^{\mathrm{pc};\#\{C'_1,\ldots,C'_g\}}$$

Let us replace the cyclic union $\mathcal{Y}$ with a single AEI isomorphic to the left-hand term of this equality.

By virtue of these substitutions, we obtain an acyclic topology $\mathcal{A}'$ such that:

- $\mathcal{A}'$ preserves 1. by virtue of Thm. 2.13.
- $\mathcal{A}'$ preserves 3. By virtue of 3. applied to $\mathcal{A}$, the path $C_1, \ldots, C_n$, with $n \geq 0$, connects K^h to K^l and satisfies 3.b. In particular, assume that $\mathcal{T} = \{K^h, C_1, \ldots, C_i\}$ is the path satisfying the equality of Prop. 4.5. Then, also in $\mathcal{A}'$ there exists an AEI such that the equality of Prop. 4.5 holds, as we now show by induction on the number m of cyclic unions of $\mathcal{A}$ intersecting $\mathcal{T}$.

  The base case $m = 0$ is trivial ($\mathcal{A}'$ inherits $\mathcal{T}$ from $\mathcal{A}$ with no changes).

  Now, let use assume $m > 0$ and let $\{C'_1, \ldots, C'_g\}$, with $1 \leq g \leq n + 2$, be a set of adjacent AEIs in $\mathcal{A}$ traversed by a cyclic union $\mathcal{Y}$, such that $\{C'_1, \ldots, C'_g\} \cap \mathcal{T} \neq \emptyset$. Then, there exists $i' \geq i$ such that

$\{C'_1, \ldots, C'_g\} \subseteq \mathcal{T}' = \{K^h, C_1, \ldots, C_{i'}\} \subseteq \{K^h, C_1, \ldots, C_n, K^l\}$. Since, by 3.b applied to $\mathcal{A}$ it holds that:

$$[\![\mathcal{T}]\!]^{\mathrm{pc};\#\mathcal{T}}_{\mathcal{A}} /(Name - Low_{C_i}) \approx_{\mathrm{B}} [\![\mathcal{T}]\!]^{\mathrm{pc};\#\mathcal{T}}_{\mathcal{A}} \backslash High_{K^h} /(Name - Low_{C_i})$$

it also holds that (see Prop. 4.5):

$$[\![\mathcal{T}']\!]^{\mathrm{pc};\#\mathcal{T}'}_{\mathcal{A}} /(Name - Low_{C_{i'}}) \approx_{\mathrm{B}} [\![\mathcal{T}']\!]^{\mathrm{pc};\#\mathcal{T}'}_{\mathcal{A}} \backslash High_{K^h} /(Name - Low_{C_{i'}})$$

Now, observed that in $\mathcal{A}'$ the cyclic union $\mathcal{Y}$ has been replaced by a new AEI, call it C', we derive that:

$$[\![(\mathcal{T}' - \{C'_1, \ldots, C'_g\}) \cup C']\!]^{\mathrm{pc};\#(\mathcal{T}' - \{C'_1, \ldots, C'_g\}) \cup C'}_{\mathcal{A}'} /(Name - Low_{C_{i'}}) \approx_{\mathrm{B}} [\![(\mathcal{T}' - \{C'_1, \ldots, C'_g\}) \cup C']\!]^{\mathrm{pc};\#(\mathcal{T}' - \{C'_1, \ldots, C'_g\}) \cup C'}_{\mathcal{A}'} \backslash High_{K^h} /(Name - Low_{C_{i'}})$$

because $\{C'_1, \ldots, C'_g\}$ $\mathcal{P}_{\mathrm{B}}$-interoperate with the other AEIs in the cyclic union. Hence, we can apply the induction hypothesis, from which the result immediately follows.

Therefore, all the hypotheses of Prop. 4.5 are satisfied, from which the result immediately follows. ∎