Behaviour, Interaction and Dynamics*

Roberto Bruni[1], Hernán Melgratti[2], and Ugo Montanari[1]

[1] Dipartimento di Informatica, Università di Pisa, Italy
[2] Departamento de Computación, FCEyN, Universidad de Buenos Aires - Conicet, Argentina

Abstract. The growth and diffusion of reconfigurable and adaptive systems motivate the foundational study of models of software connectors that can evolve dynamically, as opposed to the better understood notion of static connectors. In this paper we investigate the interplay of behaviour, interaction and dynamics in the context of the BIP component framework, here denoted BI(P), as we disregard priorities. We introduce two extensions of BIP: 1) *reconfigurable* BI(P) allows to reconfigure the set of admissible interactions, while preserving the set of interacting components; 2) *dynamic* BI(P) allows to spawn new components and interactions during execution. Our main technical results show that reconfigurable BI(P) is as expressive as BI(P), while dynamic BI(P) allows to deal with infinite state systems. Still, we show that reachability remains decidable for dynamic BI(P).

1 Introduction

Recent years have witnessed an increasing interest about a rigorous modelling of (different classes of) connectors. The term *connector*, as used here, has been coined within the area of component-based software architectures, to name entities that can regulate the interaction of a collection of components [15]. This has led to the development of different mathematical frameworks that are used to specify, design, analyse, compare, prototype and implement suitable connectors. Our previous efforts have been focused at unifying different frameworks, in particular, the BIP component framework [2], Petri nets with boundaries [16] and the algebras of connectors [7,1] based on the tile model [12]. In [8] we have shown that BIP without priorities, written BI(P) in the following, is equally expressive as nets with boundaries. Thanks to the correspondence results in [16,10], we can define an algebra of connectors as expressive as BI(P), where a few basic connectors can be composed in series and parallel to generate any BI(P) system.

All above approaches deal with systems that have static structures, i.e., systems in which the possible interactions among components are all defined at design time and remain unchanged during runtime. Nevertheless, when shifting to connectors for systems that adapt their behaviour to changing environments,

* Research supported by the EU Integrated Project 257414 ASCENS, the Italian MIUR Project CINA (PRIN 2010/11), ANPCyT Project BID-PICT-2008-00319, and EU FP7-project MEALS 295261.

the situation is less well-understood. For example, approaches based on mobile calculi (like the π-calculus [14]) are not suited, because there the notion of connector / component is lost. In fact, a general and uniform theory for dynamic connectors is still lacking. On the one hand, static structures of connectors can be studied and executed efficiently. On the other hand, systems that can traverse a large or infinite number of connector configurations are better dealt with concise computational models that are tailored to dynamic structures.

Some recent progress has been done in [6], where Dy-BIP is proposed. We remind that an ordinary BIP component is defined by a set of ports P and a finite automaton whose transitions carry subsets of P as labels. An ordinary BIP system consists of a finite number of components (fixing the "Behaviour") whose ports are disjoint, together with a set of admissible synchronisations between the transitions of components (fixing the "Interaction"). Neither the set of components, nor the set of interactions can change over time. In contrast to BIP, the set of interactions can change dynamically in Dy-BIP, but this is obtained by ad-hoc design choices. As a consequence, the definition of Dy-BIP systems can be error-prone or lead to incomplete specifications unless the complex methodology outlined in [6] is adopted.

In order to contribute to the development of a general theory for dynamic connectors, in this paper we study two other extensions of the BI(P) framework with different degrees of "dynamism" that allow enhanced conciseness, modularity and expressiveness.

As a first step, we focus on a *reconfigurable* version of BI(P), analogous to but simpler than Dy-BIP. A reconfigurable BI(P) system allows for the dynamic modification of interactions among components, i.e., the set of available interactions changes as a side-effect of an interaction between components. Our first result proves that any reconfigurable BI(P) system is equivalent to an ordinary BI(P) system. This result is achieved by introducing a "controller" component for each interaction that can be added or removed at run-time. Roughly, the controller keeps track of whether the managed interaction is currently available or not and forces the components willing to use that interaction to synchronise also with the controller. This mapping shows that the reconfiguration capabilities provided by reconfigurable BI(P) do not increase the expressive power of BI(P). In fact, reconfigurable BI(P) only provides a more compact representation of ordinary systems, while ordinary BI(P) representations may require an exponential blow up in the number of components (it requires one controller for each possible interaction, and the interaction are subsets of ports). The crux of the proof is the fact that the set of controller components can be defined statically. In fact, the interfaces of components in reconfigurable BI(P) are static, i.e., the set of available ports in every component is fixed. As a consequence, the set of all possible interactions in a system are determined at design time (despite the fact that they can be enabled/disabled at run-time).

Our next step is to explore situations in which the interfaces of the components may change dynamically (i.e., to support the creation/elimination of ports). This requirement also imposes the necessity of handling interactions that

can be created/removed dynamically, as in reconfigurable BI(P). We take as an inspiring example the notion of correlation sets in web services [17,13], that is used to keep separate sessions between clients and servers. In these cases, when a partner call is made, then an instance of the session is initialised with suitable correlation data (e.g., specific message fields) gathered for the partner. To this aim we exploit *coloured* tokens, where the colours are freshly created session identifiers. This way, we do not need to replicate the ports and structure of components, instead we keep all the coloured tokens within the same instance of the component, distributed along its states: as in general it can happen that two or more replicas are in the same state, then it is possible that two or more coloured tokens mark the same state at the same time. An interaction is possible only when all the involved components carry correlated colours, i.e., identifiers for the same session. In fact, while session identifiers are created locally to each component (e.g., $\mathtt{s}_1$ in a first component and $\mathtt{s}_2$ in a second component), a new interaction is also created that correlates them (e.g., $\mathtt{s}_1\mathtt{s}_2$). Possibly many sessions are opened with the same partners involved. In subsequent interactions, correlation tokens are then exploited to identify the session that interaction is part of. When the session ends, the correlation tokens are discarded. At the beginning, when the system is initialised, we assume that all components carry correlated tokens, i.e., that they are all part of the same session. Correspondingly, we introduce an extension of BI(P), called *dynamic* BI(P), in which component instances and new interactions can be added/removed dynamically. In this case we obtain systems that are possibly infinite state and more expressive than ordinary BI(P) systems. However, our second main result shows that reachability is decidable for dynamic BI(P).

Structure of the paper. Section 2 recalls the basics of BI(P) systems. Section 3 introduces reconfigurable BI(P) systems and shows that they are as expressive as ordinary BI(P) systems. Section 4 introduces dynamic BI(P) systems and shows their correspondence with Place/Transition (P/T) Petri nets. Due to space limitation and to the fact that P/T Petri nets are mainly used here as a technical tool for the decidability proof, we assume the reader has some familiarity with P/T Petri nets and refer to the standard literature [11] otherwise. Both reconfigurable and dynamic BI(P) systems are illustrated over small motivating examples. Section 5 gives some concluding remarks.

2 The BIP component framework, and BI(P)

BIP [2] is a component framework that exploits a three-layered architecture: 1) the lower level is called *Behaviour* and it fixes the activities of individual atomic components; 2) the middle layer is called *Interaction* and it defines the handshaking mechanisms between components; and 3) the top level is called *Priority* and it assigns a partial order of preferences to the admissible interactions. This section recalls the formal definition of BIP using the notation from [4]. Here we disregard priorities and write BI(P) to avoid confusion.

The lower layer consists of a set of atomic components with ports. The sets of ports of components are pairwise disjoint, i.e., each port is uniquely assigned to a component. Components are automata whose transitions are labelled by sets of their ports.

Definition 1 (Component). *A* component $B = (Q, P, \rightarrow)$ *is a transition system where Q is a set of states (ranged over by $p, q, \ldots$), P is a set of ports (ranged over by $\mathtt{a}, \mathtt{b}, \ldots$), and $\rightarrow \subseteq Q \times 2^P \times Q$ is the set of labelled transitions.*

As usual, we write $q \xrightarrow{a} q'$ to denote the transition $(q, a, q') \in \rightarrow$. We say that a is enabled in q, denoted $q \xrightarrow{a}$, iff there exists q' s.t. $q \xrightarrow{a} q'$. We assume that for all q, q' it holds $q \xrightarrow{\varnothing} q'$ iff $q = q'$.

The second layer consists of connectors that specify the allowed interactions between components.

Definition 2 (Interaction). *Given a set of ports P, an* interaction *over P is a non-empty subset $a \subseteq P$.*

We write $\mathtt{a}_1\mathtt{a}_2\ldots\mathtt{a}_n$ for the interaction $\{\mathtt{a}_1, \mathtt{a}_2, \ldots, \mathtt{a}_n\}$ and $a \downarrow_{P_i}$ for the projection of $a \subseteq P$ over the set of ports $P_i \subseteq P$, i.e., $a \downarrow_{P_i} = a \cap P_i$.

Definition 3 (BI(P) system). *A* BI(P) system $B = \gamma(B_1, \ldots, B_n)$ *is the composition of a finite set $\{B_i\}_{i=1}^n$ of transitions systems $B_i = (Q_i, P_i, \rightarrow_i)$ such that their sets of ports are pairwise disjoint, i.e., $P_i \cap P_j = \varnothing$ for $i \neq j$, parametrized by a set $\gamma \subset 2^P$ of interactions over the set of ports $P = \biguplus_{i=1}^n P_i$. We call P the underlying set of ports of B, written $\iota(B)$.*

The semantics of a BI(P) system $\gamma(B_1, \ldots, B_n)$ is given by the transition system $(Q, P, \rightarrow_\gamma)$, with $Q = \Pi_i Q_i$, $P = \biguplus_{i=1}^n P_i$ and $\rightarrow_\gamma \subseteq Q \times 2^P \times Q$ is the least set of transitions satisfying the following inference rule

$$\frac{a \in \gamma \qquad \forall i \in 1..n : q_i \xrightarrow{a\downarrow_{P_i}} q_i'}{(q_1, \ldots, q_n) \xrightarrow{a}_\gamma (q_1', \ldots, q_n')}$$

Example 1. Consider the BI(P) system shown in Fig. 1, which contains a component $\mathtt{Server}$ that sequentially interacts with two clients $\mathtt{Client_1}$ and $\mathtt{Client_2}$. The $\mathtt{Server}$ starts a connection with $\mathtt{Client_i}$ thanks to the interaction $\mathtt{s_i c_i}$. Once the session is initiated, the server and the connected client synchronise over the interaction $\mathtt{ab_i}$. The session ends when the server and the connected client perform $\mathtt{e_i d_i}$. Note that the server has dedicated ports for handling the connections of different clients ($\mathtt{s_i}$ and $\mathtt{e_i}$), but it interacts analogously with all of them. The next section introduces an extension of BI(P) that allows for a more compact description of this kind of systems.

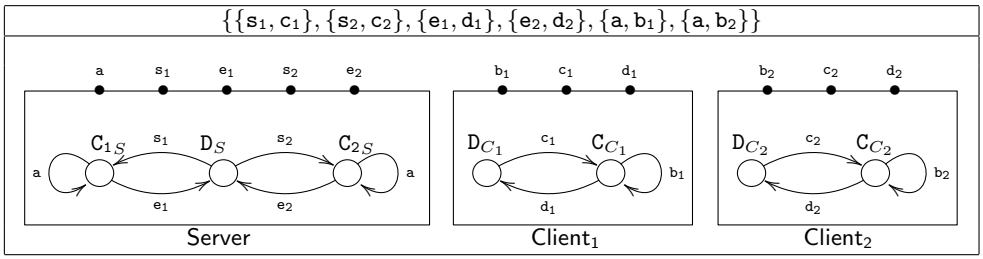


Fig. 1. A simple BI(P) system

3 Reconfigurable BI(P)

Our first extension is concerned with the possibility of enabling and disabling specific interactions dynamically, as proposed in an internal document of the ASCENS project. An interaction a can be enabled/disabled when all components involved in the interaction a agree to do so. After a is enabled, it can be used as an ordinary interaction until it gets disabled. Transitions in a reconfigurable BI(P) component are decorated with either (i) ϵ for ordinary actions over (a set of) ports (like the actions of ordinary BI(P) components), (ii) $+$ to add a new interaction, and (iii) $-$ to remove an interaction.

Definition 4 (Reconfigurable Component). *Let $\mathcal{P}$ be a set of ports. A* reconfigurable component $B = (Q, P, \longrightarrow\!\!\!\!\!\!\rightarrow)$ *is a transition system where Q is a set of states, $P \subset \mathcal{P}$ is a finite set of ports, and $\longrightarrow\!\!\!\!\!\!\rightarrow \subseteq Q \times 2^{\mathcal{P}} \times \{+, -, \epsilon\} \times Q$ is the set of labelled transitions such that $(q, a, \rho, q') \in \longrightarrow\!\!\!\!\!\!\rightarrow$ implies:*

1. *if $\rho = \epsilon$ then $a \in 2^P$;*
2. *if $\rho \in \{+, -\}$ then $a \cap P \neq \varnothing$.*

The annotation ρ indicates if the interaction a must be added $(+)$ to the set γ of global interactions, be removed $(-)$ from γ, or be already present (ϵ) in γ. Condition (1) states that the ports involved in any ordinary transition (i.e., $\rho = \epsilon$) are ports of the component, i.e., $a \in 2^P$. A transition that adds/removes a global interaction a may also refer to ports belonging to other components (Condition 2).

We write $q \xrightarrow{a\rho}\!\!\!\!\!\!\rightarrow q'$ to denote the transition $(q, a, \rho, q') \in \longrightarrow\!\!\!\!\!\!\rightarrow$. We say that a is enabled in q, denoted $q \xrightarrow{a}\!\!\!\!\!\!\rightarrow$, iff there exists q' s.t. $q \xrightarrow{a\epsilon}\!\!\!\!\!\!\rightarrow q'$. We assume that for all q, q' it holds $q \xrightarrow{\varnothing\epsilon}\!\!\!\!\!\!\rightarrow q'$ iff $q = q'$. Given a set of ports P, we write $a \# P$ if $a \cap P = \varnothing$.

Definition 5 (Reconfigurable BI(P) system). *A reconfigurable* BI(P) system $B = \gamma(B_1, \ldots, B_n)$ *is the composition of a finite set $\{B_i\}_{i=1}^n$ of reconfigurable components $B_i = (Q_i, P_i, \longrightarrow\!\!\!\!\!\!\rightarrow_i)$ such that their sets of ports are pairwise*

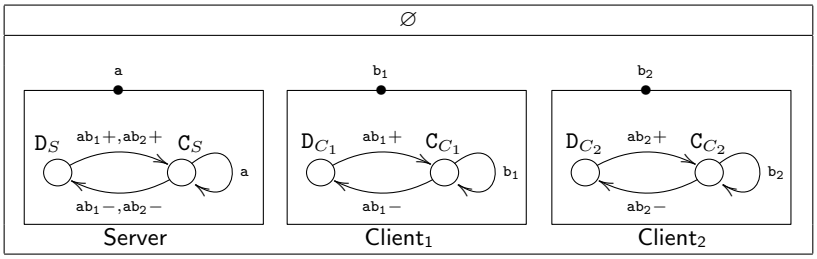


Fig. 2. A simple reconfigurable BI(P) system

$$\frac{a \in \gamma \quad \forall i \in 1..n : q_i \xrightarrow{a\downarrow_{P_i}\epsilon}\!\!\!\!\!\rightarrow q_i'}{\gamma(q_1,\ldots,q_n) \xrightarrow{a}\!\!\!\!\!\rightarrow \gamma(q_1',\ldots,q_n')}[\text{INT}]$$

$$\frac{a \in 2^P \setminus \gamma \quad \neg(a\#P_i) \implies q_i \xrightarrow{a+}\!\!\!\!\!\rightarrow q_i' \quad (a\#P_i) \implies q_i' = q_i \quad \gamma' = \gamma \cup \{a\}}{\gamma(q_1,\ldots,q_n) \xrightarrow{a}\!\!\!\!\!\rightarrow \gamma'(q_1',\ldots,q_n')}[\text{ADD}]$$

$$\frac{a \in \gamma \quad \neg(a\#P_i) \implies q_i \xrightarrow{a-}\!\!\!\!\!\rightarrow q_i' \quad (a\#P_i) \implies q_i' = q_i \quad \gamma' = \gamma \setminus \{a\}}{\gamma(q_1,\ldots,q_n) \xrightarrow{a}\!\!\!\!\!\rightarrow \gamma'(q_1',\ldots,q_n')}[\text{DEL}]$$

Fig. 3. Operational semantics of reconfigurable BI(P) systems

disjoint, i.e., $P_i \cap P_j = \varnothing$ for $i \neq j$, parametrized by a set $\gamma \subset 2^{\mathcal{P}}$. We call $P = \biguplus_{i=1}^{n} P_i$ the underlying set of ports of B, written $\iota(B)$.

Example 2. The client/server scenario introduced in Example 1 can be modelled as the reconfigurable BI(P) system depicted in Fig. 2. Now, the server and the clients have the transitions $\mathtt{ab_i}+$ and $\mathtt{ab_i}-$ that respectively allow for the dynamic enabling/disabling of the interaction $\mathtt{ab_i}$. In this case, the connection of a client to a server is modelled by the dynamic enabling of the interaction $\mathtt{ab_i}$. Analogously, the disconnection is modelled as the dynamic disabling of the interaction $\mathtt{ab_i}$.

The semantics of a reconfigurable BI(P) system $B = \gamma(B_1,\ldots,B_n)$ with $\iota(B) = P$ and $\gamma \subseteq 2^P$ is given by the transition system $(Q, \longrightarrow\!\!\!\!\!\rightarrow)$ where

- $Q = 2^P \times \Pi_i Q_i$ (we write $\gamma(q_1,\ldots,q_n)$ for $\langle \gamma, \langle q_1,\ldots,q_n\rangle\rangle \in Q$), and
- $\longrightarrow\!\!\!\!\!\rightarrow \subseteq Q \times 2^P \times Q$ is the least set of transitions satisfying the inference rules in Fig. 3.

Each state of the transition system keeps, not only the states of all components but also, the set γ of all enabled interactions. Rule [INT] stands for ordinary interactions and it is analogous to the inference rule for ordinary BI(P)

systems. Rule [ADD] accounts for the addition of a new global interaction a to the set of enabled interactions γ. Note that all components affected by the interaction a, i.e., the ones that have ports in a (condition $a\#P_i$), need to agree on the addition of the new interaction a (i.e., all of them perform the transition $\xrightarrow{a+}\!\!\!\!\!\!\rightarrow$). Remaining components do not move. Rule [DEL] specifies the removal of an enabled interaction and is analogous to [ADD].

Example 3. Consider the reconfigurable BI(P) system in Example 2. The initial state in which no connection has been established is given by the term $\varnothing(\mathtt{D}_S, \mathtt{D}_{C_1}, \mathtt{D}_{C_2})$. In this state, the system can start a session between the `Server` and either $\mathtt{Client}_1$ or $\mathtt{Client}_2$. Assuming that a session with $\mathtt{Client}_1$ is established, then the system can move as follows

$$\varnothing(\mathtt{D}_S, \mathtt{D}_{C_1}, \mathtt{D}_{C_2}) \xrightarrow{\mathtt{ab}_1+}\!\!\!\!\!\!\rightarrow\ s \quad \text{with } s = \{\mathtt{ab}_1\}(\mathtt{C}_S, \mathtt{C}_{C_1}, \mathtt{D}_{C_2})$$

After session initiation, `Server` and $\mathtt{Client}_1$ can repeatedly synchronise with interaction $\mathtt{ab}_1$ as any ordinary BI(P) system, i.e.,

$$s \xrightarrow{\mathtt{ab}_1}\!\!\!\!\!\!\rightarrow\ s \xrightarrow{\mathtt{ab}_1}\!\!\!\!\!\!\rightarrow\ \ldots \xrightarrow{\mathtt{ab}_1}\!\!\!\!\!\!\rightarrow\ s$$

At some point, both `Server` and $\mathtt{Client}_1$ decide to close the session and the system returns to the initial state by removing the interaction $\mathtt{ab}_1$, i.e.,

$$s \xrightarrow{\mathtt{ab}_1-}\!\!\!\!\!\!\rightarrow\ \varnothing(\mathtt{D}_S, \mathtt{D}_{C_1}, \mathtt{D}_{C_2})$$

3.1 Reconfigurable BI(P) in BI(P)

This section shows that Reconfigurable BI(P) is as expressive as BI(P), i.e., that adding the possibility of dynamically changing the set of interactions does not increase the expressiveness, even if a price is paid in terms of the combinatorial explosion of global states.

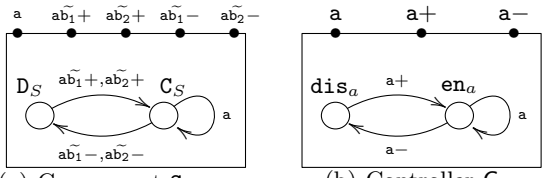
We start by introducing some auxiliary notation and definitions.

Let $B = (Q, P, \longrightarrow\!\!\!\!\!\!\rightarrow)$ be a reconfigurable component. The set of reconfigurable interactions of B is defined as follows

$$\mathcal{R}(B) = \{a \mid (q, a, \rho, q') \in \longrightarrow\!\!\!\!\!\!\rightarrow \text{ and } \rho \neq \epsilon\}$$

For any reconfigurable interaction of a BI(P) component, i.e., $a \in \mathcal{R}(B)$, we add two additional ports in the encoded component, which are denoted by $\widetilde{a}^B+$ and $\widetilde{a}^B-$. We remark that we add some decoration to the interaction a in order to avoid port clashes between the different components of a system. Note that the same dynamic interaction may appear in different components of a system (e.g., $\mathtt{ab}_1$ and $\mathtt{ab}_2$ in Fig. 2) and we need to ensure that a port appears in one component at most. Although different choices for decoration would be possible, we will use the following in the rest of the paper.

$$\widetilde{a}^B = (a \cap P) \cup \{\tilde{p} \mid p \in a \smallsetminus P\}.$$



(a) Component `Server` (b) Controller $\mathtt{C}_a$

Fig. 4. Encoding reconfigurable BI(P) in BI(P)

We write $\widetilde{\mathcal{R}(B)}$ for the set of all decorated reconfigurable interactions, i.e., $\widetilde{\mathcal{R}(B)} = \{\widetilde{a}^B \mid a \in \mathcal{R}(B)\}$. For example, $\widetilde{\mathcal{R}(\mathtt{Server})} = \{\widetilde{\mathtt{ab}_1}, \widetilde{\mathtt{ab}_2}\}$.

The notion of reconfigurable interaction is straightforwardly extended to reconfigurable BI(P) systems. Let $B = \gamma(B_1, \ldots, B_n)$ be a reconfigurable BI(P) system, then the set of reconfigurable interactions is defined as

$$\mathcal{R}(B) = \bigcup_{1 \leq i \leq n} \mathcal{R}(B_i).$$

Definition 6. *Let $B = (Q, P, \longrightarrow\!\!\!\!\!\rightarrow)$ be a reconfigurable component. The corresponding BI(P) component $[\![B]\!]$ is $(Q, P \cup (\widetilde{\mathcal{R}(B)} \times \{+, -\}), \rightarrow)$ with $\rightarrow$ defined such as $(q, a, q') \in \rightarrow$ iff*

- $(q, a, \rho, q') \in \longrightarrow\!\!\!\!\!\rightarrow$ *and* $\rho = \epsilon$*; or*
- $(q, a', \rho, q') \in \longrightarrow\!\!\!\!\!\rightarrow$*,* $\rho \neq \epsilon$ *and* $a = (\widetilde{a'}^B, \rho)$

Figure 4(a) shows the BI(P) component corresponding to the reconfigurable component `Server` depicted in Fig. 2. For simplicity, we write $\widetilde{a}^B\rho$ for a port $(\widetilde{a}^B, \rho)$, e.g., we write $\widetilde{\mathtt{ab}_1}+$ instead of $(\widetilde{\mathtt{ab}_1}, +)$. Note that we extend the interface of the component by adding two ports for each reconfigurable interaction, one for signalling the addition of the interaction and the other for the removal. Besides, the transition relation of the component remains unchanged but transitions corresponding to dynamic interactions are renamed to use the added ports.

The following result states a correspondence between the behaviour of a reconfigurable component and its encoded version.

Lemma 1. *Let $B = (Q, P, \longrightarrow\!\!\!\!\!\rightarrow)$ be a reconfigurable component and $[\![B]\!] = (Q, P, \rightarrow)$ its encoded version. Then, $q \xrightarrow{a\rho}\!\!\!\!\!\rightarrow\, q'$ if and only if*

- $\rho = \epsilon$ *and* $q \xrightarrow{a} q'$*, or*
- $\rho \neq \epsilon$ *and* $q \xrightarrow{\widetilde{a}^B\rho} q'$

Proof. It follows directly from the definition of $[\![B]\!]$.

We now address the encoding of the behaviour of reconfigurable interactions. We will associate any reconfigurable interaction with a BI(P) component that models the dynamics of an interaction that can be dynamically enabled and disabled.

Definition 7. *Let a be an interaction. A controller for a is the BI(P) component $\mathsf{C}_a = (Q_{\mathsf{C}_a}, P_{\mathsf{C}_a}, \rightarrow)$ defined in Fig. 4(b).*

Note that the net C_a has two places, one for an enabled interaction, named $\mathtt{a}_{\mathtt{en}}$, and the other for a disabled one, named $\mathtt{a}_{\mathtt{dis}}$. The only possible transition for a disable interaction is the enabling (i.e., $\mathtt{a}+$). After being enabled, the interaction can be used as a usual one ($\mathtt{a}$) until it is disabled ($\mathtt{a}-$). We remark that each of these behaviours is observed over a dedicated port.

Definition 8. *Let $B = \gamma(B_1, \ldots, B_n)$ be a reconfigurable BI(P) system with $\mathcal{R}(B) = \{a_0, \ldots, a_j\}$. The corresponding BI(P) system is defined as follows*

$$[\![\gamma(B_1, \ldots, B_n)]\!] = [\![\gamma]\!]([\![B_1]\!], \ldots, [\![B_n]\!], \mathsf{C}_{a_0}, \ldots, \mathsf{C}_{a_j})$$

where

$$[\![\gamma]\!] = (\gamma \setminus \mathcal{R}(B)) \cup (\bigcup_{a \in \mathcal{R}(B), \rho \in \{\epsilon, +, -\}} \{[\![a]\!]_\rho\})$$

with

$$[\![a]\!]_\rho = \begin{cases} \{a\rho\} \cup \{\widetilde{a}^{B_i}\rho \mid 1 \leq i \leq n \text{ and } a \in \mathcal{R}(B_i)\} & \text{if } \rho \in \{+, -\} \\ \{a\} \cup \{p \mid 1 \leq i \leq n \text{ and } p \in a \downarrow_{P_i}\} & \text{if } \rho = \epsilon \end{cases}$$

Moreover, any state $\gamma(q_1, \ldots, q_n)$ of B will be associated with a state $[\![\gamma(q_1, \ldots, q_n)]\!]$ of $[\![B]\!]$ where

$$[\![\gamma(q_1, \ldots, q_n)]\!] = (q_1, \ldots, q_n, s_0, \ldots, s_j)$$

with

$$s_i = \begin{cases} \mathtt{en}_{a_i} & \text{if } a_i \in \gamma \\ \mathtt{dis}_{a_i} & \text{if } a_i \notin \gamma \end{cases}$$

Example 4. The reconfigurable system introduced in Example 2 is encoded as the BI(P) system shown in Fig. 5, which contains five components: the encoded versions of the components $\mathtt{Server}$, $\mathtt{Client_1}$ and $\mathtt{Client_2}$, and the two interaction controllers (i.e., one for each reconfigurable interaction $\mathtt{ab_1}$ and $\mathtt{ab_2}$). The set γ contains six interactions, three for each reconfigurable interaction. The initial state of the system in Example 2 corresponds to $(\mathtt{D}_S, \mathtt{D}_{C_1}, \mathtt{D}_{C_2}, \mathtt{ab}_{\mathtt{1dis}}, \mathtt{ab}_{\mathtt{2dis}})$. Then, the transition that starts a session between $\mathtt{Server}$ and $\mathtt{Client_1}$ is simulated by

$$(\mathtt{D}_S, \mathtt{D}_{C_1}, \mathtt{D}_{C_2}, \mathtt{ab}_{\mathtt{1dis}}, \mathtt{ab}_{\mathtt{2dis}}) \xrightarrow{\{\mathtt{ab_1}+, \widetilde{\mathtt{ab_1}}+, \widetilde{\mathtt{ab}}_{\mathtt{1}}+\}}_\gamma (\mathtt{C}_S, \mathtt{C}_{C_1}, \mathtt{D}_{C_2}, \mathtt{ab}_{\mathtt{1en}}, \mathtt{ab}_{\mathtt{2dis}})$$

The synchronisation between $\mathtt{Server}$ and $\mathtt{Client_1}$ with the interaction $\mathtt{ab_1}$ is

$$(\mathtt{C}_S, \mathtt{C}_{C_1}, \mathtt{D}_{C_2}, \mathtt{ab}_{\mathtt{1en}}, \mathtt{ab}_{\mathtt{2dis}}) \xrightarrow{\{\mathtt{ab_1}, \mathtt{a}, \mathtt{b_1}\}}_\gamma (\mathtt{C}_S, \mathtt{C}_{C_1}, \mathtt{D}_{C_2}, \mathtt{ab}_{\mathtt{1en}}, \mathtt{ab}_{\mathtt{2dis}})$$

Similarly, $\mathtt{Server}$ and $\mathtt{Client_1}$ jointly disconnect with the following transition

$$(\mathtt{C}_S, \mathtt{C}_{C_1}, \mathtt{D}_{C_2}, \mathtt{ab}_{\mathtt{1en}}, \mathtt{ab}_{\mathtt{2dis}}) \xrightarrow{\{\mathtt{ab_1}-, \widetilde{\mathtt{ab_1}}-, \widetilde{\mathtt{ab}}_{\mathtt{1}}-\}}_\gamma (\mathtt{D}_S, \mathtt{D}_{C_1}, \mathtt{D}_{C_2}, \mathtt{ab}_{\mathtt{1dis}}, \mathtt{ab}_{\mathtt{2dis}})$$

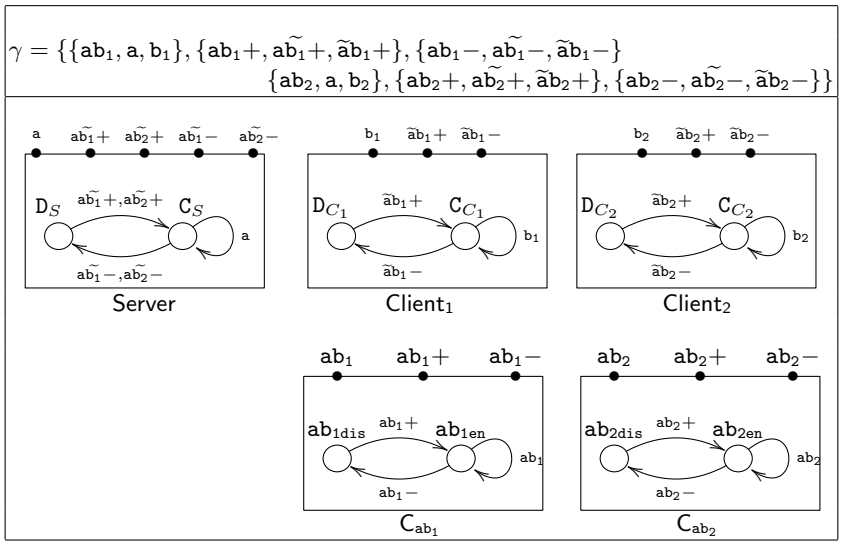


Fig. 5. A simple reconfigurable BI(P) system encoded in ordinary BI(P)

As illustrated by the above example, the transitions of a reconfigurable BI(P) system and its corresponding BI(P) system are in one-to-one correspondence, as formalised by the following result.

Theorem 1 (Correspondence). *Let* $B = \gamma(B_1, \ldots, B_n)$ *be a reconfigurable BI(P) system with* $\mathcal{R}(B) = \{a_0, \ldots, a_j\}$. *Then,* $\gamma(q_1, \ldots, q_n) \xrightarrow{a}\!\!\!\!\rightarrow \gamma'(q'_1, \ldots, q'_n)$ *iff* $\exists b \in \{a, \gamma_a, \gamma_{a+}, \gamma_{a-}\}$ *s.t.* $[\![\gamma(q_1, \ldots, q_n)]\!] \xrightarrow{b}_{[\![\gamma]\!]} [\![\gamma'(q'_1, \ldots, q'_n)]\!]$.

Proof. $\Rightarrow$) By case analysis on the derivation of $\gamma(q_1, \ldots, q_n) \xrightarrow{a}\!\!\!\!\rightarrow \gamma'(q'_1, \ldots, q'_n)$.
$\Leftarrow$) Follows by case analysis on the shape of a.

4 Dynamic BI(P)

In this section we further extend BI(P) by allowing the dynamic replication of components as result of the interaction of existing components. The idea is that upon certain interactions, where each involved component forks an instance of itself with some given initial state, some sort of session is established among (all and only) the spawned replicas that can thus interact in a sandboxed way, isolated from the rest of the system. For example, this is useful when the same server component must serve a possibly unbounded number of client requests separately but concurrently. As another example, some form of publish-subscribe mechanism can also be represented, where each subscriber has a dedicated notification handler. As explained in the Introduction, the mechanism underlying

dynamic BI(P) resembles, to some extent, the use of correlation tokens in web service computing.

Technically, we rely on an infinite set of port names $\mathcal{P}$ ranged over by $\mathtt{a}, \mathtt{b}, \ldots$, an infinite set of port variable names $\mathcal{X}$ ranged over by $x, y, \ldots$, and an infinite set of state names $\mathcal{Q}$ ranged over by $\mathtt{p}, \mathtt{q}, \ldots$. We assume $\mathcal{P}$, $\mathcal{Q}$ and $\mathcal{X}$ pairwise disjoint. As in general an interaction is related to a specific session, we sometimes decorate ports and interactions with specific correlation tokens as their subscripts. For example, for $a = \mathtt{ab}$ we write $a_{\mathtt{c}}$ for $\mathtt{a_c b_c}$.

Definition 9 (Dynamic Component). *A* dynamic component *is a tuple* $B = (Q, P, \to)$ *where* $Q \subset \mathcal{Q}$ *is a set of places,* $P \subset \mathcal{P}$ *is a set of ports, and* $\to$ *is a finite set of transitions, each having one of the following shapes:*

- $\mathtt{q}(x) \xrightarrow{a_x} \mathtt{q}'(x)$*, i.e., (a coloured version of) a BI(P) transition;*
- $\mathtt{q}(x) \xrightarrow{a_x y+} \mathtt{q}'(x) \ \oplus \ \mathtt{q}''(y)$*, i.e., a port creation;*
- $\mathtt{q}(x) \xrightarrow{x-} \varnothing$*, i.e., a port removal;*
- $\mathtt{q}(x) \xrightarrow{x} \mathtt{q}'(x)$*, i.e., an interaction over a dynamically created port.*

Ports that appear in labels of the form a_x are parametric to the correlation token and are called *static* ports; the other ports are called *dynamic*. We assume static ports cannot be used as correlation tokens. In the following we denote by P_x the set of static ports of P, by $P_{\mathtt{a}}$ the set of static ports in P parametrized by the token $\mathtt{a}$ and by $\delta(P)$ the set of dynamic ports. For example, if $P = \{\mathtt{a}, \mathtt{b}\}$ with $\mathtt{a}$ static and $\mathtt{b}$ dynamic, then $P_{\mathtt{c}} = \{\mathtt{a_c}\}$. Note that if all transitions have the form $\mathtt{q}(x) \xrightarrow{a_x} \mathtt{q}'(x)$ then B is essentially an ordinary BI(P) component.

The current state of a dynamic component $B = (Q, P, \to)$ takes the form $\langle P, f \rangle$ with $P \subset \mathcal{P}$ defining the current ports of the component (that includes opened sessions) and $f : Q \to 2^P$ such that $f(\mathtt{q}_1) \cap f(\mathtt{q}_2) = \varnothing$ for $\mathtt{q}_1 \neq \mathtt{q}_2$. The function f represents the current internal state of the component replicas. For example, if $f(\mathtt{q}) = \{\mathtt{a}, \mathtt{b}\}$ then there are two replicas of the component, one involved in session $\mathtt{a}$ and one in $\mathtt{b}$ both with current state $\mathtt{q}$. The condition $f(\mathtt{q}_1) \cap f(\mathtt{q}_2) = \varnothing$ for $\mathtt{q}_1 \neq \mathtt{q}_2$ guarantees that each replica is associated with a different session and that to each session corresponds exactly one state.

As a matter of notation we denote $f \oplus \mathtt{p(a)}$ the function defined as

$$(f \oplus \mathtt{p(a)})(\mathtt{q}) = \begin{cases} f(\mathtt{q}) & \text{if } \mathtt{q} \neq \mathtt{p} \\ f(\mathtt{q}) \cup \{\mathtt{a}\} & \text{if } \mathtt{q} = \mathtt{p} \end{cases}$$

Remark 1. Initially there is only one session opened for each component, i.e., for each component there is only one state $\mathtt{p}$ such that $f(\mathtt{p}) \neq \varnothing$ and such $f(\mathtt{p})$ must be a singleton. To shorten the notation but without loss of generality, we shall assume that such initial session identifier is void, i.e. $f(\mathtt{p}) = \{\bullet\}$ and omit the corresponding port $\bullet$ from the drawing of components.

The operational semantics of components is given by the three rules in Fig. 6.

The first rule ([Cint]) deals with both: i) the case of an ordinary interaction $a_{\mathtt{a}}$ (here coloured by the token $\mathtt{a}$); and ii) the case of a dynamic interaction over the session associated with $\mathtt{a}$.

$$\frac{\mathtt{q}(x) \xrightarrow{\alpha} \mathtt{q}'(x) \qquad \mathtt{a} \in \delta(P) \qquad \alpha \in \{a_x, x\}}{\langle P, \mathtt{q}(\mathtt{a}) \oplus f\rangle \xrightarrow{\alpha\{\mathtt{a}/x\}} \langle P, \mathtt{q}'(\mathtt{a}) \oplus f\rangle}\,[\textsc{Cint}]$$

$$\frac{\mathtt{q}(x) \xrightarrow{a_x y+} \mathtt{q}'(x) \oplus \mathtt{q}''(y) \qquad \mathtt{a} \in \delta(P) \qquad \mathtt{b} \notin P}{\langle P, \mathtt{q}(\mathtt{a}) \oplus f\rangle \xrightarrow{a_\mathtt{a}\mathtt{b}+} \langle P \cup \{\mathtt{b}\} \cup P_\mathtt{b}, \mathtt{q}'(\mathtt{a}) \oplus \mathtt{q}''(\mathtt{b}) \oplus f\rangle}\,[\textsc{Copen}]$$

$$\frac{\mathtt{q}(x) \xrightarrow{x-} \varnothing \qquad \mathtt{a} \in \delta(P)}{\langle P, q(\mathtt{a}) \oplus f\rangle \xrightarrow{\mathtt{a}-} \langle P \setminus (\{\mathtt{a}\} \cup P_\mathtt{a}), f\rangle}\,[\textsc{Cclose}]$$

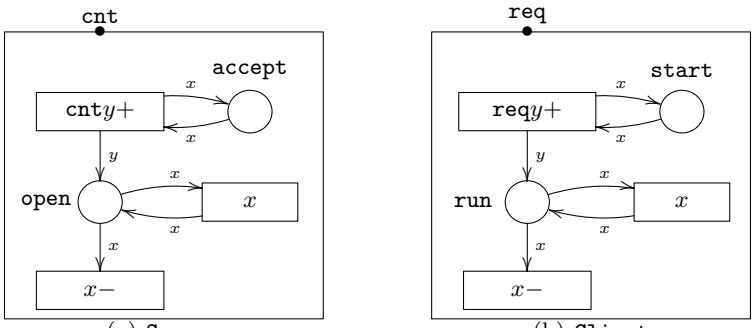
Fig. 6. Operational semantics of dynamic components

The second rule ([Copen]) is the most complex one, as it deals with component spawning and port creation. Here the freshly created session identifier is `b`, which is then used as a fresh dynamic port, together with suitable instances $P_\mathtt{b}$ of the static ports of the component. The spawned instance of the component has initial state $\mathtt{q}''(\mathtt{b})$. Ports in $P_\mathtt{b}$ will allow the spawned instance of the component to interact on static ports with some other spawned components that are part of the same session. Moreover, the spawned instance of the component will be able to interact on the port `b` by synchronising with all the other spawned components that are part of the same session. Note that although the token `b` has been created within the session `a`, such information is not maintained in the state, i.e., sessions `a` and `b` will run independently.

Finally, the third rule ([Cclose]) deals with session closure, where the token `a` and all the ports $\{\mathtt{a}\} \cup P_\mathtt{a}$ associated with the closed session `a` are discarded.

Example 5. Consider a server component that interacts with a possibly unbounded number of clients by keeping different/separate sessions. Any session starts with a client request for a new connection. After the initial connection, each client synchronises with the server by using a dedicated, private port until the client disconnects from the server. This behaviour can be modelled as the component depicted in Fig. 7(a). We rely on the standard graphical representation of coloured Petri nets, in which places are represented by circles and transitions are drawn as rectangles connected to their pre and post-set by direct arcs which are decorated with the colours of the involved tokens. In addition, we show the ports of the component as bullets drawn on the boundaries, like in BIP notation. The component in Fig. 7(a) has one static port `cnt`, two places `accept` and `open` with the following three transitions:

- $t_0 = \mathtt{accept}(x) \xrightarrow{\mathtt{cnt}_x\, y+} \mathtt{accept}(x) \oplus \mathtt{open}(y)$: if the server can accept a new connection (i.e., a token can be consumed from the place `accept`), then it performs the action `cnt` that creates a new dynamic port (to be associated with the symbol y). After performing this action, the server will still accept new connections because the token x is put back to the place `accept`. Now, the component has a new session (i.e., a dedicated port) for interacting with



(a) `Server` (b) `Client`

Fig. 7. Two dynamic components

the recently connected client as represented by the token y containing the fresh created port name in the place `open`.

- $t_1 = \texttt{open}(x) \xrightarrow{x} \texttt{open}(x)$: For any open session x, the server can repeatedly perform an action on the corresponding dynamic port. This transition does not alter the set of ports of the component.
- $t_2 = \texttt{open}(x) \xrightarrow{x-} \varnothing$: An already opened session x is closed after performing the corresponding action $x-$, that synchronises with a request from the client to close the same session.

The component modelling the behaviour of a client is depicted in Fig. 7(b), which is analogous to Fig. 7(a).

Definition 10 (Dynamic BI(P) system). *A* dynamic BI(P) system $B = \gamma(B_1, \ldots, B_n)$ *is the composition of a finite set* $\{B_i\}_{i=1}^n$ *of dynamic BI(P) components* $B_i = (Q_i, P_i, \rightarrow_i)$ *such that their sets of ports are pairwise disjoint, i.e.,* $P_i \cap P_j = \varnothing$ *for* $i \neq j$*, parametrized by a set* $\gamma \subset 2^P$ *of interactions over the set of ports* $P = \biguplus_{i=1}^n P_i$*.*

Without loss of generality, we assume that for any $a \in \gamma$ it is either the case that a contains static ports only and we call it *static* or it contains no static port at all and we call it *dynamic*. Moreover, if $a \downarrow_{P_i}$ is made of static ports, then $a \downarrow_{P_i} = a'_{\mathtt{a}_i}$ for some a' and $\mathtt{a}_i \in P_i$, i.e., all static ports in $a \downarrow_{P_i}$ are parametrized by the same session identifier $\mathtt{a}_i$. In such case, we let $ids_i(a)$ denote $\mathtt{a}_i$

In the following we write $I(a)$ to denote the set $\{i \mid \neg(a \# P_i)\}$ of indices of the components involved in a and $\overline{I(a)}$ to denote its complement $[1, n] \setminus I(a) = \{i \mid a \# P_i\}$. If a is static, we denote by $ids(a)$ the set $\{ids_i(a) \mid i \in I(a)\}$, otherwise we let $ids(a) = \varnothing$.

Given a set of substitutions $\sigma = \{\mathtt{b}_i/\mathtt{a}_i\}_{i \in I}$ and a static interaction $a \in \gamma$ such that $ids(a) \subseteq \{\mathtt{a}_i\}_{i \in I}$ we write a_σ for the interaction obtained by replacing in a each parameter $\mathtt{a}_i$ by the corresponding parameter $\mathtt{b}_i$. Moreover, we write γ_σ for the set of renamed static interactions $\{a_\sigma \mid a \in \gamma \wedge ids(a) \subseteq \{\mathtt{a}_i\}_{i \in I}\}$.

$$\frac{a \in \gamma \qquad \forall i.s_i \xrightarrow{a\downarrow_{P_i}} s'_i}{\gamma(s_1,\ldots,s_n) \xrightarrow{a} \gamma(s'_1,\ldots,s'_n)}\,[\textsc{Sint}]$$

$$\frac{\begin{array}{lll} a \in \gamma & i \in I(a) \implies s_i \xrightarrow{a\downarrow_{P_i}\mathtt{b_i}+} s'_i & \mathtt{b_i} \text{ fresh} \\ & i \in \overline{I(a)} \implies s'_i = s_i & \sigma = \{\mathtt{b}_i/ids_i(a)\}_{i\in I(a)} \end{array}}{\gamma(s_1,\ldots,s_n) \xrightarrow{a} (\gamma \cup \{\mathtt{b_i}\}_{i\in I(a)} \cup \gamma_\sigma)(s'_1,\ldots,s'_n)}\,[\textsc{Sopen}]$$

$$\frac{a \in \gamma \qquad i \in I(a) \implies s_i \xrightarrow{a\downarrow_{P_i}-} s'_i \qquad i \in \overline{I(a)} \implies s'_i = s_i}{\gamma(s_1,\ldots,s_n) \xrightarrow{a} (\gamma \ominus a)(s'_1,\ldots,s'_n)}\,[\textsc{Sclose}]$$

Fig. 8. Operational semantics of dynamic BI(P) systems

Finally, given a dynamic interaction a we write $\gamma \ominus a$ for the set of interactions in γ where the ports in a do not appear. Formally, $\gamma \ominus a = \{a' \in \gamma \mid a' \cap a = \varnothing \wedge ids(a') \cap a = \varnothing\}$

Let s_i range over $2^{P_i} \times P_i^{Q_i}$ representing a generic state of the component B_i. The semantics of a dynamic BI(P) system $\gamma(B_1,\ldots,B_n)$ is defined by the three rules in Fig. 8.

Example 6. Consider the dynamic BI(P) components introduced in Example 5. We illustrate one possible run of the server with two clients in Fig. 9. Roughly, it corresponds to the series of transitions in Fig. 10, where $\gamma, \gamma', \gamma''$ are the ones indicated in Fig. 9. The first transition is obtained by combining the server transition

$$\langle\{\mathtt{cnt}\}, \mathtt{accept}(\bullet)\rangle \xrightarrow{\mathtt{cnt\,v}+} \langle\{\mathtt{cnt}, \mathtt{cnt_v}, \mathtt{v}\}, \mathtt{accept}(\bullet) \oplus \mathtt{open}(\mathtt{v})\rangle$$

with the following transition of the first client:

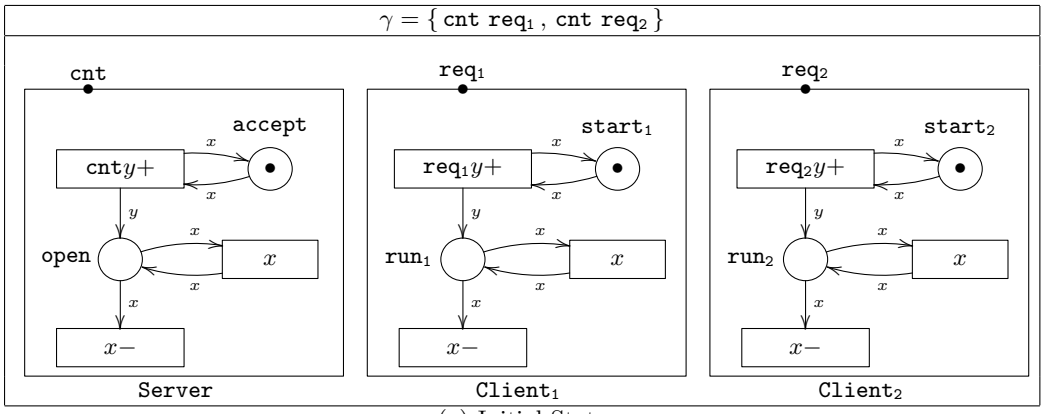
$$\langle\{\mathtt{req_1}\}, \mathtt{start_1}(\bullet)\rangle \xrightarrow{\mathtt{req_1\,m}+} \langle\{\mathtt{req_1}, \mathtt{req_{1m}}, \mathtt{m}\}, \mathtt{start_1}(\bullet) \oplus \mathtt{run_1}(\mathtt{m})\rangle$$

Analogously, for the second transition. Note that suitable replicas $\mathtt{cnt_v}$, $\mathtt{cnt_w}$, $\mathtt{req_{1m}}$, $\mathtt{req_{2n}}$ of the static ports $\mathtt{cnt}$, $\mathtt{req_1}$, $\mathtt{req_2}$ have been created locally to each component, and that the set of interactions has been enriched with suitable replicas $\mathtt{cnt_v\,req_{1m}}$ and $\mathtt{cnt_w\,req_{2n}}$ of the static interactions $\mathtt{cnt\,req_1}$ and $\mathtt{cnt\,req_2}$ together with freshly created dynamic interactions $\mathtt{v\,m}$ and $\mathtt{w\,n}$.

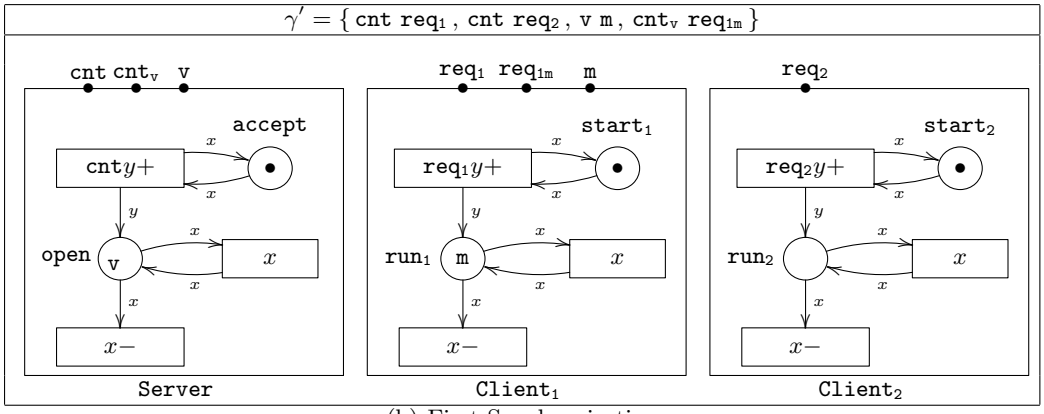
Let s denote the last state reached. Then, the server can interact with the clients by performing the interactions $\mathtt{v\,m}$ and $\mathtt{w\,n}$ as many times as needed, with the system remaining in the same state s:

$$s \xrightarrow{\mathtt{v\,m}} s \xrightarrow{\mathtt{w\,n}} s \cdots$$

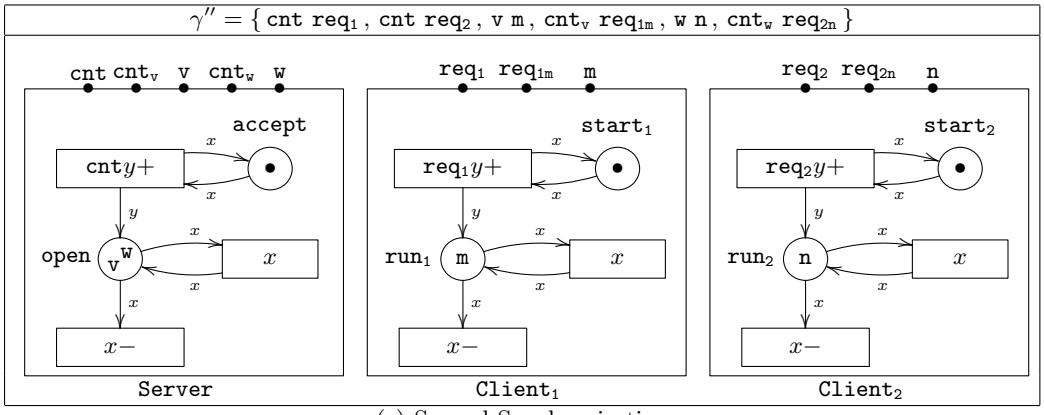
The above transitions are obtained by combining dynamic transitions of the server (labels $\mathtt{v}$ and $\mathtt{w}$) with dynamic transitions of the clients (labels $\mathtt{m}$ and $\mathtt{n}$).



(a) Initial State



(b) First Synchronisation



(c) Second Synchronisation

Fig. 9. A run of the server with two clients

$$\gamma \left(\begin{array}{l} \langle \{\mathtt{cnt}\}, \mathtt{accept}(\bullet) \rangle , \\ \langle \{\mathtt{req}_1\}, \mathtt{start}_1(\bullet) \rangle , \\ \langle \{\mathtt{req}_2\}, \mathtt{start}_2(\bullet) \rangle \end{array} \right)$$
$$\xrightarrow{\mathtt{cnt}\,\mathtt{req}_1} \gamma' \left(\begin{array}{l} \langle \{\mathtt{cnt}, \mathtt{cnt}_\mathtt{v}, \mathtt{v}\}, \mathtt{accept}(\bullet) \oplus \mathtt{open}(\mathtt{v}) \rangle , \\ \langle \{\mathtt{req}_1, \mathtt{req}_{1\mathtt{m}}, \mathtt{m}\}, \mathtt{start}_1(\bullet) \oplus \mathtt{run}_1(\mathtt{m}) \rangle , \\ \langle \{\mathtt{req}_2\}, \mathtt{start}_2(\bullet) \rangle \end{array} \right)$$
$$\xrightarrow{\mathtt{cnt}\,\mathtt{req}_2} \gamma'' \left(\begin{array}{l} \langle \{\mathtt{cnt}, \mathtt{cnt}_\mathtt{v}, \mathtt{v}, \mathtt{cnt}_\mathtt{w}, \mathtt{w}\}, \mathtt{accept}(\bullet) \oplus \mathtt{open}(\mathtt{v}) \oplus \mathtt{open}(\mathtt{w}) \rangle , \\ \langle \{\mathtt{req}_1, \mathtt{req}_{1\mathtt{m}}, \mathtt{m}\}, \mathtt{start}_1(\bullet) \oplus \mathtt{run}_1(\mathtt{m}) \rangle , \\ \langle \{\mathtt{req}_2, \mathtt{req}_{2\mathtt{n}}, \mathtt{n}\}, \mathtt{start}_2(\bullet) \oplus \mathtt{run}_2(\mathtt{n}) \rangle \end{array} \right)$$

Fig. 10. Transitions representing a run of the server with two clients

Finally, we illustrate the case when the session between the server and the second client is closed:

$$s \xrightarrow{\mathtt{w}\,\mathtt{n}} \gamma' \left(\begin{array}{l} \langle \{\mathtt{cnt}, \mathtt{cnt}_\mathtt{v}, \mathtt{v}\}, \mathtt{accept}(\bullet) \oplus \mathtt{open}(\mathtt{v}) \rangle , \\ \langle \{\mathtt{req}_1, \mathtt{req}_{1\mathtt{m}}, \mathtt{m}\}, \mathtt{start}_1(\bullet) \oplus \mathtt{run}_1(\mathtt{m}) \rangle , \\ \langle \{\mathtt{req}_2\}, \mathtt{start}_2(\bullet) \rangle \end{array} \right)$$

The above transition is obtained by combining a closing transitions of the server (label $\mathtt{w}-$) with a closing transition of the second client (label $\mathtt{n}-$). Note that the set of ports of the server and of the second client are updated consequently, by removing all ports that refer to the session identifiers $\mathtt{w}$ and $\mathtt{n}$. Similarly, the set of interactions is $\gamma' = \gamma'' \ominus \mathtt{w}\ \mathtt{n}$.

4.1 Dynamic BI(P) vs BI(P) vs P/T nets

Unlike reconfigurable BI(P) systems, dynamic BI(P) systems are strictly more expressive than ordinary BI(P) systems. This can be immediately seen by noting that BI(P) systems are finite state (see, e.g., [8], where it was shown that any BI(P) system corresponds to a safe Petri net), while this is not the case for dynamic BI(P) systems (see, e.g., Example 6).

In this section we outline a correspondence between dynamic BI(P) systems and Place/Transition Petri nets. This is interesting because: i) it shows that properties like reachability remains decidable and ii) it draws a nice analogy with the correspondence between ordinary BI(P) systems and safe Petri nets shown in [8].

Roughly, given a dynamic BI(P) system $B = \gamma(B_1, ..., B_n)$ we define a P/T Petri net $N(B)$ whose places are tuples of states from components $B_1, ..., B_n$ and whose transitions represent the possible interactions. Note that $N(B)$ is determined statically and may contain more places and transitions than those strictly necessary, i.e., $N(B)$ may contains places that will never be marked as well as transitions that will never be enabled. Still $N(B)$ is finite and it is neither an ordinary automata nor a safe Petri net because: i) it may contain transitions that are attached to two output places; and ii) during a run it may produce more than one token in the same place.

Places of $N(B)$. The places of the net will be named like pairs $(I, \{\mathtt{p}_i\}_{i\in I})$, where $\varnothing \subset I \subseteq [1,n]$ and $\forall i \in I.\mathtt{p}_i \in Q_i$. Intuitively, the place $(I, \{\mathtt{p}_i\}_{i\in I})$ represents a session that involves replicas of the components B_i such that $i \in I$, where the ith component is in state $\mathtt{p}_i$.

The initial marking of $N(B)$ has one token in the place $([1,n], \{\mathtt{p}_{0i}\}_{i\in[1,n]})$, where $\mathtt{p}_{0i}$ denotes the initial state of the ith component.

Transitions of $N(B)$. The transitions will be named like pairs (p,t) for p a place like described above and t one of the following: an ordinary interaction a or an interaction with spawning a^+, with $a \in \gamma$; a dynamic interaction I or a closing interaction I^- with $\varnothing \subset I \subseteq [1,n]$. However not all combinations of p and t are considered. A pair (p,t) where $p = (I, \{\mathtt{p}_i\}_{i\in I})$ is included in $N(B)$ if:

1. $t = a$, $I(a) \subseteq I$ and for all $i \in I(a)$ then $\mathtt{p}_i(x) \xrightarrow{a_{ix}} \mathtt{q}_i(x)$ is a transition of B_i with $a \downarrow_{P_i} = a_{i\,ids_i(a)}$. In this case we let ${}^\bullet(p,t) = p$ and $(p,t)^\bullet = (I, \{\mathtt{q}_i\}_{i\in I})$ such that $\mathtt{q}_i = \mathtt{p}_i$ whenever $i \in I \setminus I(a)$.
2. $t = a^+$, $I(a) \subseteq I$ and for all $i \in I(a)$ then $\mathtt{p}_i(x) \xrightarrow{a_{ix}y+} \mathtt{q}_i(x) \oplus \mathtt{q'}_i(y)$ is a transition of B_i with $a \downarrow_{P_i} = a_{i\,ids_i(a)}$. In this case we let ${}^\bullet(p,t) = p$ and $(p,t)^\bullet = (I, \{\mathtt{q}_i\}_{i\in I}) \oplus (I(a), \{\mathtt{q'}_i\}_{i\in I(a)})$ such that $\mathtt{q}_i = \mathtt{p}_i$ whenever $i \in I \setminus I(a)$.
3. $t = I$ and for all $i \in I$ then $\mathtt{p}(x) \xrightarrow{x} \mathtt{q}(x)$ is a dynamic transitions of B_i. In this case we let ${}^\bullet(p,t) = p$ and $(p,t)^\bullet = (I, \{\mathtt{q}_i\}_{i\in I})$.
4. $t = I^-$ and for all $i \in I$ then $\mathtt{p}(x) \xrightarrow{x-} \varnothing$ is a dynamic transitions of B_i. In this case we let ${}^\bullet(p,t) = p$ and $(p,t)^\bullet = \varnothing$.

Example 7. Consider the dynamic BI(P) system B introduced in Example 6. The corresponding P/T Petri net $N(B)$ contains, e.g., the places:

$(\{1\}, \{\mathtt{accept}\})$ $(\{1\}, \{\mathtt{open}\})$ $(\{2\}, \{\mathtt{start}_1\})$ $(\{2\}, \{\mathtt{run}_1\})$ $(\{3\}, \{\mathtt{start}_2\})$
$(\{3\}, \{\mathtt{run}_2\})$ $(\{1,2\}, \{\mathtt{accept}, \mathtt{start}_1\})$ $(\{1,2\}, \{\mathtt{accept}, \mathtt{run}_1\})$
$(\{1,2\}, \{\mathtt{open}, \mathtt{start}_1\})$ $(\{1,2\}, \{\mathtt{open}, \mathtt{run}_1\})$ $(\{1,3\}, \{\mathtt{accept}, \mathtt{start}_2\})$
$(\{1,3\}, \{\mathtt{accept}, \mathtt{run}_2\})$ $(\{1,3\}, \{\mathtt{open}, \mathtt{start}_2\})$ $(\{1,3\}, \{\mathtt{open}, \mathtt{run}_2\})$
$(\{2,3\}, \{\mathtt{start}_1, \mathtt{start}_2\})$ $(\{2,3\}, \{\mathtt{start}_1, \mathtt{run}_2\})$ $(\{2,3\}, \{\mathtt{run}_1, \mathtt{start}_2\})$
$(\{2,3\}, \{\mathtt{run}_1, \mathtt{run}_2\})$ $(\{1,2,3\}, \{\mathtt{accept}, \mathtt{start}_1, \mathtt{start}_2\})$ ...

The initial marking of $N(B)$ is $(\{1,2,3\}, \{\mathtt{accept}, \mathtt{start}_1, \mathtt{start}_2\})$.
The net $N(B)$ contains the transitions:

$((\{1\}, \{\mathtt{open}\}), \{1\})$ $((\{1,2,3\}, \{\mathtt{accept}, \mathtt{start}_1, \mathtt{start}_2\}), (\mathtt{cnt}\,\mathtt{req}_1)^+)$
$((\{1\}, \{\mathtt{open}\}), \{1\}^-)$ $((\{1,2,3\}, \{\mathtt{accept}, \mathtt{start}_1, \mathtt{run}_2\}), (\mathtt{cnt}\,\mathtt{req}_1)^+)$
$((\{2\}, \{\mathtt{run}_1\}), \{2\})$ $((\{1,2,3\}, \{\mathtt{accept}, \mathtt{start}_1, \mathtt{start}_2\}), (\mathtt{cnt}\,\mathtt{req}_2)^+)$
$((\{2\}, \{\mathtt{run}_1\}), \{2\}^-)$ $((\{1,2,3\}, \{\mathtt{accept}, \mathtt{run}_1, \mathtt{start}_2\}), (\mathtt{cnt}\,\mathtt{req}_2)^+)$
$((\{3\}, \{\mathtt{run}_2\}), \{3\})$ $((\{1,2,3\}, \{\mathtt{open}, \mathtt{run}_1, \mathtt{run}_2\}), \{1,2,3\})$
$((\{3\}, \{\mathtt{run}_2\}), \{3\}^-)$ $((\{1,2,3\}, \{\mathtt{open}, \mathtt{run}_1, \mathtt{run}_2\}), \{1,2,3\}^-)$
$((\{1,2\}, \{\mathtt{open}, \mathtt{run}_1\}), \{1,2\})$ $((\{1,2\}, \{\mathtt{accept}, \mathtt{start}_1\}), (\mathtt{cnt}\,\mathtt{req}_1)^+)$
$((\{1,3\}, \{\mathtt{open}, \mathtt{run}_2\}), \{1,3\})$ $((\{1,3\}, \{\mathtt{accept}, \mathtt{start}_2\}), (\mathtt{cnt}\,\mathtt{req}_2)^+)$
$((\{1,2\}, \{\mathtt{open}, \mathtt{run}_1\}), \{1,2\}^-)$ $((\{1,3\}, \{\mathtt{open}, \mathtt{run}_2\}), \{1,3\}^-)$
$((\{2,3\}, \{\mathtt{run}_1, \mathtt{run}_2\}), \{2,3\})$ $((\{2,3\}, \{\mathtt{run}_1, \mathtt{run}_2\}), \{2,3\}^-)$

Since for all transitions (p,t) we have ${}^\bullet(p,t) = p$, we omit to define the presets of transitions. Since for all transitions of the form (p,I) we have $(p,I)^\bullet = p$ and for all transitions of the form (p,I^-) we have $(p,I^-)^\bullet = \varnothing$, we omit to define the postsets of such transitions. Then we have:

$$(p,(\mathtt{cnt}\,\mathtt{req}_1)^+)^\bullet = p \oplus (\{1,2\},\{\mathtt{open},\mathtt{run}_1\})$$
$$(p,(\mathtt{cnt}\,\mathtt{req}_2)^+)^\bullet = p \oplus (\{1,3\},\{\mathtt{open},\mathtt{run}_2\})$$

As already said, $N(B)$ can contain dead transitions (never enabled) as well as dead places (never marked). A simple inspection reveals that the only places and transitions that are not dead are:

$p_1 = (\{1,2,3\},\{\mathtt{accept},\mathtt{start}_1,\mathtt{start}_2\})$	$(p_1,(\mathtt{cnt}\,\mathtt{req}_1)^+)$	$(p_1,(\mathtt{cnt}\,\mathtt{req}_2)^+)$
$p_2 = (\{1,2\},\{\mathtt{open},\mathtt{run}_1\})$	$(p_2,\{1,2\})$	$(p_2,\{1,2\}^-)$
$p_3 = (\{1,3\},\{\mathtt{open},\mathtt{run}_2\})$	$(p_3,\{1,3\})$	$(p_3,\{1,3\}^-)$

Theorem 2. *Reachability is decidable for any dynamic BI(P) system B.*

Proof. The proof exploits the P/T Petri net encoding $N(B)$ of B and the fact that reachability is decidable for P/T Petri nets.

Let $s = \gamma(s_1,...,s_n)$ be a state of B with $s_i = \langle P_i, f_i\rangle$ for $i \in [1,n]$, and let $\delta(\gamma)$ denote the set of dynamic interactions of γ (i.e., the opened sessions). Then we denote by $N(s)$ the marking defined as follows:

$$N(s) = \bigoplus_{a\in\delta(\gamma)} (I(a), \{\mathtt{p}_i \in B_i \mid i \in I(a) \wedge f_i(\mathtt{p}_i) \cap a \neq \varnothing\})$$

Next, we prove separately the two implications:

1. if there is a transition from s to s' in B, then there is a transition from $N(s)$ to $N(s')$ in $N(B)$;
2. if there is a transition from $N(s)$ to m in $N(B)$, then there is a state s' such that here is a transition from s to s' in B.

1st implication. Assume there is a transition $s \xrightarrow{a} s'$ in B. Then, there are four cases to consider, but due to space limitation, we show only the second case ([Sint]), which is the more interesting one.

- $s \xrightarrow{a} s'$ is a static transition obtained via rule [Sint].
- $s \xrightarrow{a} s'$ is a dynamic transition obtained via rule [Sint]. Since $a \in \gamma$, then there is a token $p = (I(a), \{\mathtt{p}_i \in B_i \mid i \in I(a) \wedge f_i(\mathtt{p}_i) \cap a \neq \varnothing\}) \in N(s)$. Moreover, there must be transitions $\mathtt{p}_i(x) \xrightarrow{x} \mathtt{q}_i(x)$ of B_i for each $i \in I(a)$. Therefore, there is a transition $(p, I(a))$ with ${}^\bullet(p,I(a)) = p$ and $(p,I(a))^\bullet = (I(a), \{\mathtt{q}_i\}_{i\in I(a)})$. It is immediate to see that $(p,I(a))$ is enabled in $N(s)$ and that its firing leads to $N(s')$.
- $s \xrightarrow{a} s'$ is a spawning transition obtained via rule [Sopen].
- $s \xrightarrow{a} s'$ is a closing transition obtained via rule [Sclose].

2nd implication. Assume that there is a transition (p,t) in $N(B)$ that is enabled in $N(s)$ and whose firing leads to a marking m. Since t is enabled, then $p \in N(s)$, i.e., it must be the case that $p = (I(a'), \{\mathtt{p}_i \in B_i \mid i \in I(a') \wedge f_i(\mathtt{p}_i) \cap a' \neq \varnothing\})$ for some $a' \in \delta(\gamma)$. Then, there are four cases to consider, but due to space limitation, we show only the third case, which is the more interesting one..

- $t = a$, $I(a) \subseteq I(a')$ and $\forall i \in I(a)$ then $\mathtt{p}_i(x) \xrightarrow{a_{ix}} \mathtt{q}_i(x)$ is a transition of B_i with $a \downarrow_{P_i} = a_{i\,ids_i(a)}$.
- $t = a^+$, $I(a) \subseteq I(a')$ and $\forall i \in I(a)$ then $\mathtt{p}_i(x) \xrightarrow{a_{ix}y+} \mathtt{q}_i(x) \oplus \mathtt{q'}_i(y)$ is a transition of B_i with $a \downarrow_{P_i} = a_{i\,ids_i(a)}$.
- $t = I(a')$ and $\forall i \in I(a')$ then $\mathtt{p}(x) \xrightarrow{x} \mathtt{q}(x)$ is a dynamic transitions of B_i. In this case we have ${}^\bullet(p,t) = p$ and $(p,t)^\bullet = (I(a'), \{\mathtt{q}_i\}_{i \in I(a')})$. Then, by rule [CINT], we know that $a' \downarrow_{P_i} = \mathtt{a}_i \in \delta(P_i)$ and $s_i = (P_i, \mathtt{p}_i(\mathtt{a}_i) \oplus f'_i) \xrightarrow{\mathtt{a}_i} (P_i, \mathtt{q}_i(\mathtt{a}_i) \oplus f'_i) = s'_i$ for each $i \in I(a')$ and by rule [SINT] we have that $s \xrightarrow{a'} \gamma(s'_1, ..., s'_n) = s'$ for $s'_i = s_i$ when $i \in [1,n] \setminus I(a')$. It is immediate to see that $N(s') = m$.
- $t = I(a')^-$ and $\forall i \in I(a')$ then $\mathtt{p}(x) \xrightarrow{x-} \varnothing$ is a dynamic transitions of B_i.

The thesis follow by the fact that reachability is decidable for $N(B)$. □

5 Concluding remarks

In this paper we have investigated two suitable extensions of BI(P) with dynamically defined behaviour and interaction. The first extension, called reconfigurable BI(P), has evolved from a previous proposal of the VERIMAG research group within the project ASCENS, before Dy-BIP was proposed in [6]. Here we prove that reconfigurable BI(P) is equally expressive as ordinary BI(P). The second extension, called dynamic BI(P), has been inspired by the use of correlation sets in web services and can be used to define systems with infinitely many states (contrary to ordinary BI(P) systems), but ensures that state reachability is decidable. Therefore, both extensions still preserve key BI(P) features in terms of analysis and verification. Notably the encodings exploited in our expressiveness results are obtained without a considerable change of the basic components, in the spirit of the glue expressiveness introduced in [3].

With respect to Dy-BIP, we think dynamic BI(P) has some advantages. While Dy-BIP imposes ad hoc restrictions (e.g., transitions of atomic components are labelled with only one single local port instead of a set of local ports) and extensions (e.g. transitions of atomic components are decorated with non-local architecture constraints that may involve port names of other components, thus compromising the modularity of the specification and moreover history variables are introduced to store the identity of interacting components), this is not necessary for dynamic BI(P). Furthermore, the number of component instances cannot change in Dy-BIP, contrary to dynamic BI(P).

In the future we plan to study the interplay between probabilities, priorities and dynamics, possibly in the compositional setting offered by the algebra of

Petri nets with boundaries [9]. We are confident that our proposals can fit well with the priorities based on the offer predicate semantics defined in [5].

Acknowledgments: We thank Simon Bliudze for several suggestions and comments on a preliminary version of this paper.

References

1. Arbab, F., Bruni, R., Clarke, D., Lanese, I., Montanari, U.: Tiles for Reo. In: WADT'08. LNCS, vol. 5486, pp. 37–55. Springer (2009)
2. Basu, A., Bozga, M., Sifakis, J.: Modeling heterogeneous real-time components in BIP. In: Fourth IEEE International Conference on Software Engineering and Formal Methods (SEFM 2006). pp. 3–12. IEEE Computer Society (2006)
3. Bliudze, S., Sifakis, J.: A notion of glue expressiveness for component-based systems. In: CONCUR. LNCS, vol. 5201, pp. 508–522. Springer (2008)
4. Bliudze, S., Sifakis, J.: Causal semantics for the algebra of connectors. Formal Methods in System Design 36(2), 167–194 (2010)
5. Bliudze, S., Sifakis, J.: Synthesizing glue operators from glue constraints for the construction of component-based systems. In: Software Composition. LNCS, vol. 6708, pp. 51–67. Springer (2011)
6. Bozga, M., Jaber, M., Maris, N., Sifakis, J.: Modeling dynamic architectures using Dy-BIP. In: Software Composition. LNCS, vol. 7306, pp. 1–16. Springer (2012)
7. Bruni, R., Lanese, I., Montanari, U.: A basic algebra of stateless connectors. Theor. Comput. Sci. 366(1-2), 98–120 (2006)
8. Bruni, R., Melgratti, H., Montanari, U.: Connector algebras, Petri nets, and BIP. In: PSI'11. LNCS, vol. 7162, pp. 19–38. Springer (2012)
9. Bruni, R., Melgratti, H., Montanari, U., Sobocinski, P.: Connector algebras for C/E and P/T nets' interactions. Logical Methods in Comp. Sci. (2013), to appear.
10. Bruni, R., Melgratti, H.C., Montanari, U.: A connector algebra for P/T nets interactions. In: CONCUR'11. LNCS, vol. 6901, pp. 312–326. Springer (2011)
11. Esparza, J., Nielsen, M.: Decidability issues for Petri nets. Petri nets newsletter 94, 5–23 (1994)
12. Gadducci, F., Montanari, U.: The tile model. In: Proof, Language, and Interaction. pp. 133–166. The MIT Press (2000)
13. Lapadula, A., Pugliese, R., Tiezzi, F.: A formal account of ws-bpel. In: COORDINATION'08. LNCS, vol. 5052, pp. 199–215. Springer (2008)
14. Milner, R., Parrow, J., Walker, D.: A calculus of mobile processes, I–ii. Inf. Comput. 100(1), 1–77 (1992)
15. Perry, D.E., Wolf, E.L.: Foundations for the study of software architecture. ACM SIGSOFT Software Engineering Notes 17, 40–52 (1992)
16. Sobocinski, P.: Representations of Petri net interactions. In: CONCUR'10. LNCS, vol. 6269, pp. 554–568. Springer (2010)
17. Viroli, M.: A core calculus for correlation in orchestration languages. J. Log. Algebr. Program. 70(1), 74–95 (2007)