Extended Connectors: Structuring Glue Operators in BIP

Eduard Baranov and Simon Bliudze
École Polytechnique Fédérale de Lausanne
Rigorous System Design Laboratory
INJ Building, Station 14, 1015 Lausanne, Switzerland
{firstname.lastname}@epfl.ch

Based on a variation of the BIP operational semantics using the offer predicate introduced in our previous work, we extend the algebras used to model glue operators in BIP to encompass priorities. This extension uses the Algebra of Causal Interaction Trees, $\mathscr{T}(P)$, as a pivot: existing transformations automatically provide the extensions for the Algebra of Connectors. We then extend the axiomatisation of $\mathscr{T}(P)$, since the equivalence induced by the new operational semantics is weaker than that induced by the interaction semantics. This extension leads to canonical normal forms for all structures and to a simplification of the algorithm for the synthesis of connectors from Boolean coordination constraints.

1 Introduction

Component-based design is based on the separation between coordination and computation. Systems are built from units processing sequential code insulated from concurrent execution issues. The isolation of coordination mechanisms allows a global treatment and analysis.

Fundamentally, each component-based design framework consists of a behaviour type $\mathscr{B}$ [3], defining the underlying semantic domain and the key properties such as the relevant equivalence relations, and a set $\mathscr{G}$ of glue operators of the form $f : 2^{\mathscr{B}} \to \mathscr{B}$. As argued in [16], an important property for glue operators is the possibility to be *flattened*: given a behaviour $g(f(B_1,\dots,B_k),B_{k+1},\dots,B_n)$ obtained by hierarchical composition with two glue operators, there must be an equivalent[1] behaviour $h(B_1,\dots,B_n)$ obtained by applying a single glue operator to the *same* atomic components. In other words, $\mathscr{G}$ must be closed under composition. Flattening enables model transformations, e.g. for optimising code generation or component placement on multicore platforms [9, 10].

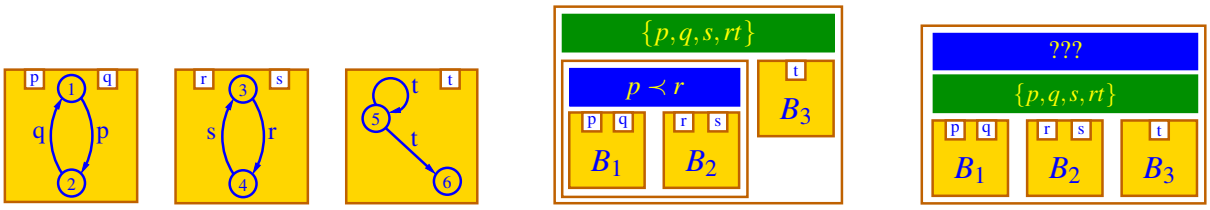
BIP is a component framework for constructing systems by superposing three layers: Behaviour, Interaction and Priorities. In the classical BIP semantics [4], behaviour is modelled by *Labelled Transition Systems* (LTS), i.e. triples $B = (Q,P,\to)$, where Q is a set of *states*, P is a set of *ports*, and $\to \subseteq Q \times 2^P \times Q$ is a set of *transitions*, each labelled by an interaction (a subset of ports). Glue operators are defined using interaction and priority models.

For a set of behaviours $\{B_i = (Q_i,P_i,\to) \,|\, i \in [1,n]\}$, an *interaction model* is a set of *interactions* $\gamma \subseteq 2^P$, where $P = \bigcup_{i=1}^n P_i$ (all P_i are assumed to be pairwise disjoint). The behaviour $\gamma(B_1,\dots,B_n)$ is defined by the behaviour $(Q,P,\to_\gamma)$, with $Q = \prod_{i=1}^n Q_i$ and the minimal transition relation $\to_\gamma$ satisfying the rule (we use set notation to group premises of the same type)

$$\frac{a \in \gamma \quad \left\{ q_i \xrightarrow{a \cap P_i} q_i' \,\middle|\, a \cap P_i \neq \emptyset,\ i \in [1,n] \right\} \quad \left\{ q_i = q_i' \,\middle|\, a \cap P_i = \emptyset,\ i \in [1,n] \right\}}{q_1 \dots q_n \xrightarrow{a}_\gamma q_1' \dots q_n'}. \tag{1}$$

[1] The notion of equivalence, in this context, is given by the behaviour type $\mathscr{B}$ [3].

To appear in EPTCS.



(a) Atomic components B_1, B_2 and B_3 (b) Composed system (c) Flat system

Figure 1: BIP component that cannot be flattened (Example 1.1).

For a behaviour $B = (Q, P, \to)$, a *priority model* is a strict partial order $\prec$ on 2^P. When $a \prec a'$, we say that the interaction a' has *higher priority* than a. We put $B_\prec \triangleq (Q, P, \to_\prec)$, with the minimal transition relation $\to_\prec$ satisfying the rule

$$\frac{q \xrightarrow{a} q' \quad \left\{ q \not\xrightarrow{a'} \;\middle|\; a \prec a' \right\}}{q \xrightarrow{a}_\prec q'}. \tag{2}$$

Each n-ary glue operator in BIP is obtained as the composition of an interaction model (an n-ary operator), composing several behaviours into a single one, and a unary priority model.[2] In general, when combined hierarchically such glue operators cannot be flattened. Indeed, consider the following example.

Example 1.1. Let B_1, B_2 and B_3 be the three atomic behaviours shown in Figure 1a and consider the composed behaviour $g(f(B_1, B_2), B_3)$ (Figure 1b), with the glue operator f defined by the interaction model $\{p, q, r, s\}$ (omitted in Figure 1b) and priority model $\{p \prec r\}$; g defined by the interaction model $\{p, q, s, rt\}$ without any additional priority model. One can prove that it is not possible to represent this behaviour as a flat one (Figure 1c). Indeed, it is not sufficient to replace the priority $p \prec r$ by $p \prec rt$: in the global state $(1, 3, 6)$ of the composed behaviour in Figure 1b, interaction p is inhibited by the priority $p \prec r$; in the same state of the composed behaviour in Figure 1c, p would not be inhibited by $p \prec rt$, since interaction rt is not enabled.

Furthermore, although this goes beyond the scope of this paper, one can prove that there is no flat glue operator h in the classical BIP semantics given by (1) and (2), such that $g(f(B_1, B_2), B_3)$ be equivalent to $h(B_1, B_2, B_3, B_4)$ with any additional helper behaviour B_4.

The impossibility of flattening in the above example is due to the fact that the information used by the priority model refers only to interactions authorised by the underlying interaction model. All the information about interactions enabled in atomic components is lost after the application of f. For instance, one can consider that, in Example 1.1, transitions p and r model respectively taking and liberating a semaphore. Thus p should be disabled whenever r is possible, independently of whether r can actually be taken on not (e.g. when r is blocked waiting for a synchronisation, as in Figure 1b).

In [8], a variation of the BIP operational semantics was introduced. In this variation, a behaviour is defined as an LTS with an additional *offer* predicate, i.e. a quadruple $B = (Q, P, \to, \uparrow)$, such that $\uparrow \subseteq Q \times P$ and, for any $q \in Q$ and $p \in P$, holds the implication $(\exists a \subseteq P : p \in a \wedge q \xrightarrow{a}) \implies q \uparrow p$. The converse implication is not required. In particular, when a transition labelled p in a sub-component of a composed

[2] Notice that both interaction and priority models can be trivial: a trivial interaction model over the set of ports P is the set of *singleton interactions* $\{\{p\} \mid p \in P\}$; a trivial priority model is empty with none of the interactions having higher priority than any other.

behaviour is blocked waiting for a synchronisation, p is still considered as offered.[3] In [8], we have established the equivalence between, on one hand, glue operators defined by sets of SOS rules having positive premises in terms of the transition relations $\rightarrow$ and offering predicates $\uparrow$ and negative premises in terms of the offering predicates only, and, on the other hand, Boolean formula with the so-called *firing* and *activation* variables. We have also studied the expressiveness of such glue operators and compared it with classical BIP.

Using the offer predicate instead of the transition relation in the negative premises of (2), ensures that the resulting set of glue operators is closed under composition.

In this paper, we show how the algebras representing interaction models can be naturally generalised to also define priorities, based on the use of activation and firing variables.

Several algebraic structures are used to define and manipulate interaction models in BIP [4, 7].

The Algebra of Interactions, $\mathscr{AI}(P)$, is isomorphic to 2^{2^P}. It provides a simple algebraic representation of interaction models simplifying the definition of the semantics of other algebras.

The Algebra of Connectors, $\mathscr{AC}(P)$, defines the connectors in the form used in the BIP language, well adapted for graphical representation and for the specification of data transfers.

The Algebra of Causal Interaction Trees, $\mathscr{T}(P)$, defines an alternative semantic domain for connectors with the explicit causality relation between ports. Coherence results for the $\mathscr{AI}(P)$ and $\mathscr{T}(P)$ semantics of connectors have been provided in [7].

Systems of Causal Rules, $\mathscr{CR}(P)$, derived from causal interaction trees define a Boolean representation of connectors, suitable for symbolic manipulation and for specification of state safety properties.

In [7], the four transformations were provided between $\mathscr{AC}(P)$ and $\mathscr{T}(P)$, and between $\mathscr{T}(P)$ and $\mathscr{CR}(P)$. In particular, this allows to synthesise connectors from $\mathbb{B}[P]$ Boolean formulæ.

In this paper, we study the extension of the above algebras to represent both interaction and priority models. Equivalence induced by the new operational semantics is weaker than that induced by the interaction semantics. We extend accordingly the axioms of $\mathscr{T}(\cdot)$ and provide corresponding normal forms for terms of the considered algebras. Finally, we show that, in this context, the connector synthesis algorithm in [7] can be simplified by considering only the causal rules with firing variables in the effect.

The rest of the paper is structured as follows. We start, in Section 2, by a short discussion of some related work. In Section 3, we briefly recall the syntax and semantics of all the considered algebras. Section 4 presents the new semantic model for BIP based on the offer predicate. Main contributions of the paper, namely the extensions of the algebras encompassing the activation and negative ports are presented in Section 5. We illustrate the extended algebras with a connector synthesis example presented in Section 6. Finally, Section 7 concludes the paper.

2 Related work

The results in this paper build on our previous work cited above. However, the following related work should also be mentioned. The approach we use for the Boolean encoding of glue constraints is close to that used for computing flows in Reo connectors in [13], where it is further extended to data flows.

Several methodologies for synthesis of component coordination have been proposed in the literature, e.g. connector synthesis in [1, 2, 14]. Both approaches are very different from ours. In [1], Reo circuits are generated from constraint automata. This approach is limited, in the first place, by the complexity of building the automaton specification of interactions. An attempt to overcome this limitation is made in

[3] As a side effect, the offer predicate can be used to distinguish between atomic and composite behaviours: a behaviour is atomic iff $(\exists a \subseteq P : p \in a \wedge q \xrightarrow{a}) \iff q \uparrow p$ [8].

[2] by generating constraint automata from UML sequence diagrams. In [14], connectors are synthesised in order to ensure deadlock freedom of systems that follow a very specific architectural style imposing both the interconnection topology and communication primitives (notification and request messages).

Recently a comparative study [12] of three connector frameworks—tile model [11], wire calculus [17] and BIP—has been performed. From the operational semantics perspective, this comparison only accounts for operators with positive premises. In particular, priority in BIP is not considered. It would be interesting to see whether using "local" offer predicate instead of "global" priorities of the classical BIP could help generalising this work.

3 Representations of the interaction model

In this section, we briefly recall the syntax and semantics of the algebras used to represent BIP interaction models. The semantics of the Algebra of Interactions is given in terms of sets of interactions by a function $\|\cdot\| : \mathscr{AI}(P) \to 2^{2^P}$. Two terms $x, y \in \mathscr{AI}(P)$ are *equivalent* $x \simeq y$ iff $\|x\| = \|y\|$. For any other algebra, $\mathscr{A}(P)$, among those mentioned in the introduction, we define its semantics by the function $|\cdot| : \mathscr{A}(P) \to \mathscr{AI}(P)$. A function $\|\cdot\| : \mathscr{A}(P) \to 2^{2^P}$ is obtained by composing $|\cdot| : \mathscr{A}(P) \to \mathscr{AI}(P)$ and $\|\cdot\| : \mathscr{AI}(P) \to 2^{2^P}$. The axiomatisation of $\mathscr{AI}(P)$ given in [4] is sound and complete with respect to $\simeq$. Hence, for other algebras, the equivalences induced by $\|\cdot\|$ and $|\cdot|$ coincide.

Below, we assume that a set of ports P is given, such that $0, 1 \notin P$.

3.1 Algebra of Interactions

Syntax. The syntax of the *Algebra of Interactions*, $\mathscr{AI}(P)$, is defined by the following grammar

$$x \quad ::= \quad 0 \mid 1 \mid p \in P \mid x \cdot x \mid x + x \mid (x), \tag{3}$$

where '$+$' and '$\cdot$' are binary operators, respectively called *union* and *synchronisation*. Synchronisation binds stronger than union.

Semantics. The semantics of $\mathscr{AI}(P)$ is given by the function $\|\cdot\| : \mathscr{AI}(P) \to 2^{2^P}$, defined by

$$\begin{array}{lcl} \|0\| = \emptyset, \quad \|1\| = \{\emptyset\}, \quad \|p\| = \Big\{\{p\}\Big\}, \\ \|x_1 + x_2\| & = & \|x_1\| \cup \|x_2\|, \\ \|x_1 \cdot x_2\| & = & \Big\{a_1 \cup a_2 \Big| a_1 \in \|x_1\|, a_2 \in \|x_2\|\Big\}, \\ \|(x)\| & = & \|x\|, \end{array} \tag{4}$$

for $p \in P$, $x, x_1, x_2 \in \mathscr{AI}(P)$. Terms of $\mathscr{AI}(P)$ represent sets of interactions between the ports P.

Sound and complete axiomatisation of $\mathscr{AI}(P)$ with respect to the semantic equivalence is provided in [4]. In a nutshell, $(\mathscr{AI}(P), +, \cdot, 0, 1)$ is a commutative semi-ring idempotent in both $+$ and $\cdot$.

3.2 Algebra of Connectors

Syntax. The syntax of the *Algebra of Connectors*, $\mathscr{AC}(P)$, is defined by the following grammar

$$\begin{array}{rcll} s & ::= & [0] \mid [1] \mid [p] \mid [x] & (\mathit{synchrons}) \\ t & ::= & [0]' \mid [1]' \mid [p]' \mid [x]' & (\mathit{triggers}) \\ x & ::= & s \mid t \mid x \cdot x \mid x + x \mid (x)\,, & \end{array} \tag{5}$$

for $p \in P$, and where '$+$' is binary operator called *union*, '$\cdot$' is a binary operator called *fusion*, and brackets '$[\cdot]$' and '$[\cdot]'$' are unary *typing* operators. Fusion binds stronger than union.

Fusion is a generalisation of the synchronisation in $\mathscr{AI}(P)$. Typing is used to form typed connectors: '$[\cdot]'$' defines *triggers* (can initiate an interaction), and '$[\cdot]$' defines *synchrons* (need synchronisation with other ports in order to interact).

Semantics. The semantics of $\mathscr{AC}(P)$ is given by the function $|\cdot| : \mathscr{AC}(P) \to \mathscr{AI}(P)$:

$$|[p]| = p\,, \qquad |x_1 + x_2| = |x_1| + |x_2|\,, \qquad \left|\prod_{i=1}^{n}[x_i]\right| = \prod_{i=1}^{n}|x_k|\,, \tag{6}$$

$$\left|\prod_{i=1}^{n}[x_i]'\prod_{j=1}^{m}[y_j]\right| = \sum_{i=1}^{n}|x_i|\left(\prod_{k\neq i}\Big(1+|x_k|\Big)\ \prod_{j=1}^{m}\Big(1+|y_j|\Big)\right) \tag{7}$$

for $x, x_1, \ldots, x_n, y_1, \ldots, y_m \in \mathscr{AC}(P)$ and $p \in P \cup \{0,1\}$.

Sound and complete axiomatisation of $\mathscr{AC}(P)$ with respect to the semantic equivalence is provided in [5]. We omit it here, since we will not need it in the rest of this paper.

3.3 Algebra of Causal Interaction Trees

Syntax. The syntax of the *Algebra of Causal Interaction Trees*, $\mathscr{T}(P)$, is given by

$$t ::= a \,|\, a \to t \,|\, t \oplus t\,, \tag{8}$$

where $a \in 2^P \cup \{0,1\}$ is an interaction, and '$\to$' and '$\oplus$' are respectively the *causality* and the *parallel composition* operators. Causality binds stronger than parallel composition. Notice that a causal interaction tree can have several roots.

The causality operator is right- (but not left-) associative, thus for interactions $a_1, \ldots, a_n$, we can abbreviate $a_1 \to (a_2 \to (\cdots \to a_n)\ldots))$ to $a_1 \to a_2 \to \cdots \to a_n$. We call this construction a *causal chain*.

Semantics. The semantics of $\mathscr{T}(P)$ is given by the function $|\cdot| : \mathscr{T}(P) \to \mathscr{AI}(P)$

$$|a| = a\,, \qquad |a \to t| = a\Big(1 + |t|\Big)\,, \qquad |t_1 \oplus t_2| = |t_1| + |t_2| + |t_1|\,|t_2|\,, \tag{9}$$

where a is an interaction and $t, t_1, t_2 \in \mathscr{T}(P)$.

A sound (although not complete) axiomatisation of $\mathscr{T}(P)$ is provided in [7]. Rather than reproduce it here, we indicate the differences after the extension provided in Section 5.1.

3.4 Systems of Causal Rules

Below, for any set X of propositional variables, we denote by $\mathbb{B}[X]$ the corresponding Boolean algebra generated by X. For presentation clarity, we will often omit the conjunction operator and write $a \vee bc$ instead of $a \vee (b \wedge c)$.

Definition 3.1. A *causal rule* is a $\mathbb{B}[P]$ formula $E \Rightarrow C$, where E (the *effect*) is either a constant, $\mathtt{tt}$, or a port variable $p \in P$, and C (the *cause*) is either a constant, $\mathtt{tt}$ or $\mathtt{ff}$, or a positive $\mathbb{B}[P \setminus \{p\}]$ formula in disjunctive normal form.

Remark 3.2. Notice that $a_1 \vee a_1 a_2 = a_1$, and therefore causal rules can be simplified by replacing $p \Rightarrow a_1 \vee a_1 a_2$ with $p \Rightarrow a_1$). We assume that all the causal rules are simplified by this absorption rule.

Definition 3.3. A *system of causal rules* is a set $R = \{p \Rightarrow x_p\}_{p \in P^t}$, where $P^t \stackrel{\Delta}{=} P \cup \{\mathtt{tt}\}$, having precisely one causal rule for each port variable $p \in P^t$. An interaction $a \in 2^P$ satisfies the system R (denoted $a \models R$), iff the characteristic valuation of a on P satisfies the formula $\bigwedge_{p \in P^t}(p \Rightarrow x_p)$. We denote by $|R| \stackrel{\Delta}{=} \sum_{a \models R} a$ the union (in terms of the Algebra of Interactions) of the interactions satisfying R. Thus we have $|\cdot| : \mathscr{CR}(P) \to \mathscr{AI}(P)$, where $\mathscr{CR}(P)$ is the set of all systems of causal rules over the set of port variables P.

3.5 Transformations between different representations

Transformations $\mathscr{AC}(P) \underset{\sigma}{\overset{\tau}{\rightleftarrows}} \mathscr{T}(P)$ and $\mathscr{T}(P) \overset{R}{\rightleftarrows} \mathscr{CR}(P)$ were defined in [7] and shown to respect $\simeq$. Below, we will only need the transformations $\sigma : \mathscr{T}(P) \to \mathscr{AC}(P)$ and $R : \mathscr{T}(P) \to \mathscr{CR}(P)$. The former is defined recursively by putting

$$\sigma(a) = [a], \qquad \sigma(a \to t) = [a]'[\sigma(t)], \qquad \sigma(t_1 \oplus t_2) = [\sigma(t_1)]'[\sigma(t_2)]'. \tag{10}$$

We define $R : \mathscr{T}(P) \to \mathscr{CR}(P)$ by putting

$$R(t) \;=\; \{p \Rightarrow c_p(t)\}_{p \in P \cup \{\mathtt{tt}\}}, \tag{11}$$

where the function $c_p : \mathscr{T}(P) \to \mathbb{B}[P]$ is defined recursively as follows. For $a \in 2^P$ (with $p \notin a$) and $t, t_1, t_2 \in \mathscr{T}(P)$, we put

$$\begin{aligned}
c_p(0) &= \mathtt{ff}, & c_{\mathtt{tt}}(0) &= \mathtt{ff},\\
c_p(p \to t) &= \mathtt{tt}, & c_{\mathtt{tt}}(1 \to t) &= \mathtt{tt},\\
c_p(pa \to t) &= a, & c_{\mathtt{tt}}(a \to t) &= a,\\
c_p(a \to t) &= a \wedge c_p(t), & &\\
c_p(t_1 \oplus t_2) &= c_p(t_1) \vee c_p(t_2), & c_{\mathtt{tt}}(t_1 \oplus t_2) &= c_{\mathtt{tt}}(t_1) \vee c_{\mathtt{tt}}(t_2).
\end{aligned}$$

Observe that this transformation associates to each port $p \in P$ a causal rule $p \Rightarrow C$, where C is the disjunction of all prefixes leading from roots of t to some node containing p, including the ports of this node other than p.

4 Modification of the semantic model

We now present the variation of the BIP operational semantics based on the offer predicate [8].

Definition 4.1. A *labelled transition system* (LTS) is a triple $(Q,P,\rightarrow)$, where Q is a set of *states*, P is a set of *ports*, and $\rightarrow \subseteq Q \times 2^P \times Q$ is a set of *transitions*, each labelled by a non-empty set of ports. For $q,q' \in Q$ and $a \in 2^P$, we write $q \xrightarrow{a} q'$ iff $(q,a,q') \in \rightarrow$. A label $a \in 2^P$ is *active* in a state $q \in Q$ (denoted $q \xrightarrow{a}$), iff there exists $q' \in Q$ such that $q \xrightarrow{a} q'$. We abbreviate $q \not\xrightarrow{a} \triangleq \neg(q \xrightarrow{a})$.

Below, it is assumed that, for all $q \in Q$, $q \xrightarrow{\emptyset} q$. All results of the paper can be reformulated without this assumption, but making it simplifies the presentation. We write pq for the set of ports $\{p,q\}$.

Definition 4.2. A *behaviour* is a pair $B = (S,\uparrow)$ consisting of an LTS $S = (Q,P,\rightarrow)$ and an *offer* predicate $\uparrow$ on $Q \times P$ such that $q\uparrow p$ holds (a port $p \in P$ is *offered* in a state $q \in Q$) whenever there is a transition from q containing p, that is $(\exists a \in 2^P : p \in a \wedge q \xrightarrow{a}) \Rightarrow q\uparrow p$. We write $B = (Q,P,\rightarrow,\uparrow)$ for $B = ((Q,P,\rightarrow),\uparrow)$.

The offer predicate extends to sets of ports: for $a \in 2^P$, $q\uparrow a \triangleq \bigwedge_{p\in a} q\uparrow p$. Notice that $q\uparrow\emptyset \equiv \mathtt{tt}$.

Remark 4.3. In the following, we assume, for any $B_i = (Q_i,P_i,\rightarrow,\uparrow)$ with $i \in [1,n]$, that $\{P_i\}_{i=1}^n$ are pairwise disjoint (i.e. $i \neq j$ implies $P_i \cap P_j = \emptyset$) and $P \triangleq \bigcup_{i=1}^n P_i$.

To avoid excessive notation, here and in the rest of the paper, we drop the indices on $\rightarrow$ and $\uparrow$, as they can always be unambiguously deduced from the corresponding state variables.

Let P be a set of ports. We denote $\dot{P} \triangleq \{\dot{p} \,|\, p \in P\}$ and $\overline{P} \triangleq \{\overline{p} \,|\, p \in P\}$. We call the elements of P, $\dot{P}$ and $\overline{P}$ respectively *activation*, *firing* and *negative port typings*.

Definition 4.4. An *interaction* is a subset $a \subseteq P \cup \dot{P} \cup \overline{P}$.

For a given interaction a, we define the following sets of ports:

- $\mathbf{act}(a) \triangleq a \cap P$, the *activation support* of a,
- $\mathbf{fire}(a) \triangleq \{p \in P \,|\, \dot{p} \in a\}$, the *firing support* of a,
- $\mathbf{neg}(a) \triangleq \{p \in P \,|\, \overline{p} \in a\}$, the *negative support* of a.

Definition 4.5. Let $B_i = (Q_i,P_i,\rightarrow,\uparrow)$, with $i \in [1,n]$ and $P = \bigcup_{i=1}^n P_i$, be a set of component behaviours. Let $\gamma \subseteq 2^{P\cup\dot{P}\cup\overline{P}}$ be a set of interactions. The composition of $\{B_i\}_{i=1}^n$ with γ is a behaviour $\gamma(B_1,\ldots,B_n) \triangleq (Q,P,\rightarrow,\uparrow)$ with

- the set of states $Q = \prod_{i=1}^n Q_i$—the cartesian product of the sets of states Q_i,
- the strongest (i.e. inductively defined) offer predicate $\uparrow$ satisfying the rules, for each $i \in [1,n]$,

$$\frac{q_i\uparrow p}{q_1\ldots q_n\uparrow p} \tag{12}$$

  (recall that the sets of ports P_i are pairwise disjoint),

- the minimal transition relation $\rightarrow$ satisfying the rule

$$\frac{a \in \gamma \quad \left\{q_i \xrightarrow{\mathbf{fire}(a)\cap P_i} q_i'\right\}_{i=1}^n \quad \left\{q_i\uparrow(\mathbf{act}(a)\cap P_i)\right\}_{i=1}^n \quad \left\{q_i \not\uparrow p \;\middle|\; p \in \mathbf{neg}(a)\cap P_i\right\}_{i=1}^n}{q_1\ldots q_n \xrightarrow{\mathbf{fire}(a)} q_1'\ldots q_n'}. \tag{13}$$

It is important to observe that, as stated by Lemma 4.6 below, the rule (13) in Definition 4.5 implies that any interaction $a \in \gamma$ such that $\mathbf{fire}(a) = \emptyset$ does not have any impact on the composed system.

Lemma 4.6. *Let $\gamma_1, \gamma_2 \subseteq 2^{P \cup \dot{P} \cup \overline{P}}$ be two sets of interactions and denote $\gamma_1 \triangle \gamma_2 \stackrel{\Delta}{=} (\gamma_1 \setminus \gamma_2) \cup (\gamma_2 \setminus \gamma_1)$ their symmetric difference. If $\mathbf{fire}(a) = \emptyset$, for all $a \in \gamma_1 \triangle \gamma_2$, then $\gamma_1(B_1, \ldots, B_n) = \gamma_2(B_1, \ldots, B_n)$.*

Proof. It's easy to see that $\gamma_1(B_1, \ldots, B_n)$ and $\gamma_2(B_1, \ldots, B_n)$ behaviours can differ only in their respective transition relations $\rightarrow$.

Application of the rule (13) in Definition 4.5 to an interaction with empty firing support generates a transition $q_1 \ldots q_n \stackrel{\mathbf{fire}(a)=\emptyset}{\longrightarrow} q_1 \ldots q_n$. As mentioned in the opening of this section, we assume that the self-loop transition labelled by an empty set is enabled in all states. Therefore, the above transition is present in both $\gamma_1(B_1, \ldots, B_n)$ and $\gamma_2(B_1, \ldots, B_n)$. Rule (13) is applied independently to all interactions, so this interaction cannot have any impact on other transitions of the composed system. Thus, as all interactions from $\gamma_1 \triangle \gamma_2$ have empty firing support, $\gamma_1(B_1, \ldots, B_n) = \gamma_1 \cap \gamma_2(B_1, \ldots, B_n) = \gamma_2(B_1, \ldots, B_n)$. □

Lemma 4.7. *Let $\gamma_1 \subseteq 2^{P \cup \dot{P} \cup \overline{P}}$ be a set of interactions, $\gamma_2 = \gamma_1 \cup \{a\}$, with $a \subseteq P \cup \dot{P} \cup \overline{P}$, such that there is an interaction $b \in \gamma_1$, $b \subseteq a$ and $\mathbf{fire}(b) = \mathbf{fire}(a)$. Then $\gamma_1(B_1, \ldots, B_n) = \gamma_2(B_1, \ldots, B_n)$.*

Proof. According to rule (13) any transition generated by the interaction a can also be generated by the interaction b. Thus, interaction a does not impact the behaviour of the composed system, and $\gamma_1(B_1, \ldots, B_n) = \gamma_2(B_1, \ldots, B_n)$. □

Taking on the Example 1.1 from the introduction, a flat composition of B_1, B_2 and B_3 equivalent to that of Figure 1b in the semantics of Definition 4.5 is shown in Figure 2 on the right. This representation follows the classical BIP approach with the exception of the priority, whereof the semantics is defined in terms of the offer predicate. In terms of Definition 4.5, this is translated by taking $\gamma = \{\dot{p}\overline{r}, \dot{q}, \dot{s}, \dot{r}\dot{t}\} \subseteq 2^{P \cup \dot{P} \cup \overline{P}}$.

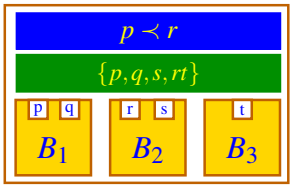


Figure 2: Flat composed system equivalent to the one shown in Figure 1b.

BIP composition operators, consisting of an interaction and a priority model, can be given new operational semantics in terms of the offer predicate: the semantics of the interaction model composition remains the same (1), whereas the rule for priority becomes

$$\frac{q \stackrel{a}{\longrightarrow} q' \quad \left\{ q \not\stackrel{a'}{\longrightarrow} \;\middle|\; a \prec a' \right\}}{q \stackrel{a}{\longrightarrow}_{\prec} q'}. \tag{14}$$

Clearly, any combination of BIP interaction and priority models can be represented by an extended interaction model $\gamma \subseteq 2^{P \cup \dot{P} \cup \overline{P}}$. A priority $a \prec p_1 \ldots p_n$ is translated into $\{\dot{a}\,\overline{p_1}, \ldots, \dot{a}\,\overline{p_n}\}$ (here $\dot{a}$ is a shorthand for the set of firing ports corresponding to ports in a). In general, when several inhibitors are defined for the same interaction, that is $a \prec p_1^i \ldots p_{n_i}^i$, for $i \in [1, m]$, this translates into $\{\dot{a}\,\overline{p_{k_1}^1} \ldots \overline{p_{k_m}^m} \,|\, k_i \in [1, n_i]\}$.

5 Algebra extensions

In Section 4, we have replaced the classical BIP combination of interaction and priority models with an extended interaction model with ports of three types: firing, activation and negative.[4] We can now extend other algebras used for the glue representation.

[4] Only firing and negative ports are necessary to define classical BIP composition operators. Activation ports allow for a full correspondence with $\mathbb{B}[P, \dot{P}]$ Boolean constraints. This correspondence and an expressivity study are given in [8].

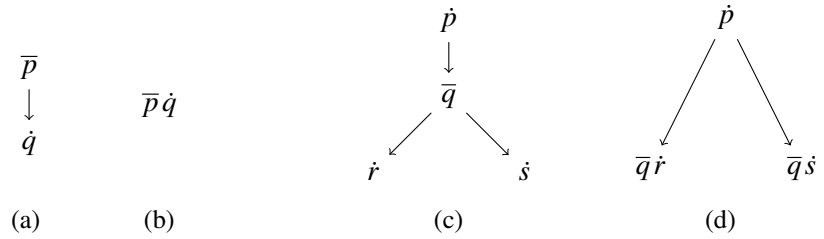


Figure 3: Two pairs of equivalent trees: (a), (b) and (c), (d).

We start by considering the extension of the Algebra of Interactions, $\mathscr{AI}(P)$. Recall that $x \simeq y$ iff $\|x\| = \|y\|$. As a simple corollary of the results in [6], $\|x\| = \|y\|$ is equivalent to $\|x\|(\mathbf{B}) = \|y\|(\mathbf{B})$, for any finite family $\mathbf{B}$ of behaviours.

Below we will consider $\mathscr{AI}(P \cup \dot{P} \cup \overline{P})$ with the latter definition of term equivalence: two terms $x, y \in \mathscr{AI}(P \cup \dot{P} \cup \overline{P})$ are equivalent iff $\|x\|(\mathbf{B}) = \|y\|(\mathbf{B})$ (in terms of Definition 4.5), for any finite family $\mathbf{B}$ of behaviours. In general, we define equivalence as follows.

Definition 5.1. Let $\mathscr{A}(P)$ be an algebra, $\|\cdot\| : \mathscr{A}(P) \to 2^{2^P}$. Two terms $x, y \in \mathscr{A}(P)$ are *equivalent* $x \sim y$ iff, for any finite family $\mathbf{B}$ of behaviours, $\|x\|(\mathbf{B}) = \|y\|(\mathbf{B})$ (in terms of Definition 4.5).

Remark 5.2. Clearly $\sim$ is weaker than $\simeq$.

We are now in position to similarly extend the other algebras. The interaction semantics of the causal interaction trees $|\cdot| : \mathscr{T}(P) \to \mathscr{AI}(P)$ is transposed without any change to $|\cdot| : \mathscr{T}(P \cup \dot{P} \cup \overline{P}) \to \mathscr{AI}(P \cup \dot{P} \cup \overline{P})$. Similarly, the functions $\tau : \mathscr{AC}(P) \to \mathscr{T}(P)$ and $\sigma : \mathscr{T}(P) \to \mathscr{AC}(P)$ are transposed identically to $\mathscr{AC}(P \cup \dot{P} \cup \overline{P})$ and $\mathscr{T}(P \cup \dot{P} \cup \overline{P})$. The same goes for the mapping $R(t)$ associating to a causal interaction tree $t \in \mathscr{T}(P)$ the corresponding system of causal rules [7]. The only difference is that, in $\mathscr{CR}(P \cup \dot{P} \cup \overline{P})$ we introduce the following additional axiom: $\dot{p} \Rightarrow p$, for all $p \in P$.

Proposition 5.3. *The equivalence relation $\sim$ on $\mathscr{T}(P \cup \dot{P} \cup \overline{P})$ is a congruence.*

Sketch of the proof. The proof is the same as for $\mathscr{T}(P)$ [7]. For any two trees $t_1, t_2 \in \mathscr{T}(P \cup \dot{P} \cup \overline{P})$ and for any context $C(z) \in \mathscr{T}(P \cup \dot{P} \cup \overline{P} \cup \{z\})$, we have to show that the equivalence $t_1 \sim t_2$ implies $C(t_1/z) \sim C(t_2/z)$, where $C(t_i/z)$ is the tree obtained, by replacing in $C(z)$ all occurrences of z by t_i. Since the semantics $\mathscr{T}$ is compositional, structural induction on the context $C(z)$ proves the proposition. □

The first consequence of the above extension is that, rather than extending the existing graphical representation of connectors, it can be directly used in its present form to express priorities and activation conditions (the use of the offer predicate in the positive premises of the rule (13)) by adding a trivalued attribute to ports: *firing*, *activation* and *negative*. It is important to observe the difference between, on one hand, adding an attribute to ports and, on the other hand, modifying the typing operator (*synchron* vs. *trigger* typing), since the latter is applied at each level of the connector hierarchy, whereas the former is applied to ports, that is only at the leaves of the connector.

5.1 Refinement of the extension

When we apply $x, y \in \mathscr{AI}(P \cup \dot{P} \cup \overline{P})$ to compose behaviour with operational semantics of Definition 4.5, $\|x\|(\mathbf{B}) = \|y\|(\mathbf{B})$ does not imply $x = y$. $\mathscr{AI}$ axioms are not complete (although still sound) with respect to $\sim$, since this equivalence is weaker than $\simeq$. Consequently, on $\mathscr{T}(P \cup \dot{P} \cup \overline{P})$, $\sim$ is also weaker than $\simeq$.

Example 5.4. Let $P = \{p,q,r,s\}$ and consider the $\mathscr{T}(P \cup \dot{P} \cup \overline{P})$ trees shown in Figure 3. The interaction semantics of the tree in Figure 3a is $\|\overline{p} \to \dot{q}\| = \{\overline{p}, \overline{p}\,\dot{q}\}$. However, the interaction $\overline{p}$ does not contain any firing ports. Therefore, as mentioned above (Lemma 4.6), it does not influence component synchronisation and we have $\overline{p} \to \dot{q} \;\sim\; \overline{p}\,\dot{q}$ (cf. Figure 3b).

The causal interaction tree in Figure 3c also defines a redundant interaction. Indeed,

$$\|\dot{p} \to \overline{q} \to (\dot{r} \oplus \dot{s})\| \;=\; \{\dot{p},\, \dot{p}\,\overline{q},\, \dot{p}\,\overline{q}\,\dot{r},\, \dot{p}\,\overline{q}\,\dot{s},\, \dot{p}\,\overline{q}\,\dot{r}\,\dot{s}\}\,.$$

Although the interaction $\dot{p}\,\overline{q}$ does contain a firing port $\dot{p}$, it is redundant (Lemma 4.7). We conclude, therefore, that the causal interaction trees in Figure 3c and Figure 3d are equivalent, since

$$\|\dot{p} \to (\overline{q}\,\dot{r} \oplus \overline{q}\,\dot{s})\| \;=\; \{\dot{p},\, \dot{p}\,\overline{q}\,\dot{r},\, \dot{p}\,\overline{q}\,\dot{s},\, \dot{p}\,\overline{q}\,\dot{r}\,\dot{s}\}\,.$$

The above example illustrates the idea that the nodes of causal interaction trees, which do not contain firing ports, can be "pushed" down the tree.

Another notable case leading to redundant interactions corresponds to trees containing *contradictory port typings*. For example, either of the two equivalent trees $\overline{p} \to \dot{p}$ and $\overline{p}\,\dot{p}$ authorises the interaction $\overline{p}\,\dot{p}$. However, when considered in the context of the rule (13), this interaction generates two conflicting premises $q_i \xrightarrow{p} q_i'$ and $q_i \not\uparrow p$. Thus, this instance of the rule (13) does not authorise any transitions and the interaction $\overline{p}\,\dot{p}$ can be safely discarded. This example corresponds to the additional axiom $\dot{p} \Rightarrow p$ imposed in [8] on the Boolean formulæ in $\mathbb{B}[P,\dot{P}]$. Similarly, redundant interactions appear when a tree contains other distinct port typings of the same port: p and $\overline{p}$, generating conflicting premises $q_i \uparrow p$ and $q_i \not\uparrow p$; p and $\dot{p}$, whereof the former generates the premise $q_i \uparrow p$ redundant alongside the premise $q_i \xrightarrow{p} q_i'$ generated by the latter.

Below, we provide a set of axioms reducing interaction redundancy. We enrich axioms for $\mathscr{T}(P \cup \dot{P} \cup \overline{P})$ from [7] by adding some new ones, specific for the trivalued port attribute.

Axioms. 1. For all $p \in P$ and $a \subseteq P \cup \dot{P} \cup \overline{P}$ such that $a \neq \emptyset$,

   (a) $a \cdot 0 = 0$,

   (b) $a \cdot 1 = a$, for $a \neq 0$,

   (c) $\dot{p} \cdot p = \dot{p}$ (cf. the additional axiom $\dot{p} \Rightarrow p$ in $\mathscr{CR}(P \cup \dot{P} \cup \overline{P})$),

   (d) $\dot{p} \cdot \overline{p} = p \cdot \overline{p} = 0$.

2. Parallel composition, '$\oplus$', is associative, commutative, idempotent, and its identity element is 0.

3. $a \to 0 = a$, for all $a \subseteq P \cup \dot{P} \cup \overline{P}$.

4. $0 \to t = 0$, for all $t \in \mathscr{T}(P \cup \dot{P} \cup \overline{P})$.

5. $c \to a \to b \to t = c \to ab \to t$ for all $a,b,c \subseteq P \cup \dot{P} \cup \overline{P}$, such that $\mathbf{fire}(a) = \emptyset$, and $t \in \mathscr{T}(P \cup \dot{P} \cup \overline{P})$.

6. $ap \to b = ap \to bp$ for all $a,b \subseteq P \cup \dot{P} \cup \overline{P}$, $p \in P \cup \dot{P} \cup \overline{P}$.

7. $a \to (t_1 \oplus t_2) = a \to t_1 \;\oplus\; a \to t_2$, for all $a \subseteq P \cup \dot{P} \cup \overline{P}$, $t_1, t_2 \in \mathscr{T}(P \cup \dot{P} \cup \overline{P})$.

Axioms 1 equalise redundant interactions due to contradictory port typings, whereas Axiom 5 eliminates the nodes with empty firing support. Axioms 2, 3, 4 and 7 are the same as in [7].[5]

Proposition 5.5. *The above axiomatisation is sound with respect to* $\sim$.

[5] The two remaining axioms from [7] are replaced by Lemmas 5.6 and 5.7 in this paper.

Proof. Since, by Proposition 5.3, the equivalence relation $\sim$ is a congruence, it is sufficient to show that all the axioms respect $\sim$. This is proved by verifying that the semantics for left and right sides coincide.

Axioms 2, 3, 4 and 7 are the same as in [7]. Hence, their respective left- and right-hand sides are related by $\simeq$, which is stronger than $\sim$. Axiom 1(a) and Axiom 1(b) are trivial. Axiom 1(c) is a consequence of Lemma 4.7. In the Axiom 1(d), both pairs p and $\overline{p}$, and $\dot{p}$ and $\overline{p}$ produce conflicting premises in the rule (13) and, therefore, do not generate any transitions. For the Axiom 5, we have

$$\|c \to a \to b \to t\| = \{c, ac, abc\} \cup \{abca_2 \,|\, a_2 \in \|t\|\} \tag{15}$$

$$\|c \to ab \to t\| = \{c, abc\} \cup \{abca_2 \,|\, a_2 \in \|t\|\} \tag{16}$$

The only difference between the interaction semantics of the two trees is the interaction ac. However, any transition authorised by the rule (13) with this interaction is also authorised with interaction c, since $\mathbf{fire}(a) = \emptyset$ (Lemma 4.7). Hence, the composed systems coincide.

For the Axiom 6, we have $\|ap \to b\| = \{ap, abp\} = \|ap \to bp\|$. Thus $ap \to b \simeq ap \to bp$, which implies $ap \to b \sim ap \to bp$. □

Notice that our axiomatisation is not complete. For instance, the equivalence $p \to q \oplus q \to p \sim p \oplus q$ cannot be derived from the axioms.

Lemma 5.6. *For all $a, b \subseteq P \cup \dot{P} \cup \overline{P}$, such that $\mathbf{fire}(b) = \emptyset$, holds the equality $a \to b = a$.*

Proof. $a \to b = a \to b \to 0 \to 0 = a \to b \cdot 0 \to 0 = a \to 0 \to 0 = a$ (Axioms 3, 5) □

Lemma 5.7. *For all $a \subseteq P \cup \dot{P} \cup \overline{P}$ and $t \in \mathscr{T}(P \cup \dot{P} \cup \overline{P})$, holds the equality $a \to 1 \to t = a \to t$.*

Proof. If $t = 0$ the statement of this lemma is a special case of Lemma 5.6 with $b = 1$. If $t \neq 0$ it can be represented as a parallel composition of non-zero trees $t = \bigoplus_{i=1}^{n} r_i \to t_i$, with $r_i \subseteq P \cup \dot{P} \cup \overline{P}$. By Axioms 5 and 7, we have

$$a \to 1 \to t \;=\; \bigoplus_{i=1}^{n} (a \to 1 \to r_i \to t_i) \;=\; \bigoplus_{i=1}^{n} (a \to r_i \to t_i) \;=\; a \to \bigoplus_{i=1}^{n} (r_i \to t_i) \;=\; a \to t\,.$$

□

Lemma 5.8. *For all $a, b_i, c \subseteq P \cup \dot{P} \cup \overline{P}$, such that $\mathbf{fire}(a) = \emptyset$ and $t_i \in \mathscr{T}(P \cup \dot{P} \cup \overline{P})$, holds the equality*

$$c \to a \to \bigoplus_{i=1}^{n} (b_i \to t_i) = c \to \bigoplus_{i=1}^{n} (ab_i \to t_i)\,.$$

Proof. As above, applying Axioms 5 and 7, we have

$$c \to a \to \bigoplus_{i=1}^{n} (b_i \to t_i) \;=\; \bigoplus_{i=1}^{n} (c \to a \to b_i \to t_i) \;=\; \bigoplus_{i=1}^{n} (c \to ab_i \to t_i) \;=\; c \to \bigoplus_{i=1}^{n} (ab_i \to t_i)\,.$$

□

Definition 5.9. A causal interaction tree $t \in \mathscr{T}(P \cup \dot{P} \cup \overline{P})$ is in *normal form* if it satisfies the following properties:

1. All nodes of t except roots have non-empty firing support.

2. There are no causal dependencies between the same typing of the same port in t, that is for any causal chain $a \to \cdots \to b$ within t, we have $a \cap b = \emptyset$.
3. There are no causal dependencies between different port typings of the same port in t, other than dependencies of the form $ap \to \cdots \to b\dot{p}$, where $a, b \subseteq P \cup \dot{P} \cup \overline{P}$, $p \in P$.

Proposition 5.10 (Normal form for causal interaction trees)**.** *Every causal interaction tree $t \in \mathscr{T}(P \cup \dot{P} \cup \overline{P})$ has a normal form $t = \tilde{t} \in \mathscr{T}(P \cup \dot{P} \cup \overline{P})$.*

Proof. Consider $t \in \mathscr{T}(P \cup \dot{P} \cup \overline{P})$. We start by computing $t_1 = t$ with all nodes, except potentially the roots, having non-empty firing support.

Let a be a non-root node of t with $\mathbf{fire}(a) = \emptyset$, such that the tree s rooted in a does not have any nodes with empty firing support. If s is empty, that is a is a leaf then remove a from the tree (Lemma 5.6). Otherwise, let c be the parent of a, which exists since a is not a root and move the parallel composition operator up using Axiom 7:

$$c \to \left((a \to s) \oplus \bigoplus_{i=1}^{n} t_i \right) = (c \to a \to s) \oplus \left(\bigoplus_{i=1}^{n} c \to t_i \right). \tag{17}$$

The sub-tree s can be further decomposed as $s = \bigoplus_{i=1}^{n} (b_i \to s_i)$, so, by Lemma 5.8, we have

$$c \to a \to s \;=\; c \to a \to \bigoplus_{i=1}^{n} (b_i \to s_i) \;=\; c \to \bigoplus_{i=1}^{n} (ab_i \to s_i). \tag{18}$$

Each of nodes ab_i has non-empty firing support, since $\mathbf{fire}(b_i) = \emptyset$ by the choice of a. Substituting (18) into (17) and applying Axiom 7, we obtain

$$\left(c \to \bigoplus_{i=1}^{n} (ab_i \to s_i) \right) \oplus \left(\bigoplus_{i=1}^{n} c \to t_i \right) = c \to \left(\left(\bigoplus_{i=1}^{n} ab_i \to s_i \right) \oplus \bigoplus_{i=1}^{n} t_i \right).$$

In the resulting tree, there is one node with empty firing support less than in t. Hence, repeating this procedure as long as there are such nodes, we will compute a tree t_1, where all nodes except roots have non-empty firing support. This computation is confluent, since the order is irrelevant among causally independent nodes, whereas among causally dependent ones it is fixed by the algorithm.

Consider a causal chain $a\tilde{p} \to \cdots \to b\hat{p}$ within t_1, with $\tilde{p}$ and $\hat{p}$ being two typings of the same port. If $\tilde{p} = p$ and $\hat{p} = \dot{p}$, there is nothing to do, since such dependencies are allowed by Definition 5.9. Otherwise, we propagate $\tilde{p}$ down by applying Axiom 6:

$$a\tilde{p} \to c_1 \to \cdots \to c_k \to b\hat{p} \;=\; a\tilde{p} \to c_1\tilde{p} \to \cdots \to c_k \to b\hat{p} = \;\ldots\; = a\tilde{p} \to c_1\tilde{p} \to \cdots \to c_k\tilde{p} \to b\hat{p}\tilde{p}.$$

Case 1: $\tilde{p} = \hat{p}$ or $\tilde{p} = \dot{p}$ and $\hat{p} = p$. We apply Axioms 1(c) and 6:

$$a\tilde{p} \to c_1\tilde{p} \to \cdots \to c_k\tilde{p} \to b\hat{p}\tilde{p} \;=\; a\tilde{p} \to c_1\tilde{p} \to \cdots \to c_k\tilde{p} \to b\tilde{p} \;=\; a\tilde{p} \to c_1 \to \cdots \to c_k \to b.$$

Case 2: $\tilde{p} \neq \hat{p}$ and either $\tilde{p} = \overline{p}$ or $\hat{p} = \overline{p}$. We apply Axioms 1(d), 3 and 6:

$$\begin{aligned} a\tilde{p} \to c_1\tilde{p} \to \cdots \to c_k\tilde{p} \to b\hat{p}\tilde{p} \;&=\; a\tilde{p} \to c_1\tilde{p} \to \cdots \to c_k\tilde{p} \to 0 \;= \\ &= \; a\tilde{p} \to c_1 \to \cdots \to c_k \to 0 \;=\; a\tilde{p} \to c_1 \to \cdots \to c_k. \end{aligned}$$

To compute $\tilde{t}$, we apply this transformation to all relevant causal chains within t. □

Definition 5.11. An $\mathscr{AC}(P \cup \dot{P} \cup \overline{P})$ connector is in *normal form* if the following conditions hold.

1. Nodes at every hierarchical level of the connector, except the bottom one, have at least one trigger.
2. Each node at the bottom hierarchical level, is a strong synchronisation of pairwise distinct ports.
3. Every node at the bottom hierarchical level, without firing ports, has only triggers as ancestors.

Corollary 5.12 (Normal form for connectors)**.** *Every connector $x \in \mathscr{AC}(P \cup \dot{P} \cup \overline{P})$ has an equivalent normal form $x \sim \tilde{x} \in \mathscr{AC}(P \cup \dot{P} \cup \overline{P})$.*

Sketch of the proof. Given a connector x, let $t = \tau(x)$ be the equivalent causal interaction tree and $\tilde{t} = t$ its normal form. Put $\tilde{x} = \sigma(\tilde{t})$. Since both σ and τ preserve $\sim$, we have $\tilde{x} \sim x$. Normality of $\tilde{x}$ is a direct consequence of that of $\tilde{t}$ and the definition (10) of σ. □

Proposition 5.13. *Any causal interaction tree $t \in \mathscr{T}(P \cup \dot{P} \cup \overline{P})$ can be represented by a system of causal rules with only firing ports as effects, i.e. having only rules of the form $\dot{p} \Rightarrow C$, where C is a DNF Boolean formula on $\dot{P} \cup P$ without negative firing variables.*

Proof. Applying the transformation $R : \mathscr{T}(P) \to \mathscr{CR}(P)$ defined in Section 3.5 to a tree $t \in \mathscr{T}(P)$, gives a system of causal rules of the form $p \Rightarrow C$, where C is a DNF Boolean formula and each monomial is a conjunction of the nodes on the way from a root of t to p (some prefix in t leading to p, excluding p).

We define the transformation $\tilde{R} : \mathscr{T}(P \cup \dot{P} \cup \overline{P}) \to \mathscr{CR}(P \cup \dot{P} \cup \overline{P})$, by putting

$$\tilde{R}(t) \stackrel{\Delta}{=} \{p \Rightarrow c_p(t)\}_{p \in \dot{P} \cup \{\mathtt{tt}\}}, \tag{19}$$

that is we omit causal rules for port variables in $P \cup \overline{P}$ (in (19), the set of rules is indexed by $p \in \dot{P} \cup \{\mathtt{tt}\}$ as opposed to $p \in P \cup \{\mathtt{tt}\}$ in (11)). To prove the equivalence $t \sim \tilde{R}(t)$ it is sufficient to show $\tilde{R}(t) \sim R(t)$.

$\tilde{R}(t)$ has less constraints than $R(t)$. Hence, it allows more interactions. Let $a \in \|\tilde{R}(t)\| \setminus \|R(t)\|$, i.e. there exists $p \in P \cup \overline{P}$, such that $p \in a$ and the rule $p \Rightarrow C_1$ is violated by a. Let $\tilde{a} = a \setminus p$.

Assume $\tilde{a} \notin \|\tilde{R}(t)\|$, i.e. there exists $\dot{q} \in \dot{P}$ and a rule $(\dot{q} \Rightarrow C_2) \in \tilde{R}(t)$, such that $\dot{q} \in \tilde{a}$ and the rule $\dot{q} \Rightarrow C_2$ is violated by $\tilde{a}$. This rule is not violated by a. Hence $C_2 = pC_2'$ and, consequently, p lies on all prefixes in t, leading to $\dot{q}$. $a \in \|\tilde{R}(t)\|$, $\dot{q} \in \tilde{a} \subseteq a$, thus there is at least one prefix in t, leading to $\dot{q}$ and contained in a. As p lies on this prefix, the rule $(p \Rightarrow C_1)$ is satisfied by a, contradicting the conclusion above. Therefore our assumption is wrong and $\tilde{a} \in \|\tilde{R}(t)\|$.

Since $\tilde{a} \in \|\tilde{R}(t)\|$ and $\mathbf{fire}(\tilde{a}) = \mathbf{fire}(a)$, we have, by Lemma 4.7, $\|\tilde{R}(t)\|(\mathbf{B}) = (\|\tilde{R}(t)\| \setminus \{a\})(\mathbf{B})$ for any family of behaviours $\mathbf{B}$. Thus, for all $a \in \|\tilde{R}(t)\| \setminus \|R(t)\|$, there exists $\tilde{a} \subsetneq a$, such that $\tilde{a} \in \|\tilde{R}(t)\|$ and $\mathbf{fire}(\tilde{a}) = \mathbf{fire}(a)$. By Lemma 4.7, we have $\|\tilde{R}(t)\|(\mathbf{B}) = \|R(t)\|(\mathbf{B})$ for any family $\mathbf{B}$, i.e. $R(t) \sim \tilde{R}(t)$. □

6 Connector synthesis (example)

Consider a system providing some given functionality in two modes: *normal* and *backup*. The system consists of four modules: the Backup module A can only perform one action a; the Main module B (Figure 4) can perform an action b corresponding to the normal mode activity, it can also be switched *on* and off, as well as perform an internal (unobservable) error transition *err*; the Monitor module M is a black box, which performs some internal logging by observing the two actions a and b through the corresponding ports a_l and b_l; finally, the black box Controller module C takes the decisions to switch on or off the main module

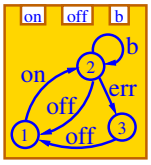


Figure 4: Main module.

through the corresponding ports on_c and off_c, furthermore, it can perform a *test* to check whether the main module can be restarted.

We want to synthesise connectors ensuring the properties below (encoded by Boolean constraints).

- The main and backup actions must be logged: $\dot{a} \Leftrightarrow \dot{a}_l$ and $\dot{b} \Leftrightarrow \dot{b}_l$;
- Only Controller can turn on the Main module: $\dot{on} \Leftrightarrow \dot{on}_c$;
- When Controller switches off the Main module must stop operation: $\dot{off}_c \Rightarrow \dot{off}$ and $\dot{b} \Rightarrow \overline{\dot{off}_c}$;
- Controller can only test the execution of Backup: $\dot{test} \Rightarrow \dot{a}$;
- Backup can only execute when Main is not possible: $\dot{a} \Rightarrow \overline{b} \vee \dot{off}$;
- Main can only switch off when ordered to do so or after a failure: $\dot{off} \Rightarrow \overline{b} \vee \dot{off}_c$;

In order to compute the required glue, we take the conjunction of the above constraints together with the *progress* constraint $\dot{a} \vee \dot{b} \vee \dot{on} \vee \dot{off} \vee \dot{test} \vee \dot{a}_l \vee \dot{b}_l \vee \dot{off}_c \vee \dot{on}_c$ stating that at every round some action must be taken. In order to simplify the resulting connectors, we also use part of the information about the behaviour of the Main module, namely the fact that *on*, on one hand, and b or off, on the other, are mutually exclusive: $on \Rightarrow \overline{b} \wedge \overline{off}$. Finally, we also apply the additional axiom imposed on Boolean constraints: $\dot{p} \Rightarrow p$. We obtain the following global constraint (omitting the conjunction symbol):

$$\begin{aligned}(\dot{a} \Rightarrow \dot{a}_l\, a\, \overline{b} \;\vee\; \dot{a}_l\, a\, \dot{off})(\dot{a}_l \Rightarrow \dot{a}\, a_l)(\dot{b} \Rightarrow \dot{b}_l\, b\, \overline{\dot{off}_c})(\dot{b}_l \Rightarrow \dot{b} b_l)(\dot{on} \Rightarrow \dot{on}_c\, on)(\dot{on}_c \Rightarrow \dot{on}\, on_c)\\ \wedge\, (\dot{off} \Rightarrow off\, \overline{b} \;\vee\; \dot{off}_c\, off)(\dot{off}_c \Rightarrow \dot{off}\, off_c)(\dot{test} \Rightarrow \dot{a}\, test)\\ \wedge\, (on \Rightarrow \overline{b}\, \overline{off})(\dot{a} \;\vee\; \dot{b} \;\vee\; \dot{on} \;\vee\; \dot{off} \;\vee\; \dot{test} \;\vee\; \dot{a}_l \;\vee\; \dot{b}_l \;\vee\; \dot{off}_c \;\vee\; \dot{on}_c)\,.\end{aligned}$$

Recall now that causal rules have the form $p \Rightarrow C$, where $p \in P \cup \dot{P} \cup \overline{P} \cup \{\mathtt{tt}\}$ and C is a DNF Boolean formula on $P \cup \dot{P}$ without negative firing variables. By Proposition 5.13, it is sufficient to consider only the rules with firing or `tt` effects. A system of causal rules is a conjunction of such clauses. Among the constraints above, there are two that do not have this form: $on \Rightarrow \overline{b}\, \overline{off}$ and $\dot{b} \Rightarrow \dot{b}_l\, b\, \overline{\dot{off}_c}$. Rewriting them as $\overline{on} \vee \overline{b}\, \overline{off}$ and $\overline{\dot{b}} \vee \dot{b}_l\, b\, \overline{\dot{off}_c}$, distributing over the conjunction of the rest of the constraints and making some straightforward simplifications, we obtain a disjunction of three systems of causal rules:

$$\begin{array}{llll}
\mathtt{tt} \Rightarrow \dot{a}\, \overline{b}\, \overline{off} \;\vee\; \dot{on} & & \mathtt{tt} \Rightarrow \dot{a}\, \overline{on} \;\vee\; \dot{off} & & \mathtt{tt} \Rightarrow \dot{b}\, \dot{b}_l & \\
\dot{a} \Rightarrow \dot{a}_l\, \overline{b} & & \dot{a} \Rightarrow \dot{a}_l\, \overline{b} \;\vee\; \dot{a}_l\, \dot{off} & & \dot{a} \Rightarrow \mathtt{ff} & \\
\dot{a}_l \Rightarrow \dot{a} & \dot{b} \Rightarrow \mathtt{ff} & \dot{a}_l \Rightarrow \dot{a} & \dot{b} \Rightarrow \mathtt{ff} & \dot{a}_l \Rightarrow \mathtt{ff} & \dot{b} \Rightarrow \mathtt{tt} \\
\dot{on} \Rightarrow \dot{on}_c & \dot{b}_l \Rightarrow \mathtt{ff} & \dot{on} \Rightarrow \mathtt{ff} & \dot{b}_l \Rightarrow \mathtt{ff} & \dot{on} \Rightarrow \mathtt{ff} & \dot{b}_l \Rightarrow \mathtt{tt} \\
\dot{on}_c \Rightarrow \dot{on} & \dot{off} \Rightarrow \mathtt{ff} & \dot{on}_c \Rightarrow \mathtt{ff} & \dot{off} \Rightarrow \overline{b} \;\vee\; \dot{off}_c & \dot{on}_c \Rightarrow \mathtt{ff} & \dot{off} \Rightarrow \mathtt{ff} \\
\dot{test} \Rightarrow \dot{a} & \dot{off}_c \Rightarrow \mathtt{ff} & \dot{test} \Rightarrow \dot{a} & \dot{off}_c \Rightarrow \dot{off} & \dot{test} \Rightarrow \mathtt{ff} & \dot{off}_c \Rightarrow \mathtt{ff}
\end{array}$$

Applying the procedure from [7], we obtain the $\mathscr{T}(P \cup \dot{P} \cup \overline{P})$ trees in Figure 5 and connectors in Figure 6. In terms of classical BIP, one can easily distinguish here two priorities: $x\ a\ a_l \prec b\ b_l$ and $x\ off \prec b\ b_l$ for all x not containing $off\ off_c$. In general, priorities are replaced by local inhibitors. In this example, these appear to characterise states of the Main module. For instance, $\dot{a}\ \dot{a}_l\ \overline{b}\ \overline{off}$ defines possible interactions involving $a\ a_l$ when neither b nor off are possible, i.e. in state 1 (see Figure 4).

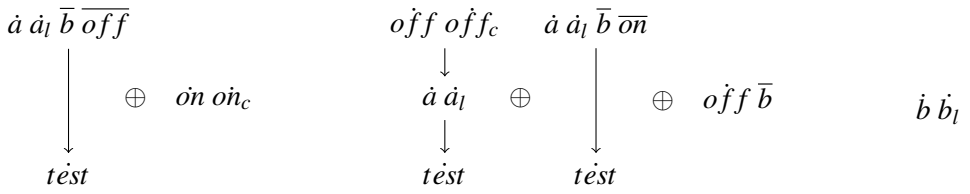


Figure 5: Three causal interaction trees.

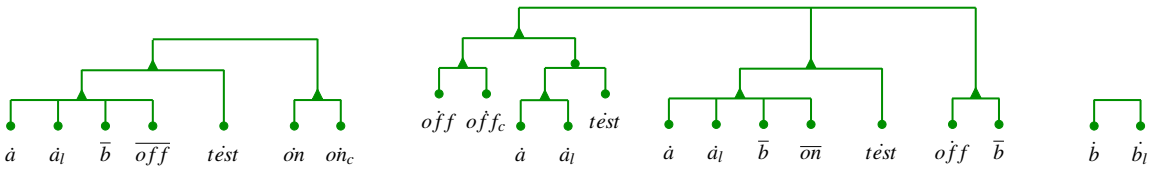


Figure 6: Connectors corresponding to trees from Figure 5.

7 Conclusion

The work presented in this paper relies on a variation of the BIP operational semantics based on the offer predicate introduced in [8]. Glue operators defined using the offer predicate are isomorphic to Boolean constraints on activation and firing port variables $\mathbb{B}[P,\dot{P}]$ with an additional axiom $\dot{p} \Rightarrow p$ [8]. By considering the negation of an activation port variable as a separate *negative* port variable (keeping, however, the axiom $p\,\overline{p} = \mathtt{ff}$), we reinterpret the combination of interaction an priority models on P as an interaction model on $P \cup \dot{P} \cup \overline{P}$. This allows us to apply algebraic theory previously developed for modelling interactions in BIP, to model interactions and priorities simultaneously. In particular, we can synthesise such new connectors from arbitrary $\mathbb{B}[P,\dot{P}]$ Boolean formulæ (in [7] we have shown how to synthesise classical connectors from formulæ on port variables without firing/activation dichotomy).

The equivalence induced by the new operational semantics on the algebras ($x \sim y \overset{\Delta}{\Leftrightarrow} \|x\|(\mathbf{B}) = \|y\|(\mathbf{B})$ for any finite set of behaviours $\mathbf{B}$) is weaker than the standard equivalence induced by the interaction semantics ($x \simeq y \overset{\Delta}{\Leftrightarrow} |x| = |y|$). Extending the axioms of the Algebra of Causal Interaction Trees accordingly, we define normal forms for connectors and causal interaction trees. This, in turn, allows us to simplify the causal rule representation, by considering only rules with firing effects. Algebra extensions are illustrated on a connector synthesis example.

In this paper, we have only extended the axiomatisation of $\mathscr{T}(P \cup \dot{P} \cup \overline{P})$. Studying corresponding extensions for the axiomatisations of other algebras as well as their completeness could be part of the future work. More urgently, we intend to study the differences between the classical BIP semantics and the offer variation. For example, it is clear that the two semantics are equivalent on flat models. The divergence on hierarchical models remains to be characterised.

References

[1] Farhad Arbab, Christel Baier, Frank de Boer, Jan Rutten & Marjan Sirjani (2005): *Synthesis of Reo Circuits for Implementation of Component-Connector Automata Specifications*. In: *Coordination Models and Languages*, *LNCS* 3454, Springer, Berlin / Heidelberg, pp. 236–251, doi:10.1007/b135676.

[2] Farhad Arbab & Sun Meng (2008): *Synthesis of Connectors from Scenario-Based Interaction Specifications*. In: *CBSE'08*, *LNCS* 5282, Springer Berlin/Heidelberg, pp. 114–129, doi:10.1007/978-3-540-87891-9.

[3] Simon Bliudze (2012): *Towards a Theory of Glue*. In: *ICE 2012: Distributed coordination, execution models, and resilient interaction*, *EPTCS* 104, pp. 48–66, doi:10.4204/EPTCS.104.6.

[4] Simon Bliudze & Joseph Sifakis (2007): *The Algebra of Connectors — Structuring Interaction in BIP*. In: *Proc. of the EMSOFT'07*, ACM SigBED, pp. 11–20, doi:10.1145/1289927.1289935.

[5] Simon Bliudze & Joseph Sifakis (2008): *The Algebra of Connectors—Structuring Interaction in BIP*. *IEEE Transactions on Computers* 57(10), pp. 1315–1330, doi:10.1109/TC.2008.26.

[6] Simon Bliudze & Joseph Sifakis (2008): *A Notion of Glue Expressiveness for Component-Based Systems*. In Franck van Breugel & Marsha Chechik, editors: *CONCUR 2008*, *LNCS* 5201, Springer, pp. 508–522, doi:10.1007/978-3-540-85361-9_39.

[7] Simon Bliudze & Joseph Sifakis (2010): *Causal semantics for the algebra of connectors*. *Formal Methods in System Design* 36(2), pp. 167–194, doi:10.1007/s10703-010-0091-z.

[8] Simon Bliudze & Joseph Sifakis (2011): *Synthesizing Glue Operators from Glue Constraints for the Construction of Component-Based Systems*. In Sven Apel & Ethan Jackson, editors: 10^{th} *International Conference on Software Composition*, *LNCS* 6708, Springer, pp. 51–67, doi:10.1007/978-3-642-22045-6_4.

[9] Borzoo Bonakdarpour, Marius Bozga, Mohamad Jaber, Jean Quilbeuf & Joseph Sifakis (2010): *From high-level component-based models to distributed implementations*. In: *Proceedings of the tenth ACM international conference on Embedded software*, EMSOFT '10, ACM, New York, NY, USA, pp. 209–218, doi:10.1145/1879021.1879049.

[10] Marius Bozga, Mohamad Jaber & Joseph Sifakis (2009): *Source-to-source architecture transformation for performance optimization in BIP*. In: *Industrial Embedded Systems, 2009. SIES '09. IEEE International Symposium on*, pp. 152–160, doi:10.1109/SIES.2009.5196211.

[11] Roberto Bruni, Ivan Lanese & Ugo Montanari (2006): *A basic algebra of stateless connectors*. *Theor. Comput. Sci.* 366(1), pp. 98–120, doi:10.1016/j.tcs.2006.07.005.

[12] Roberto Bruni, Hernn Melgratti & Ugo Montanari (2012): *Connector Algebras, Petri Nets, and BIP*. In Edmund Clarke, Irina Virbitskaite & Andrei Voronkov, editors: *Perspectives of Systems Informatics*, *Lecture Notes in Computer Science* 7162, Springer Berlin Heidelberg, pp. 19–38, doi:10.1007/978-3-642-29709-0_2.

[13] Dave Clarke, José Proença, Alexander Lazovik & Farhad Arbab (2009): *Deconstructing Reo*. *ENTCS* 229(2), pp. 43–58, doi:10.1016/j.entcs.2009.06.028.

[14] Paola Inverardi & Simone Scriboni (2001): *Connectors Synthesis for Deadlock-Free Component-Based Architectures*. In: *ASE '01*, IEEE Computer Society, Washington, DC, USA, pp. 174–181.

[15] Ivan Lanese, Cátia Vaz & Carla Ferreira (2010): *On the Expressive Power of Primitives for Compensation Handling*. In AndrewD. Gordon, editor: *Programming Languages and Systems*, *Lecture Notes in Computer Science* 6012, Springer Berlin Heidelberg, pp. 366–386, doi:10.1007/978-3-642-11957-6_20.

[16] Joseph Sifakis (2005): *A Framework for Component-based Construction*. In: 3^{rd} *IEEE Int. Conf. on Software Engineering and Formal Methods (SEFM05)*, pp. 293–300, doi:10.1109/SEFM.2005.3. Keynote talk.

[17] Pawel Sobocinski (2009): *A non-interleaving process calculus for multi-party synchronisation*. In Filippo Bonchi, Davide Grohmann, Paola Spoletini & Emilio Tuosto, editors: *ICE*, *EPTCS* 12, pp. 87–98, doi:10.4204/EPTCS.12.6.