Towards a Theory of Glue

Simon Bliudze
École Polytechnique Fédérale de Lausanne
Rigorous System Design Laboratory
INJ Building, Station 14, 1015 Lausanne, Switzerland
simon.bliudze@epfl.ch

We propose and study the notions of *behaviour type* and *composition operator* making a first step towards the definition of a formal framework for studying behaviour composition in a setting sufficiently general to provide insight into how the component-based systems should be modelled and compared. We illustrate the proposed notions on classical examples (Traces, Labelled Transition Systems and Coalgebras). Finally, the definition of *memoryless glue operators*, takes us one step closer to a formal understanding of the separation of concerns principle stipulating that computational aspects of a system should be localised within its atomic components, whereas coordination layer responsible for managing concurrency should be realised by memoryless glue operators.

1 Introduction

Component-based design is central in system engineering. Complex systems are built by assembling components. Large components are obtained by "gluing" together simpler ones. "Gluing" can be considered as an operation on sets of components.

Component-based techniques have seen significant development, especially through the use of object technologies supported by languages such as C++, Java, and standards such as UML and CORBA. There exist various component frameworks encompassing a large variety of mechanisms for composing components. They focus rather on the way components interact than on their internal behaviour. We lack adequate notions of expressiveness to compare the merits and weaknesses of these frameworks. For a meaningful and systematic comparison of component frameworks to be possible, one needs a sufficiently abstract formalisation of the notions of both behaviour and glue.

These notions should capture the properties essential for reasoning about system composition. Intuitively, one can think of coordination as imposing constraints on the joint behaviour of the components of the system [9, 12, 32]. Imposing coordination constraints means "reducing" the joint behaviour of the involved components. Beyond the simple definition of a component model, this requires the following two questions to be answered:

- *What is the behaviour of several "parallel" components without any coordination constraints?*
- *How does one compare two behaviours?*

We argue that these aspects, as well as the notions of common behaviour of two components and minimal behaviour possible in a given framework, cannot be dissociated from the notion of behaviour as a whole.

There are several goals, for which the work in this paper is a starting point.

- There is a long-standing debate with a plethora of points of view about *separation of concerns* [11, 13, 14, 21, 26] and in what sense it should be respected. In [6, 8], we have advocated for an approach respecting a separation of concerns principle, whereby all "computation" performed by the system is localised within its constituent atomic components, whereas the coordination layer

To appear in EPTCS.

responsible for managing the parallelism consists of *memoryless* glue operators. Separation of concerns applied in the context of the BIP framework resulted in a very powerful deadlock detection tool D-Finder [5]. Furthermore, it has allowed us to reduce a hard (sometimes undecidable) problem of synthesis of reactive systems [20, 25] to a less ambitious, but more tractable design methodology [9]. This separation of concerns principle is rather fragile: in [15], it is shown that a slight extension of the BIP glue renders it Turing complete, which makes it possible to construct non-trivial systems without a single atomic component. We speculate that, in the setting proposed in the present paper, operators such as prefixing and choice are not glue operators. This hypothesis, should it be verified, would give formal grounds to our view of the separation of concern principle.

- Another goal is to define a generic setting for an expressiveness study. In [7], we have proposed a first notion of glue expressiveness for component-based systems, generalising the concepts presented in [28] that guided the design of the BIP (Behaviour, Interaction, Priority) framework [4]. However, this approach lacks abstraction, since it strongly depends on the choice of the formalism for modelling both behaviour (Labelled Transition Systems; LTS) and glue (Structural Operational Semantics; SOS [24]), which makes the expressiveness comparison questionable and dependent on ad-hoc manipulation to make different frameworks comparable. For instance, in a more recent paper [9], we have proposed a slightly modified formal model of component behaviour in BIP. This entailed a corresponding modification to the definition of glue operators, which resulted in a framework only partially comparable to that in [7]. A general definition of behaviour and glue, as in the present paper, is necessary to solve this problem.
- Finally, a common practice in system engineering consists in applying existing solutions (architectures) to given components to ensure behavioural properties (mutual exclusion, scheduling, communication protocols, etc.). These architectures can be modelled as composition operators. Their simultaneous application should then ensure the application of the coordination constraints imposed by both operators. Hence, it is important to understand when and how these architectures can be combined. This brings forward another question: *How does one model the simultaneous application of several composition operators and what are the conditions ensuring the non-interference among them?*

This paper is a modest first step towards the above goals. It focuses primarily on the notions of *behaviour types* and *composition operators* defined in such a way as to allow reasoning about their essential properties in a setting sufficiently general to provide insight into how component-based systems should be modelled and compared.

To allow answering the questions that we have emphasised above, a behaviour type $\mathscr{B}$ must explicitly comprise certain elements beyond the minimal component model such as LTS. As illustrated, for instance, by CCS and SCCS [22], different parallel composition operators can be defined on the same objects. Hence a parallel composition operator $\|$ must be defined as part of a behaviour type. In order to avoid confusion with any of the existing parallel composition operators, we use the term *maximal interaction operator*, since, intuitively, the operator defines the maximal set of interactions between two components in absence of coordination constraints.

In order to address the question of comparing the behaviour, we require that two preorders be defined: a *simulation preorder* $\sqsubseteq$ and a *semantic preorder* $\preccurlyeq$. Intuitively, simulation preorder relates two components if one of them performs only actions that can also be performed by the other, thus generalising the classical simulation relation and allowing a formalisation of the notion that composing component behaviour amounts to imposing coordination constraints. The role of the semantic preorder is to relate components that behave in a similar manner, in particular, the equivalence $\simeq$ induced by this preorder

is a congruence with respect to composition operators, thus generalising such classical notions as ready simulation equivalence and bisimilarity.

Finally, we need a meet operator $\otimes$, rendering $(\mathscr{B}/\simeq, \preccurlyeq, \otimes)$ a meet-semilattice, to talk about simultaneous application of two composition operators (Section 3.2). Let f_1 and f_2 be two composition operators, which we would like to apply "simultaneously" to a behaviour B. Viewing composition operators as constraints on B, simultaneous application of the two constraints corresponds intuitively to applying their conjunction. However, this idea does not fit into the functional view of operators, since neither $f_1(f_2(B))$ nor $f_2(f_1(B))$ need correspond to the conjunction of the two constraints. Moreover, neither of these two behaviours need be defined, in particular, since the arity constraints of the operators are not respected in general. A more direct approach consists in considering the *maximal common behaviour* of $f_1(B)$ and $f_2(B)$, which is precisely $f_1(B) \otimes f_2(B)$.

The paper is structured as follows. Section 2 introduces the notion of behaviour type and provides three examples: traces, LTS and coalgebras of a particular type. Section 3 presents the notion of composition operators and some of their properties. In Section 4, we discuss some related and future work, then we conclude in Section 5.

2 Behaviour types

Definition 2.1 (Behaviour type)**.** A *behaviour type* over $\mathscr{B}$ is a tuple $(\mathscr{B}, \|, \sqsubseteq, \preccurlyeq, \otimes, \mathbf{0})$ consisting of the following data:

- A monoid $(\mathscr{B}, \|, \mathbf{0})$, where $\| : \mathscr{B}^2 \to \mathscr{B}$ is a totally defined associative binary operator with $\mathbf{0} \in \mathscr{B}$ the neutral element;
- A preorder $\sqsubseteq \subseteq \mathscr{B} \times \mathscr{B}$, such that
  1. it is preserved by $\|$, i.e. for any $B_1, B_2, B_3 \in \mathscr{B}$, $B_2 \sqsubseteq B_3$ implies $B_1 \parallel B_2 \sqsubseteq B_1 \parallel B_3$,
  2. for any $B \in \mathscr{B}$, holds $\mathbf{0} \sqsubseteq B$;
- A preorder $\preccurlyeq \subseteq \mathscr{B} \times \mathscr{B}$ and a meet operator $\otimes : \mathscr{B} \times \mathscr{B} \to \mathscr{B}$, such that
  3. $\preccurlyeq$ is preserved by $\|$, i.e. for any $B_1, B_2, B_3 \in \mathscr{B}$, $B_2 \preccurlyeq B_3$ implies $B_1 \parallel B_2 \preccurlyeq B_1 \parallel B_3$,
  4. $(\mathscr{B}/\simeq, \preccurlyeq, \otimes)$, with $\simeq \stackrel{def}{=} \preccurlyeq \cap \preccurlyeq^{-1}$, is a meet-semilattice.

Elements of $\mathscr{B}$ are *behaviours*. $\|$ is the *maximal interaction operator*. Intuitively, $B_1 \parallel B_2$ is the parallel composition of B_1 and B_2 in absence of any coordination constraints. $\sqsubseteq$ is the *simulation preorder*. It has the same meaning as the classical simulation preorder, i.e. $B_1 \sqsubseteq B_2$ means that "B_2 can perform at least the same executions as B_1". $\preccurlyeq$ is the *semantic preorder*. Intuitively, $B_1 \preccurlyeq B_2$ means that B_2 can act as B_1 in the same coordination context. $B_1 \otimes B_2$ is the greatest behaviour realisable by both B_1 and B_2.

Often behaviour definition explicitly involves an interface, consisting at least of the set of actions a component can perform. In this context, it is usually assumed that a universal set of actions is given. Furthermore, it is convenient also to assume that this set is equipped with some lattice structure. A straightforward example consists in considering finite sets rather than individual actions with the lattice structure induced by set union and intersection.

2.1 Example: Traces

Let $\mathbf{A}$ be a universal set of actions and *Traces* be the set of pairs $B = (A, T)$, where $A \subseteq \mathbf{A}$ is a set of actions and $T \subseteq A^*$ is a prefix-closed set of traces. In particular, $\varepsilon \in T$, where ε is the empty word.

For $B_1 = (T_1, A_1)$ and $B_2 = (T_2, A_2)$, we define the maximal interaction operator $\|$, as the interleaving of actions of the two behaviours, by putting $B_1 \parallel B_2 \stackrel{def}{=} (A_1 \cup A_2, T)$, where

$$T \stackrel{def}{=} \left\{ w = (w_i)_{i=1}^n \in A^* \,\middle|\, \exists I \subseteq [1,n] : (w_i)_{i\in I} \in T_1 \wedge (w_i)_{i\notin I} \in T_2 \right\}. \tag{1}$$

This makes $(Traces, \|, \mathbf{0})$, with $\mathbf{0} = (\emptyset, \{\varepsilon\})$, a commutative monoid.

Equation (1) defines the maximal interaction operator through interleaving of traces. The corresponding semantic preorder is defined through the notion of sub-sequence, which we denote $\propto$ and define by putting, for $v, w \in A^*$,

$$v \propto w \stackrel{def}{\iff} \exists I \subseteq [1, |w|] : v = (w_i)_{i\in I},$$

where $|w|$ is the length of w. Furthermore, for an alphabet A and a language T, we define $T\wr_A \stackrel{def}{=} \{v \in A^* \,|\, \exists w \in T : v \propto w\}$ (notice that, if $T \subseteq A^*$, we have $T \subseteq T\wr_A$). Finally, we define the semantic preorder and the operator $\otimes$ as follows:

$$B_1 \preccurlyeq B_2 \stackrel{def}{\iff} A_1 \subseteq A_2 \ \wedge\ T_1 \subseteq T_2\wr_{A_1}, \tag{2}$$

$$B_1 \otimes B_2 \stackrel{def}{=} (A_1 \cap A_2, T_1\wr_{A_1\cap A_2} \cap T_2\wr_{A_1\cap A_2}), \tag{3}$$

the preorder $\sqsubseteq$ coincides with $\preccurlyeq$.

Clearly, for two alphabets $A \subseteq B$ and a set of traces T, holds the equality $(T\wr_B)\wr_A = T\wr_A$, which implies immediately that the operator $\otimes$ defined by (3) is, indeed, the meet operator with respect to the preorder $\preccurlyeq$. Hence, $(Traces/\simeq, \preccurlyeq, \otimes)$ is a meet-semilattice. To prove that $(Traces, \|, \sqsubseteq, \preccurlyeq, \otimes, \mathbf{0})$ is a behaviour type we only have to show that condition 1 (and condition 3) of Definition 2.1 holds.

Proposition 2.2. *Maximal interaction operator $\|$ defined by (1) preserves the semantic preorder $\preccurlyeq$ defined by (2).*

Proof. For $i = 1,2,3$, let $B_i = (A_i, T_i) \in Traces$ be such that $B_2 \preccurlyeq B_3$. By definition (2) of $\preccurlyeq$, we then have $A_2 \subseteq A_3$ and $T_2 \subseteq T_3\wr_{A_2}$. Consequently, $A_1 \cup A_2 \subseteq A_1 \cup A_3$.

Let $B_1 \parallel B_2 = (A_1 \cup A_2, T_{12})$ and $B_1 \parallel B_3 = (A_1 \cup A_3, T_{13})$, and consider $w = (w_i)_{i=1}^n \in T_{12}$. By (1), there exists $I \subset [1,n]$ such that $(w_i)_{i\in I} \in T_1$ and $(w_i)_{i\notin I} \in T_2$. Since $T_2 \subseteq T_3\wr_{A_2}$, there exists $v \in T_3$ such that $(w_i)_{i\notin I} \propto v$. Denoting by $\tilde{v}$ the projection of v on $A_1 \cup A_2$, we have $\tilde{v} \propto v$, which implies $\tilde{v} \in T_3\wr_{A_1\cup A_2}$. It is clear that $(w_i)_{i\notin I} \propto \tilde{v}$ and that one can interleave $\tilde{v}$ with $(w_i)_{i\in I}$ in such a manner that the positions of w_i, for all $i \notin I$, be preserved. Denoting the obtained trace by u, we obviously obtain $w \propto u \in T_1\wr_{A_1\cup A_2} T_3\wr_{A_1\cup A_2} = T_{13}\wr_{A_1\cup A_2}$. Hence $T_{12} \subseteq T_{13}\wr_{A_1\cup A_2}$ and $B_1 \parallel B_2 \preccurlyeq B_1 \parallel B_3$. □

2.2 Example: Labelled Transition Systems

Let again $\mathbf{A}$ be a universal set of actions and let LTS be the set of triples $B = (Q, A, \rightarrow)$, with $Q \neq \emptyset$ a finite set of states, $A \subseteq \mathbf{A}$ a set of actions, and $\rightarrow \subseteq Q \times 2^A \times Q$ a set of transitions. The infix notation $q \stackrel{a}{\rightarrow} q'$ is commonly used to denote a transition $(q, a, q') \in \rightarrow$. Note that, with this definition, transitions are labelled with *interactions*, i.e. sets of actions. This approach is particularly well suited for characterising behaviour of components that can communicate through several ports during a single transition [6]. Such behaviour naturally arises when composite components are assembled by parallel composition of subcomponents. Thus, labels are combined by set union, often denoted by juxtaposition.

As in the previous example, we take coinciding preorders $\sqsubseteq$ and $\preccurlyeq$. For $B_1 = (Q_1, A_1, \rightarrow_1)$ and $B_2 = (Q_2, A_2, \rightarrow_2)$, we define the maximal interaction operator $\|$ and the semantic preorder $\preccurlyeq$ as follows. We put $B_1 \parallel B_2 \stackrel{def}{=} (Q_1 \times Q_2,\ A_1 \cup A_2,\ \rightarrow)$, where $\rightarrow$ is the minimal transition relation satisfying the following SOS rules:

$$\frac{q_1 \stackrel{a}{\rightarrow}_1 q_1'}{q_1 q_2 \stackrel{a}{\longrightarrow} q_1' q_2}, \quad \frac{q_2 \stackrel{b}{\rightarrow}_2 q_2'}{q_1 q_2 \stackrel{b}{\longrightarrow} q_1 q_2'}, \quad \frac{q_1 \stackrel{a}{\rightarrow}_1 q_1' \quad q_2 \stackrel{b}{\rightarrow}_2 q_2'}{q_1 q_2 \stackrel{a \cup b}{\longrightarrow} q_1' q_2'}. \tag{4}$$

With $\mathbf{0} = (\{1\}, \emptyset, \emptyset)$, it is straightforward to conclude that $(LTS, \|, \mathbf{0})$ is a commutative monoid.

For $B_1 = (Q_1, A_1, \rightarrow_1)$ and $B_2 = (Q_2, A_2, \rightarrow_2)$, such that $A_1 \subseteq A_2$ consider the maximal *simulation* relation $R \subseteq Q_1 \times Q_2$ such that

$$q_1 R q_2 \implies \forall q_1 \stackrel{a}{\rightarrow}_1 q_1',\ \exists q_2' \in Q_2, b \subseteq A_2 : \left(q_2 \stackrel{b}{\rightarrow} q_2' \ \wedge\ a \subseteq b \ \wedge\ q_1' R q_2' \right). \tag{5}$$

The semantic preorder $\preccurlyeq$, is defined by putting $B_1 \preccurlyeq B_2$ iff R is total on Q_1.

Finally, we define the meet operator by putting, for $B_1 = (Q_1, A_1, \rightarrow_1)$ and $B_2 = (Q_2, A_2, \rightarrow_2)$, $B_1 \otimes B_2 \stackrel{def}{=} (Q_1 \times Q_2, A_1 \cap A_2, \rightarrow)$, where $\rightarrow$ is the minimal transition relation satisfying the rule

$$\frac{q_1 \stackrel{a}{\rightarrow}_1 q_1' \quad q_2 \stackrel{b}{\rightarrow}_2 q_2' \quad a \cap b \neq \emptyset}{q_1 q_2 \stackrel{a \cap b}{\longrightarrow} q_1' q_2'}. \tag{6}$$

Conditions 1–3 of the Definition 2.1 clearly hold, and, to show that $(LTS, \|, \sqsubseteq, \preccurlyeq, \otimes, \mathbf{0})$ is a behaviour type, we only have to prove the following proposition.

Proposition 2.3. *$(LTS/\simeq, \preccurlyeq, \otimes)$ is a meet-semilattice.*

Proof. To prove the proposition, we have to show that, for any $\tilde{B}, B_1, B_2 \in LTS$, holds $B_1 \otimes B_2 \preccurlyeq B_1, B_2$ and $\tilde{B} \preccurlyeq B_1, B_2$ implies $\tilde{B} \preccurlyeq B_1 \otimes B_2$.

Let $\tilde{B} = (\tilde{Q}, \tilde{A}, \rightarrow)$, $B_i = (Q_i, A_i, \rightarrow)$ (for $i = 1, 2$) and $B_1 \otimes B_2 = (Q_1 \times Q_2, A_1 \cap A_2, \rightarrow)$ as defined above (for clarity, we skip the indices on the transition relations).

1. First of all $A_1 \cap A_2 \subseteq A_1, A_2$. By symmetry, it is sufficient to show that $B_1 \otimes B_2 \preccurlyeq B_1$. Let $R \subseteq (Q_1 \times Q_2) \times Q_1$ be the projection on the first component.

Consider a state $q = (q_1, q_2) \in Q_1 \times Q_2$. By definition of $\otimes$, for any $q \stackrel{c}{\rightarrow} q'$ in $B_1 \otimes B_2$, there exist transitions $q_1 \stackrel{a}{\rightarrow} q_1'$ and $q_2 \stackrel{b}{\rightarrow} q_2'$ such that $a \cap b = c$ and $q' = (q_1', q_2')$. Hence, we have $c \subseteq a$ and $q' R q_1'$, satisfying the implication (5) is satisfied. Since R is total on $Q_1 \times Q_2$, we have $B_1 \otimes B_2 \preccurlyeq B_1$.

2. Since $\tilde{B} \preccurlyeq B_1, B_2$, there exist total relations $R_i \subseteq \tilde{Q} \times Q_i$ (for $i = 1, 2$) satisfying the implication (5). We define a relation $R \subseteq \tilde{Q} \times (Q_1, Q_2)$ by putting $\tilde{q} R (q_1, q_2)$ iff $\tilde{q} R_i q_i$, for both $i = 1, 2$. Since $\tilde{A} \subseteq A_1 \cap A_2$ and R is clearly total, we only have to show that (5) is satisfied.

Let $\tilde{q} \in \tilde{Q}$ and $(q_1, q_2) \in Q_1 \times Q_2$ be related by R. Consider a transition $\tilde{q} \stackrel{c}{\rightarrow} \tilde{q}'$ in $\tilde{B}$. Since $\tilde{q} R_i q_i$, for $i = 1, 2$, there exist two transitions $q_1 \stackrel{a}{\rightarrow} q_1'$ and $q_2 \stackrel{b}{\rightarrow} q_2'$ such that $c \subseteq a, b$ and $\tilde{q}' R_i q_i'$. We then have $c \subseteq a \cap b$ and $\tilde{q}' R (q_1', q_2')$, which satisfies (5) and proves the proposition. □

2.3 Example: Coalgebras

Coalgebras [27] provide a general framework for unifying various state-based behaviour models such as both deterministic and non-deterministic automata, LTS, Mealy machines, etc. The presentation below is largely inspired by that in [29].

Table 1: Extensions of set operations to mappings ($f : X \to Y$, $f_1 : X_1 \to Y_1$ and $f_2 : X_2 \to Y_2$)

$f_1 \times f_2 : X_1 \times X_2 \to Y_1 \times Y_2$ $(x_1, x_2) \mapsto \Big(f_1(x_1), f_2(x_2)\Big)$	$f_1 + f_2 : X_1 + X_2 \to Y_1 + Y_2$ $x \mapsto \kappa_i(f_i(x'))$
$f^A : X^A \to Y^A$ $g \mapsto f \circ g$	$\mathscr{P}_\omega(f) : \mathscr{P}_\omega(X) \to \mathscr{P}_\omega(Y)$ $S \mapsto \{f(x) \,\vert\, x \in S\}$

Table 2: Structural induction definition of NDF' functors (for any set X and any mapping $f : X \to Y$)

$F(X)$	$F(f)$	Structure of F
X	f	$F = \mathbf{Id}$
$\mathbf{B}$	$id_{\mathbf{B}}$	$F = \mathbf{B}$
$F_1(X) \times F_2(X)$	$F_1(f) \times F_2(f)$	$F = F_1 \times F_2$
$G(X)^{\mathbf{A}}$	$G(f)^{\mathbf{A}}$	$F = G^{\mathbf{A}}$
$\mathscr{P}_\omega(G(X))$	$\mathscr{P}_\omega(G(f))$	$F = \mathscr{P}_\omega(G)$

We recall, in Table 1, the extensions from sets to mappings of the following four operations: Cartesian product $X \times Y$, disjoint union $X + Y$, exponentiation X^A (set of mappings $A \to X$) and powerset $\mathscr{P}_\omega(X)$ (set of finite subsets of X).

Let F be a functor on **Set**. An F-coalgebra is a pair $(S, f : S \to F(S))$, where S is the set of states. The mapping f determines the transition structure of (S, f), whereas the functor F is the *type* of the coalgebra.

Example 2.4. Functors $M = (\mathbf{B} \times \mathbf{Id})^{\mathbf{A}}$, $D = \mathbb{B} \times (1 + \mathbf{Id})^{\mathbf{A}}$ and $N = \mathbb{B}^2 \times \mathscr{P}_\omega(\mathbf{Id})^{\mathbf{A}}$, where $\mathbb{B} = \{0, 1\}$ and $1 = \{*\}$, are respectively the types of the coalgebraic definitions for, respectively, input-enabled Mealy machines with the input domain $\mathbf{A}$ and output domain $\mathbf{B}$, deterministic and non-deterministic automata. For instance, in a D-coalgebra $(S, f : S \to D(S))$, the components of the mapping $f = \omega \times \delta$, with $\omega : S \to \mathbb{B}$ and $\delta : S \to (1 + S)^A$, determine, for each state $s \in S$, whether it is a final state (mapping ω) and the set $\left\{ \Big(s, a, \delta(s)(a)\Big) \,\middle|\, a \in A, \delta(s)(a) \in S \right\}$ of transitions leaving s. The case where there is no transition labelled a leaving the state s is reflected by $\delta(s)(a) = * \in 1$. Notice that, in this example, we do not account for initial states of automata; determinism, here, means that only one transition is possible from any given state with a given action.

We define a class NDF' of *non-deterministic functors*[1] on the category **Set** of sets, defined by the following grammar

$$F ::= \mathbf{Id} \,|\, \mathbf{B} \,|\, F \times F \,|\, F^{\mathbf{A}} \,|\, \mathscr{P}_\omega(F)\,, \tag{7}$$

where $\mathbf{A}$ is the universal set of actions, $\mathbf{Id}$ is the identity functor and $\mathbf{B} \neq \emptyset$ is a join-semilattice with bottom. Typical examples of semilattices used to define non-deterministic functors are $\mathbb{B} = \{0, 1\}$ with $0 \vee 1 = 1$ and $0 \wedge 1 = 0$; and the trivial lattice $1 = \{*\}$. The functors in $NDF-$ are defined by structural induction in Table 2 (cf. also Table 1).

Consider $F \in NDF'$ and let (S_1, f_1) and (S_2, f_2) be two F-coalgebra. For a relation $R \subseteq S_1 \times S_2$, we

[1] NDF' is a subclass of the class NDF of non-deterministic functors as defined, for example, in [29].

define a relation $\leq_R^F \subseteq F(S_1) \times F(S_2)$ by structural induction on F. For $x \in F(S_1)$ and $y \in F(S_2)$, we put

$$x \leq_R^F y \quad \stackrel{def}{\Longleftrightarrow} \quad \begin{cases} (x,y) \in R, & \text{if } F = \mathbf{Id}, \\ x \vee y = y, & \text{if } F = \mathbf{B}, \\ x_1 \leq_R^{F_1} y_1 \wedge x_2 \leq_R^{F_2} y_2, & \text{if } F = F_1 \times F_2,\ x = (x_1,x_2) \text{ and } y = (y_1,y_2), \\ \forall a \in \mathbf{A},\ x(a) \leq_R^G y(a), & \text{if } F = G^{\mathbf{A}}, \\ \forall x' \in x, \exists y' \in y : x' \leq_R^G y', & \text{if } F = \mathscr{P}_\omega(G). \end{cases}$$

We define the simulation preorder on F-coalgebra by putting $(S_1,f_1) \sqsubseteq (S_2,f_2)$, iff there exists a relation $R \subseteq S_1 \times S_2$ total on S_1 and such that $\forall (s_1,s_2) \in R,\ f_1(s_1) \leq_R^F f_2(s_2)$.

Definition 2.5 (Coalgebra homomorphism)**.** An *F-homomorphism* of two F-coalgebras (S_1,f_1) and (S_2,f_2) is a mapping $h : S_1 \to S_2$ preserving the transition structure, i.e. such that the following diagram commutes ($f_2 \circ h = F(h) \circ f_1$)

$$\begin{array}{ccc} S_1 & \xrightarrow{\ h\ } & S_2 \\ {\scriptstyle f_1}\downarrow & & \downarrow{\scriptstyle f_2} \\ F(S_1) & \xrightarrow[F(h)]{} & F(S_2) \end{array}$$

where $F(h)$ is the image of h by the functor F. In particular, for $F \in NDF'$, $F(h)$ is defined by the second column in Table 2.

Definition 2.6 (Bisimulation)**.** Let (S_1,f_1) and (S_2,f_2) be two F-coalgebras. A relation $R \subseteq S_1 \times S_2$ is a *bisimulation* iff there exists a *witness mapping* $g : R \to F(R)$, such that projections $\pi_i : R \to S_i$, for $i = 1,2$, are coalgebra homomorphisms, i.e. the following diagram commutes

$$\begin{array}{ccccc} S_1 & \xleftarrow{\ \pi_1\ } & R & \xrightarrow{\ \pi_2\ } & S_2 \\ {\scriptstyle f_1}\downarrow & & {\scriptstyle g}\downarrow & & \downarrow{\scriptstyle f_2} \\ F(S_1) & \xleftarrow[F(\pi_1)]{} & F(R) & \xrightarrow[F(\pi_2)]{} & F(S_2) \end{array} \tag{8}$$

Remark 2.7. In the context of Definition 2.6, if $F = F_1 \times F_2$, $g = g_1 \times g_2$ and $f_i = f_i^1 \times f_i^2$ $(i = 1,2)$, the left diagram in (9) commutes, for $j = 1,2$, and defines a bisimulation on F_j-coalgebras (S_1,f_1^j) and (S_2,f_2^j). Similarly, for $F = G^{\mathbf{A}}$, the right diagram in (9) commutes, for all $a \in \mathbf{A}$, and defines a bisimulation on G-coalgebras $(S_1,f_1(a))$ and $(S_2,f_2(a))$.

$$\begin{array}{ccccc} S_1 & \xleftarrow{\ \pi_1\ } & R & \xrightarrow{\ \pi_2\ } & S_2 \\ {\scriptstyle f_1^j}\downarrow & & {\scriptstyle g_j}\downarrow & & \downarrow{\scriptstyle f_2^j} \\ F_j(S_1) & \xleftarrow[F_j(\pi_1)]{} & F_j(R) & \xrightarrow[F_j(\pi_2)]{} & F_j(S_2) \end{array} \qquad \begin{array}{ccccc} S_1 & \xleftarrow{\ \pi_1\ } & R & \xrightarrow{\ \pi_2\ } & S_2 \\ {\scriptstyle f_1(a)}\downarrow & & {\scriptstyle g(a)}\downarrow & & \downarrow{\scriptstyle f_2(a)} \\ G(S_1) & \xleftarrow[G(\pi_1)]{} & G(R) & \xrightarrow[G(\pi_2)]{} & G(S_2) \end{array} \tag{9}$$

Lemma 2.8 ([27])**.** *Consider coalgebras (S,f), (S_1,f_1) and (S_2,f_2) with coalgebra homomorphisms $g : S \to S_1$ and $h : S \to S_2$. The image $(g,h)(S) = \{(g(s),h(s)) \,|\, s \in S\}$ is a bisimulation on S_1 and S_2.*

Theorem 2.9 ([27])**.** *The union $\bigcup_k R_k$ of a family $\{R_k\}_k$ of bisimulations on coalgebras (S_1,f_1) and (S_2,f_2) is again a bisimulation.*

Lemma 2.10. *Consider F-coalgebras (S_1, f_1), (S_2, f_2) and (S_3, f_3) with $F \in NDF'$. The composition $R_1 \circ R_2$ of two bisimulations $R_1 \subseteq S_1 \times S_2$ and $R_2 \subseteq S_2 \times S_3$ is a bisimulation on S_1 and S_3.*

Proof. This lemma follows immediately from the following two facts: 1) comosition of bisimulations on F-coalgebras is a bisimulation when F preserves weak pullbacks [27]; 2) NDF' functors preserve weak pullbacks [30]. □

Although Theorem 2.9 allows us to speak of the maximal bisimulation on two coalgebras, the witness mapping on this maximal bisimulation need not be unique. The following proposition provides a construction of a canonical witness mapping on a bisimulation of two coalgebras.

Proposition 2.11. *Consider a bisimulation $R \subseteq S_1 \times S_2$ on two F-coalgebras (S_1, f_1) and (S_2, f_2) with $F \in NDF'$. Let $g_1, g_2 : R \to F(R)$ be two witness mappings. The mapping $g_1 \overset{F}{\cup} g_2 : R \to F(R)$ defined below is again a witness for bisimulation R.*

$$g_1 \overset{F}{\cup} g_2 \quad \overset{def}{=} \quad \begin{cases} g_1 = g_2, & \text{if } F = \mathbf{Id} \text{ or } F = \mathbf{B} \\ (g_1^1 \overset{F_1}{\cup} g_2^1) \times (g_1^2 \overset{F_2}{\cup} g_2^2) & \text{if } F = F_1 \times F_2, \quad g_i = g_i^1 \times g_i^2 \ (i = 1,2) \\ \lambda a. \Big(g_1(a) \overset{G}{\cup} g_2(a) \Big), & \text{if } F = G^{\mathbf{A}} \\ g_1 \cup g_2, & \text{if } F = \mathscr{P}_\omega(G), \end{cases} \tag{10}$$

with $g_1 \cup g_2 : (s_1, s_2) \mapsto g_1(s_1, s_2) \cup g_2(s_1, s_2)$.

Proof. To prove the proposition, we have to show that $\overset{F}{\cup}$ is well defined, i.e. that $g_1 = g_2$ for either $F = \mathbf{Id}$ or $F = \mathbf{B}$, and that projections $\pi_i : R \to S_i$ $(i = 1,2)$ are coalgebra homomorphisms from $(R, g_1 \overset{F}{\cup} g_2)$ to (S_i, f_i).

For $F = \mathbf{Id}$, we have, $g_1 = (\pi_1 \circ g_1, \pi_2 \circ g_1) = (f_1 \circ \pi_1, f_2 \circ \pi_2) = (\pi_1 \circ g_2, \pi_2 \circ g_2) = g_2$.

For $F = \mathbf{B}$, we have $F(\pi_1) = id_{\mathbf{B}}$ and $g_1 = id_{\mathbf{B}} \circ g_1 = f_1 \circ \pi_1 = id_{\mathbf{B}} \circ g_2 = g_2$.

The proof that π_i are coalgebra homomorphisms is by structural induction on F. It is trivial for the cases $F = \mathbf{Id}$ and $F = \mathbf{B}$. For the cases $F = F_1 \times F_2$ and $F = G^{\mathbf{A}}$, it follows immediately from Remark 2.7. Finally, for $F = \mathscr{P}_\omega(G)$, we have $\mathscr{P}_\omega(G)(\pi_i) \circ (g_1 \cup g_2) = (\mathscr{P}_\omega(G)(\pi_i) \circ g_1) \cup (\mathscr{P}_\omega(G)(\pi_i) \circ g_2) = (f_i \circ \pi_i) \cup (f_i \circ \pi_i) = (f_i \circ \pi_i)$. □

We define the semantic preorder on F-coalgebras with $F \in NDF'$ by putting $(S_1, f_1) \preccurlyeq (S_2, f_2)$ iff they have a bisimulation total on S_1. The meet operator is defined by putting $(S_1, f_1) \otimes (S_2, f_2) \overset{def}{=} (R, g)$, where R is their maximal bisimulation and $g : R \to F(R)$ is the witness mapping maximal with respect to $\overset{F}{\cup}$ defined by (10). Observe that $\overset{F}{\cup}$ can be applied point-wise. Furthermore, for each $(s_1, s_2) \in S_1 \times S_2$, there is only a finite number of possible values for the bisimulation witness mapping. Hence, (R, g) is defined uniquely.

Proposition 2.12. $(\mathbf{Set}_{NDF'}/\simeq, \preccurlyeq, \otimes)$ *is a meet-semilattice, where* $\mathbf{Set}_{NDF'}$ *is the category of F-coalgebra with $F \in NDF'$.*

Proof. First of all, Lemma 2.10 implies the transitivity of $\preccurlyeq$. To complete the proof, we must show that, in the above context, (R, g) is, indeed, the meet of (S_1, f_1) and (S_2, f_2).

Let (S, f) be an F-coalgebra such that $(S, f) \preccurlyeq (S_i, f_i)$, for $i = 1, 2$. Then there exist two corresponding coalgebras (R_i, g_i) such that each $R_i \subseteq S \times S_i$ is a bisimulation on (S, f) and (S_i, f_i) total on S. Since R_1 and R_2 are total on S, and since R is maximal, R is total on the image $R_1(S) \subseteq S_1$.

Since both id_R and the projection $\pi : R \to S_1$ are coalgebra homomorphisms (the latter by definition of bisimulation), Lemma 2.8 implies that the image $(\pi, id_R)(R)$ is a bisimulation on (S_1, f_1) and (R, g). Moreover, it is total on $R_1(S)$. We conclude, by observing that, by Lemma 2.10, $R_1 \circ (\pi, id_R)$ is a bisimulation on (S, f) and (R, g), total on S, i.e. $(S, f) \preccurlyeq (R, g)$.[2] □

In the general case, the subject of coalgebra composition has been considered, for example, in [3, 17]. Here we provide, for a functor $F \in NDF'$, an example of the F-coalgebra composition operator along the lines of [17]. We define the maximal interaction operator $\|$ by putting, for any two F-coalgebras (S_1, f_1) and (S_2, f_2),

$$(S_1, f_1) \parallel (S_2, f_2) \stackrel{def}{=} (S_1 \times S_2, f_1 \parallel f_2), \tag{11}$$

with

$$\begin{array}{c} (f_1 \parallel f_2) : S_1 \times S_2 \to F(S_1 \times S_2) \\ (s_1, s_2) \mapsto f_1(s_1) \underset{S_1,S_2}{\overset{F}{\bowtie}} f_2(s_2) \end{array}, \tag{12}$$

where $\underset{X,Y}{\overset{F}{\bowtie}} : F(X) \times F(Y) \to F(X \times Y)$ is defined by structural induction on F. For any $x \in F(X)$ and $y \in F(Y)$, we put

$$x \underset{X,Y}{\overset{F}{\bowtie}} y \stackrel{def}{=} \begin{cases} (x, y), & \text{if } F = \mathbf{Id}, \\ x \vee y, & \text{if } F = \mathbf{B}, \\ \left(x_1 \underset{X,Y}{\overset{F_1}{\bowtie}} y_1,\ x_2 \underset{X,Y}{\overset{F_2}{\bowtie}} y_2\right), & \text{if } F = F_1 \times F_2,\ x = (x_1, x_2) \text{ and } y = (y_1, y_2), \\ \lambda a. \left(x(a) \underset{X,Y}{\overset{G}{\bowtie}} y(a)\right), & \text{if } F = G^{\mathbf{A}}. \\ \left\{ x' \underset{X,Y}{\overset{G}{\bowtie}} y' \,\middle|\, x' \in x, y' \in y \right\}, & \text{if } F = \mathscr{P}_\omega(G). \end{cases} \tag{13}$$

As it has been observed in [17], the composition operator defined by (11) and (12) is well-behaved, provided that $\overset{F}{\bowtie}$ is a natural transformation of functors from $F \times F$ to F. In particular, this would guarantee that this composition operator preserves coalgebra homomorphisms and, consequently the semantic preorder defined above.

Proposition 2.13. *$\overset{F}{\bowtie}$ is a natural transformation from $F \times F : \mathbf{Set}^2 \to \mathbf{Set}$ to $F : \mathbf{Set}^2 \to \mathbf{Set}$.*

Proof. We prove, by structural induction on F that, for any morphism $h = (h_1 \times h_2) : X \times Y \to X' \times Y'$, the diagram

$$\begin{array}{ccc} F(X) \times F(Y) & \xrightarrow{\underset{X,Y}{\overset{F}{\bowtie}}} & F(X \times Y) \\ {\scriptstyle (F\times F)(h)}\downarrow & & \downarrow{\scriptstyle F(h)} \\ F(X') \times F(Y') & \xrightarrow[\underset{X',Y'}{\overset{F}{\bowtie}}]{} & F(X' \times Y') \end{array}$$

[2] Observe that the maximality of g is only used to prove the uniqueness of (R, g). Thus, $(S, f) \preccurlyeq (R, g')$, for any witness mapping $g' : R \to F(R)$.

is commutative, i.e. that, for any $x \in F(X)$ and $y \in F(Y)$, holds

$$F(h_1 \times h_2)\left(x \underset{X,Y}{\overset{F}{\bowtie}} y\right) \quad = \quad F(h_1)(x) \underset{X',Y'}{\overset{F}{\bowtie}} F(h_2)(y)\,. \tag{14}$$

Case 1. ($F = \mathbf{Id}$)

$$\mathbf{Id}(h_1 \times h_2)\left(x \underset{X,Y}{\overset{\mathbf{Id}}{\bowtie}} y\right) = (h_1 \times h_2)(x,y) = (h_1(x), h_2(y)) = \mathbf{Id}(h_1)(x) \underset{X',Y'}{\overset{\mathbf{Id}}{\bowtie}} \mathbf{Id}(h_2)(y)\,.$$

Case 2. ($F = \mathbf{B}$)

$$\mathbf{B}(h_1 \times h_2)\left(x \underset{X,Y}{\overset{\mathbf{B}}{\bowtie}} y\right) = id_{\mathbf{B}}(x \vee y) = x \vee y = id_{\mathbf{B}}(x) \vee id_{\mathbf{B}}(y) = \mathbf{B}(h_1)(x) \underset{X',Y'}{\overset{\mathbf{B}}{\bowtie}} \mathbf{B}(h_2)(y)\,.$$

Case 3. ($F = F_1 \times F_2$) Let $x = (x_1, x_2)$ and $y = (y_1, y_2)$.

$$\begin{aligned}
&(F_1 \times F_2)(h_1 \times h_2)\left((x_1,x_2) \underset{X,Y}{\overset{F_1 \times F_2}{\bowtie}} (y_1,y_2)\right) = \\
&\quad = (F_1(h_1 \times h_2) \times F_2(h_1 \times h_2))\left(x_1 \underset{X,Y}{\overset{F_1}{\bowtie}} y_1,\ x_2 \underset{X,Y}{\overset{F_2}{\bowtie}} y_2\right) && \text{by (13) and the definition in Table 2} \\
&\quad = \left(F_1(h_1 \times h_2)\left(x_1 \underset{X,Y}{\overset{F_1}{\bowtie}} y_1\right),\ F_2(h_1 \times h_2)\left(x_2 \underset{X,Y}{\overset{F_2}{\bowtie}} y_2\right)\right) && \text{by the definition in Table 1} \\
&\quad = \left(F_1(h_1)(x_1) \underset{X',Y'}{\overset{F_1}{\bowtie}} F_1(h_2)(y_1),\ F_2(h_1)(x_2) \underset{X',Y'}{\overset{F_2}{\bowtie}} F_2(h_2)(y_2)\right) && \text{by the induction hypothesis} \\
&\quad = \left(F_1(h_1)(x_1), F_2(h_1)(x_2)\right) \underset{X',Y'}{\overset{F_1 \times F_2}{\bowtie}} \left(F_1(h_2)(y_1), F_2(h_2)(y_2)\right) && \text{by (13)} \\
&\quad = (F_1 \times F_2)(h_1)(x_1,x_2) \underset{X',Y'}{\overset{F_1 \times F_2}{\bowtie}} (F_1 \times F_2)(h_2)(y_1,y_2) && \text{by the definition in Table 1.}
\end{aligned}$$

Case 4. ($F = G^{\mathbf{A}}$)

$$\begin{aligned}
&G^{\mathbf{A}}(h_1 \times h_2)\left(x \underset{X,Y}{\overset{G^{\mathbf{A}}}{\bowtie}} y\right) = \\
&\quad = G(h_1 \times h_2) \circ \lambda a.\left(x(a) \underset{X,Y}{\overset{G}{\bowtie}} y(a)\right) && \text{by (13) and the definitions in Tables 1 and 2} \\
&\quad = \lambda a.\left(G(h_1 \times h_2)\left(x(a) \underset{X,Y}{\overset{G}{\bowtie}} y(a)\right)\right) && \text{by the definition of function composition} \\
&\quad = \lambda a.\left(G(h_1)(x(a)) \underset{X',Y'}{\overset{G}{\bowtie}} G(h_2)(y(a))\right) && \text{by the induction hypothesis} \\
&\quad = \left(\lambda a.G(h_1)(x(a))\right) \underset{X',Y'}{\overset{G^{\mathbf{A}}}{\bowtie}} \left(\lambda a.G(h_2)(y(a))\right) && \text{by (13)} \\
&\quad = G^{\mathbf{A}}(h_1)(x) \underset{X',Y'}{\overset{G^{\mathbf{A}}}{\bowtie}} G^{\mathbf{A}}(h_2)(y) && \text{by the definitions in Tables 1 and 2.}
\end{aligned}$$

Case 5. ($F = \mathscr{P}_\omega(G)$)

$$
\begin{aligned}
&\mathscr{P}_\omega(G)(h_1 \times h_2)\left(x \underset{X,Y}{\overset{\mathscr{P}_\omega(G)}{\bowtie}} y\right) = \\
&\quad = \left\{ G(h_1 \times h_2)\left(x' \underset{X,Y}{\overset{G}{\bowtie}} y'\right) \;\middle|\; \begin{array}{l} x' \in x \\ y' \in y \end{array} \right\} && \text{by (13) and the definitions in Tables 1 and 2} \\
&\quad = \left\{ G(h_1)(x') \underset{X',Y'}{\overset{G}{\bowtie}} G(h_2)(y') \;\middle|\; \begin{array}{l} x' \in x \\ y' \in y \end{array} \right\} && \text{by induction hypothesis} \\
&\quad = \left\{ x'' \underset{X',Y'}{\overset{G}{\bowtie}} y'' \;\middle|\; \begin{array}{l} x'' \in \{G(h_1)(x') \,|\, x' \in x\} \\ y'' \in \{G(h_2)(y') \,|\, y' \in y\} \end{array} \right\} \\
&\quad = \left\{ x'' \underset{X',Y'}{\overset{G}{\bowtie}} y'' \;\middle|\; \begin{array}{l} x'' \in \mathscr{P}_\omega(G(h_1))(x) \\ y'' \in \mathscr{P}_\omega(G(h_2))(y) \end{array} \right\} && \text{by the definition in Table 1} \\
&\quad = \mathscr{P}_\omega(G)(h_1)(x) \underset{X',Y'}{\overset{\mathscr{P}_\omega(G)}{\bowtie}} \mathscr{P}_\omega(G)(h_2)(y) && \text{by (13) and the definition in Table 2.}
\end{aligned}
$$

□

Finally, we put $\mathbf{0}_F = (1, f^0_F)$, with $f^0_F : 1 \to F(1)$ defined, once again, by structural induction on F:

$$
f^0_F(*) \quad \stackrel{def}{=} \quad \begin{cases} *, & \text{if } F = \mathbf{Id}, \\ \bot, & \text{if } F = \mathbf{B}, \\ (f^0_{F_1}(*), f^0_{F_2}(*)), & \text{if } F = F_1 \times F_2, \\ \lambda a. f^0_G(*), & \text{if } F = G^{\mathbf{A}}, \\ \{f^0_G(*)\}, & \text{if } F = \mathscr{P}_\omega(G), \end{cases}
$$

where $\bot \in \mathbf{B}$ is the bottom of $\mathbf{B}$.

Clearly, taken together, the elements defined in this section form a behaviour type over the family of F-coalgebras.

3 Behaviour composition

3.1 Composition operators

Assume that a behaviour type $(\mathscr{B}, \|, \sqsubseteq, \preccurlyeq, \otimes, \mathbf{0})$ is given.

Definition 3.1 (Composition operator)**.** An n-ary operator $f : \mathscr{B}^n \to \mathscr{B}$ is a *composition operator* iff it satisfies the following properties, for any $B_1, \ldots, B_n, \tilde{B} \in \mathscr{B}$:

1. $f(B_1, \ldots, B_n) \sqsubseteq B_1 \,\|\, \cdots \,\|\, B_n$,
2. For any $i \in [1,n]$, $B_i \preccurlyeq \tilde{B}$ implies $f(B_1, \ldots, B_i, \ldots, B_n) \preccurlyeq f(B_1, \ldots, \tilde{B}, \ldots, B_n)$.

We denote by $\mathscr{C}$ the set of all composition operators. $\mathscr{C} = \bigcup_{n \geq 1} \mathscr{C}^{(n)}$, where $\mathscr{C}^{(n)}$ is the set of all n-ary composition operators.

Among the immediate consequences of the above definition, one should notice the following facts.

Lemma 3.2. *1. The equivalence relation $\simeq \,= \preccurlyeq \cap \preccurlyeq^{-1}$ is a congruence with respect to composition operators;*

2. *The maximal interaction operator $\|$ is a composition operator;*
3. *For any $B_1, B_2 \in \mathscr{B}$, one has $B_1 \sqsubseteq B_1 \parallel B_2$.*

Proof. The first two statements of the lemma are trivial. The third one is proven by observing that $\mathbf{0} \sqsubseteq B_2$ and, consequently, $B_1 \simeq B_1 \parallel \mathbf{0} \sqsubseteq B_1 \parallel B_2$. □

Definition 3.3 (Composition of operators)**.** For an n-ary operator $f_1 : \mathscr{B}^n \to \mathscr{B}$, an m-ary operator $f_2 : \mathscr{B}^m \to \mathscr{B}$, and $i \in [1,n]$, the $(n+m-1)$-ary operator $f_1 \circ_i f_2$ is defined by

$$(f_1 \circ_i f_2)\Big(B_1, \ldots, B_{n+m-1}\Big) \stackrel{def}{=} f_1\Big(B_1, \ldots, B_{i-1}, f_2(B_n, \ldots, B_{n+m-1}), B_i, \ldots, B_{n-1}\Big). \tag{15}$$

Lemma 3.4. *A composition of two composition operators is also a composition operator.*

Example 3.5 (BIP interaction model)**.** A convenient way of defining composition operators is through the use of SOS rules. For example, consider the behaviours specified by LTS (see Section 2.2). In the BIP interaction model [6], given $\gamma \subseteq 2^{\mathbf{A}}$, the corresponding n-ary composition operator is defined on behaviours $B_i = (Q_i, A_i, \to_i)$, for $i \in [1,n]$, by putting $\gamma(B_1, \ldots, B_n) \stackrel{def}{=} (\prod_{i=1}^n Q_i, \bigcup_{i=1}^n A_i, \to)$, where $\to$ is the minimal transition relation satisfying the following set of SOS rules

$$\left\{ \left. \frac{\left\{ q_i \xrightarrow{a_i}_i q_i' \right\}_{i \in I} \quad \left\{ q_i = q_i' \right\}_{i \notin I} \quad \bigcup_{i \in I} a_i = a}{q_1 \ldots q_n \xrightarrow{a} q_1' \ldots q_n'} \right| a \in \gamma \right\}. \tag{16}$$

Proposition 3.6. *Any operator f defined as above is a composition operator on LTS.*

Proof. We have to show that f preserves the preorder $\preccurlyeq$. By symmetry, it is sufficient to prove that, for $B_1 \preccurlyeq \tilde{B}_1$ and $B_2, \ldots, B_n$, we have $f(B_1, B_2, \ldots, B_n) \preccurlyeq f(\tilde{B}_1, B_2, \ldots, B_n)$. First of all, since $A_1 \subseteq \tilde{A}_1$, we have $\bigcup_{i=1}^n A_i \subseteq \tilde{A}_1 \cup \bigcup_{i=2}^n A_i$.

Since $B_1 \preccurlyeq \tilde{B}_1$, there exists a relation $\mathscr{R}_1 \subseteq Q_1 \times \tilde{Q}_1$ total on Q_1 and satisfying (5). We can then define a relation $\mathscr{R} \subseteq (Q_1 \times \prod_{i=2}^n Q_i) \times (\tilde{Q}_1 \times \prod_{i=2}^n Q_i)$, by putting $(q_1, q_2, \ldots, q_n) \mathscr{R} (\tilde{q}_1, q_2', \ldots, q_n')$ iff $q_1 \mathscr{R}_1 \tilde{q}_1$ and $q_i = q_i'$, for $i \in [2,n]$. This relation is clearly total. Let $q_1 q_2 \ldots q_n \xrightarrow{a} q_1' \ldots q_n'$ be a transition in $f(B_1, B_2, \ldots, B_n)$ and let $q_1 \mathscr{R}_1 \tilde{q}_1$. By definition of the operator f, this transition must be inferred by the rule (16) from a set of transitions $\{q_i \xrightarrow{a_i}_i q_i'\}_{i \in I}$ with $a = \bigcup_{i \in I} a_i$. If $1 \notin I$, then clearly $q_1 = q_1'$ and the corresponding transition $\tilde{q}_1 q_2 \ldots q_n \xrightarrow{a} \tilde{q}_1 q_2' \ldots q_n'$ is possible in $f(\tilde{B}_1, B_2, \ldots, B_n)$. If $1 \in I$, by (5), there exists a transition $\tilde{q}_1 \xrightarrow{b_1} \tilde{q}_1{}'$ such that $a_1 \subseteq b_1$ and $q_1' \mathscr{R}_1 \tilde{q}_1{}'$. Hence, $a = a_1 \cup \bigcup_{i \in I \setminus \{1\}} a_i \subseteq b \cup \bigcup_{i \in I \setminus \{1\}} a_i \stackrel{def}{=} b$ and, by (16), $\tilde{q}_1 q_2 \ldots q_n \xrightarrow{b} \tilde{q}_1{}' q_2 \ldots q_n$, which proves the proposition. □

Example 3.7 (Negative premises)**.** Consider the family of SOS operators with negative premises, that is defined by the rules of the form

$$\frac{\left\{ q_i \xrightarrow{a_i}_i q_i' \,\middle|\, i \in I \right\} \quad \left\{ q_i = q_i' \,\middle|\, i \notin I \right\} \quad \left\{ q_j \stackrel{b_j^k}{\not\longrightarrow}_j \,\middle|\, j \in J, k \in K_j \right\} \quad a = \bigcup_{i \in I} a_i}{q_1 \ldots q_n \xrightarrow{a} q_1' \ldots q_n'}, \tag{17}$$

where $q_j \stackrel{b_j^k}{\not\longrightarrow}_j$ means that there is no transition labelled b_j^k possible from the state q_j of behaviour B_j.

It is easy to see that such operators do not preserve simulation relation $\preccurlyeq$ and therefore are not composition operators on the LTS behaviour type as defined in Section 2.2. An adaptation of the ready simulation relation from the initial proposition by Bloom [10] is necessary to obtain a behaviour type, for which such operators would, indeed, be composition operators.

Let us now revisit the *separation of concerns* principle mentioned in the introduction. In our context, this principle consists in separating the computation of a system from the application of *glue operators* coordinating its atomic components. Intuitively, this means that no additional behaviour should be introduced by application of a glue operator. Not only glue operators are limited to restricting the behaviour of atomic components by imposing coordination constraints (cf. condition 1 of Definition 3.1), but, on top of that, they should be *memoryless*. Intuitively, this corresponds to requiring that two conditions be satisfied: 1) the glue operator should not add *state* to the coordinated system and 2) the actions possible in a global state of the composed system are completely determined by the properties of the corresponding states of the constituent subsystems. The latter condition is justified by the fact that a memoryless operator is unaware of any states of the system other than the current one. In order to impose these additional requirements, we need a formal notion of state, provided precisely by coalgebras.

Definition 3.8. Let F be a **Set**-functor. An n-ary composition operator gl on a behaviour type over the family of F-coalgebras is a *glue operator* iff there exists a natural transformation $sync : F^n \to F$, such that, for any set $\{B_i = (S_i, f_i) \,|\, i \in [1,n]\}$ of F-coalgebras, $gl(B_1, \ldots, B_n) = (S, f)$ with $S = \prod_{i=1}^{n} S_i$ and $f(s) = sync(f_1(s_1), \ldots, f_n(s_n))$, for all $s = (s_1, \ldots, s_n) \in S$.

We now consider two classical operators, namely the *prefixing operator* and *choice*. The former is a unary operator, which consists in executing a given action before "running" the behaviour to which the prefixing is applied. The latter is an associative and commutative binary operator, which consists in running exactly one of the behaviours to which it is applied. In order to implement either of these two operators, one has to "remember" that, respectively, the initial transition or the choice of the two components has been made, thus adding state to the composed system.

For behaviour types based on coalgebras, the binary choice operator $+$ can be formally defined as the coproduct functor on the category of coalgebras, i.e. $(S_1, f_1) + (S_2, f_2) \stackrel{def}{=} (S_1 + S_2, f_1 + f_2)$.

Hypothesis 3.9. *Let F be a* **Set***-functor weakly preserving pullbacks. In a behaviour type over F-coalgebras with the semantic preorder defined as in Section 2.3 there is no glue operator gl, in the sense of Definition 3.8, such that $gl \simeq +$.*

To formulate a similar hypothesis for the prefixing operator, a notion of *initial state* is necessary. Although such a notion is provided by *pointed coalgebras*, we omit this discussion in this paper.

Notice also the distinction between choice and *interleaving*. The former consists in choosing once and for all the behaviour to run, whereas the latter makes this choice independently at each execution step.

3.2 Combining composition operators

Although composition of operators introduced in Definition 3.3 allows to combine several operators hierarchically, it does not allow "simultaneous" application of the constraints imposed by two operators.

Example 3.10 (Simultaneous application of two operators)**.** Indeed, consider, for example, for $i \in [1,4]$, $B_i = (Q_i, A_i, \to) \in LTS$ (see Section 2.2) and four given actions a_i, such that $a_i \in A_i$ and $a_i \notin A_j$ for $i \neq j$. Consider also two quaternary composition operators $f_1, f_2 \in \mathscr{C}^{(4)}$, such that the action of f_1 consists in synchronising the actions a_1 and a_2 (cf. Example 3.5) of the first two components it is applied

to, whereas f_2 synchronises the actions a_3 and a_4 of the last two of its arguments. More precisely, $f_1(B_1,B_2,B_3,B_4) = \gamma_{a_1a_2}(B_1,B_2) \parallel B_3 \parallel B_4$ and $f_2(B_1,B_2,B_3,B_4) = B_1 \parallel B_2 \parallel \gamma_{a_3a_4}(B_3,B_4)$, where the binary operator γ_a, parametrised by an interaction a (here, $a = a_1a_2$, for f_1, and $a = a_3a_4$, for f_2) is defined by the following rules, x and y being action variables,

$$\frac{q_1 \xrightarrow{x} q_1' \qquad x \notin a}{q_1q_2 \xrightarrow{x} q_1'q_2}, \quad \frac{q_2 \xrightarrow{x} q_2' \qquad x \notin a}{q_1q_2 \xrightarrow{x} q_1q_2'}, \quad \frac{q_1 \xrightarrow{x} q_1' \qquad q_2 \xrightarrow{y} q_2' \qquad x,y \notin a}{q_1q_2 \xrightarrow{x \cup y} q_1'q_2'}, \tag{18}$$

$$\frac{\left\{ q_i \xrightarrow{a \cap A_i} q_i' \,\middle|\, a \cap A_i \neq \emptyset \right\} \qquad \left\{ q_i = q_i' \,\middle|\, a \cap A_i = \emptyset \right\}}{q_1q_2 \xrightarrow{a} q_1'q_2'}. \tag{19}$$

Intuitively, simultaneous application of f_1 and f_2 to any components B_1, B_2, B_3, B_4 should enforce, on one hand, the synchronisation of a_1 and a_2 in B_1 and B_2 respectively, and, on the other hand, the synchronisation of respectively a_3 and a_4 in B_3 and B_4. However, the result $f_1(B_1,B_2,B_3,B_4)$ of applying f_1 is a single component, to which f_2 cannot be applied any more, since the latter is a quaternary operator.

Furthermore, considering the specification of the γ_a operator above as *"enforcing the synchronisation of actions belonging to the interaction a"*, one would expect the simultaneous application of $\gamma_{a_1a_2}$ and $\gamma_{a_3a_4}$ to any set of behaviours to achieve the same effect as above, without having to explicitly define operators f_1 and f_2 and avoiding the associated problem discussed in the previous paragraph.

First of all, the semantic preorder $\preccurlyeq$ and the operator $\otimes$ can be canonically extended to composition operators, provided they have the same arity: for any $n \geq 1$ and $gl_1, gl_2 \in \mathscr{C}^{(n)}$,

$$f_1 \preccurlyeq f_2 \overset{def}{\iff} \forall B_1,\ldots,B_n \in \mathscr{B},\ \Big(f_1(B_1,\ldots,B_n) \preccurlyeq f_2(B_1,\ldots,B_n)\Big), \tag{20}$$

$$\forall B_1,\ldots,B_n \in \mathscr{B},\ (f_1 \otimes f_2)(B_1,\ldots,B_n) \overset{def}{=} f_1(B_1,\ldots,B_n) \otimes f_2(B_1,\ldots,B_n). \tag{21}$$

Proposition 3.11. *$(\mathscr{C}^{(n)}/\simeq, \preccurlyeq, \otimes)$ is a meet-semilattice.*

Proof. Let $f_1, f_2 \in \mathscr{C}^{(n)}$ be two composition operators and consider $f \in \mathscr{C}^{(n)}$ such that $f_1 \otimes f_2 \preccurlyeq f \preccurlyeq f_1, f_2$. The right-hand relation implies that, for any $B \in \mathscr{B}$, $f(B) \preccurlyeq f_1(B) \otimes f_2(B) = (f_1 \otimes f_2)(B)$. Hence $f \preccurlyeq f_1 \otimes f_2$ and, together with the left-hand relation, this implies $f \simeq f_1 \otimes f_2$. □

Example 3.12 (Preorder on SOS operators)**.** Consider once again the *LTS* behaviour type defined in Section 2.2 and the corresponding family of composition operators defined by (16) in Example 3.5. Identifying each operator with the set of its defining SOS rules, it follows directly from [7, Lemma 3] that, for two such operators f_1 and f_2, $f_1 \preccurlyeq f_2$ is equivalent to $f_1 \subseteq f_2$. Hence, $f_1 \otimes f_2 = f_1 \cap f_2$.

Going back to Example 3.10, $(f_1 \otimes f_2)(B_1,B_2,B_3,B_4)$ represents the behaviour where both the actions of B_1 are synchronised with those of B_2, and the actions of B_3 are synchronised with those of B_4, as defined respectively by f_1 and f_2.

In order to allow application of a given composition operator to any set of component behaviours with cardinality at least the arity of the composition operator in question, we introduce below the *arity extension* for composition operators.

Definition 3.13. The *arity extension* of an n-ary composition operator $f \in \mathscr{C}^{(n)}$ to arity $m \geq n$ is the composition operator $f^{(m)} \in \mathscr{C}^{(m)}$ defined by putting, for all $B_1,\ldots,B_n \in \mathscr{B}$,

$$f^{(m)}(B_1,\ldots,B_n) \overset{def}{=} \bigotimes_{\sigma \in S_m} \Big(f\left(B_{\sigma(1)},\ldots,B_{\sigma(n)}\right) \parallel B_{\sigma(n+1)} \parallel \cdots \parallel B_{\sigma(m)}\Big), \tag{22}$$

where S_m is the group of all permutations of $[1,m]$. The right-hand side of (22) consists in simultaneously applying f to all possible subsets of n components.

For operators f_1 and f_2 of Example 3.10, we have $f_1 = (\gamma_{a_1a_2})^{(4)}$ and $f_2 = (\gamma_{a_3a_4})^{(4)}$.

Lemma 3.14 (Isotony of arity extension)**.** *Arity extension preserves the semantic preorder, that is, for any $f_1, f_2 \in G^{(n)}$ and $m \geq n$, $f_1 \preccurlyeq f_2$ implies ${f_1}^{(m)} \preccurlyeq {f_2}^{(m)}$.*

Example 3.15 (Disjoint behaviours)**.** Going back to Example 3.10 of Section 3.2 and considering that, for $i \neq j$, $a_i \notin A_j$, one can see that

$$f_1 \otimes f_2 \simeq {\gamma_{a_1a_2}}^{(4)} \otimes {\gamma_{a_3a_4}}^{(4)} \simeq (\gamma_{a_1a_2} \otimes \gamma_{a_3a_4})^{(4)}\,.$$

The assumption that, for $i \neq j$, $a_i \notin A_j$ is essential here. It guarantees that the operators $\gamma_{a_1a_2}$ and $\gamma_{a_3a_4}$ do not *interfere* with the behaviour of components B_3, B_4 and B_1, B_2 respectively. Precise characterisation of such a notion of *non-interference* will be an important part of our future work.

3.3 Symmetrical composition operators

Notice that the operator $f^{(m)}$ defined by (22) is symmetrical in the sense of the following definition.

Definition 3.16. An operator $f : X^n \to Y$ is called *symmetrical* iff, for any permutation $\sigma \in S_n$ and any $x_1, \ldots, x_n \in X$, holds the equation $f(x_1, \ldots, x_n) = f(x_{\sigma(1)}, \ldots, x_{\sigma(n)})$.

Clearly, an operator $f \in \mathscr{C}^{(n)}$ is symmetrical iff $f \simeq f^{(n)}$. Furthermore, for symmetrical operators, one can also unambiguously define the arity reduction.

Definition 3.17. The *arity reduction* of a symmetrical n-ary composition operator $f \in \mathscr{C}^{(n)}$ to arity $m \leq n$ is the composition operator $f^{(m)} \in \mathscr{C}^{(m)}$ defined by putting, for all $B_1, \ldots, B_m \in \mathscr{B}$,

$$f^{(m)}(B_1, \ldots, B_n) \stackrel{def}{=} f(B_1, \ldots, B_n, \mathbf{0}, \ldots, \mathbf{0})\,. \tag{23}$$

Given a symmetrical operator $f \in \mathscr{C}$, one can combine both concepts—arity extension and reduction—to define the operator $\tilde{f} : \mathscr{P}_\omega(\mathscr{B}) \to \mathscr{B}$, by putting, for any finite subset $\mathbf{B} \subseteq \mathscr{B}$, $\tilde{f}(\mathbf{B}) \stackrel{def}{=} f^{(|\mathbf{B}|)}(\mathbf{B})$.

Observe that several popular parallel composition operators, such as the ones used in CCS [23], CSP [18] and BIP [6], are symmetrical.

4 Discussion and related work

The complete bibliography, as related to the motivations behind the present paper, is yet to be established. However, several contributions can already be mentioned.

Based on the same observations about the importance of studying glue as a first-class notion, Abstract Behavior Types (ABM) were proposed in [2] guiding the design of the Reo language. ABT propose to characterise components as channels (or dataflow transformers) that do not provide any information as to the manner in which the defining transformations are computed. This approach is radically different from the one taken in the present paper, since the main emphasis is put on one aspect of component behaviour (namely the dataflow transformation) and its expressive power with regards to assembling more complex transfer functions. In particular composition of channels boils down to pipeline assembly. Although, ABT are likely to provide another interesting case study for the behaviour types we propose, they lack the abstraction necessary to model a larger class of behaviour as intended here.

It would be interesting to further verify the robustness of the proposed framework by defining behaviour types based on interface [1] or modal [19] automata, whereof the particularity is that the corresponding standard refinement relations are contravariant as opposed to the examples of this paper.

Furthermore, although all the examples provided in this paper can be interpreted as state-based behaviour types and, therefore, modelled as coalgebras of the appropriate types, the intention is to keep the abstraction level sufficiently high in order to be able to accommodate for behaviour types that do not have a clearly identifiable notion of state. In particular, it would be interesting to investigate applicability of this framework to continuous time systems as modelled in Simulink[3] or Modelica[4].

As mentioned above, most—if not all—behaviour types, of interest for the author of this paper, can be modelled as coalgebras of a suitable type. Since their introduction as a model for system behaviour, coalgebras have been subject to extensive studies. It seems, however, that most of these studies were focusing primarily on coalgebras as a way to model individual component behaviour. Although, in [3, 17], coalgebra composition has been addressed in a much more general form than in the present paper, some questions remain unanswered: *What is the class of coalgebra types, for which a meaningful maximal interaction operator can be constructively defined? How does one characterise the composition operators other than maximal interaction?* The latter question is related to the work presented in [16, 31], where GSOS-style operational semantics is studied from the categorical point of view.

Last but not least, an important subject that we have mentioned in this paper and that we are planning to address as part of our future work is the interference between glue operators. Consider two glue operators gl_1 and gl_2, and a family of behaviours $\{B_i\}_{i=1}^n$. Assume, furthermore, that, for $i = 1,2$, ${gl_i}^{(n)}(B_1,\ldots,B_n)$ satisfies some given property P_i. *What conditions have to be satisfied by gl_1 and gl_2, on one hand, and $\{B_i\}_{i=1}^n$, on the other hand, for any of the composite behaviours ${gl_i}^{(1)}({gl_j}^{(n)}(B_1,\ldots,B_n))$ $(i \neq j)$, $(gl_1 \otimes gl_2)^{(n)}(B_1,\ldots,B_n)$, etc. to satisfy $P_1 \wedge P_2$? Given computable representations of gl_1 and gl_2 how does one compute the operator imposing $P_1 \wedge P_2$?* The latter question was partially addressed in Example 3.12 of Section 3.2. Furthermore, our result in [7] suggests that this example can be generalised, on a suitable behaviour type, to SOS operators with negative premises. *Can a similar result be obtained for a larger class of coalgebras types?*

A first, naïve approach consists in defining a class of *distributive* behaviour types characterised by the distributivity of their maximal interaction operator over the meet operator, that is, for any $B_1, B_2, B_3 \in \mathscr{B}$,

$$(B_1 \otimes B_2) \parallel B_3 \quad \simeq \quad (B_1 \parallel B_3) \otimes (B_2 \parallel B_3)\,.$$

It is easy to see that distributivity of a behaviour type implies that of the arity extension of composition operators over the meet operator: for any $f_1, f_2 \in \mathscr{C}^{(n)}$ and $m \geq n$,

$$(f_1 \otimes f_2)^{(m)} \simeq {f_1}^{(m)} \otimes {f_2}^{(m)}\,.$$

Assuming that we know how to compute $f_1 \otimes f_2$, we can then reasonably expect the obtained operator to satisfy both properties imposed by f_1 and f_2.

Although none of behaviour types considered in this paper appear to be distributive, all the counter-examples we have considered while preparing the paper were based on the fact that several components shared certain actions. As illustrated by Example 3.15 of Section 3.2, "local distributivity" can be achieved provided the "absence of conflicts" in the component interfaces. An important question to be addressed in the future work is: *How can these notions of "local distributivity" and "absence of conflicts" be formalised and generalised to larger classes of behaviour types?*

[3] http://www.mathworks.com/products/simulink/

[4] http://www.modelica.org/

5 Conclusion

The goal of this paper was to make a first step towards the definition of a formal framework for studying behaviour composition in a setting sufficiently general to provide insight into how the component-based systems should be modelled and compared.

We have proposed the notions of *behaviour type* and *composition operator*, which, while striving for generality, allow to capture some essential properties expected when reasoning intuitively about component composition. We have illustrated the notion of behaviour type on three examples, namely Traces, Labelled Transition Systems and F-coalgebras with a restricted class of non-deterministic functors.

In the framework proposed in this paper, a *behaviour type* is a tuple $(\mathscr{B}, \|, \sqsubseteq, \preccurlyeq, \otimes, \mathbf{0})$, where $\mathscr{B}$ is the set of underlying behaviours (Traces, Labelled Transition Systems, Coalgebras, etc.); $\|$ is the maximal interaction operator defining the joint behaviour of two components without any coordination constraints; $\sqsubseteq$ and $\preccurlyeq$ are two preorders used respectively to represent "containment", or simulation, relation between two behaviours and the semantic relation reflecting the fact that one component can be substituted by another one while essentially preserving the intended behaviour. The relation $\simeq = \preccurlyeq \cap \preccurlyeq^{-1}$ is a congruence for composition operators. The example provided in Section 2.3 supports our idea that these two preorders need not necessarily be the same. We require that $(\mathscr{B}/\simeq, \preccurlyeq, \otimes)$ be a meet-semilattice, whereby the meet operator $\otimes$ identifies the common behaviour of two components. Finally, the notion of zero behaviour $\mathbf{0}$ serves as a "sanity check" for the behaviour type. Intuitively, it represents a component that does nothing. On one hand, it should not influence the behaviour of other components when placed in parallel (for any $B \in \mathscr{B}$, $B \,\|\, \mathbf{0} \simeq B$) and, on the other hand, it should be simulated by all other behaviours (for any $B \in \mathscr{B}$, $0 \sqsubseteq B$).

The requirement that the semantic preorder of a given behaviour type induce a meet-semilattice structure has allowed us to define a meet of composition operators, representing their simultaneous application to a given set of behaviours and, consequently, to extend any composition operator to a symmetrical one, applicable to any finite set of behaviours. This, together with the definition of *memoryless glue operators*, takes us one step closer to a formal understanding of the separation of concerns principle that we have advocated in our previous papers, and which stipulates that the computational aspects of the system should be localised in the atomic components, whereas the coordination layer responsible for managing concurrency should be realised by memoryless glue operators.

Finally, we have discussed some related work and key questions, arising from the proposed framework, that are important for the understanding of fundamental principles of component-based design.

Acknowledgements

I would like to express my gratitude to the anonymous reviewers for the instructive discussion on the ICE 2012 forum. I would also like to thank Ana Sokolova and Alexandra Silva for the discussion after the workshop and the pointers to the literature allowing me to include the powerset functor in Section 2.3.

References

[1] Luca de Alfaro & Thomas A. Henzinger (2001): *Interface automata*. *SIGSOFT Softw. Eng. Notes* 26(5), pp. 109–120, doi:10.1145/503271.503226.

[2] Farhad Arbab (2005): *Abstract Behavior Types: A foundation model for components and their composition*. *Sci. Comput. Program.* 55(1–3), pp. 3–52, doi:10.1016/j.scico.2004.05.010.

[3] Luís Soares Barbosa (2000): *Components as Processes: An Exercise in Coalgebraic Modeling*. In Scott F. Smith & Carolyn L. Talcott, editors: *FMOODS*, *IFIP Conference Proceedings* 177, Kluwer, pp. 397–418.

[4] Ananda Basu, Marius Bozga & Joseph Sifakis (2006): *Modeling Heterogeneous Real-time Components in BIP*. In: 4^{th} *IEEE Int. Conf. on Software Engineering and Formal Methods (SEFM06)*, pp. 3–12, doi:10.1109/SEFM.2006.27. Invited talk.

[5] Saddek Bensalem, Andreas Griesmayer, Axel Legay, Thanh-Hung Nguyen, Joseph Sifakis & Rongjie Yan (2011): *D-Finder 2: towards efficient correctness of incremental design*. In: *Proceedings of the* 3^{rd} *international conference on NASA Formal methods*, NFM'11, Springer-Verlag, Berlin, Heidelberg, pp. 453–458, doi:10.1007/978-3-642-20398-5_32. Available at `http://dl.acm.org/citation.cfm?id=1986308.1986344`.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] Bard Bloom (1989): *Ready Simulation, Bisimulation, and the Semantics of CCS-Like Languages*. Ph.D. thesis, Massachusetts Institute of Technology.

[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] 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.

[13] Paul C. Clements (1995): *From Subroutines to Subsystems: Component-Based Software Development*. *The American Programmer* 8(11).

[14] Geoff Coulson, Gordon Blair, Paul Grace, Francois Taiani, Ackbar Joolia, Kevin Lee, Jo Ueyama & Thirunavukkarasu Sivaharan (2008): *A generic component model for building systems software*. *ACM Trans. Comput. Syst.* 26(1), pp. 1:1–1:42, doi:10.1145/1328671.1328672.

[15] Cinzia Di Giusto & Jean-Bernard Stefani (2011): *Revisiting Glue Expressiveness in Component-Based Systems*. In Wolfgang De Meuter & Gruia-Catalin Roman, editors: *COORDINATION*, *Lecture Notes in Computer Science* 6721, Springer, pp. 16–30, doi:10.1007/978-3-642-21464-6_2.

[16] Ichiro Hasuo (2011): *The Microcosm Principle and Compositionality of GSOS-Based Component Calculi*. In Andrea Corradini, Bartek Klin & Corina Cîrstea, editors: *CALCO*, *Lecture Notes in Computer Science* 6859, Springer, pp. 222–236, doi:10.1007/978-3-642-22944-2_16.

[17] Ichiro Hasuo, Bart Jacobs & Ana Sokolova (2008): *The Microcosm Principle and Concurrency in Coalgebra*. In Roberto M. Amadio, editor: *FoSSaCS*, *LNCS* 4962, Springer, pp. 246–260, doi:10.1007/978-3-540-78499-9_18.

[18] C. A. R. Hoare (1985): *Communicating Sequential Processes*. Prentice Hall International Series in Computer Science, Prentice Hall.

[19] Kim Larsen, Ulrik Nyman & Andrzej Wąsowski (2007): *Modal I/O Automata for Interface and Product Line Theories*. In Rocco De Nicola, editor: *Programming Languages and Systems*, *Lecture Notes in Computer Science* 4421, Springer Berlin / Heidelberg, pp. 64–79, doi:10.1007/978-3-540-71316-6_6.

[20] Yoad Lustig & Moshe Y. Vardi (2009): *Synthesis from Component Libraries*. In: *FOSSACS'09: Proceedings of the* 12^{th} *International Conference on Foundations of Software Science and Computational Structures*, Springer-Verlag, Berlin, Heidelberg, pp. 395–409, doi:10.1007/978-3-642-00596-1_28.

[21] Philip K. McKinley, Seyed Masoud Sadjadi, Eric P. Kasten & Betty H.C. Cheng (2004): *Composing adaptive software*. *Computer* 37(7), pp. 56–64, doi:10.1109/MC.2004.48.

[22] Robin Milner (1983): *Calculi for synchrony and asynchrony*. *Theoretical Computer Science* 25(3), pp. 267–310, doi:10.1016/0304-3975(83)90114-7.

[23] Robin Milner (1989): *Communication and Concurrency*. Prentice Hall International Series in Computer Science, Prentice Hall.

[24] Gordon D. Plotkin (1981): *A Structural Approach to Operational Semantics*. Technical Report DAIMI FN-19, University of Aarhus. Available at `http://citeseer.ist.psu.edu/plotkin81structural.html`.

[25] Amir Pnueli & Roni Rosner (1990): *Distributed reactive systems are hard to synthesize*. *Annual IEEE Symposium on Foundations of Computer Science* 2, pp. 746–757, doi:10.1109/FSCS.1990.89597.

[26] Awais Rashid, Peter Sawyer, Ana Moreira & João Araújo (2002): *Early aspects: a model for aspect-oriented requirements engineering*. In: *Proceedings of the IEEE Joint International Conference on Requirements Engineering (RE'02)*., pp. 199–202, doi:10.1109/ICRE.2002.1048526.

[27] Jan J. M. M. Rutten (2000): *Universal coalgebra: a theory of systems*. *Theor. Comput. Sci.* 249(1), pp. 3–80, doi:10.1016/S0304-3975(00)00056-6.

[28] J. Sifakis (2005): *A Framework for Component-based Construction*. In: *Proceedings of the Third International Conference on Software Engineering and Formal Methods (SEFM)*, IEEE Computer Society, pp. 293–300.

[29] Alexandra Silva (2010): *Kleene Coalgebra*. Ph.D. thesis, CWI, Amsterdam, The Netherlands.

[30] Ana Sokolova (2005): *Coalgebraic Analysis of Probabilistic Systems*. Ph.D. thesis, TU Eindhoven, Eindhoven, The Netherlands.

[31] Daniele Turi & Gordon D. Plotkin (1997): *Towards a Mathematical Operational Semantics*. In: *LICS*, IEEE Computer Society, pp. 280–291, doi:10.1109/LICS.1997.614955.

[32] Peter Wegner (1996): *Coordination as constrained interaction (extended abstract)*. In: *Proc. of the First International Conference on Coordination Languages and Models*, *LNCS* 1061, Springer-Verlag, Springer Berlin / Heidelberg, pp. 28–33, doi:10.1007/3-540-61052-9.