A Notion of Glue Expressiveness for Component-Based Systems

Simon Bliudze[1] and Joseph Sifakis[1]

VERIMAG, Centre Équation, 2 av de Vignate, 38610, Gières, France
{bliudze, sifakis}@imag.fr

Abstract. Comparison between different formalisms and models is often by flattening structure and reducing them to behaviorally equivalent models e.g., automaton and Turing machine. This leads to a notion of expressiveness which is not adequate for component-based systems where separation between behavior and coordination mechanisms is essential. The paper proposes a notion of glue expressiveness for component-based frameworks characterizing their ability to coordinate components.
Glue is a closed under composition set of operators mapping tuples of behavior into behavior. Glue operators preserve behavioral equivalence. They only restrict the behavior of their arguments by performing memoryless coordination.
Behavioral equivalence induces an equivalence on glue operators. We compare expressiveness of two glues G_1 and G_2 by considering whether glue operators of G_1 have equivalent ones in G_2 (strong expressiveness). Weak expressiveness is defined by allowing a finite number of additional behaviors in the arguments of operators of G_2.
We propose an SOS-style definition of glues, where operators are characterized as sets of SOS-rules specifying the transition relation of composite components from the transition relations of their constituents. We provide expressiveness results for the glues of BIP and of process algebras such as CCS, CSP and SCCS. We show that for the considered expressiveness criteria, glues of the considered process calculi are less expressive than general SOS glue. Furthermore, glue of BIP has exactly the same strong expressiveness as glue definable by the SOS characterization.

1 Introduction

A central idea in systems engineering is that 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 behavior. We lack adequate notions of expressiveness to compare the merits and weaknesses of these frameworks.

Usually, comparison between formalisms and models is by flattening structure and reduction to a behaviorally equivalent model e.g., automaton and Turing machine. In this manner, all finite state formalisms turn out to be expressively equivalent independently of the features used for composition of behaviors. Many models and languages are Turing-expressive, while their coordination capabilities are tremendously different. This fact motivated work on the expressive power of programming languages. Felleisen [1] has provided a framework formally capturing meanings of expressiveness for sequential programming languages and taking into account not only the semantics but also the primitives of languages. Although the general framework is interesting, for component-based systems we need specific results focusing on composition and allowing comparison of different composition operators.

This paper proposes a notion of glue expressiveness for component-based systems. It builds on results and concepts presented in [2] that guided the design of the BIP (Behavior, Interaction, Priority) framework [3]. BIP allows the construction of complex components from atomic ones represented as labeled transition systems, by using two classes of operators: 1) Interaction operators which compose the behavior of their arguments by using interactions (strongly synchronized transitions); 2) Priority operators which are unary operators used to restrict non-determinism of their arguments by disabling an interaction when some interaction of higher priority is enabled.

We consider a framework where composite components are built by application of glue operators. Components are characterized by their behavior and represented in some semantic domain $\mathcal{B}$ equipped with an equivalence relation $\mathcal{R}$. For instance, behaviors can be modeled as labeled transition systems, with an equivalence relation such as strong bisimulation, ready simulation, or simulation. In this case, the behavior of a component consists of all its states and transitions. In general, the behavior is not modified when the component takes a transition. This constitutes an important difference with the process algebra setting, where processes evolve to become other processes.

The concept of glue operator is a generalization of operators in BIP. Parallel composition operators of the process calculi CCS, SCCS, or CSP, are glue operators as well as their unary operators such as labeling, hiding and restriction. Contrary to interleaving, non-deterministic choice is not a glue operator, as it requires additional memory: the choice is applied only once and remains the same for all subsequent transitions of the composed system.

A glue on $\mathcal{B}$ is a closed under composition set G of operators transforming behavior, i.e. mapping tuples of behaviors to behaviors. We require that glue operators only restrict the behavior of their arguments without adding new one, i.e. perform memoryless coordination of behavior.

The equivalence relation $\mathcal{R}$ on $\mathcal{B}$ induces an equivalence relation on glue operators: two n-ary glue operators are equivalent if for any n-tuple of behaviors from $\mathcal{B}$ they give equivalent behaviors. The proposed notion of expressiveness allows the comparison of two glues G_1 and G_2 on the same semantic domain $\mathcal{B}$ by considering whether for any glue operator $gl_1 \in G_1$ there exists an equivalent

operator $gl_2 \in G_2$ (strong expressiveness). Weak expressiveness is defined by allowing a finite number of additional behaviors in the arguments of gl_2.

The main results of the paper can be summarized as follows:

- We propose an SOS-style definition of glues, where operators are characterized as sets of SOS-rules specifying the transition relation of composite components from the transition relations of their constituents. The premises of the rules may involve both positive and negative predicates specifying respectively enabledness or non-enabledness of transitions of components.
- We show that relations, induced on glues by strong bisimulation, ready simulation (preorder and equivalence), and simulation equivalence, coincide. This allows a simple characterization of glue operators as formulae of an algebra generated from the set of the ports of the components by using disjunction, conjunction, and negation operators. The algebra has most of the axioms of a boolean algebra. It does not have the absorption axiom, which is replaced by a weaker one.
- Using this algebraic characterization of glues, we provide expressiveness results for the glues of BIP and of process algebras such as CCS, CSP and SCCS. They show that, for the considered expressiveness criteria, glues of the considered process calculi are less expressive than general glue operators. Furthermore, glue of BIP has exactly the same strong expressiveness as glue definable by the SOS characterization.

The paper is structured as follows. In Sect. 2, we define basic notions that we use in the paper: LTS, SOS-style glue operators, and equivalence relations on these; we provide some results that allow, in particular, to define the algebra of glue formulae, $\mathcal{AGF}(P)$, used to encode the glue operators. In Sect. 3 we introduce the notions of glue expressiveness. In Sect. 4, we use $\mathcal{AGF}(P)$ to compare expressiveness of the glues of CCS, CSP, SCCS, and BIP. We conclude, in Sect. 5, by discussing the results and some directions for future work.

2 Labeled Transition Systems and Glue Operators

In this section, we introduce labeled transition systems (LTS), used to describe behavior, as well as composition operators on these defined in terms of SOS [4].

2.1 Labeled Transition Systems

Definition 1. *A* labeled transition system *is a triple $B = (Q, P, \rightarrow)$, 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 labeled by an* action *(i.e. a subset of ports).*

For $q, q' \in Q$ and $a \in 2^P$, we write $q \xrightarrow{a} q'$, if $(q, a, q') \in \rightarrow$. An interaction a is enabled *in state q, denoted $q \xrightarrow{a}$, if there exists $q' \in Q$ such that $q \xrightarrow{a} q'$. If such q' does not exist, a is* disabled*, denoted $q \not\xrightarrow{a}$.*

Notice that reachability related issues are not in the scope of this paper. Consequently, we do not speak of initial states of LTS.

Definition 2. *Let $B_1 = (Q_1, P_1, \rightarrow)$ and $B_2 = (Q_2, P_2, \rightarrow)$ be two LTS, and let $\mathcal{R} \subseteq Q_1 \times Q_2$ be a binary relation. $\mathcal{R}$ is*

1. *a* simulation *iff, for all $q_1 \mathcal{R}\, q_2$, $q_1 \xrightarrow{a} q_1'$ implies $q_2 \xrightarrow{a} q_2'$, for some $q_2' \in Q_2$ such that $q_1' \mathcal{R}\, q_2'$.*
2. *a* ready simulation *iff it is a simulation and, for $q_1 \mathcal{R}\, q_2$, $q_1 \xrightarrow{a}$ implies $q_2 \xrightarrow{a}$.*
3. *a* bisimulation *iff both $\mathcal{R}$ and $\mathcal{R}^{-1}$ are simulations.*

We write $B_1 \sqsubseteq_S B_2$ (resp. $B_1 \sqsubseteq_{RS} B_2$) if there exists a simulation (resp. ready simulation) relating each state of B_1 to some state of B_2. $\sqsubseteq_S$ and $\sqsubseteq_{RS}$ are respectively the simulation *and the* ready simulation preorders *on behaviors. We denote by $\simeq_X = \sqsubseteq_X \cap \sqsubseteq_X^{-1}$, with $X \in \{S, RS\}$, the corresponding equivalences.*

Similarly, $B_1 \leftrightarrows B_2$, iff there exists a bisimulation relating all states of both B_1 and B_2. $\leftrightarrows$ is the bisimulation equivalence *on behaviors.*

Remark 1. It is well known (e.g., [5]) that these relations are connected by the following inclusions: $\leftrightarrows \subseteq \simeq_{RS} \subseteq \simeq_S$ and $\sqsubseteq_{RS} \subseteq \sqsubseteq_S$.

2.2 Glue Operators

Structural Operational Semantics (SOS) [4, 6] has been used to define the semantics of programs in terms of LTS. A number of SOS formats have been developed, using various syntactic features [7].

We consider a very simple setting focusing exclusively on behavior composition. In the context of component-based systems, definition of glue only requires the specification of parallel composition operators, as sequential and recursive computation can be represented by individual behaviors. Below, we propose an SOS rules format for component-based composition.

Definition 3. *An n-ary* glue operator *gl is defined as follows. The application of gl to behaviors $B_i = (Q_i, P_i, \rightarrow)$, for $i \in [1, n]$, is a behavior $gl(B_1, \ldots, B_n) = (Q, P, \rightarrow)$, with state space $Q = \prod_{i=1}^{n} Q_i$ the Cartesian product of Q_i, set of ports $P = \bigcup_{i=1}^{n} P_i$, and the maximal transition relation derivable with a set of inference rules of the form*

$$r = \frac{\{B_i : q_i \xrightarrow{a_i} q_i'\}_{i \in I} \qquad \{B_j : q_j \xrightarrow{b_{j_k}} \mid k \in [1, m_j]\}_{j \in J}}{gl(B_1, \ldots, B_n) : q_1 \ldots q_n \xrightarrow{a} \widetilde{q_1} \ldots \widetilde{q_n}} \begin{matrix}(premises)\\(conclusion)\end{matrix}, \quad (1)$$

where $I, J \subseteq [1, n]$; $a = \bigcup_{i \in I} a_i$; and $\widetilde{q_i} = q_i'$, for $i \in I$, and $\widetilde{q_i} = q_i$, for $i \notin I$. Premises of the form $B : q \xrightarrow{a} q'$ are called positive*, those of the form $B : q \xrightarrow{a}$ are called* negative*. Additionally, we require that*

1. *for each $i \in [1, n]$, r has at most one positive premise involving B_i;*
2. *r has at least one positive premise;*
3. *a label can appear either in positive or in negative premises, but not in both.*[1]

[1] A rule with contradictory premises would never be applicable. We include this condition, as it simplifies further proofs and formulations.

We denote by $Pos(r)$ and $Neg(r)$ the sets of positive and negative premises of r respectively (notice that a rule is completely defined by its premises). We identify the glue operator gl with its defining set of derivation rules. A glue operator having no negative premises in any of its derivation rules is called a positive glue operator.

Lemma 1 ([5]). *Glue operators preserve ready simulation and bisimulation, i.e. $B_1 \mathcal{R}\, B_1'$ implies, $gl(B_1, B_2, \ldots, B_n) \mathcal{R}\, gl(B_1', B_2, \ldots, B_n)$, for any behaviors $B_1, \ldots, B_n$, and B_1', an n-ary glue operator gl, and $\mathcal{R} \in \{\sqsubseteq_{RS}, \simeq_{RS}, \leftrightarrows\}$.*

The simulation preorder is preserved by positive glue operators.

Example 1 (Rendezvous). Consider the family of binary operators $\rho_{a,b}$, parameterized by two labels. For each pair of labels $a, b \in 2^P$, the composite behavior $\rho_{a,b}(B_1, B_2)$ is inferred from B_1 and B_2 by the set of rules

$$\frac{B_1 : q_1 \xrightarrow{a} q_1' \qquad B_2 : q_2 \xrightarrow{b} q_2'}{\rho_{a,b}(B_1, B_2) : q_1 q_2 \xrightarrow{ab} q_1' q_2'}. \tag{2}$$

and, for all $x \neq a$ and $y \neq b$,

$$\frac{B_1 : q_1 \xrightarrow{x} q_1'}{\rho_{a,b}(B_1, B_2) : q_1 q_2 \xrightarrow{x} q_1' q_2}, \quad \frac{B_2 : q_2 \xrightarrow{y} q_2'}{\rho_{a,b}(B_1, B_2) : q_1 q_2 \xrightarrow{y} q_1 q_2'}. \tag{3}$$

For two behaviors B_1, B_2 having transitions labeled respectively by a and b, $\rho_{a,b}(B_1, B_2)$ is the parallel composition of B_1 and B_2, where a strong synchronization of a and b is the only possible action.

Example 2 (Broadcast). Consider the family of binary operators $\beta_{a,b}$, parameterized by two labels. For each pair of labels $a, b \in 2^P$, the composite behavior $\beta_{a,b}(B_1, B_2)$ is inferred from B_1 and B_2 by the set of rules

$$\frac{B_1 : q_1 \xrightarrow{a} q_1'}{\beta_{a,b}(B_1, B_2) : q_1 q_2 \xrightarrow{a} q_1' q_2}, \quad \frac{B_1 : q_1 \xrightarrow{a} q_1' \qquad B_2 : q_2 \xrightarrow{b} q_2'}{\beta_{a,b}(B_1, B_2) : q_1 q_2 \xrightarrow{ab} q_1' q_2'}. \tag{4}$$

For two behaviors B_1, B_2 having transitions labeled respectively by a and b, $\beta_{a,b}(B_1, B_2)$ is the parallel composition of B_1 and B_2, where interactions a and b are weakly synchronized with a being the trigger. In other words, B_2 can perform a transition on b only if it is synchronized with a transition of B_1 on a.

Example 3 (Priority). Consider the family of unary operators $\pi_{a,b}$, parameterized by two labels. For each pair of labels $a, b \in 2^P$, the composite behavior $\pi_{a,b}(B)$ is inferred from B by the set of rules

$$\frac{B : q \xrightarrow{a} q' \qquad B : q \not\xrightarrow{b}}{\pi_{a,b}(B) : q \xrightarrow{a} q'}, \quad \frac{q \xrightarrow{b} q'}{\pi_{a,b}(B) : q \xrightarrow{b} q'}. \tag{5}$$

For a behavior B having transitions labeled by a and b, $\pi_{a,b}(B)$ is the restriction of B, where the interaction a can only happen if b is not possible, i.e. a has lower priority than b.

2.3 Relations on Glue Operators

The relations on LTS defined above are canonically extended to glue operators.

Definition 4. *For $\mathcal{R} \in \{\sqsubseteq_S, \sqsubseteq_{RS}, \simeq_S, \simeq_{RS}, \leftrightarrows\}$, the relation $\mathcal{R}$ is extended to glue operators by putting, for any two n-ary glue operators gl_1 and gl_2,*

$$gl_1 \mathcal{R}\, gl_2 \stackrel{def}{\Longleftrightarrow} \forall B_1, \ldots, B_n,\ gl_1(B_1, \ldots, B_n) \mathcal{R}\, gl_2(B_1, \ldots, B_n)\,. \tag{6}$$

Clearly, the inclusions of Rem. 1 also hold for relations on glue operators.

Lemma 2. *Two glue operators $gl_1 = \{r_1\}$ and $gl_2 = \{r_1, r_2\}$, with $Pos(r_1) = Pos(r_2)$ and $Neg(r_1) \subseteq Neg(r_2)$, are bisimilar $gl_1 \leftrightarrows gl_2$.*

Proof. The proof follows immediately from the definition of bisimilarity. It is based on the fact that, whenever r_2 is applicable, r_1 can also be applied. □

Definition 5. *If a glue operator does not have redundant rules as in Lem. 2, we say that it is* without redundancy.

Lemma 3. *Let gl_1, gl_2 be glue operators, and gl_1 be without redundancy. $gl_1 \sqsubseteq_S gl_2$ implies that, for each rule $r_1 \in gl_1$, there is a rule $r_2 \in gl_2$ having $Pos(r_2) = Pos(r_1)$ and $Neg(r_2) \subseteq Neg(r_1)$.*

Proof. Consider the rule (cf. Def. 3)

$$r_1 \;=\; \frac{\{B_i : q_i \xrightarrow{a_i} q_i'\}_{i \in I} \qquad \{B_j : q_j \stackrel{b_{j_k}}{\not\longrightarrow} \mid k \in [1, m_j]\}_{j \in J}}{gl(B_1, \ldots, B_n) : q_1 \ldots q_n \xrightarrow{a} \widetilde{q_1} \ldots \widetilde{q_n}} \;\in gl_1\,,$$

and, for $i \in [1, n]$, $B_i^1 = (Q_i, P, \rightarrow_i)$ having $Q_i = \{q^i\}$ and $\rightarrow_i$ defined by

$$\rightarrow_i = \begin{cases} \{q^i \xrightarrow{a} q^i \mid a \in 2^P\}, & \text{for } i \notin J, \\ \{q^i \xrightarrow{a} q^i \mid a \in 2^P\} \setminus \{q^i \xrightarrow{b_{i_k}} q^i \mid k \in [1, m_i]\}, & \text{for } i \in J\,. \end{cases} \tag{7}$$

Both behaviors obtained by applying gl_1 and gl_2 to $B_1^1, \ldots, B_n^1$ have exactly one state that we denote respectively q' and q''.

All the premises of r_1 are satisfied in q'. Hence $q' \xrightarrow{a} q'$ in $gl_1(B_1^1, \ldots, B_n^1)$. By simulation $gl_1 \sqsubseteq_S gl_2$, we also have $gl_1(B_1^1, \ldots, B_n^1) \sqsubseteq_S gl_2(B_1^1, \ldots, B_n^1)$. Hence, $q'' \xrightarrow{a} q''$ in $gl_2(B_1^1, \ldots, B_n^1)$, and there exists a rule $r_2 \in gl_2$ enabling this transition. Thus, $Pos(r_2) = Pos(r_1)$ and $Neg(r_2) \subseteq Neg(r_1)$. □

Proposition 1. *Let gl_1, gl_2 be glue operators without redundancy. Then $gl_1 \simeq_S gl_2$ implies $gl_1 = gl_2$, where $=$ is the equality of sets of derivation rules.*

Proof. Consider a rule $r_1 \in gl_1$. By Lem. 3, $gl_1 \sqsubseteq_S gl_2$ implies that there exists $r_2 \in gl_2$ having $Pos(r_2) = Pos(r_1)$ and $Neg(r_2) \subseteq Neg(r_1)$, whereas $gl_2 \sqsubseteq_S gl_1$ implies that there exists $r_1' \in gl_1$ having $Pos(r_1') = Pos(r_2)$ and $Neg(r_1') \subseteq Neg(r_2)$. By non-redundancy of gl_1, we conclude $r_1' = r_1 = r_2$, i.e. $gl_1 \subseteq gl_2$. By symmetry, this proves the proposition. □

Proposition 2. *Let gl_1, gl_2 be glue operators without redundancy. Then $gl_1 \sqsubseteq_{RS} gl_2$ implies $gl_1 = gl_2$, where $=$ is the equality of sets of derivation rules.*

Proof. 1) Let $gl_1 \sqsubseteq_{RS} gl_2$ and consider a rule $r_1 \in gl_1$. By Lem. 3, there exists $r_2 \in gl_2$, such that $Pos(r_2) = Pos(r_1)$ and $Neg(r_2) \subseteq Neg(r_1)$. Suppose that $Neg(r_2) \subsetneq Neg(r_1)$, i.e. there exists a negative premise $(B : q \stackrel{b}{\not\rightarrow}) \in Neg(r_1) \setminus Neg(r_2)$. Consider, for $i \in [1, n]$, the behaviors $B_i^2 = (Q_i, P, \rightarrow_i)$, constructed as in (7), but removing the transition corresponding to this premise. As in the proof of Lem. 3, we denote q' and q'' the unique states of $gl_1(B_1^2, \ldots, B_n^2)$ and $gl_2(B_1^2, \ldots, B_n^2)$ respectively. Clearly all the premises of r_2 are still satisfied and a transition $q'' \stackrel{a}{\rightarrow} q''$ is possible in $gl_2(B_1^2, \ldots, B_n^2)$. On the other hand, the premises of r_1 are no longer satisfied.

Suppose that there exists another rule $r_1' \in gl_1$, which allows the transition $q' \stackrel{a}{\rightarrow} q'$ in $gl_1(B_1^2, \ldots, B_n^2)$. As above, we have $Pos(r_1') = Pos(r_1) = Pos(r_2)$ and $Neg(r_1') \subseteq Neg(r_2) \subseteq Neg(r_1)$, which violates the non-redundancy assumption.

Assuming that there is no such rule $r_1' \in gl_1$, we conclude that $q' \stackrel{a}{\not\rightarrow}$ in $gl_1(B_1^2, \ldots, B_n^2)$, which, by ready simulation, implies a contradiction: $q'' \stackrel{a}{\not\rightarrow}$ in $gl_2(B_1^2, \ldots, B_n^2)$. Hence, $Neg(r_2) = Neg(r_1)$, i.e. $r_1 = r_2 \in gl_2$ and $gl_1 \subseteq gl_2$.

2) Assume now that $gl_1 \subsetneq gl_2$, i.e. there exists a rule $r_2 \in gl_2 \setminus gl_1$ with conclusion labeled by some interaction a. We consider again the behaviors B_i^1, for $i \in [1, n]$, constructed as above. Again, we have $q'' \stackrel{a}{\rightarrow} q''$ in $gl_2(B_1^1, \ldots, B_n^1)$, and, by ready simulation, $q' \stackrel{a}{\rightarrow} q'$ in $gl_1(B_1^1, \ldots, B_n^1)$. Hence, there exists $r_1 \in gl_1 \subseteq gl_2$ enabling this transition. As above, we have $Pos(r_1) = Pos(r_2)$ and $Neg(r_1) \subseteq Neg(r_2)$, which contradicts the non-redundancy of r_2. □

This proves the following theorem, implying that to compare glue operators it is sufficient to compare the corresponding sets of SOS rules.

Theorem 1. *Bisimulation, ready simulation preorder and equivalence, and simulation equivalence on glue operators coincide: $\leftrightarrows = \simeq_{RS} = \simeq_S = \sqsubseteq_{RS}$. All these relations coincide with the equality of operators as sets of rules.*

2.4 The Algebra of Glue Formulae

Theorem 1 also allows to define an algebraic encoding of glue operators, which we use, in particular, to define the composition of glue operators. Glue composition must preserve essential information about atomic behavior. For instance, if interaction a is inhibited by some other interaction b, this relation must be maintained even when, in the composed system, b cannot be fired for some other reason: b must be synchronized with another interaction that is not enabled in the current state, or it is itself inhibited by another interaction.

For instance, assume that firing the interaction a takes one of the components to a critical state, for which mutual exclusion must be ensured, whereas firing b takes another component out of such state. If b is possible, a should not be fired (as this would violate the mutual exclusion) even if b is inhibited by another interaction c.

Although, a definition of composition, which respects these requirements, can be given directly in terms of glue operators, it is much simpler and more intuitive to give it in terms of the algebra presented below. An up to bisimulation one-to-one correspondence between formulae of the algebra and glue operators ensures the translation of composition back to glue operators.

Syntax Let P be a set of ports, such that $0, 1 \notin P$. The syntax of the algebra of glue formulae, $f \in \mathcal{AGF}(P)$, is given by

$$\begin{array}{l} f ::= f \vee f \mid f \wedge t \mid e\,, \\ t ::= (t \vee t) \mid \neg e \mid e\,, \\ e ::= e \vee e \mid e \wedge e \mid (e) \mid a \in 2^P \mid 0 \mid 1\,, \end{array} \tag{8}$$

where the three operations, denoted by '$\neg$', '$\wedge$', and '$\vee$' are respectively unary *negation* and binary *conjunction* and *disjunction* (in order of decreasing binding power). We often omit '$\wedge$' and represent conjunction by simple juxtaposition.

Intuitively, e represents a positive expression, whereas t is a term which can have a negation at top level, i.e. the negated term must be purely positive. As t can only appear in conjunction with another term, a negative term, in $\mathcal{AGF}(P)$ formulae, is always in conjunction with a positive term.

Axioms Both $\wedge$ and $\vee$ are associative, commutative, idempotent, and distribute over each other; 0 is the unit for $\vee$; 1 is the unit for $\wedge$; $f \wedge 0 = 0$. Negation satisfies all the axioms except double negation and excluded middle:

1. $\neg 0 = 1$ and $\neg 1 = 0$,
2. $f \wedge \neg f = 0$,
3. $\neg f_1 \wedge \neg f_2 = \neg(f_1 \vee f_2)$,
4. $\neg f_1 \vee \neg f_2 = \neg(f_1 \wedge f_2)$.

Lemma 4 (Restricted absorption). $\forall f_1, f_2 \in \mathcal{AGF}(P)$, $f_1 \neg f_2 \vee f_1 = f_1$.

Lemma 5. *Each formula $f \in \mathcal{AGF}(P)$ has a disjunctive normal form, i.e. a representation as a disjunction of conjunctions of positive and negative variables.*

Proof (Sketch). This follows from the existence of the DNF of boolean formulae and the fact that the syntax (8) of $\mathcal{AGF}(P)$ guarantees that negation only appears at the top level. This property is also preserved by de Morgan's laws. □

Semantics The semantics of $\mathcal{AGF}(P)$ is given in terms of glue operators. It depends on the mapping of ports in P to components. For a system with n atomic components $B_1, \ldots, B_n$, the partition of P is given by a function $\kappa : P \to [1, n]$, such that a port $p \in P$ belongs to the component $B_{\kappa(p)}$. The function κ trivially extends to interactions within one component.

The meaning of a clause $a_1 \ldots a_k \wedge \neg b_1 \ldots \neg b_l$ is given by the rule r, having

$$\begin{array}{l} Pos(r) = \left\{ B_{\kappa(a_i)} : q_{\kappa(a_i)} \xrightarrow{a_i} q'_{\kappa(a_i)} \;\middle|\; i \in [1,k] \right\}, \\ Neg(r) = \left\{ B_{\kappa(b_i)} : q_{\kappa(b_i)} \not\xrightarrow{b_i} \;\middle|\; i \in [1,l] \right\}. \end{array}$$

Clearly, the DNF of $f \neq 0, 1$, does not contain occurrences of neither 0 nor 1. The meaning of the formula $f \neq 0, 1$ is then given by the glue operator

gl_f defined by the set of rules, corresponding to clauses of the DNF of f. The meaning of 0 is the operator defined by the empty set of derivation rules, which blocks all interactions of all the components in the system.

Proposition 3. *The axiomatization of $\mathcal{AGF}(P)$ is sound and complete, i.e., for two formulae $f_1, f_2 \in \mathcal{AGF}(P)$, $f_1 = f_2$ if and only if $gl_{f_1} \leftrightarrows gl_{f_2}$.*

For any glue operator gl, there exists $f \in \mathcal{AGF}(P)$, such that $gl \leftrightarrows gl_f$.

Proof (Sketch). Clearly, the semantic construction above is one-to-one between $\mathcal{AGF}(P)$ formulae and glue operators without redundancy. The second part follows directly from Lemmas 2 and 4. □

This proposition allows to identify the glue operators with their corresponding glue formulae. We use this to define a composition of glue operators. The usual composition is not compatible with the restriction that all interactions in positive premises of a rule must participate in the conclusion (cf. Def. 3).

Example 4. Consider two operators defined by the corresponding formulae $f = a\neg b \vee b \vee c$ and $g = a \vee b\neg c \vee c$ (cf. also the opening of this section). The usual composition $f \circ g$ consists here in substituting $b\neg c$ for b in f. Thus, in f, $a\neg b$ becomes $a\neg(b\neg c) = a\neg b \vee a\neg\neg c$, b becomes $b\neg c$, and c stays the same. This gives $f \circ g = a\neg b \vee a\neg\neg c \vee b\neg c \vee c$. However, $a\neg\neg c$ is not authorized by the syntax of $\mathcal{AGF}(P)$. In terms of SOS, this would correspond to having a positive premise c that would not participate in the conclusion of the rule.

Consider two glue operators defined by the formulae $f = \bigvee_{i\in I} a_i x_i$ and $g = \bigvee_{j\in J} b_j y_j$, where, for $i \in I$ and $j \in J$, a_i and b_j are conjunctions of positive interaction variables, whereas x_i and y_j are purely negative expressions.

Definition 6. *The composition of f with g is defined by*

$$f * g \stackrel{def}{=} \bigvee_{i\in I} \bigvee_{K\subseteq J} \left(x_i \wedge \bigwedge_{k\in K} b_k y_k \right) = \bigvee_{i\in I} \bigvee_{K\subseteq J} \left(a_i x_i \wedge \bigwedge_{k\in K} y_k \right), \qquad (9)$$

where the inner disjunction is taken over all $K \subseteq J$, such that $\bigwedge_{k\in K} b_k = a_i$.

Example 5. Taking on the previous example, we have $f * g = a\neg b \vee b\neg c \vee c$. Thus, when all three interactions a, b, and c are ready to be fired, both a and b are inhibited by b and c respectively.

3 Expressiveness of Glue

Let $\mathcal{B}$ be a set of behaviors with a fixed equivalence relation $\mathcal{R} \subseteq \mathcal{B} \times \mathcal{B}$. A glue is a set G of operators on $\mathcal{B}$. We denote by $\mathcal{G}lue$ the set of all glues on $\mathcal{B}$. We denote $G^{(n)} \subseteq G$ the set of all n-ary operators in G. Thus, $G = \bigcup_{n\geq 1} G^{(n)}$.

To determine whether one glue is more expressive than another, we compare their respective sets of behaviors *composable* from the same *atomic* ones. Several

approaches to comparing the expressiveness of glues can be considered according to the type of modifications of the system that one allows in order to perform the comparison. In any case, this consists in exhibiting for each operator of one glue an equivalent operator in the other one. Below, we define two criteria for the comparison of glue expressiveness:

1. *Strong* expressiveness, where the exhibited glue operator must be applied to the same set of behaviors as the original one,
2. *Weak* expressiveness, where the exhibited glue operator must be applied to the same set of behaviors as the original one, with potentially an addition of some fixed set of *coordination behaviors*.

Definition 7. *For a given set $\mathcal{B}$ and an equivalence $\mathcal{R}$ on $\mathcal{B}$, the* strong *expressiveness preorder $\preccurlyeq_S \subseteq \mathcal{G}lue \times \mathcal{G}lue$ w.r.t. $\mathcal{R}$ is defined by putting, for G_1, $G_2 \in \mathcal{G}lue$, $G_1 \preccurlyeq_S G_2$ if, for any $n \geq 1$ and $B_1, \ldots, B_n \in \mathcal{B}$,*

$$\forall gl_1 \in G_1^{(n)} \ \exists gl_2 \in G_2^{(n)} : gl_1(B_1, \ldots, B_n) \,\mathcal{R}\, gl_2(B_1, \ldots, B_n)\,. \tag{10}$$

Definition 8. *For a given set $\mathcal{B}$ and an equivalence $\mathcal{R}$ on $\mathcal{B}$, the* weak *expressiveness preorder $\preccurlyeq_W \subseteq \mathcal{G}lue \times \mathcal{G}lue$ w.r.t. $\mathcal{R}$ is defined by putting, for G_1, $G_2 \in \mathcal{G}lue$, $G_1 \preccurlyeq_W G_2$ if there exists a finite subset $\mathcal{C} \subset \mathcal{B}$ of coordination behaviors such that, for any $n \geq 1$ and $B_1, \ldots, B_n \in \mathcal{B}$,*

$$\begin{aligned} \forall gl_1 \in G_1^{(n)} \ \exists C_1, \ldots, C_m \in \mathcal{C}, gl_2 \in G_2^{(n+m)} : \\ gl_1(B_1, \ldots, B_n) \,\mathcal{R}\, gl_2(B_1, \ldots, B_n, C_1, \ldots, C_m)\,. \end{aligned} \tag{11}$$

These two preorders allow to define the following notions for glue comparison.

Definition 9. *Let $G_1, G_2 \in \mathcal{G}lue$. The following relations are defined w.r.t. $\mathcal{R}$.*

1. *G_1 and G_2 are* strongly equivalent *if $G_1 \preccurlyeq_S G_2$ and $G_2 \preccurlyeq_S G_1$.*
2. *G_1 and G_2 are* weakly equivalent *if $G_1 \preccurlyeq_W G_2$ and $G_2 \preccurlyeq_W G_1$.*
3. *G_1 is* strongly more expressive *than G_2 if $G_2 \preccurlyeq_S G_1$ and $G_1 \not\preccurlyeq_W G_2$.*
4. *G_1 is* weakly more expressive *than G_2 if $G_2 \preccurlyeq_W G_1$ and $G_1 \not\preccurlyeq_W G_2$.*

Remark 2. The two order relations ("strongly more expressive" and "weakly more expressive") defined above are partial orders (as opposed to preorders). However, notice that we define the relation "strongly more expressive" to be stronger than the canonical order induced by the preorder $\preccurlyeq_S$. As $G_1 \preccurlyeq_S G_2$ implies $G_1 \preccurlyeq_W G_2$, the case $G_1 \preccurlyeq_S G_2$ and $G_2 \preccurlyeq_W G_1$ fits the case 2 above, i.e. G_1 and G_2 are weakly equivalent.

Example 6. We consider behaviors to be LTS. Let $\mathcal{P}$ be a universal set of ports. We define two glues Bin and Ter generated respectively by families of binary and ternary rendezvous operators: $\rho^{(2)}_{a,b}$ and $\rho^{(3)}_{a,b,c}$ for all $a, b, c \in 2^{\mathcal{P}}$ (cf. Ex. 1).

Clearly, $Ter \preccurlyeq_S Bin$, as for any $a, b \in 2^{\mathcal{P}}$, and any B_1, B_2, B_3, we have $\rho^{(3)}_{a,b,c}(B_1, B_2, B_3) = \rho^{(2)}_{a,bc}\Big(B_1, \rho^{(2)}_{b,c}(B_2, B_3)\Big)$. On the contrary, $Bin \not\preccurlyeq_W Ter$, as any two components at any given state can only perform two actions (one action each), whereas three are needed for a ternary synchronization. We conclude that Bin is strongly more expressive than Ter.

We have supposed so far that systems are built from components with disjoint sets of ports and that all actions are observable. To compare expressiveness of formalisms that do not meet this requirements, we adapt the definition of glue expressiveness by using labeling functions that modify the labeling of transitions without otherwise affecting the transition relation.

Definition 10. *Let $\mathcal{B}$ be a set of behaviors with a universal set of ports P and $\mathcal{R}$ an equivalence on $\mathcal{B}$. Let $\varphi, \psi : P \to P$ be two given labeling functions, such that $\psi \circ \varphi = id$. The strong (φ, ψ)-expressiveness preorder $\preccurlyeq_S^{(\varphi,\psi)} \subseteq \mathcal{G}lue \times \mathcal{G}lue$ w.r.t. $\mathcal{R}$ is defined by putting, for G_1, $G_2 \in \mathcal{G}lue$, $G_1 \preccurlyeq_S^{(\varphi,\psi)} G_2$ iff, for any $n \geq 1$ and $B_1, \ldots, B_n \in \mathcal{B}$,*

$$\forall gl_1 \in G_1^{(n)} \; \exists gl_2 \in G_2^{(n)} : gl_1(B_1, \ldots, B_n) \mathcal{R} \, \psi\left(gl_2\Big(\varphi(B_1), \ldots, \varphi(B_n)\Big)\right) , \quad (12)$$

where e.g., $\varphi(B)$ is the behavior obtained from B by applying φ to labels of all the transitions in B.

Definitions 8 and 9 are adapted analogously.

Example 7. Taking on the previous example, consider $\tau \notin \mathcal{P}$, and let $C = (\{1\}, \{\tau\}, \rightarrow)$ be an LTS with the only transition $1 \xrightarrow{\tau} 1$.

For any B_1, B_2 and $a, b \in 2^{\mathcal{P}}$, we have $\rho^{(2)}_{a,b}(B_1, B_2) \, \mathcal{R} \, \psi\Big(\rho^{(3)}_{a,b,\tau}(B_1, B_2, C)\Big)$, where ψ is a labeling function erasing all occurrences of τ, and $\mathcal{R}$ is any of the equivalence relations discussed in Sect. 2.3. Thus, $Bin \preccurlyeq_W^{(id,\psi)} Ter$, i.e. Bin and Ter are weakly (id, ψ)-equivalent.

4 Glues of BIP and Process Algebras

In the following sections, we compare the glues of BIP [3] and those of classical calculi: CSP [8], CCS [9], and SCCS [10]. All these glues are positive and consist in their respective parallel composition and restriction operators.

Lemma 6. *Let G_1 and G_2 be two positive glues (i.e. consisting of only positive glue operators). $G_1 \preccurlyeq_S G_2$ with respect to any of $\simeq_S$, $\simeq_{RS}$, and $\leftrightarrows$ iff $G_1 \subseteq G_2$.*

Proof (Sketch). Consider a family of behaviors, each having one state with loop transitions on all corresponding interactions. To enable exactly the same transitions, two positive glue operators must have exactly the same rules. □

4.1 BIP

In BIP [3], behavior composition is by means of interaction models – sets of interactions, described by connectors [11] – and priority models (partial orders on interactions), used to enforce scheduling policies applied to interactions.

The composition operator, defined by a set of interactions $\gamma \subseteq 2^P$, is given by the $\mathcal{AGF}(P)$ formula $int_\gamma = \bigvee \gamma$ (the disjunction of all the interactions in

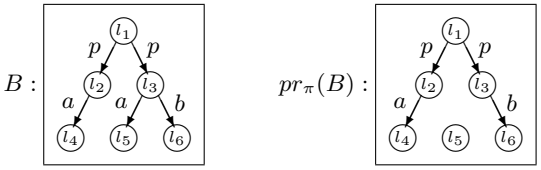

Fig. 1. Example behavior for the proof of Prop. 5

γ). We denote by IM the set of all such glue operators. As interactions are sets of ports, operators in IM are purely positive (each clause of such an operator being a conjunction of positive port variables).

A *priority model* π is a strict partial order on 2^P. For $a, a' \in 2^P$, we write $a \prec a'$ iff $(a, a') \in \pi$, meaning that interaction a has less priority than a'. The priority operator is given by the $\mathcal{AGF}(P)$ formula

$$pr_\pi \;=\; \bigvee_{a \in 2^P} \left(a \wedge \bigwedge_{a \prec a'} \neg a' \right) .$$

We denote BIP the glue consisting of all operators obtained by composition (cf. Def. 6) of interaction and priority operators.

Proposition 4. *IM is strongly equivalent to the set of all positive glue operators, whereas BIP is strongly equivalent to the set of all glue operators.*

Proof (Sketch). The first affirmation is trivial. It is also clear from the above presentation that any operator in BIP is a glue operator in the sense of Def. 3. To show that any glue operator can be realized in BIP, we represent it as a DNF formula $f \in \mathcal{AGF}(P)$. Each conjunctive clause of f has a positive and a negative part. The positive parts of all clauses uniquely define the the set of interactions γ, whereas regrouping (by de Morgan's laws) the negative parts of all clauses with the same positive part defines the priority model. □

Proposition 5. *BIP is strongly more expressive than IM w.r.t. $\simeq_S$ (*a fortiori *$\simeq_{RS}$ and $\leftrightarrows$).*

Proof (Sketch). First of all, $IM \subset BIP$ and IM contains only positive operators. Hence, by Lem. 6, $IM \preccurlyeq_S BIP$ and $BIP \not\preccurlyeq_S IM$. As, in BIP, all interactions are visible this also implies $BIP \not\preccurlyeq_W IM$. It is easy to show $BIP \not\preccurlyeq_W^{(id,\psi)} IM$, with ψ erasing all the ports of coordination behavior, by considering the priority model $\pi = \{(a, b)\}$ applied to the behavior B (see Fig. 1). □

4.2 CCS and SCCS

In both CCS and SCCS [9, 10], one considers the set A of actions along with the set $\overline{A} = \{\overline{a} \,|\, a \in A\}$ of complementary actions and the non-observable action τ. $L = A \cup \overline{A} \cup \{\tau\}$ is the set of labels.

To render the action sets of different components disjoint, we consider (φ, ψ)-expressiveness. We define $\varphi(B)$ to be a behavior obtained from B by renaming any action $a \in A \cup \overline{A}$ of B to $B.a$. Conversely $\psi(B)$ renames any action $B.a$ of B to a. Furthermore, for any behaviors B_1, B_2 we put $\psi(B_1.a\, B_2.\overline{a}) = \tau$.

The glue of CCS consists of operators obtained by hierarchical composition of the parallel composition and restriction operators. Parallel composition $\|$ operator allows binary synchronization of complementary actions $a, \overline{a} \in L$. Restriction $\backslash a$ excludes a given action $a \in A$ and its complement $\overline{a}$ from communication, thus enforcing synchronization $a\,\overline{a}$, when it is possible.

For a system composed of n atomic behaviors $B_1, \ldots, B_n$, we consider prefixed labels as ports, i.e. $P = \{B_i.a \,|\, i \in [1, n], a \in L\}$. The CCS parallel composition operator is expressed in $\mathcal{AGF}(P)$ by the formula

$$par_{CCS} = \bigvee_{a \in A} \bigvee_{i,j=1}^{n} B_i.a\, B_j.\overline{a} \quad \vee \quad \bigvee_{a \in A} \bigvee_{i=1}^{n} (B_i.a \vee B_i.\overline{a} \vee B_i.\tau)\,. \tag{13}$$

The unary restriction (i.e. applied to a single component) and the n-ary restriction (i.e. combined with the parallel composition of n components) operators are given respectively by

$$rst_{a,1} = \bigvee_{i=1}^{n} \bigvee_{l \neq a, \overline{a}} B_i.l\,, \tag{14}$$

$$rst_{a,n} = \bigvee_{l \in A} \bigvee_{i,j=1}^{n} B_i.l\, B_j.\overline{l} \quad \vee \quad \bigvee_{l \in A \setminus \{a\}} \bigvee_{i=1}^{n} (B_i.l \vee B_i.\overline{l} \vee B_i.\tau)\,. \tag{15}$$

All operators in the CCS glue are positive. Moreover, a conjunctive clause of the corresponding $\mathcal{AGF}(P)$ formula consists of at most two ports. As the labeling ψ can only erase ports, this remains true even in presence of coordination behavior. These observations allow us to conclude that, with φ and ψ as above, IM is strongly more (φ, ψ)-expressive than the CCS glue.

In SCCS, labels are elements of a free Abelian group Act generated by A (with $\overline{a}$ being the inverse of a). The glue of SCCS also consists of hierarchical combinations of the parallel composition and restriction operators. Parallel composition $\times$ forces all components to synchronize. It is given by the formula

$$par_{SCCS} = \bigwedge_{i=1}^{n} \left(B_i.\tau \vee \bigvee_{a \in A} B_i.a \right)\,. \tag{16}$$

Restriction operator in SCCS is complementary to that of CCS, i.e. it states the actions that are visible, rather than those that are invisible. Although, it can be easily defined in terms of $\mathcal{AGF}(P)$, we omit this definition here.

The SCCS glue is also positive, which, as above, allows to conclude that it is strongly less (φ, ψ)-expressive than IM. The opposite relation remains to be investigated. The fact that conjunctive clauses of $SCCS$ operators can comprise more than two ports suggests that CCS glue is not weakly more expressive than SCCS glue.

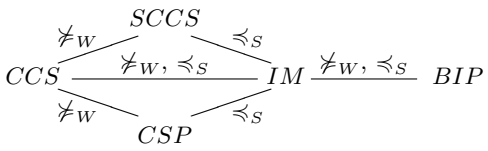

Fig. 2. Summary of relations between glues

4.3 CSP

In CSP [8], processes communicate over a set C of *channels* common to the system. Again, we consider the same relabeling functions φ and ψ as in the previous sections, and ports $P = \{B_i.c \,|\, i \in [1,n], c \in C \cup \{\tau\}\}$.

Again the glue of CSP consists of hierarchical combinations of the parallel composition and restriction operator. Parallel composition $\|_{C'}$ is parameterized by the subset $C' \subseteq C$ of channels. Interactions on the channels in C' must synchronize, whereas interactions on other channels interleave. This is given by the formula

$$par_{CSP} = \bigvee_{c \in C'} \bigwedge_{i=1}^{n} B_i.c \;\vee\; \bigvee_{c \notin C'} \bigvee_{i=1}^{n} (B_i.\tau \vee B_i.c)\,. \tag{17}$$

Again, for the sake of brevity, we omit the restriction operator.

The CSP glue is also positive. It can be observed that conjunctive clauses of the corresponding $\mathcal{AGF}(P)$ formulae consist exclusively of ports corresponding to the same channel. This suggests that, as for the CCS glue, IM is strongly more (φ, ψ)-expressive than the CSP glue. Again, the fact that conjunctive clauses of the operators of the CSP can comprise more than two ports suggests that the CCS glue is not weakly more expressive than the CSP glue.

Relations between the glues considered above are summarized in Fig. 2.

5 Conclusion

We studied notions for comparing expressiveness of glues in component-based frameworks. In contrast to usual notions, they enforce separation between behavior and composition operators. For instance, it is not possible to have as in process algebras, expansion theorems expressing parallel composition in terms of non-deterministic choice and prefixing by actions.

The definition of glue operators considers transitions of composite components as the result of the transitions of their constituents. We showed that bisimilarity, ready simulation (preorder and equivalence), and simulation equivalence coincide, when canonically extended to glue operators. This allows the characterization of glues as formulae, which drastically simplifies the comparison and composition of glues.

We have not yet completely explored possible relations between glues of process calculi. However, they cannot be as expressive as glues with negative

premises and this weakness cannot be overcome even by allowing additional behavior for coordination.

We have kept the framework as simple as possible. We only consider behavioral preorders where all the ports are observable. The robustness of the presented results for expressiveness based on observational relations should be investigated. Furthermore, we have not considered rules with lookahead premises (e.g., [7]) which seems to increase expressiveness of positive rules.

We proposed a framework for dealing with expressiveness of composition operators. This is a step towards breaking with reductionistic approaches which consider glue operators only as behavior transformers. It allows setting up criteria for comparing component-based languages and understanding their strengths and weaknesses.

Acknowledgements

The authors are grateful to Philippe Bidinger, Yassine Lakhnech, and the anonymous reviewers for valuable discussion and constructive comments regarding this paper.

References

1. Felleisen, M.: On the expressive power of programming languages. In: 3^{rd} European Symposium on Programming (ESOP'90). Volume 432 of LNCS., Springer (1990) 134–151
2. Sifakis, J.: A framework for component-based construction. In: 3^{rd} IEEE Int. Conf. on Software Engineering and Formal Methods (SEFM05). (September 2005) 293–300 Keynote talk.
3. Basu, A., Bozga, M., Sifakis, J.: Modeling heterogeneous real-time components in BIP. In: 4^{th} IEEE Int. Conf. on Software Engineering and Formal Methods (SEFM06). (September 2006) 3–12 Invited talk.
4. Plotkin, G.D.: A structural approach to operational semantics. Technical Report DAIMI FN-19, University of Aarhus (1981)
5. Bloom, B.: Ready Simulation, Bisimulation, and the Semantics of CCS-Like Languages. PhD thesis, Massachusetts Institute of Technology (1989)
6. Aceto, L., Fokkink, W., Verhoef, C.: Chapter 3. Structural Operational Semantics. In: Handbook of Process Algebra. Elsevier (2001) 197–292
7. Mousavi, M., Reniers, M.A., Groote, J.F.: SOS formats and meta-theory: 20 years after. Theoretical Computer Science **373**(3) (2007) 238–272
8. Hoare, C.A.R.: Communicating Sequential Processes. Prentice Hall International Series in Computer Science. Prentice Hall (April 1985)
9. Milner, R.: Communication and Concurrency. Prentice Hall International Series in Computer Science. Prentice Hall (1989)
10. Milner, R.: Calculi for synchrony and asynchrony. Theoretical Computer Science **25**(3) (1983) 267–310
11. Bliudze, S., Sifakis, J.: The algebra of connectors — Structuring interaction in BIP. In: Proc. of the EMSOFT'07, ACM SigBED (October 2007) 11–20