Causal Semantics for the Algebra of Connectors
(Extended Abstract)

Simon Bliudze and Joseph Sifakis

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

Abstract. The Algebra of Connectors $\mathcal{AC}(P)$ is used to model structured interactions in the BIP component framework. Its terms are *connectors*, i.e. relations describing synchronization constraints between the ports of component-based systems. Connectors are structured combinations of two basic synchronization protocols between ports: *rendezvous* and *broadcast*. They are generated from the ports of P by using a binary *fusion* operator and a unary *typing* operator. Typing associates with terms (ports or connectors) synchronization types: *trigger* or *synchron*.

In a previous paper, we studied interaction semantics for $\mathcal{AC}(P)$ which defines the meaning of connectors as sets of interactions. This semantics reduces broadcasts into the set of their possible interactions and thus blurs the distinction between rendezvous and broadcast. It leads to exponentially complex models that cannot be a basis for efficient implementation. Furthermore, the induced semantic equivalence is not a congruence.

For a subset of $\mathcal{AC}(P)$, we propose a new *causal* semantics that does not reduce broadcast into a set of rendezvous and explicitly models the causal dependency relation between triggers and synchrons. The Algebra of Causal Trees $\mathcal{CT}(P)$ formalizes this subset. It is the set of the terms generated from interactions on the set of ports P, by using two operators: a *causality* operator and a *parallel composition* operator. Terms are sets of trees where the successor relation represents causal dependency between interactions: an interaction can participate in a global interaction only if its parent participates too. We show that causal semantics is consistent with interaction semantics. Furthermore, it defines an isomorphism between $\mathcal{CT}(P)$ and the set of the terms of $\mathcal{AC}(P)$ involving triggers.

Finally, we define for causal trees a boolean representation in terms of *causal rules*.

1 Introduction

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

F.S. de Boer et al. (Eds.): FMCO 2007, LNCS 5382, pp. 179–199, 2008.
© Springer-Verlag Berlin Heidelberg 2008

One of the main limitations of the current state-of-the-art is the lack of a unified paradigm for describing and analyzing information flow between components. Such a paradigm would allow system designers and implementers to formulate their solutions in terms of tangible, well-founded and organized concepts instead of using disparate coordination mechanisms such as semaphores, monitors, message passing, remote call, protocols etc. A unified paradigm should allow a comparison of otherwise unrelated architectural solutions and could be a basis for evaluating them and deriving implementations in terms of specific coordination mechanisms.

A number of paradigms for unifying interaction in heterogeneous systems have been studied in [1,2,3]. In these works, unification is achieved by reduction to a common low-level semantic model. Interaction mechanisms and their properties are not studied independently of behavior.

We propose a new *causal semantics* for the *Algebra of Connectors* studied in [4]. This algebra considers connectors as the basic concept for modelling coordination between components.

The term "connector" is widely used in the component frameworks literature with a number of different interpretations. In general, connectors have two main aspects: in the *data flow* setting, connectors define the way data is transferred between components; alternatively, in what we call *control flow* setting, connectors rather define synchronization constraints leaving aside or completely abstracting the data flow.

Control flow connectors are often specified in an operational setting, usually a process algebra. In [5], a process algebra is used to define an *architectural type* as a set of component/connector instances related by a set of attachments among their interactions. In [6], a connector is defined as a set of processes, with one process for each role of the connector, plus one process for the "glue" that describes how all the roles are bound together. A similar approach is developed by J. Fiadeiro and his colleagues in a categorical framework for CommUnity [7].

All the above models define connectors that can exhibit complex behavior. That is, computation is not limited to the components, but can be partly performed in the connectors. In [8], an algebra of connectors is developed that allows, in particular, an algebraic translation of the categorical approach used in CommUnity. This algebra allows *stateless* connectors to be constructed from a number of basic ones.

Reo [9,10] is a channel-based exogenous coordination model, which presents both data and control flow aspects. It uses connectors compositionally built out of different types of channels formalized in data-stream semantics and interconnected by using nodes. The connectors in Reo allow computation, but it is limited to the underlying channels. The nodes of connectors realize coordination between these channels.

Our approach is closest to that of [8], as it focuses on stateless connectors in a control flow setting. We consider connectors as relations between ports with synchronization types, which allows one to describe complex coordination patterns with an extremely small set of basic primitives. Thus, our main subject,

in this paper, is structuring interactions among components. Although, in the composed system, data exchange can take place upon synchronisation, this is out of the scope of this paper.

In a previous paper [4], we studied an *interaction semantics* for the Algebra of Connectors $\mathcal{AC}(P)$, which is used to model interactions in the BIP component framework [11,12]. Terms of $\mathcal{AC}(P)$ are *connectors*. The interaction semantics defines the meaning of a connector as the set of the interactions it allows.

$\mathcal{AC}(P)$ is defined from a set P of ports. Its terms represent sets of interactions which are non empty sets of ports. Within a connector, an interaction can take place in two situations: either an interaction is fired when all involved ports are ready to participate (strong synchronization), or some subset of ports triggers the interaction without waiting for other ports. Thus, connectors are generated from the ports of P by using a binary *fusion* operator and a unary *typing* operator. Typing associates with terms (ports or connectors) synchronization types: *trigger* or *synchron*. Trigger and synchron terms form connectors as described below.

A *Simple* (or *flat*) connector is an expression of the form $p'_1 \dots p'_k p_{k+1} \dots p_n$, where primed ports p'_i are triggers, and unprimed ports p_j are synchrons. For a flat connector involving the set of ports $\{p_1, \dots, p_n\}$, interaction semantics defines the set of its interactions by the following rule: *an interaction is any non empty subset of* $\{p_1, \dots, p_n\}$ *which contains some port that is a trigger; otherwise (if all the ports are synchrons), the only possible interaction is the maximal one, that is* $p_1 \dots p_n$. As usual, we abbreviate $\{p_1, \dots, p_n\}$ to $p_1 \dots p_n$.

In particular, two basic synchronization protocols can be modelled naturally: 1) *rendezvous*, when all the related ports are synchrons, and the only possible interaction is the maximal one containing all ports of the connector; 2) *broadcast*, when the port that initiates the interaction is a trigger, all other ports are synchrons, and possible interactions are those containing the trigger. Connectors, representing these two protocols for ports s, r_1,r_2, and r_3, are shown in Fig. 1(a, b). Triangles represent triggers, and circles represent synchrons.

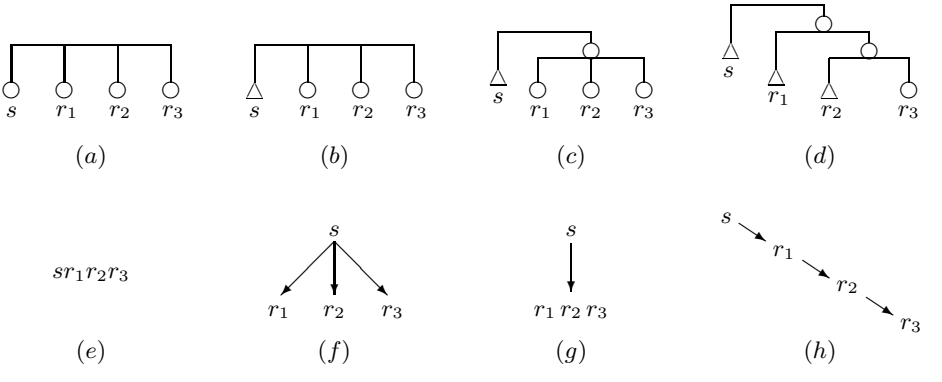

Fig. 1. Connectors and causal trees representing a rendezvous (a, e), a broadcast (b, f), an atomic broadcast (c, g), and a causal chain (d, h)

Hierarchical connectors are expressions composed of typed ports and/or typed sub-connectors. Fig. $1(c)$ shows a connector realizing an atomic broadcast from a port s to ports r_1, r_2, and r_3. The port s is a trigger, and r_1, r_2, r_3 are strongly synchronized in a sub-connector, itself typed as a synchron. The corresponding $\mathcal{AC}(P)$ term is $s'[r_1r_2r_3]$, and the possible interactions are: s and $sr_1r_2r_3$. Here the term in brackets $[\cdot]$ is a sub-connector typed as a synchron. Primed brackets $[\cdot]'$ denote a sub-connector typed as a trigger. The connector shown in Fig. $1(d)$ is a causal chain of interactions initiated by the port s. The corresponding $\mathcal{AC}(P)$ term is $s'[r_1'[r_2'r_3]]$, and the possible interactions are s, sr_1, sr_1r_2, $sr_1r_2r_3$: a trigger s alone or combined with some interaction from the sub-connector $r_1'[r_2'r_3]$, itself a shorter causal chain.

As shown in the above examples, interaction semantics reduces a connector into the set of its interactions. This leads to exponentially complex representations. Furthermore, it blurs the distinction between rendezvous and broadcast as each interaction of a broadcast can be realized by a rendezvous. In [4], we have shown that this also has deep consequences on the induced semantic equivalence: broadcasts may be equivalent to sets of rendezvous but they are not congruent.

The deficiencies of interaction semantics have motivated the investigation of a new *causal* semantics for a subset of connectors of $\mathcal{AC}(P)$, formalized as the Algebra of Casual Trees $\mathcal{CT}(P)$. This semantics distinguishes broadcast and rendezvous by explicitly modelling the causal dependency relation between triggers and synchrons in broadcasts. The terms of $\mathcal{CT}(P)$ represent sets of interactions, generated from atomic interactions on the set of ports P, by using two operators:

- A *causality* operator $\rightarrow$ which defines the causal relationship. The term $a_1 \rightarrow a_2 \rightarrow a_3$ is a causal chain meaning that interaction a_1 may trigger interaction a_2 which may trigger interaction a_3. The possible interactions for this chain are a_1, a_1a_2, $a_1a_2a_3$.
- An associative and commutative *parallel composition* operator $\oplus$. A causal tree can be considered as the parallel composition of all its causal chains. For instance, the term $a_1 \rightarrow (a_2 \oplus a_3)$ is equivalent to $(a_1 \rightarrow a_2) \oplus (a_1 \rightarrow a_3)$ (both describing the set of four interactions: a_1, a_1a_2, a_1a_3, and $a_1a_2a_3$).

Terms of $\mathcal{CT}(P)$ are naturally represented as sets of causal trees where $\rightarrow$ corresponds to the parent/son relation. Fig. $1(e-h)$ shows the causal trees for the four connectors discussed above.

The main results of the paper are the following:

- We define causal semantics for $\mathcal{AC}(P)$ in terms of causality trees, as a function $\mathcal{AC}(P) \rightarrow \mathcal{CT}(P)$. Causal semantics is sound with respect to interaction semantics. An important result is that the algebra of causal trees $\mathcal{CT}(P)$ is isomorphic to classes of *causal connectors* $\mathcal{AC}_c(P)$ and *causal sets of interactions* $\mathcal{AI}_c(P)$. A causal set of interactions is closed under synchronization. A causal connector has a trigger in each sub-connector (including itself). We have shown that the equivalence and the congruence of $\mathcal{AC}(P)$ coincide for the set of causal connectors $\mathcal{AC}_c(P)$.

– We define for causal trees, $\mathcal{CT}(P)$ a boolean representation by using *causal rules*. Terms are represented by boolean expressions on P. The boolean valuation of port p is interpreted as the presence/absence of a port in an interaction. This representation is used for their symbolic manipulation and simplification as well as for performing boolean operations on connectors. It is applied for the efficient implementation of BIP, in particular, to compute the possible interactions for a given state.

The paper is structured as follows. Sect. 2 provides a succinct presentation of the basic semantic model for BIP and in particular, its composition parameterized by interactions. Sect. 3 presents the Algebra of Connectors, $\mathcal{AC}(P)$, and its global interaction semantics. Sect. 4 presents a semantics for $\mathcal{AC}(P)$ in terms of the Algebra of Causal trees, $\mathcal{CT}(P)$. It also shows how a boolean representation for connectors can be obtained from their representation as causal trees.

2 The BIP Component Framework

BIP is a component framework for constructing systems by superposing three layers of modelling: Behavior, Interaction, and Priority. The lower layer consists of a set of atomic components representing transition systems. The second layer models interactions between components, specified by connectors. These are relations between ports equipped with synchronization types. Priorities are used to enforce scheduling policies applied to interactions of the second layer.

The BIP component framework has been implemented in a language and a tool-set. The BIP language offers primitives and constructs for modelling and composing layered components. Atomic components are communicating automata extended with C functions and data. Their transitions are labelled with sets of communication ports. The BIP language also allows composition of components parameterized by sets of interactions as well as application of priorities.

The BIP tool-set includes an editor and a compiler for generating C++ code from BIP programs. The generated C++ code can be compiled for execution on a dedicated platform (see [11,13]).

We provide a succinct formalization of the BIP component model focusing on the operational semantics of component interaction.

Definition 1. *For a set of ports P, an* interaction *is a non-empty subset $a \subseteq P$ of ports. To simplify notation we represent an interaction $\{p_1, p_2, \ldots, p_n\}$ as $p_1 p_2 \ldots p_n$.*

Definition 2. *A 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 labelled by an interaction.*

For any pair of states $q, q' \in Q$ and interaction $a \in 2^P$, we write $q \xrightarrow{a} q'$, iff $(q, a, q') \in \rightarrow$. When the interaction is irrelevant, we simply write $q \rightarrow q'$.

An interaction a is enabled *in state q, denoted $q \xrightarrow{a}$, iff there exists $q' \in Q$ such that $q \xrightarrow{a} q'$.*

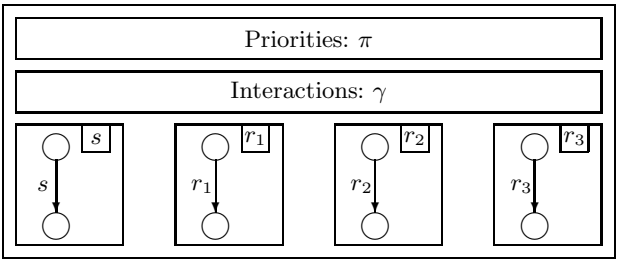

Fig. 2. A system with four atomic components

In BIP, a system can be obtained as the composition of n components, each modelled by a transition system $B_i = (Q_i, P_i, \rightarrow_i)$, for $i \in [1, n]$, such that their sets of ports are pairwise disjoint: for $i, j \in [1, n]$ $(i \neq j)$, we have $P_i \cap P_j = \emptyset$. We take $P = \bigcup_{i=1}^{n} P_i$, the set of all ports in the system.

The *composition* of components $\{B_i\}_{i=1}^{n}$, parameterized by a set of interactions $\gamma \subseteq 2^P$ is the transition system $B = (Q, P, \rightarrow_\gamma)$, where $Q = \bigotimes_{i=1}^{n} Q_i$ and $\rightarrow_\gamma$ is the least set of transitions satisfying the rule

$$\frac{a \in \gamma \quad \wedge \quad \forall i \in [1, n],\ (a \cap P_i \neq \emptyset \Rightarrow q_i \overset{a \cap P_i}{\rightarrow}_i q_i')}{(q_1, \ldots, q_n) \overset{a}{\rightarrow}_\gamma (q_1', \ldots, q_n')}, \tag{1}$$

where $q_i = q_i'$ for all $i \in [1, n]$ such that $a \cap P_i = \emptyset$. We write $B = \gamma(B_1 \ldots, B_n)$.

Notice that an interaction $a \in \gamma$ is enabled in $\gamma(B_1, \ldots, B_n)$, only if, for each $i \in [1, n]$, the interaction $a \cap P_i$ is enabled in B_i; the states of components that do not participate in the interaction remain unchanged.

Several distinct interactions can be enabled at the same time, thus introducing non-determinism in the product behavior. This can be restricted by means of priorities [4,13]. Here, we omit formal definition of priorities, as we only use the *maximal progress* rule, which is implicitly assumed throught the paper: whenever two interactions, a and a', such that $a \subsetneq a'$, are possible, we always choose a'.

Example 1 (Sender/Receivers). Fig. 2 shows a component $\gamma(S, R_1, R_2, R_3)$ obtained by composition of four atomic components: a sender, S, and three receivers, R_1, R_2, R_3 with a set of interactions γ. The sender has a port s for sending messages, and each receiver has a port r_i $(i = 1, 2, 3)$ for receiving them. The following table specifies γ for four different interaction schemes.

Interaction scheme	Interactions
Rendezvous	$sr_1r_2r_3$
Broadcast	s, sr_1, sr_2, sr_3, sr_1r_2, sr_1r_3, sr_2r_3, $sr_1r_2r_3$
Atomic Broadcast	s, $sr_1r_2r_3$
Causal Chain	s, sr_1, sr_1r_2, $sr_1r_2r_3$

Rendezvous means strong synchronization between S and all R_i. This is specified by a single interaction involving all the ports. This interaction can occur only if all the components are in states enabling transitions labelled respectively by s, r_1, r_2, r_3.

Broadcast means weak synchronization, that is a synchronization involving S and any (possibly empty) subset of R_i. This is specified by the set of all interactions containing s. These interactions can occur only if S is in a state enabling s. Each R_i participates in the interaction only if it is in a state enabling r_i.

Atomic broadcast means that either a message is received by all R_i, or by none. Two interactions are possible: s, when at least one of the receiving ports is not enabled, and the interaction $sr_1r_2r_3$, corresponding to strong synchronization.

Causal chain means that for a message to be received by R_i it has to be received by all R_j, for $j < i$. This interaction scheme is common in reactive systems.

Example 2 (Modulo-8 counter). Fig. 3 shows a model for the Modulo-8 counter presented in [14], obtained by composition of three Modulo-2 counter components. Ports p, r, and t correspond to inputs, whereas q, s, and u correspond to outputs. It can be easily verified that the interactions pqr, $pqrst$, and $pqrstu$ happen, respectively, every second, fourth, and eighth occurrence of an input interaction through the port p.

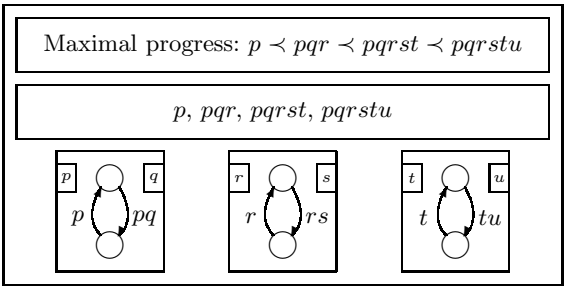

Fig. 3. Modulo-8 counter

Notice that the composition operator can express usual parallel composition operators [4], such as the ones used in CSP [15] and CCS [16]. By enforcing maximal progress, priorities allow to express broadcast.

3 The Algebra of Connectors

In this section, we introduce the *algebra of connectors* $\mathcal{AC}(P)$, which formalizes the concept of connector, supported by the BIP language [11]. For the sake of

simplicity, we consider the subset of terms of $\mathcal{AC}(P)$ that do not involve union, that is the subset of *monomial connectors* (cf. [4]).

3.1 The Algebra of Interactions

We introduce the *algebra of interactions* $\mathcal{AI}(P)$, used to define the interaction semantics of $\mathcal{AC}(P)$.

Let P be a set of ports, such that $0, 1 \notin P$. Recall (Def. 1) that an *interaction* is a non-empty subset $a \subseteq P$. We abbreviate $\{p_1, p_2, \ldots, p_n\}$ to $p_1\, p_2 \ldots p_n$.

Syntax. The algebra of interactions $\mathcal{AI}(P)$, is defined by the following syntax

$$x ::= 0 \mid 1 \mid p \in P \mid x \cdot x \mid x + x\,, \tag{2}$$

where $+$ and $\cdot$ are binary operators, respectively called *union* and *synchronization*. Synchronization binds stronger than union.

Axioms

1. Union $+$ is idempotent, associative, commutative, and has an identity element 0;
2. Synchronization $\cdot$ is associative, commutative, has an identity element 1, and an absorbing element 0; synchronization distributes over union. Furthermore, it is idempotent for monomial terms (terms without $+$).

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

$$\begin{aligned} &\|0\| = \emptyset, \quad \|1\| = \{\emptyset\}, \quad \|p\| = \Big\{\{p\}\Big\}, \\ &\|x_1 + x_2\| = \|x_1\| \cup \|x_2\|, \\ &\|x_1 \cdot x_2\| = \Big\{a_1 \cup a_2 \,\Big|\, a_1 \in \|x_1\|,\ a_2 \in \|x_2\|\Big\}, \end{aligned} \tag{3}$$

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

Remark 1. In Def. 1, interactions are non-empty subsets of P, i.e. $a \in 2^P \setminus \{\emptyset\}$. In the following, we lift this restriction. Thus, $1 \in \mathcal{AI}(P)$ represents a singleton subset $\{\emptyset\} \subseteq 2^P$ (cf. (3)). The term $0 \in \mathcal{AI}(P)$ corresponds to an empty subset of 2^P and does not represent any interaction. Thus interactions correspond to non-zero monomial terms of $\mathcal{AI}(P)$.

Proposition 1 ([4]). *The axiomatization of $\mathcal{AI}(P)$ is sound and complete, that is, for any $x, y \in \mathcal{AI}(P)$, $x = y$ iff $\|x\| = \|y\|$.*

Example 3 (Sender/Receiver continued). The second column of Table 1 shows the representation in $\mathcal{AI}(P)$ for the four interaction schemes of Ex. 1.

Table 1. $\mathcal{AI}(P)$, $\mathcal{AC}(P)$, and $\mathcal{CT}(P)$ representations of four basic interaction schemes

	$\mathcal{AI}(P)$	$\mathcal{AC}(P)$	$\mathcal{CT}(P)$
Rendezvous	$s\, r_1\, r_2\, r_3$	$s\, r_1\, r_2\, r_3$	$s\, r_1\, r_2\, r_3$
Broadcast	$s\,(1 + r_1)$ $(1 + r_2)\,(1 + r_3)$	$s'\, r_1\, r_2\, r_3$	$s \to (r_1 \oplus r_2 \oplus r_3)$
Atomic Broadcast	$s\,(1 + r_1\, r_2\, r_3)$	$s'\,[r_1\, r_2\, r_3]$	$s \to r_1\, r_2\, r_3$
Causal Chain	$s\,(1 + r_1\,(1 +$ $+ r_2\,(1 + r_3)))$	$s'\,[r_1'\,[r_2'\, r_3]]$	$s \to r_1 \to r_2 \to r_3$

3.2 Correspondence with Boolean Functions

$\mathcal{AI}(P)$ can be bijectively mapped to the free boolean algebra $\mathbb{B}[P]$ generated by P. We define a mapping $\beta : \mathcal{AI}(P) \to \mathbb{B}[P]$ by setting:

$$\beta(0) \;=\; \mathit{false}\,, \qquad \beta(x + y) \;=\; \beta(x) \;\vee\; \beta(y)\,,$$
$$\beta(1) \;=\; \bigwedge_{p \in P} \overline{p}\,, \qquad \beta(p_{i_1} \ldots p_{i_k}) \;=\; \bigwedge_{j=1}^{k} p_{i_j} \;\cdot\; \bigwedge_{i \neq i_j} \overline{p_i}\,,$$

for $p_{i_1}, \ldots, p_{i_k} \in P$, and $x, y \in \mathcal{AI}(P)$, where in the right-hand side the elements of P are considered to be boolean variables. We denote by *false* (resp. *true*) the least (resp. greatest) element in $\mathbb{B}[P]$. For example, consider the correspondence table for $P = \{p, q\}$ shown in Table 2.

Table 2. Correspondence between $\mathcal{AI}(\{p, q\})$ and boolean functions with two variables

$\mathcal{AI}(P)$	$\mathbb{B}[P]$
0	*false*
$1 \quad p \quad q \quad pq$	$\overline{p}\,\overline{q} \quad p\overline{q} \quad \overline{p}q \quad pq$
$p+1 \quad q+1 \quad pq+1 \quad p+q \quad p+pq \quad q+pq$	$\overline{q} \quad \overline{p} \quad \overline{p}\,\overline{q} \vee pq \quad p\overline{q} \vee \overline{p}q \quad p \quad q$
$p+q+1 \quad pq+p+1 \quad pq+q+1 \quad pq+p+q$	$\overline{p} \vee \overline{q} \quad p \vee \overline{q} \quad \overline{p} \vee q \quad p \vee q$
$pq+p+q+1$	*true*

The mapping β is an order isomorphism, and consequently techniques specific to boolean algebras can be applied to the boolean representation of $\mathcal{AI}(P)$ (e.g. BDDs).

Any interaction $a \in 2^P$ defines a valuation on P with, for each $p \in P$, $p = \mathit{true}$ iff $p \in a$. Notice that the constant valuation *false* is associated to the interaction 1, which corresponds to the empty set of ports $\emptyset \in 2^P$ (cf. Rem 1 and Table 2).

Definition 3. *An interaction $a \in 2^P$* satisfies *a formula $R \in \mathbb{B}[P]$ (denoted $a \models R$) iff the corresponding boolean valuation satisfies R. A term $x \in \mathcal{AI}(P)$ satisfies R (denoted $x \models R$) iff all interactions belonging to x satisfy R, that is*

$$x \models R \quad \stackrel{def}{\iff} \quad \forall a \in \|x\|, \quad a \models R\,.$$

Remark 2. Let R_1 and R_2 be two equivalent formulae. They are satisfied by the same interactions:

$$\forall a \in 2^P, \quad a \models R_1 \iff a \models R_2\,.$$

Proposition 2. *An interaction belongs to the set described by an expression $x \in \mathcal{AI}(P)$ if and only if it satisfies $\beta(x)$, that is*

$$\|x\| = \left\{ a \in 2^P \,\middle|\, a \models \beta(x) \right\}. \tag{4}$$

Remark 3. As $\|0\| = \emptyset$, according to Def. 3, it satisfies all formulae in $\mathbb{B}[P]$, and in particular $0 \models \mathit{false}$. This is the only term in $\mathcal{AI}(P)$ satisfying the constant predicate *false*. Recall (Rem 1) that $0 \notin 2^P$.

The advantage of $\mathcal{AI}(P)$ over its boolean representation is that it provides a more intuitive description of sets of interactions. For example, the term $p{+}pq \in \mathcal{AI}(P)$ represents the set of interactions $\{p, pq\}$ for any set of ports P containing p and q. The boolean representation of $p + pq$ depends on P: if $P = \{p, q\}$ then $\beta(p + pq) = p$, whereas if $P = \{p, q, r, s\}$ then $\beta(p + pq) = p\,\overline{r}\,\overline{s}$.

Synchronization of two interactions in $\mathcal{AI}(P)$ is by simple concatenation, whereas for their boolean representation there is no simple context-independent composition rule.

Example 4. Let $P = \{p, q, r, s\}$. The representation of p is $\beta(p) = p\,\overline{q}\,\overline{r}\,\overline{s}$, the representation of q is $\beta(q) = \overline{p}\,q\,\overline{r}\,\overline{s}$, and the representation $\beta(pq) = p\,q\,\overline{r}\,\overline{s}$ of the synchronization pq is obtained by combining the "positive" variables p and q from $\beta(p)$ and $\beta(q)$ respectively with the "negative" variables $\overline{r}$ and $\overline{s}$ belonging to both.

To formalize the above example, let $x, y \in \mathcal{AI}(P)$ be two terms represented respectively by boolean functions

$$\beta(x) = \bigwedge_{p \in P_x} p \cdot \bigwedge_{q \in Q_x} \overline{q}\,, \quad \text{and} \quad \beta(y) = \bigwedge_{p \in P_y} p \cdot \bigwedge_{q \in Q_y} \overline{q}\,, \tag{5}$$

where $P_x, P_y \subseteq P$ and $Q_x, Q_y \subseteq P$ are respectively the sets of positive and negative variables in $\beta(x)$ and $\beta(y)$, then the synchronization xy corresponds to

$$\beta(xy) = \bigwedge_{p \in P_x \cup P_y} p \cdot \bigwedge_{q \in Q_x \cap Q_y} \overline{q} \tag{6}$$

In the general case, when the boolean representations of x and y contain multiple summands of the form (5), the representation of their synchronization

xy can be obtained by applying the above operation pairwise to the summands of $\beta(x)$ and $\beta(y)$ and taking the sum of the obtained conjunctions.

On the other hand, the interactions belonging to the intersection of x and y, that is to $\|x\| \cap \|y\|$, are clearly characterized by $\beta(x) \wedge \beta(y)$.

Thus, we have a correspondence between $\mathcal{AI}(P)$ equipped with union, synchronization, and intersection, and $\mathbb{B}[P]$ equipped with disjunction, the operation above described by (5) and (6), and conjunction.

3.3 Syntax and Interaction Semantics for $\mathcal{AC}(P)$

Syntax. Let P be a set of ports, such that $0, 1 \notin P$. The syntax of the algebra of connectors, $\mathcal{AC}(P)$, is defined by

$$\begin{aligned} s &::= [0] \mid [1] \mid [p] \mid [x] \quad (\textit{synchrons}) \\ t &::= [0]' \mid [1]' \mid [p]' \mid [x]' \ (\textit{triggers}) \\ x &::= s \mid t \mid x \cdot x\,, \end{aligned} \tag{7}$$

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

Fusion is a generalization of synchronization in $\mathcal{AI}(P)$. Typing is used to form connectors: $[\cdot]'$ defines *triggers* (which can initiate an interaction), and $[\cdot]$ defines *synchrons* (which need synchronization with other ports).

Definition 4. *In a system with a set of ports P,* connectors *are elements of $\mathcal{AC}(P)$.*

Notation. We write $[x]^\alpha$, for $\alpha \in \{0, 1\}$, to denote a typed connector. When $\alpha = 0$, the connector is a synchron, otherwise it is a trigger.

In order to simplify notation, we will omit brackets on 0, 1, and ports $p \in P$, as well as '$\cdot$' for the fusion operator.

The algebraic structure of $\mathcal{AC}(P)$ inherits most of the axioms of $\mathcal{AI}(P)$.

Axioms

1. Fusion $\cdot$ is associative, commutative, distributive, idempotent, and has an identity element [1].
2. Typing satisfies the following axioms, for $x, y, z \in \mathcal{AC}(P)$ and $\alpha, \beta \in \{0, 1\}$:
   (a) $[0]' = [0]$,
   (b) $\left[[x]^\alpha\right]^\beta = [x]^\beta$.

Semantics. The semantics of $\mathcal{AC}(P)$ is given by the function $|\cdot| : \mathcal{AC}(P) \to \mathcal{AI}(P)$, defined by the rules (we use the product symbol '$\prod$' to denote fusion)

$$|p| = p\,, \tag{8}$$

$$\left|\prod_{i=1}^{n} [x_i]\right| = \prod_{i=1}^{n} |x_i|\,, \tag{9}$$

$$\left|\prod_{i=1}^{n}[x_i]' \cdot \prod_{j=1}^{m}[y_j]\right| = \sum_{i=1}^{n}|x_i| \prod_{k\neq i}\Big(1+|x_k|\Big)\prod_{j=1}^{m}\Big(1+|y_j|\Big), \tag{10}$$

for $p \in P \cup \{0,1\}$ and $x, x_1, \ldots, x_n, y_1, \ldots, y_m \in \mathcal{AC}(P)$. The sum in (10) is the union operator of $\mathcal{AI}(P)$.

Example 5. Consider a system consisting of two Senders with ports s_1, s_2, and three Receivers with ports r_1, r_2, r_3. The meaning of $s_1'\, s_2'\, r_1\, [r_2\, r_3]$ is

$$\begin{aligned}
&|s_1'\, s_2'\, r_1\, [r_2\, r_3]| \;=\\
&\overset{(10)}{=} |s_1|\,(1+|s_2|)\,(1+|r_1|)\,(1+|r_2\, r_3|) \;+\; |s_2|\,(1+|s_1|)\,(1+|r_1|)\,(1+|r_2\, r_3|)\\
&\overset{(9)}{=} \Big(|s_1|\,(1+|s_2|) \;+\; |s_2|\,(1+|s_1|)\Big)\,(1+|r_1|)\,(1+|r_2|\,|r_3|)\\
&\overset{(8)}{=} \Big(s_1\,(1+s_2) \;+\; s_2\,(1+s_1)\Big)\,(1+r_1)\,(1+r_2\, r_3)\,,
\end{aligned}$$

which corresponds to the set of the interactions containing at least one of s_1 and s_2, and possibly r_1 and a synchronization of both r_2 and r_3.

Proposition 3 ([4]). *The axiomatization of $\mathcal{AC}(P)$ is sound, that is, for $x, y \in \mathcal{AC}(P)$, the equality $x = y$ implies $|x| = |y|$.*

Example 6 (Sender/Receiver continued). The third column of Table 1 shows the connectors for the four interaction schemes of Ex. 1.

Notice that $\mathcal{AC}(P)$ allows compact representation of interactions and, moreover, explicitly captures the difference between broadcast and rendezvous. The typing operator induces a hierarchical structure.

Example 7 (Modulo-8 counter continued). In the model shown in Fig. 4, the causal chain pattern is applied to connectors p, $q\, r$, $s\, t$, and u. Interactions are modelled by a single structured connector $p'\,\Big[[q\, r]'\,[[s\, t]'\, u]\Big]$:

$$\Big|p'\;\Big[[q\, r]'\,\Big[[s\, t]'\, u\Big]\Big]\Big| = p + p\, q\, r + p\, q\, r\, s\, t + p\, q\, r\, s\, t\, u\,.$$

These are exactly the interactions of the Modulo-8 counter of Fig. 3.

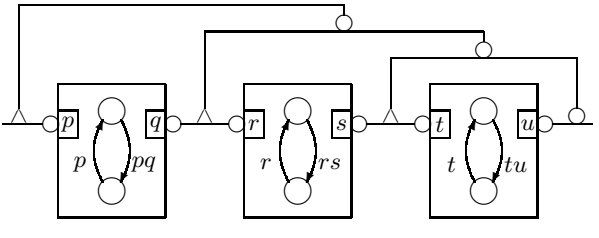

Fig. 4. Modulo-8 counter

Definition 5. *Two connectors $x, y \in \mathcal{AC}(P)$ are* equivalent *(denoted $x \simeq y$), iff they have the same sets of interactions, i.e. $x \simeq y$ if and only if $|x| = |y|$.*

Notice that, in general, two equivalent terms are not congruent. For example, $p' \simeq p$, but $p'q \simeq p + pq \not\simeq pq$, for $p, q \in P$. Furthermore, the following terms are equivalent, but not congruent: pqr, $p[qr]$, and $[pq]r$, as different sets of interactions are obtained, when these terms are fused with a trigger. For instance, $s'[pq]r \simeq s + spq + sr + spqr$, whereas $s'p[qr] \simeq s + sp + sqr + spqr$.

Definition 6. *We denote by '$\cong$' the largest congruence relation contained in $\simeq$, that is the largest relation satisfying*

$$x \cong y \quad \Longrightarrow \quad \forall E \in \mathcal{AC}(P \cup \{z\}),\ E(x/z) \simeq E(y/z)\,, \tag{11}$$

where $x, y \in \mathcal{AC}(P)$, $z \notin P$, $E(x/z)$, and (resp. $E(y/z)$) denotes the expression obtained from E by replacing all occurrences of z by x (resp. y).

Theorem 1 ([4]). *For $x, y \in \mathcal{AC}(P)$, we have $x \cong y$ iff the three following conditions hold simultaneously*

1. $x \simeq y$,
2. $x \cdot 1' \simeq y \cdot 1'$,
3. $\#x > 0 \Leftrightarrow \#y > 0$,

where, for $x = \prod_{i=1}^{n} [x_i]^{\alpha_i}$, we denote by $\#x$ the number of triggers in this fusion, that is $\#x \stackrel{def}{=} \#\{i \in [1, n] \,|\, \alpha_i = 1\}$.

Corollary 1. *For $x, y \in \mathcal{AC}(P)$, holds* $[x]'\,[y]' \ \cong\ \Big[[x]'\,[y]'\Big]'$.

4 Causal Semantics for Connectors

In this section, we propose a new *causal* semantics for $\mathcal{AC}(P)$ connectors. This allows us to address two important points:

1. (*Congruence*). As we have shown in the previous section, the equivalence relation $\simeq$ on $\mathcal{AC}(P)$ is not a congruence. The causal semantics allows us to define a subset $\mathcal{AC}_c(P) \subsetneq \mathcal{AC}(P)$ of *causal connectors* such that a) every equivalence class on $\mathcal{AC}(P)$ has a representative in $\mathcal{AC}_c(P)$; and b) the equivalence $\simeq$ and congruence $\cong$ relations coincide on $\mathcal{AC}_c(P)$.
2. (*Boolean representation*). In [4], we showed that efficient computation of boolean operations (e.g. intersection, complementation) is crucial for efficient implementation of some classes of systems, e.g. synchronous systems. In this section, we present a method for computing boolean representations for $\mathcal{AC}(P)$ connectors through a translation into the algebra of causal trees $\mathcal{CT}(P)$. The terms of the latter have a natural boolean representation as sets of causal rules (implications). This boolean representation avoids complex enumeration of the interactions of connectors entailed by the method in Sect. 3.2.

The key idea for causal semantics is to render explicit the causal relations between different parts of the connector. In a fusion of typed connectors, triggers are mutually independent, and can be considered *parallel* to each other. Synchrons participate in an interaction only if it is initiated by a trigger. This introduces a causal relation: the trigger is a *cause* that can provoke an *effect*, which is the participation of a synchron in an interaction.

There are essentially three possibilities for connectors involving ports p and q:

1. A strong synchronization pq.
2. One trigger $p'q$, i.e. p is the cause of an interaction and q a potential effect, which we will denote in the following by $p \to q$.
3. Two triggers $p'q'$, i.e. p and q are independent (parallel), which we will denote in the following by $p \oplus q$.

This can be further extended to chains of causal relations between interactions. For example, $(p \oplus q) \to rs \to t$ corresponds to the connector $p'q'\,[\,[rs]'\,t]$. It means that any combination of p and q (i.e. p, q, or pq) can trigger an interaction in which both r and s may participate (thus, the corresponding interactions are p, q, pq, prs, qrs, and $pqrs$). Moreover, if r and s participate then t may do so, which adds the interactions $prst$, $qrst$, and $pqrst$.

Causal trees constructed with these two operators provide a compact and clear representation for connectors that shows explicitly the atomic interactions (p, q, rs, and t in the above example) and the dependencies between them. They also allow to exhibit the boolean *causal rules*, which define the necessary conditions for a given port to participate in an interaction. Intuitively, this corresponds to expressing arrows in the causal trees by implications.

A causal rule is a boolean formula over P, which has the form $p \Rightarrow \bigvee_{i=1}^{n} a_i$, where p is a port and a_i are interactions that can provoke p. Thus, in the above example, the causal rule for the port r is $r \Rightarrow ps \vee qs$, which means that for the port r to participate in an interaction of this connector, it is necessary that this interaction contain either ps or qs.

A set of causal rules uniquely describes the set of interactions that satisfy it (cf. Sect. 3.2), which provides a simple and efficient way for computing boolean representations for connectors by transforming them first into causal trees and then into a conjunction of the associated causal rules.

In the following sub-sections we formalize these ideas.

4.1 Causal Trees

Syntax. Let P be a set of ports such that $0, 1 \notin P$. The syntax of the *algebra of causal trees*, $\mathcal{CT}(P)$, is defined by

$$t ::= a \,|\, t \to t \,|\, t \oplus t\,, \tag{12}$$

where $a \in \mathcal{AI}(P)$ is 0, 1, or an interaction from 2^P, and $\to$ and $\oplus$ are respectively the *causality* and the *parallel composition* operators. Causality binds stronger than parallel composition.

Although the causality operator is not associative, for $t_1, \ldots, t_n \in \mathcal{CT}(P)$, we abbreviate $t_1 \to (t_2 \to (\ldots \to t_n)\ldots))$ to $t_1 \to t_2 \to \ldots \to t_n$. We call this construction a *causal chain*.

Axioms

1. Parallel composition, $\oplus$, is associative, commutative, idempotent, and its identity element is 0.
2. Causality, $\to$, satisfies the following axioms:
   (a) $t \to 1 = t$,
   (b) $t_1 \to (1 \to t_2) = t_1 \to t_2$,
   (c) $t \to 0 = t$,
   (d) $0 \to t = 0$.
3. The following axioms relate the two operators:
   (a) $(t_1 \to t_2) \to t_3 = t_1 \to (t_2 \oplus t_3)$,
   (b) $t_1 \to (t_2 \oplus t_3) = t_1 \to t_2 \ \oplus \ t_1 \to t_3$,
   (c) $(t_1 \oplus t_2) \to t_3 = t_1 \to t_3 \ \oplus \ t_2 \to t_3$.

Semantics. The *interaction semantics* of $\mathcal{CT}(P)$ is given by the function $|\cdot| : \mathcal{CT}(P) \to \mathcal{AI}(P)$, defined by the rules

$$|a| = a\,, \tag{13}$$

$$|a \to t| = a\Big(1 + |t|\Big)\,, \tag{14}$$

$$|t_1 \oplus t_2| = |t_1| + |t_2| + |t_1|\,|t_2|\,, \tag{15}$$

where a is an interaction of 2^P, and $t, t_1, t_2 \in \mathcal{CT}(P)$, and the rules induced by axioms (3a) and (3c). The set semantics of a causal tree $t \in \mathcal{CT}(P)$ is obtained by applying the semantic function $\|\cdot\| : \mathcal{AI}(P) \to 2^{2^P}$ to $|t|$. We denote $\|t\| \stackrel{def}{=} \| \, |t| \, \|$.

Example 8 (Causal chain). Consider interactions $a_1, \ldots, a_n \in 2^P$ and a causal chain $a_1 \to a_2 \to \ldots \to a_n$. Iterating rule (14), we then have

$$\begin{aligned} |a_1 \to a_2 \to \ldots \to a_n| &= a_1\Big(1 + |a_2 \to \ldots \to a_n|\Big) \\ &= a_1 + a_1a_2\Big(1 + |a_3 \to \ldots \to a_n|\Big) \\ &= \ldots \\ &= a_1 \ + \ a_1a_2 \ + \ldots + \ a_1a_2\ldots a_n\,. \end{aligned}$$

Proposition 4. *The axiomatization of $\mathcal{CT}(P)$ is sound with respect to the semantic equivalence, i.e. for $t_1, t_2 \in \mathcal{CT}(P)$, $t_1 = t_2$ implies $|t_1| = |t_2|$.*

Remark 4. According to the axioms of $\mathcal{CT}(P)$ any causal tree can be represented as a parallel composition of its causal chains (see Fig. 5). Thus an interaction belonging to a causal tree is a synchronization of any number of prefixes (cf. Ex. 8) of the corresponding causal chains, i.e. branches of this tree.

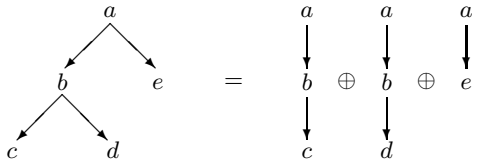

Fig. 5. A causal tree is the parallel composition of its causal chains

Example 9 (Sender/Receiver continued). The fourth column of Table 1 shows the causal trees for the four interaction schemes of Ex. 1.

Example 10 (Modulo-8 counter continued). The connector applied to the three Modulo-2 counter components in Ex. 7 consists of a causal chain pattern applied to rendezvous connectors p, $q\,r$, $s\,t$, and u. Thus, the corresponding causal tree is clearly $p \to qr \to st \to u$. In general, the transformation of $\mathcal{AC}(P)$ connectors into causal trees is presented in the section below.

Definition 7. *Two causal trees* $t_1, t_2 \in \mathcal{CT}(P)$ *are* equivalent*, denoted* $t_1 \sim t_2$*, iff* $|t_1| = |t_2|$.

4.2 Correspondence with $\mathcal{AC}(P)$

In order to provide the transformation from $\mathcal{AC}(P)$ to $\mathcal{CT}(P)$, we introduce two helper functions $root : \mathcal{CT}(P) \to \mathcal{AI}(P)$ and $rest : \mathcal{CT}(P) \to \mathcal{CT}(P)$ defined by

$$\begin{aligned} root(a) &= a\,, & rest(a) &= 0 \\ root(a \to t) &= a\,, & rest(a \to t) &= t\,, \\ root(t_1 \oplus t_2) &= root(t_1) + root(t_2)\,, & rest(t_1 \oplus t_2) &= rest(t_1) \oplus rest(t_2)\,, \end{aligned}$$

for $a \in 2^P$ and $t, t_1, t_2 \in \mathcal{CT}(P)$. In general $t \neq root(t) \to rest(t)$. The equality holds only if t is of the form $a \to t_1$, for some interaction a and $t_1 \in \mathcal{CT}(P)$.

We define the function $\tau : \mathcal{AC}(P) \to \mathcal{CT}(P)$ associating a causal tree with a connector. By Cor. 1, any term can be rewritten to have at most one trigger. Therefore, the following three equations are sufficient to define τ:

$$\tau\left([x]' \prod_{i=1}^{n} [y_i]\right) = \tau(x) \to \bigoplus_{i=1}^{n} \tau(y_i)\,, \tag{16}$$

$$\tau\left(\prod_{i=1}^{n} [x_i]'\right) = \bigoplus_{i=1}^{n} \tau(x_i)\,, \tag{17}$$

$$\tau\left(\prod_{i=1}^{n} [y_i]\right) = \bigoplus_{j=1}^{m} \left(a_j \to \bigoplus_{i=1}^{n} rest\Big(\tau(y_i)\Big)\right)\,, \tag{18}$$

where $x, x_1, x_2, y_1, \ldots, y_n \in \mathcal{AC}(P)$, and, in (18), a_j are such that

$$\sum_{j=1}^{m} a_j = \prod_{i=1}^{n} root\Big(\tau(y_i)\Big)\,.$$

Example 11. Consider $P = \{p, q, r, s, t, u\}$ and $p'q'\Big[[r's][t'u]\Big] \in \mathcal{AC}(P)$. We have

$$\begin{aligned}\tau\left(p'q'\Big[[r's][t'u]\Big]\right) &= \tau\left(\Big[p'q'\Big]'\Big[[r's][t'u]\Big]\right)\\ &= \tau(p'q') \to \tau\Big([r's][t'u]\Big)\\ &= (p \oplus q) \to \Big(rt \to (s \oplus u)\Big)\,.\end{aligned}$$

We also define the function $\sigma : \mathcal{CT}(P) \to \mathcal{AC}(P)$, associating connectors to causal trees:

$$\sigma(a) = [a]\,, \tag{19}$$
$$\sigma(a \to t) = [a]'\,[\sigma(t)]\,, \tag{20}$$
$$\sigma(t_1 \oplus t_2) = [\sigma(t_1)]'\,[\sigma(t_2)]'\,. \tag{21}$$

Proposition 5. *The functions $\sigma : \mathcal{CT}(P) \to \mathcal{AC}(P)$ and $\tau : \mathcal{AC}(P) \to \mathcal{CT}(P)$, satisfy the following properties*

1. $\forall x \in \mathcal{AC}(P),\ |x| = |\tau(x)|$,
2. $\forall t \in \mathcal{CT}(P),\ |t| = |\sigma(t)|$,
3. $\tau \circ \sigma = id$,
4. $\sigma \circ \tau \simeq id$ *(that is* $\forall x \in \mathcal{AC}(P),\ \sigma(\tau(x)) \simeq x$*).*

The above proposition says that the diagram shown in Fig. 6 is commutative except for the loop $\mathcal{AC}(P) \xrightarrow{\tau} \mathcal{CT}(P) \xrightarrow{\sigma} \mathcal{AC}_c(P) \hookrightarrow \mathcal{AC}(P)$.

In this diagram, $\mathcal{AC}_c(P) \subsetneq \mathcal{AC}(P)$ is the set of *causal connectors*, which is the image of $\mathcal{CT}(P)$ by σ. Note that any connector has an equivalent representation in $\mathcal{AC}_c(P)$. Similarly, $\mathcal{AI}_c(P) \subsetneq \mathcal{AI}(P)$ is the set of *causal interactions*, the image of $\mathcal{CT}(P)$ by the semantic function $|\cdot|$. The following proposition provides a characteristic property of the set of causal interactions.

Proposition 6. *The set of the causal interactions is closed under synchronization, that is $x \in \mathcal{AI}_c(P)$ iff $\forall a, b \in \|x\|,\ ab \in \|x\|$.*

As mentioned above, the semantic equivalence $\simeq$ on $\mathcal{AC}(P)$ is not a congruence. Prop. 7 and Cor. 2 below state that the restriction of $\simeq$ to $\mathcal{AC}_c(P)$ is a congruence. By definition of $\mathcal{AC}_c(P)$, each equivalence class $\simeq$ on $\mathcal{AC}(P)$ has a representative in $\mathcal{AC}_c(P)$.

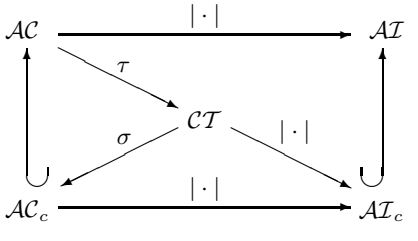

Fig. 6. A diagram relating the algebras

Proposition 7. $\forall t_1, t_2 \in \mathcal{CT}(P),\ t_1 \sim t_2 \Rightarrow \sigma(t_1) \cong \sigma(t_2)$.

Corollary 2. *The $\mathcal{AC}(P)$ equivalence restricted to $\mathcal{AC}_c(P)$ is a congruence, that is, for $x_1, x_2 \in \mathcal{AC}_c(P)$, $x_1 \simeq x_2$ implies $x_1 \cong x_2$.*

4.3 Boolean Representation of Connectors

Definition 8. *A* causal rule *is a $\mathbb{B}[P]$ formula $E \Rightarrow C$, where E (the* effect*) is either a constant, true, or a port variable $p \in P$, and C (the* cause*) is either a constant (true or false) or a disjunction of interactions, i.e. $\bigvee_{i=1}^{n} a_i$ where, for all $i \in [1, n]$, a_i are conjunctions of port variables.*

Causal rules without constants can be rewritten as formulas of the form $\overline{p} \vee \bigvee_{i=1}^{n} a_i$ and, by distributivity of $\wedge$ over $\vee$, are conjunctions of dual Horn clauses, i.e. disjunctions of variables whereof at most one is negative.

In line with Def. 3, an interaction $a \in 2^P$ satisfies the rule $p \Rightarrow \bigvee_{i=1}^{n} a_i$, iff $p \in a$ implies $a_i \subseteq a$, for some $i \in [1, n]$, that is, for a port to belong to an interaction, at least one of the corresponding causes must belong there too.

Example 12. Let $p \in P$, $a \in 2^P$, and $x \in \mathcal{AI}(P)$. Three particular types of causal rules can be set apart:

1. For an interaction to satisfy the rule $\mathit{true} \Rightarrow a$, it is necessary that it contain a.
2. Rules of the form $p \Rightarrow \mathit{true}$ are satisfied by all interactions.
3. An interaction can satisfy the rule $p \Rightarrow \mathit{false}$ only if it does not contain p.

Remark 5. Notice that $a_1 \vee a_1\, a_2 = a_1$, and therefore causal rules can be simplified accordingly:

$$(p \Rightarrow a_1 \vee a_1\, a_2) \ \rightsquigarrow\ (p \Rightarrow a_1)\,. \tag{22}$$

We assume that all the causal rules are simplified by using (22).

Definition 9. *A* system of causal rules *is a set $R = \{p \Rightarrow x_p\}_{p \in P^t}$, where $P^t \stackrel{def}{=} P \cup \{\mathit{true}\}$. An interaction $a \in 2^P$ satisfies the system R (denoted $a \models R$), iff $a \models \bigwedge_{p \in P^t}(p \Rightarrow x_p)$. We denote by $|R|$ the union of the interactions satisfying R:*

$$|R| \stackrel{def}{=} \sum_{a \models R} a\,.$$

A causal tree $t \in \mathcal{CT}(P)$ is equivalent to a system of causal rules R iff $|t| = |R|$.

We associate with $t \in \mathcal{CT}(P)$ the system of causal rules

$$R(t) \stackrel{def}{=} \{p \Rightarrow c_p(t)\}_{p \in P^t}\,, \tag{23}$$

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

$$c_p(0) = \mathit{false}\,, \tag{24}$$

$$c_p(p \to t) = \mathit{true}\,, \tag{25}$$
$$c_p(pa \to t) = a\,, \tag{26}$$
$$c_p(a \to t) = a\, c_p(t)\,, \tag{27}$$
$$c_p(t_1 \oplus t_2) = c_p(t_1) \vee c_p(t_2)\,, \tag{28}$$

Similarly, we define $c_{true}(t)$ by

$$\begin{aligned} c_{true}(0) &= \mathit{false}\,,\\ c_{true}(1 \to t) &= \mathit{true}\,,\\ c_{true}(a \to t) &= a\,,\\ c_{true}(t_1 \oplus t_2) &= c_{true}(t_1) \vee c_{true}(t_2)\,. \end{aligned}$$

Remark 6. It is important to observe that, for any $t \in \mathcal{CT}(P)$, the system of causal rules $R(t)$, defined by (23), contains exactly one causal rule for each $p \in P^t$ (i.e. each $p \in P$ and *true*). For ports that do not participate in t, the rule is $p \Rightarrow \mathit{false}$. For ports that do not have any causality constraints, the rule is $p \Rightarrow \mathit{true}$.

Proposition 8. *For any causal tree $t \in \mathcal{CT}(P)$, $|t| = |R(t)|$.*

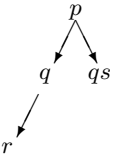

Fig. 7. Graphical representation of the causal tree $t = p \to (q \to r \ \oplus \ qs)$

Example 13. Consider the causal tree $t = p \to (q \to r \ \oplus \ qs)$ shown in Fig. 7. The associated system $R(t)$ of causal rules is

$$\{\mathit{true} \Rightarrow p\,, \quad p \Rightarrow \mathit{true}\,, \quad q \Rightarrow p\,, \quad r \Rightarrow pq\,, \quad s \Rightarrow pq\}\,.$$

Notice that $c_q(t) = p\Big(c_q(q \to r) \vee c_q(qs)\Big) = p \vee ps = p$.

The corresponding boolean formula is then

$$(\mathit{true} \Rightarrow p) \wedge (p \Rightarrow \mathit{true}) \wedge (q \Rightarrow p) \wedge (r \Rightarrow pq) \wedge (s \Rightarrow pq) \ = \ p\,q \vee p\,\overline{r}\,\overline{s}\,.$$

5 Conclusion

The paper provides a causal semantics for the algebra of connectors. This semantics leads to simpler and more intuitive representations which can be used for efficient implementation of operations on connectors in BIP. In contrast to interaction semantics equivalence, the induced equivalence is compatible with

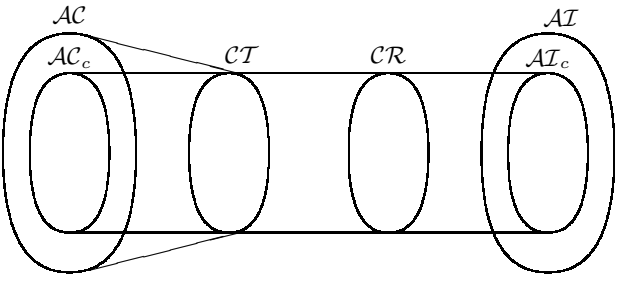

Fig. 8. A graphical representation of the relations between different algebras

the congruence on $\mathcal{AC}(P)$. Causal semantics allows a nice characterization of the set of causal connectors, which is isomorphic to the set of causal trees. The set of causal connectors also corresponds to the set of causal interactions, which are closed under synchronization. The relation between the different algebras is shown in Fig. 8.

The Algebra of Causal Trees, $\mathcal{CT}(P)$, breaks with the reductionist view of interaction semantics as it distinguishes between symmetric and asymmetric interaction. It allows structuring of global interactions as the parallel composition of chains of interactions. This is a very intuitive and alternate approach to interaction modeling especially for broadcast-based languages such as synchronous languages. Causal trees are very close to structures used to represent dependencies between signals in synchronous languages, e.g. [17]. This opens new possibilities for unifying asynchronous and synchronous semantics.

$\mathcal{CT}(P)$ can be extended in a straightforward manner to incorporate guards, necessary for conditional interaction. It is a basis for computing boolean representations for connectors, adequate for their symbolic manipulation and computation of boolean operations. These can be used for efficient implementations of component-based languages such as BIP.

References

1. Balarin, F., Watanabe, Y., Hsieh, H., Lavagno, L., Passerone, C., Sangiovanni-Vincentelli, A.: Metropolis: An integrated electronic system design environment. IEEE Computer 36(4), 45–52 (2003)
2. Balasubramanian, K., Gokhale, A., Karsai, G., Sztipanovits, J., Neema, S.: Developing applications using model-driven design environments. IEEE Computer 39(2), 33–40 (2006)
3. Eker, J., Janneck, J., Lee, E., Liu, J., Liu, X., Ludvig, J., Neuendorffer, S., Sachs, S., Xiong, Y.: Taming heterogeneity: The Ptolemy approach. Proceedings of the IEEE 91(1), 127–144 (2003)
4. Bliudze, S., Sifakis, J.: The algebra of connectors — Structuring interaction in BIP. In: Proceeding of the EMSOFT 2007. Salzburg, Austria, pp. 11–20. ACM SigBED (October 2007)
5. Bernardo, M., Ciancarini, P., Donatiello, L.: On the formalization of architectural types with process algebras. In: SIGSOFT FSE. pp. 140–148 (2000)

6. Spitznagel, B., Garlan, D.: A compositional formalization of connector wrappers. In: ICSE, pp. 374–384. IEEE Computer Society, Los Alamitos (2003)
7. Fiadeiro, J.L.: Categories for Software Engineering. Springer, Heidelberg (2004)
8. Bruni, R., Lanese, I., Montanari, U.: A basic algebra of stateless connectors. Theor. Comput. Sci. 366(1), 98–120 (2006)
9. Arbab, F.: Reo: a channel-based coordination model for component composition. Mathematical Structures in Computer Science 14(3), 329–366 (2004)
10. Arbab, F.: Abstract behavior types: a foundation model for components and their composition. Sci. Comput. Program. 55(1-3), 3–52 (2005)
11. Basu, A., Bozga, M., Sifakis, J.: Modeling heterogeneous real-time components in BIP. In: 4^{th} IEEE International Conference on Software Engineering and Formal Methods (SEFM 2006). pp. 3–12 (invited talk) (September 2006)
12. Sifakis, J.: A framework for component-based construction. In: 3^{rd} IEEE International Conference on Software Engineering and Formal Methods (SEFM05). pp. 293–300 (September 2005) (keynote talk)
13. BIP, `http://www-verimag.imag.fr/~async/BIP/bip.html`
14. Maraninchi, F., Rémond, Y.: Argos: an automaton-based synchronous language. Computer Languages 27, 61–92 (2001)
15. Hoare, C.A.R.: Communicating Sequential Processes, 1985. Prentice Hall International Series in Computer Science. Prentice-Hall, Englewood Cliffs (1985)
16. Milner, R.: Communication and Concurrency. International Series in Computer Science. Prentice-Hall, Englewood Cliffs (1989)
17. Nowak, D.: Synchronous structures. Inf. Comput. 204(8), 1295–1324 (2006)